IEEE Signal Processing MAGAZINE

VOLUME 27 NUMBER 2 MARCH 2010

MULTICORE PLATFORMS

RIDING THE WAVE OF TOMORROW, PART 2

THE CURVELET TRANSFORM

SETI– ARE WE ALONE?

A FLEXIBLE WINDOW FUNCTION

Click here to access **Content Gazette** Supplement

IEEE Signal Processing Society

IEEE

+20 dBm Power Amplifiers with a choice of gain!

GVA AMPLIFIERS

DC to 7 GHz from $1.82 ea. (qty.25)

Mini-Circuits' monolithic, surface-mount GVA amplifiers are extremely broadband, with wide dynamic range and the right gain to fit your application. Based on high-performance InGaP HBT technology, patented GVA amplifiers cover DC* to 7 GHz, with a selection of gain choices 10, 15, 20 or 24dB, (measured at 1 GHz). They provide better than +20 dBm typical output power, with typical IP3 performance as high as +41 dBm at 1 GHz. Supplied in RoHS-compliant, SOT-89 housings, low-cost GVA amplifiers feature excellent input/output return loss and high reverse isolation. With built-in ESD protection, GVA amplifiers are unconditionally stable and designed for a single 5-V supply. For more on broadband GVA amplifiers, visit the Mini-Circuits' web site at www.minicircuits.com.

*US patent 6,943,629 *Low frequency determined by coupling cap.*

Mini-Circuits...Your partners for success since 1969

Mini-Circuits®
ISO 9001 ISO 14001 AS 9100 CERTIFIED
P.O. Box 350166, Brooklyn, New York 11235-0003 (718) 934-4500 Fax (718) 332-4661

YONI2™ Patent Pending ***The Design Engineers Search Engine*** *finds the model you need, Instantly* • For detailed performance specs & shopping online see minicircuits.com

IF/RF MICROWAVE COMPONENTS

458 rev C

[VOLUME 27 NUMBER 2]

[CONTENTS]

[COVER] CHAD BAKER/DIGITAL VISION/ GETTY IMAGES

SCOPE: *IEEE Signal Processing Magazine* publishes tutorial-style articles on signal processing research and applications, as well as columns and forums on issues of interest. Its coverage ranges from fundamental principles to practical implementation, reflecting the multidimensional facets of interests and concerns of the community. Its mission is to bring up-to-date, emerging and active technical developments, issues, and events to the research, educational, and professional communities. It is also the main Society communication platform addressing important issues concerning all members.

IEEE SIGNAL PROCESSING MAGAZINE (ISSN 1053-5888) (ISPREG) is published bimonthly by the Institute of Electrical and Electronics Engineers, Inc., 3 Park Avenue, 17th Floor, New York, NY 10016-5997 USA (+1 212 419 7900). Responsibility for the contents rests upon the authors and not the IEEE, the Society, or its members. Annual member subscriptions included in Society fee. Nonmember subscriptions available upon request. Individual copies: IEEE Members $20.00 (first copy only), nonmembers $175.00 per copy. Copyright and Reprint Permissions: Abstracting is permitted with credit to the source. Libraries are permitted to photocopy beyond the limits of U.S. Copyright Law for private use of patrons: 1) those post-1977 articles that carry a code at the bottom of the first page, provided the per-copy fee indicated in the code is paid through the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923 USA; 2) pre-1978 articles without fee. Instructors are permitted to photocopy isolated articles for noncommercial classroom use without fee. For all other copying, reprint, or republication permission, write to IEEE Service Center, 445 Hoes Lane, Piscataway, NJ 08854 USA. Copyright©2010 by the Institute of Electrical and Electronics Engineers, Inc. All rights reserved. Periodicals postage paid at New York, NY, and at additional mailing offices. Postmaster: Send address changes to IEEE Signal Processing Magazine, IEEE, 445 Hoes Lane, Piscataway, NJ 08854 USA. Canadian GST #125634188
Printed in the U.S.A.

Digital Object Identifier 10.1109/MSP.2009.935379

[from the EDITOR]

Li Deng
Editor-in-Chief
deng@microsoft.com

New Focus, New Challenge

Rapid advancement of our information society necessitates prompt update and expansion of the technical scope and focus of interest of our IEEE Signal Processing Society (SPS). Compared with just some years ago, the current focus of signal processing as an enabling technology has been significantly broadened. Now it encompasses theories, architectures, algorithms, implementations, and applications for the transformation of information contained in many different physical, symbolic, or abstract formats that we broadly designate as "signals." Methodology wise, signal processing uses mathematical, statistical, computational, heuristic, and/or linguistic representations, formalisms, and techniques for sensing, acquisition, extraction, representation, modeling, analysis, synthesis, compression, detection, recovery, decomposition, enhancement, rendering, display, learning, recognition, understanding, securing, authenticating, and communicating of information and signals. Such diverse "processing" tasks are accomplished by either digital or analog devices or algorithms, and in the form of either software, hardware, or firmware.

New elements in the updated focus of interest above are reflected particularly by the expanded members of the "signal," as pursued currently by a wide range of work of the SPS's 11 technical areas. The updated "signal" members cover any abstract, symbolic, or physical manifestation of information with examples that include: audio, music, speech, text, image, graphics, video, multimedia, sensor, communication, geophysical, sonar, radar, biological, chemical, molecular, genomic, medical, data, or sequences of symbols, attributes, or numerical quantities.

The expanded technical scope of our Society presents new challenges for *IEEE Signal Processing Magazine*, especially with respect to its role of educating our readers in new trends and in cross-pollinating technical areas. Signal processing is a vibrant and inherently

(continued on page 148)

Digital Object Identifier 10.1109/MSP.2009.935593

Digital Object Identifier 10.1109/MSP.2010.936095

300 MHz-12 GHz

LTCC MIXERS

from $3.95 ea. (Qty. 1000) IN STOCK

RoHS COMPLIANT

For Commercial, Military, and Industrial Use, Mini-Circuits proudly presents the MCA1 series of Low Temperature Co-fired Ceramic (LTCC) frequency mixers. Highly reliable, only 0.080" in height, and "tough as nails", these patented mixers have all circuitry hermetically imbedded inside the ceramic making them temperature stable and impervious to most environmental conditions. The process also gives you high performance repeatability and very low cost. There's a variety of broadband models and LO power levels to choose from, so you can use these mixers in a multitude of designs and applications. And MCA1 mixers are ideal for the COTS program! Just check all the specs on our web site. Then, choose the model that best fits your needs. Our team is ready to handle your requirements with quick off-the-shelf shipments, custom designs, and fast turn-around/high volume production.

Mini-Circuits...Your partners for success since 1969

For RoHS compliant requirements, ADD + SUFFIX TO BASE MODEL No. *Example: MCA1-85L+*

Model	LO Level (dBm)	Freq. Range (MHz)	Conv. Loss (dB)	LO-RF Isol. (dB)	Price $ ea. (Qty. 10)
MCA1-85L	4	2800-8500	6.0	35	9.45
MCA1-12GL	4	3800-12000	6.5	38	11.95
MCA1-24	7	300-2400	6.1	40	5.95
MCA1-42	7	1000-4200	6.1	35	6.95
MCA1-60	7	1600-6000	6.2	30	7.95
MCA1-85	7	2800-8500	5.6	38	8.95
MCA1-12G	7	3800-12000	6.2	38	10.95
MCA1-24LH	10	300-2400	6.5	40	6.45
MCA1-42LH	10	1000-4200	6.0	38	7.45
MCA1-60LH	10	1700-6000	6.3	30	8.45
MCA1-80LH	10	2800-8000	5.9	35	9.95
MCA1-24MH	13	300-2400	6.1	40	6.95
MCA1-42MH	13	1000-4200	6.2	35	7.95
MCA1-60MH	13	1600-6000	6.4	27	8.95
MCA1-80MH	13	2800-8000	5.7	27	10.95
MCA1-80H	17	2800-8000	6.3	34	11.95

Dimensions: (L) 0.30" x (W) 0.250" x (H) 0.080"
U.S. Patent # 7,027,795

Mini-Circuits®
ISO 9001 ISO 14001 AS9100 CERTIFIED

P.O. Box 350166, Brooklyn, New York 11235-0003 (718) 934-4500 Fax (718) 332-4661

YONI2™ Patent Pending ***The Design Engineers Search Engine*** *finds the model you need, Instantly* • For detailed performance specs & shopping online see minicircuits.com

IF/RF MICROWAVE COMPONENTS

385 rev M

[president's **MESSAGE**]

Mostafa (Mos) Kaveh
2010–2011 SPS President
mos@umn.edu

SPS: Forging Ahead with New Innovations for Its Members

I had the good fortune to join some 1,200 attendees from around the world by participating in the record-setting 2009 IEEE International Conference on Image Processing in Cairo, Egypt. I congratulate General Chair Prof. Magdy Bayoumi and his conference organizing committee for a terrific program and a superb venue. The outstanding technical program was complemented by Egyptian hospitality fit for the pharaohs and the opportunity for attendees to experience transcendent historical and cultural treasures. As the first major event for the Society on the African continent, the conference underscored the IEEE Signal Processing Society's (SPS's) renewed emphasis on regional member development and engagement, reflected most visibly by the planned addition of four elected regional directors to its Board of Governors (BoG).

The Cairo meeting also provided the opportunity for the BoG to review and approve a number of recommendations by the Society's boards and ad hoc volunteer and staff committees. The Society's governance is codified in its Constitution, and the organization operates under its Bylaws, which are aided by a set of policies and procedures. In the January 2010 issue of *IEEE Signal Processing Magazine,* Past President José Moura reported on the work of an ad hoc committee of volunteers and staff chaired by President-Elect Ray Liu that had carried out a significant update of the bylaws and policies and procedures. As José had anticipated, the BoG approved these recommendations, which establish clear collaboration and reporting structures for volunteer leaders and staff. The approved documents expand and make more open and transparent the nominations for Society elections. They also increase the portfolio of the vice president of awards and membership to include the awards board with its Fellow Reference Committee, and the membership board, which now includes the Industrial Relations Committee, the new Membership Services Committee, and the Chapters Standing Committee.

[THE WORK OF THE SOCIETY'S VOLUNTEERS, STAFF, AND ITS OPERATIONS, PROGRAMS, PRODUCTS, AND SERVICES RECEIVED OUTSTANDING MARKS BY THE REVIEW COMMITTEE AT TAB'S NOVEMBER 2009 MEETING.]

The governance of the membership services committee, in turn, includes the aforementioned four regionally elected directors. The updated Bylaws are more concise, clear, and consistent, and the new policies and procedures provide the detail necessary to manage the enterprise, and at the same time, are amenable to updating as necessary for a dynamic field such as signal processing.

The BoG also had the opportunity to honor the Society's 1994–1995 president, Prof. Tariq Durrani, OBE, on the occasion of his retirement from the faculty and long-time academic leadership at Strathclyde University in Glasgow, Scotland. The BoG adopted a resolution recognizing Prof. Durrani's significant contributions to the field and the Society (and, indeed, the IEEE). Another endorsement of honors by the BoG was the election of the Distinguished Lecturers for 2010–2011 (see the "Society News" column on page 6). Congratulations to our colleagues Sheila Hemami, Shrikanth Narayanan, Antonio Ortega, Venu Veeravalli, and Abdelhak Zoubir.

Last summer, the SPS underwent its mandated five-year review by the IEEE Technical Activities Board (TAB). The work of the Society's volunteers, staff, and its operations, programs, products, and services received outstanding marks by the review committee at TAB's November 2009 meeting. It is easy to be complacent in the face of such accomplishments and accolades. But the SPS continues to forge ahead with new innovations and services for its members and the broader community. As Society members, you are now receiving electronic access to SPS publications as part of your membership dues, and digital delivery of SPS publications has been approved for 2010. The inaugural IEEE Thematic Meetings on Signal Processing (THEMES) will take place the week of 15 March 2010 in conjunction with the IEEE International Conference on Acoustics, Speech, and Signal Processing in Dallas, Texas. The Society is also pursuing active involvement in IEEE's Smart Grid Initiative. We are interested in learning of activities in our field that have been applied to this energy-related topic. I look forward to seeing many of you in Dallas and/or hearing your suggestions by e-mail. [SP]

M. Kaveh

Digital Object Identifier 10.1109/MSP.2009.935594

A new era, a new ESC

New tracks. New courses. New focus.

ESC Silicon Valley brings together systems architects, design engineers, suppliers, analysts, and media from across the globe. With cutting edge product demonstrations, visionary keynotes, and hundreds of essential training classes, ESC is the ideal conference for the embedded design community to learn, collaborate, and recognize excellence.

ESC Silicon Valley 2010 features:

- Two "Build Your Own Embedded System" programs featuring the new Intel Multicore Atom and the Freescale Tower development system
- Multicore Expo and Conference co-located with ESC Silicon Valley
- ARM Partner Pavilion
- Display and Connected Devices Emerging Technology Zones
- Keynotes from Dr. Michio Kaku, theoretical physicist and bestselling author and Richard Templeton, CEO & President, Texas Instruments

Start your own personal development at ESC 2010. You can't afford to miss it.

Silicon Valley
McEnery Convention Center, San Jose, April 26 - 29, 2010
Conference: April 26 - 29, 2010
Expo: April 27 – 29, 2010

Register Today.
www. embedded.com/sv

Learn today. Design tomorrow.
ESC

Distinguished Lecturers, Fellows, Awards, and Calls for Nominations

In this column, profiles are given for the IEEE Signal Processing Society's (SPS's) 2010 class of Distinguished Lecturers, the 2010 SPS Fellows are introduced, award recipients are announced, and nominations are sought for directors-at-large and Board of Governors members-at-large.

2010 CLASS OF DISTINGUISHED LECTURERS

The SPS's Distinguished Lecturer Program provides the means for Chapters to have access to well-known educators and authors in the fields of signal processing to lecture at Chapter meetings. Chapters interested in arranging lectures by the Distinguished Lecturers can obtain information from the Society's Web page (http://www.signalprocessingsociety.org/lecturers/distinguished-lecturers/) or by sending an e-mail to sp.info@ieee.org.

Candidates for the Distinguished Lecturer Program are solicited from the Society Technical Committees, Editorial Boards, and Chapters by the Awards Board. The Awards Board vets the nominations, and the Board of Governors approves the final selection. Distinguished Lecturers are appointed for a term of two calendar years.

2010 DISTINGUISHED LECTURERS

SHEILA S. HEMAMI

Sheila S. Hemami received the B.S.E.E. degree from the University of Michigan in 1990 and the M.S.E.E. and Ph.D. degrees from Stanford University in 1992 and 1994, respectively. Her Ph.D. thesis was "Reconstruction of Compressed Images and Video for Lossy Packet Networks," and she was one of the first researchers to work on what we now call error concealment. In 1994, she was with Hewlett-Packard Laboratories, Palo Alto, California. She joined the School of Electrical Engineering at Cornell University in 1995, where she is a professor and directs the Visual Communications Laboratory.

Dr. Hemami's research interests broadly concern communication of visual information, both from a signal processing perspective (signal representation, source coding, and related issues) and from a psychophysical perspective.

Dr. Hemami is an IEEE Fellow and has held various visiting positions, most recently at the University of Nantes, France, and Ecole Polytechnique Federale de Lausanne, Switzerland. She has received numerous college and national teaching awards, including Eta Kappa Nu's C. Holmes MacDonald Award. She is editor-in-chief of *IEEE Transactions on Multimedia* (2008–2010); member-at-large of the SPS Board of Governors (2009–2011), and an SPS Distinguished Lecturer (2010–2011). She chaired the IEEE Image and Multidimensional Signal Processing Technical Committee (2006–2007) and was associate editor for *IEEE Transactions on Signal Processing* (2000–2006).

Dr. Hemami's lecture topics include the following:

- "From Single Media to Multimedia—Perception, Coding, and Quality"
- "A Signal-Processing Approach to Modeling Vision and Applications"
- "Task-Based Imaging—Image Usefulness and Its Relationship to Image Quality."

SHRIKANTH NARAYANAN

Shrikanth Narayanan received his M.S., Engineer, and Ph.D. degrees in electrical engineering, from the University of California, Los Angeles, in 1990, 1992, and 1995, respectively. He is the Andrew J. Viterbi Professor of Engineering at the University of Southern California (USC), where he has been since 2000, and professor in the Signal and Image Processing Institute of USC's Electrical Engineering Department. He also holds joint appointments as professor in computer science, linguistics and psychology. From 1995 to 2000, he was with AT&T, first as a senior member and later as a principal member of its technical staff.

Dr. Narayanan is editor, *Computer, Speech and Language Journal,* and associate editor, *IEEE Transactions on Multimedia* (2009) and the *Journal of Acoustical Society of America.* He was also associate editor, *IEEE Transactions of Speech and Audio Processing* (2000–2004) and *IEEE Signal Processing Magazine* (2005–2008). He is on the Speech Communication and Acoustic Standards Committees of the Acoustical Society of America and the Advisory Council of the International Speech Communication Association. He served on the SPS Speech Processing Technical Committee (2003–2007) and the SPS Multimedia Signal Processing Technical Committee (2005–2008).

He is an IEEE Fellow; fellow, Acoustical Society of America; and member, Tau Beta Pi, Phi Kappa Phi, and Eta Kappa Nu. He is the recipient of an NSF CAREER Award, USC Engineering Junior Research Award, USC Electrical Engineering Northrop-Grumman Research Award, Mellon Award for Mentoring Excellence, Okawa Research Award, IBM Faculty Award, and a faculty fellowship from the USC Center for interdisciplinary research. He received the 2005 SPS Best Paper Award. Papers coauthored with his students have won awards at InterSpeech 2009 Emotion Challenge, IEEE DCOSS 2009, IEEE MMSP 2007, IEEE MMSP 2006, ICASSP 2005, and ICSLP 2002.

His research interests are in signals and systems modeling with an interdisciplinary emphasis on speech, audio, language;

Digital Object Identifier 10.1109/MSP.2009.935586

Discover These New Titles from Wiley and Wiley-IEEE Press

Advanced Signal Integrity for High-Speed Digital Designs

Stephen H. Hall, Howard L. Heck

9780470192351 • Cloth • 680pp • $120.00 • March 2009
Wiley-IEEE Press

This book is designed to provide contemporary readers with an understanding of the emerging high-speed signal integrity issues that are creating roadblocks in digital design. Written by the foremost experts on the subject, it leverages concepts and techniques from non-related fields such as applied physics and microwave engineering and applies them to high-speed digital design—creating the optimal combination between theory and practical applications.

Bayesian Signal Processing: Classical, Modern and Particle Filtering Methods

James V. Candy

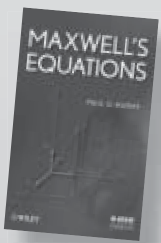
9780470180945 • Cloth • 445pp • $130.00 • April 2009
Wiley-IEEE Press

This text enables readers to fully exploit the many advantages of the "Bayesian approach" to model-based signal processing. It clearly demonstrates the features of this powerful approach compared to the pure statistical methods found in other texts. Readers will discover how easily and effectively the Bayesian approach, coupled with the hierarchy of physics-based models developed throughout, can be applied to signal processing problems that previously seemed unsolvable.

Maxwell's Equations

Paul G. Huray

9780470542767 • Cloth • 312pp • $110.00 • November 2009
Wiley-IEEE Press

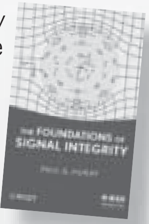


Maxwell's Equations is a practical guide to one of the most remarkable sets of equations ever devised. Professor Paul Huray presents techniques that show the reader how to obtain analytic solutions for Maxwell's equations for ideal materials and boundary conditions. These solutions are then used as a benchmark for solving real-world problems.

The Foundations of Signal Integrity

Paul G. Huray

9780470343609 • Cloth • 339pp • $125.00 • November 2009
Wiley-IEEE Press

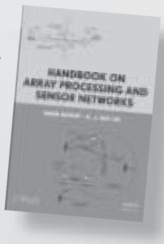


The Foundations of Signal Integrity is the first of its kind—a reference that examines the physical foundation of system integrity based on electromagnetic theory derived from Maxwell's Equations. Drawing upon the cutting-edge research of Professor Paul Huray's team of industrial engineers and graduate students, it develops the physical theory of wave propagation using methods of solid state and high-energy physics, mathematics, chemistry, and electrical engineering before addressing its application to modern high-speed systems.

Handbook on Array Processing and Sensor Networks

Simon Haykin, K. J. Ray Liu

9780470371763 • Cloth • 904pp • $185.00 • January 2010
Wiley-IEEE Press



Featuring contributions by world-renowned experts in their fields, Handbook on Sensor and Array Processing collects tutorial articles on recent advancements and state-of-the-art results by providing a comprehensive overview of sensor and array processing. This first published book on the subject covers fundamental principles as well as applications, making it a must have for researchers, professors, graduate students, and practitioners.

Adaptive Signal Processing: Next Generation Solutions

Tulay Adali, Simon Haykin

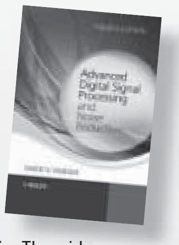
9780470195178 • Cloth • 424pp • $135.00 • March 2010
Wiley-IEEE Press

This book presents the latest research results in adaptive signal processing with an emphasis on important applications and theoretical advancements. Each chapter is self-contained, comprehensive in its coverage, and written by a leader in his or her field of specialty. A uniform style is maintained throughout the book and each chapter concludes with problems for readers to reinforce their understanding of the material presented. The book can be used as a reliable reference for researchers and practitioners or as a textbook for graduate students.

Advanced Digital Signal Processing and Noise Reduction, 4th Edition

Saeed V. Vaseghi

9780470754061 • Cloth • 544pp • $150.00 • March 2009
Wiley



The fourth edition of *Advanced Digital Signal Processing and Noise Reduction* updates and extends the chapters in the previous edition and includes two new chapters on MIMO systems, Correlation and Eigen analysis and independent component analysis. The wide range of topics covered in this book include Wiener filters, echo cancellation, channel equalisation, spectral estimation, detection and removal of impulsive and transient noise, interpolation of missing data segments, speech enhancement and noise/interference in mobile communication environments. This book provides a coherent and structured presentation of the theory and applications of statistical signal processing and noise reduction methods.

Complex Valued Nonlinear Adaptive Filters: Noncircularity, Widely Linear and Neural Models

Danilo Mandic, Vanessa Su Lee Goh

9780470066355 • Cloth • 344pp • $150.00 • June 2009
Wiley

This book was written in response to the growing demand for a text that provides a unified treatment of linear and nonlinear complex valued adaptive filters, and methods for the processing of general complex signals (circular and noncircular). It brings together adaptive filtering algorithms for feedforward (transversal) and feedback architectures and the recent developments in the statistics of complex variable, under the powerful frameworks of CR (Wirtinger) calculus and augmented complex statistics. This offers a number of theoretical performance gains, which is illustrated on both stochastic gradient algorithms, such as the augmented complex least mean square (ACLMS), and those based on Kalman filters. This work is supported by a number of simulations using synthetic and real world data, including the noncircular and intermittent radar and wind signals.

Enter Promotion Code **SNP10** to receive 20% off featured titles at checkout.

Order Information

1 (877) 762-2974 North America
+ 44 (0) 1243 843294 in Rest of World
Log on to www.wiley.com

WILEY IEEE

multimodal and biomedical problems; and applications with direct societal relevance. He has published over 350 papers and has seven granted and ten pending U.S. patents.

Dr. Narayanan's lecture topics include the following:

- Human-Centered Speech and Audio Processing
- Expressive Human Communication: Automatic Recognition and Synthesis
- Speech-to-Speech Translation: Advances, Challenges, and Opportunities
- Speech Production: Data, Models and Technology Applications
- Designing Multimodal Interfaces for Children
- Multimodal Behavioral Signal Processing.

ANTONIO ORTEGA

Antonio Ortega received the telecommunications engineering degree, Universidad Politecnica de Madrid, Spain in 1989 and the Ph.D. in electrical engineering, Columbia University, New York, in 1994. In 1994, Dr. Ortega joined the Electrical Engineering-Systems Department, University of Southern California (USC), where he is currently a professor and associate chair of Electrical Engineering Systems. He has served as director, Signal and Image Processing Institute at USC.

He is an IEEE Fellow and a member of ACM. He has been chair and member, Image, Video, and Multidimensional Signal Processing Technical Committee (2004–2005) and (2006-present), respectively; and member, SPS Board of Governors (2002). He has been Technical Program cochair of ICIP 2008, MMSP 1998, and ICME 2002. He is associate editor, *IEEE Transactions on Image Processing* (2007–2010) and area editor (feature articles), *IEEE Signal Processing Magazine* (2009–present). He was also associate editor, *IEEE Signal Processing Letters* (2001–2002) and *EURASIP Journal on Advances in Signal Processing*. He received the NSF CAREER Award, the IEEE Communications Society Leonard G. Abraham Prize Paper Award (1997), the IEEE Signal Processing Society Magazine Award (1999), and the EURASIP Journal of Advances in Signal Processing Best Paper Award (2006).

His research interests are multimedia compression, communications, and signal analysis. His recent work is focusing on distributed compression, multiview coding, error tolerant compression, wavelet-based signal analysis, and information representation in wireless sensor networks.

Dr. Ortega's lecture topics include the following:

- Practical Applications of Distributed Source Coding
- Multiview Video: Coding Efficiency and Flexible Decoding
- Wavelets on Graphs and Trees: Constructions and Applications
- Seeing the Signals: Applying Signal Processing Tools to Real World Data Analysis Problems.

VENUGOPAL V. VEERAVALLI

Venugopal V. Veeravalli received the Ph.D. degree from the University of Illinois at Urbana-Champaign (1992), the M.S. degree from Carnegie-Mellon University, Pittsburgh, Pennsylvania (1987), and the B.Tech. degree from the Indian Institute of Technology, Bombay (1985), all in electrical engineering. He joined the University of Illinois at Urbana-Champaign in 2000, where he is currently professor, Department of Electrical and Computer Engineering, and research professor, Coordinated Science Laboratory. He is also director, Illinois Center for Wireless Systems (ICWS). He was program director for Communications Research, U.S. National Science Foundation in Arlington, Virginia (2003–2005). He has held academic positions at Harvard University, Rice University, and Cornell University and has been on sabbatical at MIT, IISc Bangalore, and Qualcomm, Inc.

His research interests include distributed sensor systems and networks, wireless communications, detection and estimation theory, and information theory. He is an IEEE Fellow. He was on the Board of Governors of the IEEE Information Theory Society (2004–2007) and associate editor of *IEEE Transactions on Information Theory* (2000–2003) and *IEEE Transactions on Wireless Communications* (1999–2000). He is on the editorial boards of *Communications in Information and Systems* and *Journal of Statistical Theory and Practice*.

He received the IEEE Browder J. Thompson Best Paper Award (1996); the National Science Foundation CAREER Award (1998); the Presidential Early Career Award for Scientists and Engineers (PECASE) (1999); the Michael Tien Excellence in Teaching Award from the College of Engineering, Cornell University (1999); and the Xerox Award for faculty research from the College of Engineering, University of Illinois (2003).

Dr. Veeravalli's lecture topics include the following:

- Quickest Change Detection with Distributed Sensors and Its Applications
- Smart Sleeping Policies for Inference in Sensor Networks
- Distributed Regression and Estimation in Sensor Networks
- Dynamic Spectrum Access with Learning for Cognitive Radio
- Interference Management in Wireless Networks.

ABDELHAK M. ZOUBIR

Abdelhak M. Zoubir received his Dr.-Ing. from Ruhr-Universität Bochum, Germany, in 1992. He was associate professor, Queensland University of Technology, Australia (1992–1998); professor of telecommunications, Curtin University of Technology, Australia (1999); interim head, School of Electrical and Computer Engineering (2001–2003); professor and head of the Signal Processing Group, Technische Universität Darmstadt, Germany (2003).

His research interests are statistical methods for signal processing with emphasis on bootstrap techniques, robust detection and estimation and array processing applied to telecommunications, radar, sonar, car engine monitoring, and biomedicine. He published over 300 journal and conference papers on these areas. He coauthored *Bootstrap Techniques for Signal Processing* (Cambridge University Press, 2004), and he was a guest editor of a special issue on the bootstrap and its applications in *IEEE Signal Processing Magazine* (2007). He coauthored the paper "Detection of Sources Using Bootstrap Techniques," which received the 2003 IEEE SPS Young Author Best Paper Award.

He was deputy technical chair (plenary and special sessions), IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 1994); technical chair, IEEE Workshop on Statistical Signal Processing (SSP 2001); general cochair, IEEE International Symposium on Signal Processing & Information Technology (ISSPIT 2003); and general cochair, IEEE Workshop on Sensor Array and Multichannel Signal Processing (SAM 2008). He was the plenary sessions cochair, ICASSP 2008. He was associate editor, *IEEE Transactions on Signal Processing* (1999–2005), and he is on the editorial boards of *Signal Processing* and the *Journal on Advances in Signal Processing*. He has been an editorial board member, *IEEE Journal on Selected Topics in Signal Processing* (2009); member, Signal Processing Theory and Methods Technical Committee (2002); vice-chair (2008–2009); and chair (2010–2011); member, Sensor Array and Multichannel Signal Processing Technical Committee (2007–present); member, Signal Processing Education Technical Committee (2006–2008); and an elected member, AdCom for the European Association for Signal and Image Processing.

Dr. Zoubir's lecture topics include the following:

- Source Separation for Nonstationary Signals
- The Bootstrap Paradigm in Signal Processing: Estimation, Detection, and Model Selection
- Robust Statistics for Parameter Estimation and Signal Detection
- Signal Processing for Automotive Monitoring.

38 SPS MEMBERS ELEVATED TO FELLOW

Each year, the IEEE Board of Directors confers the grade of Fellow on up to one-tenth percent of the members. To be considered, an individual must have been a Member, normally for five years or more, and a Senior Member at the time for nomination to Fellow. The grade of Fellow recognizes unusual distinction in IEEE's designated fields.

The SPS congratulates these 38 SPS members who were recognized with the grade of Fellow as of 1 January 2010.

Martin J. Bastiaans, Eindhoven, The Netherlands: For contributions to signal processing for optical signals and systems.

Lorenzo Bruzzone, Trento, Italy: For contributions to pattern recognition and image processing for remote sensing.

Ahmet Enis Cetin, Ankara, MN, Turkey: For contributions to signal recovery and image analysis algorithms.

Laurent Cohen, Neuilly-Sur-Seine, France: For contributions to computer vision technology for medical imaging.

David Daniels, Leatherhead, Surrey, United Kingdom: For contributions to ground-penetrating radar.

Michel Defrise, Brussels, Belgium: For contributions to computer tomography.

Ray Dolby, San Francisco, California: For leadership in developing and commercializing practical noise reduction technology.

Hesham M. El-Gamal, Columbus, Ohio: For contributions to multiple-input multiple-output and cooperative communications.

Mário Alexandre Teles Figueiredo, Lisboa, Portugal: For contributions to pattern recognition and computer vision.

Daniel R. Fuhrmann, Houghton, Michigan: For contributions to adaptive radar signal processing.

Marc Hillel Goldburg, Redwood City, California: For leadership in the development and commercialization of spectrally efficient wireless communications systems.

Matti A. Karjalainen, Espoo, Finland: For contributions to perceptual audio signal modeling and processing.

Bart Kosko, Los Angeles, California: For contributions to neural and fuzzy systems.

B.V.K. Vijaya Kumar, Pittsburgh, Pennsylvania: For contributions to biometric recognition methods.

Andrew Francis Laine, New York: For contributions to wavelet applications in digital mammography and ultrasound image analysis.

Seong-Whan Lee, Seoul, Korea: For contributions to pattern recognition for biometrics and document image analysis.

Peyman Milanfar, Santa Cruz, California: For contributions to inverse problems and super-resolution in imaging.

Randolph Lyle Moses, Columbus, Ohio: For contributions to statistical signal processing.

Aria Nosratinia, Richardson, Texas: For contributions to multimedia and wireless communications.

Robert Nowak, Madison, Wisconsin: For contributions to statistical signal and image processing.

Roberto Pieraccini, New York: For contributions to statistical natural language understanding and spoken dialog management and learning.

Douglas A. Reynolds, Lexington, Massachusetts: For contributions to Gaussian-mixture-model techniques for automatic speaker recognition.

Giuseppe Riccardi, Povo-Trento, Italy: For contributions to algorithms for automatic speech recognition and spoken language processing.

Yong Rui, Beijing, China: For contributions to image and video analysis, indexing, and retrieval.

Motoyuki Sato, Sendai, Miyagi-ken, Japan: For contributions to radar remote sensing technologies in environmental and humanitarian applications.

Mihaela Schaar, Los Angeles, California: For contributions to multimedia compression and communications.

Robert Schober, Vancouver, BC, Canada: For contributions to wireless communications.

Dan Schonfeld, Glenview, Illinois: For contributions to image and video analysis.

Andrew C. Singer, Urbana, Illinois: For contributions to signal processing techniques for digital communication.

Malcolm Graham Slaney, Palo Alto, California: For contributions to perceptual signal processing and tomographic imaging.

Frank K. Soong, Beijing, China: For contributions to speech processing.

Milica Stojanovic, Boston, Massachusetts: For contributions to underwater acoustic communications.

Daniel Trudnowski, Butte, Montana: For contributions to algorithms for characterizing power-system small-signal stability properties.

Vishu R. Viswanathan, Plano, Texas: For contributions to speech coding and synthesis and objective speech quality evaluation.

Howard C. Yang, Shanghai, China: For leadership in mixed-signal integrated circuit design and manufacturing.

Feng Zhao, Issaquah, Washington: For contributions to networked embedded computing and sensor networks.

Wenwu Zhu, Beijing, China: For contributions to video communications over the internet and wireless.

Xinhua Zhuang, Columbia, Missouri: For contributions made to digital image processing, image coding, and computer vision.

The following individual was evaluated by the SPS, but is not an SPS member:

Alevoor Ravishankar Rao, Yorktown Heights, New York: For contributions to understanding of image texture and applications to machine vision solutions.

CALL FOR NOMINATIONS: REGIONAL DIRECTORS-AT-LARGE AND BOARD OF GOVERNORS MEMBERS-AT-LARGE

In accordance with the SPS Bylaws, the membership will elect, by direct ballot, three members-at-large to the Board of Governors for three-year terms commencing 1 January 2011 and ending 31 December 2013, as well as one regional director-at-large for the corresponding regions: Regions 1–6 (U.S.), Regions 7 and 9 (Canada and Latin America), Region 8 (Europe/Middle East/Africa) and Region 10 (Asia/Pacific Rim) for two-year terms commencing 1 January 2011 and ending 31 December 2012.

Regional directors-at-large are elected locally by members of the corresponding region. They serve as nonvoting members of the Board of Governors and voting members of the Membership Board. They promote and foster local activities and encourage new chapter development; represent their regions to the core of SPS; offer advice to improve membership relations, provide recruiting and service to their regions; guide and work with their corresponding chapters to serve their members; and assist the vice president-Awards in conducting chapter reviews.

Board of Governors members-at-large are directly elected by the Society's membership to represent the member viewpoint in Board decision making. They typically review, discuss, and act upon a wide range of items affecting the actions, activities, and health of the Society.

José M.F. Moura, SPS past president and chair of the Nominations and Appointments (N&A) Committee, has provided the following formal procedures for the SPS's 2010 regional directors-at-large and BoG members-at-large elections.

- Publication of a call for nominations for positions of BoG members-at-large and regional directors-at-large. Nominees must hold SPS Member grade (IEEE Member grade or higher *and* Member of SPS) to hold elective office (March).
- From the responses received, a list of candidates will be assembled for each election by the past president for presentation to the N&A Committee (April).
- The N&A Committee ballots to create a short list of at least six candidates (by bylaw, at least two candidates must be submitted for each BoG member-at-large position becoming vacant) (April–May). Currently, there is no minimum number of candidates required for the regional directors-at-large race. Nevertheless, the Society has a stated preference for contested elections, so more than one nomination per race is desirable.
- After the N&A ranking ballot, the top candidates who are willing and able to serve for director-at-large and member-at-large are advanced for ballot to the SPS's voting members (July).
- Collection and tabulation of returned ballots will again be handled by the IEEE Technical Activities Society Services Department on behalf of the SPS (July–September).
- The three candidates receiving the highest number of votes who confirm their ability to serve will be declared elected members-at-large to the Board of Governors with three-year terms commencing 1 January 2011 and one candidate from each specified region receiving the highest number of votes who confirm their ability to serve will be declared elected a regional director-at-large with a two-year term commencing 1 January 2011 (September).

Please provide nominations for regional director-at-large and member-at-large to Past President José M.F. Moura via e-mail to t.argiropoulos@ieee.org or via fax to +1 732 235 1627. Please provide the name, address, phone, fax, e-mail, or other contact information of the nominee, along with a brief background on the individual (no more than 100 words, please) and any information about the individual's current activities in the SPS, IEEE, or other professional societies.

2009 IEEE SPS AWARDS PRESENTED IN DALLAS, TEXAS

The IEEE SPS congratulates the SPS members who received the Society's prestigious awards during ICASSP 2010 in Dallas, Texas.

The Society Award honors outstanding technical contributions in a field within the scope of the IEEE SPS and outstanding leadership in that field. The Society Award comprises a plaque, a certificate, and a monetary award of US$2,500. It is the highest-level award bestowed by the IEEE SPS. This year's recipient was Rama Chellappa "for pioneering and fundamental contributions to image and video-based analysis and understanding."

The IEEE Signal Processing Magazine Best Paper Award honors the author(s) of an article of exceptional merit and broad interest on a subject related to the Society's technical scope and appearing in the Society's magazine. The prize comprises US$500 per author (up to a maximum of US$1,500 per award) and a certificate. If there are more than three authors, the maximum prize shall be divided equally among all authors and each shall receive a certificate. The 2009 IEEE Signal Processing Magazine Best Paper Award recipients are Neal Patwari, Joshua N. Ash, Spyros Kyperountas, Alfred O. Hero, III, Randolph L. Moses, and Neiyer S. Correal, for the article "Locating the Nodes: Cooperative Localization in Wireless Sensor Networks," *IEEE Signal Processing Magazine*, vol. 22, no. 4, July 2005.

The IEEE Signal Processing Magazine Best Column Award honors the author(s) of a column of exceptional merit and broad interest on a subject related to the Society's technical scope and appearing in the Society's magazine. The prize shall consist of US$500 per author (up to a maximum of US$1,500 per award) and a certificate. If there are more than three authors, the maximum prize shall be divided equally among all authors and each shall receive a certificate. This year's IEEE Signal Processing Magazine Best Column Award recipient is Richard G. Baraniuk, for the article "Compressive Sensing [Lecture

Notes]," in *IEEE Signal Processing Magazine*, vol. 24, no. 4, July 2007.

Two Technical Achievement Awards were presented this year. Alan S. Willsky received the award "for originality and innovation in stochastic multiresolution modeling in statistical and graphical modeling, and in probabilistic control-driven signal processing." K.J. Ray Liu was recognized "for pioneering and outstanding contributions for the advances of signal processing in multimedia forensics, security, and wireless communications." The Technical Achievement Award honors a person who, over a period of years, has made outstanding technical contributions to the theory and/or practice in technical areas within the scope of the Society, as demonstrated by publications, patents, or recognized impact on this field. The prize is a monetary award of US$1,500, a plaque, and a certificate.

The Meritorious Service Award was presented this year to Arye Nehorai "for exceptional and dedicated service as a leader in a broad range of activities for the Society and profession" and to Isabel Maria Martins Trancoso "for outstanding service and leadership to the worldwide community in the field of speech processing." The award comprises a plaque and a certificate; judging is based on dedication, effort, and contributions to the Society.

The SPS Education Award honors educators who have made pioneering and significant contributions to signal processing education. Judging is based on a career of meritorious achievement in signal processing education as exemplified by writing of scholarly books and texts, course materials, and papers on education; inspirational and innovative teaching; and creativity in the development of new curricula and methodology. The award comprises a plaque, a monetary award of US$1,500, and a certificate. The recipient of the SPS Education Award is Robert M. Gray "for outstanding contributions to education and mentoring in signal processing."

Six Best Paper Awards were awarded, honoring the author(s) of a paper of exceptional merit dealing with a subject related to the Society's technical scope, and appearing in one of the Society's transactions, irrespective of the author's age. The prize is US$500 per author (up to a maximum of US$1,500 per award), and a certificate. Eligibility is based on a five-year window preceding the year of election, and judging is based on general quality, originality, subject matter, and timeliness. Up to six Best Paper Awards may be presented each year. This year, the awardees were:

- Zhou Wang, Alan Conrad Bovik, Hamid Rahim Sheikh, and Eero P. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, Apr. 2004.
- Zhi-Quan (Tom) Luo and Shuzhong Zhang, "Dynamic Spectrum Management: Complexity and Duality," *IEEE Journal of Selected Topics in Signal Processing*, vol. 2, no. 1, Feb. 2008.
- Quentin H. Spencer, A. Lee Swindlehurst, and Martin Haardt, "Zero-Forcing Methods for Downlink Spatial Multiplexing in Multiuser MIMO Channels," *IEEE Transactions on Signal Processing*, vol. 52, no. 2, Feb. 2004.
- Chul Min Lee and Shrikanth S. Narayanan, "Toward Detecting Emotions in Spoken Dialogs," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 2, Mar. 2005.
- Jan Lukáš, Jessica Fridrich, and Miroslav Goljan, "Digital Camera Identification from Sensor Pattern Noise," *IEEE Transactions on Information Forensics and Security*, vol. 1, no. 2, June 2006.
- Genyuan Wang and Moeness G. Amin, "Imaging Through Unknown Walls Using Different Standoff Distances," *IEEE Transactions on Signal Processing*, vol. 54, no. 10, Oct. 2006.

The Young Author Best Paper Award honors the author(s) of an especially meritorious paper dealing with a subject related to the Society's technical scope and appearing in one of the Society's transactions and who, upon date of submission of the paper, is less than 30 years of age. Eligibility is based on a three-year window preceding the year of election, and judging is based on general quality, originality, subject matter, and timeliness.

Two Young Author Best Paper Awards were presented this year:

- Tomoki Toda, for the paper coauthored with Alan W. Black and Keiichi Tokuda, "Voice Conversion Based on Maximum-Likelihood Estimation of Spectral Parameter Trajectory," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 8, November 2007.
- Florian Luisier, for the paper coauthored with Thierry Blu and Michael Unser, "A New SURE Approach to Image Denoising: Interscale Orthonormal Wavelet Thresholding," *IEEE Transactions on Image Processing*, vol. 16, no. 3, March 2007.

SPS MEMBERS RECEIVE IEEE AWARDS

Ronald Schafer has been selected as the IEEE Jack S. Kilby Signal Processing Medal recipient "for leadership and pioneering contributions to the field of digital signal processing." The medal will be presented to Prof. Schafer at the IEEE Honors Ceremonies.

The James L. Flanagan Speech and Audio Processing Technical Field Award will be presented to Prof. Sadaoki Furui "for contributions to and leadership in the field of speech and speaker recognition towards natural communication between humans and machines." This award was founded and is sponsored by the IEEE SPS.

The IEEE Alexander Graham Bell Medal will be presented to Prof. John M. Cioffi "for pioneering discrete multitone modem technology as the foundation of the global DSL industry."

The IEEE Edison Medal will be presented to Prof. Ray Dolby "for leadership and pioneering applications in audio recording and playback equipment for both professional and consumer electronics."

The IEEE Dennis J. Picard Medal for Radar Technologies and Applications will be presented to Prof. Alfonso Farina "for continuous, innovative, theoretical and practical contributions to radar systems and adaptive signal processing techniques."

[SP]

[spotlight REPORT]

Ron Schneiderman

SETI–Are We (Still) Alone?

It's an age-old question: Are we alone in the universe?

The fact is, we still don't know—for sure. But that hasn't stopped us from looking. And we're looking harder than ever.

Movies like *The Day the Earth Stood Still* (1951), *Close Encounters* (1977), and more recently *District 9*, which has been described in reviews as a social satire about a spacecraft that stalls over Johannesburg, have all been box office hits—a pretty strong indication what we all still want to know: Is anyone else out there?

Be assured that a lot of highly qualified people are still trying to find out.

"When we do radio search for extraterrestrial intelligence (SETI), what we're looking for is a narrowband signal with one spot on the radio dial. That's been true ever since Frank Drake—since that first experiment in 1960," says Seth Shostak, senior astronomer for the SETI Institute, based in Mountain View, California. That's the kind of signal where you pump all of your transmitter power into 1 Hz on the dial. "That's what we traditionally look for," says Shostak.

The way to find those, he says, is the incoming cosmic static to your antenna. "It's just Fourier-transformed and you look for a whole bunch of energy, a whole bunch of power. It's the kind of signal that's non-natural. We get a lot of radio static from the cosmos—quasars, pulsars, hot gas, cold gas, even Saturn and Jupiter and the Sun; they all make a lot of radio noise. But it's not narrowband. So, it would easily be distinguished from the natural static."

DRAKE'S EQUATION

Dr. Frank Drake (formerly the board chair of the SETI Institute, and still involved in SETI activities) was a young astronomer working at the National Radio Astronomy Observatory in Green Bank, Virginia, when he estimated the number of technical civilizations that may exist in the galaxy. It quickly became known as the Drake equation, and identifies specific factors thought to play a role in the development of these civilizations although, after years of searching, some SETI scientists aren't as comfortable with Drake's thinking as they used to be. The equation, first presented by Drake in 1961, was originally written as $N = R^* . fp . ne . fl . fi . fc . L$, where

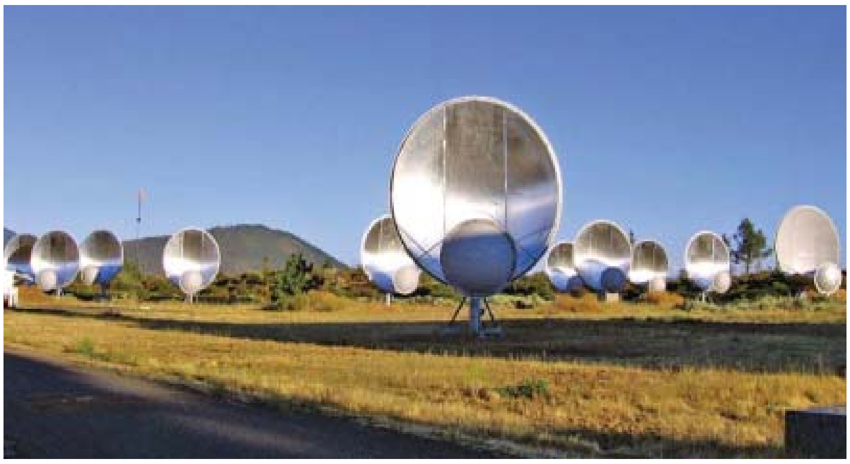
- $N =$ is the number of civilizations in the Milky Way Galaxy whose electromagnetic emissions are detectable.
- R^* is the rate of formation of stars suitable for the development of intelligent life.
- fp is the fraction of those stars with planetary systems.
- ne equals the number of planets, per solar system, with an environment suitable for life.
- fl is the fraction of life-bearing planets on which intelligent life emerges.
- fi equals the fraction of life-bearing planets on which intelligent life emerges.
- fc is the fraction of civilizations that develop a technology that releases detectable signs of their existence into space.
- L is the length of time such civilizations release detectable signals into space.

A NEW TELESCOPE ARRAY

A lot has changed since 1961. In early 2007, the SETI Institute and the University of California (UC)-Berkeley, which works closely with the institute and has its own major SETI program, activated an entirely new system for searching for extraterrestrial intelligence, the Allen telescope array (ATA) (see Figure 1).

The ATA is a network of 42 6-m diameter, mass-produced radio dishes, but the plan is to increase the array to 350 telescopes over the next three years. That is, if the institute and UC-Berkeley can get them funded.

The total cost of the project to date, including research, development, and


[FIG1] The ATA, activated in October 2007, is a joint project of UC-Berkeley and the SETI Institute. Currently made up of 42 radio dishes, the system is expected to grow to 350 dishes to advance the search for extraterrestrial life and radio astronomy research.

Digital Object Identifier 10.1109/MSP.2009.935382

construction of the array and the necessary radio astronomy and SETI signal detectors, was US$50 million. About half of that seed money was donated by Microsoft Cofounder Paul G. Allen. Additional funding has come from the SETI Institute, UC-Berkeley, the National Science Foundation, former Microsoft chief scientist Nathan Myhrvold, Greg Papadopoulos, Xilinx, and other corporations and individual donors.

Completing the array is expected to cost about another US$40 million. Although donations are welcome, the SETI Institute and UC-Berkeley hope to speed up the completion process with a proposal that they submitted to the National Science Foundation in August 2009, which would double the size of the ATA.

While still somewhat limited in sensitivity, a fully developed ATA would significantly expand the radio frequency band for conducting the search, and could detect fainter and more distant signals with more telescopes. At 4.5 octaves of frequency, it already can collect a fairly large amount of data.

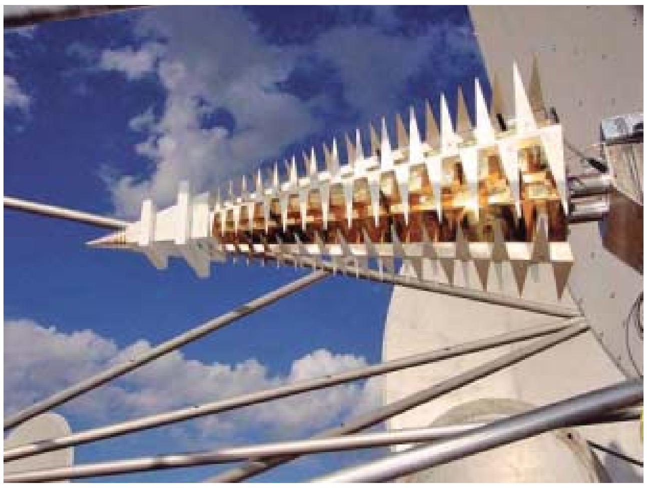
The current array of 42 dishes is spread out over an area of about a half a kilometer. Located near the town of Hat Creek, just north of Lassen Volcanic National Park in northern California, the dishes working together can take in five square degrees of sky at a time—a box as wide as ten full moons. For SETI, in particular, this means that over the next few dozen years, the ATA will get a thousand times more data than has been accumulated in the past 45 years (see Figure 2).

"What you need to do," says Shostak, "is to cross-correlate the antennas. They're putting out streams of bits and you want to correlate them. You want to multiply a string of bits from one antenna against another and that means you can't have them very far apart." (See Figure 3.)

Unlike previous telescopes used in SETI programs, the ATA has several features designed specifically for the SETI mission, including one that filters out noise from man-made interference that in many radio telescopes would render much of the data unusable.

What ATA managers didn't anticipate was interest from the U.S. Air Force Space Command (AFSC), which sees the array as a way to expand its space surveillance capability. "The Air Force is interested in the technology, not the aliens," notes Shostak, and that essentially means adopting the additional sensors provided by the ATA to observe orbiting objects during the day. Because its electro-optical sensors are affected by light pollution during the day, limiting observations that can be conducted at that time, most of the ATA's primary mission is conducted at night. This gives the array its best pointing stability and avoids a decrease in strength of narrowband signals that comes from scattering by the solar wind.

Initial tests run by the Air Force suggests the ATA could track transmitting communication satellites in low- and medium-Earth orbits and, most promising, in geosynchronous orbit, home to the most costly and highly utilized satellites that orbit Earth. If demonstrations are successful, the AFSC says the ATA may prove to be a viable all-weather, day and night contributor to its space surveillance network.


[FIG2] This log periodic dual polarization feed covers 0.5–11.2 GHz. Each of the feeds from the 42 telescopes in the array outputs an equivalent of 200 GB/s.

[FIG3] Seth Shostak, a senior astronomer with the Mountain View, California-based SETI Institute, says he expects the ATA (named after Paul Allen, the cofounder of Microsoft) to produce a thousand times more data over the next few dozen years than has been accumulated in the past 45 years.

WHO ELSE IS LOOKING?

The UC-Berkeley search, called the Search for Extraterrestrial Radio Emissions from Nearby Developed Intelligent Populations (SERENDIP), is run out of the Arecibo Observatory in Puerto Rico, which in the past has been used part-time by the SETI Institute. Other long-time programs include the Planetary Society, an independent, privately funded organization, which operates Project BETA at Harvard University as well as in Argentina. Ohio State University has been conducting a full-time search for years with a large volunteer staff. Other much smaller and private SETI programs are underway in Italy and Australia, although they're not believed to be well funded.

Shostak says one of the most interesting SETI data-processing developments in the last few years is a suggested replacement for the Fourier transform currently used to analyze radio spectra with the Koenen-Loeuve transform, a concept being promoted in Italy that theoretically could simultaneously tap into both narrow and broadband (including spread-spectrum) signals. Italy is an active member of the

International Academy of Astronautics' SETI Permanent Study Group.

Australia's Commonwealth Scientific and Industrial Research Organization commissioned the Parkes Observatory in the 1970s and has become legendary in astronomical circles for its studies of radio galaxies, quasars, pulsars, and the Milky Way's nearest galactic neighbors, the Magellanic Clouds. At one point in the mid-1990s, the SETI Institute had 24 active, funded projects going at Parkes with a staff of 80 people, but using the 64-m Parkes telescope for SETI activity is very low key at this point. An Australian SETI team known as Southern SERENDIP is attempting to "piggyback" its search onto the Parkes Observatory's much larger, more broad-based, astronomy program.

A SETI program is also just getting underway in Korea that piggybacks onto a radio astronomy experiment that's running in that country. While they won't have the luxury of pointing the telescope wherever they want, the Korean SETI scientists have access to the data generated by the telescope to look for SETI-like signals.

TAKING SETI PERSONALLY

Another big SETI program, run by UC-Berkeley, is SETI@Home (see Figure 4).

Think of thousands of personal computers (PCs) all over the world, all working simultaneously to analyze different parts of data collected by the Arecibo telescope, which is the biggest single telescope on Earth.

Essentially, the SETI@Home project borrows computer time from anyone who volunteers for the program when they aren't using their computers for other tasks. It does this with a screen saver that gets data from Berkeley over the Internet, analyzes that data, and then reports the results to Berkeley. The program is entirely voluntary. When you need your PC back, the Berkeley screen saver instantly shuts down and only continues its analysis when you're not using your PC. SETI@Home connects only when transferring data.

All of this is accomplished by breaking up the data into small pieces. Data is recorded on high-density tapes at Arecibo. Since Arecibo does not have a high bandwidth Internet connection, the data is sent to Berkeley very slowly where it is divided into 0.25-MB chunks (called "work units"). These are sent from the SETI@Home server over the Internet for analysis to people around the world.

UC Berkley keeps track of the work-units with a large database. Its computers look for new work units to be processed and these are sent out and marked "in progress" in its database. If you can't complete the work unit, or if your computer crashes and you lose your results, the data isn't lost.

WHERE'S NASA?

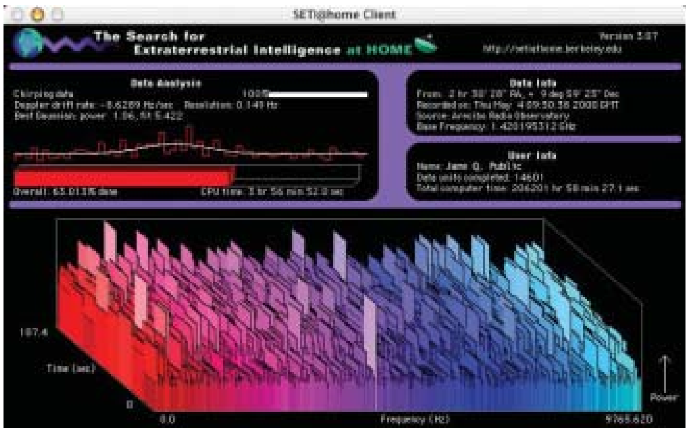
Where does NASA fit into all of this activity? For all of its interest in finding some form of life in space (most recently searching for traces of water on Mars and by firing a rocket into a crater of the moon) NASA has been eliminated from any SETI-specific activities by an act of Congress.

NASA established SETI programs as early as the late 1970s that evolved into a fairly ambitious program known as the High-Resolution Microwave Survey (HRMS). That came to a halt in 1993 when U.S. Senator Richard Bryan of Nevada, citing budget pressures, successfully introduced an amendment to a bill that eliminated all funding for the HRMS program. (HRMS amounted to less than 0.1% of NASA's annual budget.)

NASA does, however, support the SETI Institute's much less publicized research in astrobiology, and its new US$600 million Kepler telescope could become an extremely important factor in the search for extraterrestrial intelligence.

Launched in March 2009, Kepler is designed to survey the Milky Way galaxy to search for Earth-size and smaller planets and determine how many of the billions of stars in our galaxy have such planets. SETI scientists believe that Earth-size planets in our galaxy offer the best chance for finding intelligent life in space.

To conduct its search, the Kepler Mission uses a specially designed 0.95-m diameter telescope that acts as a very sophisticated photometer to measure the size and orbit of every planet that passes in front of the more than 100,000 stars located in what astronomers believe is the most promising region of the Milky Way. Kepler has a very large field of view for an astronomical telescope—105 square degrees, which is comparable to the area of a hand held at arm's length. (The fields of view of most telescopes are less than one square degree.) Kepler needs that large a field to observe the large number of stars. It will look at the same star field for the entire mission and continuously and simultaneously monitor the brightness of the stars for the life of the mission, which is three and a half years.



[FIG4] This screensaver looks for specific pulse signals. It's part of the SETI@HOME project that links thousands of PCs of volunteers all over the world, all working simultaneously to analyze different parts of data when SETI@HOME PCs are idle. Data from the program is collected by the Arecibo Observatory in Puerto Rico.

THE NEXT BIG THING

The next big thing in the SETI community, although not designed specifically for searching for extraterrestrial civilizations, is a US$2 billion project called the square kilometer array (SKA). This is a huge international effort to develop an aperture plane phased array with telescopes that can do many different astronomical observations simultaneously—including SETI—with a million square meters of collecting area.

The SKA is still in the design stage, but testing is already underway of prototype telescopes and new signal processing devices that will be used for cross-correlating the SKA antennas. The array is expected to cover a frequency range of ~0.1–~20 GHz, and involve at least two technologically different antenna concepts. When completed in the second half of the next decade, the SKA will be able to scan and map the sky with a sensitivity ~100 times greater than is currently possible.

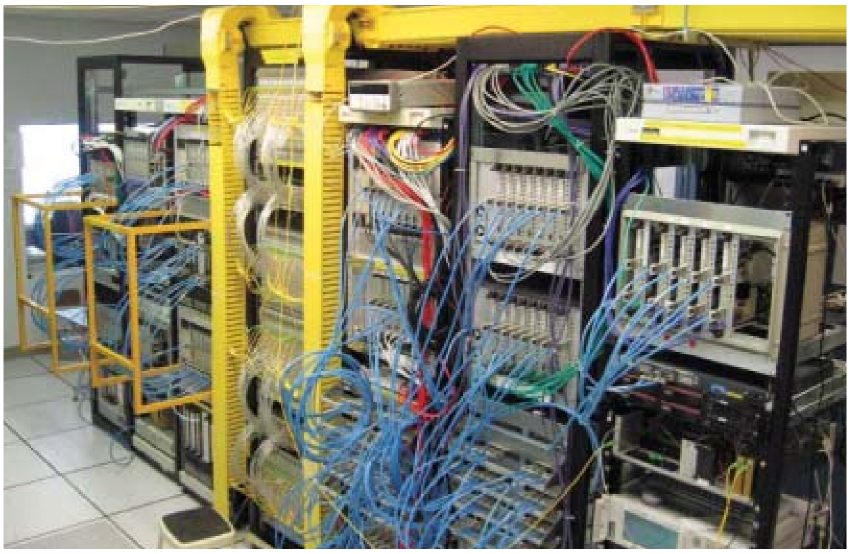
The project is partly funded by the European Community Sixth Framework Programme, with partners from 26 institutes in 13 countries.

Participants in the program haven't yet come up with the US$2 billion needed to complete the SKA, but Dan Werthimer, director of the SETI program at UC Berkeley, says Europeans, Australians, and South Africans have put US$100 million into the program. The U.S. effort, led by the National Science Foundation, has only advanced US$12 million at this point. Werthimer calls this "prototyping money," which is being used primarily for technology development (see Figure 5).

With Arecibo doing a good job covering the northern hemisphere, Werthimer says the SKA will be built either in Australia or South Africa—first, to cover the southern hemisphere, but also to place the new telescope in a very quiet location. "Finding a quiet place is important, but we're learning how to get rid of the radio frequency interference; it's all about signal processing." (See Figure 6.)

Werthimer adds, "We're doing a huge amount of work at Berkeley in collaboration with other groups trying to figure out how to build next-generation telescopes. Not just for SETI, but for other kinds of astronomy, too. And it's dominated by signal processing. We expect to spend about US$300 million on signal processing [development] as part of this program."

[FIG5] Dan Werthimer, director of the SETI program at UC-Berkeley, says his program SERENDIP is conducting a significant amount of work in collaboration with other interested groups in producing next-generation telescopes, with much of the focus on developing new techniques in signal processing. Arecibo is the world's largest radio telescope and is located in Puerto Rico. The group expects to spend about US$300 million on signal-processing development as part of its program.


[FIG6] These ATA racks are part of the scalable DSP instrumentation that are based largely on general-purpose FPGA signal processing boards developed by the Collaboration for Astronomy Signal Processing and Electronics Research (CASPER). (More information can be found at http://casper.berkeley.edu.)

Werthimer also says the SKA team is working with several companies, most

(continued on page 142)

[reader's **CHOICE**]

Top Downloads in IEEE *Xplore*

This issue's "Reader's Choice" contains a list of articles published by the IEEE Signal Processing Society (SPS) that ranked among the top 100 most downloaded IEEE *Xplore* articles from May to October 2009. The highest rank obtained by an article in this time frame is indicated in bold. Your suggestions and comments are welcome and should be sent to Associate Editor Berna Erol at berna_erol@yahoo.com. [SP]

TITLE, AUTHOR, PUBLICATION YEAR IEEE SPS JOURNALS	ABSTRACT	RANK IN IEEE TOP 100 (MAY–OCT 2009) OCT	SEP	AUG	JUL	JUN	MAY	*N* TIMES IN TOP 100 SINCE JAN 2006
COMPLEX WAVELET STRUCTURAL SIMILARITY: A NEW IMAGE SIMILARITY INDEX Sampat, M.P.; Wang, Z.; Gupta, S.; Bovik, A.C.; Markey, M.K. *IEEE Transactions on Image Processing,* vol.18, no. 11, Nov. 2009, pp. 2385–2401	The article introduces a new measure of image similarity that is called the complex wavelet structural similarity (CW-SSIM) index and shows its applicability as a general purpose image similarity index.	**7**						1
A HISTOGRAM MODIFICATION FRAMEWORK AND ITS APPLICATION FOR IMAGE CONTRAST ENHANCEMENT Arici, T.; Dikbas, S.; Altunbasak, Y. *IEEE Transactions on Image Processing,* vol. 18, no. 9, Sep. 2009, pp. 1921–1935	The paper presents a general framework based on histogram equalization for image contrast enhancement, where contrast enhancement is posed as an optimization problem that minimizes a cost function.	15	14	**4**				3
FAST GRADIENT-BASED ALGORITHMS FOR CONSTRAINED TOTAL VARIATION IMAGE DENOISING AND DEBLURRING PROBLEMS Beck, A.; Teboulle, M. *IEEE Transactions on Image Processing,* vol. 18, no. 11, Nov. 2009, pp. 2419–2434	This paper studies gradient-based schemes for image denoising and deblurring problems based on the discretized total variation (TV) minimization model with constraints.	**16**						1
FROM LAGRANGE TO SHANNON... AND BACK: ANOTHER LOOK AT SAMPLING Prandoni, P.; Vetterli, M. *IEEE Signal Processing Magazine,* vol. 26, no. 5, Sep. 2009, pp. 138–144	This article examines the interplay between analog and digital signals, casting discrete-time sequences in the lead role, with continuous-time signals entering the scene as a derived version of their gap-toothed archetypes.	25	**2**					2
AN INTRODUCTION TO COMPRESSIVE SAMPLING Candes, E.J.; Wakin, M.B. *IEEE Signal Processing Magazine,* vol. 25, no. 2, Mar. 2008, pp. 21–30	This article surveys the theory of compressive sampling, also known as compressed sensing or CS, a novel sensing/sampling paradigm that goes against the common wisdom in data acquisition.	34	**16**	61	35	56	65	18

Digital Object Identifier 10.1109/MSP.2009.935420

TITLE, AUTHOR, PUBLICATION YEAR IEEE SPS JOURNALS	ABSTRACT	RANK IN IEEE TOP 100 (MAY–OCT 2009)						*N* TIMES IN TOP 100 SINCE JAN 2006
		OCT	SEP	AUG	JUL	JUN	MAY	
ANALYSIS OF THE SECURITY OF PERCEPTUAL IMAGE HASHING BASED ON NON-NEGATIVE MATRIX FACTORIZATION Khelifi, F.; Jiang, J. *IEEE Signal Processing Letters*, vol. 17, no. 1, Jan. 2010 (first published Sep. 2009), pp. 43–46	This article analyzes the security of a perceptual image hashing technique based on non-negative matrix factorization which was recently proposed and reported in the literature.	**48**						1
FACE RECOGNITION UNDER VARYING ILLUMINATION USING GRADIENTFACES Zhang, T.; Tang, Y.Y.; Fang, B.; Shang, Z.; Liu, X. *IEEE Transactions on Image Processing*, vol. 18, no. 11, Nov. 2009, pp. 2599–2606	This paper proposes a novel method to extract illumination insensitive features for face recognition under varying lighting called the gradient faces.	**56**						1
A THEORY OF PHASE SINGULARITIES FOR IMAGE REPRESENTATION AND ITS APPLICATIONS TO OBJECT TRACKING AND IMAGE MATCHING Qiao, Y.; Wang, W.; Minematsu, N.; Liu, J; Takeda, M.; Tang, X. *IEEE Transactions on Image Processing*, vol. 18, no. 10, Oct. 2009, pp. 2153–2166	This paper studies phase singularities (PSs) for image representation and shows that PSs calculated with Laguerre-Gauss filters contain important information and provide a useful tool for image analysis.	58	**27**					2
FACE RECOGNITION USING DUAL-TREE COMPLEX WAVELET FEATURES Liu, C.C.; Dai, D.Q. *IEEE Transactions on Image Processing*, vol. 18, no. 11, Nov. 2009, pp. 2593–2599	This paper proposes a novel facial representation based on the dual-tree complex wavelet transform for face recognition, which is effective in representing the geometrical structures in facial image with low redundancy.	**66**						1
TRAINING AN ACTIVE RANDOM FIELD FOR REAL-TIME IMAGE DENOISING Barbu, A. *IEEE Transactions on Image Processing,* vol. 18, no. 11, Nov. 2009, pp. 2451–2462	This paper proposes to train Markov random fields (MRF)/conditional random fields (CRF) model together with a fast and suboptimal inference algorithm, which results in considerable gains in speed and accuracy.	**79**						1
AUTOMATIC IMAGE SEGMENTATION BY DYNAMIC REGION GROWTH AND MULTIRESOLUTION MERGING Ugarriza, L. G.; Saber, E.; Vantaram, S.R.; Amuso, V.; Shaw, M.; Bhaskar, R. *IEEE Transactions on Image Processing,* vol. 18, no. 10, Oct. 2009, pp. 2275–2288	This paper presents a new unsupervised color image segmentation algorithm, which exploits the information obtained from detecting edges in color images in the CIE L*a*b* color space.	84	**83**					2
COLLABORATIVE CYCLOSTATIONARY SPECTRUM SENSING FOR COGNITIVE RADIO SYSTEMS Lunden, J.; Koivunen, V.; Huttunen, A.; Poor, H.V. *IEEE Transactions on* Signal Processing, vol. 57, no. 11, Nov. 2009, pp. 4182–4195	This paper proposes an energy efficient collaborative cyclostationary spectrum sensing approach for cognitive radio systems.	**95**						1
APPLICATION OF SIGNAL PROCESSING TO THE ANALYSIS OF FINANCIAL DATA Drakakis, K. *IEEE Signal Processing Magazine,* vol. 26, no. 5, Sep. 2009, pp. 158–160	This article highlights some of the techniques used to represent and predict the main features of price evolution and to classify stock so as to design diversified investment portfolios.		**11**					1
GAME THEORY AND THE FLAT-FADING GAUSSIAN INTERFERENCE CHANNEL Larsson, E.; Jorswieck, E.; Lindblom, J.; Mochaourab, R. *IEEE Signal Processing Magazine,* vol. 26, no. 5, Sep. 2009, pp. 18–27	This article describes basic concepts from noncooperative and cooperative game theory and illustrates them by three examples using the interference channel model.		**41**					1

TITLE, AUTHOR, PUBLICATION YEAR IEEE SPS JOURNALS	ABSTRACT	RANK IN IEEE TOP 100 (MAY–OCT 2009)						*N* TIMES IN TOP 100 SINCE JAN 2006
		OCT	SEP	AUG	JUL	JUN	MAY	
CONTENT BASED IMAGE RETRIEVAL USING UNCLEAN POSITIVE EXAMPLES Zhang, J.; Ye, L. *IEEE Transactions on Image Processing*, vol. 18, no. 10, Oct. 2009, pp. 2370–2375	This paper presents a scheme for training CBIR systems with unclean positive samples. To handle the noisy positive samples, the paper proposes a new two-step strategy by incorporating the methods of data cleaning and noise tolerant classifier.		**63**					1
FLEXIBLE DESIGN OF COGNITIVE RADIO WIRELESS SYSTEMS Scutari, G.; Palomar, D.; Pang, J.-S.; Facchinei, F. *IEEE Signal Processing Magazine*, vol. 26, no. 5, Sep. 2009, pp. 107–123	This article presents that many unsolved resource allocation problems in the field of cognitive radio (CR) networks fit naturally either in the game theoretical paradigm or in the more general theory of VI.		**65**					1
GAME THEORY AND THE FREQUENCY SELECTIVE INTERFERENCE CHANNEL Leshem, A.; Zehavi, E. *IEEE Signal Processing Magazine,* vol. 26, no. 5, Sep. 2009, pp. 28–40	The article discusses the importance of the frequency selective interference channel and shows that it has many intriguing aspects from a game theoretic point of view.		**66**					1
A TUTORIAL ON PARTICLE FILTERS FOR ONLINE NONLINEAR/NON-GAUSSIAN BAYESIAN TRACKING Arulampalam, M.S.; Maskell, S.; Gordon, N.; Clapp, T. *IEEE Transactions on Signal Processing*, vol. 50, no. 2, Feb. 2002, pp. 174–188	This paper reviews both optimal and suboptimal Bayesian algorithms for nonlinear/non-Gaussian tracking problems, with a focus on particle filters.		67	60	44	**25**	29	38
COMPRESSIVE-PROJECTION PRINCIPAL COMPONENT ANALYSIS Fowler, J.E. *IEEE Transactions on Image Processing*, vol. 18, no. 10, Oct. 2009, pp. 2230–2242	This paper presents a process that effectively shifts the computational burden of PCA from the resource-constrained encoder to a presumably more capable base-station decoder.		**70**					1
NOISE-DRIVEN ANISOTROPIC DIFFUSION FILTERING OF MRI Krissian, K.; Aja-Fernandez, S. *IEEE Transactions on Image Processing*, vol. 18, no. 10, Oct. 2009, pp. 2265-2274	This paper presents a new filtering method to remove Rician noise from magnetic resonance images.		**94**					1
RANDOM DISCRETE FRACTIONAL FOURIER TRANSFORM Hsue, W.L.; Pei, S.C. *IEEE Signal Processing Letters,* vol. 16, no. 12, Dec. 2009, pp. 1015–1018	This article proposes a random discrete fractional Fourier transform (RDFRFT) kernel matrix with random DFT eigenvectors and eigenvalues.		**98**					1
SUPER-RESOLUTION WITHOUT EXPLICIT SUBPIXEL MOTION ESTIMATION Takeda, H.; Milanfar, P.; Protter, M.; Elad, M. *IEEE Transactions on Image Processing,* vol. 18, no. 9, Sep. 2009, pp. 1958–1975	This paper introduces a novel framework for adaptive enhancement and spatiotemporal upscaling of videos containing complex activities without explicit need for accurate motion estimation.			**36**				1
A TOTAL VARIATION-BASED ALGORITHM FOR PIXEL-LEVEL IMAGE FUSION Kumar, M.; Dass, S. *IEEE Transactions on Image Processing,* vol. 18, no. 9, Sep. 2009, pp. 2137–2143	This paper proposes a total variation (TV) based approach for pixel-level fusion to fuse images acquired using multiple sensors.			**48**				1
N-SIFT: N-DIMENSIONAL SCALE INVARIANT FEATURE TRANSFORM Cheung, W.; Hamarneh, G. *IEEE Transactions on Image Processing,* vol. 18, no. 9, Sep. 2009, pp. 2012–2021	This paper proposes the n-dimensional scale invariant feature transform (n-SIFT) method for extracting and matching salient features from scalar images of arbitrary dimensionality.			**55**				1

TITLE, AUTHOR, PUBLICATION YEAR IEEE SPS JOURNALS	ABSTRACT	RANK IN IEEE TOP 100 (MAY–OCT 2009)						*N* TIMES IN TOP 100 SINCE JAN 2006
		OCT	SEP	AUG	JUL	JUN	MAY	
IMAGE QUALITY ASSESSMENT BASED ON MULTISCALE GEOMETRIC ANALYSIS Gao, X.; Lu, W.; Tao, D.; Li, X. *IEEE Transactions on Image Processing*, vol. 18, no. 7, July 2009, pp. 1409–1423	This paper proposes a novel framework for image quality assessment (IQA) to mimic the human visual system (HVS) by incorporating the merits from multiscale geometric analysis (MGA), contrast sensitivity function (CSF), and the Weber's law of just noticeable difference (JND).			57	**30**	75		3
BRIDGING THE GAP BETWEEN SIGNAL AND POWER Bollen, M.H.J.; Gu, I.Y.H.; Santoso, S.; McGranaghan, M.F.; Crossley, P.A.; Ribeiro, M.V.; Ribeiro, P.F. *IEEE Signal Processing Magazine*, vol. 26, no. 4, Jul. 2009, pp. 12–31	This article focuses on problems and issues related to PQ and power system diagnostics, in particular those where signal processing techniques are extremely important.			**64**				1
IMAGE SEGMENTATION USING INFORMATION BOTTLENECK METHOD Bardera, A.; Rigau, J.; Boada, I.; Feixas, M.; Sbert, M. *IEEE Transactions on Image Processing*, vol. 18, no. 7, July 2009, pp. 1601–1612	This paper presents new image segmentation algorithms based on a hard version of the information bottleneck method.			69	**54**			2
AN ADAPTABLE K-NEAREST NEIGHBORS ALGORITHM FOR MMSE IMAGE INTERPOLATION Ni, K. S.; Nguyen, T. Q. *IEEE Transactions on Image Processing,* vol. 18, no. 9, Sep. 2009, pp. 1976–1987	The paper proposes an image interpolation algorithm that is nonparametric and learning-based, primarily using an adaptive k-nearest neighbor algorithm with global considerations through Markov random fields.			**80**				1
SUPER-RESOLUTION IMAGE RECONSTRUCTION: A TECHNICAL OVERVIEW Park, S.C.; Park, M.K.; Kang, M.G. *IEEE Signal Processing Magazine,* vol. 20, no. 3, May 2003, pp. 21–36	This article presents the technical review of various existing super resolution (SR) methodologies and models the low-resolution (LR) image acquisition process.			86	95	98	**63**	11
SIGNAL PROCESSING: A VIEW OF THE FUTURE, PART 2 Treichler, J. *IEEE Signal Processing Magazine,* vol. 26, no. 3, May 2009, pp. 83–86	This article attempts to produce a behavioral model for the field of signal processing and then use that model to predict the field's future.			91	55	82	**36**	5
IMAGE DENOISING USING MIXTURES OF PROJECTED GAUSSIAN SCALE MIXTURES Goossens, B.; Pizurica, A.; Philips, W. *IEEE Transactions on Image Processing*, vol. 18, no. 8, Aug. 2009, pp. 1689–1702	This paper proposes a new statistical model for image restoration in which neighborhoods of wavelet subbands are modeled by a discrete mixture of linear projected Gaussian scale mixtures (MPGSM).			94	**33**			2
BEYOND BANDLIMITED SAMPLING Eldar, Y.; Michaeli, T. *IEEE Signal Processing Magazine,* vol. 26, no. 3, May 2009, pp. 48–68	This survey article presents several extensions of the Shannon theorem, which treat a wide class of input signals as well as nonideal sampling and nonlinear distortions.				53	81	**28**	4
SUPER RESOLUTION WITH PROBABILISTIC MOTION ESTIMATION Protter, M.; Elad, M. *IEEE Transactions on Image Processing,* vol. 18, no. 8, Aug. 2009, pp. 1899–1904	This paper presents a new framework that leads to the same algorithm as the authors' prior work but with an approach that is much simpler and more intuitive.				**67**			1
MIMO DETECTION METHODS: HOW THEY WORK Larsson, E.G. *IEEE Signal Processing Magazine,* vol. 26, no. 3, May 2009, pp. 91–95	This tutorial article provides an overview of different MIMO detection approaches, in the communications receiver context.				80	65	**56**	4

(continued on page 155)

Yen-Kuang Chen,
Chaitali Chakrabarti,
Shuvra Bhattacharyya, and
Bruno Bougard

Signal Processing on Platforms with Multiple Cores: Part 2–Applications and Design

Platforms with multiple cores are now prevalent everywhere from desktops and graphics processors to laptops and embedded systems. By adding more parallel computational resources while managing power consumption, multicore platforms offer better programmability, performance, and power efficiency. Signal processing systems of tomorrow will be and must be implemented on platforms with multiple cores. Writing efficient parallel applications that utilize the computing capability of many processing cores require some effort. Signal processing algorithm designers must understand the nuances of a multicore computing engine; only then can the tremendous computing power that such platforms provide be harnessed efficiently. To give a thorough perspective of the area, we have organized two special issues on this topic.

The first special issue, published in November 2009, provided an overview of multiple core systems along with some key methodologies. The articles provided coverage of key trends and emerging directions in architectures, design methods, software tools, and application development for the design and implementation of multicore signal-processing systems. There were three articles surveying the multicore architectures, from general-purpose processors and digital signal processors (DSPs) to multiprocessor system-on-chip. These were followed by four articles discussing software development methodology, including compilation tools that discover parallelism automatically, parallel programming languages where programmers can annotate the parallelism, and approaches that require programmer to explicitly express the parallelism. Part 1 of the two-part special issue ended with four design-example articles spanning fast Fourier transform (FFT), video processing, video coding, and speech recognition.

This special issue aims at 1) describing novel applications that can be enabled by platforms with multiple cores and 2) providing more extensive design examples to demonstrate useful techniques for developing efficient signal processing applications on platforms with multiple cores. Because multicore processors provide better programmability, performance, and power efficiency, many computationally demanding applications are now feasible at a much lower cost. To illustrate this point, we look at applications that can be enabled by multicore platforms. Furthermore, to provide more comprehensive examples on how applications can be realized, we review implementation details on a larger set of applications.

[SIGNAL PROCESSING SYSTEMS OF TOMORROW WILL BE AND MUST BE IMPLEMENTED ON PLATFORMS WITH MULTIPLE CORES.]

There are a total of ten articles in this special issue. They can be broadly classified into novel applications that can be enabled by platforms with multiple cores (articles one–four), and design examples illustrating useful techniques to enable efficient implementations on these platforms (articles five–ten). Some of the articles represent both categories because it is often difficult to demonstrate novel applications without providing some relevant implementation details. In such cases, we have categorized the articles according to whether they emphasize the enabled applications or focus more on implementation techniques.

The first article by Palkovic et al. shows that software-defined radio (SDR), an attractive solution for handling diverse and evolving wireless standards, makes effective use of multicore platforms. This article provides an overview of multicore architectures for SDR platforms along with the specifics of their mapping flows. The authors also show how this technology can be harnessed to handle more complex workloads of emerging wireless communication standards.

The second article by Rzeszutek et al. shows that object segmentation with interactive frame rates can be enabled by the computational capability of multicore platforms. Segmenting the boundaries of objects in a video sequence is a complex and time-consuming task. Furthermore, many objects of interest, such as people and animals, have highly complex and irregular shapes. This article presents a rotoscoping method that takes advantage of the ubiquitous multicore processors such as graphics processing units (GPUs) to assist artists.

The third article by Samsi et al. explores the acceleration of computationally intensive applications by applying commonly used tools to exploit multicore platforms. The authors show that with small changes to sequential MATLAB code, it is possible to effectively utilize today's multicore systems and reduce simulation time. Two signal processing kernels (FFT and convolution) and two full applications (synthetic aperture radar imaging and superconducting quantum interference devices) are used to illustrate the use of parallel MATLAB.

Digital Object Identifier 10.1109/MSP.2009.935527

The last article of the novel application category shows that medical imaging, which is highly computation intensive, can now be implemented on an easily available multicore platform. The focus is on medical image registration, which is an integral part of image-guided intervention and therapy systems. Shams et al. provide an extensive survey of image registration algorithms and their performance on various multiprocessor platforms, including cluster computers, GPUs, and CELL.

The first design example article by Plishker et al. is a specific design example of medical imaging on GPUs. It demonstrates that we must exploit hierarchical parallelism properly to get the best utilization of the platforms. Although multicore platforms offer significant performance potential, there are challenges in finding and exploiting the parallelism. The authors depict a synergistic approach that first organizes application parallelism into a domain-specific taxonomy and then structures the algorithm to target a set of multicore platforms.

The article by di Bisceglie et al. demonstrates the need to pay attention to the usage of resources (e.g., device memories, kernel functions, and synchronizations) and choose appropriate data transfer granularity to use the GPU resources efficiently. The authors show a design example of synthetic aperture radar on the GPU. While signal processing algorithms for synthetic aperture radar are becoming mature, it is a challenge to produce an accurate image in real time without a mainframe computer. This article provides an example on how to implement an important subset of focusing algorithms on general-purpose GPUs.

The next article by Cheung et al. advocates the need to structure the algorithm to expose as much data parallelism as possible to utilize the computational capability. The article illustrates this for video codecs running on GPUs. While the GPU offers high peak performance, it is challenging to achieve it because of the dependencies imposed by the codec. The authors demonstrate that with the proper algorithm redesign, the performance of the fast motion estimation algorithm, for example, on GPU can be improved by three to four times.

The article by Daudet also illustrates that we must reformulate the algorithm to expose the parallelism. He demonstrates this for the matching pursuit algorithm that is often used to solve very large sparse approximation problems. While matching pursuit is considered intrinsically sequential, a small modification of the algorithm can break the data dependencies and enable efficient implementation on multicore processors.

[THE GOAL OF THE TWO-PART SPECIAL ISSUE WAS TO CAPTURE STATE OF THE ART IN SIGNAL PROCESSING ON PLATFORMS WITH MULTIPLE CORES.]

The article by Kim et al. shows that we must consider the underlying architectural features in parallelizing workloads. Image processing applications have an abundance of parallelism and benefit significantly from multicore systems. However, simply exploiting parallelism is not enough to achieve the best performance. Optimization must take into account underlying architecture characteristics such as wide vector and limited bandwidth. The article presents techniques that can be used to optimize performance for multicore x86 systems on three key image processing kernels, FFT, convolution, and histogram.

The last article by van Nieuwpoort and Romein demonstrates that we must have different implementation and optimization strategies for multicore architectures with different performance characteristics. The authors choose correlation computation in radio astronomy signals to compare the performance, optimization, and programmability of multiple multicore platforms. The article shows how to predict what performance can be achieved on many-core platforms and where bottlenecks can be expected. The authors also provide guidelines for optimizing on the different platforms.

In short, this special issue showed that many novel applications can be enabled by platforms with multiple cores. These include image processing, video processing, rotoscoping, medical imaging, SDR, synthetic aperture radar, and radio astronomy signal processing. This special issue also demonstrated useful techniques to develop efficient signal processing applications on platforms with multiple cores. The era of signal processing on systems with multiple/many cores has just started. We hope that you enjoy the articles in this special issue of *IEEE Signal Processing Magazine* as much as those in the November 2009 issue and that you find the contents informative and useful.

We sincerely thank the authors for their valuable contributions, as well as the anonymous reviewers for their help in ensuring the quality of this special issue. The goal of the two-part special issue was to capture state of the art in signal processing on platforms with multiple cores. Many papers were not selected because they did not fit into the scope of the special issue, even though some of them would have been excellent matches for other publications. Furthermore, the high quality requirements of *IEEE Signal Processing Magazine* forced us to reject papers even after multiple rounds of reviews. In fact, of the 144 white papers that were submitted, only 21 articles were published in these two special issues. We would like to thank everyone who submitted articles for their effort and express our regret that due to limited space and the need for balanced coverage, not all high-quality submissions could be included.

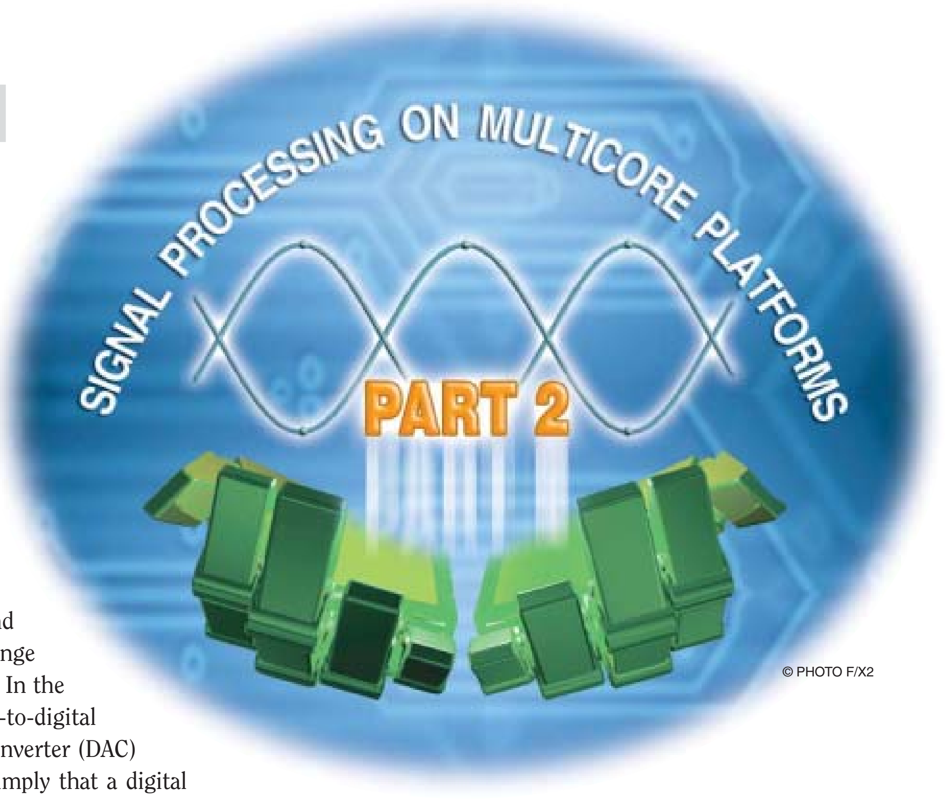
Finally, we would like to thank Li Deng, Geri Krolin-Taylor, and Dan Schonfeld for their help and support in organizing the two-part special issue.

[SP]

Martin Palkovic, Praveen Raghavan, Min Li,
Antoine Dejonghe, Liesbet Van der Perre, and Francky Catthoor

Future Software-Defined Radio Platforms and Mapping Flows

An overview of their multicore architectures



© PHOTO F/X2

A software-defined radio (SDR) system is a radio communication system in which physical layer components are implemented on a programmable or reconfigurable platform. The modulation and demodulation is performed in software and thus the radio is able to support a broad range of frequencies and functions concurrently. In the ideal SDR transceiver scheme, an analog-to-digital converter (ADC) and a digital-to-analog converter (DAC) are attached to the antenna. This would imply that a digital signal processor (DSP) is connected to the ADC and the DAC, directly performing signal processing for the streams of data from/to antenna [1]. Today, the ideal SDR transceiver scheme is still not feasible and thus some processing has to happen in the reconfigurable analog front end [2].

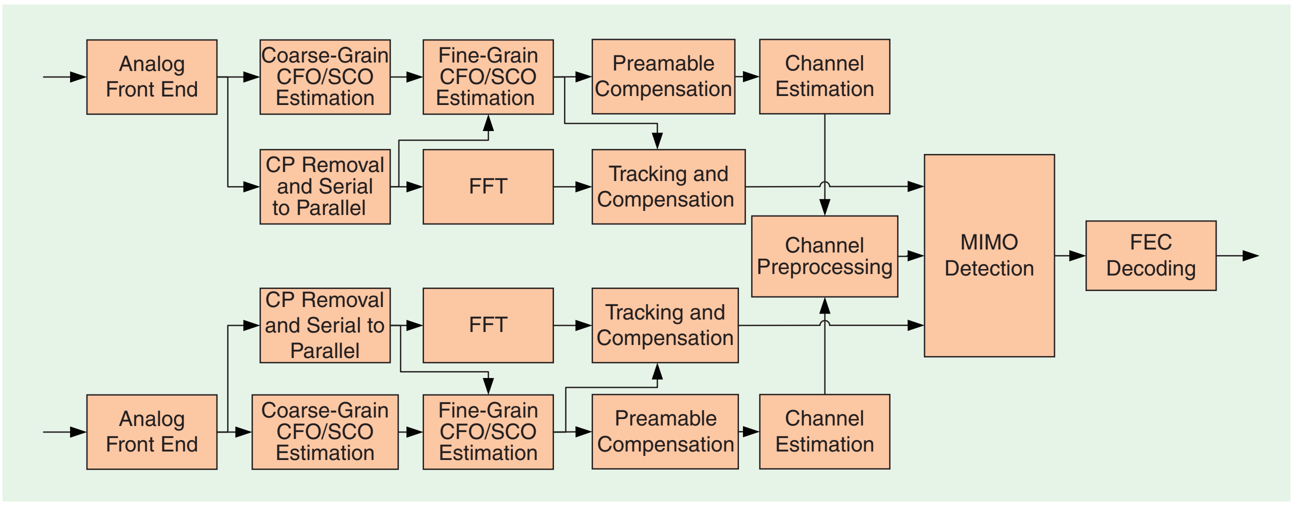
INTRODUCTION

In the past, SDR was mainly attractive for the military and wireless infrastructure segments. Recently, the SDR paradigm has also entered into the consumer electronics segment. This is driven by three main factors. First, the soaring chip development cost and respin rate in deep submicro era has driven chip vendors to share the development costs across product lines. Second, extremely diversified market demand for different wireless standards has triggered the need for an ideal mobile terminal to support these standards (from cellular to broadcasting) within a tight cost budget. Third, the fast evolution of wireless standards has caused a shorter time-to-market, which makes a programmable SDR solution attractive.

SDR: THE NEED FOR MULTICORES AND WHAT IT BRINGS

The physical layer of a typical radio transceiver consists of an inner modem and an outer modem. The inner modem contains

Digital Object Identifier 10.1109/MSP.2009.935386

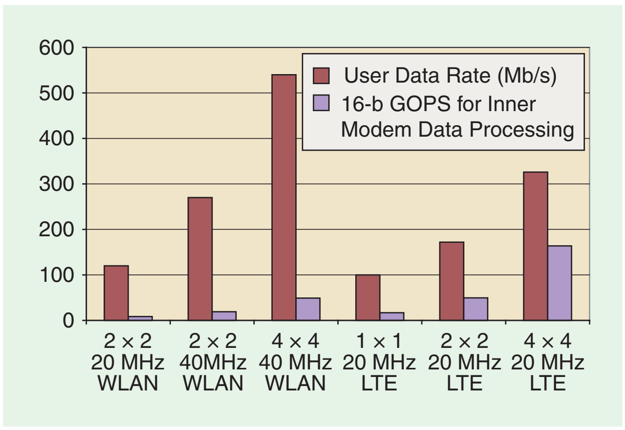

Analog Front End
Coarse-Grain CFO/SCO Estimation
Fine-Grain CFO/SCO Estimation
Preamble Compensation
Channel Estimation
CP Removal and Serial to Parallel
FFT
Tracking and Compensation
Channel Preprocessing
MIMO Detection
FEC Decoding
CP Removal and Serial to Parallel
FFT
Tracking and Compensation
Analog Front End
Coarse-Grain CFO/SCO Estimation
Fine-Grain CFO/SCO Estimation
Preamble Compensation
Channel Estimation

[FIG1] **Block diagram of typical WLAN RX functionality.**

transmission and environment parameters estimation (including various acquisition and tracking tasks) and data detection (e.g., demapping and multiantenna detection), while the outer modem mostly performs the decoding/coding of the received/transmitted stream of data (see Figure 1). On top of these two compute-intensive functionalities, multiple access control (MAC) functionalities is also needed to ensure appropriate timing of the operations and appropriate acknowledgment schemes. The various functionalities that are present in a radio transceiver need different types of signal processing tasks and have different duty cycles. Furthermore the core computation of these different functionalities also vary substantially. For example, the inner modem requires mostly a fairly irregular computation with a large variation across the different standards, whereas the outer modem computation is a much more regular computation that needs limited flexibility. Each of these blocks also exhibit different types of parallelism (data level, instruction level, and task level). To reach maximal energy efficiency it is more efficient to tune one (or more) cores to a given functionality rather than to make a very flexible core. Various research works such as [3]–[5] have pointed towards optimally adapting the processor to make an application-specific instruction-set processor (ASIP). This gives almost an order of magnitude difference in the energy efficiency compared to a DSP or a reduced instruction set computer (RISC). Given the above reasoning, it is quite evident that for a power efficient platform there exists a need for a heterogeneous multicore solution.

Furthermore within a single standard [e.g., wireless local area network (WLAN)] multiple modes exist. Each mode may require different signal processing tasks (e.g., different multiantenna transmission schemes) and also different computational load. Figure 2 shows the performance requirement for some of the different modes of WLAN and long-term evolution (LTE) standards. The performance requirement is based on the number of 16-b RISC giga-operations per second (GOPS) required for the inner-modem for peak-payload processing only. Performing all the computation on the same core running at a higher speed would be a very energy-inefficient solution. This confirms the need for a multicore solution for SDR. Furthermore, the computation have to be performed on ASIP-like architectures instead of a DSP. A back-of-the-envelope computation for WLAN multiple input multiple output (MIMO) 2×2 40 MHz that requires approximately 25 GOPS shows that an application-specific integrated circuit (ASIC) that has an energy per operation of 5–10 pJ/op [3] gives a power of 125 mW, a DSP that has a power efficiency of 125–250 pJ/op gives a power of 3 W, whereas an SDR ASIP that has a power efficiency of 15–30 pJ/op would give 375 mW. The power efficiency of 15–30 pJ/op is based on Interuniversity Microelectronics Centre's (IMEC's) SDR solution [6]. Because the heat dissipation in a handheld device should be kept under 3 W [7] and RF parts and user interface consumes approximately 1.5 W [8], we see that using 3 W DSP is not feasible. This further confirms the need for a more specialized and heterogeneous ASIP solution rather than a general purpose solution.

An evolving step in the SDR community is the need for supporting multiple standards on the platform in parallel. This is



[FIG2] **Performance requirements for inner modem peak data/payload processing for WLAN and LTE.**

one of the essential features that would be needed for evolution into a truly cognitive radio solution. A multicore solution is therefore essential to support such a case.

SDR: THE NEED FOR MAPPING TOOLS

Multicore solutions make the mapping of the application(s) more difficult than ever before. The sequential aspect of the program that was kept before from the initial specification to final implementation has to change to parallel during application mapping. The demand for mapping tools that make this process easier rises drastically.

Wireless standards have hard real-time constraints on top of demanding throughput requirements, which makes the parallelization task even more complex. Figure 2 shows the throughput requirement for WLAN and LTE as well as an estimate of the computational requirement for the different modes. The computational load for only the inner modem computation ranges from a few 16-b GOPS to 150 GOPS on the most demanding case. Note that these are estimates of the performance under various assumptions of algorithmic choice, channel conditions, etc. To provide such a scalability, a need exists for tools to enable and explore the different parallelization schemes on the multiprocessor system.

This problem is further augmented by the possibility of exploiting parallelization at different levels. Platforms today exploit parallelization across the cores, across the threads within one core, across different functional units (FUs) within one core and within one FU (in a data parallel way). Exploring this large mapping space manually is not feasible. Thus, a selective tool support at different levels is crucial.

An important aspect for the designer is not only to pick right tools to help him/her with the parallelization but also to think about the whole mapping flow and combination and interoperability of the tools to achieve wished global optimality. The ordering of parallelization exploration at different levels is also crucial as we will see later in the text.

MULTICORE SDR ARCHITECTURES: A COMBINATION OF HETEROGENEOUS AND HOMOGENEOUS MULTICORE APPROACHES

Radio transceiver ASICs for one standard consist of accelerators for the different functional blocks in the transmitter/receiver chain. These ASICs give the extreme end of the spectrum with little or no flexibility. Next-generation radios have been evolving into more programmable and more configurable solutions. The increased amount of standards to support and the dynamism in each of these standards have pushed baseband radio implementations towards a software centric end. This has lead to an evolution where the baseband radio is implemented on flexible and programmable processors (SDR platforms) instead of pure ASIC-based solution. An overview of the freedom and the evolution of the flexibility is shown in [9].

Future SDR platforms will require multi-Gb/s connectivity, concurrency support, and spectrum sensing capabilities. This is not feasible without multicore SDRs. Different multicore approaches exist in reconfigurable radio architectures. Mostly, the combination of heterogeneous and homogeneous multicore approach is a viable option. At the top level, the platform resembles a heterogeneous system, with a specialized digital front end (DFE), inner modem, and outer modem (forward-error correction) part. Each part potentially consists then of homogeneous multicore subsystem. Such a system should be able to run multiple future standards up and beyond to 1 Gb/s, allow sharing of hardware resources among several standards and support runtime (RT) mechanisms at hardware and software level as well as supporting spectrum sensing capabilities.

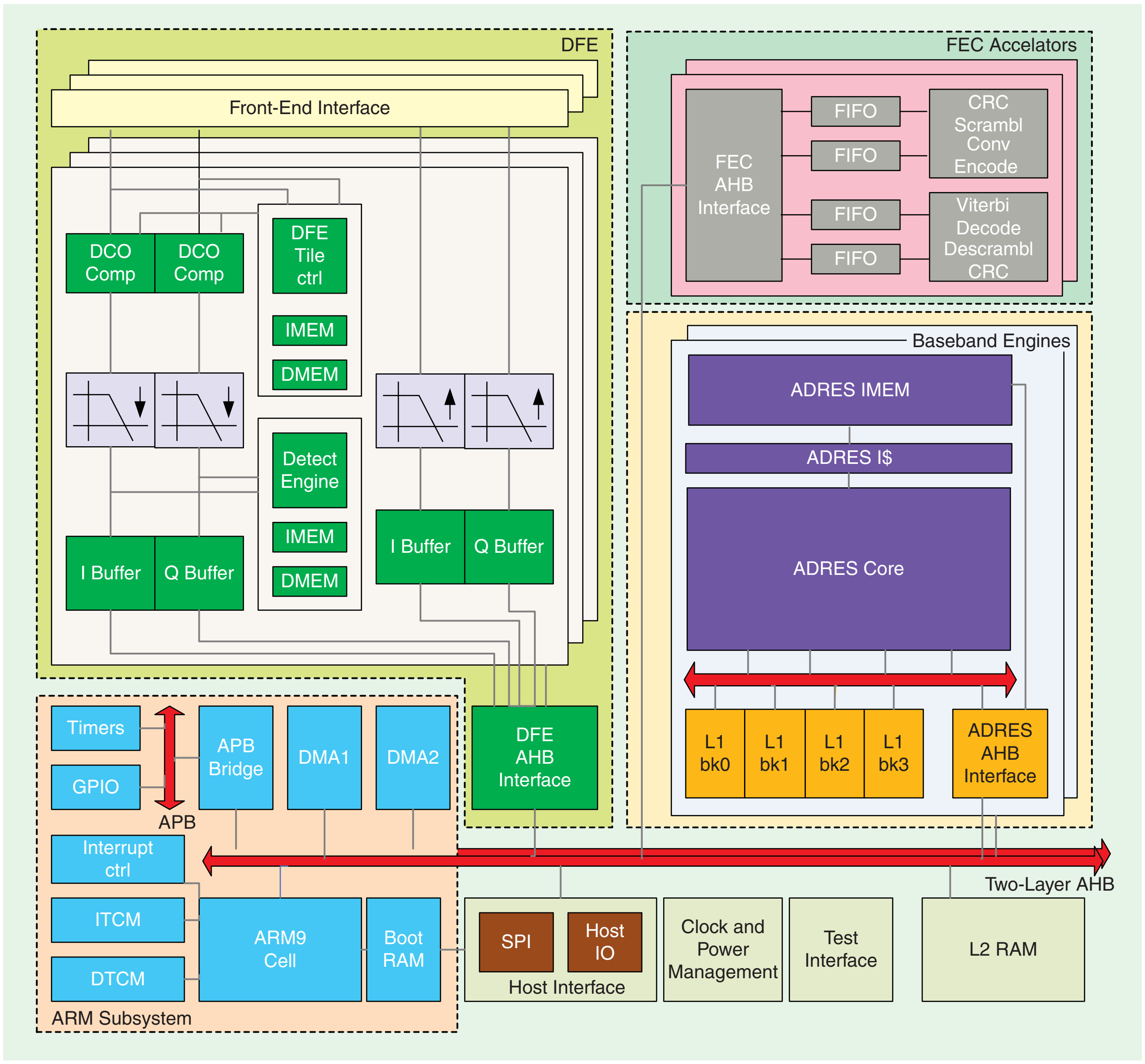
Given the large amount of software present inside these standards, to meet the high-performance and low-power requirements, there is a need for efficient exploitation of the different types of parallelism present. Broadly the parallelism can be broken down into three types: instruction-level parallelism (ILP), data-level parallelism (DLP), and task-level parallelism (TLP). ILP is when multiple instructions are executed in the processor in parallel, DLP is when multiple data elements undergo the same processing in parallel, and TLP is when multiple threads or tasks run in parallel on the processor or platform. Note that TLP can be exploited inside a single processor (intracore) or among processors (intercore).

Different SDR platforms exploit different types of parallelism in a better or worse way. Next, we give an overview of state-of-the-art multicore SDR platforms and highlight their important features. In the section "Comparative Study of Different Solutions," we highlight the features of the different platforms and summarize the pros and cons of the listed SDR platforms.

IMEC'S BEAR PLATFORM

IMEC's baseband engine for adaptive radio (BEAR) platform is a multicore heterogeneous platform consisting of six cores and two accelerators (see Figure 3). The six processors include three ASIPs for coarse time synchronization (DFE), one ARM processor for control (ARM subsystem), and two architecture for dynamically reconfigurable embedded systems (ADRES) processors (baseband engines) for baseband inner-modem processing. The coarse time synchronization ASIP is a low-power very-long instruction word (VLIW) with two scalar and three vector issue slots. The ADRES processor is a coarse grain reconfigurable array (CGRA) processor that is highly flexible and energy efficient. More information on the ADRES processor template can be found in [10]. The platform also contains accelerators for Viterbi decoding [forward error correction (FEC) accelerators]. The ARM processor is capable of performing control on the platform as well as to perform the MAC processing on the data stream. All the different cores are connected via an advanced microcontroller bus architecture (AMBA) for communication.

The BEAR platform offers a good mix of homogeneous and heterogeneous intercore TLP. Both the ADRES as well as the DFE processors have been designed to provide the appropriate mix of DLP and ILP for their corresponding tasks. Since the outer modem processing requires low flexibility



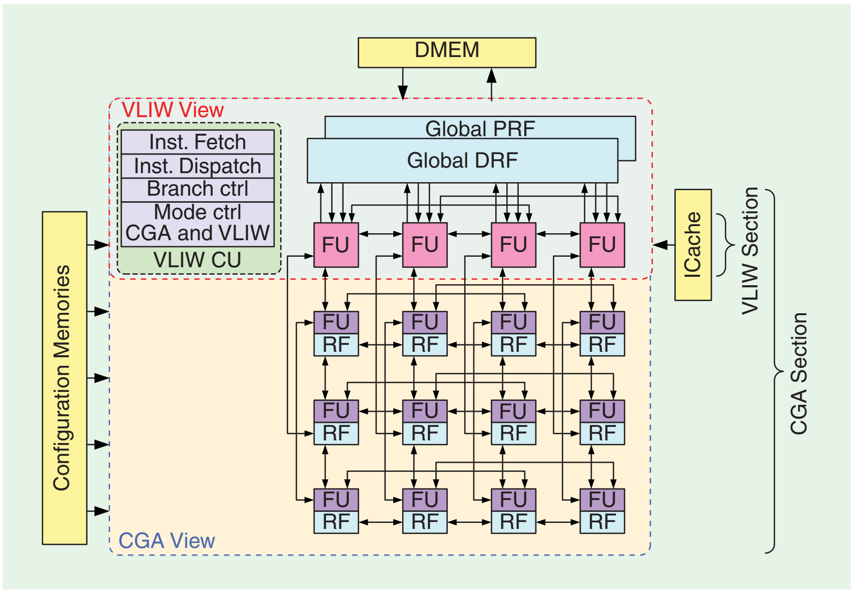
[FIG3] IMEC's BEAR SDR platform.

and high computation, it has been implemented as an ASIC accelerator. The inner modem, which requires high flexibility across different standards and inside one standard, has been implemented as a programmable processor. More detailed measurement results of the platform can be found in [6].

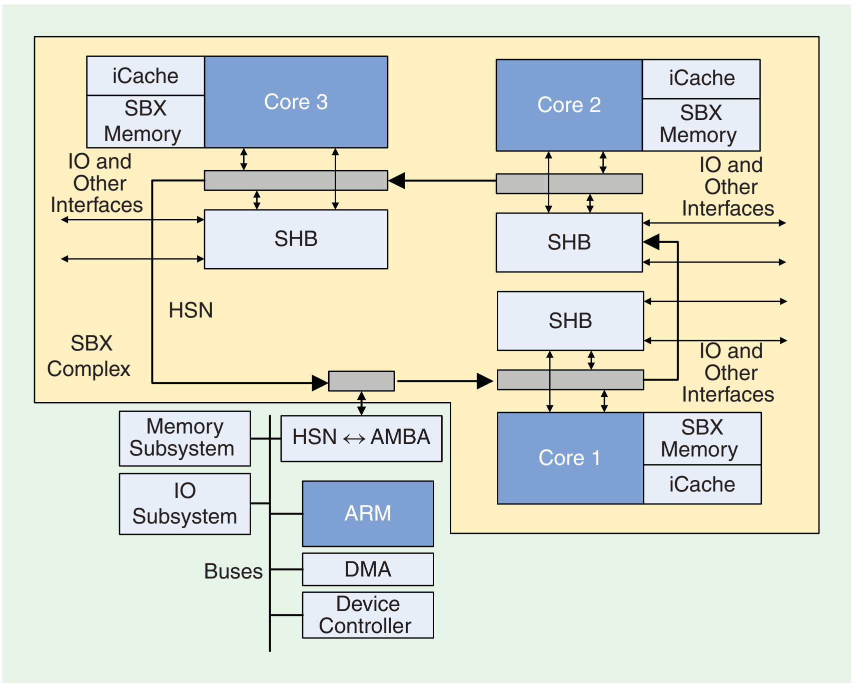
For the baseband processing, two ADRES processors are used. One such ADRES processor is shown in Figure 4. Each of the different FUs in the ADRES processor supports single instruction multiple data (SIMD) operations. Given that each ADRES processor offers ILP and DLP, and there are two such processors (TLP can be also used), a good parallelization strategy is a must to obtain an efficient mapping. Because the wireless application domain offers all the three different types of parallelism, various tradeoffs are possible on the type of parallelization strategy chosen, and each choice would have a different cost impact. These tradeoffs become even more varied when each standard to be mapped has various different modes, each of which have different computation and communication requirements.

SANDBRIDGE/SB3500

Sandbridge's SB3500 [11], [12] is the latest SDR platform generation from Sandbridge. The block diagram of the platform is depicted in Figure 5. This platform consists of four cores connected on an AMBA bus. Similar to other SDRs the control and the management of the platform is performed on the ARM processor. The remaining three cores on the platform are custom cores from Sandbridge called Sandblaster. Thus, the heterogeneity on this platform is very limited, differentiating only between the control (ARM) and the data processing (Sandblaster). All the inner- and outer-modem processing is done on the three Sandblaster cores.



[FIG4] IMEC's ADRES processor in the BEAR platform.



[FIG5] Sandbridge SB3500 platform architecture.

Each of the three Sandblaster cores has support for SIMD instructions and thus it can exploit the DLP available in the application. Because the platform consists of three data processing cores, inter-TLP among the different tasks in the application can be also exploited on the platform. Each Sandblaster core also offers a fine-grain intra-TLP inside a single core. This intracore parallelism is also referred to as "token triggered threading" (T^3), which is a form of simultaneous multithreading (SMT). Support for SMT allows the core to switch between different threads and their contexts quickly. However, the Sandblaster core has only limited ILP where only four instructions can be executed in parallel.

INFINEON MUSIC

Infineon's MuSIC-1 platform [9] is a heterogeneous multicore platform that consists of various accelerators along with four programmable cores. Each of these four programmable cores provides DLP and is used for the inner modem PHY processing with the help of filter accelerators. The turbo/Viterbi accelerators are used for performing the outer modem PHY processing. The block diagram of the platform is depicted in Figure 6.

The multicore nature of the MuSIC-1 platform supports intercore TLP, which allows the mapping of different tasks on different cores. Similar to Sandbridge, the ILP inside a single core is limited.

ST-ERICSSON EXTREME VECTOR PROCESSOR PLATFORM

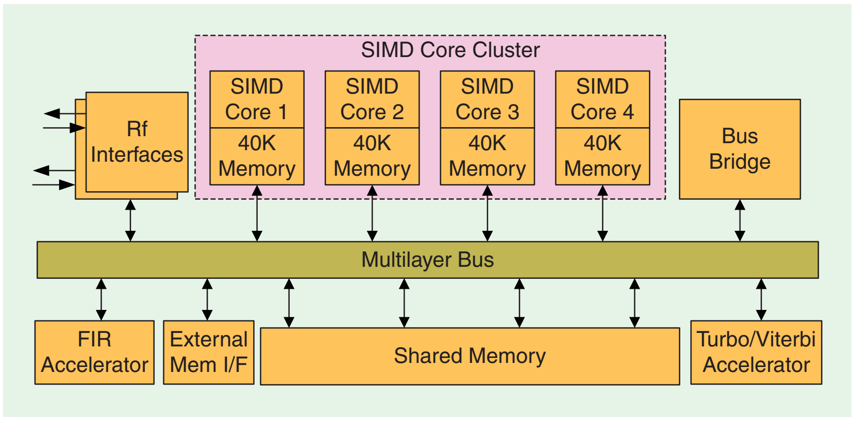
The extreme vector processor (EVP) [13] consists of 16-wide SIMD processor with five issue slots. Three of the five slots operate on vector data and two operate on scalar data. This processor exploits both data- and instruction-level parallelism in the application. However, not much public information is available on the complete platform architecture and how many cores would be needed to support a wireless standard.

ARM/UNIVERSITY OF MICHIGAN'S ARDBEG PLATFORM

ARM/University of Michigan's Ardbeg platform [14] consists of three processor cores. Two cores are allocated for baseband processing and one core for control. The platform also consists of a turbo coprocessor for outer-modem processing (see Figure 7). The platform enables TLP to be exploitable between the four functional blocks (control processor, two baseband cores, and a turbo accelerator). Each of the baseband cores is 512-b wide and is capable of performing 64-way, 32-way, and 16-way SIMD on 8-b, 16-b, and 32-b data, respectively. However, the baseband core does not allow a large amount of ILP inside the core. The baseband processor is also used to perform certain outer-modem functionality such as Viterbi decoding.

OTHER SOLUTIONS

Other platforms include Silicon Hive's CSP series [15]. These processors allow a large mix of ILP and DLP that can be exploited in the processor. However, there is not enough public information so it is not clear how these processors would fit on a platform, what parts of the PHY would map on the processor, and where the MAC processing would be mapped. Picochip's PC205 [16] solution also offers a mix of ILP, DLP, and TLP on the platform. However, this platform consists of a large number of hardware accelerators that pushes the platform towards a less flexible solution. Ceva's Ceva-XC platform [17] also consists of a mix of DLP and ILP available on the platform. The vector cores used on the platform consist of special instructions to accelerate the outer-modem processing.

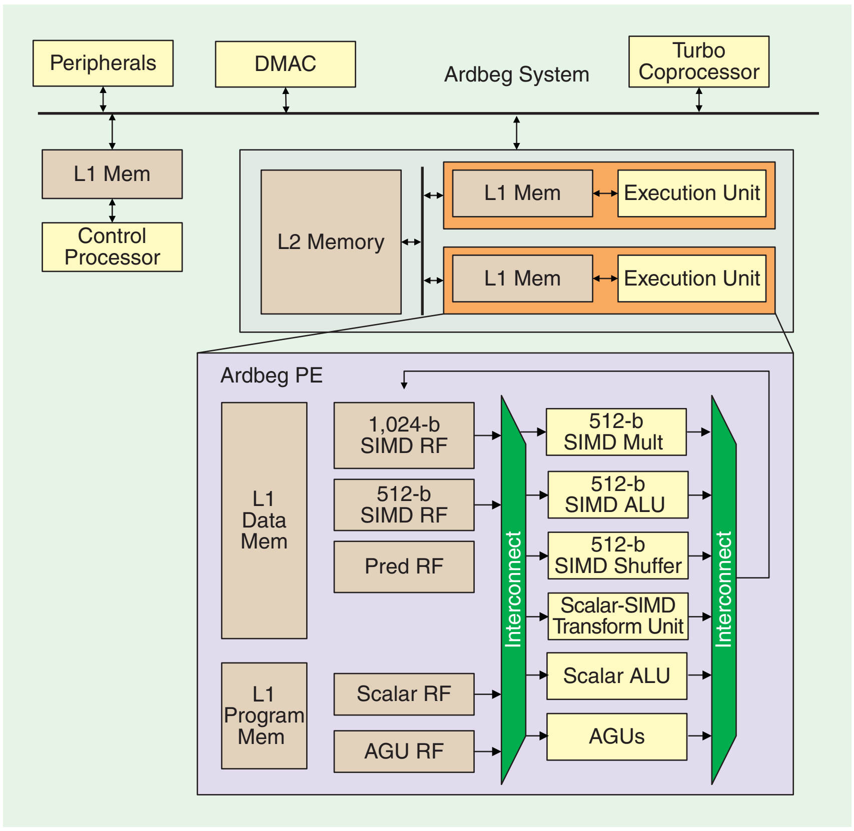


[FIG6] Infineon MuSIC-1 platform architecture.

COMPARATIVE STUDY OF DIFFERENT SOLUTIONS

Table 1 gives a comparative study of the parallelism offered by the different platforms. Each platform offers a different mix of parallelism to the programmer. It is interesting to note that all these platforms offer a high- to medium-data level parallelism. This is largely because of the fact that DLP is an energy-efficient way to exploit parallelism and most standards offer data-level parallelism. However, the other types of parallelisms are quite varied across the different platforms. To reach the required performance, each platform exploits different types of parallelism on top of the DLP.

In terms of area and power, it is very difficult to perform a good comparison with the information available in the public domain. Based on public information, it is not clear what the precise set of functionality is running on the platform, what the duty cycle of the processing is, what the area includes, what mode it is measured under, and what level of power/area estimation is used. Furthermore, the objective function that is even harder to compare is flexibility, platforms may have a specialized instruction set or accelerators that may heavily limit flexibility to port another standard on it.



[FIG7] ARM/University of Michigan's Ardbeg architecture.

[TABLE 1] COMPARISON OF PARALLELISM OFFERED BY DIFFERENT PLATFORMS (L: LOW, M: MEDIUM, H: HIGH).

PLATFORM	DLP	ILP	INTER TLP	INTRA TLP
CEVA-XC [17]	M	M	L	L
IMEC BEAR [6]	M	H	M	L
UMICH/ARM [14]	H	L	L	L
ST-NXP [13]	H	L	L	L
INFINEON'S MUSIC-1 [9]	M	M	L	M
SANDBRIDGE'S SB3500 [11], [12]	M	L	M	H

MAPPING FLOWS FOR MULTICORE RADIO ARCHITECTURES

As shown in previous sections, different state-of-the-art SDR platforms exhibit different granularity and type of parallelism.

Programming such radio architectures requires having an appropriate mapping flow supported by the tools that can explore TLP, DLP, and ILP. Mapping process starts from algorithmic specification of a radio standard that is tuned to a optimized code utilizing DLP and ILP and keeping task distribution (TLP) in mind.

SYSTEMATIC GLOBAL FLOW PRINCIPLES FOR EXPLOITING PARALLELIZATION

As mentioned before, different types of parallelism can be exploited in the application: TLP, DLP, and ILP. These types of parallelism have different granularity and impose different constraints on the remaining part of the mapping flow. Because of its middle granularity, ILP is the most restrictive type of parallelism and it should be applied as the last step in the parallelization mapping flow. Applying ILP too early can seriously restrict some TLP options. For example, when applying the transformations to achieve a good ILP in a certain part of the application, it might not be possible to split this part to two separate tasks any more. Reversed ordering, i.e., applying ILP after TLP, is less restrictive. DLP has the fine granularity and thus it can be applied very locally. Still, when exploiting ILP, DLP should be already explored. This can be motivated by the fact that DLP can always be broken down into ILP and not vice versa. However, DLP should be exploited in the individual tasks, because different tasks can utilize different DLP strategies. Thus, DLP exploration should be placed between TLP and ILP exploration. TLP itself has two subclasses, interprocessor TLP, which is TLP across several processor cores, and intraprocessor TLP, which is TLP across several threads within one core. To perform interprocessor TLP first and then intraprocessor TLP is natural order as it provide the lowest constraints on the available search space freedom. The tasks operating on different cores should have minimal communication, whereas the tasks running on different threads within one core can afford more communication overhead.

When exploiting TLP, functional and/or data split can be applied. The functional split assigns different functionality to different tasks. The data split assigns different iteration ranges of the same functionality to different threads. Also, a combination of both is possible. A specific combination is task pipelining, when different functionality in different iteration ranges is executed in parallel. Data split in TLP might limit the DLP, however, the freedom for DLP is not so limited as it would be in reversed ordering. This also confirms the need to apply TLP before DLP.

WORKLOAD ESTIMATION ISSUES IN THE MAPPING FLOW

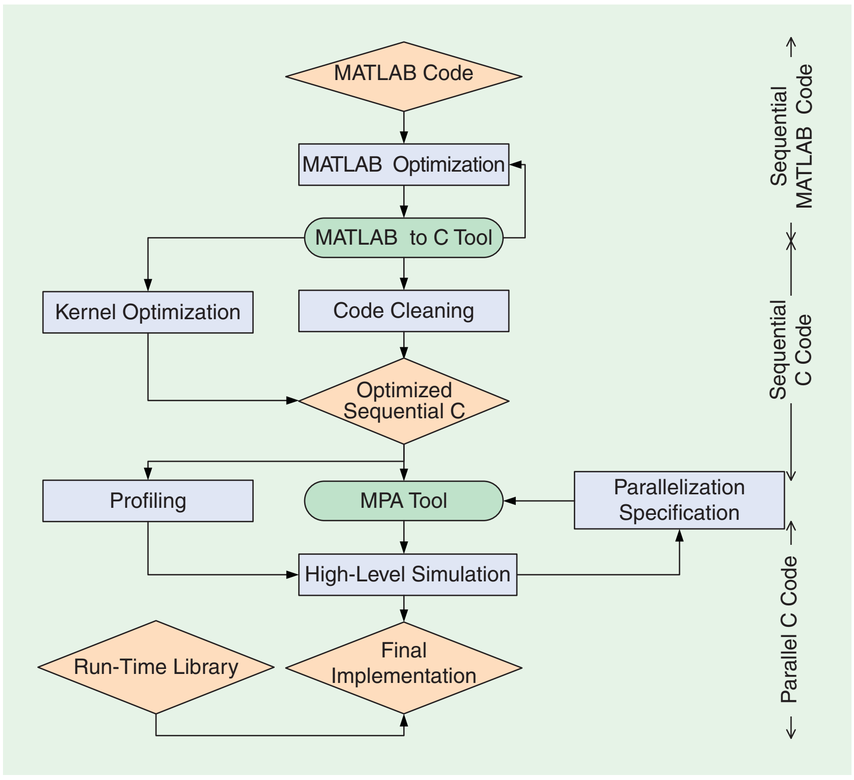
In the section "Systematic Global Flow Principles for Exploiting Parallelization," the flow requires an estimate of the workload for load balancing in TLP and estimation of communication overhead. This requires exploration of DLP and ILP first to obtain the timing information. There are two possibilities that can solve this issue. The first one is to utilize high-level estimators during decision on the TLP parallelization strategy. Those estimators provide the designer with the upper and lower bounds of DLP and ILP. The second possibility is to rely on an experienced designer that can perform those estimations "in his head." We can observe this in most practical mapping flows. Even when we start with DLP and ILP parallelism that can be exploited within one processing core, we also implicitly explore TLP at the beginning of the flow. This type of practical mapping flow was also confirmed by the Multicore Association, which encompasses many important industrial players [18]. If the designer is not experienced enough and performs wrong implicit TLP exploration, he/she will enter in the global loop where he/she has to return to the DLP or ILP exploration when no satisfactory TLP solution has been found. Those loops are, of course, too costly and are very rarely entered by an experienced designer. However, it is still possible. The high-level estimators can eliminate these errors and thus are a crucial component for a more automated future mapping flows, especially when RT managers have to make these decisions (see the section "Dynamic Scalability of Baseband Signal Processing: Impact on Mapping").

BEAR MAPPING FLOW

Every mapping flow starts with initial algorithmic specification and ends with final implementation targeting the best performance, energy and/or area on a given platform. During the mapping flow, different transformations are applied that expose certain properties of the application. In the BEAR mapping flow, we have mainly focused on exposing parallelization at different granularity levels, that allows us utilizing the platform resources efficiently. These transformations are orthogonal with transformations targeting other issues such as optimizing the memory hierarchy system [19], [20].

The BEAR mapping flow in Figure 8 starts with initial MATLAB algorithmic specification. Then, high-level MATLAB transformations are applied. Those optimizations allow efficient C code generation later and also include global data-flow and loop transformations [19] , [20]. The code is quantized and MATLAB to C conversion tool [21] is used to generate the C code. The C code is split into kernels such as FFT, tracking, channel compensation, demodulation, and skeleton code that is calling these kernels. The skeleton code is cleaned, i.e., the constructs not supported by the parallelization flow are rewritten (such as dynamic allocation). The kernels are optimized in separate path. First, special intrinsic instructions of the target ADRES processor [10] are used that allow SIMD operations. After exploiting DLP, ILP is exploited by (low-level) loop transformations such as loop unrolling, loop coalescing, if conversion, code hoisting etc. Note the difference with the loop transformations at MATLAB level that are applied more globally, not only within one loop nest. Precompilation of the kernels using our DRESC compiler [10] is ending the kernel optimization process.

After kernel optimization, precompiled kernels can be combined with the cleaned skeleton code resulting in efficient sequential implementation that is profiled. To exploit TLP across the kernels, we utilize our MPA in-house tool [22]. MPA takes as input the sequential C code and a parallelization specification, based on which the parallel code is generated. The synchronization is automatically inserted in the parallel code when needed to obey the original dependencies. The parallelization specification distributes each iteration instance of the different kernel among the different threads and/or processors (intra- and inter-TLP). When knowing the durations of the kernels in each iteration via the profiling (see Figure 8), the performance of the parallelized code can be rapidly evaluated by our high-level simulator. The parallelization is specified by parallelization specification file written by the designer. As previously mentioned, the experienced designer will have the possible parallelization specifications "in his head" even before starting the DLP and ILP exploration. After selecting the best parallelization strategy, the parallelized code is combined with the RT library to bridge the MPA tool high-level application programming inteface (API) and the platform low-level API to achieve the final implementation. On the BEAR platform, the threads and communication between them is controlled by the ARM processor.



[FIG8] **BEAR mapping flow.**

Our mapping flow allowed exploration of different parallelization possibilities such as an antenna- and symbol-based split for WLAN MIMO 2 $\times$ 2 40 MHz. It also allowed achievement of the real-time throughput behavior [23] and exploration of the maximum feasible parallelism for increased number of processing cores for the same application. We also experienced the global loops in the flow that are mentioned in the previous section. When we first optimized the kernels for DLP and ILP with focusing on per-symbol TLP split [23], it was not feasible with the same DLP and ILP solution to perform the antenna TLP split. The mapping flow can be used as in multiprocessor context (inter-TLP) so in multithreading context (intra-TLP).

SANDBRIDGE MAPPING FLOW

In the case of the Sandbridge architecture described in the section "Sandbridge/SB3500," not many details are available on a full methodology. Based on the mapping strategies described in [24] and [25], it is clear that the intercore thread-level parallelism is first exploited, followed by intracore thread-level parallelism. The process of deciding on the intracore threads is fine-grained compared to other platforms as the Sandbridge processor exploits the T^3 technology. The T^3 technique allows quick context switches among multiple threads on the same core to up to eight threads on a single core. After deciding the inter- and intrathreads, the data level and the instruction-level parallelism is finally exploited. The decision on the parallelization strategy and the parallelization itself is left to the developer.

TEXAS INSTRUMENTS' ALGORITHM ARCHITECTURE MATCHING METHODOLOGY

The algorithm architecture matching (AAM) methodology developed by Texas Instruments maps an algorithm that is described as a graph to a physical architecture given a set of constraints [26]. The architecture is described as the architecture graph in which vertices represent operators (DSP cores) and the edges represent communication. The AAM methodology takes the two graphs and set of constraints and it performs placement and scheduling of the algorithm graph nodes over architecture graph nodes. This resembles IMEC's dynamically reconfigurable embedded system compiler (DRESC) modulo scheduling approach [10], but on a much coarser level. The architectural nodes in AAM are complete DSP cores where as in the DRESC, the architectural nodes are FUs in the CGRA part of the processor.

The AAM methodology is employed using the parallel realtime embedded executives scheduling method (PREESM) tool. Even when the tool can generate code, the input of the tool has to be described as a synchronous data flow (SDF) graph, which is increasing the manual effort during the mapping process. On the other side, once the SDF graph is present, the exploration for different architectures is straightforward. Of course, appropriate architecture graphs have to be present.

INFINEON MUSIC MAPPING FLOW

For the MuSIC platform described in the section "Infineon MuSIC," the mapping flow starts from a functional C description of the complete standard. This is further refined by choosing the right parallelization strategy in both the thread level and data level [27], [28]. The DLP parallelization (SIMD) is then exploited using extensions to the C language as described in [29]. As mentioned in the section "Workload Estimation Issues in the Mapping Flow," the SIMD estimations are performed in the head of the designer before performing the thread-level parallelization. The DLP parallelization (using SIMD instructions) itself is performed manually, which is often common in most design flows. Furthermore, the MuSIC platform also has a lightweight real-time operating system (RTOS) called ILTOS to enable multithreaded programming. The API of ILTOS provides basic functions for thread administration, memory management, synchronization etc. Communication API is similar to the API provided by IMEC's MPA tool.

THE FAST EVOLUTION OF WIRELESS STANDARDS HAS CAUSED A SHORTER TIME-TO-MARKET, WHICH MAKES A PROGRAMMABLE SDR SOLUTION ATTRACTIVE.

SPEX—A PROGRAMMING LANGUAGE FOR SDR

SPEX [30], from the University of Michigan, is an object-oriented programming language based on C++ semantic targeting the SDR platforms. Three additional keywords are added to the language kernel, stream, and synchronous to distinguish the sequential C kernels in the application, concurrent data streaming, and discrete real-time computations. This provides a clear interface among the DSP algorithm description in kernel SPEX, parallel execution in stream SPEX, and system integration in synchronous SPEX. The different SPEXs are compiled by different compilers in different phases of the mapping; kernel SPEX with the SIMD and VLIW compiler, stream SPEX with the data flow compiler, and synchronous SPEX with the real-time compiler. The SPEX flow is shown in Figure 9.

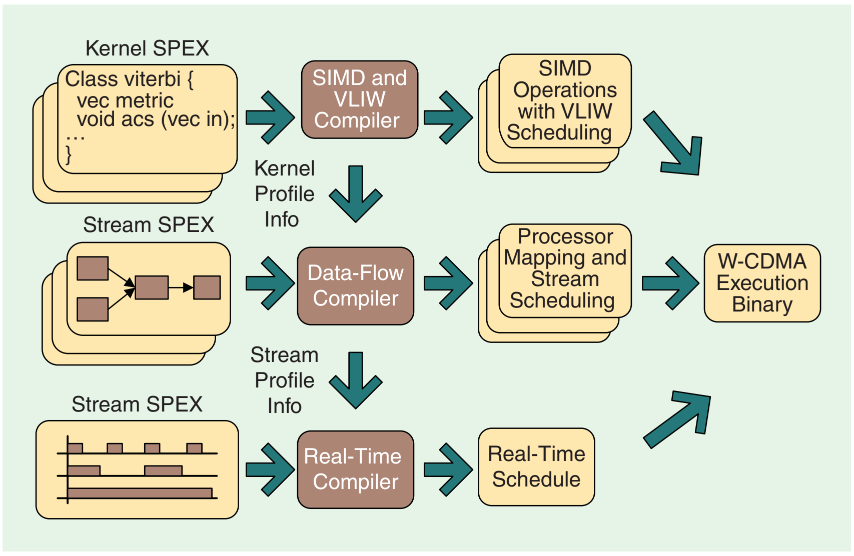
We consider this approach similar to our BEAR baseband mapping flow approach (see the section "BEAR Mapping Flow"). The kernel approach is similar to IMEC's approach where kernels are optimized and precompiled by the DRESC. The stream SPEX can be compared to the mixture of the parallelization phase in BEAR flow (using MPA) and the first part of system integration phase. Synchronous SPEX can be considered the last phase of the IMEC system integration step.

OTHER MAPPING FLOWS

In a task-transaction level (TTL) methodology [31] developed at University of Twente and Philips Research, an application is modeled as a task graph, where a task is an entity that performs computations. The implementations are encapsulated in TTL shells that are exposed to the platform. The TTL approach raises the level of the abstraction, and it tries to close the gap between application models used for specification and the optimized implementation of the application by its combination. However, it seems that the placement and scheduling process it manual to the large extend compared e.g., to the AAM approach described in the section "Texas Instruments' Algorithm Architecture Matching Methodology."

COMPARISON OF DIFFERENT MAPPING FLOWS

In previous sections, we first looked at the SDR mapping methodology from the meta-level perspective and then provided instantiations developed by different groups for different SDR platforms. The common drawback we see for most of the methodologies is either the need for graph description of the application (e.g., AAM) or just providing programming models for the SDR mapping (e.g., TTL) without any tool support. Both approaches can result in tedious and error-prone mapping. Also, most methodologies focus only on mapping for a particular homogeneous system (e.g., SPEX is targeting multicore SIMD DSP processor platform only) and we miss heterogeneity of given solutions. Currently this is the main challenging task we foresee in the SDR mapping. Some attempts have been made recently to cover this gap, i.e., extend the OpenMP for heterogeneous multicore systems [32]. In this context, we see the BEAR mapping flow as one of more complete solutions. Even though some parts of the flow are fully manual, the critical parts of the mapping are automated and tool support is present.



[FIG9] **SPEX design flow [30].**

FUTURE RESEARCH DIRECTIONS: MOVING TOWARD DYNAMIC MULTICORE COGNITIVE RADIO PLATFORMS

Multicore SDR and the associated mapping techniques enable applications that were not feasible in the single-core SDR era. In the following sections, we will discuss the trends that are enabled by multicore SDR. First, we will focus on concurrent data streams/connections and their impact on RT management. Then we will highlight dynamic scalability of wireless systems and its impact on mapping. Finally, we will tackle new complexity challenges for fourth generation (4G) and beyond, multiuser and network MIMO, and spectrum sensing.

CONCURRENT MULTIPLE CONNECTIONS/STREAMS

MOTIVATION

In the past, the management of concurrent data streams or connections mostly happened for SDR base stations. However, this functionality of management of concurrent streams is moving to the mobile terminals as well. For instance, a mobile terminal may need to stream encoded music data from a WLAN connection, and the decoded music needs to be sent to a wireless earphone with a Bluetooth link. The scenario of voice over Internet protocol calls with a wireless earphone is very similar. In many other cases, although multiple data streams are not explicitly visible for the user, the baseband processing still needs to tackle multiple connections with different streams of raw data. For instance, when the user is browsing the Internet with the WLAN connection of his or her mobile, in the background the mobile may be continuously ranging and synchronizing with multiple base stations for cellular links such as LTE or LTE-advanced. In the long term, tackling concurrent connections and streams will become even more important. In future wireless mesh and ad-hoc network, terminals may need to relay multiple data streams in the background, whereas the foreground data streams still have to be guaranteed. Clearly, tackling multiple connections and data streams simultaneously will be essential for future SDR devices.

IMPACT ON RT MANAGEMENT

Multiple streams and standards operation on the SDR platform in parallel will have also an impact on the RT management of the platform. The platform control should be more distributed compared to today's centralized control. The platform and the platform control should support coexistence of multiple standards and the hand-over between the standards. We foresee the flexibility supporting this during the RT in the next three planes.

- The multistandard plane supporting concurrent run of multiple standards in parallel ensures that each standard can run on separate processor core and/or as separate threads within a core, if the core can support intraprocessor TLP. Note that a standard can be distributed across several threads and/or cores.
- The horizontal plane ensures the flexibility and implementation scalability of a standard (within a given mode). For example, there are several possible implementations of WLAN MIMO 2 × 2 40 MHz, resulting in energy-time-area tradeoff, where one implementation can be energy efficient but not so performing as other, energy-hungry implementation. The implementations can differ also in code size that is reflected in the area axis of the tradeoff.
- The vertical plane ensures mode scalability of a standard, i.e., that we can switch among different modes of the given standard. A typical example is the single input single output/MIMO mode support where a decision is made based on the incoming signal field (in the case of WLAN).

Those planes are important also in a hand-over scenario, where the situation could be as follows. One standard that is running on the platform scales down (horizontal plane) to free space for the second standard. For a certain period of time, the two standards run parallel and the hand-over will happen. Then the first standard is stopped and the second standard is scaled to all available resources. Thus, the hand-over issue is naturally connected to concurrent connections and RT support for the future platforms should be adapted to this.

DYNAMIC SCALABILITY OF BASEBAND SIGNAL PROCESSING: IMPACT ON MAPPING

In wireless systems, both the environment and the user requirement contain abundant dynamics. First, the environment is inherently time varying, e.g., channel conditions, interferences, and spectrum utilization. In addition, the user requirement also contains lots of dynamics, e.g., data rate, tolerable error, tolerable jitter, and tolerable latency. Baseband with dynamic scalability can adjust processing complexity according to the above dynamics [33]. For instance, the search range of nonlinear MIMO detectors, the modulation accuracy of orthogonal frequency division multiple access modulator, and the tracking strength of channel estimators can be adjusted based on different metrics or requirements. These adaptations will lead to heavily reduced computations and memory accesses, which eventually translates into substantially reduced average energy consumption.

Although such dynamic scalability can improve energy efficiency, it imposes many challenges on multicore mapping. Importantly, the work load of tasks are not completely deterministic anymore, it will depend on channel conditions and input data. This brings complex situations when handling real-life baseband signal processing on a multicore platform. Worst-case mapping would suffer severely from efficiency, therefore a mix of design-time and RT decisions would be needed to reach an efficient solution.

NEW COMPLEXITY CHALLENGES

BASEBAND COMPLEXITY OF 4G AND BEYOND

International Mobile Telecommunications-Advanced (IMT-Advanced), has already initialized massive effort for the evolution beyond the 3rd Generation Partnership Project (3GPP), LTE, and IEEE802.16e. The planned improvement spreads over system architecture, radio resource management, MAC scheme, and air interface. Regarding the air interface, it is clear that very large bandwidth (e.g., 100 MHz) and larger MIMO systems (e.g., 4 × 4 or even higher) [34] will be implemented. This increased bandwidth directly translates to a complexity challenge that requires multiple-core baseband platform and efficient mapping therefore becomes more crucial.

To enable this, there is a need for further improvement in the platform architecture as well as in the algorithms to obtain the best combination of architecture and algorithm. Better use of the parallelism is definitely one of the key challenges to be addressed. Importantly, future algorithmic engineering will have to take into account architecture characteristics from the very beginning of the mapping flow, to ensure that TLP, DLP, and ILP can be effectively exploited in later mapping steps.

[CLEARLY, FUTURE MULTICORE SDR DEVICES WILL BE ESSENTIAL FOR TACKLING MULTIPLE CONNECTIONS AND DATA STREAMS SIMULTANEOUSLY.]

MULTIUSER MIMO AND NETWORK MIMO
Multiuser MIMO and network MIMO have been considered as a promising candidate technology for highly spectrum efficient wireless systems. Whereas traditional MIMO leaped from the traditional multipath avoidance and compensation paradigm to the multipath exploitation paradigm, multiuser MIMO and network MIMO jump one step further, shaping interference and exploiting interference. However, such a new paradigm brings higher spectrum efficiency at the cost of significantly increased computation complexity. For multiuser MIMO, the condition or capacity of many user-specific MIMO channels have to be analyzed, steered, or compensated. Complex signal processing algorithms, such as eigenvalue spread analysis, will become necessary. For network MIMO, joint processing and detection will increase the dimension of signal processing, the increment of complexity would be orders of magnitudes.

SPECTRUM SENSING
One of the key enablers of next-generation cognitive radio systems would be "spectrum sensing" [1], [35]. This would imply that each of the different users would be able to sense the spectrum usage in the air and use the spectrum that would be the most optimal in a more dynamic way. This would enable each user to use the spectrum in a more effective way. Various standardization efforts like IEEE 802.22 [36] are currently in progress to standardize cognitive radio and sensing requirements. This may not only happen in the digital TV bands but spectrum sensing would also help a better coexistence scenarios between various standards like 802.11 and 802.15.4. Such spectrum sensing techniques would also bring higher levels of dynamism and complexity to the mobile terminal.

CONCLUSIONS
In this article, we provided an overview of multicore architectures for future SDR platforms and their mapping flows. First, we motivated the need for scalable and reconfigurable heterogeneous multicore SDR platforms driven by technology constraints, user demands, and business aspects. We also highlighted the urgent need for appropriate mapping flows when mapping on those platforms. Then we gave an overview of the most popular multicore SDR platforms on the market with a short comparative study among them. In the mapping flow section, we started a discussion on general mapping flow going to different instances of the mapping flows. Finally, we discussed the new aspects and applications the multicore SDR era with proper mapping flow will bring.

AUTHORS
Martin Palkovic (palkovic@imec.be) received his M.Sc. degree in electrical engineering (with highest distinction) from the Slovak University of Technology, Bratislava, Slovakia, in 2001, and his M.Sc. degree in economics from the University of Economics, Bratislava, Slovakia, in 2000. He joined IMEC in 2001, where he was a researcher in the design technology group from 2001 to 2008 and has been a senior researcher in the wireless group since 2009. From 2002 to 2007, he worked towards his Ph.D. degree in the Department of Electrical Engineering at Technische Universiteit Eindhoven, The Netherlands. His research interests include high-level optimizations in data dominated multimedia applications and wireless systems, platform architectures for low power, and mapping techniques for multicore SDR.

Praveen Raghavan (ragha@imec.be) received his bachelor's degree from the National Institute of Technology, Trichy, India, in 2002. He received his master's degree in electrical engineering from Arizona State University in 2004, and his Ph.D. degree from K.U. Leuven in electrical engineering in 2009. He is currently a wireless systems researcher in the wireless group at IMEC. His research interests include low-power design, system design, low-power architectures, and SDR.

Min Li (limin@imec.be) received his B.E. degree (with the highest honor) in 2001 from Zhejiang University, Hangzhou, China. From 2001 to 2004, he was a postgraduate student with Zhejiang University. In 2003, he was an intern with Lucent Bell Labs Research China, working on network processors. From September 2003 to September 2004, he interned with Microsoft Research Asia, working on low-power mobile computing. From 2004 to 2009, he was a Ph.D. researcher with IMEC and a Ph.D. student with the ESAT group at K.U. Leuven. He received the 2007 IEEE SIPS Best Paper Award. He is currently a researcher with IMEC. His research interests include low-power signal processing and implementation.

Antoine Dejonghe (dejonghe@imec.be) received the electrical engineering degree and the Ph.D. degree from the Université Catholique de Louvain (UCL), Louvain-la-Neuve, Belgium, in 2000 and 2004, respectively. From 2000 to 2004, he pursued a Ph.D. degree with the Communications and Remote Sensing

Laboratory as a research fellow of the Belgian National Fund for Scientific Research. Since December 2004, he has been with the wireless group of IMEC, Leuven, where he is currently a program manager for cognitive reconfigurable baseband activities. He is the author and coauthor of over 75 scientific publications.

Liesbet Van der Perre (vdperre@imec.be) received the M.Sc. degree in electrical engineering from K.U. Leuven, Belgium, in 1992. The research for her thesis was completed at the Ecole Nationale Superieure de Telecommunications in Paris, France. She graduated with a Ph.D. degree in electrical engineering from K.U. Leuven in 1997. After joining IMEC in 1997, her focus was on ASIC architectures for OFDM, turbo coding, and SDR radio. Currently, she is a program director for the cognitive reconfigurable radios and mm-wave communications programs at IMEC. She is a part-time professor at K.U. Leuven, Belgium, and she is an author and coauthor of over 200 scientific publications.

Francky Catthoor (catthoor@imec.be) received the engineering and a Ph.D. degrees in electrical engineering from Katholieke Universiteit Leuven, Belgium, in 1982 and 1987, respectively. Between 1987 and 2000, he headed several research domains in the area of high-level and system-synthesis techniques and architectural methodologies, including related application and deep submicron technology aspects, all at IMEC, Heverlee, Belgium. He is an IMEC fellow. He is a part-time full professor in the Electrical Engineering Department of K.U. Leuven. In 1986, he received the Marconi International Fellowship Council's Young Scientist Award. He has been an associate editor for several IEEE and ACM journals, including *IEEE Transactions on VLSI Signal Processing, IEEE Transactions on Multimedia,* and *ACM TODAES*. He was the program chair of several conferences including ISSS'97 and SIPS'01. He is an IEEE Fellow.

REFERENCES

[1] J. Mitola, "Cognitive radio architecture evolution," *Proc. IEEE*, vol. 97, no. 4, pp. 626–641, Apr. 2009.

[2] V. Giannini, J. Craninckx, S. D'Amico, and A. Baschirotto, "Flexible baseband analog circuits for software-defined radio front-ends," *IEEE Solid-State Circuits*, vol. 42, no. 7, pp. 1501–1512, July 2007.

[3] W. J. Dally, J. Balfour, D. B. Shaffer, J. Chen , R. C. Harting, V. Parikh, J. Park and D. Sheffield, "Energy efficient computing," *IEEE Comput. Mag.*, vol. 41, no. 7, pp. 27–32, July 2008.

[4] M. Gries, K. Ketuzer, H. Meyr and G. Martin, *Building ASIPS: The Mescal Methodology*. New York: Springer, 2005.

[5] P. Ienne and R. Leupers, *Customizable Embedded Processors: Design Technologies and Applications*. San Francisco, CA: Morgan Kaufman, 2006.

[6] V. Derudder, B. Bougard, A. Couvreur, A. Dewilde, S. Dupont, L. Folens, L. Hollevoet, F. Naessens, D. Novo, P. Raghavan, T. Schuster, K. Stinkens, J.-W. Weijers, and L. Van der Perre, "A 200 Mbps+ 2.14nJ/b digital baseband multi processor system-on-chip for SDRs," in *Proc. VLSI Circuits Symp.*, June 2009, pp. 292–293.

[7] Y. Neuvo, "Cellular phones as embedded systems," in *Proc. IEEE Int. Solid-State Circuits Conf.*, Feb. 2004, vol. 1, pp. 32–37.

[8] O. Silven and K. Jyrkka, "Observations on power-efficiency trends in mobile communication devices," *EURASIP J. Embedded Syst.*, vol. 2007, no. 1, pp. 17–27, 2007.

[9] U. Ramacher, "Software-defined radio prospects for multistandard mobile phones," *IEEE Comput. Mag.*, vol. 40, no. 10, pp. 62–69, Oct. 2007.

[10] B. Mei, S. Vernalde, D. Verkest, H. De Man, and R. Lauwereins, "ADRES: An architecture with tightly coupled VLIW processor and coarse-grained reconfigurable matrix," in *Proc. IEEE Conf. Field-Programmable Logic and Applications (FPL)*, Lisbon, Portugal, Sept. 2003, pp. 61–70.

[11] Sandbridge Technologies. (2009). *The Sandblaster Architecture* [Online]. Available: http://www.sandbridgetech.com

[12] M. Moudgill, J. Glossner, S. Agrawal, and G. Nacer, "The sandblaster 2.0 architecture and SB3500 implementation," in *Proc. Software Defined Radio Technical Forum*, Oct. 2008 [Online]. Available: http://www.sdrforum.org/sdr08_papers/2.5/2.5-3.pdf

[13] K. van Berkel, F. Heinle, P. P. E. Meuwissen, K. Moerman, and M. Weiss, "Vector processing as an enabler for software-defined radio in handheld devices," *EURASIP J. Appl. Signal Process.*, vol. 2005, no. 16, pp. 2613–2625, 2005.

[14] M. Woh, Y. Lin, S. Seo, S. Mahlke, T. Mudge, C. Chakrabarti, R. Bruce, D. Kershaw, A. Reid, M. Wilder, and K. Flautner, "From soda to scotch: The evolution of a wireless baseband processor," in *Proc. 2008 41st IEEE/ACM Int. Symp. Microarchitecture (MICRO-41)*, Nov. 2008, pp. 152–163.

[15] Silicon Hive. SiliconHive HiveFlex CSP (2009). [Online]. Available: http://www.siliconhive.com

[16] Picochip. Picochip PC205 product brief (2009). [Online]. Available: http://www.picochip.com

[17] CEVA Inc. CEVA DSP core X1641 product note (2009). [Online]. Available: http://www.parthus.com

[18] *Multicore Association. Multicore-Programming Practices Group* (2009). [Online]. Available: http://www.multicore-association.org/workgroup/mpp.php

[19] F. Catthoor, K. Danckaert, C. Kulkarni, E. Brockmeyer, P. G. Kjeldsberg, T. Van Achteren, and T. Omnes, *Data Access and Storage Management for Embedded Programmable Processors*. Norwell, MA: Kluwer, 2002.

[20] P. R. Panda, N. Dutt, and A. Nicolau, *Memory Issues in Embedded Systems-on-Chip: Optimizations and Exploration*. Norwell, MA: Kluwer, 1998.

[21] Agility. Agility MATLAB to C (2009) [Online]. Available: http://www.agilityds.com/products/matlab_based_products/default.aspx

[22] J.-Y. Mignolet, R. Baert, T. J. Ashby, P. Avasare, H.-O. Jang, J. C. Son, "MPA: Parallelizing an application onto a multicore platform made easy," *IEEE Micro*, vol. 29, no. 3, pp. 31–39, May/June, 2009 [Online]. Available: http://www.computer.org/portal/web/csdl/doi/10.1109/MM.2009.46

[23] M. Palkovic, A. Folens, H. Cappelle, M. Glassee, and L.Van der Perre, "Optimization and parallelization of 40 mhz MIMO SDM-OFDM baseband processing to achieve real-time throughput behaviour," in *Proc. ICT-MobileSummit 2009 Conf.*, Santander, Spain, June 2009.

[24] D. Iancu, H. Ye, J. Glossner, A. Iancua, and J. Takala, "Software only implementation of DVB-H," in *Proc. SPIE Multimedia on Mobile Devices*, vol. 6821. San Jose, CA, Jan. 2008, p. 68210F.

[25] D. Iancu, H. Ye, M. Senthilvelana, V. Kotlyar, J. Glossner, S. Agrawal, S. Jinturkar, A. Iancua, G. Nacer, S. Stanleya, M. Sima, and J. Takala, "Hand held analog television over WiMAX executed in SW," in *Proc. SPIE Multimedia on Mobile Devices*, vol. 6507. San Jose, CA,2007, p. 650709.

[26] M. Pelcat, S. Aridhi, and J. F. Nezan, "Optimization of automatically generated multi-core code for the LTE RACH-PD algorithm," in *Proc. Design and Architectures for Signal and Image Processing Conf. (DASIP)*, Bruxelles, Belgium, Nov. 2008, pp. 69–76 [Online]. Available http://www-labsticc.univ-ubs.fr/~gogniat/DASIP2008/proceedingDASIP2008.pdf

[27] Y. Jiang, W. Xu, and C. Grassmann, "Implementing a DVB-T/H receiver on a software-defined radio platform," *Int. J. Digital Multimedia Broadcast*, vol. 2009.

[28] C. Grassmann, M. Richter, and M. Sauermann, "Mapping the physical layer of radio standards to multiprocessor architectures," in *Proc. DATE*, Apr. 2007, pp. 1412–1417.

[29] Crescent Bay Software. Data Parallel C Extensions(DPCE) [Online]. Available: http://www.crescentbaysoftware.com/dpce/index.html

[30] Y. Lin, R. Mullenix, M. Woh, S. Mahlke, T. Mudge, A. Reid, and K. Flautner, "Spex: A programming language for software defined radio," 2008 [Online]. Available: http://data.memberclicks.com/site/sdf/SDR06-2.3-3.pdf

[31] Q. Zhang, A. B. J. Kokkeler, and G. J. M. Smit, "Cognitive radio design on an mpsoc reconfigurable platform," in *Proc. IEEE 2nd Int. Conf. Cognitive Radio Oriented Wireless Networks and Communications (CrownCom 2007), IEEE Communications Society*, Aug. 2007, pp. 187–191.

[32] L. Huang and B. Chapman, "Programming heterogeneous systems based on openMP," in *Proc. Int. Conf. Parallel Computing (ParCo'09)*, ENS, Lyon, France, Sept. 2009, submitted for publication.

[33] M. Li, B. Bougard, L. Van der Perre, and F. Catthoor, "Energy-aware algorithm and implementation of SDR oriented HSDPA chip-level equalizer," *J. Signal Process. Syst.*, vol. 56, no. 2–3, pp. 327–340, Sept. 2009.

[34] S. Parkvall, E. Dahlman, A. Furuskar, Y. Jading, M. Olsson, S. Wanstedt, and K. Zangi, "LTE-advanced evolving LTE towards IMT-advanced," *Proc. VTC*, pp. 1–5, Sept. 2008.

[35] P. F. Marshall, "Extending the reach of cognitive radio," *Proc. IEEE*, vol. 97, no. 4, pp. 612–625, Apr. 2009.

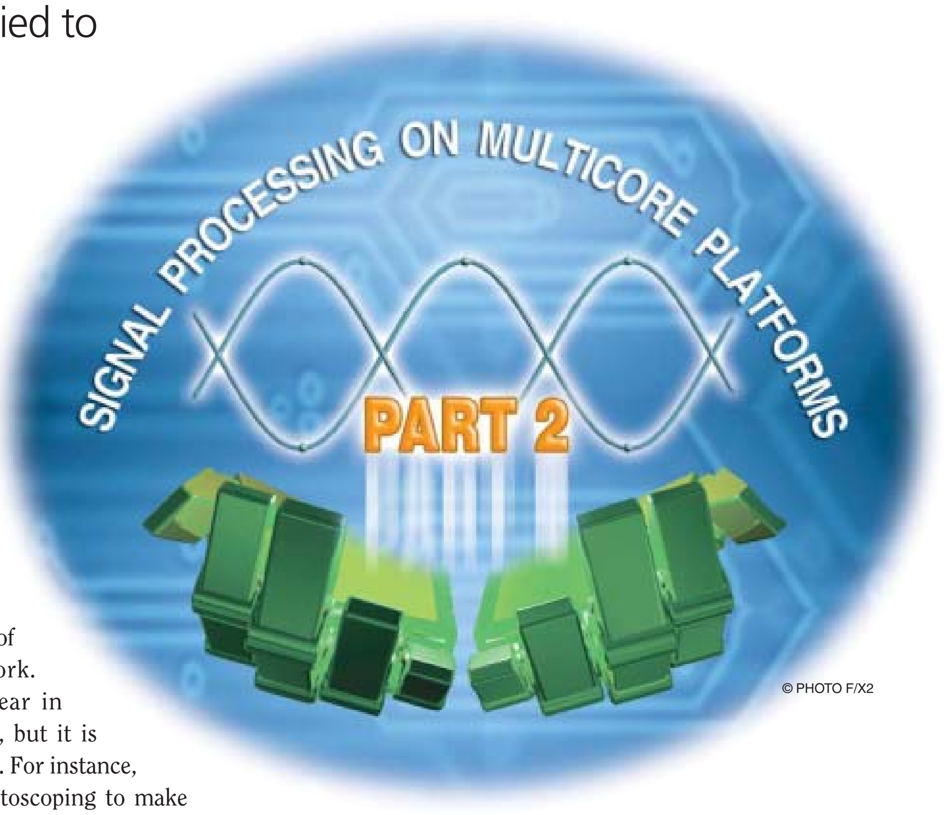
[36] *IEEE 802.22 Work Group on WRANs (Wireless Regional Area Networks)*, IEEE 802.22, Feb. 2009.

[SP]

[Richard Rzeszutek, Thomas F. El-Maraghi, Dimitrios Androutsos, and Samuel Zhou]

An Advantageous Rotoscoping Method

[Demonstrating how multicore processors can be applied to practical applications]



© PHOTO F/X2

Rotoscoping is a very old, complex, and time-consuming post-processing technique used by an animator to manually produce segmentation masks for a video sequence. Specifically, it is the act of tracing or outlining objects that appear in the frames. Doing so on a frame-by-frame basis means that a couple minutes of footage will require many hours of work. Tracing or outlining objects that appear in frames is done for a number of reasons, but it is generally to apply special effects to a scene. For instance, the 2006 film *A Scanner Darkly* uses rotoscoping to make the live action film appear animated. Rotoscoping can also be used to manually generate mattes, i.e., masks, in scenes where it may be impossible to use other methods, such as chromakey (colloquially known as "green screen"). Furthermore, most objects tend to have highly complex and irregular shapes (e.g., people, animals, and foliage). This makes it difficult for an artist to use tools such as Bézier curves to properly produce the required masks.

In this article, we present a rotoscoping method that takes advantage of ubiquitous multicore processors such as GPUs to assist an artist with the rotoscoping process. We have implemented this rotoscoping method as a plug-in to a commercial compositing application to demonstrate how multicore processors can be applied to practical applications.

Digital Object Identifier 10.1109/MSP.2009.935392

INTRODUCTION

Recently, rotoscoping has been applied to the problem of converting a conventional image sequence into a stereoscopic image sequence [1] (also referred to as "two-dimensional (2-D) to three-dimensional (3-D) image conversion"). Rotoscoping is used to extract all of the key objects in the scene so that they can be manipulated to produce the left and right eye image pair. This particular application of rotoscoping is more difficult than

1053-5888/10/$26.00©2010IEEE

other applications since it requires a high degree of accuracy when describing the object boundaries. If the boundaries are not accurate, visual aberrations, such the background appearing to be the same depth as the foreground, can occur.

A KEY CONSIDERATION FOR ANY ROTOSCOPING ALGORITHM IS HOW IT TREATS THE CONTROL POINTS LAID DOWN BY THE ARTIST.

Traditional rotoscoping methods use parametric curves, such as Bézier curves [2], to describe the object boundaries rather than having the artist generate each matte from scratch. The curves themselves are described by control points, which are placed by the artist at specific locations on the object boundary. The shape of the curve between control points is specified by handles that represent the "in" and "out" tangents of the curve. Figure 1 demonstrates the basic setup of a parametric curve. Because only the control points are specified, the artist only has to manipulate those points rather than attempt to draw the entire matte manually.

Unfortunately, arbitrarily complex shapes are not easily described by parametric curves. Sharp corners (such as an N-sided polygon) require their own control points since each corner represents a discontinuity in the first derivative of the function describing the object's shape. Therefore, many control points are required to adequately describe such a boundary. This makes the process more difficult since the artist must now keep track of these points over many frames.

A number of methods have been proposed to assist with the rotoscoping process. Some methods actively modify the points laid down by the artist and use optical tracking methods to adjust the curve between keyframes [3]. Other methods [4] use "active contours" [5] to produce a time-varying mask based on the initial curve. The main disadvantages of these methods are that they are either difficult to correct or require an experienced user. Our proposed method [6] simply corrects the curve laid down by the animator and trusts their judgement on the locations of the control points.

ASSISTED ROTOSCOPING

Our assisted rotoscoping method is based on the random walks framework by Grady et al. [7], [8]. Certain properties of the algorithm make it very useful for rotoscoping applications. In particular, since it is the solution to a system of linear equations, it is highly amenable to parallel implementations. Furthermore, it is a locally operating algorithm in that unconnected pixels do not affect each other. Unfortunately, random walks is rather susceptible to noise, so we augment the algorithm with noise filtering to make it more robust [9].

RANDOM WALKS

Random walks treats an image as an undirected, N-connected lattice (grid) with the adjacency matrix $\mathbf{A}$. Each edge in the graph, G_{ij} is weighted by the function

$$G_{ij} = \frac{2}{1 + \exp\{\beta d_{ij}\}}, \tag{1}$$

where d_{ij} is the normalized Euclidean distance between two color vectors. The distance function is normalized so that $0 \le d_{ij} \le 1$, to make the algorithm invariant to the data being processed. Therefore, the operation of the algorithm is not dependent on the type of data being sent to it.

As stated in [7], the random walks algorithm is simply the solution to the linear system

$$\mathbf{L}_u \vec{x} = \vec{b}, \tag{2}$$

where $\mathbf{L} = \deg(\mathbf{A}) - \mathbf{A}$, i.e., the Laplacian matrix of the image graph and $\mathbf{L}_u$ is the submatrix for just the unknown nodes. The vector, $\vec{x}$, is the vector of unknown values and $\vec{b}$ is the boundary vector. The boundary vector is defined as

$$\vec{b} = -\mathbf{L}_b \vec{s}, \tag{3}$$

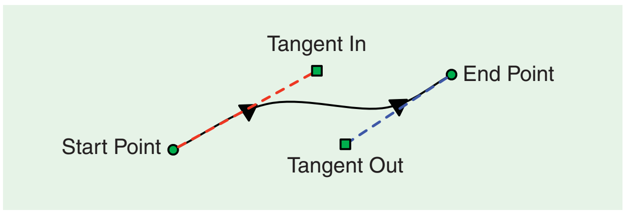
where $\vec{s}$ is the vector containing the initial (i.e., boundary) values of known nodes. Once the linear system has been solved, the resulting vector $\vec{x}$ contains the likelihood of each pixel being a member of the foreground or background. We refer to this as a potential map, denoted by $P[x, y]$. Please refer to [7] and [8] for a full derivation of the algorithm.

EXTENSION INTO SCALE SPACE

To improve the performance of the algorithm in noisy conditions, we extend the algorithm so that it operates on a scale space. Scale space is a form of multiresolution signal analysis where the signal is filtered through a series of isometric Gaussian kernels [10]. This allows the analysis of the image across multiple scales so that structures of varying size can be examined.

A useful property of scale space is that it preserves overall structure across multiple scales. Therefore, it becomes possible to use scale space to preserve overall image structure while also filtering out noise. We do this by generating a N_S-scale scale space, $\mathcal{S}$, for some set of scales, σ, and linking them in a 3-D structure as shown in Figure 2.

Random walks is then applied to this 3-D structure, producing a potential map for each scale in the scale space. This potential scale space, $\mathcal{P}$, is reduced to a single potential map through the use of a geometric average such that



[FIG1] An example of a parametric curve. The arrows indicate the curve direction while the dashed lines indicate the tangents at the control points.

$$P[x, y] = \left(\prod_{i=0}^{N_S-1} P[x, y|i] \right)^{1/N_S}, \qquad (4)$$

where $P[x, y|k]$ is the potential map at scale k. We refer to this augmented algorithm as scale-space random walks (SSRW).

ROTOSCOPE LABELING

A key consideration for any rotoscoping algorithm is how it treats the control points laid down by the artist. Our method respects these points by generating a labeling in such a way that the location of curve is free to vary but the points may not. Figure 3 shows how this labeling is constructed.

It should be noted that while the presented labeling assumes a linear interpolation between the two control points, this does not, in fact, have to be the case. If the curve is not linear then the unknown region simply follows curve. This allows the artist not to have to use an arbitrarily large unknown region for a curved shape.

Consider the trivial example presented in Figure 4(a). Here, the control points are not on the boundary so the estimated curve at those points is erroneous but the remainder of the curve is not. Figure 4(b) shows the result when the control points are on the boundary. As expected, there is no error in the boundary.

The returned potential maps, shown in Figure 5, show how this labeling produces these results.

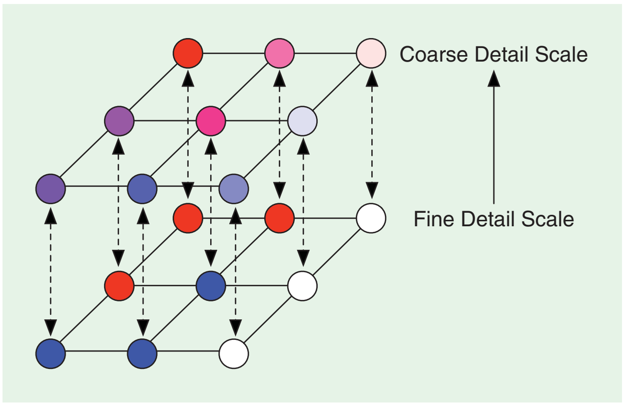
In effect, this labeling acts very much like the conventional trimap used in image segmentation and alpha matting algorithms. The difference, however, is that control points are respected due to the "pinching" at those points.

In cases where an image has been corrupted by noise, we use SSRW rather than random walks. Noise may result from either limitations in the technology used to capture the data, such as the low-cost complimentary metal-oxide-semiconductor sensors inside of cell phone cameras, or being added intentionally for artistic effect, such as film grain. Figure 6 shows how SSRW can be used to significantly improve the segmentation quality on an image corrupted by Gaussian noise with a variance of 0.01 and a mean of zero.

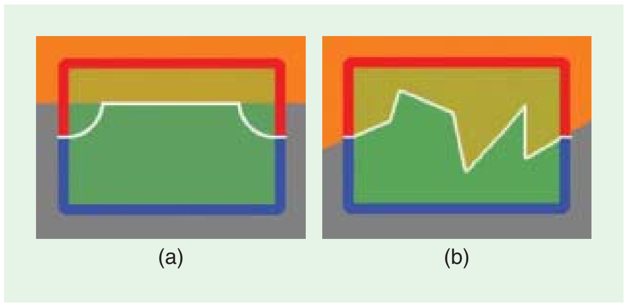
[A USEFUL PROPERTY OF SCALE SPACE IS THAT IT PRESERVES OVERALL STRUCTURE ACROSS MULTIPLE SCALES.]

Applying this labeling to an entire path (i.e., a collection of points and their connecting curves) is trivial. Each curve in the path can be treated independently of every other curve and, as such, the labeling is simply applied to each curve. Figure 7 shows how this is achieved.

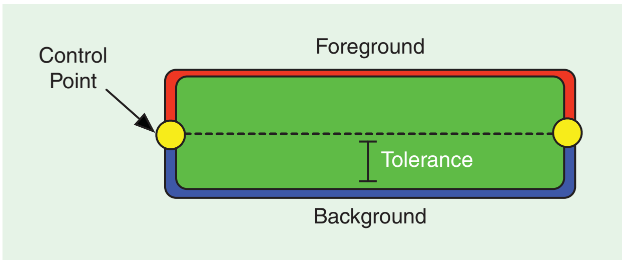
The purpose of this labeling scheme is primarily for ease of use. The labeling operates on the path laid down by the rotoscoping artist. Therefore, the artist does not require any specialized training to use this assisted rotoscoping method. In fact, the artist merely has to be familiar with the application used for the rotoscoping. The assisted rotoscoping method can then be implemented as a plug-in inside of that application.



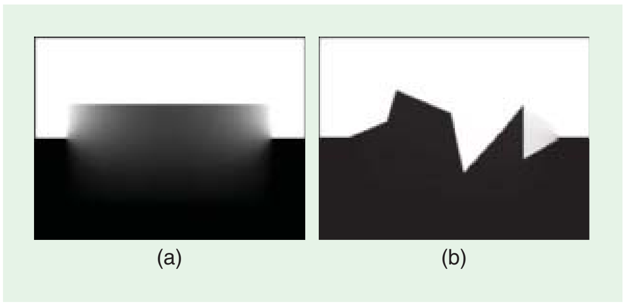
[FIG2] Scale-space graph structure.



[FIG4] Segmentation results for accurate and inaccurate control points: (a) erroneous control points and (b) accurate control points.



[FIG3] Labeling used for assisted rotoscoping.



[FIG5] Potential map for the segmentations shown in Figure 4: (a) erroneous control points and (b) accurate control points.

The fact that this rotoscoping method can be easily implemented as a plug-in should not be downplayed. Many of the tools used by rotoscoping artists, such as keyframing, curve creation/editing, and so forth, are nontrivial to implement. Commercial compositing applications provide much of this functionality for "free," making them desirable as part of an implementation.

MORE COMPLEX CALCULATIONS, SUCH AS MATRIX-VECTOR AND MATRIX-MATRIX MULTIPLICATION CAN ALSO BE SPED UP IN A MANNER SIMILAR TO A PARALLEL REDUCTION.

GPU IMPLEMENTATION

A useful property of linear algebra is that many calculations can be done in parallel. A canonical example is the dot-product between two D-dimensional vectors $\vec{x}$ and $\vec{y}$, defined as

$$\vec{x} \cdot \vec{y} = \sum_{i=0}^{D-1} x_i y_i. \quad (5)$$

Computing a dot-product requires $D(D-1)$ operations making it $O(D^2)$ in time on a serial processor where each operation has be done sequentially. However, on a multicore processor [such as a graphics processing unit (GPU)], operations can be done in parallel. Therefore, the dot-product is decomposed into two stages.

First, the vectors are multiplied in a pair-wise fashion to produce an intermediary vector, $\vec{z}$, such that

$$z_i = x_i y_i. \quad (6)$$

Because the multiplication is pairwise, each element can be computed in parallel. To find the value of the dot-product, $\vec{z}$ is "reduced" by summing pairs of elements to produce a new vector, $\vec{z}_{1/2}$, half the length of $\vec{z}$. This is repeated until the resulting vector is of length one (Figure 8). This procedure, known as a parallel reduction [11], is an $O(\lg D)$ operation.

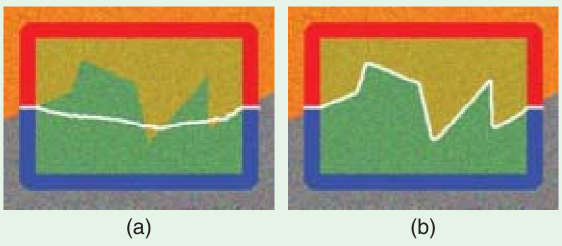
Note that even though the same number of operations are being performed, the time complexity has dropped significantly due to the parallel nature of the processor being used. This makes parallel processors an attractive option for implementing these types of calculations. The GPU is simply one type of parallel processor and this reasoning can easily be extended to field-programmable gate arrays, multicore central processing units (CPUs), or any other multiprocessor architecture.

More complex calculations, such as matrix-vector and matrix-matrix multiplication, can also be sped up in a manner similar to a parallel reduction. The key is identifying what operations are independent and which ones are dependent. Even very complex operations, such as solving a linear system can be broken down into smaller, simpler operations that can be implemented in a parallel fashion.

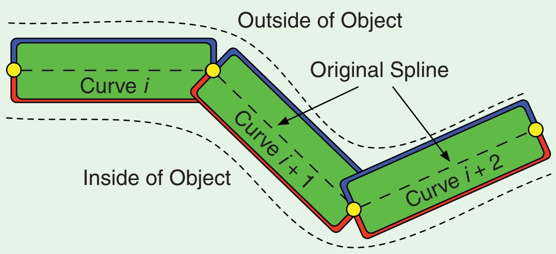
LINEAR SYSTEM SOLVER

While there are a number of different methods for solving a linear system, we chose to use the conjugate gradient method (CGM) [12] for a number of reasons. First, it has relatively low memory requirements. This is important when considering a GPU implementation since the memory requirements are quite strict. Second, it has seen an early GPU implementation [13], where 3-D graphic libraries were used rather than a dedicated application programming interface such as Compute Unified Device Architecture (CUDA). Finally, it has already been used in a previous random walks implementation [7]. For these reasons, the CGM was a natural choice as the system solver.

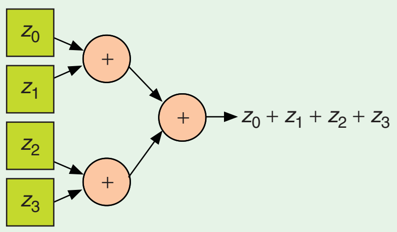
The CGM itself is composed of vector-vector additions, dot-products, and matrix-vector multiplications. The algorithm is iterative, meaning that the solution is incrementally



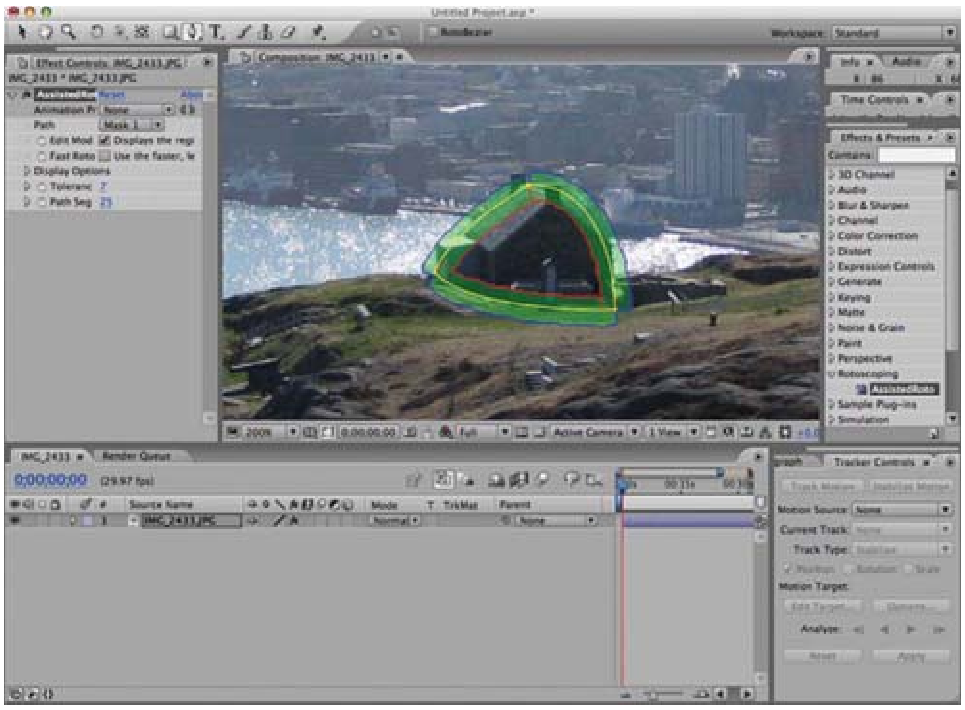
[FIG6] Segmentation of a noisy image using random walks and SSRWs: (a) random walks and (b) SSRWs.



[FIG7] Application of the labeling shown in Figure 3 to an entire path.



[FIG8] Example of parallel reduction.


[FIG9] Screen capture of the plug-in.

After Effects was able to handle the generation of the rotoscoping paths and the animation of the control points. It also presented an efficient way to read and write image data. Figure 9 shows a screen capture of the plug-in operating in After Effects.

The plug-in is designed to present a natural interface to the artist. The artist can lay down their curves without having to be specifically concerned with the plug-in itself. Once the plug-in is loaded, the artist is presented with an overlay that displays the width of the unknown region [Figure 10(a)]. By using the provided controls, the artist is able to adjust the width before generating the final mask [Figure 10(b)]. While not shown, it is possible for the artist to also view the probabilities that generated the particular segmentation.

All of the back-end processing (i.e., solving the SSRW system) was all done through Nvidia's CUDA library. We chose this library since it is well supported and CUDA-enabled graphics cards are common. This was a pragmatic choice and does not preclude the use of other libraries, such as the cross-platform OpenCL. A future implementation may use OpenCL since it is designed to take advantage of any multicore hardware, be it CPU or GPU.

For implementation reasons, we developed our own linear algebra routines but libraries have been developed so a user does not have to start programming from scratch. For example, the MAGMA project [14] is a dense linear algebra solver that is being developed to operate across multiple architectures while CULA [15] is a linear algebra library developed specifically for CUDA. As multicore processors become more prevalent, more and more libraries will be developed.

The actual performance is heavily dependent on the length of the curve and the hardware being used. Intuitively, the longer the curve, the more nodes that need to be solved for. Similarly, more powerful GPUs contain more processing cores, allowing for more elements to be processed in parallel. As a result, the individual calculations are faster and the algorithm converges more quickly.

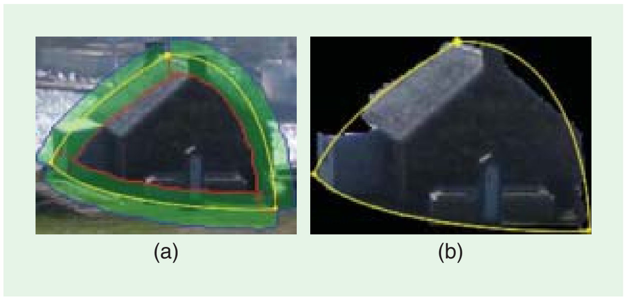
Currently, the plug-in processes each curve in the path sequentially. For example, if the original mask is composed of five curves and if each curve takes a relatively quick 250 ms to process, then processing the entire path (all five splines) will take 1.25 s. The majority of the delay in user interaction actually results from this implementation choice. Therefore, even if the

updated until some error term has been minimized. When performing the CGM, there are three basic stages: compute the solution, calculate the error in the solution, and determine how best to update the solution. Because these three stages are sequential, the entire CGM cannot be implemented in a parallel fashion.

However, because the CGM is composed of easily parallelized vector operations, it can still benefit from a GPU implementation. What a parallel implementation will do is speed up the individual operations and not the CGM itself. In other words, given the same system, a serial processor and a multicore processor will still take the same number of iterations to complete. However, the parallel version will complete those iterations faster since the individual operations themselves are executing much more quickly.

APPLICATION PLUG-IN

The assisted rotoscoping method was implemented as a plug-in to Adobe After Effects, a popular compositing application.



[FIG10] Plugin operating modes: (a) edit model and (b) mask model.

average computation time per segment is small, the overall computation time increases linearly with the number of curves in the path. This delay is very noticeable and can dramatically slow down a user's workflow.

THE FACT THAT THIS ROTOSCOPING METHOD CAN BE EASILY IMPLEMENTED AS A PLUG-IN SHOULD NOT BE DOWNPLAYED.

However, by recognizing each curve is independent from every other curve, all of the linear systems for each of the N curves, $L_1, L_2, \ldots, L_N$, can be combined into one "super-system," $L^{'}$, such that

$$L^{'} = \begin{bmatrix} L_1 & \cdots & 0 \\ \vdots & L_i & \vdots \\ 0 & \cdots & L_N \end{bmatrix}. \quad (7)$$

This system can then be solved using the CGM as before. The difference is now that all of the curves (i.e., the entire path) will be computed in parallel, providing a significant speedup. We intend to apply this optimization to future implementations of the plug-in.

CONCLUSIONS

We have shown how a multicore processor can be used in a practical application that is transparent to the user. Through CUDA, we used the processing power contained in a GPU to accelerate an assisted rotoscoping algorithm. Furthermore, using a GPU means that no special hardware is required by the end user. Because GPUs are present in most commercially sold computers, leveraging their computational power is highly desirable.

ACKNOWLEDGMENTS

We would like to gratefully acknowledge the contributions of IMAX and the Ontario Centres of Excellence who, without their support and funding, this research would not have been possible.

AUTHORS

Richard Rzeszutek (rrzeszut@ee.ryerson.ca) received his B.Eng. degree in computer engineering and his M.A.Sc. degree in electrical and computer engineering from Ryerson University, Toronto, Ontario, in 2007 and 2009, respectively. His research interests include GPU processing and its applications to multimedia, image segmentation, and image and video processing. He is currently a Ph.D. student at Ryerson University.

Thomas F. El-Maraghi (thomas.elmaraghi@gmail.com) received his B.Sc. degree in electrical and computer engineering in 1994, and his M.Sc. degree in computer science in 1996, both from Queen's University in Kingston, Ontario. In 2003, he received his Ph.D. degree from the University of Toronto in computer science. He completed a postdoctoral fellowship at Ryerson University in Toronto, Ontario. His research interests are in the fields of computer vision and graphics, with a particular emphasis on 3-D visualization, segmentation, and tracking. He currently works for the 3-D technology company Spatial View in Toronto, Ontario.

Dimitrios Androutsos (dimitri@ee.ryerson.ca) received his B.A.Sc., M.A.Sc., and Ph.D. degrees from the University of Toronto, Canada in 1992, 1994, and 1999, respectively, all in electrical and computer engineering. His research interests are in the areas of image and video processing, 3-D digital cinema, multimedia archiving and retrieval, distributed multimedia databases, image and video compression, object segmentation, object tracking, image enhancement, and image filtering. He is currently an associate professor at Ryerson University, Canada, and chair of the Department of Electrical and Computer Engineering. He is a Senior Member of the IEEE.

Samuel Zhou (szhou@imax.com) received the B.Eng. degree from Beijing Jiao Tong University, the M.Eng degree from Shanghai Jiao Tong University, and the Ph.D. degree from the University of Toronto. He has been with IMAX since 1994. He has a number of patents in the fields of digital cinema and image processing, either granted or pending. Currently, he is the vice president of image technology at IMAX.

REFERENCES

[1] S. Zhou, P. Judkins, and P. Ye, "Methods and systems for converting 2d motion pictures for stereoscopic 3-D exhibition," U.S. Pat. 20090116732, 2009.

[2] P. Bézier and P. Nicolau, *Emploi des Machines à Commande Numérique*. Paris: Masson, 1970.

[3] A. Agarwala, A. Hertzmann, D. H. Salesin, and S. M. Seitz, "Keyframe-based tracking for rotoscoping and animation," in *Proc. Int. Conf. Computer Graphics and Interactive Techniques*. New York: ACM Press, 2004, pp. 584–591.

[4] A. Agarwala, "SnakeToonz: A semi-automatic approach to creating Cel animation from video," in *Proc. 2nd Int. Symp. Non-Photorealistic Animation and Rendering*. New York: ACM Press, 2002, pp. 139–146.

[5] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *Int. J. Comput. Vis.*, vol. 1, no. 4, pp. 321–331, 1988.

[6] R. Rzeszutek, T. El-Maraghi, and D. Androutsos, "Interactive rotoscoping through scale-space random walks," in *Proc. IEEE Int. Conf. Multimedia and Expo, 2009 (ICME 2009)*, June 28, 2009–July 3, 2009, pp. 1334–1337.

[7] L. Grady, T. Schiwietz, S. Aharon, and R. Westermann, "Random walks for interactive alpha-matting," in *Proc. Int. Conf. Visualization Imaging, and Image Processing (VIIP05)*, 2005, pp. 423–429.

[8] L. Grady, "Random walks for image segmentation," *IEEE Trans. Pattern Anal. Machine Intell.*, 2006, pp. 1768–1783.

[9] R. Rzeszutek, T. El-Maraghi, and D. Androutsos, "Image segmentation using scale-space random walks," in *Proc. 16th Int. Conf. Digital Signal Processing*, July 2009, pp. 1–4.

[10] A. P. Witkin, "Scale-space filtering," in *Readings in Computer Vision: Issues, Problems, Principles, and Paradigms*, M.A. Fischler and O. Firschein, Eds. San Francisco: Morgan Kaufmann, 1987, pp. 329–332.

[11] M. J. Harris.(2007). *Optimizing parallel reduction in CUDA* [Online]. Available: http://developer.download.nvidia.com/compute/cuda/1_1/Website/projects/reduction/doc/red

[12] M. R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *J. Res. Nat. Bur. Stand.*, vol. 49, no. 6, pp. 409–436, 1952.

[13] J. Kruger and R. Westermann, "A GPU framework for solving systems of linear equations," *GPU Gems*, vol. 2, pp. 703–718, 2005.

[14] Magma: Matrix algebra on GPU and multicore architectures Web site [Online]. Available: http://icl.cs.utk.edu/magma/

[15] CULA Web site [Online]. Available: http://www.culatools.com

[SP]

[Siddharth Samsi, Vijay Gadepally, and Ashok Krishnamurthy]

MATLAB for Signal Processing on Multiprocessors and Multicores

[A review of three variations of multiprocessor parallel MATLAB]

© PHOTO F/X2

MATLAB is a popular choice for algorithm development in signal and image processing. While traditionally done using sequential MATLAB running on desktop systems, in recent years there has been a surge of interest in running MATLAB in parallel to take advantage of multiprocessor and multicore systems. In this article, we discuss three variations of multiprocessor parallel MATLAB, two of which are available as commercial, supported products. We also consider running MATLAB with key computations speeded up using multithreaded computations on multicore general-purpose graphical processing units (GPGPUs). Two signal processing kernels (fast Fourier transform (FFT) and convolution) and two full applications [synthetic aperture radar (SAR) imaging and superconducting quantum interference devices (SQIF)] are used to illustrate the use of parallel MATLAB.

Digital Object Identifier 10.1109/MSP.2009.935421

INTRODUCTION

Developments in microprocessor technologies have resulted in most processors having multiple computing cores in a single chip. As a result, today's distributed memory high-performance computers (HPCs) have multiple central processing units (CPUs) (2–4) in each node, with each CPU having multiple cores (2–8). The typical programming methodology for such distributed memory HPCs is using some form of a message passing paradigm, typically message passing interface (MPI). On the other hand, GPGPUs and graphics processing units (GPUs) are emerging as an alternative architecture for many computationally intensive tasks, including signal processing. GPGPUs have

1053-5888/10/$26.00©2010IEEE

large number of processor cores (up to 240 in some NVIDIA GPUs) and are typically programmed using threads. MATLAB is a popular choice for algorithm development in signal and image processing, and it has been traditionally used on desktop systems. Parallel MATLAB has been actively developed over the past several years, and there are several commercial and academic versions available [1]–[5]. Using MATLAB with GPGPUs is a relatively recent development, and the products are not as well developed. The options for multicore GPGPUs are the following: a) create and compile CUDA-based MATLAB executable (MEX) functions [6] or b) use MATLAB add-ons such as Jacket [7] or GPUmat [8], which aim to accelerate MATLAB functions. Signal processing algorithm developers who use MATLAB need to know the different options and tradeoffs to stay productive.

[MATLAB IS A POPULAR CHOICE FOR ALGORITHM DEVELOPMENT IN SIGNAL AND IMAGE PROCESSING.]

In this article, we walk the reader through the following different multiprocessor MATLAB choices:
- Parallel Computing Toolbox (PCT) and the MATLAB Distributed Computing Server (MDCS) [9]
- Star-P from Interactive Supercomputing Inc. [10]
- pMATLAB/bcMPI from MIT Lincoln Laboratories/Ohio Supercomputer Center (OSC) [11], [12].

We then look at different multicore MATLAB choices for a) CUDA-based MEX functions and b) MATLAB add-ons. For each of these technologies, we compare individual programming effort and performance improvements observed with popular signal processing kernels and applications. The main message for the reader is that it is possible to exploit today's multicore and multiprocessor systems to effectively simulate signal processing problems that are large in memory and/or computation requirements, while staying in the familiar MATLAB environment. The required changes to sequential MATLAB code are usually quite small and can be performed with ease. As the multicore and multiprocessor implementations reported in this article have been carried out on different systems and for different problem sizes, the results are not compared directly.

MULTITHREADING IN MATLAB

The simplest approach to leveraging multiple processor cores in MATLAB is through the use of multithreading. Since MATLAB supports multithreading natively [13], this approach is a simple, nonintrusive way to leverage multiple cores on a system. This type of multithreading can be broadly compared to the OpenMP [14], [15] approach to parallelism. The built-in multithreading in MATLAB does not require any intervention on the part of the user and is enabled by default. However, the maximum number of parallel threads cannot exceed the number of cores available on the system. The performance gain obtained by using multiple cores on a single system are also limited and vary based on the specific computation as well as the data size. Figure 1 illustrates this point. On a 16-core system, a maximum speedup of slightly over seven was seen for the multiplication and `sqrt` operations. Conversely, the trigonometric function `sin()` has a speedup of slightly under three. This test was performed on a four-socket quad core AMD Opteron-based system with 64 GB of RAM running Red Hat Enterprise Linux.

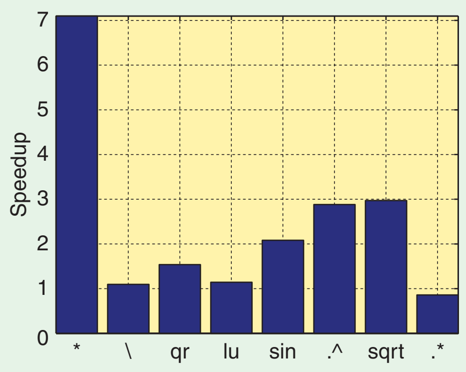
While multithreaded computations are the easiest entry into parallel computing with MATLAB, performance gains are usually limited. This approach should only be viewed as a first step in improving the code performance.

MULTIPROCESSOR MATLAB

The most common approach to overcoming the performance limitations of sequential MATLAB involves distributing an application over multiple nodes of a commodity computing cluster. Typical performance limitations for sequential MATLAB can be broadly classified into two areas: capacity and capability. The problem of capacity manifests itself as the inability for existing hardware and software to perform the desired computations in a practical amount of time. For example, this can include parameter sweeps that may take days or weeks that thus limits the range of analyses performed. Similarly, the data being collected may be so large that it is not feasible to analyze the complete data set in any reasonable manner. In these cases, while the existing hardware and software are capable of performing the desired analysis, it may not be practical to run the entire computation. The problem of capability is brought about by the actual physical limitations of the system. Thus, issues related to the total memory on a system or processor speeds may limit the amount of analysis performed. While this problem can be solved in a limited way by system upgrades, there is an upper limit to this approach that is dictated by technology and cost factors. In the case of such problems, the problem may be split up into smaller, more manageable chunks that can be performed in parallel.

PARALLEL COMPUTING TOOLS FOR MATLAB

There are several options for leveraging the availability of multiple processors and multiple cores to solve the performance limitations in serial MATLAB. These range from utilizing multiple



[FIG1] Relative speedup using 16 threads on a 16-core, four-socket system with the built-in multithreading in MATLAB.

cores on a single processor to leveraging hundreds of processors on a HPC-distributed memory cluster to split up the problem. Based on the type of analysis being done, one or more of the approaches may be ideally suited to the problem. In [1] and [5], the authors highlight various tools currently available for parallel computing in MATLAB. While these tools have been designed to minimize programming complexity, in our experience, three multiprocessor MATLAB technologies stand out in terms of user base, user support, and active development: pMATLAB+bcMPI, Star-P, and the PCT with the MDCS. These toolboxes enable users to parallelize algorithms in MATLAB using an embarrassingly parallel approach or through the use of distributed arrays/matrices and (with the exception of Star-P), implicit message passing between multiple MATLAB processes running on different processors.

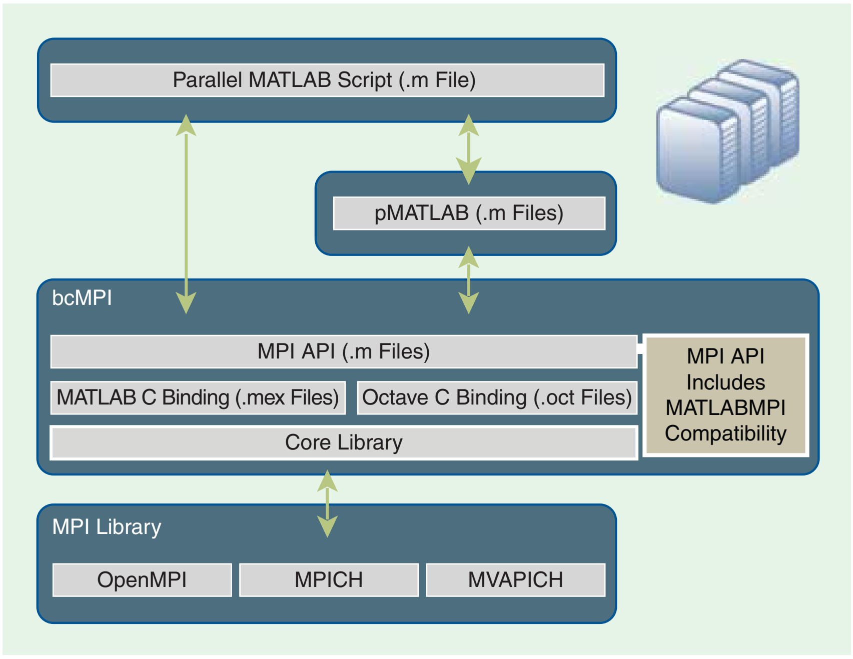
ON THE OTHER HAND, GPGPUs AND GRAPHICS PROCESSING UNITS ARE EMERGING AS AN ALTERNATIVE ARCHITECTURE FOR MANY COMPUTATIONALLY INTENSIVE TASKS, INCLUDING SIGNAL PROCESSING.

1) *pMATLAB+bcMPI:* bcMPI is an open-source software library that is developed by the OSC. bcMPI provides an alternative to MatlabMPI [16] and is geared towards large shared supercomputers. bcMPI interfaces with pMATLAB [17] from MIT Lincoln Laboratories, which supports distributed arrays. The combination of pMATLAB and bcMPI is denoted as pMATLAB+bcMPI. pMATLAB+bcMPI uses a layer of abstraction beyond traditional MPI calls and reduces programming complexity when compared to traditional MatlabMPI programs. Figure 2 shows the architecture of bcMPI.

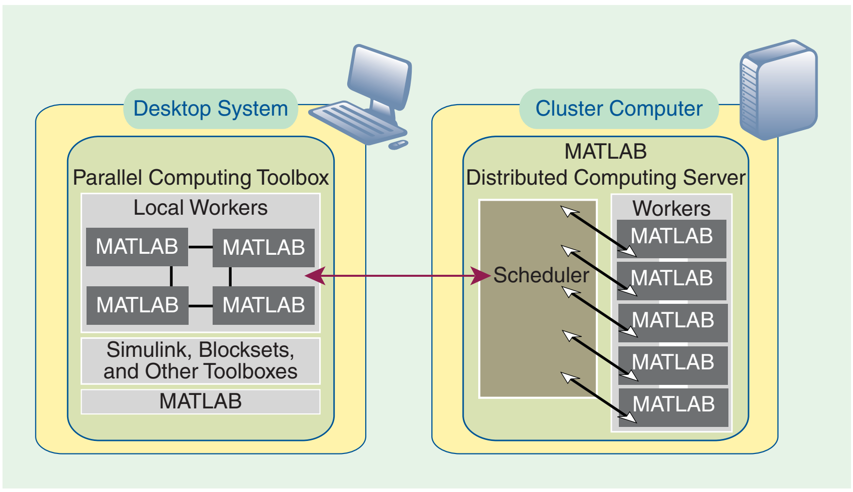
2) *PCT:* The PCT with the MDCS are commercial products offered by The MathWorks.

The PCT provides the ability to run up to eight MATLAB processes on a single system without the use of the MDCS. It thus provides a convenient environment to develop and test parallel MATLAB code locally and then scale up the same code to much larger scales on a large computing cluster through the use of the MDCS as shown in Figure 3.

3) *Star-P:* Star-P is a client-server parallel computing platform for MATLAB available from Interactive Supercomputing. Star-P supports fine-grained parallel as well as embarrassingly parallel modes of operation. However, Star-P does not provide functionality for explicit message passing between the processes running in parallel. Any required interprocessor communication is performed by the software itself without any intervention from the user. Figure 4 shows the architecture of Star-P.


Parallel MATLAB Script (.m File)
pMATLAB (.m Files)
bcMPI
MPI API (.m Files)
MATLAB C Binding (.mex Files)
Octave C Binding (.oct Files)
Core Library
MPI API Includes MATLABMPI Compatibility
MPI Library
OpenMPI
MPICH
MVAPICH

[FIG2] Architecture of pMATLAB/bcMPI from MIT Lincoln Laboratories and OSC.



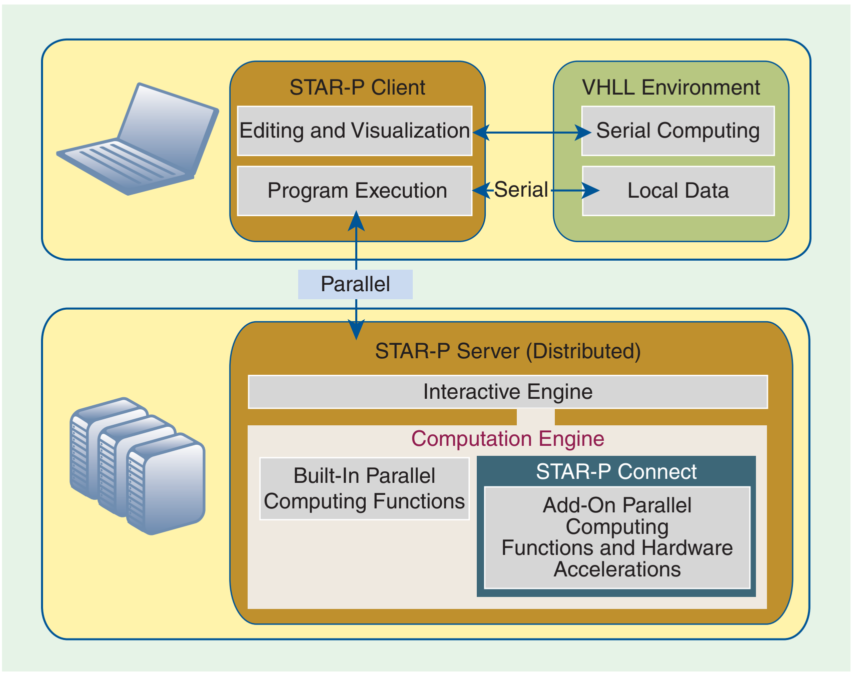
[FIG3] Architecture of PCT and MDCS from The MathWorks.

MULTIPROCESSOR APPROACHES

Parallel computing in MATLAB consists of splitting up the problem across multiple processors or multiple compute nodes in a variety of ways. This section discusses the different approaches used and illustrates each approach with an example.

EMBARRASSINGLY PARALLEL APPROACH

The embarrassingly parallel approach is quite common in practice and arises when a problem can be split into a number of independent tasks or computations that can be completed in an order-independent manner. For example, one may wish to analyze the effectiveness of an algorithm on a given data set by varying the parameters of the algorithm over a wide range. In such cases, each parameter set can be farmed out to a different processor, thus reducing the total time required to complete the entire analysis. Similarly, multiple, independent data sets stored in separate files can be analyzed in parallel by splitting up the work across multiple processors.


STAR-P Client
Editing and Visualization
Program Execution
VHLL Environment
Serial Computing
Local Data
Serial
Parallel
STAR-P Server (Distributed)
Interactive Engine
Computation Engine
Built-In Parallel Computing Functions
STAR-P Connect
Add-On Parallel Computing Functions and Hardware Accelerations

[FIG4] Architecture of the Star-P system from Interactive Supercomputing (recently acquired by Microsoft).

THE `parfor()` COMMAND

The PCT provides a simple way to parallelize MATLAB for-loops. The `parfor()` [9] command can be used to distribute the individual loop iterations across processors without any additional code modifications. This construct is suited for loops in which the computations are order independent. Let us consider the following simple algorithm for calculating π:

- Initialize a counter to zero.
- Generate two independent random numbers, x and y that are uniformly distributed between zero and one.
- If the point (x, y) lies inside the unit circle, increment counter.
- Repeat above two steps N time, where N is some very large number .
- Calculate π using the formula – $\pi = (4*N)/\text{count}$.

The above algorithm can be written in MATLAB as shown in Table 1.

[TABLE 1] CALCULATING π: SEQUENTIAL AND PARALLEL IMPLEMENTATION USING `PARFOR()`.

```
count = 0;
for k = 1:N
  p = rand(1, 2);
  if sqrt(sum(p.^2)
< 1
  count = count+1;
  end
end
pival = 4*count/N;
```
	```
	count = 0;
	parfor k = 1:N
	  p = rand(1, 2);
	  if sqrt(sum(p.^2)
	< 1
	  count = count + 1;
	  end
	end
	pival = 4*count/N;
	```

This algorithm is embarrassingly parallel and can be easily parallelized through the use of the `parfor` function provided by the PCT. By simply changing the `for` to `parfor`, the algorithm can be run on multiple processors, assuming that some preconditions are met. For full details on using the `parfor` construct, readers are referred to the toolbox documentation and the work in [18]. In a similar manner, suppose one needs to run the same image analysis algorithm on a large set of images, a `for-loop` can be used to process all the files and can be parallelized by using the `parfor` command.

USING DISTRIBUTED ARRAYS

A second approach to parallel computing in MATLAB is through the use of distributed arrays/matrices. The concept of distributed arrays/matrices is based on the partitioned global address space (PGAS) programming model in which multiple processors share a global address space [18] and each processor can read to/write from any section of the global address space [19]. The data being processed is thus distributed across multiple processors, with parts of the data being local to each processor. This distribution of data also enables the use of large data structures that may not be practical on a single processor.

In a similar manner, the different parallel MATLAB techniques discussed here enable the user to create and manipulate arrays/matrices in MATLAB that are distributed across multiple processors or on a cluster of computers. For example, the PCT allows the creation of distributed arrays/matrices by concatenating matrices that reside on different processors, by distributing a large matrix that initially exists on a single processor, or by using custom constructors provided by the toolbox. Similarly, Star-P enables the distribution of matrices through the use of the `*p` construct. One of the biggest advantages of this approach to parallel computing is that the data distribution is handled by the underlying library. The programmer does not need to know where the data actually resides and can focus on the actual algorithm.

Table 2 shows an example of a parallel two-dimensional (2-D) convolution using pMATLAB+bcMPI. The first step in this approach is to create a map that defines the distribution of the data. In this example, the `map` function defined by pMATLAB is used to distribute the rows of the random matrix `data` across `Np` number of processors. Each processor then performs a convolution on the part of the data that resides in its local memory and puts the results back into the global address space. Figure 5 shows the reduction in the total compute time for a 2-D convolution kernel on a matrix of size 1,024 × 1,024.

RESEARCHERS CAN EXPLOIT MULTICORE AND MULTIPROCESSOR SYSTEMS TO SIMULATE SIGNAL PROCESSING PROBLEMS WITH LARGE MEMORY AND/OR COMPUTATION REQUIREMENTS FROM THE FAMILIAR MATLAB ENVIRONMENT.

Table 3 shows an example of a 2-D FFT operation on a distributed matrix using Star-P. The code also illustrates the ease with which Star-P can be used to create distributed arrays by using the "`*p`" construct. An N × N distributed matrix is created using the MATLAB function `rand` and the FFT can be calculated by simply calling the overloaded `fft2()` function.

Figure 6 shows the run times for a parallel 2-D FFT using Star-P for varying data sizes. The parallel algorithm was run on four systems each having a four-core AMD Opteron processor, for a total of 16 cores. It can be seen that for small problem sizes, the parallel implementation is actually slower. This is because of the large amount of interprocessor communication that has to occur when the matrix is transposed. This also illustrates one of the pitfalls that users must be aware of when using distributed matrices. Algorithms must be designed so as to minimize redistribution of data that can lead to a reduction in performance due to excessive communication between processors.

FINE-GRAINED PARALLELISM

The third approach to parallelism involves the use of message passing similar to the traditional parallel programming paradigm. Programmers can control algorithm flow, exchange data between different instances of MATLAB running on different processors, and distribute the analysis through explicit message passing between the MATLAB processes. This approach gives the programmer maximum control over the parallel implementation of the algorithm, but it can be most time consuming to develop and test.

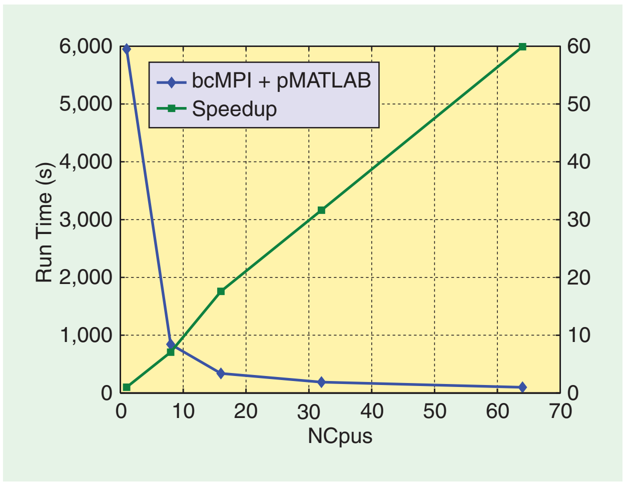
The approach to fine-grained parallelism leverages the MPI programming paradigm. The MPI standard [20] defines the language bindings for point-to-point message passing, collective communication, process creation and management and several other protocols required for the message passing parallel programming model [21]. bcMPI and the PCT offer MPI bindings for MATLAB. These bindings include the basic MPI functions

[TABLE 2] PARALLELISM USING DISTRIBUTED ARRAYS: PARALLEL 2-D CONVOLUTION USING pMATLAB+bcMPI.

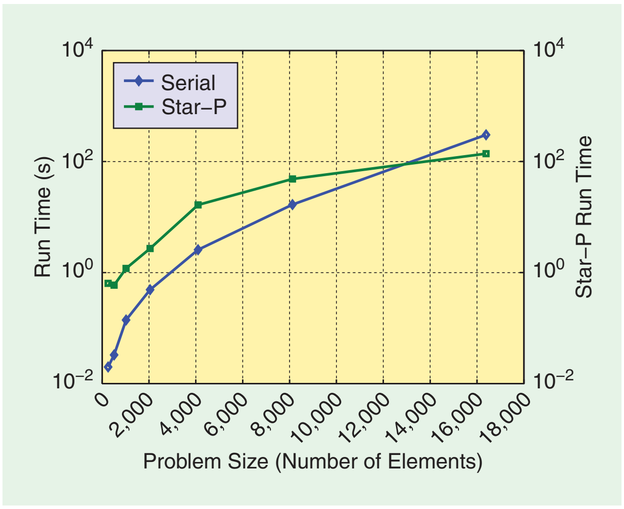
```
dist_map = map([Np 1], {}, [0:Np-1]);
data = rand(4000, dist_map);
locData = local(data);
locData = conv2(locData, H, 'same');
data = put_local(data, locData);
```

[TABLE 3] PARALLELISM USING DISTRIBUTED ARRAYS: PARALLEL 2-D FFT USING STAR-P.

```
N = 1024;
x = rand(N, N*p);
X_fft = fft2(x);
```



[FIG5] Multiprocessor versus sequential run time for 2-D convolution: Performance of parallel 2-D convolution on multiprocessor system using pMATLAB+bcMPI.



[FIG6] Multiprocessor versus sequential run time for 2-D FFT: Run-time curves obtained by varying problem size and parallelized using Star-P.

that enable point-to-point communication between the MATLAB processes running in parallel.

Table 4 shows an example of point to point communication between multiple processors. In this example, all processors with rank greater than zero send their local data to the rank zero processor using the `MPI_Send` command, which is then received by the rank zero processor when it calls the `MPI_Recv` command. It should be noted here that deadlocks can occur if a processor sends data without a corresponding `MPI_Recv` from the intended recipient. This is one of the major aspects of fine-grained parallelism that programmers must be careful to address.

TYPICAL PERFORMANCE LIMITATIONS FOR SEQUENTIAL MATLAB CAN BE BROADLY CLASSIFIED INTO TWO AREAS: CAPACITY AND CAPABILITY.

MATLAB ON GPGPUs

Another technique for speeding up sequential MATLAB code involves using the multiple cores of CPUs and/or GPGPUs for multithreaded computing. The main difference between this form of parallel MATLAB and multiprocessor MATLAB is that multicore MATLAB uses threading as the underlying parallel computing mechanism. Currently, there are two examples of multicore architectures: conventional multicore CPUs (typically with two eight-cores) and unconventional multicore processors such as GPGPUs (with tens or hundreds of cores). For the purpose of our discussion, we will concentrate on the utilization of multiple cores of GPGPUs. This form of parallel MATLAB is relatively new and the number of options available is limited.

GPUs

Recent trends in hardware development have led to GPUs evolving into highly parallel, multicore computing platforms. Current GPGPUs such as the Quadro FX 5600 have 128 cores and newer hardware such as the Tesla platform from NVIDIA can contain up to 240 processing cores per graphics card.

CUDA is a parallel programming model and software environment developed by NVIDIA that enables programmers to take advantage of the multicore GPGPU with standard programming languages [6]. CUDA provides extensions to the C programming language that enable the programmer to write fine-grained parallel algorithms that can be executed using multiple, simultaneous threads on the GPGPU. Recent work [22]–[24] has shown the performance gains possible through the use of CUDA to accelerate a variety of algorithms.

The CUDA programming model enables programmers to run fine-grained parallel code by launching multiple threads on the GPGPU. The threads are divided into blocks that can be scheduled to run independently across the GPGPU compute cores. The ability to schedule and run multiple threads simultaneously enables code scalability with the number of cores. Complete details on the CUDA programming model can be found in [6].

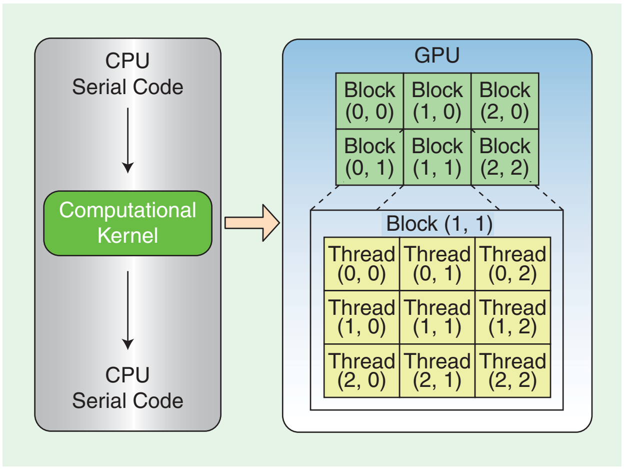
As shown in Figure 7, the serial code running on the CPU invokes a computational kernel that is to be run on the GPGPU. Since the CPU and GPGPU memory spaces are distinct from each other, data to be used in the computations must be transferred to the GPGPU. This can be the major penalty incurred in the process and programmers must avoid unnecessary data transfer between the CPU and the GPGPU to avoid the performance penalty. The computational kernel is executed on the GPGPU through the use of grids that are comprised of multiple thread blocks each of which executes on a single multiprocessor.

INTERFACING MATLAB WITH GPGPUs

Several toolboxes for MATLAB have been developed to allow the offloading of computations to the GPGPU by simply casting MATLAB data into the toolbox-defined GPGPU data type [7], [8], [25]. The availability of such toolboxes makes it very easy for researchers to try out GPGPU computing without having to write optimized C code that can take hours to develop and debug. Scientists can focus on the research without worrying about the intricacies of the C/CUDA programming paradigm. These toolboxes, however, are currently under development and may not support every MATLAB function. The most common functions supported include one-dimensional (1-D) and 2-D FFT, convolution, and standard mathematical operations.

[TABLE 4] AN EXAMPLE OF FINE-GRAINED PARALLELISM USING BCMPI.

```
my_cpu = MPI_Comm_Rank(comm););
if (my_cpu > 0)
 tag = my_cpu;
 MPI_Send(0, tag, comm, data);
else
 globalsum = 0;
 for k = 1:ncpu-1
 tag = k;
 data_k = MPI_Recv(k, tag, comm);
 globalsum = globalsum + data_k;
 end
end
```



[FIG7] The GPU architecture.

One approach to offloading computations to the GPGPU is to use the GPGPU to perform small kernels such as FFT, convolution, and FIR filtering, which are often the most time consuming operations in an application. With this approach, the researcher can remain in the familiar MATLAB environment while running massively parallel algorithms transparently on the GPGPU.

MATLAB PLUG-IN FOR CUDA

The MATLAB plug-in for CUDA available from NVIDIA's Web site [26] provides the tools necessary to convert CUDA programs to MATLAB-callable MEX functions. The use of this plug-in allows programmers to write custom applications that are optimized for the given problem. This can be a challenging and time consuming task due to the fact that the desired code must be written (often rewritten) in C.

MATLAB TOOLBOXES FOR GPGPU COMPUTING

Currently, three toolboxes are available for the use of CUDA from MATLAB. The toolboxes are GPGPUmat [8], gpulib [25], (both available for free), and Jacket [7], which is a commercial product. Each of these toolboxes offers the ability to offload computations to the GPGPU by simply casting MATLAB data types to a toolbox provided GPGPU data type. The simplicity of using these toolboxes should be considered carefully because the code can incur heavy penalties due to data transfers between the main CPU memory and the GPGPU memory.

CONSIDERATIONS FOR GPGPU COMPUTING

The use of GPGPUs for offloading computations brings additional considerations. The typical operating procedure for GPGPU computing consists of the transfer of data from the CPU to GPGPU memory when GPGPU functions are called. This transfer of data from the CPU to the GPGPU can lead to a performance penalty. When writing CUDA programs, the programmer has significant control over the data transfers and the CUDA application should be carefully designed to minimize such transfers. In contrast, the premise of the MATLAB toolboxes available for GPGPU computing is that users can accelerate their code simply by casting the data to the GPGPU data-type and performing calculations as usual. However, this must be done carefully so as to avoid the penalty incurred during data transfers. For example, in our third scalable synthetic compact application (SSCA #3) described below, the FFT kernel is called multiple times in the algorithm. Using the GPGPU to offload the FFT calculations is an obvious path to speeding up the application. However, it was observed that using the GPGPU did not provide the expected speedup. Upon closer examination and profiling of the code it was observed that the FFT kernel was indeed faster on the GPGPU but the performance penalty incurred in the data transfers negated the gains.

[TABLE 5] CODE SNIPPETS OF pMATLAB+bcMPI ADDITIONS FOR genSARimage().

SERIAL CODE	PARALLEL CODE
	PFMAP = MAP([1 NCPUS], {}, [0:NCPUS-1])
F = SINGLE(ZEROS(NX,M);	PF = ZEROS(NX,M,PFMAP);
	PFLOCAL = IFFT(PFLOCAL, [],2);
	PF = PUT_LOCAL(PF, PFLOCAL);
	Z = TRANSPOSE_GRID(PF);
	ZLOCAL = IFFT(LOCAL(Z), [],1);
SPATIAL = IFFT(IFFT(F, [], 2));	Z = PUT_LOCAL(Z,ZLOCAL);
	Z = AGG(Z);
	SPATIAL = ABS(Z)';

THE SSCA #3 APPLICATION

The SSCA #3 benchmark [27] from the Defense Advanced Research Projects Agency High Productivity Computing System Program [28], performs SAR processing. SAR processing creates a composite image of the ground from signals generated by a moving airborne radar platform. It is a computationally intense process, requiring image processing and extensive file I/O. The proposed solution to the SSCA #3 application uses a data parallel approach to parallelization. The SSCA #3 application consists of signal processing kernels such as FFTs, convolutions, and interpolation. To parallelize the SSCA #3 application, the MATLAB profiler ran on the serial implementation. The profiler showed that 67.5% of the time required for computation is spent in the image formation function of Kernel 1 (K1). Within `formImage`, the function `genSARimage` is responsible for the computationally intense task of creating the SAR image. `genSARimage` consists of two compute-intensive parts, namely, the interpolation loop and the 2-D inverse Fourier transform (IFT). In addition, multiple 1-D Fourier transforms are computed. The interpolation loop involves iteratively interpolating sections of the SAR raw image. The number of iterations is often in the tens of thousands, and each iteration contains multiple matrix multiplications and matrix additions. The 1-D and 2-D FFTs are carried out only once per `genSARimage` function call. The problem size (size of input image) can be increased by modifying the SCALE variable. For a SCALE value of ten, the time taken by K1 is approximately 200 s. Amdahl's law [29] states that the maximum speedup of a parallel application is inversely proportional to the percentage of time spent in sequential execution. In our parallelization of SSCA #3, the function genSARimage, which accounted for 67.5% of overall execution time, was parallelized. The remaining execution time (32.5%) remains serial and, therefore, the theoretical speedup on p cores is $1/(0.325 + (0.675/p))$. The maximum speedup possible is about 3.0.

1) *Multiprocessor Implementation:* In the multiprocessor implementation, a matrix F (which is interpolated to give the output image, see Table 5) is distributed as contiguous blocks of columns across all processors. The code within the interpolation loop remains functionally equivalent with the parallel version altered such that each processor performs its calculations on a smaller, local part of the global F matrix. After the interpolation loop, the 2-D IFFT is carried out through the usage of pMATLAB's transpose_grid operation that changes the distributed F matrix from a column to row distribution. A snippet of the required changes are shown in Table 5 (for

variables of interest). The absolute performance times and relative speedups for image formation are given in Figure 8.

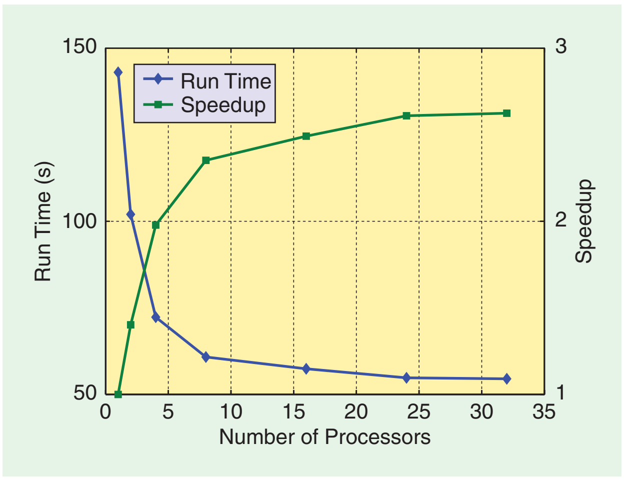
For this application, nearly 67.5% of the code was parallelized by increasing the number of source lines of code by just 5.5% (approximately 50 additional lines of code).

2) *GPGPU Implementation:* From the application analysis, it appears that using GPGPUs for these calculations would reduce overall computational time for SSCA #3. To investigate this, the GPUmat toolbox was used to port the serial genSARimage function code to GPGPU enabled MATLAB code. This porting was a simple process, and required casting variables and matrices as GPUsingle data types. This casting moves the data from CPU memory to GPGPU memory.

After enabling the GPGPU code, it was noticed that the overall execution time post-GPGPU porting was larger than the sequential (pre-GPGPU) run time. The MATLAB profiler was used to investigate this unexpected behavior. It was observed that as expected, functions such as FFTs, matrix multiplications, etc. showed a large reduction in computation time when using the GPGPUs. Results obtained before and after GPGPU porting for certain functions are listed in Table 6.

However, additional overhead due to communication between CPU and GPGPU was also observed. This overhead caused an increase in overall run time for the GPGPU enabled code. Examples of large overhead components are shown in Table 7.

Due to these additional components (which are not present in the pre-GPGPU code) extensive modifications would be required to efficiently use GPGPUs.



[FIG8] Multiprocessor results for SSCA#3 application: Speedup of SSCA#3 for a fixed scale (SCALE=6), and parallelized using pMATLAB+bcMPI.

[TABLE 6] RUN TIMES BEFORE AND AFTER GPGPU PORTING.

OPERATION	PRE-GPU TIME (S)	POST-GPU TIME (S)
INTERPOLATION	45.54	18.18
WINDOWING	24.91	11.18
2-D FFT	1.64	0.36

[TABLE 7] GPGPU DATA TRANSFER OVERHEAD.

OPERATION	OVERHEAD TIME (S)
DELETING GPU SINGLE	17.7
ASSIGNING GPU SINGLE	13.6
SUBSCRIPTING GPU SINGLE	10.2

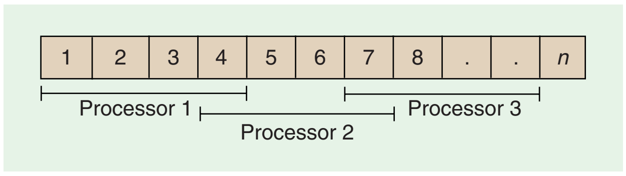
THE SQIF APPLICATION

Superconducting quantum interference devices (SQUIDs) and arrays of SQUIDs or SQIFs have a wide variety of applications [30]. SQUIDs are the world's most sensitive detectors of magnetic signals (sensitivity femto-Teslas) and are used for the detection and characterization of signals small enough to be virtually immeasurable by any other known sensor technology. They have applications in the detection of buried facilities from space, and the detection of weak signals in noise limited environments. The SQIF application is intended to solve large scale problems for the study and characterization of interference patterns, flux-to-voltage transfer functions, and parameter spread robustness for large SQIF loop size configurations and SQIF array fault tolerance. The technical background for the SQIF application can be found in [30]. The particular application developed was intended to run the SQIF program in an optimized fashion to either reduce run time and/or increase the size of the problem being solved. The SQIF application involves iteratively solving ordinary differential equations as outlined in [30]. Application of the MATLAB profiler on the SQIF application using 100 SQUIDs yielded a run time of approximately 20 min. A detailed analysis showed most (approximately 88%) of the time spent in the function evaluations for the differential equation. Optimization was carried out on this function. Further review of the profiler results showed a linear increase in the time taken by the code as the number of SQUIDs was increased. Results of the multiprocessor approach were obtained using pMATLAB+bcMPI on the OSC's AMD Opteron "Glenn" cluster.

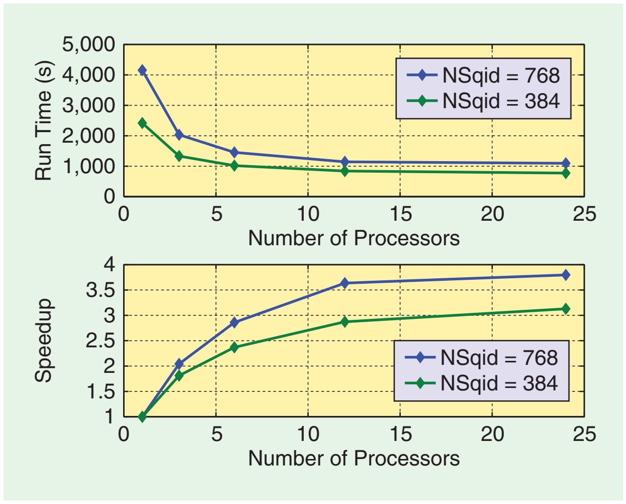
The SQIF application was also parallelized using CUDA and results are discussed in the following sections. Since the GPGPU computations are performed in single precision while the MATLAB computations are in double precision, direct comparisons between the two is not strictly valid. However, the performance numbers of each technology help illustrate the gains possible. The multicore GPGPU implementation requires a significant amount of programming effort for the large gains observed and this is a tradeoff that must be evaluated before choosing the technology to be used.

1) *Multiprocessor Implementation:* At its core, the SQIF application involves solving a partial differential equation over a desired time range. At each time step, flux is calculated for each SQUID element in a vector based on its adjacent neighbors. Thus, in a fine-grained parallel implementation, each processor needs only one data point from its left and right neighbors. Figure 9 shows the idea behind the fine-grained parallel

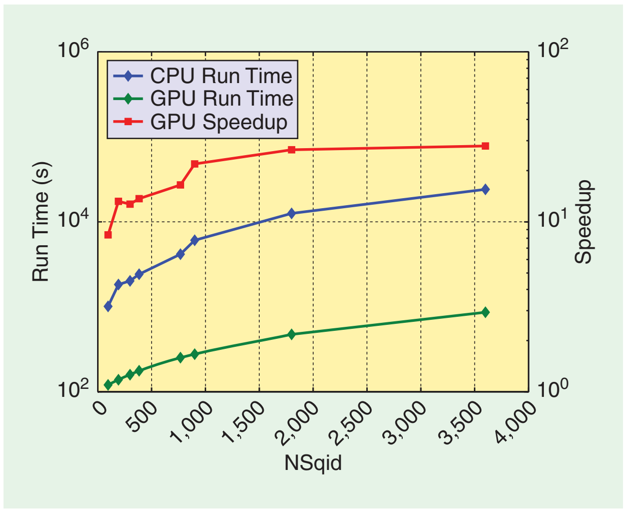
implementation in MATLAB. As shown in the figure, the input data of length n is distributed across P processors with an overlap of one data point with each neighbor. The first and last elements of the output data are special cases and are calculated separately.



[FIG9] **Parallelism strategy for SQIF application.**



[FIG10] **Multiprocessor run time for SQIF application: For varying NSquids and using bcMPI+pMATLAB.**



[FIG11] **Multicore run time for SQIF application: Run time and speedup curves for a GPGPU implementation parallelized using the CUDA MEX interface. Results are for varying NSquids. Also shown are the run times for a CPU-based parallelization using four cores.**

RECENT TRENDS IN HARDWARE DEVELOPMENT HAVE LED TO GPUs EVOLVING INTO HIGHLY PARALLEL, MULTICORE COMPUTING PLATFORMS.

At the beginning of the evaluation of the differential equations, input data is distributed across P processors with the required overlap. At each step of the differential equation evaluation, a small amount of data exchange occurs between processors. For example, as shown in Figure 9, Processor 2 receives data from Processor 1 and sends data to Processor 3. In the pMATLAB implementation, this communication must occur across different compute nodes of the cluster over the Infiniband network. While the Infiniband network offers bandwidths up to 10 GB/s, the communication overhead can add up as the number of processors is increased. The key to achieving a good speedup is to ensure that the computation/communication ratio is large. The time required for the SQIF application increases nonlinearly as the number of devices being simulated increases as shown in Figure 10.

2) *Multicore Implementation Using CUDA*: As described in the previous section, the SQIF application can see the most performance gains from a fine-grained parallel implementation of the algorithm. In this approach, while each processor performs most of its calculations independently, it needs to exchange data with at least one processor. The CUDA implementation of the SQIF application involved a translation of the MATLAB code into CUDA-enabled C code. In this implementation, hundreds of threads are launched on the GPGPU for performing the calculations. For a simple comparison, each thread can be considered as a single MATLAB process used in the pMATLAB implementation. The main difference here is that each thread only calculates a single data point in the output. Since the entire algorithm data also resides on the GPGPU, the overhead necessary to access the memory is significantly lower than the overhead in communicating over the network interface in the pMATLAB implementation. The CUDA implementation in this case was a simple implementation without any application specific optimizations. Even with this naïve implementation, a speedup of almost 28 was observed when simulating 3,600 devices. Figure 11 shows the run time and speedup achieved through the CUDA-based parallelization. The same plot also shows the run time for the CPU-based implementation on a single four-core AMD Opteron-based system. The GPGPU speedup observed is relative to the run time for the CPU-based serial implementation.

CONCLUSION

In this article, we have described how to speed up MATLAB code for signal processing kernels and applications using multiple processors and multicores. The examples illustrate when speedups in processing time can be expected, and we provided guidelines on how to proceed with the code parallelization. We provided code snippets to illustrate the additional programming required and compared the performance when ported to run on distributed memory HPC clusters and commodity GPGPUs. The two applications show that one can take advantage of parallelism attained by using multiple CPU cores as well as the GPGPU. We also examined the use of currently available toolboxes for running MATLAB code on the GPGPU without the need for reprogramming algorithms in CUDA. Researchers now have access to a variety of interesting computer architectures that can be leveraged for significant performance gains. No single approach may be optimal for all types of applications and the choice of the appropriate approach to parallelization should be made based on the user requirements.

ACKNOWLEDGMENTS

The authors would like to thank Thomas Kerwin and Brad Hittle at the Visualization Lab, OSC for their assistance with CUDA. We would also like to thank Yogesh Chidambarnathan from the Birla Institute of Technology and Sciences, Pilani, India, for his assistance in running the kernel tests.

We also gratefully acknowledge Dr. Fernando Escobar (SPAWAR Systems Center PACIFIC, Code 7113, San Diego, California), and Dr. Juan Carlos Chaves (HPTi Inc.) for providing the serial SQIF code. The original SQIF application code was written by Dr. Patrick Loghini (SPAWAR Systems Center PACIFIC, Code 7173, San Diego, California).

AUTHORS

Siddharth Samsi (samsi@osc.edu) received his bachelor's degree in electronics and telecommunications from the College of Engineering, Pune, India, and his master's degree in electrical engineering from The Ohio State University. He is currently a Ph.D. student in the Department of Electrical and Computer Engineering at The Ohio State University. He joined the OSC in 2006 and conducts research in parallel computing using high-level languages and image processing for medical applications.

Vijay Gadepally (vijayg@osc.edu) earned his bachelor's degree in electrical engineering from the Indian Institute of Technology, Kanpur. He is currently pursuing his Ph.D. degree in electrical and computer engineering at The Ohio State University. His research interests include signal/image processing, high-performance computing, and parallel algorithms.

Ashok Krishnamurthy (ashok@osc.edu) earned his bachelor's degree in electrical engineering from the Indian Institute of Technology in Madras, India, in 1979. He received his master's degree and doctorate in electrical engineering at the University of Florida in 1981 and 1983, respectively. He is interim codirector of the OSC and an associate professor in the Department of Electrical and Computer Engineering at The Ohio State University. He conducts research in signal/image processing, high-performance computing, parallel high-level language applications, and computational models of hearing.

REFERENCES

[1] A. Krishnamurthy, J. Nehrbass, J. Chaves, and S. Samsi, "Survey of parallel MATLAB techniques and applications to signal and image processing," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, vol. 4, 2007, pp. 1181–1184.

[2] A. Krishnamurthy, D. Hudak, J. Nehrbass, S. Samsi, and V. Gadepally, "Parallel MATLAB in production supercomputing with applications in signal and image processing," in *Proc. Conf. Parallel Processing for Scientific Computing*. Philadelphia, PA: SIAM, 2008.

[3] A. Krishnamurthy. (2007). *Parallel and distributed MATLAB applications in signal and image processing* [Online]. Available: http://www.icassp2007.org/SpecialSessions.asp

[4] D. Hudak, N. Ludban, V. Gadepally, and A. Krishnamurthy, "Developing a computational science IDE for HPC systems," in *Proc. 3rd Int. Workshop on Software Engineering for High Performance Computing Applications*. Washington, DC: IEEE Comput. Soc., 2007, p. 5.

[5] R. Choy and A. Edelman, "Parallel MATLAB: Doing it right," *Proc. IEEE*, vol. 93, no. 2, pp. 331–341, 2005.

[6] NVIDIA, *CUDA Programming Guide 2.0*. NVIDIA Corporation, 2008.

[7] AccelerEyes, "Jacket v 1.01," (2009). [Online]. Available: http://www.accelereyes.com

[8] The GP-you Group (2009). "GPUmat: GPU toolbox for MATLAB." [Online]. Available: http://gp-you.org/

[9] The MathWorks (2009). "Parallel computing toolbox—Documentation" [Online]. Available: http://www.mathworks.com/access/helpdesk/help/toolbox/distcomp/

[10] Interactive Supercomputing Inc., (2009). "StarP for MATLAB Users," [Online]. Available: http://www.interactivesupercomputing.com/products/starpandmatlab.php

[11] N. T. Bliss and J. Kepner (2009). "pMATLAB: Parallel MATLAB toolbox," [Online]. Available: http://www.ll.mit.edu/mission/isr/pmatlab/pmatlab.html

[12] Blue Collar Computing Software team (2009). "bcMPI," [Online]. Available: http://www.osc.edu/bluecollarcomputing/applications/bcMPI/index.shtml

[13] The MathWorks (2009). "The Mathworks–Support," [Online]. Available: http://www.mathworks.com/support

[14] L. Dagum and R. Menon, "OpenMP: An industry standard API for shared-memory programming," *IEEE Computat. Sci. Eng.*, vol. 5, no. 1, pp. 46–55, Jan.–Mar., 1998.

[15] The OpenMP Architecture Review Board (2009). "OpenMP–The OpenMP API specification for parallel programming," [Online]. Available: http://www.openmp.org

[16] J. Kepner and S. Ahalt. (2004). "MatlabMPI. J. Parallel Distrib. Comput.," [Online], 64(8), 997–1005. Available: http://dx.doi.org/10.1016/j.jpdc.2004.03.018

[17] N. T. Bliss and J. Kepner, "pMATLAB parallel MATLAB library," *Int. J. High Perform. Comput. Appl.*, vol. 21, no. 3, pp. 336–359, 2007.

[18] G. Sharma and J. Martin. (2009). "MATLAB: A language for parallel computing," *Int. J. Parallel Program.* [Online]. Available: http://dx.doi.org/10.1007/s10766-008-0082-5

[19] K. Yelick, D. Bonachea, W.-Y. Chen, P. Colella, K. Datta, J. Duell, S. L. Graham, P. Hargrove, P. Hilfinger, P. Husbands, C. Iancu, A. Kamil, R. Nishtala, J. Su, M. Welcome, and T. Wen, "Productivity and performance using partitioned global address space languages," in *Proc. 2007 Int. Workshop Parallel Symbolic Computation (PASCO'07)*. New York: ACM, 2007, pp. 24–32.

[20] MPI Forum (2009). "Message passing interface," [Online]. Available: http://www.mcs.anl.gov/research/projects/mpi/

[21] M. P. I. Forum, *MPI: A Message-Passing Interface, Version 2.2*, 2009.

[22] M. Garland, S. L. Grand, J. Nickolls, J. Anderson, J. Hardwick, S. Morton, E. Phillips, Y. Zhang, and V. Volkov, "Parallel computing experiences with CUDA," *IEEE Micro*, vol. 28, no. 4, pp. 13–27, 2008.

[23] D. Luebke, "CUDA: Scalable parallel programming for high-performance scientific computing," in *Proc. 5th IEEE Int. Symp. Biomedical Imaging: From Nano to Macro*, 2008, pp. 836–838.

[24] M. Boyer, D. Tarjan, S. T. Acton, and K. Skadron, "Accelerating leukocyte tracking using CUDA: A case study in leveraging manycore coprocessors," in *Proc. IEEE Int. Symp. Parallel and Distributed Processing*. Washington, DC: IEEE, 2009, pp. 1–12.

[25] T.-X. Corp., "GPILib: Products: Tech-X Corporation," [Online]. Available: http://www.txcorp.com/products/GPULib

[26] NVIDIA (2009). "MATLAB plug-in for CUDA," [Online]. Available: http://www.nvidia.com/object/matlab\cuda.html

[27] D. Bader, K. Madduri, J. Gilbert, V. Shah, J. Kepner, T. Meuse, and A. Krishnamurthy, "Designing scalable synthetic compact applications for benchmarking high productivity computing systems," *Cyberinfrastruct. Technol. Watch*, vol. 2, no. 4B, pp. 41–51, 2006.

[28] P. Luszczek, J. Dongarra, D. Koester, R. Rabenseifner, B. Lucas, J. Kepner, J. McCalpin, D. Bailey, and D. Takahashi, "Introduction to the HPC Challenge benchmark suite," *Lawrence Berkeley National Lab.*, LBNL Tech. Rep. 57493, 2005.

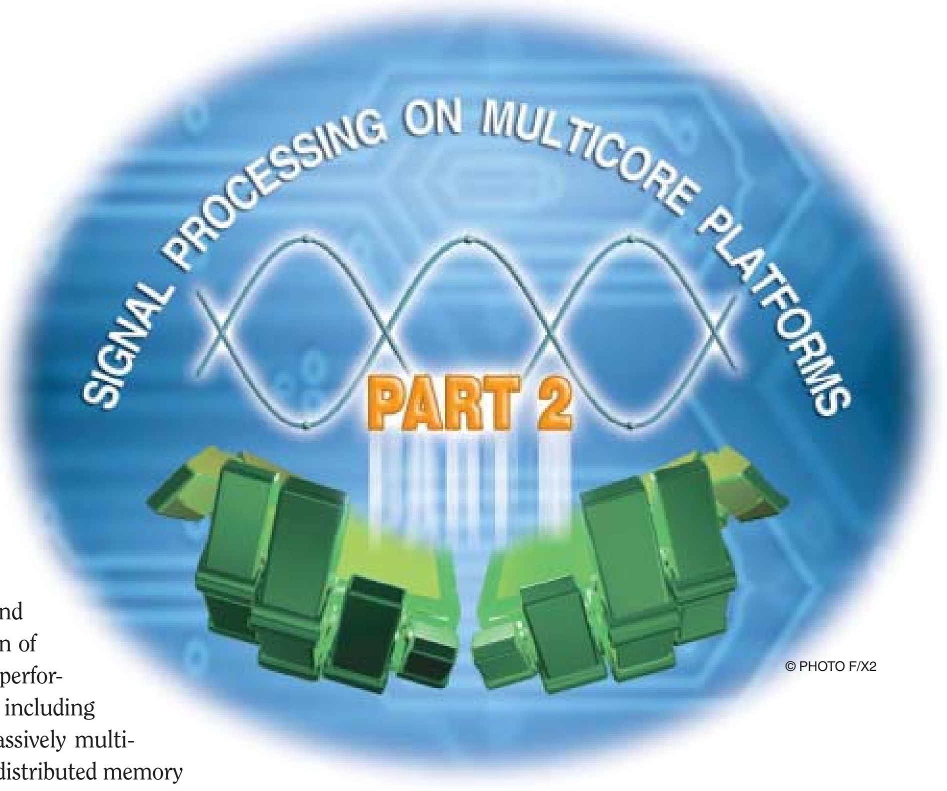
[29] G. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *Proc. Spring Joint Computer Conf.* New York: ACM Press, 1967, pp. 483–485.

[30] A. Palacios, J. Aven, and P. Longhini, "Cooperative dynamics in coupled noisy dynamical systems near a critical point: The dc superconducting quantum interference device as a case study," *Phys. Rev. E*, vol. 74, 2006. [SP]

[Ramtin Shams, Parastoo Sadeghi, Rodney A. Kennedy, and Richard I. Hartley]

A Survey of Medical Image Registration on Multicore and the GPU

[A look at early, recent, and state-of-the-art methods using high-performance computing architectures]



In this article, we look at early, recent, and state-of-the-art methods for registration of medical images using a range of high-performance computing (HPC) architectures including symmetric multiprocessing (SMP), massively multiprocessing (MMP), and architectures with distributed memory (DM), and nonuniform memory access (NUMA). The article is designed to be self-sufficient. We will take the time to define and describe concepts of interest, albeit briefly, in the context of image registration and HPC. We provide an overview of the registration problem and its main components in the section "Registration." Our main focus will be HPC-related aspects, and we will highlight relevant issues as we explore the problem domain. This approach presents a fresh angle on the subject than previously investigated by the more general and classic reviews in the literature [1]–[3]. The sections "Multi-CPU Implementations" and "Accelerator Implementations" are organized from the perspective of high-performance and parallel- computing with the registration problem embodied. This is meant to equip the reader with the knowledge to map a registration problem to a given computing architecture.

IN AN OPERATING ROOM NOT SO FAR INTO THE FUTURE

A surgeon is performing a potentially life-saving pancreatectomy on a patient in early stages of pancreatic cancer. Two small incisions of no more than half an inch allow laparoscopic tools including a video camera and an ultrasound probe to be guided inside the abdominal cavity. A third, larger incision, is occupied by a hand-access device that facilitates the operation. The surgeon is able to locate the tumor in the ultrasound view with ease. This is largely possible due to a newly installed

Digital Object Identifier 10.1109/MSP.2009.935387

1053-5888/10/$26.00©2010IEEE

three-dimensional (3-D) navigation and visualization system that virtually renders the patient transparent.

The visualization system combines data from preoperative magnetic resonance (MR) and computed tomography (CT) scans with intraoperative laparoscopic ultrasound data to produce real-time high quality and dynamic 3-D images of the patient, in a process better known as multimodal registration and fusion. The high quality 3-D images of the tumor and the surrounding tissue allow the surgeon to resect the malignant cells with little damage to healthy structures.

Such a minimally invasive approach avoids the trauma of open surgery, and a faster recovery time means that the patient will be released from the hospital in just two days.

MULTIPROCESSING IN AN OPERATING ROOM

Image-guided therapy (IGT) systems play an increasingly important role in clinical treatment and interventions. By providing more accurate information about a patient during a procedure, these systems improve the quality and accuracy of procedures and make less invasive options for treatment available. They contribute to reduced morbidity rate, intervention time, post-intervention care, and procedure costs. For practical reasons, however, imaging systems that can be deployed in an operating room produce images with lower resolutions and lower signal to noise ratios than can be achieved by the state-of-the-art imaging systems preoperatively. Therefore, it is desirable to be able to use preoperative images of a patient together with those acquired during a procedure for best results. In brain surgery, for example, the main challenge is to remove as much as the malignant tissue as possible without affecting critical structures and while minimizing damage to healthy tissue. The surgeon uses high quality CT and MR scans of the patient to carefully plan a procedure. During a procedure, however, the brain undergoes varying levels of deformations at different stages of the surgery known as the brain shift. This brain shift, a result of change in the intracranial pressure, leakage of cerebrospinal fluid and removal of tissue, affects the accuracy of earlier planning and needs to be compensated for. The surgeon may take a number of intraoperative scans to correct the plan based on patient's current state and also to detect complications such as bleeding. To support the surgeon, the IGT system needs to register intraoperative scans with the patient and with preoperative images.

Modern medical imaging technologies are capable of producing high resolution 3-D or four-dimensional (4-D) (3-D + time) images. This makes medical image processing tasks at least one dimension more compute-intensive than standard two-dimensional (2-D) image processing applications. The higher computational cost of medial image analysis together with the time constraints imposed by the medical procedure determine the range of tools that can be practically offered through an IGT platform. It also often means that an IGT platform has to rely on HPC hardware and highly parallelized software. There are other practical considerations. For example, equipment used in an operating room should be designed to minimize footprint, power consumption, operating noise, and cost.

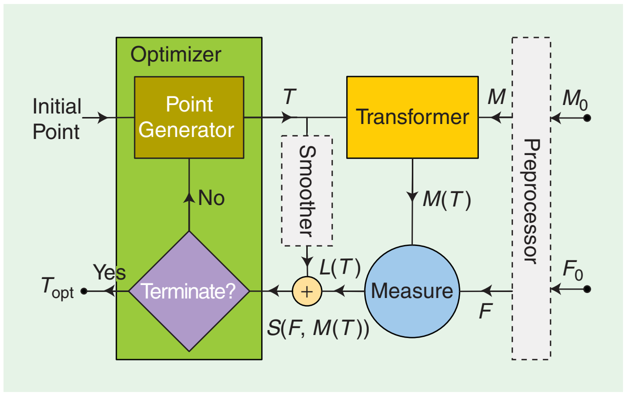
The continued development of multicore and massively multiprocessing architectures in recent years holds great promise for interventional setups. In particular, massively multiprocessing graphics units with general-purpose programming capabilities have emerged as front runners for low-cost high-performance processing. HPC, in the order of 1 TFLOPS, is available on commodity single-chip graphics processing units (GPUs) with power requirements not much greater than an office computer. Multi-GPU systems with up to eight GPUs can be built in a single host and can provide a nominal processing capacity of eight TFLOPS with less than 1,500 W power consumption under full load.

Hardware and architectural complexities in designing multicore systems aside, perhaps as big a challenge is an overhaul of existing application design methodologies to allow efficient implementation on a range of massively multicore architectures. As one quickly might find, direct adaptation of existing serial algorithms is more often than not neither possible due to hardware constraints nor computationally justified.

REGISTRATION

Registration is a fundamental task frequently encountered in image processing applications [1]. In medical applications, images of similar or differing modalities often need to be aligned as a preprocessing step for many planning, navigation, data-fusion and visualization tasks. Registration of images has been extensively researched in the medical imaging domain. Image based registration may use features that are derived from the subject's anatomy or those artificially introduced specifically for registration purposes. The former class of registration methods are known as intrinsic and the latter as extrinsic [2]. Extrinsic methods involve introduction of foreign objects such as stereotactic frames or fiducial markers and may be invasive. Once attached to the subject, markers remain fixed for multiple imaging sessions and can be used to align the images. Intrinsic methods, on the other hand, are noninvasive and can be used retrospectively. They may match a small set of corresponding anatomical and geometrical landmarks, use a set of structures obtained through segmentation, or be based on the entire content of images (e.g., voxel intensities). Content-based methods are particularly of interest since they can be fully automated but are typically compute-intensive. The focus of this survey is content-based registration methods.

Figure 1 shows various components of a general registration solver, with the main components being a transformer, a measure, and an optimizer. Registration as depicted here is an iterative process where one image is transformed within a predetermined parameter space and compared against the other. We call the former the moving and the latter the fixed image. A measure of similarity or distance is computed between the images at each step and used to determine if they are "sufficiently" aligned. This process is controlled by the optimizer that starts from an initial guess and determines subsequent



[FIG1] A general registration solver and its main components: *F, M,* and *M(T)* are fixed, moving, and transformed moving images, respectively.

steps to reach an optimal alignment. We will discuss each component in more detail in the following subsections.

TRANSFORMER

The transformer maps points in the moving image to new locations in the transformed image. Depending on the registration problem, a transformation can be collinear or deformable. Collinear transformations are line-preserving i.e., map straight lines onto straight lines. Collinear transformations can be described by a 4×4 matrix acting on homogeneous vectors representing 3-D points. Examples of collinear transformations include rigid, similarity, affine, and projective (projective transformations are rarely required in medical imaging applications). For this reason, these types of transformations have nearly identical complexity. Methods that implement rigid registration can be easily extended to affine, often without any change to the transformer.

Deformable transformation methods can be further categorized as parametric and nonparametric. Nonparametric methods are based on a variational formulation of the registration problem, where the transformation is described by an arbitrary displacement field regularized by some smoothing criteria [4]. Parametric methods are based on some piecewise polynomial interpolation of a displacement field using a set of control points placed in the image domain. B-splines are the favorites because they induce local deformations that limit the computational complexity of a large grid of control points [5]. Other functions such as thin-plate splines and Bezier functions have also been used. There are efficient methods for nonparametric registration including multigrid solvers. While parametric methods are more demanding, they yield themselves more easily to multimodal registration applications.

The transformer determines the intensity of the points in the transformed image by interpolating intensity values of corresponding points in the moving image. The simplest and fastest interpolation method is the nearest neighbor interpolation. Nearest neighbor should never be used in practice, as it results in poorly shaped cost functions, but may be useful to establish the baseline performance of the transformer. The most commonly used interpolation method is linear interpolation. Other methods include quadratic, cubic, cubic B-spline, and Gaussian interpolation [6].

A transformer spends the majority of its time performing interpolations. As noted by Castro-Pareja et al. [7] interpolation of the transformed moving image does not benefit from standard memory caching mechanisms due to nonsequential pattern of access to memory with low locality. As a result, transformer performance can well become memory bound.

MEASURE

A method of measuring the similarity or distance between images is required for automatic registration. Ideally a similarity measure attains its maximum, where the images are perfectly aligned and decreases as the images move farther away. A distance measure, on the other hand, attains its minimum where the images are aligned.

Commonly used similarity and distance measures are summarized in Table 1. Just as different classes of transformations are suitable for modeling different geometric distortions between the images, different similarity measures are used for different intensity distortions between the images. Measures are broadly categorized based on their suitability for single-modality and multimodality problems. All of the single-modality measures listed in Table 1 can be calculated by independent computations at each spatial location. From a parallelization point of view, this makes them readily adaptable to single instruction multiple data (SIMD) instruction sets and architectures such as GPUs. Multimodality measures determine statistical (mutual information) or functional (correlation ratio) dependance of images where each image is assumed to be a realization of an underlying discrete random variable. These methods require estimation of joint and marginal probability mass functions (pmfs) of the underlying discrete random variables from image data. Methods of pmf computation can be parallelized with varying degrees of difficulty and performance improvement. We will discuss this issue in more detail in the context of MI computation on the GPU in the section "GPUs."

OPTIMIZER

The optimizer is responsible for an efficient and often nonexhaustive strategy to search the transformation parameter space for the best match between the images. In image registration, optimizers can be broadly categorized as gradient-based or gradient-free, global or local, and serial or parallelizable.

Gradient-based methods require computation of partial derivatives of a cost function in addition to frequent computation of the cost function itself. Therefore, from an implementation perspective, they are more involved than gradient-free methods. The gradient computation can be based on the numerical estimation of the derivatives using finite differences. Alternatively, direct computation of the gradient can be performed when closed-form equations for the partial derivatives can be derived.

[TABLE 1] COMMONLY USED MEASURES.

MEASURE	ACRONYM	TYPE	USAGE	FORMULA[1]
SUM OF SQUARED DIFFERENCES	SSD	DIST.	SINGLE-MOD	$\mathcal{D}_{\text{SSD}}(\mathcal{I},\mathcal{J}) = \sum_{\mathbf{x}\in\Omega}(\mathcal{I}(\mathbf{x}) - \mathcal{J}(\mathbf{x}))^2$
SUM OF ABSOLUTE DIFFERENCES	SAD	DIST.	SINGLE-MOD	$\mathcal{D}_{\text{SAD}}(\mathcal{I},\mathcal{J}) = \sum_{\mathbf{x}\in\Omega}\lvert\mathcal{I}(\mathbf{x}) - \mathcal{J}(\mathbf{x})\rvert$
NORMALIZED CROSS CORRELATION [1]	NCC	SIM.	SINGLE-MOD	$\mathcal{S}_{\text{NCC}}(\mathcal{I},\mathcal{J}) = \sum_{\mathbf{x}\in\Omega}\frac{\mathcal{I}(\mathbf{x})\,\mathcal{J}(\mathbf{x})}{\sqrt{\mathbb{E}[\mathcal{I}(\mathbf{x})^2]\mathbb{E}[\mathcal{J}(\mathbf{x})^2]}}$
CORRELATION COEFFICIENT [1]	CC	SIM.	SINGLE-MOD	$\mathcal{S}_{\text{CC}}(\mathcal{I},\mathcal{J}) = \sum_{\mathbf{x}\in\Omega}\frac{(\mathcal{I}(\mathbf{x}) - \mathbb{E}[\mathcal{I}(\mathbf{x})])(\mathcal{J}(\mathbf{x}) - \mathbb{E}[\mathcal{J}(\mathbf{x})])}{\sigma(\mathcal{I})\sigma(\mathcal{J})}$
GRADIENT CORRELATION	GC	SIM.	SINGLE-MOD	$\mathcal{S}_{\text{GC}}(\mathcal{I},\mathcal{J}) = \frac{1}{d}\sum_{i=1}^{d}\mathcal{S}_{\text{CC}}\left(\frac{\partial\mathcal{I}}{\partial x_i}, \frac{\partial\mathcal{J}}{\partial x_i}\right)$
MUTUAL INFORMATION [8, 9]	MI	SIM.	MULTI-MOD	$\mathcal{S}_{\text{MI}}(\mathcal{I},\mathcal{J}) = \sum_{i}\sum_{j}p_{\mathcal{I}\mathcal{J}}(i,j)\log\frac{p_{\mathcal{I}\mathcal{J}}(i,j)}{p_{\mathcal{I}}(i)p_{\mathcal{J}}(j)}$
NORMALIZED MUTUAL INFO. [10]	NMI	SIM.	MULTI-MOD	$\mathcal{S}_{\text{NMI}}(\mathcal{I},\mathcal{J}) = \frac{2\mathcal{S}_{\text{MI}}(\mathcal{I},\mathcal{J})}{H(\mathcal{I}) + H(\mathcal{J})}$ (SEE NOTE 2)
CORRELATION RATIO [11]	CR	SIM.	MULTI-MOD	$\mathcal{S}_{\text{CR}}(\mathcal{I};\mathcal{J}) = \frac{\sigma^2(\mathbb{E}[\mathcal{J}\mid\mathcal{I}]}{\sigma^2(\mathcal{I})}$

[1]$\Omega \subset \mathbb{R}^d$ represents a d-dimensional image domain.
[2]Entropy is defined as $H(\mathcal{I}) = \sum_i p_{\mathcal{I}}(i)\log 1/p_{\mathcal{I}}(i)$, where image $\mathcal{I}$ is assumed to be a discrete random variable with a probability mass function (pmf) given by $p_{\mathcal{I}}(\cdot)$.

Local methods find a local optimum in the vicinity of an initial point and within their capture range. They may converge to an incorrect alignment if not properly initialized. Global methods, however, find the global optimum within a given range of parameters. They are robust with respect to selection of the initial point but at the cost of slower convergence. Global and local methods may be combined to improve robustness while maintaining a reasonable convergence rate.

Some optimization algorithms are only suited for serial execution, where each optimization step is dependent on the outcome of previous step(s). Others may be amenable to parallelization. For example, each step of the gradient descent optimization in N-dimensional space requires computation of N partial derivatives of the cost function. Here, there is limited opportunity to run up to N tasks in parallel, and of course the additional line minimization step that may follow cannot be readily parallelized. We call such methods partially parallelizable. And finally, we refer to optimization methods that have been designed for parallel execution with minimal interstep dependency as fully parallelizable.

Table 2 lists some optimization algorithms used for image registration and their respective classification.

The overall performance of a registration algorithm is dependent on the effectiveness of the optimization strategy. This in turn depends on the iterations needed for the algorithm to converge. For gradient-free optimization, we define an iteration as a step that involves a single computation of the cost function. For gradient-based optimization, an iteration is defined as a step that involves a single computation of the gradient. Depending on the type of gradient-based method this may involve several evaluations of the cost function.

Gradient-based optimizers do more in a single iteration and they also converge with a significantly lower number of iterations compared to gradient-free methods. The convergence rate of an optimizer depends on many factors including the size of the parameter space, optimizer settings (e.g., convergence criteria), and the misalignment between the images. It is also often data dependent.

The computational bottleneck of registration is not the optimizer but the computation of the transformation and the measure. Most researchers have focused on fine-grained parallelization of these components. A few have considered coarse-grained parallelization, which involves parallelization of the optimizer itself [18], [19].

PREPROCESSOR

We have shown the preprocessor in dotted lines in Figure 1 to emphasize that it is an optional component. Preprocessing

[TABLE 2] CLASSIFICATION OF SOME OPTIMIZATION METHODS.

METHOD	CLASSIFICATION		
POWELL [12]	GRADIENT FREE	LOCAL	SERIAL
SIMPLEX [13]	GRADIENT FREE	LOCAL	PARTIALLY PARALLELIZABLE
SOBLEX[1] [14]	GRADIENT FREE	COMBINED	PARTIALLY PARALLELIZABLE
MDS[1,2] [15]	GRADIENT FREE	LOCAL	PARTIALLY PARALLELIZABLE
GRADIENT DESCENT [12]	GRADIENT BASED	LOCAL	PARTIALLY PARALLELIZABLE
QUASI-NEWTON [12]	GRADIENT BASED	LOCAL	PARTIALLY PARALLELIZABLE
LEVENBERG-MARQUARDT [12]	GRADIENT BASED	LOCAL	PARTIALLY PARALLELIZABLE
SIMULATED ANNEALING [12]	GRADIENT FREE	COMBINED	PARTIALLY PARALLELIZABLE
DIRECT[3] [16]	GRADIENT FREE	GLOBAL	FULLY PARALLELIZABLE
GENETIC [17]	GRADIENT FREE	GLOBAL	FULLY PARALLELIZABLE

[1] A simplex variant, [2] multidirectional search, [3] dividing rectangles.

encapsulates a wide range of tasks that may be performed on images outside the optimization loop and at the beginning of the process. These may include filtering, rectification, gradient computation, pyramid construction, feature detection, etc. An example is given in one of the earlier efforts to parallelize image registration by Warfield et al. [20]. They extract features based on tissue labels given by prior segmentation and parallelize a feature-based interpatient registration method on a cluster of multiprocessor computers. They use the number of mismatching labels (NML) as a measure of distance in their registration algorithm.

Given that preprocessor is not in the critical pass, there is little incentive for parallelizing it. Unless of course the registration process itself is sped up to the point that preprocessing becomes a bottleneck. This is likely to become the case as registration algorithms enter the real-time domain.

COMPUTATIONAL EXPENSE OF IMAGE REGISTRATION

Image registration in general is computationally expensive and has been largely confined to preoperative applications. The main bottlenecks are typically the transformer and the computation of the measure. Single modality measures such as sum of squared differences (SSD) and correlation coefficient (CC) are less compute-intensive than multimodality measures such as mutual information (MI) and correlation ratio (CR). (Some authors use "normalized cross correlation" to refer to correlation coefficient. We prefer correlation coefficient, which is the accepted term in statistics.) Computation of MI requires an estimation of the joint probability density of image intensities. This typically entails, computing a joint histogram of image intensities. A seemingly simple task that is far from trivial on some massively parallel architectures such as GPUs [21].

A sample breakdown of computations in one iteration of a gradient-free optimization algorithm is given in Table 3 for affine registrations using a single modality and a multimodality measure. The measurements are based on a Quad core Intel Core i7 920 and an NVIDIA GTX 295. The time spent outside of the measure and transformation components is negligible compared to the measure and transformation. On the CPU, the execution time is dominated by the transformer whereas on the GPU, the time spent in computing the measure, particularly for the MI, exceeds the transformer time. This is expected as GPUs are designed to speed up geometric transformations. Obviously, for more complex transformation models such as the deformable B-splines, more time will be spent in the transformer for both platforms.

[TABLE 3] A SAMPLE BREAKDOWN OF COMPUTATIONS FOR AFFINE REGISTRATIONS ON A MULTICORE CPU AND A GPU.

	AFFINE (SSD)		AFFINE (MI)	
	MEASURE	TRANSFORM	MEASURE	TRANSFORM
CPU	4.3%	95.7%	13.5%	86.5%
GPU	50.4%	49.2%	86.9%	13.0%

We note that optimization algorithms make decisions based on the measure and do not directly use the intermediate results of the transformer. As such, transformation and similarity measure computations may be performed in one step and within the same module to remove the need for storage and subsequent retrieval of transformed image data. This obviously improves performance, especially where input/output traffic is an issue. However, it also makes it more difficult to modularize the implementation and cater for arbitrary combinations of transformations and measures.

MULTI-CPU IMPLEMENTATIONS

SYMMETRIC MULTIPROCESSING

In SMP architectures, multiple CPUs/cores have access to a single shared main memory. This makes parallelization of serial code relatively straightforward. The main methods for parallelization on SMP architectures are POSIX threads (pthreads) and OpenMP [22], [23]. The pthreads standard defines an application programming interface (API) for explicit instantiation, management and synchronization of multiple threads, whereas OpenMP mainly consists of a set of compiler directives (and a supporting API) that allows for implicit parallelization.

Most serial programs can be parallelized on SMP architectures with minimal modification. The ease with which parallelization can be achieved, especially with OpenMP, can be deceiving. There is an adage in HPC circles that says "OpenMP does not make parallel programming easy, it only makes bad parallel programming easy." We should emphasize that there is nothing inherently inhibiting about OpenMP or SMP platforms. It is only that optimal parallelization usually requires a change in the algorithm, programming model and memory access pattern in addition to the syntax. We encourage the reader to be prepared to reevaluate the approach to solving a problem on parallel systems and avoid the temptation of simply mapping a serial code to multiple threads. We also advise that use of synchronization primitives should be limited to a minimum and alternative methods to achieve an outcome without synchronization should be investigated. Synchronization refers to any mechanism for coordinating multiple threads or processes to complete a task. Examples of synchronization primitives include mutual exclusion (mutex), thread-join, and barrier. Atomic operations also involve implicit synchronization.

A good example of SMP parallelization of a registration algorithm is given by Rohlfing et al. [24]. They use pthreads to parallelize B-spline deformable registration on 64 CPUs. They exploit a combination of procedural (precomputation, multiresolution, and adaptive activation of control points) and architectural elements (e.g., data partitioning) to optimize their method. While the hardware has been long superseded, their approach is still relevant today. We would not change much about their method except that they use synchronized reduction of partial joint histograms into a global histogram in the MI computation phase by using the mutex lock. One can avoid the need for synchronization by dividing partial histograms and the resulting global histogram among the available threads. For N threads, this divides each partial histogram into N equally sized nonoverlapping regions. Each thread, then, computes part of the global histogram by summing values across corresponding regions of partial histograms. Since the regions are nonoverlapping, the computations are guaranteed to be free of write-conflicts and no synchronization is required.

MULTIPROCESSING WITH NUMA

Efficient memory access is an important design consideration in multiprocessor systems with many cores where maintaining an efficient cache coherency on a single-shared-bus becomes less practical as the number of processors increases. NUMA architecture divides memory into multiple banks; each assigned to one processor. Processors have faster access to their local bank than remote banks attached to other processors.

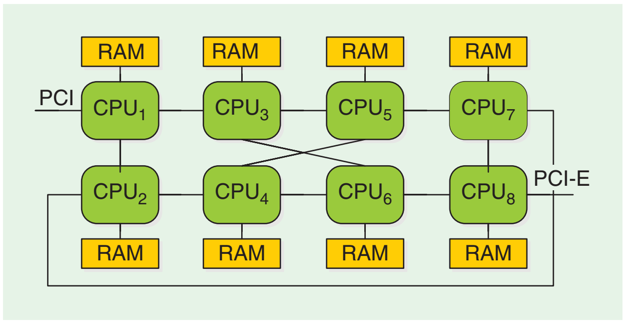
Access to memory on remote banks can be several times slower than access to local memory. This is due to data traveling through a longer path and also transient access requests by other processors that may require the memory bus to be shared. Figure 2 shows the schematic of a multiprocessor system with a NUMA architecture. An algorithm that is optimally designed for NUMA makes only infrequent attempts to access data on remote banks. A parallel application can theoretically achieve linear scalability with respect to memory throughput whenever proper distribution of memory to local banks is possible.

Image registration can be efficiently implemented on NUMA architectures as shown in Figure 3. Both the transform and measure computation can work on a spatial subset of the images. To achieve optimal performance, the fixed image F is divided among the memory banks and the corresponding portion of the transformed moving image $M(T)$ will also be stored on the same memory bank. However, the path taken by the optimization algorithm cannot be determined a priori and the transformer will use different areas of M to create the local portion of $M(T)$ at each iteration. As such, each memory bank will need to receive a local copy of the moving image M during the initialization step. Given that the optimization algorithm will take several iterations to converge, this initial overhead is justified.

The distribution of resources to specific memory banks requires setting an appropriate memory and processor affinity. Processor affinity refers to explicit binding of a thread to a specific processor. Memory affinity is explicit allocation of data on a specific memory bank. This is operating system dependent and will make the code less portable. The alternative is, of course, to be completely oblivious to the memory architecture and hope that the compiler and the operating system will make the right decisions. This may not be an entirely unreasonable strategy, depending on the number of processors and whether a program is memory bound or CPU bound. However, as the number of available CPUs increases or for programs that are memory intensive, it becomes more important to design an optimal memory access strategy.

MULTIPROCESSING WITH DISTRIBUTED MEMORY

DM architectures are characterized by lack of access to a global shared memory available to all processors. DM systems are typically built by clustering SMP or NUMA nodes. As such, in distributed architectures, subgroups of processors have access to shared memory.

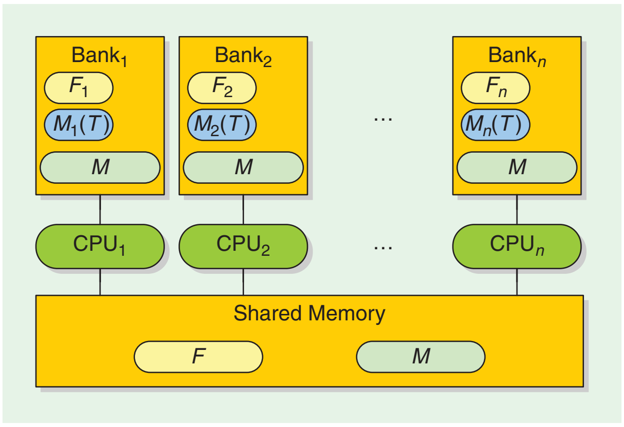


[FIG2] SunFire X4600 M2 schematic with eight NUMA nodes. A CPU can access remote memory through a maximum of three hops.

From a programming standpoint, these systems are characterized by the need for explicit data distribution and interprocess communication. The former has to be built into the application design and the latter is most commonly achieved through the message passing interface (MPI) [25].

The model given for data distribution in NUMA Figure 3 can be equally applied here. An early implementation is given by Butz and Thiran [18], where a Linux cluster was used to speed up MI-based registration for a global genetic optimizer. In [26], Ino et al. further partition the moving image to reduce the memory usage. This is motivated by the need to process large images in the order of $1{,}024 \times 1{,}024 \times 590$ voxels. Partitioning both images also reduces traffic on the network during initialization. This can be an important consideration as the number of nodes increases and the overhead of the initialization phase compared to the optimization phase can no longer be ignored. Partitioning the moving image requires a prior estimate of the range of transformation parameters to ensure that a large enough portion of the image is loaded for the transformer.

A variation is given by distributed shared memory (DSM) architectures, where a large virtual address space is made available to all processes across all nodes. DSM can only hide the



[FIG3] Partitioning of the data set among multiple memory banks for improved access. The original data is loaded from a shared storage medium.

mechanism of communication between processes and not the associated latency. We argue that if the end goal is to achieve the highest performance, little benefit can be drawn from the convenience of a DSM architecture and the program should be designed to be aware of the locality of data.

Wachowiak and Peters [19] develop MI-based registration for a DSM architecture. Their implementation does not take memory locality into account, but they use DIRECT and MDS parallel optimization methods to their advantage. This coarse-grained parallelization results in lower communication-to-computation overhead.

As some authors have pointed out [27], a major benefit of DM clusters is their lower cost compared to many-core SMPs or DSM systems.

ACCELERATOR IMPLEMENTATIONS

GRAPHICS PROCESSING UNITS

The majority of recent research in multicore adaptation of registration algorithms has been focused on GPUs [28]–[34]. There are several reasons for the interest in GPUs. Thanks to fierce competition and driven by the gaming industry, GPUs today provide some of the highest performance per dollar and the lowest power consumption per FLOPS of any computing platform. While not every radiology department can afford the cost and space needed by a conventional HPC data center, they can certainly benefit from unlocking the computational power of the GPUs in their existing computers.

GPU implementations tend to be more challenging than multicore CPU implementations and are more rewarding in terms of achievable performance gains. Earlier work in this area (mainly prior to 2007) [35]–[42] involved devising methods to map the registration problem onto a graphics pipeline that was not specifically designed for general-purpose computing. The GPU landscape has since gone through a seismic change with the introduction of native general-purpose computing capabilities in late 2006. The GPU registration literature prior to 2007 has been superseded from both hardware and software perspectives. We will focus on the latest technology for general-purpose computing on GPUs in this section.

The modern software platforms for general-purpose programming on the GPU are currently NVIDIA's CUDA [43] and AMD/ATI's Brook+ [44]. These platforms are vendor-specific, however, OpenCL compliant implementations that provide hardware-independence are being gradually released by the vendors. This essentially invalidates the only remaining argument in favor of using the graphics pipeline for general-purpose programming, which has been better portability.

None of the papers we considered for this survey developed their methods for ATI Brook+. It appears that the research community has almost exclusively adopted CUDA as their preferred GPU platform. This is likely to change with wider support for OpenCL in non-GPU architectures such as IBM's Cell/BE and Intel's Larrabee.

Modern GPUs extend the single instruction multiple data (SIMD) paradigm to a single instruction multiple threads architecture (SIMT). SIMT provides more flexibility by parallelism for (almost) independent threads as well as data-parallel code. GPUs achieve their computational performance by dedicating more transistors to their arithmetic logic units (ALUs) for data processing, at the expense of reduced flow control and data caching. They extend the conventional thread-level parallelism by introducing two additional layers of parallelism in the form of closely knit groups of threads known as warps or wavefronts, and groups of warps/wavefronts known as thread blocks or simply blocks. Warps are significant since they define the unit of flow control in a GPU. Threads in a warp are bound to execute the same instruction (on different data). Diverging paths of execution for threads in a warp result in serial execution of all paths. Hence, an important consideration in adapting parallel code to GPU architecture is minimizing diversion in warps. This can be achieved by designing warp-aware algorithms and reorganizing data to optimize flow control. An example of such an approach is given in [33].

Another notable technical feature in the current generation of GPUs is the availability of an abundance of high bandwidth on-board RAM. The memory bandwidth of top-of-the-line GPUs exceeds 140 GB/s and cards with up to 4 GB of memory are available. This is particularly important for medical image analysis applications that have to deal with large 3-D data sets. Despite an extremely high bandwidth, the GPU's main memory is largely uncached and suffers from a rather large latency. Hence to fully utilize the bandwidth and achieve an optimal performance, one needs to understand the hardware architecture and its various memory and limited caching models. Optimum use of memory such as coalesced transfers may speed up an application by an order of magnitude. This level of flexibility is typically available with lower-level APIs and runtime SDKs such as CUDA (NVIDIA) [43] and CAL (ATI/AMD) [44]. Programs developed with a lower-level API lack portability and need to be maintained as the hardware evolves. Abstraction layers such as OpenCL and Brook+ avoid these issues by hiding memory management details from the developer. However, better portability may come at the cost of suboptimal performance.

GPUs are well equipped for speeding up geometric transformations. Geometric transformations (regardless of their type) require some sort of interpolation that involves reading the content of adjacent voxels in a cubic region of memory. Standard computer architectures are designed to optimize sequential memory access through their caching mechanism. This does not fully benefit 3-D interpolations over a cubic mesh. Modern GPUs, on the other hand, support a 3-D texture addressing mode that takes the geometric locality into account for caching purposes. This greatly improves the efficiently of transformations on the GPU.

Different MI computation methods on the GPU have been reported in the literature. Shams et al. compute MI by computing joint histograms on the GPU in [21], [29], and [33]. A main finding is that for different sized histograms (number of bins used for MI computation), the optimal algorithm differs. For bin ranges typical in MI computation (100 × 100 and above) an efficient histogram computation algorithm specifically designed for massively multiprocessing architectures is presented in [33]. The paper describes a new method for histogram computation (sort and count) that removes the need for synchronization or atomic operations, based on sorting chunks of data with a parallel sort algorithm such as bitonic sort. Lin and Medioni [30] report an adaptation of Viola's MI computation approach [8]. Their method approximates the joint pmf by stochastic sampling of the image intensities and using Parzen windowing. This method lends itself well to parallelization on the GPU, reduces the computational burden of transformations by only using a subset of image data, and provides analytic equations for computation of MI derivatives. However, sparse sampling of the data set may compromise accuracy of the registration [37]. A sampling method specifically designed for the GPU is given by Shams and Barnes [29]. This method samples the bin space for computing histograms rather than the intensity space. The method improves performance of computations and is subject to the same trade off between performance and accuracy. We note that a majority of researchers use direct computation of the histogram [3].

A natural extension to parallelization of registration algorithms on the GPU is horizontal parallelization across multiple GPUs. Multi-GPU systems belong to DM class of parallel architectures. An implementation on such systems involves data partitioning and the use of MPI as discussed in the section "Multiprocessing with Distributed Memory." We recommend the reader to refer to a more detailed discussion of the subject by Plishker et al. [45].

CELL BROADBAND ENGINE

Cell broadband engine (Cell/BE) is an asymmetric heterogeneous multicore processor with a DM architecture. It comprises a general-purpose PowerPC core known as a PPE and eight specialized vector processing cores known as SPEs. Each SPE is equipped with a four-way SIMD engine and has its own small (uncached) memory known as the local storage. Local storage is only 256 KB in the current generation of hardware, and it is shared between data and kernel instructions.

Optimal implementation of registration algorithms on Cell/BE architectures involves task-level parallelization, data partitioning, and vectorization of the code for the SPEs' SIMD engine. It also involves handling the transfer of data between the system memory and SPEs' local storage. The results by Ohara et al. [46], [47] and Rohrer and Gong [48] provide good insight into challenges involved in designing registration on this architecture for collinear and deformable registration, respectively.

FIELD PROGRAMMABLE GATE ARRAYS

A custom field programmable gate array (FPGA) accelerator prototype for MI-based rigid registration is given by Castro-Pareja et al. in [7]. They argue that a major bottleneck in MI computation using Collignon's method [9] is partial volume (PV) interpolation and that it is memory bound. They improve performance by parallelizing access to memory and assigning independent memory controllers and types of memory for storage and access to the fixed image, the moving image, and the joint histogram. A cubic addressing scheme is used for the moving image to speed up the interpolation. This is similar to caching available in GPUs for access to texture memory. An enhanced version of [7] is presented in [49] and a multirigid version with volume subdivisions is given by Dandekar [50].

FPGAs allow one to design customized hardware for specific registration tasks. However, they provide less flexibility compared to software-based implementations. With flexible general-purpose programming capabilities of modern GPUs, it is doubtful if FPGA-based implementations present any real benefit in this area.

SUMMARY OF THE LITERATURE

We have summarized existing contributions in HPC of registration methods in Table 4. The table serves as a quick reference to an array of methods on various platforms and by different groups.

Researchers have used various methods to present their performance results. All groups report at least the speedup results compared to a single-core CPU implementation. When inter-architecture comparisons are drawn, it is not always clear how well the CPU implementation has been optimized, if the streaming SIMD extensions (SSE) instruction set has been used, whether the code has been compiled as 64- or 32-b, or if 64- or 32-b floating point operations have been used. For these reasons, speedup results should be interpreted with caution, more so when the reported speedups are in the order of a hundred times or more.

Most groups report their speedups for the entire registration algorithm and for specific data sets. Comparison of different results is further complicated as authors may have implemented a multiresolution scheme to further speed up the process and used different convergence criteria. We have reported/estimated the results for the finest resolution in Table 4, whenever possible. As discussed earlier, the execution time is an almost linear function of the number of iterations of the optimization algorithm. Convergence criteria are most commonly based on the value of the measure and its relative improvement in a given step of the optimization. A less common approach is to set a fixed number of iterations as the convergence criterion. We call the former strategy dynamic convergence and the latter static convergence. Lack of associativity for floating point operations have the inevitable consequence that the same optimization algorithm operating on the same data set converges with different number of iterations on different architectures when dynamic convergence is employed. Even on the same architecture, compiler optimization of floating point operations results in variations. Unless experiments are performed on a large set of images, this skews the performance results one way or the other.

[TABLE 4] SUMMARY OF HIGH-PERFORMANCE IMAGE REGISTRATION METHODS IN THE LITERATURE.

		TRANSFORM	MEAS.	OPTIMIZER	HARDWARE	YEAR	PERF.[1]	COMMENTS	GROUP
CPU	COLLINEAR	SIMIL.	NML	POWELL	2 × SUN ENT. 5000 (2 × 8 ULTRASPARC I 167 MHZ)	1998	–	SMP CLUSTER, FEATURE BASED	WARFIELD [20]
		AFFINE	MI	GENETIC	PC CLUSTER (10 × 2 PENTIUM III 550 MHZ)	2001	–	DM, MI IS GRADIENT BASED	BUTZ [18]
		RIGID	LLC	?	PC CLUSTER (10 × 2 PENTIUM III 933 MHZ)	2002	–	BLOCK MATCHING WITH LOCAL LINEAR CORRELATION MEASURE (LLC)	OURSELIN [27]
		RIGID	MI, NMI	DIRECT, MDS	SGI ALTIX 3000 (20 ITANIUM II 1.3 GHZ)	2006	–	DMS (NUMAFLEX)	WACHOWIAK [19]
		RIGID	MI	POWELL	SUN SPARC T5120 (8 × ULTRASPARC T2 1.2 GHZ)	2009	47.7	SMP, SOLARIS	SHAMS[2]
		RIGID	MI	POWELL	INTEL Q6600 (PENTUIM CORE 2 QUAD 2.4 GHZ, FOUR CORES)	2009	15.8	SMP, 64-B LINUX	SHAMS[2]
		RIGID	MI	POWELL	INTEL CORE i7 920 (QUAD 2.66 GHZ, EIGHT THREADS)	2009	13.2	SMP, 64-B WINDOWS VISTA	SHAMS[2]
		RIGID	MI	POWELL	SUNFIRE X4600 M2 (8 × 2 OPTERON 2.6 GHZ)	2009	10.5	NUMA, 64-B LINUX	SHAMS[2]
	DEF.	B-SPLINE	NMI	GRAD. DESC. (D)	SGI ORIGIN 3800 (128 MPIS 12K)	2003	–	SMP, MAX. CPUS USED: 64	ROHLFING [24]
		B-SPLINE	NMI	GRAD. DESC. (D)	PC CLUSTER (64 × 2 PENTIUM III 1GHZ)	2005	–	DM (MYRINET)	INO [26]
GPU	COLLINEAR	RIGID	SSD	SIMPLEX	GEFORCE 6800	2006	98.0	CODED IN OPENGL AND GLSL	KÖHN [51]
		RIGID	SSD	GRAD. DESC. (B)	GEFORCE 6800	2006	858	CODED IN OPENGL AND GLSL	KÖHN[3] [51]
		RIGID	GC	?	QUADRO FX 1400, FX 3400, GTX 7800	2006	–	2-D/3-D REGISTRATION	INO [38]
		RIGID	VARIOUS	CUSTOM	GEFORCE 6800 GT	2006	–	VARIOUS MEASURES (E.G., SSD, CC, GC)	KHAMENE[3] [37]
		RIGID	VARIOUS	ARS + BN	GEFORCE 7800 GS	2008	–	2-D/3-D, VARIOUS MEASURES (E.G., SSD, CC, GC), ADAPTIVE RANDOM SEARCH + BEST NEIGHBOR	KUBIAS [42]
		RIGID	MI	SIMPLEX	GTX 8800 (16 MP/ 128 CORES)	2007	6.17	CUDA 1.0 (NO SUPPORT FOR 3-D TEXTURES), MI ESTIMATED BY BIN SAMPLING	SHAMS [29]
		RIGID	SSD	SIMPLEX	GTX 8800 (16 MP/ 128 CORES)	2008	6.05	CUDA 2.0	PLISHKER[3] [31]
		AFFINE	MI	GRAD. DESC. (A)	GTX 8800 (16 MP/ 128 CORES)	2008	–	MI ESTIMATED BY SAMPLING	LIN [30]
		RIGID	MI	POWELL	GTX 280 (30 MP/ 240 CORES)	2009	4.06	CUDA 2.0, USING 3-D TEXTURES, MI COMPUTED USING BITONIC SORT AND COUNT	SHAMS [33]
	DEFORMABLE	BEZIER	MI	POWELL	GEFORCE3 64 MB	2002	–		SOZA [35]
		NON-PAR.	SSD	GRAD. DESC.	GEFORCE FX 5800 ULTRA	2004	–	2-D/2-D, MULTIGRID SOLVER USED	STRZODKA [36]
		NON-PAR.	SSD	GRAD. DESC. (B)	GEFORCE 6800	2006	465	CODED IN OPENGL AND GLSL	KÖHN[3] [51]
		NON-PAR.	MI + KL	GRAD. DESC. (C)	GTX 7800	2007	2860	COMBINED MI AND KULLBACK-LEIBLER MEASURE, CODED IN OPENGL AND GLSL	VETTER[3] [39]
		NON-PAR.	MI + KL	GRAD. DESC. (C)	GTX 8800 ULTRA (16 MP/128 CORES)	2008	324	COMBINED MI AND KULLBACK-LEIBLER MEASURE, CODED IN OPENGL AND GLSL	FAN[3] [40]
		DEMONS	SSD	ITERATIVE	QUADRO FX 1400	2007	1050	CODED IN CG, PUBLISHED IN 2008	COURTY [41]
		DEMONS	SSD	ITERATIVE	GTS 8800 (12 MP/96 CORES)	2007	11.7	CODED IN BROOK, SSD EXCLUDED IN PERFORMANCE RESULTS	SHARP[3] [28]
		DEMONS	CC	ITERATIVE	QUADRO FX 5600 (16 MP/128 CORES)	2008	9.25	CUDA 0.9	ÖZÇELIK [32]
		B-SPLINE	SSD	GRAD. DESC. (C)	GTX 8800 (16 MP/ 128 CORES)	2008	3710	CUDA 2.0	PLISHKER[3] [31]
		POLYNOM.	MI	EXHAUSTIVE	QUADRO FX 5600 (16 MP/128 CORES)	2009	–	2-D/2-D, ULTRA LARGE 2-D IMAGES	RUIZ [34]
OTHER ACCELERATORS	COLLINEAR	RIGID	MI	N/A	FPGA (2 × ALTERA 1K100 80 MHZ)	2003	101	MI PARTIALLY COMPUTED IN H/W	PAREJA [7]
		RIGID	MI	N/A	FPGA (1 × ALTERA EP1S40 200 MHZ)	2004	20.0	MI FULLY COMPUTED IN H/W	PAREJA [49]
		AFFINE	MI	GRAD. DESC.	QS20 (2 × CELL/BE.: 2 × 1 PPE AND EIGHT SPEs)	2007	98.8	MI ESTIMATED BY SAMPLING	OHARA[3] [47]
	DEF.	MULTIRIGID	MI	SIMPLEX	FPGA (1 X ALTERA EP2S180 200 MHZ)	2007	13.4	MI FULLY COMPUTED IN H/W	DANDEKAR[3] [50]
		B-SPLINE	MI	GRAD. DESC.	QS20 (2 × CELL/BE.: 2 × 1 PPE AND EIGHT SPE)	2008	66.9	MI ESTIMATED BY SAMPLING	ROHRER [48]

[1] Normalized performance in milliseconds per mega voxel per iteration (ms/MVoxel/itr). [2] Previously unpublished result. [3] Additional information provided by the authors used to complete the table or to compute normalized performance results.

We have given normalized performance results in Table 4 where possible. The word "performance" is ambiguous in the context of registration. It is sometimes used to refer to the degree of success for a registration algorithm based on accuracy of the registration results. In this article, we use "performance" in its computational capacity referring to execution efficiency of the registration algorithm. The purpose of normalizing the reported results is to give the reader an indication of the speed-ups expected from a method without dependence on the size of images involved, convergence criteria, use of a multiresolution scheme, and to some extent the type of optimization algorithm. Normalized results are given in terms of average execution time in milliseconds for a single iteration of the optimization algorithm and for processing 1,000,000 voxel pairs (ms/MVoxel/itr).

Many authors have used gradient descent as their optimization algorithm, largely due to its simple structure and ease of implementation. Once the gradient is computed, the choices include taking a single step in a direction opposite to the gradient where the step size may be adjusted over time, or use of a line minimization algorithm such as Brent's [12]. Line minimization usually involves several computations of the cost function alone without its derivatives.

When comparing results it is important to identify which variation of the gradient descent is used. We have come across four different implementations:

- Type A: closed-form differentiation with a single step.
- Type B: closed-form differentiation with line minimization.
- Type C: numerical differentiation with a single step.
- Type D: numerical differentiation with line minimization.

Most authors exclude initialization time, including disk IO and loading data from host memory to GPU memory. This is a reasonable practice since initialization time is typically a small fraction of the registration task. Initialization occurs at the beginning of the registration algorithm whereas the optimization loop is executed several times.

Some of the information presented in Table 4 were not immediately available in the original manuscripts and were provided by the authors of the respective papers. Unless specifically specified, listed methods are for 3-D/3-D registration.

FINAL WORDS

Over the last decade, a rich and diverse literature on HPC of medical image registration has emerged. Research in this area continues to be motivated by the need to minimize the overhead of image registration that is used as an integral part of image-guided intervention and IGT systems. The continued research in this area will also facilitate the adaption of existing preoperative tools to real-time intraoperative environments.

From a technical perspective, there has been a gradual shift away from expensive SMP supercomputers to less expensive clusters of commodity computers and more recently inexpensive massively multiprocessing GPUs. This trend has the potential to lead to more widespread use of medical imaging tools in everyday clinical practice by making them affordable outside of research facilities and expensive operating theaters.

ACKNOWLEDGMENTS

We would like to thank Prof. Alistair Rendell for making staff and computational resources available for our experiments. We would also like to thank Ahmed El Zein and Benjamin Murphy for their efforts in porting the original code to Solaris, Linux, and its adaptation to Brook+.

We also thank many authors, whose work we have referred in this article, for providing additional information and for clarification of their results.

AUTHORS

Ramtin Shams (ramtin.shams@anu.edu.au) is an Australian postdoctoral Fellow in the College of Engineering and Computer Science at the Australian National University (ANU). He received his B.E. and M.E. degrees in electrical engineering from Sharif University of Technology, Tehran, and completed his Ph.D. degree at ANU in 2009 with a thesis in medical image registration. He was the recipient of a Fulbright scholarship in 2008. He has more than ten years of industry experience in the ICT sector and worked as the CTO of GPayments Pty. Ltd between 2001 to 2007. His research interests include medical image analysis, HPC, and wireless communications.

Parastoo Sadeghi (parastoo.sadeghi@anu.edu.au) is a Fellow (senior lecturer) at the Research School of Information Sciences and Engineering at ANU. She received her B.E. and M.E. degrees in electrical engineering from Sharif University of Technology, Tehran, and her Ph.D. degree in electrical engineering from The University of New South Wales in Sydney, in 2006. In 2003 and 2005, she received two IEEE Region 10 Paper Awards for her research in the information theory of time-varying fading channels. Her research interests include applications of signal processing, information theory, and HPC in telecommunications and medical image analysis.

Rodney A. Kennedy (rodney.kennedy@anu.edu.au) received his B.E. degree from the University of New South Wales, Australia, his M.E degree from the University of Newcastle, and his Ph.D. degree from ANU. For three years, he worked for the Commonwealth Scientific and Industrial Research Organization on the Australia Telescope Project. He is currently a professor and director of research at the College of Engineering and Computer Science at the ANU. His research interests are in the fields of signal processing, digital and wireless communications, and acoustical signal processing.

Richard I. Hartley (richard.hartley@anu.edu.au) is a member of the computer vision group in the College of Computer Science and Engineering at ANU. He also belongs to the Vision Science Technology and Applications Program in National ICT Australia. He graduated from the University of Toronto in 1976 with a thesis in knot theory and worked in this area for several years before joining the General Electric Research and Development Center, where he worked from 1985 to 2001. In 1991, he was awarded GE's Dushman Award for this work. In 2000, he coauthored a book on multiple view geometry. He has authored close to 200 scholarly papers and holds 32 U.S. patents.

REFERENCES

[1] L. G. Brown, "A survey of image registration techniques," *ACM Comput. Surv.*, vol. 24, no. 4, pp. 325–376, Dec. 1992.

[2] J. B. A. Maintz and M. A. Viergever, "A survey of medical image registration," *Med. Image Anal.*, vol. 2, no. 1, pp. 1–36, 1998.

[3] J. P. W. Pluim, J. B. A. Maintz, and M. A. Viergever, "Mutual-information-based registration of medical images: A survey," *IEEE Trans. Med. Imag.*, vol. 22, no. 8, pp. 986–1004, Aug. 2003.

[4] J. Modersitzki, *Numerical Methods for Image Registration*. New York: Oxford Univ. Press, 2004.

[5] D. Rueckert, L. I. Sonoda, C. Hayes, D. L. G. Hill, M. O. Leach, and D. J. Hawkes, "Nonrigid registration using free-form deformations: Application to breast MR images," *IEEE Trans. Med. Imag.*, vol. 18, no. 8, pp. 712–721, Aug. 1999.

[6] T. M. Lehmann, C. Gönner, and K. Spitzer, "Survey: Interpolation methods in medical image processing," *IEEE Trans. Med. Imag.*, vol. 18, no. 11, pp. 1049–1075, Nov. 1999.

[7] C. R. Castro-Pareja, J. M. Jagadeesh, and R. Shekhar, "FAIR: A hardware architecture for real-time 3-D image registration," *IEEE Trans. Inform. Technol. Biomed.*, vol. 7, no. 4, pp. 426–434, Dec. 2003.

[8] P. Viola and W. M. Wells, III, "Alignment by maximization of mutual information," in *Proc. Int. Conf. Computer Vision (ICCV)*, June 1995, pp. 16–23.

[9] A. Collignon, F. Maes, D. Delaere, D. Vandermeulen, P. Suetens, and G. Marchal, "Automated multimodality medical image registration using information theory," in *Proc. Int. Conf. Information Processing in Medical Imaging: Computational Imaging and Vision 3*, Apr. 1995, pp. 263–274.

[10] C. Studholme, D. L. G. Hill, and D. J. Hawkes, "An overlap invariant entropy measure of 3D medical image alignment," *Pattern Recognit.*, vol. 32, no. 1, pp. 71–86, 1999.

[11] A. Roche, G. Malandain, X. Pennec, and N. Ayache, "The correlation ratio as a new similarity measure for multimodal image registration," in *Proc. Medical Image Computing and Computer Assisted Intervention (MICCAI)*, Oct. 1998, pp. 1115–1124.

[12] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes: The Art of Scientific Computing*, 3rd ed. Cambridge, U.K.: Cambridge Univ. Press, 2007.

[13] J. A. Nedler and R. Mead, "A simplex method for function minimization," *Comput. J.*, vol. 7, no. 4, pp. 308–331, 1965.

[14] R. Shams, R. A. Kennedy, P. Sadeghi, and R. Hartley, "Gradient intensity-based registration of multi-modal images of the brain," in *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, Rio de Janeiro, Brazil, Oct. 2007.

[15] J. E. Dennis, Jr. and V. Torczon, "Direct search methods on parallel machines," *SIAM J. Optim.*, vol. 1, no. 4, pp. 448–474, 1991.

[16] D. R. Jones, C. D. Perttunen, and B. E. Stuckman, "Lipschitzian optimization without the Lipschitz constant," *J. Optim. Theory Appl.*, vol. 79, no. 1, pp. 157–181, 1993.

[17] E. Cantú-Paz, "A survey of parallel genetic algorithms," *Calculateurs Paralleles*, vol. 10, no. 2, pp. 141–171, 1998.

[18] T. Butz and J-P. Thiran, "Affine registration with feature space mutual information," in *Proc. Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2001, pp. 549–556.

[19] M. P. Wachowiak and T. M. Peters, "High-performance medical image registration using new optimization techniques," *IEEE Trans. Inform. Technol. Biomed.*, vol. 10, no. 2, pp. 344–353, Apr. 2006.

[20] S. Warfield, F. Jolesz, and R. Kikinis, "A high performance computing approach to the registration of medical imaging data," *Parallel Comput.*, vol. 24, no. 9-10, pp. 1345–1368, Sept. 1998.

[21] R. Shams and R. A. Kennedy, "Efficient histogram algorithms for NVIDIA CUDA compatible devices," in *Proc. Int. Conf. Signal Processing and Communications Systems (ICSPCS)*, Gold Coast, Australia, Dec. 2007, pp. 418–422.

[22] (2009). *OpenMP application programming interface, version 3.0, OpenMP* [Online]. Available: http://openmp.org/wp/openmp-specifications/

[23] B. Chapman, G. Jost, and R. van der Pas, *Using OpenMP: Portable Shared Memory Parallel Programming*. Cambridge, MA: MIT Press, 2008.

[24] T. Rohlfing and C. R. Maurer, Jr., "Nonrigid image registration in shared-memory multiprocessor environments with application to brains, breasts, and bees," *IEEE Trans. Inform. Technol. Biomed.*, vol. 7, no. 1, pp. 16–25, Mar. 2003.

[25] E. Lusk W. Gropp, and A. Skjellum, *Using MPI: Portable Parallel Programming with the Message Passing Interface*, 2nd ed. Cambridge, MA: MIT Press, 1999.

[26] F. Ino, K. Ooyama, and K. Hagihara, "A data distributed parallel algorithm for nonrigid image registration," *Parallel Comput.*, vol. 31, no. 1, pp. 19–43, Jan. 2005.

[27] S. Ourselin, R. Stefanescu, and X. Pennec, "Robust registration of multi-modal images: Towards real-time clinical applications," in *Proc. Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2002, pp. 140–147.

[28] G. C. Sharp, N. Kandasamy, H. Singh, and M. Folkert, "GPU-based streaming architectures for fast cone-beam CT image reconstruction and demons deformable registration," *Phys. Med. Biol.*, vol. 52, no. 19, pp. 5771–5783, 2007.

[29] R. Shams and N. Barnes, "Speeding up mutual information computation using NVIDIA CUDA hardware," in *Proc. Digital Image Computing: Techniques and Applications (DICTA)*, Adelaide, Australia, Dec. 2007, pp. 555–560.

[30] Y. Lin and G. Medioni, "Mutual information computation and maximization using GPU," in *Proc. IEEE Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2008, pp. 1–6.

[31] W. Plishker, O. Dandekar, S. S. Bhattacharyya, and R. Shekhar, "Towards systematic exploration of tradeoffs for medical image registration on heterogeneous platforms," in *Proc. IEEE Biomedical Circuits and Systems Conf.*, Nov. 2008, pp. 53–56.

[32] P. Muyan-Özçelik, J. D. Owens, J. Xia, and S. S. Samant, "Fast deformable registration on the GPU: A CUDA implementation of demons," in *Proc. Int. Conf. Computational Science and Its Applications (ICCSA)*, 2008, pp. 5–8.

[33] R. Shams, P. Sadeghi, R. A. Kennedy, and R. Hartley, "Parallel computation of mutual information on the GPU with application to real-time registration of 3D medical images," *Comput. Meth. Programs Biomed.*, to be published.

[34] A. Ruiz, M. Ujaldon, L. Cooper, and K. Huang, "Non-rigid registration for large sets of microscopic images on graphics processors," *J. Signal Process. Syst.*, vol. 55, no. 1-3, pp. 229–250, Apr. 2009.

[35] G. Soza, M. Bauer, P. Hastreiter, C. Nimsky, and G. Greiner, "Non-rigid registration with use of hardware-based 3D Bézier functions," in *Proc. Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2002, pp. 549–556.

[36] R. Strzodka, M. Droske, and M. Rumpf, "Image registration by a regularized gradient flow. A streaming implementation in DX9 graphics hardware," *Computing*, vol. 73, no. 4, pp. 373–389, Nov. 2004.

[37] A. Khamene, R. Chisu, W. Wein, N. Navab, and F. Sauer, "A novel projection based approach for medical image registration," in *Proc. 3rd Int. Workshop Biomedical Image Registration (WBIR)*, Utrecht, The Netherlands, June 2006, pp. 247–256.

[38] F. Ino, J. Gomita, Y. Kawasaki, and K. Hagihara, "A GPGPU approach for accelerating 2-D/3-D rigid registration of medical images," in *Proc. Parallel and Distributed Processing and Applications*, Feb. 2006, pp. 939–950.

[39] C. Vetter, C. Guetter, C. Xu, and R. Westermann, "Non-rigid multi-modal registration on the GPU," in *Proc. SPIE Medical Imaging: Image Processing*, Feb. 2007, pp. 651228-1–651228-8.

[40] Z. Fan, C. Vetter, C. Guetter, D. Yu, R. Westermann, A. Kaufman, and C. Xu, "Optimized GPU implementation of learning-based non-rigid multi-modal registration," in *Proc. SPIE Medical Imaging: Image Processing*, 2008.

[41] N. Courty and P. Hellier, "Accelerating 3D non-rigid registration using graphics hardware," *Int. J. Image Graph.*, vol. 8, no. 1, pp. 1–18, Jan. 2008.

[42] A. Kubias, F. Deinzer, T. Feldmann, S. Paulus, D. Paulus, B. Schreiber, and T. Brunner, "2D/3D image registration on the GPU," *Pattern Recognit. Image Anal.*, vol. 18, no. 3, pp. 381–389, Sept. 2008.

[43] (2009). *Compute unified device architecture (CUDA) programming guide, version 2.2, NVIDIA* [Online]. Available: http://developer.nvidia.com/object/cuda.html

[44] (2009). *ATI stream computing user guide, version 1.4.0.a, ATI* [Online]. Available: http://developer.amd.com/

[45] W. Plishker, O. Dandekar, S. S. Bhattacharyya, and R. Shekhar, "Utilizing hierarchical multiprocessing for medical image registration," *IEEE Signal Processing Mag.*, vol. 27, no. 2, pp. 62–68, Mar. 2010.

[46] M. Ohara, H. Yeo, F. Savino, G. Iyengar, L. Gong, H. Inoue, H. Komatsu, V. Sheinin, and S. Daijavad, "Accelerating mutual-information-based linear registration on the cell broadband engine processor," in *Proc. IEEE Int. Conf. Multimedia and Expo*, 2007, pp. 272–275.

[47] M. Ohara, H. Yeo, F. Savino, G. Iyengar, L. Gong, H. Inoue, H. Komatsu, V. Sheinin, S. Daijavad, and B. Erickson, "Real-time mutual-informatoin-based linear registration on the cell broadband engine processor," in *Proc. IEEE Int. Symp. Biomedical Imaging (ISBI)*, 2007, pp. 33–36.

[48] J. Rohrer and L. Gong, "Accelerating mutual information based 3D non-rigid registration using the cell/B.E. processor," in *Proc. Workshop on Cell Systems and Applications (WCSA)*, 2008, pp. 32–40.

[49] C. R. Castro-Pareja, J. M. Jagadeesh, and R. Shekhar, "FPGA-based acceleration of mutual information calculation for real-time 3D image registration," in *Proc. SPIE Medical Imaging: Image Processing*, 2008, pp. 212–219.

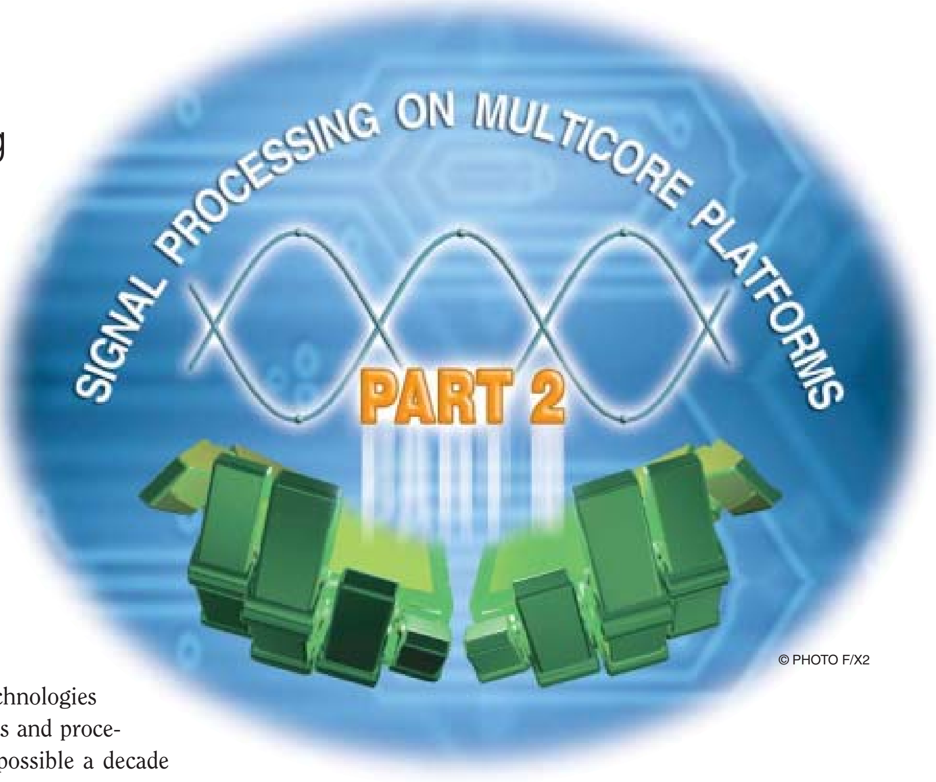
[50] O. Dandekar and R. Shekhar, "FPGA-accelerated deformable image registration for improved target-delineation during CT-guided interventions," *IEEE Trans. Biomed. Circuits Syst.*, vol. 1, no. 2, pp. 116–127, June 2007.

[51] A. Köhn, J. Drexl, F. Ritter, M. König, and H. O. Peitgen, "GPU accelerated image registration in two and three dimensions," in *Proc. Bildverarbeitung für die Medizin*, 2006, pp. 261–265. [SP]

William Plishker, Omkar Dandekar, Shuvra S. Bhattacharyya, and Raj Shekhar

Utilizing Hierarchical Multiprocessing for Medical Image Registration

The possibilities and challenges of combining acceleration approaches that utilize complementary types of parallelism



© PHOTO F/X2

Advances in medical imaging technologies have enabled medical diagnoses and procedures that were simply not possible a decade ago. The accelerating speed of acquisition and the increasing resolution of images have given doctors more information, which is taken less invasively about their patients. However, because of the multitude of imaging modalities [e.g., computed tomography (CT), positron emission tomography (PET), magnetic resonance imaging (MRI), and ultrasound (US)] and the sheer volume of data being acquired, utilizing this new data effectively has become problematic. One way to tap into the potential of this raw data is to merge these images into one integrated view through a procedure called image registration.

Digital Object Identifier 10.1109/MSP.2009.935419

COMPUTE-ENABLED MEDICINE

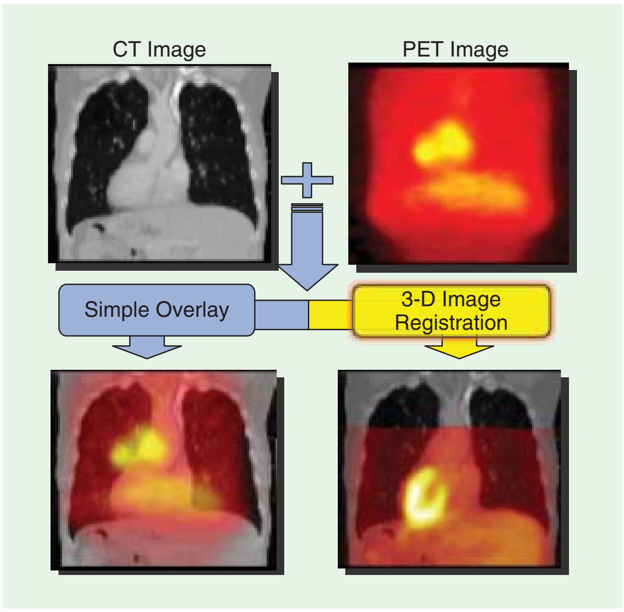
For the past decade, improving performance and accuracy has been a driving force of innovation in automated medical image registration. The ultimate goal of accurate, robust, real-time image registration will enhance diagnoses of patients and enable new image-guided intervention techniques. With such a computationally intensive and multifaceted problem, improvements have been found in high-performance platforms such as graphics processing units (GPUs) and general-purpose multicore systems, but there has yet to be a solution fast enough and effective enough to gain widespread clinical use.

To achieve the necessary speed, we believe that a synergy of approaches will be needed, requiring many cores organized at different levels of granularity. We call such processing hierarchical multiprocessing, as it requires the use of multiple styles of parallelism to be properly utilized. To accelerate medical image registration, we explore some of the key issues of hierarchical multiprocessing by leveraging a novel domain-specific framework to design and implement an image-registration algorithm on a GPU and on a cluster of GPUs, and compare them in terms of speed and accuracy. Using a set of representative images, we achieve execution times as low as 2.5 s and accuracy varying from submillimeter to 2.4 mm of average error.

IMAGE REGISTRATION IS THE PROCESS OF COMBINING IMAGES SUCH THAT THE FEATURES IN AN IMAGE ARE ALIGNED WITH THE FEATURES OF ONE OR MORE OTHER IMAGES.

INTRODUCTION

Image registration is the process of combining images such that the features in an image are aligned with the features of one or more other images. An example pairing is shown in Figure 1. The first phase of most registration techniques is correcting for whole image misalignment, called rigid registration. Nonrigid registration often follows such that nonlinear local deformation from breathing or changes over time is corrected. While automatic and robust registration algorithms exist, they tend to be computationally intensive, often taking minutes to execute on high-end general-purpose processors for rigid registration and hours for nonrigid registration. Such complexity has inhibited the adoption of registration technology in the clinical workflow. While specific requirements vary from application to application, real-time registration must be on the order of seconds to be viable in most image-guided intervention scenarios.



[FIG1] A typical medical image registration case. The CT is the fixed image and the PET is the moving image. A simple overlay of these two gives a clinician little information about the case, but an accurately registered result overlays the metabolic information of PET on the structural information of CT.

To bring more accurate and more robust image registration algorithms into the clinical setting, a significant body of research has been dedicated to acceleration techniques for image registration. A thorough discussion of this appears in the article “A Survey of Medical Image Registration on Multicore and the GPU,” by R. Shams, et al. in this issue of *IEEE Signal Processing Magazine*. Many multicore platforms already in use for this purpose are GPUs, clusters of general-purpose processors, the Cell, and even custom hardware from implementations on field programmable gate arrays (FPGAs). While many of these works have shown performance improvements, no single technique has accelerated registration sufficiently for all clinical applications. We believe real-time image registration will require a combination of parallelism styles that can be used to accelerate different aspects of the application. Proper utilization of hierarchical platforms can lead to multiplicative effects from complementary acceleration techniques.

Utilizing parallelism for any single platform may be a challenging and time consuming implementation task. It carries the normal tasks of programming (e.g., creating a software architecture, developing algorithms, testing, debugging, and performance tuning), and the added difficulties of managing parallel threads (e.g., managing communication, debugging race conditions, and load balancing). Utilizing parallelism on a hierarchical multiprocessing platform is even worse, often involving multiple programming models and environments, which designers must decide how to use before beginning to write code. To facilitate a design process in which the structure of an application is considered when mapping to these diverse programming models, we leverage a novel framework based on an image registration specific taxonomy. To demonstrate the utility of this approach, we employ this framework on a commonly used a rigid and a nonrigid registration algorithm.

To explore the challenges and potential benefits of targeting hierarchical multiprocessing platforms for image registration, we study two in particular: a single GPU and a cluster of GPUs. Using these platforms, we took our single-threaded code base on a general-purpose processor and, for the most time-consuming kernels, added acceleration tailored to the target platform. In particular, we focus on utilizing parallelism described in our taxonomy. Based on these results, we discuss the possibilities and the challenges of combining acceleration approaches that utilize complementary types of parallelism. This article expands on preliminary work on this subject, which is presented in [1] and [2].

INTENSITY-BASED REGISTRATION

Intensity-based image registration algorithms rely on similarities between voxel [three-dimensional (3-D pixel)] intensities. These algorithms are known to be robust but tend to be computationally intensive. In an intensity-based image-registration algorithm, a transformation is often described as a deformation field, in which all parts of the image to be deformed (the moving image) have a specific deformation such that they align with the other image (the fixed image). Construction of the deformation field can start from just a few parameters in the case of rigid registration or from a set of "control points" that capture the nonuniformity of nonrigid registration. The final transformation contains the information necessary to deform all of the voxels in the moving image. Once a transformation is constructed, it is applied to the moving image. This transformed image can be compared to the fixed image using a variety of similarity metrics, such as mean squared difference (MSD) or mutual information.

For iterative approaches, the similarity value is returned so that it may guide the registration algorithm towards a solution. A variety of iterative optimization algorithms have been developed for medical image registration with different convergence properties and computational efficiency. Problem parameters may also change during run time to improve speed and accuracy including sampling rates, interpolation strategies, and varying grid resolutions.

While image registration is a computationally intensive problem, it can be readily accelerated by exploiting parallelism. Enhancements that focus on acceleration through parallelism can be binned into levels based on the basic unit of computation considered. Each of these must use a parallel platform as the target to exploit the exposed parallelism. These platforms support a standard parallel processing approach, a few of which are covered in the next section.

MULTICORE PLATFORMS

Many multicore systems are viable acceleration platforms for medical image registration. They vary in a variety of dimensions including number of processing elements, size and hierarchy of memory, the bandwidth and topology of on-chip interconnect, single-chip versus multichip, and specialized instructions or coprocessors [3]. Perhaps most importantly for this work, these high-performance multicore platforms expose different programming models, which exhibit different threading models, memory models, and even different language constructs for utilizing platform intrinsics. In this section, we discuss a few commonly used options that are applicable to image registration.

Message passing interface (MPI) is a popular standard for explicitly parallelizing code on multiprocessor systems. Threads have local memory that can be readily implemented on distributed memory platforms such as clusters. Threads then exchange data across the cluster with explicitly defined communication links. Because communication may be over relatively long latency links, threads tend to be more loosely coupled, which lends MPI to being utilized at the highest levels of parallelism. As an example, to employ MPI for medical image registration, gradient computation of the similarity measure can be distributed equally across nodes in a cluster [4]. This distribution is possible by virtue of the fact that each finite difference calculation for each control point is independent and requires only the neighboring voxels and control points to calculate.

A GPU is an array of processing elements customized for pixel processing. The increasing programmability of GPUs have made them excellent candidates for many other applications including image registration [5], [6], [17]. High-level languages are emerging to aid the task of programming GPUs such as NVIDIA's Compute Unified Device Architecture (CUDA) [7]. The GPU programming models export the architecture as a large number of lightweight threads. With CUDA, threads are grouped into blocks that may coordinate on one tightly clustered set of processing elements that are arrayed on NVIDIA GPUs. Some memory is shared while others are distributed, but each programming approach has explicit constructs to ensure high-speed input/output (I/O). As an independent streaming operation, the task could be efficiently distributed across the processing elements of the GPU.

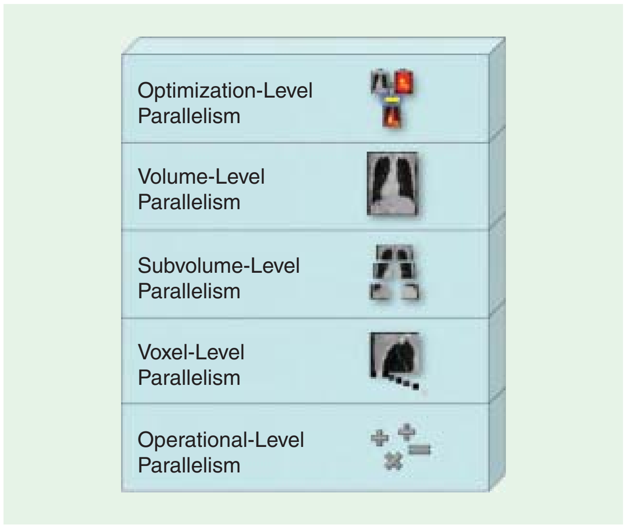
For the lowest level of parallelism, hardware description languages (HDLs) are often deployed. With HDLs, the final implementation is not destined for a processor, so designers lay out their application structurally, exposing interfaces and cycle-by-cycle control. A significant departure from traditional software programming languages, HDLs have no threading model and completely distributed memory structure. FPGAs can accelerate the voxel processing of transformation application and the similarity measure calculation in medical image registration [8]. The independence of tasks allows for many memory accesses, operations, and I/O to be performed in the same clock cycle.

Since many of these acceleration techniques are independent of each other and implemented on different types of platforms, a heterogeneous computational platform that supported all of these approaches would create a powerful new image registration engine. Orthogonal acceleration techniques such as CUDA and MPI techniques could provide multiplicative speedup effects. But to properly utilize such a heterogeneous multicore system, parallelism in the application domain must be identified and properly mapped to the target architecture.

IMAGE-REGISTRATION SPECIFIC TAXONOMY OF PARALLELISM

While acceleration techniques, in general, may modify functionality, we focus on the categorization of techniques that rely on parallelism for performance improvements. As with classical general-purpose categorizations, we classify acceleration techniques into "levels," which are depicted in Figure 2. Like the classical levels of parallelism (bit, data, instruction, task/thread, and process level), higher levels are specializations (or restricted forms) of lower levels. Our taxonomy can be mapped to classical levels in different ways (e.g., voxel-level parallelism could be implemented with data-level parallelism or task-level parallelism), but some mappings are impractical (e.g., optimization-level parallelism cannot be implemented with bit-level parallelism).

Conversely, it tends to be easier to reap the rewards of higher-level parallelism than lower. Many architectural and application factors affect this tendency both positively and negatively


Optimization-Level Parallelism
Volume-Level Parallelism
Subvolume-Level Parallelism
Voxel-Level Parallelism
Operational-Level Parallelism

[FIG2] Our domain-specific organization of parallelism.

(e.g., communication patterns, memory sizes, topology, and compiler performance), but for typical applications, the higher the level of parallelism expressed, the more readily applications can be accelerated.

OPTIMIZATION-LEVEL PARALLELISM

Optimization-level parallelism represents those parts of an algorithm that can run in parallel given the basic unit is an iteration of the image registration routine. Ino et al. [9] use this idea (which they call "speculative parallelism") to promote faster convergence in their time-critical registration application. Since the best optimization parameters are difficult to identify a priori, multiple instances of the same algorithm are launched with different parameters. Ultimately, after a specified time period, the best solution from these instances is selected. The multiple optimization instances require minimal communication and coordination and therefore are easy to execute in parallel.

Butz and Thiran [10] perform registration by utilizing a genetic algorithm in which fitness (or the metric of survival) is determined by how well the transformed image matches the fixed image. They implement this approach with an existing genetic solver parallelized using the MPI on a ten-node cluster. This optimization-level parallelism is naturally utilized because evaluating the population of solutions is an inherently independent act. With this class of parallelism, utilizing it is straightforward and can be efficiently implemented, but an individual optimization instance is not accelerated. For this, application designers must tap into opportunities at lower levels of parallelism.

VOLUME-LEVEL PARALLELISM

Volume-level parallelism is a generalization of optimization-level parallelism where the computational units operate on entire volumes. For example, an optimization iteration could be pipelined (applying one trial transform to the moving image while generating another candidate transform). Ino et al. [9] discuss the potential of "task parallelism" in accelerating the gradient computation of a rigid registration algorithm. This is possible, since independent finite difference calculations are done using the entire volume. While volume-level parallelism is simple to capture, its use is limited. For many algorithms, the number of independent, entire-volume calculations is small. Furthermore, distributing volumes to processing elements can suffer from high communication overhead. Lower levels of parallelism have tended to offer more opportunities for acceleration.

SUBVOLUME-LEVEL PARALLELISM

In medical image registration, subvolume-level parallelism is perhaps the most popular. In this approach, the computation is performed on subvolumes of image. Often designers can divide volume-level work into smaller subvolumes that are later recombined to produce the final solution. While this creates many opportunities for parallelism, it comes at the price of additional overhead such as coordinating how volumes will be split, managing overlap regions, and consolidating results.

Rohlfing and Maurer [11] employ subvolume-level parallelism for accelerating the similarity calculation. The volume is broken into equally sized sections such that a thread computes its local mutual histogram for mutual information (MI) and then merges its result into the global one. Ourselin et al. [12] use a block matching approach to find the deformation field. Inspired by video compression, the block matching technique compares "blocks" of one image against blocks of the other. These calculations are distributed across processors using MPI. In the same implementation, the authors accelerate image resampling with OpenMP. By distributing computation on individual multiprocessor machines, processes can share image memory and reduce the communication overhead incurred by transmitting images. They simultaneously utilize two programming paradigms to improve performance results.

Ino et al. [9] use "data parallelism" by distributing "small parts" of the image to subtasks that are assigned to different processors. Ino, et al. leverage this same level of parallelism in [4] by distributing the gradient computation of the similarity measure for control points across a distributed memory system. Such a distribution not only load balances the computation, but also reduces the memory requirements on an individual node.

Stefanescu et al. [13] parallelize the demons algorithm [14], which is based on optical flow, onto a 15-node cluster. The authors split the image into subvolumes to perform matching and filtering. Stenfanescu et al. [15] perform similar parallelization using a different registration technique. Subvolumes are assigned to different processors and communication is regularized over them. Hardware-based approaches can also utilize subvolume-level parallelism. Dandekar et al. [16] create an architecture in an FPGA that solves the registration problem recursively on subvolumes. Since each subvolume is an independent local registration problem, datapaths can be replicated for additional performance.

Greater effort has been applied to this level of parallelism to achieve speedups in medical image registration. This form is the most general flavor of parallelism and can be readily exploited by the most commonly used parallel platform clusters. While clusters are not optimally suited to lower levels of parallelism, researchers have been finding opportunities for parallelism at lower levels using different platforms.

VOXEL-LEVEL PARALLELISM

Voxel-level parallelism describes parallelism in terms of single voxels. In this case, the regional benefit of using subvolumes is not present, so application designers find parallelism in independent voxel computations. Strzodka et al. [17] implement a gradient flow algorithm optimized for a GPU. This algorithm maps well to a GPU as images are stored into texture memory and the operations used are supported by the GPU hardware. Warfield et al. [18] utilized a "workpile" of threads to process a voxel independent classification method. They used threads on a shared memory platform to accelerate the task. Voxel-level parallelism that cannot be modeled as subvolume parallelism turns out to be rare. But with the rise in popularity of GPUs, efforts that utilize this parallelism are likely to increase. The last level explored by designers is the parallelism present in processing an individual voxel.

OPERATION-LEVEL PARALLELISM

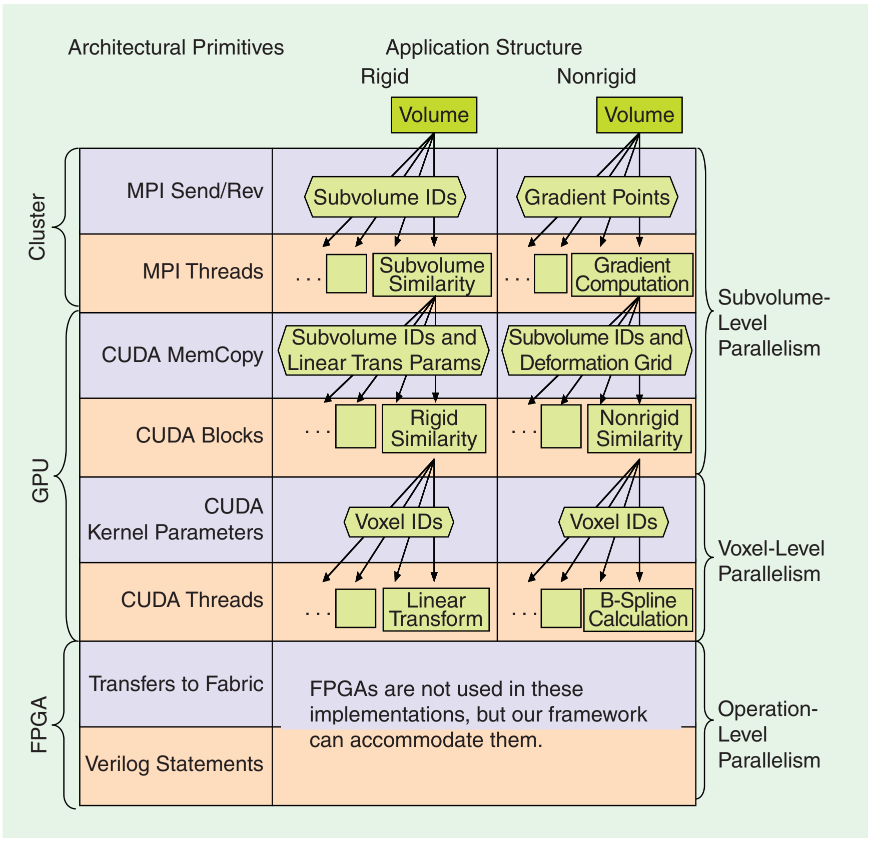
Operational-level parallelism is the lowest, most general form of parallelism. At this level, parallelism can be explored in many ways as the basic computational unit is no longer defined. Image-registration application designers have found parallel activities to accelerate when processing a single voxel. Castro-Pareja and Shekhar [8] construct an architecture that parallelizes the computation of transforming, interpolating, and computing the MI of voxels in an image in milliseconds. Designed in a HDL, it can perform MI-based rigid registration in about one minute. Beyond taking advantage of the instruction level parallelism transparently on a modern processor, operation level parallelism is the most difficult to utilize. Only custom hardware platforms are suitable to effectively exploit this level of parallelism.

STRUCTURED PARALLELISM IMPLEMENTATION

To evaluate the potential performance benefits of utilizing different levels of parallelism, we construct a design framework based on the image registration specific design taxonomy. If we were just dealing with individual platforms, we could simply implement each acceleration technique into the code base as necessary. But since we want to experiment with combined approaches, we start by expressing different types of parallelism in a structured fashion. After exposing and categorizing application parallelism, we map these to architectural primitives as presented by the programming models of our respective targets.

Mapping of parallelism to architectural resources requires insight about the application and architecture. Future work would assist in automating this procedure, but for now we rely on designer guidance. Once mapped, we employ the tools and design principles specific to the target platform component. For instance, a GPU's array of pixel-processing elements is often abstracted by the programming model as a set of threads with a language like CUDA. By using the target-specific programming environment, we exploit an efficient compilation path to the target with direct access to platform intrinsic crucial for performance. The development experience is also enriched through debuggers, visualization engines, and simulation environments.

For example, Figure 3 depicts both rigid and nonrigid applications each represented by a tree. Each registration algorithm is mapped to a hierarchical multiprocessing platform: a set of hosts networked together with MPI where each host has a GPU acceleration based on the CUDA programming model. The location of each application module indicates which computational resource it is mapped to. For instance, the "Linear Transformation" box in the "Rigid" application pictorially represents an assignment of a rigid registration's linear transformations to CUDA threads. Similar to the denotation of computational mapping, we represent the system mapping of application communication to an architectural primitive. For a nonrigid registration algorithm, the gradient



[FIG3] A rigid and a nonrigid registration algorithm described in our framework targeting a hierarchical multiprocessing platform with graphics processors in a cluster.

computation for a free-form deformation (FFD) grid can be readily accelerated using a general-purpose cluster as well as a GPU.

To adhere to our hierarchical multiprocessing approach, the code was written to maintain the interfaces described by the structured mapping of parallelism. For example in Figure 3, subvolumes are sized to ensure that they are properly divisible inside an MPI thread. These interfaces allow for methodical changes of both the platform and the user interface.

EVALUATION

To evaluate our implementation framework, we choose a representative algorithm and high-performance multicore implementation vehicles.

ALGORITHM

Based on the structure of the framework and insights of the previous section, we implement the same image-registration algorithm on a single GPU and a GPU cluster. For rigid registration, the optimization method is based on downhill simplex [19] and the similarity measure is MSD with nearest-neighbor interpolation. The nonrigid algorithm is based on Rueckert et al.'s method [20] with an FFD grid utilizing B-spline interpolation between control points and trilinear interpolation between voxels.

For our rigid algorithm, parallelism comes from the independence of the MSD calculation performed on separate subvolumes. Large subvolumes are a good match to the granularity of MPI nodes and small volumes map naturally to CUDA blocks (an abstraction of the GPU pixel multiprocessors), so both the GPU and the cluster can exploit the similarity measure calculation parallelism. Each could be used for a single acceleration platform, but we combine them by constructing the MPI suvolumes to be large enough to be divided into smaller subvolumes used by the GPU.

For our nonrigid implementation, we utilize the subvolume-level parallelism of the gradient calculation for each control point in MPI. The gradient calculation using finite difference requires multiple similarity measure calculations with the addition of B-spline interpolation of control points to determine local deformation. A GPU can effectively accelerate this calculation by utilizing cooperative multithreading: mapping plane interpolation to a set of threads, row interpolation to a subset of the same threads, and finally the point interpolation to a single thread. As with rigid registration, the separation of these two parallelism constructs inside a structured framework allows us to utilize them on individual platforms as well as on a combined hierarchical multiprocessing platform.

EXPERIMENTAL SETUP

We based our experimental implementations on a single-threaded code base utilizing double precision floating point computations. Using this single-code base, we incorporated acceleration techniques wrapped by preprocessing directives. At compile time the software could be targeted for a specific parallel platform. The considered parallel platforms are as follows:

- a single GPU-NVIDIA GeForceGTX 285 with 1 GB of RAM targeted with CUDA SDK 2.2
- a GPU cluster: Four GPUs in separate PCs connected via gigabit Ethernet with the structure described in Figure 3.

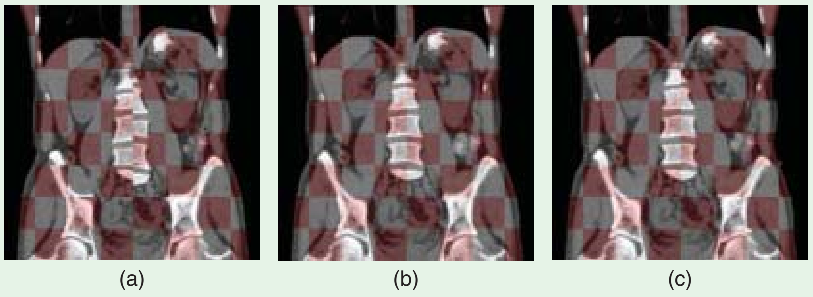
The GPU implementations utilized single precision floating point calculations to optimize performance on the platform. For rigid registration, the implementations were profiled with five pairs of CT images of the torso where the translation and rotation vectors were known. Each image was $256 \times 256 \times 256$ with 8 b representing voxel intensity. The deformation parameters were determined at random for each case and the rigid and nonrigid registration cases were separate so that we could study both scenarios individually. There was no rigid misalignment in the nonrigid registration cases and no nonrigid misalignment in the rigid registration cases. The rotation ranged between $-25°$ and $25°$ on each axis and between -25 mm to 25 mm in each dimension. With nonrigid registration, five new pairs were created by deforming a torso with a grid of size $5 \times 5 \times 5$ overlaid. Each control point was randomly moved in each dimension by up to 2 cm in either direction. The grid used to correct this was of the same size. The image voxel size was $1.38 \times 1.38 \times 1.5$ mm. The algorithm stopped when a minimum step size was reached at which there was no improvement. The downhill simplex and gradient descent parameters (such as starting position, initial step size, and stopping criterion) were held constant across all cases and implementations.

We constructed MPI subvolumes equal to the number of nodes in the cluster that made them large enough to be divided into GPU subvolumes to match the GPU blocks. The GPU block dimensions were $8 \times 8 \times 4$ for rigid registration and $4 \times 4 \times 4$ for nonrigid registration. In general, larger blocks are more beneficial to performance since more threads are available to keep GPU utilization high, but in our case, nonrigid blocks were smaller because more resources are used for the nonrigid registration similarity measure calculation. This limits the number of threads that can be assigned to one processing element of the GPU.

RESULTS

Rigid registration results were of high quality for the GPU accelerated implementation, while nonrigid registration results vary as shown visually by one case in Figure 4, in which the moving and fixed images are tiled together in a checkerboard fashion. A perfectly registered result should show no misalignment at the tile boundaries. By observing the alignment of the checkerboard at the spine, one can see the improvement of the GPU accelerated result and the original result. Both greatly improve on the initially nonrigidly unaligned image, but the original result is superior to the GPU accelerated result. This is due to the fact that our GPU implementation currently uses single precision floating point arithmetic optimized for graphics, while the original implementation utilizes full double precision math. As a result, the GPU accelerated implementation is unable to arrive at the same quality of solutions in nonrigid registration

A summary of the test results is shown in Table 1. Since the images were artificially deformed with a known deformation field, we calculated the average distance between the known and recovered deformation over all voxels. As a point of comparison, the original unaccelerated implementation achieved an average accuracy of 0.1 mm and 0.6 mm for the rigid and nonrigid cases, respectively. Note that the single GPU and the multiple GPU solution produced equivalent registration accuracy as MPI does not change the behavior of our implementation. In an effort to produce a clearer picture of how well the kernels perform on the GPU, the timing results reported in Table 1 do not include the time for initialization, file I/O, or the one time image loads into GPU memory. The total time for this overhead takes under two seconds, most of which could be amortized by new images streaming into the GPU during the registration of prior images. Considering acquisition and reconstruction time of intraprocedural medical images, both rigid registration GPU implementations are feasible in a clinical setting by producing a new aligned image in just a few seconds. Clinicians using this platform during a procedure would see aligned preprocudural images refresh every few seconds based on the changes captured by the intraprocedural images, which would provide meaningful, timely guidance for a variety of procedures.



[FIG4] Example registration (Case 1) of an image and its nonrigidly deformed version fused with a checkerboard pattern: (a) uncorrected, corrected with the original, (b) unaccelerated CPU implementation, and (c) corrected with the implementation with GPU acceleration.

DISCUSSION

The single GPU significantly improve the performance of the original code base, in both the rigid and nonrigid registration cases. In each of these implementations, the performance improvement is derived from structuring the application so that the code can be methodically targeted to hierarchical multiprocessing platforms. We observed performance derived from this GPU implementation comes at the expense of inaccuracy over the original implementation comes from the limited floating point precision present in the GPU. When a control point is varied for its finite difference calculation, it makes only a minor change in the similarity measure. Even though the GPU approximates well most of double precision finite difference calculations, some subset of them is poorly estimated during each step. Since all points advance simultaneously after the gradient is calculated, even a few wayward control points can significantly skew the similarity measure between the fixed and moving image, inhibiting the overall convergence.

SUMMARY

Hierarchical multiprocessing offers the potential of significant performance improvement to some compute intensive applications, but it is accompanied by new design challenges including finding and exploiting parallelism. In this work, we discussed our approach to utilizing hierarchical multiprocessing in the context of medical image registration. By first organizing application parallelism into a domain-specific taxonomy, we structured an algorithm to target a set of multicore platforms. We demonstrated the approach on a cluster of GPUs requiring the use of two parallel programming environments to achieve fast execution times. There is negligible loss in accuracy for rigid registration when employing GPU acceleration, but it does adversely effect our nonrigid registration implementation due to our usage of a gradient descent approach.

Towards our goal of robust real-time registration, we believe that the advantages of GPU and multi-GPU acceleration could be reaped by running different phases of image registration on different platforms (e.g., using GPU acceleration first for a fast, coarse solution, and then not using it for more accuracy towards the end, as the imaging scenario would permit). Alternatively, a different algorithm could be employed for the GPU that would be less sensitive to precision effects or utilizing double precision floating point units now on high-end GPUs. We believe the structured approach presented here will enable our continued exploration into these and other implementations.

As we consider more complex acceleration techniques to combine, a robust system of capturing the parallelism of the application will be needed. Programming with formal underpinnings would give programmers a more natural way of expressing each type of parallelism without having to dive into low-level, idiosyncratic GPU languages, for example.

[TABLE 1] SPEED AND ACCURACY RESULTS OF RIGID REGISTRATION AND NONRIGID REGISTRATION, AVERAGED OVER FIVE SEPARATE CASES EACH.

	PLATFORM	AVERAGE ACCURACY (MM)	AVERAGE NUMBER OF ITERATIONS	AVERAGE TIME PER REGISTRATION (S)
RIGID CASES	ONE GPU	0.10	313	7.9
	FOUR GPUs	0.10	313	2.5
NONRIGID CASES	ONE GPU	2.43	12.6	250
	FOUR GPUs	2.43	12.6	98

AUTHORS

William Plishker (plishker@umd.edu) graduated in 2006 from the University of California at Berkeley with a Ph.D. degree in electrical engineering. He is a post-doctoral researcher at the University of Maryland. His focus is on application acceleration using dataflow modeling and leveraging different forms of parallelism. He has published papers on new dataflow models and scheduling techniques as well as application acceleration on multiple platforms including clusters, GPUs, FPGAs, and network processors. His application areas of interest include medical imaging, software defined radio, networking, and high energy physics. His Ph.D. research centered around the acceleration of network applications on network processors.

Omkar Dandekar (dandekar@ieee.org) received the B.E. degree in biomedical engineering from the University of Mumbai, India, in 2000, the M.S. degree in electrical engineering from The Ohio State University, Columbus, in 2004, and the Ph.D. degree in electrical and computer engineering from the University of Maryland, College Park, in 2008. He is currently a senior engineer with Intel Corporation, Hillsboro, Oregon. He has previously worked as a graduate research assistant at the University of Maryland, School of Medicine and at the Cleveland Clinic Foundation. His primary research interests include medical imaging, digital VLSI design, and hardware acceleration of image processing algorithms, with special focus on real-time 3-D imaging and advanced image processing and analysis for image-guided interventions. He has published over 30 refereed technical articles in this field. Currently, his research work is focused on computational lithography and patterning techniques for advanced technology nodes.

Shuvra S. Bhattacharyya (ssb@umd.edu) received the B.S. degree from the University of Wisconsin at Madison and the Ph.D. degree from the University of California at Berkeley. He is a professor in the Department of Electrical and Computer Engineering University of Maryland, College Park. He holds a joint appointment in the University of Maryland Institute for Advanced Computer Studies and an affiliate appointment in the Department of Computer Science. He is the coauthor/coeditor of five books and the author/coauthor of more than 150 refereed technical articles. His research interests include signal processing systems, architectures, and software; biomedical circuits and systems; embedded software; and hardware/software codesign. He has held industrial positions as a researcher at the Hitachi America Semiconductor Research Laboratory (San Jose, California), and compiler developer at Kuck and Associates (Champaign, Illinois).

Raj Shekhar (rshekhar@umm.edu) received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Kanpur, in 1989, the M.S. degree in bioengineering from Arizona State University, Tempe, in 1991, and the Ph.D. degree in biomedical engineering from The Ohio State University, Columbus, in 1997. He worked as a senior researchengineer for Picker International (now Philips Healthcare) for two years before joining the Department of Biomedical Engineering, Cleveland Clinic Foundation, Ohio, in 1998. He is an associate professor of diagnostic radiology at the University of Maryland School of Medicine. He is also an affiliate professor of bioengineering and electrical and computer engineering. He has been a researcher and innovator in the field of medical imaging for over ten years, during which time he has published over 50 refereed technical papers. His research interests include medical imaging, image processing, platform acceleration, and image-guided interventions.

REFERENCES

[1] W. Plishker, O. Dandekar, S. S. Bhattacharyya, and R. Shekhar, "Towards systematic exploration of tradeoffs for medical image registration on heterogeneous platforms," in *Proc. IEEE Biomedical Circuits and Systems Conf.*, Baltimore, MD, Nov. 2008, pp. 53–56.

[2] W. Plishker, O. Dandekar, S. S. Bhattacharyya, and R. Shekhar, "A taxonomy for medical image registration acceleration techniques," in *Proc. IEEE-NIH Life Science Systems and Applications Workshop*, Bethesda, MD, Nov. 2007, pp. 215–218.

[3] G. Blake, R. G. Dreslinski, and T. Mudge, "A survey of multicore processors," *IEEE Signal Processing. Mag.,* vol. 26, no. 6, pp. 26–37, Nov. 2009.

[4] F. Ino, K. Ooyama, and K. Hagihara, "A data distributed parallel algorithm for nonrigid image registration," *Parallel Comput.*, vol. 31, no. 1, pp. 19–43, 2005.

[5] R. Shams and N. Barnes, "Speeding up mutual information computation using NVIDIA CUDA hardware," in *Proc. Digital Image Computing: Techniques and Applications (DICTA)*, Adelaide, Australia, Dec. 2007, pp. 555–560.

[6] R. Shams and R. A. Kennedy, "Efficient histogram algorithms for NVIDIA CUDA compatible devices," in *Proc. Int. Conf. Signal Processing and Communications Systems (ICSPCS)*, Gold Coast, Australia, Dec. 2007, pp. 418–422.

[7] NVIDIA, *NVIDIA CUDA, Compute Unified Device Architecture, Programming Guide*. 2007.

[8] C. R. Castro-Pareja and R. Shekhar, "Hardware acceleration of mutual information-based 3D image registration," *J. Imaging Sci. Technol.*, vol. 49, no. 2, pp. 105–113, 2005.

[9] F. Ino, Y. Kawasaki, T. Tashiro, Y. Nakajima, Y. Sato, S. Tamura, and K. Hagihara. "A parallel implementation of 2-D/3-D image registration for computer-assisted surgery," *Int. J. Bioinformatics Res. Appl.,* vol. 2, no. 4, pp. 341–358, 2006.

[10] T. Butz and J.-P. Thiran, "Affine registration with feature space mutual information," in *Medical Image Computing and Computer-Assisted Intervention*, vol. 2208 (Lecture Notes in Computer Science), W. J. Niessen and M. A. Viergever, Eds. Berlin, Germany: Springer-Verlag, 2001, pp. 549–556.

[11] T. Rohlfing and C. R. Maurer, "Nonrigid image registration in shared-memory multiprocessor environments with application to brains, breasts, and bees," *IEEE Trans. Inform. Technol. Biomed.,* vol. 7, no. 1, pp. 16–25, 2003.

[12] S. Ourselin, R. Stefanescu, and X. Pennec, "Robust registration of multi-modal images: Towards real-time clinical applications," in *Proc. Medical Image Computing and Computer-Assisted Intervention (MICCAI'02)* (Lecture Notes in Computer Science), 2002, pp. 140–147.

[13] R. Stefanescu, X. Pennec, and N. Ayache, "Parallel non-rigid registration on a cluster of workstations," in *Proc. HealthGrid*, 2003.

[14] J. P. Thirion, "Non-rigid matching using demons," in *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'96)*, San Francisco, CA, USA, 1996, pp. 245–251.

[15] R. Stefanescu, X. Pennec, and N. Ayache, "Grid powered nonlinear image registration with locally adaptive regularization," *Med. Image Anal.*, vol. 8, no. 3, pp. 325–342, 2004.

[16] O. Dandekar and R. Shekhar, "FPGA-accelerated deformable image registration for improved target-delineation during CT-guided interventions," *IEEE Trans. Biomed. Circuits Syst.,* vol. 1, no. 2, pp. 116–127, 2007.

[17] R. Strzodka, M. Droske, and M. Rumpf, "Fast image registration in DX9 graphics hardware," *J. Med. Inform. Technol.*, vol. 6, pp. 43–49, 2003.

[18] K. Warfield Simon, A. J. Ferenc, and R. Kikinis, "A high performance computing approach to the registration of medical imaging data," *Parallel Comput.*, vol. 24, no. 9–10, pp. 1345–1368, 1998.

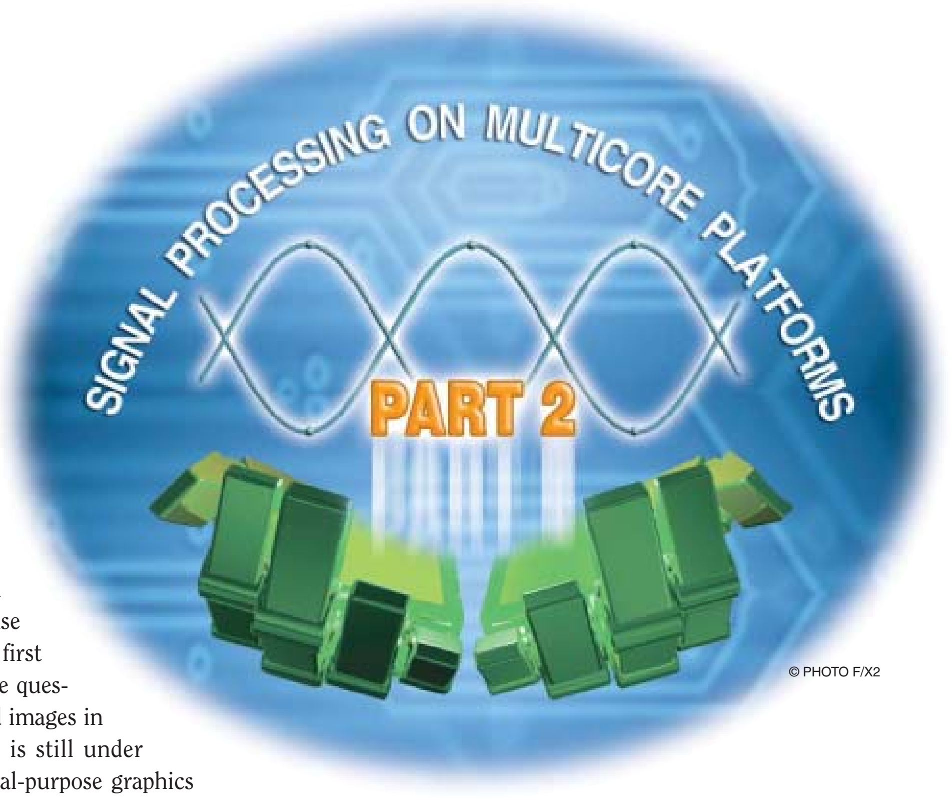
[19] A. Nelder and R. Mead, "A simplex method for function minimization," *Comput. J.,* vol. 7, no. 4, pp. 308–313, 1964.

[20] D. Rueckert, L. I. Sonoda, C. Hayes, D. L. G. Hill, M. O. Leach, and D. J. Hawkes, "Nonrigid registration using free-form deformations: Application to breast MR images," *IEEE Trans. Med. Imag.,* vol. 18, no. 8, pp. 712–721, 1999. [SP]

[Maurizio di Bisceglie, Michele Di Santo, Carmela Galdi, Riccardo Lanari, and Nadia Ranaldo]

Synthetic Aperture Radar Processing with GPGPU

[Focusing on the methodologies behind the technicalities]



© PHOTO F/X2

Synthetic aperture radar (SAR) processing is a complex task that involves advanced signal processing techniques and intense computational effort. While the first issue has now reached a mature stage, the question of how to produce accurately focused images in real time, without mainframe facilities, is still under debate. The recent introduction of general-purpose graphics processing units (GPGPUs) seems to be quite promising in this view, especially for the decreased per-core cost barrier and for the affordable programming complexity.

This article focuses on methodologies with recurrent use to code examples that try to couple with the flow of the main steps of the SAR processing. The possibility to be comprehensive was prevented by the wide scenario of variations of the focusing algorithm as well as the spread of applications. The reader should look at this work as a sample of possibilities offered by this new technology and a collection of suggestions and considerations that may guide to new applications and horizons.

Digital Object Identifier 10.1109/MSP.2009.935383

INTRODUCTION

The elegance of Earth's view shown by orbiting satellites appears even inessential if compared with the formidable volume of information that is gathered by such a wonderful sight. The observational capability is extended to the microwave region of the electromagnetic spectrum, where spaceborne SAR systems offer the best way to achieve fine spatial resolution, long-term global coverage and short revisit time. The interest of a wider and wider community in gathering SAR data is confirmed by the increasing number of existing and recently proposed platforms (for example, see TerraSAR-X, RadarSAT-2, Cosmo-SkyMed, Palsar, Sentinel-I, and Biomass) that operate, or will operate, at nearly all wavelengths within the microwaves: from P-band (when foliage penetration is required) to Ka-Band (when the goal is high resolution and lightweight).

Applications are tailored to specific observational programs including disaster observation and management, geological mapping, snow/ice mapping, mapping of renewable resources, and strategic surveillance of military sites. In the next decade, a significant leap forward will be enabled by the assimilation of multitemporal data products into complex computational models working at a regional or global scale: seismic studies, monitoring of the buildings stability, and soil moisture evaluation for flood monitoring are enlightening examples where some weak correlation patterns cannot be displayed if space and time observations are not jointly used [1]. The preeminence of these applications drives the need for high-performance computing of SAR data, often under real-time constraints.

Many-core platforms and related architectures are the general frameworks where intensive applications can be generated and optimized. Works deriving from the SPIRAL framework demonstrate that orders-of-magnitude performance increase can be obtained using commercial off-the-shelf architectures such as Cell BE or multicore platforms [2]–[4]. At the same time, it is apparent that such achievements cannot be obtained without proper domain-specific languages with high abstraction level or, preferably, with automatic generation of high-performance codes.

With the recent introduction of GPGPU, the designer is faced with units that exceeds CPU performance in terms of raw processing power and this trend is expected to continue for some time. Consequently, workstations equipped with GPGPUs are becoming the platforms of choice for low-cost supercomputing [5], [6]. To achieve the best speedups from GPU mapping, programmers are required to select applications naturally emphasizing parallelism and maximizing arithmetic intensity (the number of arithmetic operations per memory access), but also to choose algorithms that cut up computations in great numbers of independent batches, to structure a code so as to minimize incoherent branching within the threads of each single batch and to optimize the masking of memory latency [5].

[THE QUESTION OF HOW TO PRODUCE ACCURATELY FOCUSED IMAGES IN REAL TIME, WITHOUT MAINFRAME FACILITIES, IS STILL UNDER DEBATE.]

De facto, many SAR image and product formation steps are independent and separable, so they turn out to be good candidates for massively parallel computation on GPUs. Along the hierarchy of SAR processing procedures, a considerable amount of processing is required to generate single-look complex images from the acquired raw data (readers should agree that raw images are quite unimpressive). This stage of processing requires forward/backward fast Fourier transforms (FFTs) in one or two dimensions (depending on the philosophy of choice) and a number of filtering and interpolation steps. As displayed in Table 1, the overall computational charge per orbit appears relevant.

This matter is, however, rapidly evolving in many directions. Better architectures will be discovered but, nonetheless, data will become spatially and temporally more continuous. However, as A.M. Turing says at the end of his paper "Computing Machinery and Intelligence," [22] "We can only see a short distance ahead, but we can see plenty there that needs to be done."

SAR GEOMETRY AND PROCESSING ARCHITECTURES

A significant role in the theory of SAR imaging is played by the acquisition geometry, including here the relevant signal features induced by the platform motion. The antenna system, looking across the flight direction, transmits a short, chirped waveform, $f(t)$, with a pulse repetition time 1/PRF much longer than the waveform duration; the echoes backscattered by the earth's surface are received through the antenna pattern and digitized line by line at each platform position. The equirange surfaces are concentric spheres whose intersection with the flat terrain generates concentric circles. Surfaces with identical Doppler shift are coaxial cones with the flight line as the axis and the radar platform as the apex. The intersection of these cones with the flat terrain generates hyperbolas. Objects lying along the same hyperbole will provide equi-Doppler returns.

The most natural reference geometry is perhaps the cylindrical system where the axis coincides with the flight trajectory (supposed to be a straight line) [7], [8] and the other two coordinates are the range from sensor to target and the viewing angle (see Figure 1). The acquisition geometry makes the SAR

[TABLE 1] COMPARISON OF MISSION PARAMETERS FOR RECENT AND FUTURE SAR SYSTEMS.

INSTRUMENT OR MISSION	AZIMUTH RESOLUTION	CARRIER FREQUENCY	MISSION OBJECTIVE	DATA RATE	RANGE SAMPLES
BIOMASS	12.5 [M]	P-BAND	OBSERVATION OF GLOBAL FOREST BIOMASS	100 GB / ORBIT	N/A
ALOS-PALSAR	7–100 [M]	L-BAND	EARTH OBSERVATION	200–400 GB/DAY	688–11,488
RADARSAT-2	3–100 [M]	C-BAND	USE OF SAR DATA FOR COMMERCIAL USE	UP TO 200 GB/ORBIT	8,000–16,000
SENTINEL- I	5–80 [M]	C-BAND	GLOBAL MONITORING FOR ENVIRONMENT AND SECURITY	60 GB/ORBIT	N/A
COSMO-SKYMED	1–100 [M]	X-BAND	EARTH OBSERVATION AND DEFENSE	560 GB/DAY	N/A
TERRASAR-X	1–18 [M]	X-BAND	EARTH OBSERVATION AND SATELLITE INTERFEROMETRY	70 GB/DAY	N/A

raw data processing operation (often referred to as focusing or compression) an intrinsically two-dimensional (2-D) and space-varying problem. To understand this concept we should consider the response from a single point placed on a nonreflecting surface. Suppose that, at a time t', the radar is in the position of closest approach to a point scatterer of coordinates (x, r) in the cylindrical reference. At each acquisition, the radar moves along track, of a step v/PRF, where v is the sensor-target relative velocity, and at a new position, say $(vt', 0)$, the acquisition time of the leading edge is changed. Actually the acquisition time shortens when the radar moves toward the point, lengthens when the radar moves forward. The range $R(t')$ from radar to target can be expressed as

$$R(t', x, r) = [r^2 + (vt' - x)^2]^{1/2},$$

which, under the approximation that the antenna aperture is not too wide, can be used to define the range migration term as $\delta R(t', x, r) = R(t', x, r) - r \simeq (vt' - x)^2/2r$ [7]. The generation of a SAR image corresponds to the coherent combination of the backscattered echoes of each target, collected within the synthetic aperture, accounting for the phase changes due to the sensor-to-target distance variations and for the range migration effect. The range dependence of the synthetic aperture and the 2-D characteristics of the range migration effect makes the SAR image formation process a 2-D space-variant (range-dependent) problem.

To provide a sufficiently general statement of the SAR acquisition and processing problem, define

- an object space as the 2-D domain where the scatterers having a reflectivity $\gamma(x, r)$ are represented. The cylindrical coordinate system (x, r) is here assumed to represent a point (the viewing angle is omitted because it does not contribute to the sensor-target distance)
- a data space as the domain where the received data $u(t, t')$ are represented as a function of t (the fast time) and t' (the slow time)
- an image space as the space of the processed SAR data $I(t, t')$.

The SAR signal acquisition is a mapping from the object space to the data space. It is the space-variant superposition of returns from elementary scatterers defined by

$$u(t, t') = \iint_{\mathbb{R}^2} \gamma(x, r) f(t - 2R(t', x, r)/c)\, dx dr. \qquad (1)$$

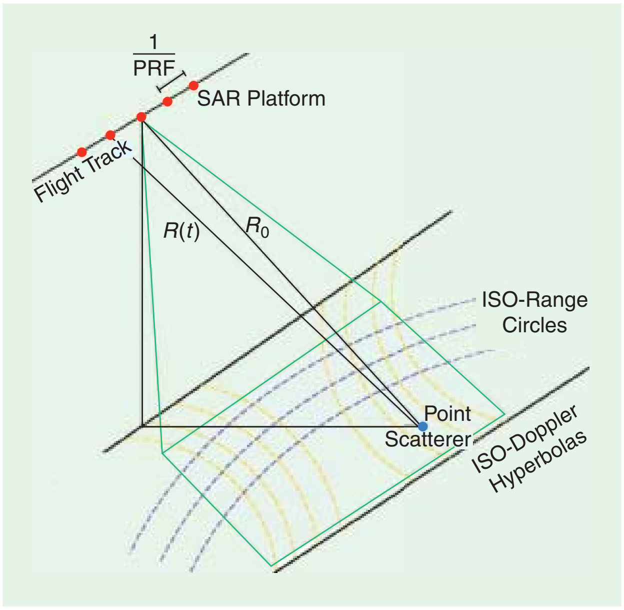
There are some major questions emerging from the model (1). How well can the kernel $f(t - 2R(t', x, r)/c)$ be approximated in a computationally efficient scheme? How well can the space continuity be accommodated in the discrete data representation (otherwise, what is the impact of interpolation)? What is the role played by platform parameters with respect to the overall goal of accuracy and computational efficiency? The relevant works of the last two decades give us a clear perspective opening us a view over the next two major classes of algorithms.

A SIGNIFICANT ROLE IN THE THEORY OF SAR IMAGING IS PLAYED BY THE ACQUISITION GEOMETRY, INCLUDING HERE THE RELEVANT SIGNAL FEATURES INDUCED BY THE PLATFORM MOTION.

1) The range-Doppler (RD) algorithm and its variations [9]. The RD algorithm was the first developed SAR processor and is probably still the most widely used SAR focusing technique. In the first stage, the range compression is applied; it is a matched filtering in the range direction, which may involve a partial correction of the range cell migration (RCM). The second stage, the azimuth focusing, is a range dependent matched filtering embedding a partial or total RCM correction. It includes

- Azimuth FFT. A set of one-dimensional (1-D) FFT in the azimuth direction; the resulting data lie in the RD plane.
- RCM correction. In the RD domain, variations of the point-to-radar distance induce a range-dependent Doppler frequency shift in the received signal. This stage includes range shifts, complex multiplications and interpolations.
- Azimuth compression. A matched filtering obtained by multiplying the azimuth transformed data by the range-dependent azimuth reference function in the frequency domain.
- Azimuth IFFT. The inverse 1-D set of FFTs to obtain the final image.

2) The $\omega - k$ and the fully 2-D algorithms. This class of algorithms performs the processing in the frequency domain and the related techniques are often referred to as $\omega - k$ [10] and 2-D frequency domain [11] algorithms. The original idea underlying the $\omega - k$ algorithm derives from the field of wave



[FIG1] **SAR system geometry.**

propagation while the 2-D frequency domain approach is based on the stationary phase methods. Essentially, these algorithms rely on the following major steps:

- A 2-D FFT.
- Compensation of the system transfer function at mid range. This stage is often referred to as *bulk compression*.
- A remapping in the frequency-wavenumber or in the 2-D-frequency domain through a focusing operator (Stolt mapping) compensating for the range-dependent component of the system transfer function. The change of variables requires an interpolation stage that is more complicate than that required by RCM correction in the RD algorithm. However, the Stolt mapping corrects all major 2-D signal range dependencies.
- A 2-D inverse FFT (IFFT).

As final remark we underline that further implementations of the $\omega - k$ and 2-D frequency domain algorithms have been more recently proposed, that mainly allow to achieve high computational efficiency by applying the chirp scaling [12] and chirp z-transform [13] techniques.

MANY-CORE PLATFORMS AND RELATED ARCHITECTURES ARE THE GENERAL FRAMEWORKS WHERE INTENSIVE APPLICATIONS CAN BE GENERATED AND OPTIMIZED.

DESIGNING THE GPU CODE

Graphics cards, originally designed as accelerators for computer graphics, have gradually evolved to support general-purpose computations, attracting the interest of scientific application developers as commodity, low-cost, and high-performance data-parallel coprocessors. Therefore many-core GPU-based platforms are becoming the enabling technology for massively parallel processing of SAR data, so that the most accurate processing techniques will be rapidly available for real-time (possibly airborne) applications at a low cost [14].

An important contribution in this direction is the recent introduction of some programming environments whose architectures offer a layer of abstraction between application programmers and modern GPUs, developed with the aim to reduce GPU programming complexity without sacrificing performance. Among these, we find two main actors, both adopting the ANSI C programming language with extensions for expressing data-parallel functions:

a) Compute Unified Device Architecture (CUDA) from Nvidia, a parallel computing architecture developed to exploit the power of GPUs as massively data parallel coprocessors with their own memory systems [15]

b) Open Computing Language (OpenCL), an open standard for general-purpose parallel programming intended for a portable and efficient access to the power of heterogeneous processing platforms [16].

Both CUDA and OpenCL adopt a single program multiple data (SPMD) programming model providing a level of abstraction for GPU hardware architectures. In particular, CUDA extends the C programming language and allows the definition of functions, called kernels, each one to be executed in parallel by multiple, extremely lightweight threads running on the processors of a GPU and organized into a computational grid of thread blocks. At each kernel call, the program can specify how many threads per block and how many blocks are to be launched. Blocks and grids may be structured into one, two or three dimensions and each block of the grid and each thread of a block have associated unique identification numbers. The threads of a block can synchronize their activities through an extremely lightweight barrier primitive, while an implicit barrier synchronization serializes the execution of successive kernels. The blocks of a grid must be executable in any order, in parallel or in series. Memory has a hierarchical organization: every thread has a private local memory and every block has a shared memory accessible to all its threads. Moreover, all the threads of a program have access to a global memory, a read-only constant memory, and a read-only texture memory.

At a lower abstraction level, when a CUDA kernel is called, the blocks of its computational grid are distributed for execution to the available GPU multiprocessors, with all threads of a single block resident on a single multiprocessor. Therefore, the number of blocks running in parallel is limited by the number of available multiprocessors. Within every multiprocessor, a number of scalar processor (SP) cores execute threads, further organized into groups called warps, by using a single instruction multiple thread (SIMT) execution model. Full efficiency is achieved when there is no branch diversion via a data-dependent conditional branch, as then all the threads in a warp, at each issue time, fetch and execute the same instruction. Instead, when there is a diversion, the different paths are serially executed, with a loss of efficiency.

Even exploiting the described programming abstractions, the development of parallel applications to be efficiently executed on GPUs is much more difficult than writing sequential programs. Following [5], programmers should try to think in terms of structures and hierarchies, having in mind a work plan and some caveats. Some practical design guidelines are the following.

1) Identify the computational domain of interest as a structured hierarchy of threads that are executable in parallel.

2) Write SPMD kernels, by remembering that the code of a kernel describes the actions to be executed by each single thread in the computational grid created at kernel call.

3) Carefully plan memory usage and data transfers among the different levels of memory hierarchy. Global memory has a high latency (hundreds of clock cycles) and is prone to access conflicts when many threads work on the same data, but its performance can improve even of an order of magnitude when accesses from groups of threads meet alignment and ordering criteria, a feature referred to as coalescing. Moreover, data transfers from global memory to shared memory should be considered when the global memory is accessed multiple times by many threads.

4) Accurately manage the divergences within each kernel code. As we shortly described, to avoid efficiency

penalties, code should be organized so that threads in the same warp do not give rise to divergent conditional branches or at least minimize their number. Therefore, the complexities of SIMT thread execution can be largely ignored by programmers, but only if they give up the idea of obtaining peak performance.

Following the concept scheme of the RD algorithm, the main aspects of the code design will be reviewed by discussing three code projects where the design strategies are analyzed and explained.

THE SAR SIGNAL ACQUISITION IS A MAPPING FROM THE OBJECT SPACE TO THE DATA SPACE.

CODE PROJECT 1 (FROM RAW DATA TO THE RD REPRESENTATION)

This algorithm naturally decomposes in a large set of independent computations. To obtain the best performance from a GPU, programmers have to find their own way to make the best use of resources in each single case. In particular, with respect to the parallelisms degree, as described above, the computation launched by a kernel call is distributed in a grid of thread blocks, each block containing the same number of threads, all executing the code of the kernel. The block and grid dimensions, that are collectively known as the execution configuration, can be set at kernel invocation and are typically based on the size and dimensions of the data to be processed.

With respect to GPU memory resources, the programmer can decide to exploit different memory spaces, taking into account their different scope, dimension, and bandwidth. In particular, beyond the per-streaming multiprocessor register file (that contains a large number of registers dynamically allocated to threads), and a slower per-thread local memory automatically used when local registers are not sufficient, the threads of a block can allocate data in a small dimension, high bandwidth per-block shared memory. While threads in a block can cooperate through the shared memory and synchronize with a barrier primitive, different blocks execute independently and in parallel, without any enforced order among them. Moreover, all the threads may access a global memory. Unfortunately, global memory is the slowest of all memory spaces and is not cached.

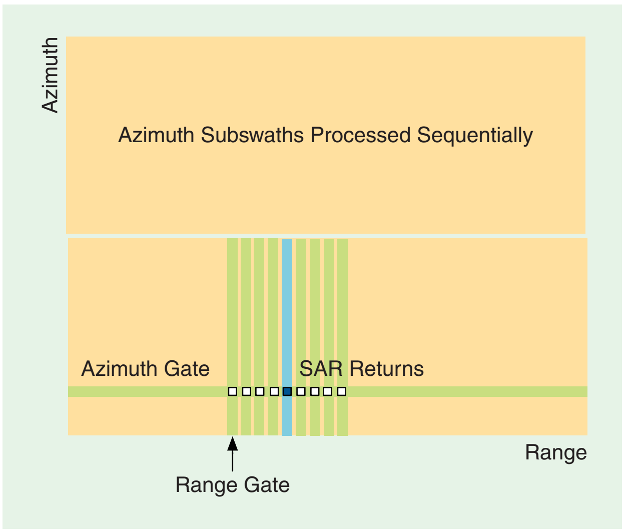
So, accesses to global memory have high latencies that can be partially hidden by defining execution configurations that allow for thousands of blocks and hundreds of threads per block, and by carefully designing kernels so to 1) use the shared memory, both because of the speed of R/W operations and the access scope granted to the entire thread block, 2) transfer data from global memory to shared memory and vice-versa in blocks avoiding a large number of transactions and 3) use the highest number of arithmetic operations on data read by each thread. This allows the GPU to perform arithmetic operations in a better way, while hardly minimizing the cost and the latency of memory operations, and grants an excellent increase in performances.

Having in mind these features, the GPU implementation of this processing subtask is organized by first sectioning the data matrix in subswaths, as in Figure 2 and then by executing FFT computations in batched mode. The batched-mode FFT kernel will be illustrated later in Code Project 3; essentially, this means that a large number of FFTs are computed in parallel, or, more correctly, that a large amount of data belonging to different FFTs are computed in parallel.

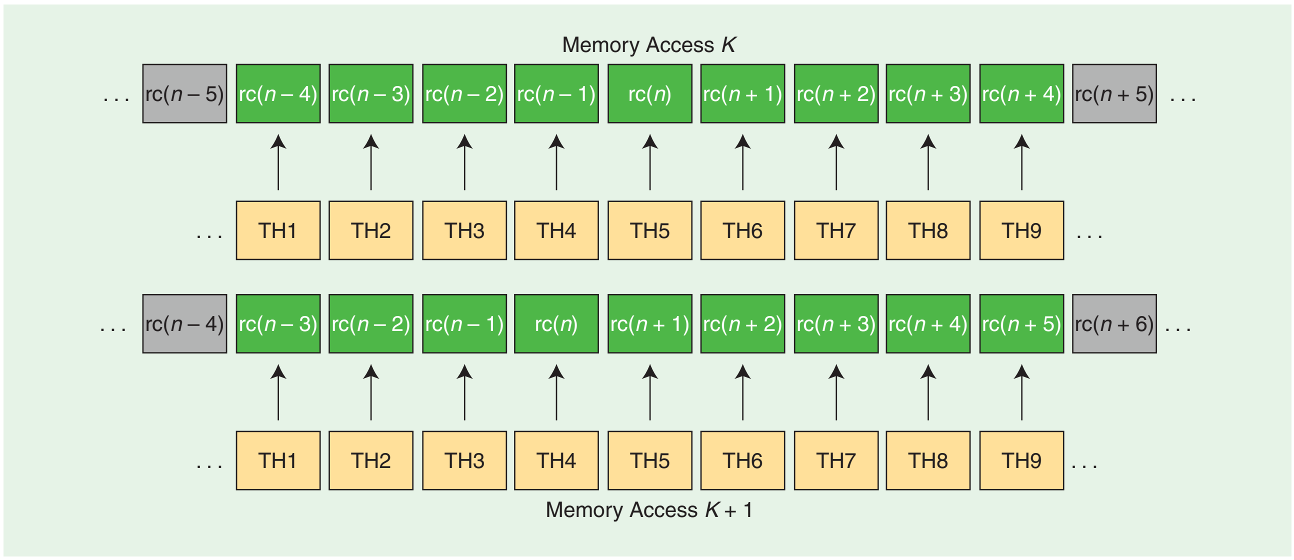
Define a range gate as the set of N_a radar returns that are mapped into a column of the data matrix. Equivalently, an azimuth gate is the set of N_r radar returns that are mapped into a row of the data matrix (see Figure 2). The matched filtering in the frequency domain can be computed very efficiently because all azimuth cells require the same (Fourier transformed) range reference function. This task is executed in parallel by a kernel where each thread reads a range gate, executes the product between range gate values and the corresponding value of the range reference function (it is the same for each range gate!) and write the result in the global memory. To optimize usage of global memory, reading-from and writing-to global memory are coalesced, with a contiguous region of memory accessed by a block of threads. This is obtained by opportunely using thread indices to write memory access instructions so that the kth thread in each half warp accesses the kth element in the block being read. Looking at the excerpt of the range compression kernel, we note

a) the coalesced read of the frequency transformed raw data (code lines 8–9)

b) the coalesced write of the compressed data (code lines 10–11) deriving from the alignment of threads index and memory blocks (code lines 2 and 8–11).



[FIG2] **Matrix deployment of SAR data and scheme of the block processing strategy.**


Memory Access K
... rc(n – 5) rc(n – 4) rc(n – 3) rc(n – 2) rc(n – 1) rc(n) rc(n + 1) rc(n + 2) rc(n + 3) rc(n + 4) rc(n + 5) ...
... TH1 TH2 TH3 TH4 TH5 TH6 TH7 TH8 TH9 ...
... rc(n – 4) rc(n – 3) rc(n – 2) rc(n – 1) rc(n) rc(n + 1) rc(n + 2) rc(n + 3) rc(n + 4) rc(n + 5) rc(n + 6) ...
... TH1 TH2 TH3 TH4 TH5 TH6 TH7 TH8 TH9 ...
Memory Access K + 1

[FIG3] Conflict–free access for eight-samples interpolation required in RCM correction.

RANGE COMPRESSION KERNEL (EXCERPT)

```
__global__ void RangeCompression
   (cufftComplex *rawVecDev, cufftComplex
   *chDev, int nrfft, int nazft) {
    int indexThread = blockIdx.x *
        blockDim.x+threadIdx.x;
    cufftComplex cv;
    __shared__ cufftComplex ch;
    if(indexThread<nazft) {
      /* loop over
      range gates */
     for (int i=0; i <
       nrfft; i++){
       /* read a
       sample of
       matched ilter
       in the shared memory */
       ch = chDev[i];
       cv.x =
         rawVecDev[indexThread +
         i*nrfft].x;
     cv.y =
         rawVecDev[indexThread+i*nrfft].y;
     rawVecDev[indexThread+i*nrfft].x =
                  cv.x*ch.x - cv.y*ch.y;
     rawVecDev[indexThread+i*nrfft].y =
                  cv.x*ch.x + cv.y*ch.y;
  }
  }
}
```

CODE PROJECT 2 (MIGRATION CORRECTION)

The RCM correction in the RD plane is a 1-D interpolation problem. The evaluation of the signal values along the migration path is usually carried out with an interpolation function of a sufficiently high order (an eight-order sinc kernel is accurate even for the most demanding applications). Fortunately, all points lying on the same straight path collinear with the radar flight track exhibit, with a good approximation, the same migration curve; a symmetry that can be usefully exploited in the design of an efficient parallel version of the RCM correction algorithm. Another feature is that there is a repeated use of the data values during the interpolation of adjacent range gates. Therefore, we are quite naturally guided to assign a thread to a range gate and to transfer chunks of data from global memory to shared memory (this method, referred to as tiling, is used to circumvent the tradeoff between the large global memory and the small but distributed shared memory; data are partitioned into smaller subsets so that each tile fits into the shared memory and the kernel computations on that tile can be carried out independently).

[PROGRAMMERS SHOULD TRY TO THINK IN TERMS OF STRUCTURES AND HIERARCHIES, HAVING IN MIND A WORK PLAN AND SOME CAVEATS.]

Our planning requires that the kernel executes computations on a range gate; we define a block as a collection of 64 threads and a grid of as many blocks to include all the range gates. A tile is the subset of range cells spanned by a block. The kernel sequentially moves a tile from the global memory to the shared memory, executes interpolation of the data values using eight surrounding samples (four from each side), and writes the result in the global memory. The read and write operations cannot be coalesced because threads accesses matrix elements column-wise (a range gate is a column of the matrix) but interpolation is carried out using samples that are adjacent in a row-major arrangement. Although the relaxed coalescing restrictions for CUDA compute capability greater than 1.1 allow strided or even random access in a memory segment, this cannot be obtained in our case.

Access to data from different threads is also subject to possible bank conflicts because requests to different banks

IEEE SignalProcessing Previous Page | Contents | Zoom in | Zoom out | Front Cover | Search Issue | Next Page qMags

address cannot be served simultaneously; if this happens, the hardware serializes the accesses and performance decrease. To achieve close to maximum speed, memory-read is organized as in Figure 3 where it is evident that a memory address is always accessed sequentially. During the memory accesses $k, k+1, \ldots, k+9$, thread one reads data values $rc(n-4), rc(n-3), \ldots, rc(n+4)$ that are used sequentially for interpolation on the nth sample. Thus the kernel computes sequentially

$$x_k(n) = x_{k-1}(n) + rc(n-i)\,\mathrm{sinc}(i) \quad i = 1, \ldots, 9\,. \qquad (2)$$

The excerpt of the RCM correction kernel reports the CUDA code

a) Define the number of data samples to be copied in the shared memory. The interpolation order is eight (code-line 1).

b) Samples in the azimuth direction are copied in the shared memory (tiling) (code-line 4).

c) Synchronize all threads for write in the shared memory (code-line 5). All threads should have completed the copy before starting the interpolation.

d) Read from shared memory, bank conflicts are avoided because threads access different memory locations (code–lines 8–9).

e) Interpolation is carried out, I_STEP is the sampling rate of sinc interpolator and `frac` is the fractional part of the range migration (code-lines 7–13).

RCM CORRECTION KERNEL (EXCERPT)

```
1. __shared__
   float2 rcVecShared[blockDim.x+8];
   /* blockDim.x is 64 */
   /*loop on azimuth gates
2. for (j=0; j < nazft; j++){
3.  int ir2 = threadIdx.x+4;
4.  rcVecShared[ir2] =
                 rcVecDev[ir1*nazft+j];
    /* ir1 is the center sample
       for migration */
5.  __syncthreads();
6.  azl = rcVecShared[ir2];
7.  for(k=1; k <= ORDER/2; k++){
     /*  sinc interpolation ORDER is 9 */
8.   rc1=rcVecShared[ir2+k];
9.   rc2=rcVecShared[ir2-k];
10.  indx = nintD(I_STEP*(frac-k));
11.  azl.x +=(rc1.x+rc2.x)*sincDev[k-indx];
12.  azl.y +=(rc1.y+rc2.y)*sincDev[k-indx];
13. }
14.}
```

THE SEQUENTIAL PSEUDOCODE IS HEREAFTER REPORTED FOR COMPARISON

```
for each range gate
    for each azimuth sample
        for i<interpolation order
            use sinc to interpolate the
            azimuth sample
        end
    end
end
```

We are now on the way to complete the SAR processing with the azimuth compression, which requires a minor design effort. After range migration correction, almost all range-azimuth dependencies have been compensated and the matched filtering can be easily implemented with a range-dependent function.

From the computational point of view, we perform a) a set of N-points complex FFTs of the range dependent reference function, with N defined by the subswath length, b) a set of complex multiplications, and c) a set of IFFTs.

[THERE ARE TWO MAIN REASONS WHY IT IS QUITE DIFFICULT TO DETERMINE THE EXECUTION CONFIGURATION ACHIEVING CLOSE TO OPTIMAL PERFORMANCE IN A GPU-BASED APPLICATION.]

CODE PROJECT 3 (FFTCT CODE)

The RD algorithm makes an intensive use of 1-D FFT and IFFT for complex data of radix-2 size. In CUDA-based implementations, a possible approach is the direct use of NVIDIA CUFFT [17], a library that delivers a parallel implementation of complex and real data transforms in one, two, and three dimensions. Based on the FFTW library [18], it executes a plan, i.e., an optimal execution configuration for the specific transform with respect to the data size, the data domain (complex or real), and the hardware. CUFFT computes FFTs in batching mode, an approach where GPU resources are exploited through the concurrent computation of a great amount of (typically independent) data. Best performances are achieved for a dense arithmetic throughput where context switch operations are automatically performed by the GPU controller among, for example, threads of a batch waiting for a costly read or write to global memory or threads of another batch ready to execute an arithmetic operation.

The main limitation of CUFFT is that there is no specific optimization with respect to the GPU resources. The horizon of imaginable improvements possibly achieved by more tailored implementations moved us to investigate a very basic but quite optimized algorithm for FFT computation.

The proposed library (FFTCT) is based upon the work presented in [19] by Volkov and Kazian, which, in turn, relies upon the Cooley and Tukey framework, and currently supports transform of batched 1-D arrays of complex values. The parallel implementation rescales the original problem into a large number of small radix FFTs, where an efficient use of register files and the per-block shared memory is allowed. The main features of the approach can be summarized as follows:

IEEE SignalProcessing Previous Page | Contents | Zoom in | Zoom out | Front Cover | Search Issue | Next Page qMags

[TABLE 2] ARCHITECTURAL CONSTRAINTS IN THE NVIDIA CUDA PLATFORM IMPLEMENTATION.

MAXIMUM NUMBER OF THREADS PER BLOCK	512
MAXIMUM SIZES OF THE X, Y, AND Z DIMENSIONS OF A THREAD BLOCK	512, 512, 64
MAXIMUM SIZE OF EACH DIMENSION OF A GRID OF THREAD BLOCKS	65,535
WARP SIZE	32 THREADS
NUMBER OF REGISTERS PER MULTIPROCESSOR	16,384
SHARED MEMORY AVAILABLE PER MULTIPROCESSOR	16 KB INTO 16 BANKS
CONSTANT MEMORY	64 KB
MAXIMUM NUMBER OF ACTIVE BLOCKS PER MULTIPROCESSOR	8
MAXIMUM NUMBER OF ACTIVE WARPS PER MULTIPROCESSOR	32
MAXIMUM NUMBER OF ACTIVE THREADS PER MULTIPROCESSOR	1,024

- use different kernels for managing different vector sizes
- need of a final per-batch transposition
- code optimizations, such as shifts for power-of-two divisions or modulus operations, loop unrollings, and so on.

The model assigns each thread a little subset of values from the input arrays, to compute the base-case transform.

An example is quite clarifying: for a 4,096-points, radix-8 FFTs are required, thus a single thread is charged for the following operations on eight data samples:

a) read data from global memory and save them in the fast register file (see code lines 4–7)

b) calculate their FFT using a hard-coded, optimized routine (called at code line 8)

c) multiply by twiddle factor (see code lines 9–12).

[TABLE 3] SAR PROCESSOR PARAMETERS USED FOR SIMULATION SETUP.

SATELLITE NAME	ERS-2
PULSE REPETITION FREQUENCY (PRF)	1,679 [HZ]
RANGE COMPRESSION FFT LENGTH N_r	4,096 [SAMPLES]
RANGE MATCHED FILTER LENGTH	704 [SAMPLES]
AZIMUTH COMPRESSION FFT LENGTH N_a	4,096 [SAMPLES]
AZIMUTH MATCHED FILTER LENGTH	1,103 [SAMPLES]
NUMBER OF AZIMUTH PATCHES	10
COMPUTATION WORD LENGTH	COMPLEX 32 [BIT]

Finally, values are reordered and transferred back to global memory (see code lines 13–17) for a global per-batch transposition. This computation is performed by a separate parallel kernel.

FFT KERNEL (EXCERPT)

```
1.  __global__
    void fftff_8_points(cmplxValue *v ) {
2.  cmplxValue temp[8];
3.  /* offset computation in the array v
       of the 8 values to store in the
           temp array /* code omitted */
4.  #pragma unroll 8
5.  for( int i = 0; i < 8; i++ )
6.   temp[i] = v[i<<9];
7.  }
8.  radix8_fft( temp );
9.  #pragma unroll 8
10. for( int j = 1; j < 8; j++ )
11.  temp[j] = temp[j] * twiddleFactor_8(
         offset, j, 4096,FFTFF_FORWARD);
12. }
13. #pragma unroll 8
14. for( int i = 0; i < 8; i++ ){
15.  int ind = indexReverse_8(i);
16.  v[i<<9] = temp[ind];
17. } ...
```

Experimental results of FFTCT computations delivered by this implementation (without the final transposition) show a typical speedup of 3.6× compared to CUFFT library in CUDA version 1.1, for the same testbed adopted in the experimental results section.

EXTENSIONS RELATED TO 2-D SAR PROCESSING

In this class of algorithms, all phases of data compensation and interpolation are moved to the 2-D Fourier domain. The perspective is quite different, but we do not expect a necessarily more complex computational scenario.

The technique for evaluating 2-D FFTs (direct and inverse) on a GPU is straightforward: simply, 1-D FFTs are executed in parallel along the azimuth gates, then 1-D FFTs are executed in parallel along the range gates.

The Stolt mapping remaps the uniformly spaced samples, in the (ω, k_x) domain, to nonuniformly spaced samples in the (k_r, k_x) domain that are not suitable for the inverse transformation stage. An interpolation, similar to that introduced in the RD algorithm may solve the problem, but in general, it should be noted that the Stolt mapping may produce quite unevenly mapped samples and a particular attention should be payed in the application of such interpolator.

[TABLE 4] EXECUTION TIME FOR THE MAIN SECTIONS OF THE PARALLEL SAR PROCESSOR. THE CONFIGURATION ON NVIDIA TESLA C1060 IS TUNED FOR BEST PERFORMANCE WITH ERS-2 DATA. REFERENCE WORKSTATION FOR SAR PROCESSOR (NONPARALLELIZED, NONOPTIMIZED) IS 3.00 GHz, INTEL CORE 2 DUO E6850, 4 GB RAM.

CUDA KERNEL	# THREADS PER BLOCK	GPU μS	CPU μS	SPEEDUP
RANGE COMPRESSION	64	0.9×10^6	6.6×10^6	7.3×
PARAMETER ESTIMATION	32	7.1×10^2	3.1×10^4	44.6×
AZIMUTH MATCHED FILTER	64	5.9×10^5	5.0×10^6	8.5×
RCMC AND AZIMUTH COMPRESSION	64	1.7×10^6	31.3×10^6	18.4×
FFTCT		4.4×10^5	21×10^6	47.8×
TOTAL FOCUSING		4.4×10^6	70.1×10^6	15.9×

RESULTS AND OPTIMIZATION KEYNOTES

There are two main reasons why it is quite difficult to determine the execution configuration achieving close to optimal performance in a GPU-based application. The first is that there is not a linear dependence between performance and block size [20], the second is that generally, there is a domain of influence among the different parameters of a GPU, so changing a design parameter may affect the position of the optimum point of work along other coordinates of the parameter space. For example, a loop unrolling removes dynamic instructions but affects internal memory usage and, thus, threads execution.

For these reasons, the optimization has been carried out using an empirical (constrained) search, with a reduced code running many times with different parametric templates. Given the relative simplicity of the optimization task, we may consider a two-levels strategy [21], based on

a) optimization of the execution configuration of the single kernels, that is the number of thread blocks and the number of threads per block

b) optimization of loops.

All the optimizations have been checked automatically, using a simple branch-pruning strategy to constrain the optimization space and improve search efficiency.

We selected the Nvidia Tesla C1060 as device board; it is a high-performance GPGPU offering 30 multiprocessors, each one with eight core processors. Other device parameters as well as CUDA version capabilities are shown in Table 2. Performance of the SAR processor have been evaluated with reference to a standard ERS-2 satellite pass with parameters as in Table 3.

Now is the time to make some final considerations on how the effort is rewarded by good performances. It is necessary to explain the terms of comparison, especially regarding what is available in terms of standard or multithreaded SAR processing software. Unfortunately, SAR processors are precious works of art, often covered by patents with undisclosed source code. This makes the comparison issue quite unfavorable. Our reference SAR processor is a commercial open-source product widely used by research institutions and data processing agencies, so that we are quite confident on the reference algorithm and on the code section we are comparing. It is a nonparallel, RD algorithm performing the same steps as our GPU code. Results in Table 4 show, for the main steps of processing, the performance of the reference and GPU processors and the speedup obtained through the GPU implementation. The reference system is a single workstation, equipped with Intel Core 2 Duo E6850 at 3.00 GHz (FSB at 1,333 MHz and 4 MB of L2 cache), 4 GB RAM with Linux Ubuntu 9.04 OS. The image is an ERS-2, 14 July 1995 pass, centered in the Campania region (Italy) and the size of the raw data matrix is 26,880 × 4,912 azimuth-range gates.

[OUR HOPE IS THAT THE READER HAS GAINED A CLEAR IDEA ON HOW TO DESIGN FINE-GRAINED PARALLEL PROCESSORS FOR NEAR REAL-TIME SAR FOCUSING.]

A few comments are in order, concerning results in Table 4. First, the moderate speedup in the range compression kernel is not surprising: most of the time is used to move data from CPU to the GPU device. This is also what happens in the azimuth matched filter stage. Second, the parameter estimation kernel is very quick; this is also due to extensive use of intrinsic math functions implemented in the special function unit of the GPU. Third, the FFT is no longer the key aspect of parallel SAR processing. Attention should be moved toward less canonical aspects like memory transactions and computational intensity.

CONCLUSIONS

Our hope is that the reader has gained a clear idea on how to design fine-grained parallel processors for near real-time SAR focusing. Without being comprehensive, we have tried to focus on methodologies behind the technicalities, with the aim of unveiling the core features of parallel SAR processing. We believe that the reader that is confident with this basic approach may now proceed safely to develop one of the different focusing algorithms or to switch to applications related to SAR data where we find a great number of techniques, ranging from ocean studies to forest monitoring and high-accuracy, high-resolution surface deformation analysis.

If we try to see further away, including small satellites and airborne remote sensing systems, we may look at an increasing number of applications where the enabling technology is high-performance computing in a small box. The SAR images will often be required in less than one second for real-time operations, and we realize that there is a formidable chance for this, offered by the GPU technology but, as we mentioned earlier in the introduction, there are lot of things to still be done.

ACKNOWLEDGMENTS

We gratefully acknowledge Carmine Clemente for support in the code design and for many valuable discussions, Marcello Spinelli for support in the design of the FFT algorithm, and the Italian Ministry of Environment for financing part of this research.

AUTHORS

Maurizio di Bisceglie (dibisceg@unisannio.it) received the Ph.D. degree in electronics and telecommunications from Università degli Studi di Napoli “Federico II,” Napoli, Italy. Since 1998, he has been with Università degli Studi del Sannio, Benevento, Italy, as a professor of telecommunications. He was a visiting scientist at the University College of London, U.K. and at the Defense Evaluation and Research Agency, Malvern, U.K. His research interests are in the field of statistical signal processing with applications to radar and remote sensing. He was on the organizing committee of the Italian phase of the European AQUA Thermodynamic Experiment (EAQUATE) mission in 2004, cochair of the 2005 NASA Direct Readout Conference, and part of the organizing committee of the 2008 IEEE Radar Conference. He is a Member of the IEEE.

Michele Di Santo (disanto@unisannio.it) graduated cum laude in electronics engineering at the University of Napoli "Federico II," Italy, in 1971. After teaching at the Universities of Napoli "Federico II," Calabria, and Salerno, he is now a professor of computer engineering at the University of Sannio in Benevento, Italy, where he serves as coordinator of the computer engineering program. He also served as dean of the faculty of engineering of the University of Sannio from November 2000 to October 2006 and as coordinator of the program in doctoral studies in information engineering from May 2000 to February 2008. He teaches courses on digital design, computer architecture, programming, programming languages, and history of computing. His primary research interests are in the areas of programming languages and parallel and distributed systems. He is a Member of the IEEE Computer Society and ACM.

Carmela Galdi (galdi@unisannio.it) received the Dr. Eng. and Ph.D. degrees in electronic engineering from the Università degli Studi di Napoli, "Federico II," Italy. In 1993, she worked as a software engineer at Alcatel Italia, Salerno, Italy. She was with the Università di Napoli, "Federico II" until 2000, when she joined the Università del Sannio, Benevento, Italy, where she is currently a professor of telecommunications. She was a visiting scientist at the University College of London and at the Defense Evaluation and Research Agency, Malvern, United Kingdom. Her research interests are in the field of statistical signal processing, non-Gaussian models of radar backscattering, and remote sensing applications. She was on the organizing committee of the 2008 IEEE Radar Conference.

Riccardo Lanari (lanari.r@irea.cnr.it) received the degree in electronic engineering (cum laude) from the University of Napoli, "Federico II," Naples, Italy, in 1989. After graduation, he joined the Istituto di Ricerca per l'Elettromagnetismo e i Componenti Elletronici (IRECE) [now called the Istituto per il Rilevamento Elettromagnetico dell'Ambiente (IREA)], where he is currently a senior researcher. He has been a visiting scientist at different foreign research institutes and a research fellow at the Institute of Space and Astronautical Science, Japan, in 1993. He holds two patents on SAR data-processing techniques. His main research interests are in the field of SAR data-processing and SAR interferometry. He received NASA recognition for the development of the ScanSAR processor used for the SRTM mission. He is chair and cochair at several international conferences and a member of the Technical Program Committee of numerous symposia and conferences. He is a Senior Member of the IEEE.

Nadia Ranaldo (ranaldo@unisannio.it) received the Ph.D. degree in computer science from University of Sannio, Benevento, Italy, in 2005. She is a research assistant with the Department of Engineering, University of Sannio. Her main research interests include resource management systems, frameworks for distributed systems, parallel computing, wireless and sensor networks, and grid and service computing.

REFERENCES

[1] GESS Study Team, "Global earthquake satellite system (GESS), a 20-year plan to enable earthquake prediction," NASA-JPL Tech. Rep., 2003.

[2] M. Puschel, J. M. F. Moura, J. R. Johnson, D. Padua, M. M. Veloso, B. W. Singer, J. Xiong, F. Franchetti, A. Gacic, Y. Voronenko, K. Chen, R. W. Johnson, and N. Rizzolo, "Spiral: Code generation for DSP transforms," *Proc. IEEE*, vol. 93, no. 2, pp. 232–275, Feb. 2005.

[3] D. S. McFarlin, F. Franchetti, M. Püschel, and J. M. F. Moura, "High-performance synthetic aperture radar image formation on commodity multicore architectures," in *Proc. Society of Photo-Optical Instrumentation Engineers (SPIE) Conf. Series*, May 2009, vol. 7337.

[4] J. Rudin, "Implementation of polar format SAR image formation on the IBM cell broadband engine," in *Proc. High Performance Embedded Computing (HPEC)*, 2007.

[5] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips, "GPU computing," *Proc. IEEE*, vol. 96, no. 5, pp. 879–899, May 2008.

[6] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. E. Lefohn, and T. Purcell, "A survey of general-purpose computation on graphics hardware," *Comput. Graph. Forum*, vol. 26, no. 1, pp. 80–113, Mar. 2007.

[7] G. Franceschetti and R. Lanari, *Synthetic Aperture Radar Processing*. Boca Raton, FL: CRC, 1999.

[8] G. Fornaro, E. Sansosti, R. Lanari, and M. Tesauro, "Role of processing geometry in SAR raw data focusing," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 38, no. 2, pp. 441–454, Apr. 2002.

[9] I. G. Cumming and F. H. Wong, *Digital Processing of Synthetic Aperture Radar Data: Algorithms and Implementation*. Norwood, MA: Artech House, 2005.

[10] C. Cafforio, C. Prati, and F. Rocca, "SAR data focusing using seismic migration techniques," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 27, no. 2, pp. 194–207, Mar. 1991.

[11] G. Franceschetti and G. Schirinzi, "A SAR processor based on two-dimensional FFT codes," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 26, no. 2, pp. 356–366, Mar. 1990.

[12] R. K. Raney, H. Runge, R. Bamler, I. G. Cumming, and F. H. Wong, "Precision SAR processing using chirp scaling," *IEEE Trans. Geosci. Remote Sensing*, vol. 32, no. 4, pp. 786–799, July 1994.

[13] R. Lanari, "A new method for the compensation of the SAR range cell migration based on the chirp z-transform," *IEEE Trans. Geosci. Remote Sensing*, vol. 33, no. 5, pp. 1296–1299, Sept. 1995.

[14] G. C. Pryde, K. D. Beckett, L. M. Delves, C. J. Oliver, and R. G. White, "Design of a real-time high-quality SAR processor," in *Proc. Soc. Photo-Optical Instrumentation Engineers (SPIE) Conf. Series*, D. A. Giglio, Ed., June 1994, vol. 2230, pp. 148–159.

[15] NVIDIA Tesla GPU computing solutions for HPC [Online]. Available: http://www.nvidia.com/object/tesla_computing_solutions.html

[16] Khronos OpenCL Working Group, *The OpenCL Specification Version: 1.0*, Aaftab Munshi, 2009.

[17] *CUDA CUFFT Library 1.1* [Online]. Available: http://developer.download.nvidia.com/compute/cuda/1_1/CUFFT_Library_1.1.pdf

[18] M. Frigo and S. G. Johnson, "FFTW: An adaptive software architecture for the FFT," in *Proc. 1998 IEEE Int. Conf. Acoustics, Speech and Signal Processing*, 1998, vol. 3, pp. 1381–1384.

[19] V. Volkov and B. Kazian, "Fitting FFT onto G80 architecture," UC Berkeley CS258 project report, May 2008.

[20] S. Ryoo, C. I. Rodrigues, S. S. Stone, S. S. Baghsorkhi, S. Ueng, J. A. Stratton, and W. W. Hwu, "Program optimization space pruning for a multithreaded GPU," in *CGO 08: Proc. 6th Annu. IEEE/ACM Int. Symp. Code Generation and Optimization*, 2008, pp. 195–204.

[21] Y. Liu, E. Z. Zhang, and X. Shen, "A cross-input adaptive framework for GPU programs optimization," Dept. Comput. Sci., College of William & Mary, Tech. Rep. N. WM-CS-2008-09, 2008.

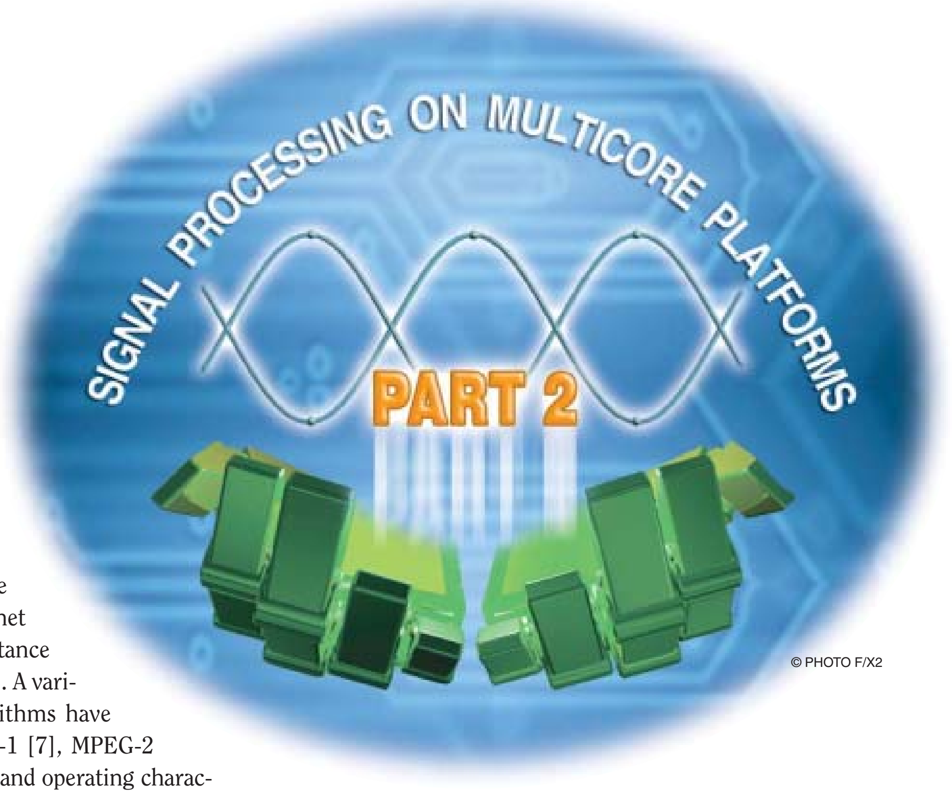
[22] A. M. Turing, "Computing machinery and intelligence," *Mind*, vol. 59, no. 236, pp. 433–460, Oct. 1950.

[SP]

[Ngai-Man Cheung, Xiaopeng Fan, Oscar C. Au, and Man-Cheung Kung]

Video Coding on Multicore Graphics Processors

[The challenges and advantages of the GPU implementation]



© PHOTO F/X2

Today, video coding [1]–[5] has become the central technology in a wide range of applications. Some of these include digital TV, DVD, Internet streaming video, video conferencing, distance learning, and surveillance and security [6]. A variety of video coding standards and algorithms have been developed (e.g., H.264/AVC [5], VC-1 [7], MPEG-2 [8], AVS [9]) to address the requirements and operating characteristics of different applications. With the prevalent applications of video coding technologies, it is important to investigate efficient implementation of video coding systems on different computing platforms and processors [10], [11].

Recently, graphics processing units (GPUs) have emerged as coprocessing units for central processing units (CPUs) to accelerate various numerical and signal processing applications [10], [12]–[14]. Modern GPUs may consist of hundreds of highly decoupled processing cores capable of achieving immense parallel computing performance. For example, the NVIDIA GeForce 8800 GTS processor has 96 individual stream processors each running at 1.2 GHz [15]. The stream processors can be grouped together to perform single instruction multiple data (SIMD) operations suitable for arithmetic intensive applications. With the advances in the GPU programing tools such as thread computing and C programming interface [16], [17], GPUs can be efficiently utilized to perform a variety of processing tasks in addition to conventional vertex and pixel operations.

With many personal computers (PCs) or game consoles equipped with multicore GPUs capable of performing general purpose computing, it is important to study how the GPU can be utilized to assist the main CPU in computation-intensive tasks such as video compression/decompression [18]. In fact, as high-definition (HD) contents are getting more popular, video coding

Digital Object Identifier 10.1109/MSP.2009.935416

would require more and more computing power. Therefore, leveraging the computing power of GPUs could be a cost-effective approach to meet the requirements of these applications. Note that with dozens of available video coding standards (H.264, MPEG-2, AVS, VC-1, WMV, DivX) it is advantageous to pursue a flexible solution based on software.

LEVERAGING THE COMPUTING POWER OF GPUs COULD BE A COST-EFFECTIVE APPROACH TO MEET THE REQUIREMENTS OF VIDEO CODING.

INTRODUCTION

Focusing on software-based video coding applications running on PCs or game consoles equipped with both CPUs and GPUs, this article investigates how GPUs can be utilized to accelerate video encoding/decoding. Recent work has proposed applying multicore GPUs/CPUs for various video/image processing applications. Table 1 summarizes some of them. In this article, we survey prior work on video encoding and decoding to illustrate the challenges and advantages of the GPU implementation. Specifically, we discuss previous work on GPU-based motion estimation (ME), motion compensation (MC), and intraprediction. Our focus is on how the algorithms can be designed to harness the massive parallel processing capability of the GPU. In addition, we discuss previous work on partitioning the decoding flow between CPUs and GPUs (for completeness, we also report the speedup results in previous work. However, since the GPU/multicore software/hardware technologies have evolved dramatically over the last few years, some of the results could be outdated). After that, we investigate a GPU-based fast ME. We discuss some strategies to break dependency between different data units and examine the tradeoff between speedup and coding efficiency.

BACKGROUND

VIDEO CODING

The latest video coding standards have achieved state-of-the-art coding performance. For example, H.264/AVC, which is the latest international video coding standard approved by the International Telecommunications Union (ITU-T) and the International Organization for Standardization/International Electrotechnical Commission (ISO/IEC), typically requires 60% or less of the bit rate compared to previous standards to achieve the same reconstruction quality [5]. Other advanced video coding algorithms, such as AVS-Video developed by the Audio and Video Coding Standard Working Group of China [9], or VC-1 initially developed by Microsoft [7], have also achieved competitive compression performance. Next, we provide an overview of the H.264 video coding standard.

The H.264 video coding standard is designed based on the block-based hybrid video coding approach [2], [5], which has been used since earlier video coding standards. The coding algorithm exploits spatial correlation between neighboring pixels of the same picture. In addition, it also exploits temporal correlation between neighboring pictures in the input video sequence to achieve compression. Figure 1 depicts the encoder block diagram. The input picture is partitioned into different blocks, and each block may undergo intraprediction using neighboring reconstructed pixels in the same frame as predictor. H.264 supports intraprediction block sizes of 16×16, 8×8, and 4×4, and it allows different ways to construct the prediction samples from the adjacent reconstructed pixels. Alternatively, the input block may undergo interprediction using the reconstructed blocks in the reference frames as predictor. Interprediction can be based on partition size of 16×16, 16×8, 8×16, 8×8, 8×4, 4×8, or 4×4. Displacement between the current block and the reference block can be up to quarter-pel accuracy and is signaled by the motion vector and the reference picture index [2].

The prediction residue signal from intraprediction or interprediction would then undergo transformation to decorrelate the data. In H.264, a 4×4 separable integer transform is used, which is similar to 4×4 DCT but avoids the mismatch between forward and inverse transform. Then, the transform coefficients would be scalar quantized and zig-zag scanned. The context-adaptive variable length coding (CAVLC) may then be employed to entropy code the scanned transform coefficients. CAVLC is an adaptive coding scheme, and it may switch between different codeword tables during encoding depending on the values of the already-coded elements. Alternatively, the transform coefficients may be coded by context-adaptive binary arithmetic coding (CABAC). To mitigate blocking artifacts, an adaptive in-loop deblocking filter would be applied to the reconstruction from the feedback loop.

GPUs

Originally designed as specialized hardware for three-dimensional (3-D) graphics, GPUs have recently emerged as coprocessing units to accelerate arithmetic intensive applications in

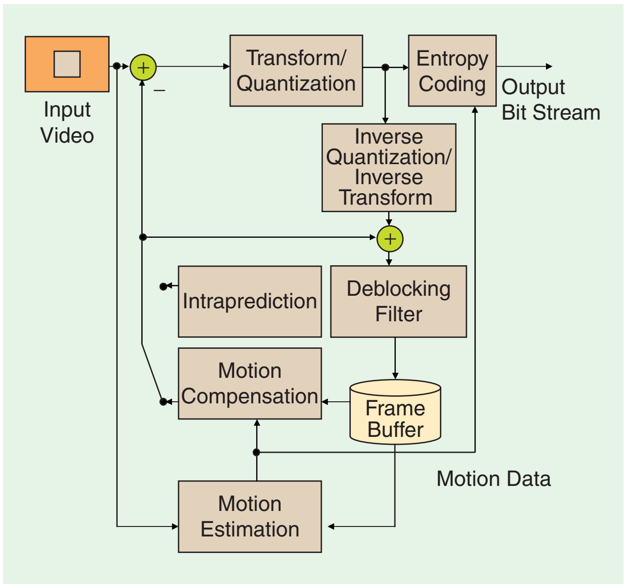
[TABLE 1] VIDEO AND IMAGE PROCESSING APPLICATIONS ON MULTICORE PROCESSORS.

APPLICATIONS	EXAMPLES
VIDEO ENCODING	MOTION ESTIMATION [19]–[23], INTRAPREDICTION [24]–[27], TRANSFORM [28]
VIDEO DECODING	MOTION COMPENSATION [10], [29], DECODER DESIGN [10], [30]–[32]
HIGH DYNAMIC RANGE IMAGES	TEXTURE COMPRESSION [33]
VIDEO WATERMARKING	REAL-TIME VIDEO WATERMARKING SYSTEM [14]
SIGNAL PROCESSING KERNELS	MATRIX AND VECTOR COMPUTATIONS [12], FFT, AND CONVOLUTION [13]
IMAGE ANALYSIS	HOUGH TRANSFORM [34], RADON TRANSFORM [35], [36], CHIRPLET TRANSFORM [35], FEATURE EXTRACTION [37]

PCs or game consoles. A key feature of modern GPUs is that they offer massive parallel computation capability through hundreds of highly decoupled processing cores [38]. For example, NVIDIA GeForce 8800 GTS processor consists of 96 stream processors each running at 1.2 GHz [15].

The design philosophy of GPUs is quite different from that of general-purpose CPUs. Throughout the years, GPUs have been designed with an objective to support the massive number of calculations and huge amount of data transfer required in advanced video games [38], [39]. In addition, they need to meet the stringent cost requirement of consumer applications. Therefore, GPUs have become very cost effective for arithmetic computation. Furthermore, the peak computation capability of GPUs is increasing at a faster pace than general-purpose CPUs (Figure 2).

Besides arithmetic computation capability, there are other fundamental differences between CPUs and GPUs. First, to address a wide range of applications, general-purpose CPUs would use many transistors to implement sophisticated control hardware that can support some advanced control functions such as branch prediction [40]. On the contrary, GPUs would instead devote chip area to arithmetic computation. As a consequence, GPUs may not perform well for programs with many conditional statements. Second, CPUs use a lot of chip area to implement cache memory to reduce instruction and data access latencies. GPUs, on the other hand, use much simpler memory models but rely on the high degree of parallelism in an application to hide the memory access latency. Thus, it is central to expose a large amount of data parallelism in the GPU programs.



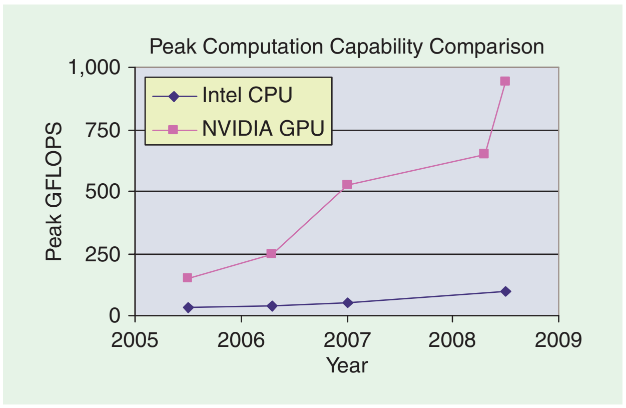
[FIG1] H.264/AVC encoding algorithm [5].

GPU-ASSISTED VIDEO CODING: CHALLENGES

Following from the previous discussion, it is clear that only certain types of computation are suitable for the GPU execution. In particular, to fully harness the computational power in the GPU, one would need to design the algorithm to utilize the massive number of processing cores in parallel. As an example, a good application may run up to thousands of threads simultaneously on a high-end GPU so as to keep all the processing cores working continuously [38]. Therefore, one of the main challenges to utilize the GPU for video coding is how to structure a certain module to expose as much data parallelism as possible. Note that this may not be trivial for some video coding modules since dependency may exist between different data units in the computation, as pointed out by previous work [22], [24], [25], [27]. Moreover, flow control instructions (`if`, `switch`, `do`, `for`, `while`) can significantly degrade the performance of the GPU execution, since such instructions may cause different threads to follow different execution paths and the execution would need to be serialized [39]. Therefore, using GPUs for entropy coding such as CAVLC could be challenging. Furthermore, an implementation should try to avoid as much as possible off-chip data access, which may incur considerable latency (recall that the GPU is not optimized for memory access latency). For example, some GPUs may require from 400 to 600 cycles latency for off-chip memory access (while they can perform single-precision floating-point multiply-add in a single cycle in each core) [39]. Note that it is possible to hide such memory access latency if there are enough independent arithmetic computations. Therefore, if possible, a video coding module should be implemented with high *arithmetic intensity* (which is defined as the number of mathematical operations per memory access operation). In some situations, it could be more efficient to recalculate some variables rather than loading them from the off-chip memory.

PREVIOUS WORK

In this section, we review previous work on applying GPUs for video coding. Previous work has proposed to utilize GPUs to undertake ME [19]–[22], intraprediction [24]–[27], and MC [10], [29]. Note that ME, intraprediction, and MC are some of the most computation-intensive modules in interframe encoding, intraframe encoding, and decoding, respectively. Therefore, it is important to understand how these modules can be efficiently implemented on GPUs. In addition to these



[FIG2] Peak computation capability of GPUs and CPUs [39], [41].

modules, GPU-based discrete cosine transform (DCT) has been discussed in [28]. There seems to be no previous work on the GPU-based deblocking filter. Since the deblocking filter involves some conditional statements to determine the strength of the filter at each block boundary, some study may be necessary to determine its performance on GPUs.

WE DISCUSS SOME STRATEGIES TO BREAK DEPENDENCY BETWEEN DIFFERENT DATA UNITS AND EXAMINE THE TRADEOFF BETWEEN SPEEDUP AND CODING EFFICIENCY.

MOTION ESTIMATION ON GPUs

ME is one of the most computation-intensive modules in video encoding and there has been a lot of interest to offload it to GPUs to improve the overall encoding performance. Earlier work in this area focuses on ME algorithms where sum of absolute differences (SAD) is used in block matching to determine the best candidate. SAD computation can be easily parallelized as each individual pixel in the current block is compared independently with the corresponding pixel in the candidate reference block. Note that SAD-based ME is commonly used in MPEG-1/2 and H.263.

More recent video encoding algorithms, on the other hand, may employ rate-distortion (RD)-optimized ME that considers both the rate and distortion in selecting the best candidate. For example, one common metric is the weighted sum of the SAD (between the current block and the candidate block) and the encoding rate of the motion vectors (MVs). In the H.264 standard, predictive coding is used to encode the MV of the current block, and the predictor is the median of the MVs in the adjacent left, top and top-right blocks. Therefore, in RD-optimized ME, the MVs of the neighboring blocks would need to be first determined. Then, based on the median of the neighboring MVs, the encoding rate of the current MV can be determined and the cost of the current block can be computed in the block matching. Such dependency makes it difficult to utilize GPUs for RD-optimized ME. We will discuss example designs to address this issue.

GPU-BASED ME BASED ON LOOP UNROLLING

To increase the degree of parallelism, [20] proposed to unroll the computation loop in SAD-based full search ME. The ME computation loop is shown in Figure 3, and loop unrolling is possible since there is no dependency between individual macroblocks (MBs) when SAD is used as metric for matching. Due to resource constraint in earlier GPUs, the algorithm in [20] needs to be partitioned into two separate passes so that the GPU memory can accommodate the instructions. The experiments in [20] compared full search ME on an INTEL Pentium 4 3.0 GHz CPU and on a NVIDIA GeForce 6800 GT GPU, and the results suggest the GPU-based ME can achieve up to two times and 14 times of speedup for integer-pel and half-pel ME, respectively. The considerable improvement in the half-pel ME is due to the fact that [20] utilizes the built-in hardware support in the GPU for interpolation.

```
Loop (rows of macroblocks) {
  Loop (columns of macroblocks) {
    Loop (rows of search range) {
      Loop (columns of search range) {
        SAD computation;
        SAD comparison;
      }
    }
  }
}
```

[FIG3] Pseudocode of conventional integer-pel ME based on SAD.

Note that with loop unrolling it is possible to schedule a massive number of parallel threads (subject to device's constraint). Consider an example to assign one thread to compute one SAD between an MB and a candidate block in the search window. Then, in the case of full search, the number of independent threads could be as large as the number of MBs times the number of candidate blocks per MB (search window size). For HD 720p videos ($1{,}280 \times 720$, 3,600 MBs per frame), and a search range of 64 (129×129 search window size), the number of threads could be as many as $3{,}600 \times 129 \times 129 = 59{,}907{,}600$.

Although full search is highly parallel, it may have only little practical interest because of the prohibitive computational requirement, especially for HD video contents. Moreover, when MBs are processed independently and MVs are computed concurrently in different threads, it becomes difficult to use motion vector prediction, where MVs of neighboring blocks are used to initialize the search of current MB, and this may affect ME performance when the search window is small. In the next section, we will discuss the GPU implementation of fast ME, which can (in general) achieve comparable coding performance as full search with a much smaller number of computations [42].

GPU-BASED ME BASED ON REARRANGING THE ENCODING ORDER

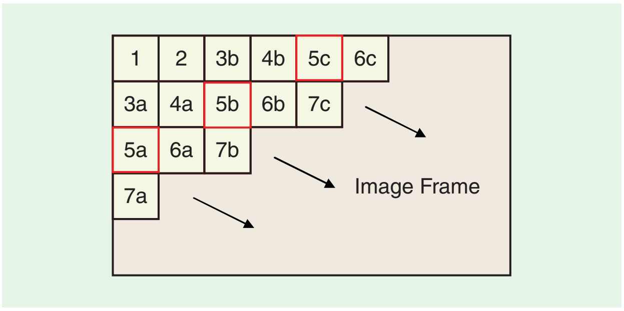
Due to the dependency between adjacent blocks as discussed, RD-optimized ME commonly employed in recent video coding standards cannot be parallelized simply by loop unrolling. In [21] and [22], rearrangement of the encoding order is proposed to increase the degree of parallelism. In these algorithms, instead of processing the blocks in the conventional raster-scan order, the blocks are processed along the diagonal direction to address the dependency issue. This is shown in Figure 4 for the case of 4×4 ME. In their proposed encoding order, at each iteration, the ME will process all the blocks of which the neighboring blocks (left, top, and top-right) have been processed. That is, the ME processes at each iteration all the blocks of which neighboring MVs have been computed and median predictors are available. By processing blocks along the diagonal direction the proposed rearrangement can substantially increase the degree of parallelism. For example, [22] reported that the maximum degree of parallelism can be up to 44, 160, and 240 for CIF (352×288), 720p, and 1080p video, respectively. Note that for each 4×4 block, individual search points in the search window can be examined in parallel (in the cases of full search or some fast search with regular sampling of search window). Therefore, with block-level parallelism of 240 (i.e., 240 4×4 blocks in the current frame can be processed in parallel) and a search range of 64 (129×129 search window size), $240 \times 129 \times 129 = 3{,}993{,}840$ independent threads can be launched simultaneously in principle. Pixel level parallelism can also be implemented, e.g., by decomposing the SAD calculation into several threads. The results in [22] suggest that over 40 times of speedup can be achieved in a system with an INTEL Pentium 4 3.2 GHz CPU and a NVIDIA GeForce 8800 GTS graphics processor. Note that Pentium 4 CPUs are relatively slow compared with more recent CPUs. Also, the program code on Pentium might not have been well optimized. Thus the reported speedups in [22] could be higher than those w.r.t. more efficient CPU implementation. Nonetheless, the results still suggest RD-optimized ME can be implemented efficiently on GPUs.

RD-OPTIMIZED INTRAMODE DECISION ON GPUs

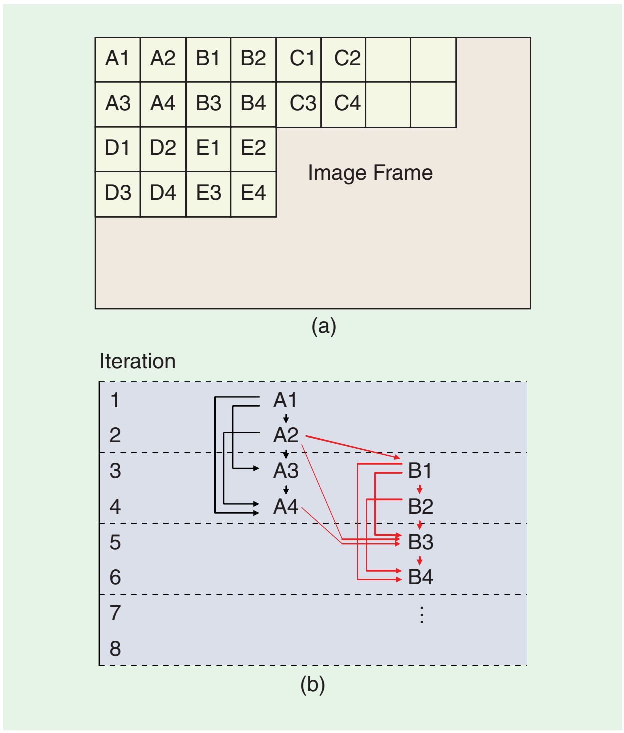
Recent video encoding algorithms use RD-optimized intramode selections to determine the optimal intraprediction direction. In these methods, the encoder would compute the Lagrangian costs of all the candidate prediction modes and select the prediction mode that minimizes the cost. The Lagrangian cost can be the weighted sum of the sum of square differences (SSD) between the original and reconstructed block and the encoding rate for header and quantized residue block. To calculate the cost for a candidate mode, it may involve computing the intraprediction residue, transformation, and quantization on the prediction residue, inverse quantization and inverse transformation, and entropy coding of the quantized transform coefficients. Therefore, the computational complexity of RD-optimized intramode selections could be very significant [43]–[45].

Achieving massive parallelization of RD-optimized intradecision can be challenging. It is because, in intraprediction, the reconstructed pixels of the neighboring blocks are used to compute the reference samples. Therefore, the intraprediction modes of the neighboring blocks would need to be first determined, and these blocks would be encoded and reconstructed accordingly. Then, different candidate modes of the current block can be evaluated based on the reconstructed pixels in the neighboring blocks. Such a dependency hinders the parallelization of the RD-optimized intradecision for GPU implementation.

To address the dependency issue, previous work has proposed different strategies to modify the block processing order [26], [27], [46]. In particular, [27] analyzes the dependency constraint and proposes to process the blocks following a greedy strategy: in each iteration, the encoder would process all the blocks of which parent blocks have been encoded (in the dependency graph, Block A is the parent block of block B if block B requires the reconstructed pixels from block A under various candidate prediction modes). Also, in the greedy strategy, a video block will be scheduled for processing immediately after all its parent blocks have been processed. Figure 5 depicts the dependency constraint in H.264 4×4 intraprediction and the scheduling under the greedy strategy. [27] argues that the greedy strategy is optimal for H.264 and AVS encoding: under the specific constraints imposed by H.264/AVS, and among all encoding orders obeying the constraints, the greedy-based encoding order



[FIG4] Block encoding order proposed in [22] for H.264 4 × 4 RD-optimized ME. Each square represents a 4 × 4 block. Blocks with the same number (e.g., 5a, 5b, 5c) are to be processed in parallel.



[FIG5] (a) Notations for dependency graph: each block corresponds to a 4 × 4 block. (b) Dependency graph when processing an image frame in H.264 RD-optimized intramode selection. Each node represents a 4 × 4 block (see (a) for notations). A directed edge going from block A (parent node) to block B (child node) indicates that block B requires the reconstructed pixels from block A to determine the RD costs of various candidate prediction modes. The graph is processed following the greedy strategy proposed in [27], and the figure shows the iteration at which each block is processed.

requires the minimum number of iterations to process all the blocks. Simulation results suggest that, using the greedy strategy, GPU-based intramode decision can achieve about up to two times speedup in a system with an INTEL Pentium 4 3.2 GHz CPU and a NVIDIA GeForce 8800 GTS graphics processor (Table 2). According to [27], the average parallelism is about 127 for 1080p videos, and a two times speedup seems to agree with our results given in the section "Case Study: GPU-Based Fast Motion Estimation."

[TO FULLY HARNESS THE COMPUTATIONAL POWER IN THE GPU, ONE WOULD NEED TO DESIGN THE ALGORITHM TO UTILIZE THE MASSIVE NUMBER OF PROCESSING CORES IN PARALLEL.]

MOTION COMPENSATION ON GPUs

GPU-based MC has been proposed by [10] and [29] for Windows media video (WMV) and H.264 video decoding, respectively. MC requires a lot of computations, since video coding standards allow MVs to point to subpixel locations (e.g., half-pel or quarter-pel) and intensive pixel interpolation would be necessary to generate the prediction samples for motion displacements with fractional values. For example, in H.264, a half-pel sample is generated from six other samples using a six-tap interpolation filter. And to generate a quarter-pel sample it may require an additional linear interpolation.

The work in [10] discusses techniques to address the overflow and rounding problem in interpolation that arose in MC. Note that MC can be parallelized since each block can be processed independently using its motion vector information, and this is implemented by a pipeline of vertex/pixel shader procedures in [10]. In their GPU implementation, they use a multipass technique that handles the residuals and rounding control parameter in a separate pass to avoid overflow while preserving the precision. In addition, [10] discusses how different modules in video decoding can be partitioned between the CPU and the GPU, and how the CPU computation can be maximally overlapped with the GPU computation (this will be further discussed). Simulation results suggest that, in a system with an INTEL Pentium III 667 MHz GPU and a NVIDIA GeForce3 Ti200 GPU, by leveraging the GPU the system can achieve more than three times of speedup, and it is possible to achieve real-time WMV (version 8) decoding of HD video of resolution up to 1280 × 720 [10].

[TABLE 2] COMPARISON BETWEEN THE PARALLEL H.264 INTRAPREDICTION ON GPU PROPOSED IN [27] AND CONVENTIONAL H.264 INTRAPREDICTION ON THE CPU. THE NUMBERS ARE THE RATIOS OF THE CPU RUNNING TIME TO THE GPU RUNNING TIME. NOTE THAT THE GPU RUNNING TIME INCLUDES ALL THE DATA TRANSFER OVERHEAD.

	QP = 28	QP = 36	QP = 44
CIF:			
FLOWER_CIF	1.14	1.12	1.14
PARIS_CIF	1.12	1.14	1.12
MOBILE_CIF	1.14	1.12	1.12
AVERAGE (CIF)	1.13	1.13	1.13
1,280 × 720:			
CREW	1.38	1.40	1.37
NIGHT	1.49	1.42	1.39
CITY	1.48	1.47	1.43
AVERAGE (1,280 × 720)	1.45	1.43	1.39
1,920 × 1,080:			
BLUE_SKY	1.90	1.82	1.73
RIVERBED	1.93	1.82	1.76
STATION	1.89	1.81	1.80
AVERAGE (1,920 × 1,080)	1.91	1.82	1.76

TASK PARTITION BETWEEN THE CPU AND THE GPU

To obtain competitive system performance, the CPU and the GPU need to be considered together for encoding/decoding. Investigating the optimal partition of computation tasks between the CPU and the GPU, however, could be very involved, and it requires serious evaluation on many issues. For example:

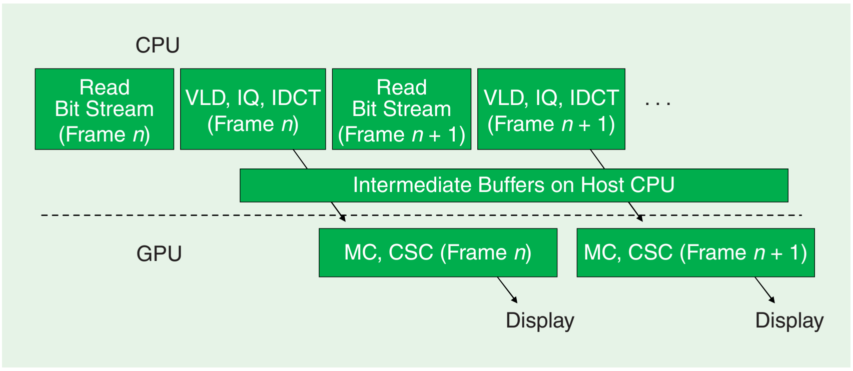
- It is necessary to investigate how to allocate the tasks such that the GPU computation can overlap with the CPU computation as much as possible, thereby achieving maximal parallel processing.
- Since the bandwidth between the GPU memory and main memory could be slow, it is important to investigate how to minimize the data transfer between main memory and the GPU memory.
- It is also important to study and evaluate which modules in the encoding/decoding flow can be efficiently offloaded to the GPU, while others would be executed on CPU.

Focusing on WMV decoding, [10] proposes a partition strategy where the whole feedback loop, including MC and color space conversion (CSC), is offloaded to the GPU. By doing so, they can avoid transferring the data back from the GPU to the CPU. Since read-backs from the GPU memory to main memory could be slow due to common asymmetric implementation of the memory bus [10], such read-backs should be minimized. Figure 6 depicts the partition strategy. Note that while the GPU is performing MC and CSC of frame n, CPU would be performing variable-length decoding (VLD), inverse quantization (IQ), and inverse DCT (IDCT) of the frame $n + 1$. Note also that intermediate memory buffer is used between the CPU and the GPU to absorb the jitters in CPU/GPU processing time. Simulation results in [10] suggest intermediate buffer size of four frames can considerably improve the overall decoding speed.

While [11] has discussed some issues (e.g., bandwidth requirement) on offloading ME to the GPU, there seems to be no prior work on rigorous investigation on how video encoding may be partitioned between the CPU and the GPU. We remark that the GPU implementation of several important encoding modules (including ME, intramode decision, MC, and transform) have been investigated in the past, while that of deblocking filter and entropy coding need further research.

CASE STUDY: GPU-BASED FAST MOTION ESTIMATION

To illustrate some design considerations in using GPUs for video coding, we discuss in detail in this section a GPU-based fast ME (the GPU ME code was developed by the authors based on the H.264 JM 14.2 reference software). The focuses are on how to address the data dependency in the algorithm to harness the parallel processing capability of GPUs, and on how to tradeoff the speedup with RD performance.


CPU
Read Bit Stream (Frame n)
VLD, IQ, IDCT (Frame n)
Read Bit Stream (Frame n + 1)
VLD, IQ, IDCT (Frame n + 1)
. . .
Intermediate Buffers on Host CPU
GPU
MC, CSC (Frame n)
MC, CSC (Frame n + 1)
Display
Display

[FIG6] **Task partitioning in WMV decoding proposed by [10].**

FAST MOTION ESTIMATION

Our GPU implementation of fast ME is based on *simplified unsymmetrical multihexagon search (smpUMHexagonS)* [42], which is one of the fast ME algorithms adopted by the H.264 JM reference software. We select smpUMHexagonS because it can achieve very good tradeoff between computational complexity and coding efficiency. For example, on a Pentium 4 CPU it was reported smpUMHexagonS can achieve up to 94% reduction in ME execution time with comparable RD efficiency, when compared with the fast full search in the JM software [42]. In addition, smpUMHexagonS is quite compact, so it could meet the memory constraint of the GPU. In our implementation, all the GPU kernels that deal with integer-pel estimation have about 600 lines of code.

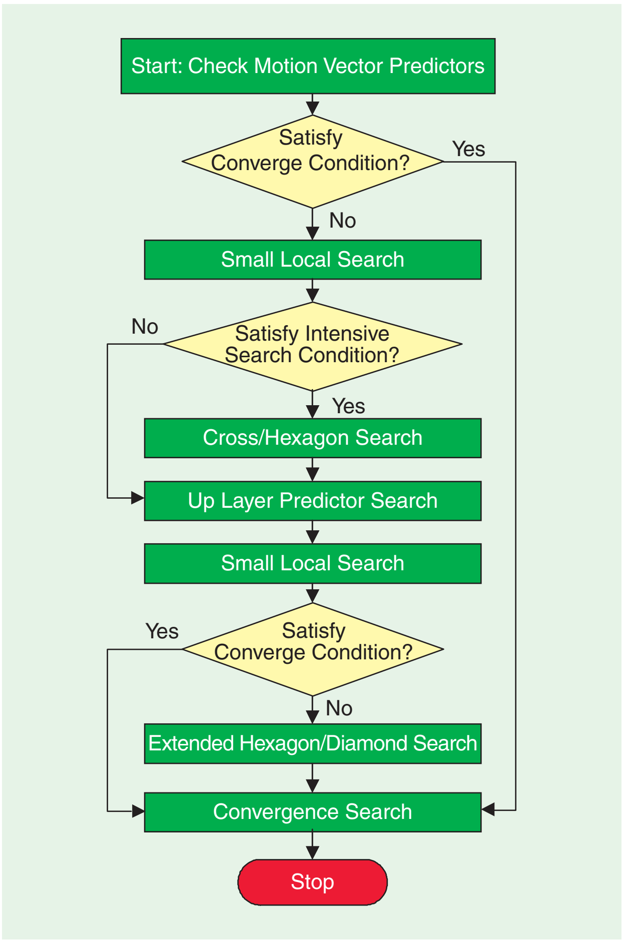
Figure 7 depicts the flow chart of smpUMHexagonS. For each MB, smpUMHexagonS computes the MVs for all the MB partitions (16×16, 16×8, $\dots$ 4×4). MVs are selected by minimizing the Lagrangian cost $D + \lambda R$, where D is the SAD between the current block and the candidate, and R is the bit-rate to encode the MV. In smpUMHexagonS, computation reduction is achieved mainly by sampling the search space judiciously, using several techniques including motion vector prediction, different search patterns (cross, hexagon, and diamond) and early termination. In particular, MVs from spatially adjacent blocks and from other MB partitions are used to initialize the search for the current partition. Notice that as depicted in Figure 7, smpUMHexagonS uses several tests to determine if the search (of the current partition) can be terminated based on the minimum cost computed so far. As a result, different MBs with different contents may undergo different processing paths (which is typical in many fast ME algorithms [47]), and this may affect the performance of the GPU implementation.

THE GPU IMPLEMENTATION USING TILING

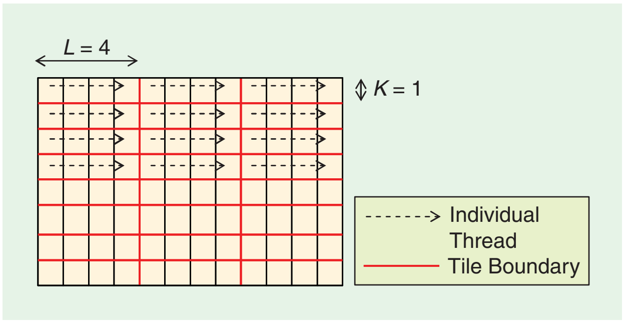
To utilize the parallelism in the GPU, we partition the current frame into multiple tiles, and each tile contains K (height) $\times$ L (width) MBs. For example, Figure 8 depicts the case with $K = 1$, $L = 4$. Each tile is processed by a single GPU thread, i.e., each thread processes $K \times L$ MBs in a tile sequentially, and different tiles are processed by different independent threads concurrently on the GPU.

Following from the discussion in the section "Fast Motion Estimation," individual MBs are not independent under smpUMHexagonS. In particular, MBs depend on their neighbors in the following ways:

- First, to compute the rate term R in the Lagrangian cost the MVs of the neighboring MBs are required. If a neighboring MB belongs to another tile, we assume its motion vector is equal to zero in computing R. Therefore, with tiling, the


Start: Check Motion Vector Predictors
Satisfy Converge Condition?
Yes
No
Small Local Search
No
Satisfy Intensive Search Condition?
Yes
Cross/Hexagon Search
Up Layer Predictor Search
Small Local Search
Yes
Satisfy Converge Condition?
No
Extended Hexagon/Diamond Search
Convergence Search
Stop

[FIG7] **Fast ME using smpUMHexagonS [42]. The figure depicts the steps for integer-pel search for an MB partition.**



[FIG8] GPU-based fast ME: the current frame is divided into multiple tiles to facilitate parallel processing in ME. Here each square represents an MB.

computed Lagrangian cost may not be very accurate and suboptimal MVs may be chosen by smpUMHexagonS as a result. The impact of tiling in this case depends on the value of λ and hence the target bit rate. For low bit-rate applications (rate constrained), encoders would focus more on rate efficiency, and large λ would be chosen and the rate term would dominate the Lagrangian cost [48]. Tiling therefore shall have a more pronounced negative impact on the performance of smpUMHexagonS for low bit-rate applications (since tiling affects the rate term).

- Second, smpUMHexagonS (and many other fast ME [47]) uses motion vector prediction, i.e., MVs of the neighboring MBs are used to initialize the search. Under tiling, some information about neighboring MVs is not available, and this may result in poor-quality initial search points, and suboptimal MVs may get selected at the end of the search (hence the RD performance is compromised). Moreover, since smpUMHexagonS employs early termination, poor initial points may also result in longer processing time, as more search points would need to be examined until the cost is small enough to terminate the search (e.g., we observe about 4% increase in the ME processing time when encoding the HD 720p sequence *Harbor* using tiling $K = 1, L = 1$ in the sequential smpUMHexagonS).

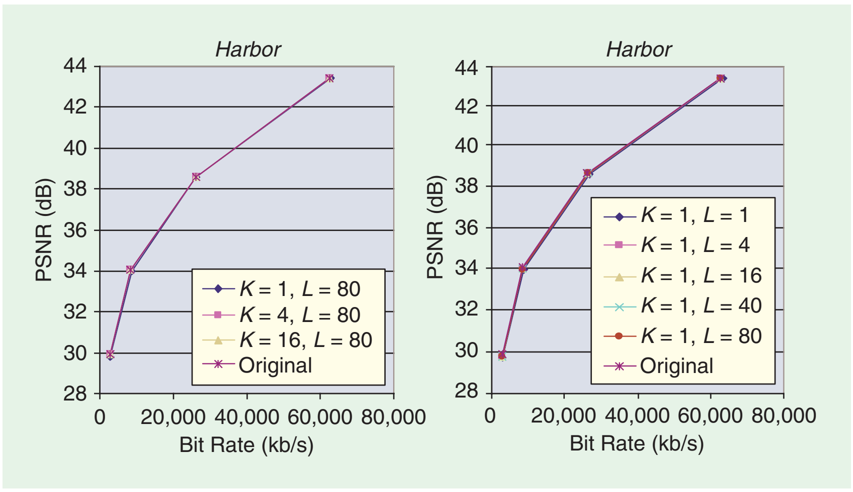
The above discussions are also applicable to many other fast ME algorithms. Note that in our simulation, tiling is used only in ME to facilitate the GPU computation, and the rest of the encoding proceeds in the same manner as in the reference software. Therefore, our tiling is different from other partitioning ideas such as slice [47], where individual partitions are treated independently in most of the encoding.

EXPERIMENTS

To examine the performance of the GPU-based fast ME using tiling, we conduct experiments on PCs equipped with one GeForce 8800 GTS PCIe graphics card with 96 stream processors [15], and an Intel Core 2 Quad Q9400 2.66 GHz CPU with 3.23 GB of RAM. We use NVIDIA's Compute Unified Device Architecture (CUDA) [39] to implement the GPU code. We choose CUDA solely because of the availability of the NVIDIA device in our laboratory, and we remark that there are other well-designed GPU programming models such as ATI CTM [49], Stream Computing SDK, and Brook+ [50].

We first evaluate how tiling may affect the RD performance. We use JM 14.2 to encode HD 720p sequences (1280×720, 60 frame per second) *Crew*, *City*, *Harbor* and *Night* (We focus on encoding HD videos because of its high computational requirement, and because of the growing interest on HD contents.) We use H.264 high profile with search range of 64. All the pictures are encoded as P-frames except the initial I-frame. Figure 9 depicts the RD performance with different tile sizes for the *Harbor* sequence. As shown in the figure the impact of tiling is small in this case until tile size is down to $K = 1, L = 1$, when the degradation is about 0.2 dB compared to the original reference software (with smpUMHexagonS). Table 3 shows the average peak signal-to-noise ratio (PSNR) degradation and the average increase in bit rate using different tile sizes, measured by Bjontegaard Delta PSNR (BDPSNR) and Bjontegaard Delta bit rate (BDBR), respectively. Note that BDPSNR and BDBR are used frequently in the video standardization community [51]. The results suggest tiling may lead to average degradation between 0.08 dB to 0.4 dB for these sequences with tile size $K = 1, L = 1$.

We then discuss how tiling may affect the speedup. Table 4 shows the GPU execution time (in integer-pel ME) with different tile sizes, and Figure 10 shows the speedup between the GPU implementation (with tiling and using parallel processing on multicore) and the sequential CPU implementation (without tiling and using sequential processing on a single core). Comparison with parallel program code on multiple CPU cores will be discussed next. The GPU execution time includes the overhead to transfer the video frames from system memory to the GPU memory. Compiler optimization is



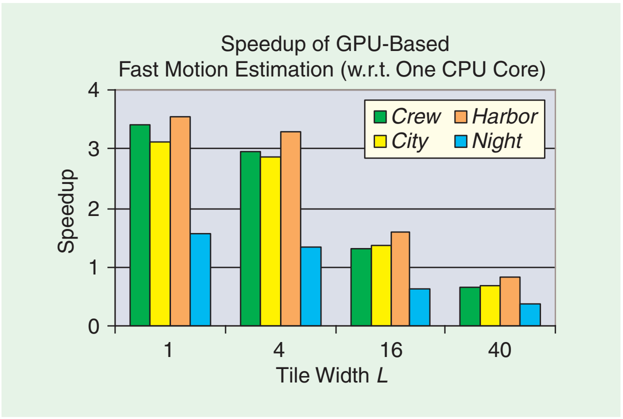
[FIG9] RD performance of *Harbor* with different tile sizes in fast ME. Here "original" refers to the reference software (that is, without tiling).

[TABLE 3] TRADEOFF BETWEEN TILE SIZE AND RD PERFORMANCE. AVERAGE INCREASE IN BIT RATE AND AVERAGE PSNR DEGRADATION ARE COMPUTED WITH RESPECT TO THE REFERENCE SOFTWARE (I.E., WITHOUT TILING).

		CREW		*CITY*		*HARBOR*		*NIGHT*	
TILE SIZE	NUMBER OF TILES	BDBR (%)	BDPSNR (DB)	BDBR (%)	BDPSNR (DB)	BDBR (%)	BDPSNR (DB)	BDBR (%)	BDPSNR (DB)
$K = 1,\ L = 1$	3,600	3.135	−0.082	12.933	−0.407	5.578	−0.221	4.636	−0.17
$K = 1,\ L = 4$	900	3.081	−0.079	11.115	−0.352	2.385	−0.094	3.546	−0.13
$K = 1,\ L = 16$	225	3.116	−0.08	11.171	−0.35	2.246	−0.089	3.415	−0.125
$K = 1,\ L = 40$	90	3.224	−0.083	10.821	−0.339	2.205	−0.087	3.4	−0.124
$K = 4,\ L = 80$	12	0.63	−0.016	1.412	−0.044	0.57	−0.022	1.19	−0.043
$K = 16,\ L = 80$	3	0.094	−0.003	0.261	−0.008	0.07	−0.003	0.161	−0.006

applied to both the GPU program and the CPU program. However, both the GPU/CPU code have rooms for further speed improvement. In particular, the GPU code stores pixel data in global memory (off-chip memory), which has considerable access latency [39]. As ME is fairly memory access intensive (SAD calculation performs only three mathematical operations per two memory loads, giving an arithmetic intensity of 1.5, which is rather small for the GPU computation [37]), such latency may impact the GPU code performance. Therefore, the code can be improved by judicious use of shared memory (on-chip memory) [37]. As shown in Figure 10 speedup increases with smaller tile size, as more independent threads can be scheduled. This is particularly important in the current GPU code to hide the memory access latency. Note also that different sequences have different GPU execution time and speedups, as different video contents may lead to different execution paths in smpUMHexagonS and different amount of penalty incurred by execution serialization. Figure 10 suggests speedups of 1.5–3.5 can be achieved in integer-pel smpUMHexagonS in these sequences using tile size $K = 1, L = 1$.

Figure 11 shows the speedup between the GPU implementation and a parallel CPU implementation using the four CPU cores on the Intel Core 2 Quad. To achieve parallel CPU processing, the current frame is partitioned into four tiles of equal number of MB rows (i.e., L = width of the video frame in MB, K = height of the video frame in MB/4), and each tile is processed by an independent thread running on a CPU core. We use OpenMP to implement the parallel CPU program [52]. We observe the parallelization reduces the CPU running time by a factor of three approximately. Note that the theoretical maximum speedup of four cannot be achieved by this parallelization strategy, as smpUMHexagonS may spend different execution time on each MB and optimal load balancing cannot be achieved by simple tiling. Figure 11 suggests the running time of the GPU implementation



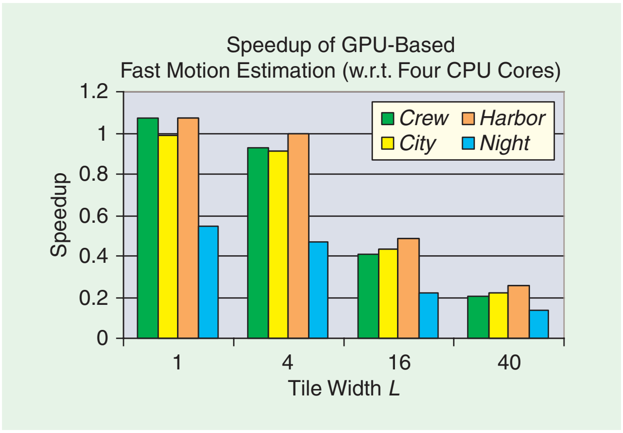
[FIG10] Tradeoff between tile width and speedup (tile height K is equal to one). Speedup is the ratio of the CPU running time (sequential program code on one CPU core) to the GPU running time (including data transfer overhead).

and the parallel CPU implementation can be comparable in some cases (while the GPU implementation incurs some RD performance degradation as depicted in Table 3).

In the experiment, we observe the overhead to transfer a frame from the CPU to the GPU is about 1.6 ms, and this is about 0.1–0.2% of the running time of integer pel ME (see Table 4). In general, data transfer overhead could be a less serious issue in interframe encoding compared with decoding and

[TABLE 4] GPU EXECUTION TIME FOR FAST INTEGER-PEL MOTION ESTIMATION WITH DIFFERENT TILE WIDTH (TILE HEIGHT K IS EQUAL TO ONE). DATA TRANSFER OVERHEADS ARE INCLUDED.

		GPU EXECUTION TIME (MS)			
TILE WIDTH	NUMBER OF THREADS	*CREW*	*CITY*	*HARBOR*	*NIGHT*
$L = 1$	3,600	835.05	927.32	1,248.95	1,688.50
$L = 4$	900	959.16	1,005.55	1,341.45	1,975.95
$L = 16$	225	2,169.25	2,108.71	2,763.79	4,175.44
$L = 40$	90	4,373.63	4,165.28	5,318.38	6,920.73



[FIG11] Tradeoff between tile width and speedup (tile height K is equal to one). Speedup is the ratio of the CPU running time (parallel program code on four CPU cores) to the GPU running time (including data transfer overhead).

intraframe encoding, since interframe encoding requires a significantly larger amount of execution time in general.

Finally, we would like to remark that both the CPU and the GPU implementations can be further optimized. Our discussion has suggested that it is nontrivial to achieve the peak performance offered by these multicore devices in video coding, and more algorithm research and instruction level optimization would be needed.

IT IS NONTRIVIAL TO ACHIEVE THE PEAK PERFORMANCE OFFERED BY THESE MULTICORE DEVICES IN VIDEO CODING, AND MORE ALGORITHM RESEARCH WOULD BE NEEDED.

CONCLUSIONS AND DISCUSSION

We have reviewed previous work on using GPUs for video encoding and decoding. In particular, we have discussed how some video coding modules can be implemented in certain ways to expose as much data parallelism as possible, so that the massive parallel-processing capability of GPUs can be fully utilized. Simulation results in previous work suggest GPU-based implementations can achieve considerable speedups for some of the most computation-intensive modules in video coding. Therefore, it could be a cost-effective approach to leverage the computing power of GPUs to meet the data processing requirement in video coding. We have also discussed an example to partition the video decoding flow between CPUs and GPUs to achieve maximum overlapping of computation. In addition, we have discussed a GPU-based fast ME and examined the tradeoff between speedup and RD performance.

There are several related research issues. First, there seem to be no studies on partitioning the encoding flow between CPUs and GPUs. Second, with the availability of many different video formats (e.g., SD and HD) and coding standards there is a growing need to transcode one encoded video format to another [53]–[55]. However, while there are a few commercial transcoding applications available [56], [57], there seems to be no prior work on investigating the optimal usage of GPUs for transcoding. Note that unlike video encoding/decoding, there is no standard algorithm for video transcoding, and there are many previously proposed approaches that achieve a wide range of transcoding quality with different complexity requirements [53]. This complicates the study of GPU-based transcoding.

ACKNOWLEDGMENTS

This work has been supported in part by the Innovation and Technology Commission (GHP/048/08) and the Research Grants Council (RPC07/08.EG22 and 610109) of the Hong Kong Special Administrative Region, China. The authors would also like to thank the editor and the anonymous reviewers for their helpful comments, which significantly improved the article.

AUTHORS

Ngai-Man Cheung (ncheung@stanford.edu) received the Ph.D. degree from the University of Southern California (USC), Los Angeles, in 2008. He is currently a postdoctoral researcher with the Information Systems Laboratory, Stanford University, Stanford, California. He was a research associate with Hong Kong University of Science and Technology (HKUST) from 2008 to 2009. His research interests include multimedia signal processing and compression, multimedia retrieval, and multicore and GPU applications. He received paper awards from *EURASIP Journal of Advances in Signal Processing*, 2007 IEEE International Workshop on Multimedia Signal Processing, IS&T/SPIE VCIP 2008, and the USC's Department of Electrical Engineering. He has five granted U.S. patents with others pending.

Xiaopeng Fan (eexp@ust.hk) received the B.S. and M.S. degrees in 2001 and 2003, respectively, in computer science from the Harbin Institute of Technology, China. He is currently pursuing the Ph.D. degree in the Department of Electrical and Computer Engineering, HKUST. He worked with Intel China Software Laboratory, Shanghai, from 2003 to 2005. His current research interests are in image/video coding and processing, video streaming, and wireless communication.

Oscar C. Au (eeau@ust.hk) received the Ph.D. from Princeton University in 1991. He is with the Department of Electrical and Computer Engineering of HKUST. Some of his research interests include video coding, ME, rate control, denoising, deinterlacing, multiview coding, scalable video coding, distributed video coding, and the GPU. He has over 260 papers and more than 50 patents, with some accepted into MPEG-4 and AVS standards. He is an associate editor for six journals including *IEEE Transactions on Circuits and Systems for Video Technology, IEEE Transactions on Image Processing*, and *IEEE Transactions on Circuits and Systems–I: Fundamental Theory and Applications*. He is involved with many technical and steering committees. He is the chair of the IEEE International Conference on Multimedia and Expo 2010 and the general cochair of the 2010 Packet Video Workshop. He won the 2007 SiPS and PCM Best Paper Awards.

Man-Cheung Kung (mckung@ust.hk) received the B.Eng. degree in computer engineering and the M.Phil. degree in electronic and computer engineering from HKUST, in 2006 and 2008, respectively. From 2005 to 2006, he was with the Applied Science and Technology Research Institute Company Ltd., Shatin, Hong Kong. He is currently the research and development engineer of the Visual Perception Dynamics Labs (Mobile) Ltd., Shatin, Hong Kong. His research interests include general-purpose computation on GPUs, fast ME, intraprediction, image demosaicking, and subpixel rendering.

REFERENCES

[1] Y. Wang, J. Ostermann, and Y.-Q. Zhang, *Video Processing and Communications*. Englewood Cliffs, NJ: Prentice-Hall, 2002.

[2] T. Wiegand, “Joint final committee draft for joint video specification H.264,” *Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG,* Tech. Rep. JVT-D157, 2002.

[3] AVS Video Expert Group, Information Technology—Advanced Audio Video Coding Standard Part 2: Video, in Audio Video Coding Standard Group of China (AVS), Doc. AVS-N1063, Dec. 2003.

[4] L. Yu, F. Yi, J. Dong, and C. Zhang, "Overview of AVS-Video: Tools, performance and complexity," in *Proc. Visual Communications and Image Processing (VCIP)*, 2005, pp. 679–690.

[5] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, July 2003.

[6] G. Sullivan, J.-R. Ohm, A. Ortega, E. Delp, A. Vetro, and M. Barni, "dsp Forum—Future of video coding and transmission," *IEEE Signal Processing Mag.*, vol. 23, no. 6, pp. 76–82, Nov. 2006.

[7] J. Loomis and M. Wasson. (2007). *VC-1 technical overview* [Online]. Available: http://www.microsoft.com/windows/windowsmedia/

[8] B. G. Haskell, A. Puri, and A. N. Netravali, *Digital Video: An Introduction to MPEG-2*. Berlin: Springer-Verlag, 1996.

[9] L. Fan, S. Ma, and F. Wu, "An overview of AVS video standard," in *Proc. IEEE Int. Conf. Multimedia and Expo*, 2004, pp. 423–426.

[10] G. Shen, G. Gao, S. Li, H. Shum, and Y. Zhang, "Accelerate video decoding with generic GPU," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 5, pp. 685–693, 2005.

[11] A. Mather, "GPU-accelerated video encoding," in *Proc. SIGGRAPH Tech Talks*, 2008.

[12] J. Kruger and R. Westermann, "Linear algebra operators for GPU implementation of numerical algorithms," in *Proc. Int. Conf. Computer Graphics and Interactive Techniques*, Los Angeles, CA, 2005, p. 234.

[13] O. Fialka and M. Cadik, "FFT and convolution performance in image filtering on GPU," in *Proc. Conf. Information Visualization*, 2006, pp. 609–614.

[14] A. Brunton and J. Zhao, "Real-time video watermarking on programmable graphics hardware," in *Proc. Canadian Conf. Electrical and Computer Engineering*, 2005, pp. 1312–1315.

[15] NVIDIA, "NVIDIA GeForce 8800 architecture technical brief," NVIDIA, Tech. Rep., 2006.

[16] NVIDIA. (2009). *CUDA—Compute unified device architecture* [Online]. Available: http://www.nvidia.com/object/cuda_home.html

[17] Khronos Group. (2009). OpenCL—The open standard for parallel programming of heterogeneous systems [Online]. Available: http://www.khronos.org/opencl/

[18] M. Macedonia, "The GPU enters computing's mainstream," *Computer,* vol. 36, no. 10, pp. 106–108, Oct. 2003.

[19] F. Kelly and A. Kokaram, "General purpose graphics hardware for accelerating motion estimation," in *Proc. Irish Machine Vision and Image Processing Conf.*, 2003.

[20] Y. Lin, P. Li, C. Chang, C. Wu, Y. Tsao, and S. Chien, "Multi-pass algorithm of motion estimation in video encoding for generic GPU," in *Proc. IEEE Int. Symp. Circuits and Systems*, 2006.

[21] C.-W. Ho, O. Au, G. Chan, S.-K. Yip, and H.-M. Wong, "Motion estimation for H.264/AVC using programmable graphics hardware," in *Proc. IEEE Int. Conf. Multimedia and Expo*, 2006, pp. 2049–2052.

[22] M. Kung, O. Au, P. Wong, and C. Liu, "Block based parallel motion estimation using programmable graphics hardware," in *Proc. Int. Conf. Audio, Language and Image Processing*, 2008, pp. 599–603.

[23] M. L. Schmit, R. M. Rawther, and R. Giduthuri, "Software video encoder with GPU acceleration," U.S. Patent 20 090 016 430, 2009.

[24] G. Jin and H.-J. Lee, "A parallel and pipelined execution of H.264/AVC intra prediction," in *Proc. IEEE Int. Conf. Computer and Information Technology*, 2006, p. 246.

[25] W. Lee, S. Lee, and J. Kim, "Pipelined intra prediction using shuffled encoding order for H.264/AVC," in *Proc. IEEE Region 10 Conf. (TENCON)*, 2006, pp. 1–4.

[26] M. Kung, O. Au, P. Wong, and C. Liu, "Intra frame encoding using programmable graphics hardware," in *Proc. Pacific Rim Conf. Multimedia (PCM)*, 2007, pp. 609–618.

[27] N.-M. Cheung, O. Au, M. Kung, H. Wong, and C. Liu, "Highly parallel rate-distortion optimized intra mode decision on multi-core graphics processors," *IEEE Trans. Circuits Syst. Video Technol. (Special Issue on Algorithm/Architecture Co-Exploration of Visual Computing)*, vol. 19, no. 11, pp. 1692–1703, Nov. 2009.

[28] A. Obukhov and A. Kharlamov, "Discrete cosine transform for 8×8 blocks with CUDA," NVIDIA, Santa Clara, CA, Tech. Rep., Oct. 2008.

[29] B. Pieters, D. Van Rijsselbergen, W. De Neve, and R. Van de Walle, "Motion compensation and reconstruction of H.264/AVC video bitstreams using the GPU," in *WIAMIS '07: Proc. 8th Int. Workshop Image Analysis for Multimedia Interactive Services*. Washington, DC: IEEE Comput. Soc., 2007, p. 69.

[30] A. Hirvonen and T. Leppanen, "H.263 video decoding on programmable graphics hardware," in *Proc. 5th IEEE Int. Symp. Signal Processing and Information Technology*, Dec. 2005, pp. 902–907.

[31] B. Han and B. Zhou, "Efficient video decoding on GPUs by point based rendering," in *GH '06: Proc. 21st ACM SIGGRAPH/EUROGRAPHICS Symp. Graphics Hardware*. New York, NY: ACM, 2006, pp. 79–86.

[32] NVIDIA. (2009). *PureVideo overview* [Online]. Available: http://www.nvidia.com/object/purevideo_overview.html

[33] J. Munkberg, P. Clarberg, J. Hasselgren, and T. Akenine-Moller, "High dynamic range texture compression for graphics hardware," *ACM Trans. Graph.*, vol. 25, no. 3, 2006, pp. 698–706.

[34] Y.-K. Chen, W. Li, J. Li, and T. Wang, "Novel parallel Hough transform on multi-core processors," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, 2008, pp. 1457–1460.

[35] J. Fung, S. Mann, and C. Aimone, "OpenVIDIA: Parallel GPU computer vision," in *Proc. ACM Multimedia*, 2005, pp. 849–852.

[36] P. M. Novotny, J. A. Stoll, N. V. Vasilyev, P. J. del Nido, P. E. Dupont, T. E. Zickler, and R. D. Howe, "GPU based real-time instrument tracking with three-dimensional ultrasound," *Med. Image Anal. (Special Issue on the 9th Int. Conf. on Medical Image Computing and Computer-Assisted Interventions—MICCAI 2006)*, vol. 11, no. 5, pp. 458–464, 2007.

[37] D. Lin, V. Huang, Q. Nguyen, J. Blackburn, C. Rodrigues, T. Huang, M. N. Do, S. J. Patel, and W.-M. W. Hwu, "The parallelization of video processing," *IEEE Signal Processing Mag.*, vol. 26, no. 6, pp. 103–112, Nov. 2009.

[38] D. Kirk and W. Hwu, "Textbook for UIUC ECE 498 AL: Programming massively parallel processors," *Draft*, 2009.

[39] NVIDIA, "CUDA programming guide," NVIDIA, Tech. Rep., 2009.

[40] J. P. Shen and M. H. Lipasti, *Modern Processor Design: Fundamentals of Superscalar Processors*. Boston: McGraw-Hill, 2005.

[41] M. Houston. (2007). *SIGGRAPH 2007 GPGPU course* [Online]. Available: http://gpgpu.org/s2007

[42] X. Yi, J. Zhang, N. Ling, and W. Shang, "Improved and simplified fast motion estimation for JM," in *Proc. JVT Meeting, Poznan, Poland,* Tech. Rep. JVT-P021, July 2005.

[43] C.-L. Yang, L.-M. Po, and W.-H. Lam, "A fast H.264 intra prediction algorithm using macroblock properties," in *Proc. IEEE Int. Conf. Image Processing (ICIP)*, 2004, pp. 461–464.

[44] R. Su, G. Liu, and T. Zhang, "Fast mode decision algorithm for intra prediction in H.264/AVC," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, 2006.

[45] M. Liu and Z.-Q. Wei, "A fast mode decision algorithm for intra prediction in AVS-M video coding," in *Proc. Int. Conf. Wavelet Analysis and Pattern Recognition*, Nov. 2007, pp. 326–331.

[46] K.-W. Yoo and H.-H. Kim, "Intra prediction method and apparatus," U.S. Patent 2005/0 089 094, 2005.

[47] I. E. Richardson, *H.264 and MPEG-4 Video Compression, Video Coding for Next-Generation Multimedia*. England: Wiley, Nov. 2003.

[48] T. Wiegand and B. Girod, "Lagrange multiplier selection in hybrid video coder control," in *Proc. IEEE Int. Conf. Image Processing (ICIP)*, 2001, pp. 542–545.

[49] Advanced Micro Devices Inc. (2006). *ATI CTM guide* [Online]. Available: http://ati.amd.com/companyinfo/researcher/documents/ATI_CTM_Guide.pdf

[50] Advanced Micro Devices Inc. (2009). *ATI stream software development kit (SDK)* [Online]. Available: http://developer.amd.com/gpu/ATIStreamSDK/Pages/default.aspx

[51] G. Bjontegaard, "Calculation of average PSNR differences between RD-curves," in *Proc. JVT Meeting,* Tech. Rep. VCEG-M33, Apr. 2001.

[52] OpenMP. (2009). *The OpenMP API specification for parallel programming* [Online]. Available: http://openmp.org/

[53] A. Vetro, C. Christopoulos, and H. Sun, "Video transcoding architectures and techniques: An overview," *IEEE Signal Processing Mag.*, vol. 20, no. 2, pp. 18–29, Mar. 2003.

[54] H. Sun, X. Chen, and T. Chiang, *Digital Video Transcoding for Transmission and Storage*. New York: CRC, 2005.

[55] I. Ahmad, X. Wei, Y. Sun, and Y.-Q. Zhang, "Video transcoding: An overview of various techniques and research issues," *IEEE Trans. Multimedia*, vol. 7, no. 5, pp. 793–804, Oct. 2005.

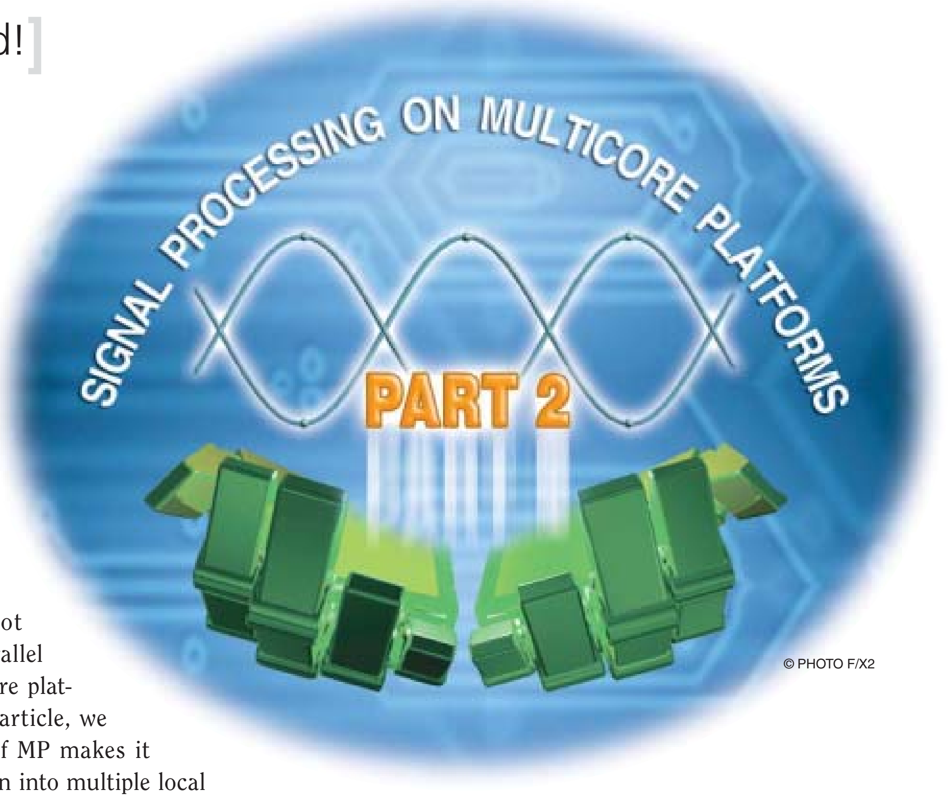
[56] Techreport. (2009). *Badaboom 1.0 uses Nvidia GPUs to transcode video* [Online]. Available: http://techreport.com/discussions.x/15763

[57] AnandTech. (2009). *AVIVO video converter Redux and ATI stream quick look* [Online]. Available: http://www.anandtech.com/video/showdoc.aspx?i=3578 [SP]

Laurent Daudet

Audio Sparse Decompositions in Parallel

[Let the greed be shared!]



Greedy methods are often the only practical way to solve very large sparse approximation problems. Among such methods, matching pursuit (MP) is undoubtedly one of the most widely used, due to its simplicity and relatively low overhead. Since MP works sequentially, however, it is not straightforward to formulate it as a parallel algorithm, to take advantage of multicore platforms for real-time processing. In this article, we investigate how a slight modification of MP makes it possible to break down the decomposition into multiple local tasks, while avoiding blocking effects. Our simulations on audio signals indicate that this parallel local matching pursuit (PLoMP) gives results comparable to the original MP algorithm but could potentially run in a fraction of the time—on-the-fly sparse approximations of high-dimensional signals should soon become a reality.

The last two decades have witnessed the advent of sparsity as a major paradigm in many areas of signal processing. Sparsity is the key to success for most of state-of-the-art multimedia compression schemes, such as still image coding (for instance JPEG-2000 [1]) and audio coding (MPEG-2/4 advanced audio coding (AAC) [2]). Basically, sparsity exploits the fact that there exist bases in which, for most natural signals, only a few of the transform coefficients are sufficient to provide a good approximation. To be more precise, given a signal, by sorting its transform coefficients by absolute decaying order, one observes a fast decay, typically a power law with some large negative exponent. This ability to concentrate most of the energy of the signals into only a few of the transform coefficients naturally leads to an increased coding efficiency. For instance, in the JPEG-2000 image coder based on an orthogonal two-dimensional (2-D) dyadic wavelet transform, only portions of the image that correspond to sharp transitions (at objects' edges for instance) will lead to large wavelet coefficients: most of the bit budget is spent in these regions. Similarly, in the

Digital Object Identifier 10.1109/MSP.2009.935388

1053-5888/10/$26.00©2010IEEE

modified discrete cosine transform (MDCT) domain, i.e., the cosine-based filterbank of AAC, the large coefficients represent the perceptually dominant sinusoidal harmonics of the musical content. With a smart quantization of these few large transform coefficients, and an efficient indexing of their parameters, these coders achieve a high compression ratio at virtually no loss of perceptual quality (typically at 1:20 or more for JPEG-2000, and 1:6 for MPEG-2/4 AAC).

[THE LAST TWO DECADES HAVE WITNESSED THE ADVENT OF SPARSITY AS A MAJOR PARADIGM IN MANY AREAS OF SIGNAL PROCESSING.]

But why does it work—where does this energy compaction come from? This is basically due to the fact that the transform basis elements "look like" elementary components of the analyzed signals: 2-D wavelets look like the edges of objects in images and discrete cosines look like the harmonics of musical notes. Only a few of these elementary building blocks are thus sufficient to well approximate the signals. It should be emphasized that the corresponding algorithms have a relatively low complexity: in the orthonormal bases described above (discrete wavelets/MDCT), selecting the set of significant coefficients involves a simple thresholding.

However, with all these nice properties also comes a major flaw: orthogonal bases are usually too rigid to accommodate even basic invariance properties of our signals. For instance, standard wavelet image codecs do not have shift- nor rotation-invariance: if the object pictured is slightly moved and/or tilted then its transform representation is fundamentally different. Similarly, in the audio domain, the MDCT is not shift-invariant: depending on the exact position of the signal with respect to the analysis frames, the transform coefficients may be radically different and so is the compression efficiency. Furthermore, the single frame length of the MDCT is inappropriate to simultaneously represent both the very sharp attack transients at the onset of percussive notes (where very short windows are desirable), and the long harmonics of tones (where a high frequency resolution is needed, hence long frame sizes).

To achieve higher sparsity, the key is to use decomposition spaces that have more basis vectors than orthonormal bases, and thereby more flexibility. These extended bases are called overcomplete, or redundant bases. Would you like time-shift invariance in your audio transform? The discrete Gabor transform, as known for one-dimensional (1-D) signals as short-time Fourier transform, is nearly shift-invariant at the cost of (at least) doubling the size of the basis. Would you like shift-invariance in your image coder? The dual-tree complex wavelet [3] offers you this (approximately), but it is now four times overcomplete. With such overcomplete bases, sparsity is improved: basically, the larger the basis (the more redundant) the more likely it is that, for every local feature of the signal, there will be one basis vector that nearly fits. Overcompleteness brings flexibility and generality in the class of signals that are sparsely represented. Recently, prototype codecs have been developed in many fields of multimedia, for example image [4], audio [5], [6], or video [7]. At very low bit rates (i.e., very high compression ratio), these new codecs outperform standard codecs based on orthogonal transforms. Besides coding, there are also many applications that benefit from this sparse energy compaction property [8], for instance information extraction [9], [10], source localization [11], or source separation [12], [13].

So, why are these sparse overcomplete transforms not used in widespread, standardized codecs? This is primarily due to their computational cost. Most modern multimedia applications now require on-the-fly encoding, and here the processing time for audio in [5] is typically 100 times slower than real time, or the codec of [4] needs up to one hour for just one still image! As we shall see, as soon as the transform basis becomes overcomplete, selecting the coefficients is not as easy as the simple thresholding used in the orthogonal case: now the coefficients are no more mutually independent, and thus selecting one coefficient affects all the others. The delicate art of sparse approximation is a constant struggle against combinatorial complexity, as described in the next section.

The study presented in this article is a proof of concept that, in the framework of sparse overcomplete decompositions, real-time processing of large streams of multimedia data can potentially be achieved with very simple parallel algorithms based on the simple and effective MP algorithm.

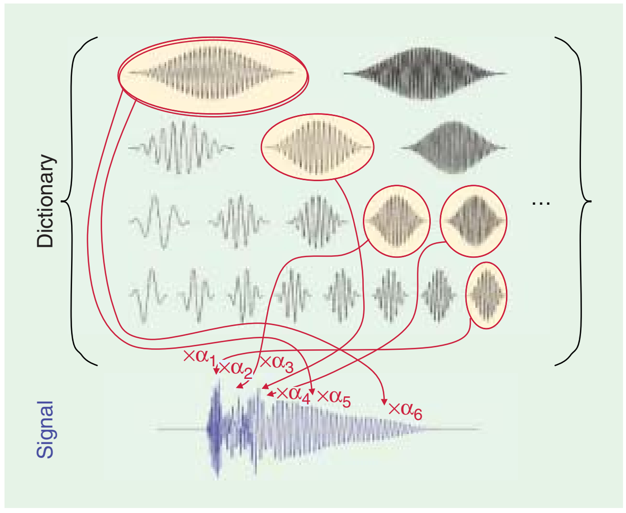
SPARSE OVERCOMPLETE DECOMPOSITIONS: A MASSIVE COMBINATORIAL PROBLEM

Let us introduce some notation: let $\mathrm{x} \in \mathbb{R}^N$ be a length-N real signal. We would like to find an optimal (sparsest) decomposition of x, as a linear combination of few elementary "atoms" g_γ that belong to a large, overcomplete set, called dictionary $\mathcal{D} = \{\mathrm{g}_\gamma, \gamma \in \Gamma\}$. The term "atom" is a slight abuse of language, as it literally implies that the decomposition is unique, which in general is not true with overcomplete dictionaries. However, this analogy with physics dates back from the 1947 pioneering work of D. Gabor on the acoustical quanta [14] and is still prevailing today. In general, dictionaries are collections of elementary waveforms that are chosen to represent the local characteristics of the class of signals under study. In the simplest case, dictionaries are concatenations of orthonormal bases (e.g., Fourier sinusoids + Dirac impulses, or local Fourier bases with different window sizes). Dictionaries can also be learned from a set of signals [15].

Here, we want to approximate x as a weighted sum of a known number K of atoms, for instance corresponding to our bit budget

$$\mathrm{x} \approx \sum_{k=0}^{K-1} \alpha_{\gamma_k} \mathrm{g}_{\gamma_k}. \qquad (1)$$

This is illustrated on Figure 1, with a signal represented by a linear combination of six atoms from the dictionary: the problem consists of finding the optimal scale-time-frequency



[FIG1] **Finding the best sparse decomposition of a signal, in a large dictionary of elementary waveforms, is a hard optimization problem that in general cannot be solved by brute force.**

parameters for each atom in this set, and the corresponding weights α. Next, we can formulate our direct sparse approximation problem: With dictionary $\mathcal{D} = \{\mathbf{g}_\gamma, \gamma \in \Gamma\}$ and K fixed, find the set of indices $\{\gamma_k\}_{k=0\ldots K-1}$ and corresponding coefficients $\{\alpha_{\gamma_k}\}_{k=0\ldots K-1}$ that minimize $\|\mathbf{x} - \sum_{k=0}^{K-1}\alpha_{\gamma_k}\mathbf{g}_{\gamma_k}\|_2$.

Unfortunately, this problem cannot be solved exactly for real-size problems, because of its combinatorial nature. Indeed, if M is the total number of atoms in the dictionary $\mathcal{D}$, there are $\binom{M}{K}$ combinations of indices $\{\gamma_k\}_{k=0\ldots K-1}$ to test, and for each of these finding the least-squares optimal set of coefficients $\{\alpha_{\gamma_k}\}_{k=0\ldots K-1}$ is an orthogonal projection problem, requiring (in general) the inversion of a $K \times K$ matrix.

To circumvent this problem, there are two different options. The first one is to design a similar problem that can be solved exactly. The other approach looks for an approximate solution of the original problem, obtained with a tractable algorithm. In both cases, one is left to trust that the obtained solution is not far from the optimal one.

As the rest of the article is devoted to the latter solution, let us quickly review the first option. Instead of having an exact sparsity constraint (number of nonzero elements bounded by some number K), we can look for signals that have few large coefficients and a lot of very small ones. Among all measures for this "relaxed sparsity problem," it is common practice to use the ℓ_1-norm $\|\mathbf{x}\|_1 = \sum_{k=0}^{N-1}|x(k)|$. Now, as the ℓ_1-norm has good convexity properties, it is possible to jointly optimize the data fidelity and the ℓ_1-sparsity of the set of coefficients, by standard quadratic programming [16], interior point methods [16], or a modification of least angle regression (LARS) technique [17]. There are also similar problems that can be solved more efficiently, such as the Dantzig selector [18] that only requires a linear-time program and is hence applicable to large data sets. It should be noted that these methods have received renewed attention lately in the framework of compressive sensing [19], [20], which involves similar types of optimization problems. In this context, there is currently a large activity to adapt these sparse solvers on multicore or GPUs [21]–[23].

However efficient these methods may be, we have chosen not to use them in this study. The main reason is that our long-term goal is to design a "real-time" (though with delay) sparse decomposition algorithm, applicable to an incoming stream of data. In general, the previously described algorithms deal with the signal as a whole, and are therefore more suited to an offline scenario. The simplest greedy methods described in the next section, however, can very easily be modified into a "local" implementation, amenable to on-the-fly processing. Another advantage of greedy methods is that they are natively scalable in complexity, hence usable on any hardware architecture, and virtually any signal size. Finally, they provide an intuitive view of the involved mechanisms. Accordingly, the rest of this article will only consider greedy methods.

[ALGORITHM 1] MATCHING PURSUIT (MP).

```
Require: signal $\mathbf{x} \in \mathbb{R}^N$, dictionary $\mathcal{D} = \{\mathbf{g}_\gamma, \gamma \in \Gamma\}$, maximum number of
    iterations $N_0$
Ensure: $\{\alpha_\gamma, \gamma \in \Gamma\}$ set of coefficients
  $n \leftarrow 0$ index of iteration
  $\mathbf{r}^0 \leftarrow x$ residual at initialization
  $\alpha_\gamma \leftarrow 0, \ \forall\gamma \in \Gamma$
  repeat
    $c_\gamma = \langle \mathbf{r}^n, \mathbf{g}_\gamma \rangle, \ \forall\gamma \in \Gamma$  scalar products computation (*)
    $\gamma_{opt} \leftarrow \text{argmax}_{\gamma\in\Gamma}|c_\gamma|$  atom selection stage (**)
    $\mathbf{r}^{n+1} \leftarrow \mathbf{r}^n - c_{\gamma_{opt}}\mathbf{g}_{\gamma_{opt}}$  residual update
    $\alpha_{\gamma_{opt}} \leftarrow \alpha_{\gamma_{opt}} + c_{\gamma_{opt}}$
    $n \leftarrow n + 1$
  until $n = N_0$ iterations performed or required precision reached
  Return $\{\alpha_\gamma, \gamma \in \Gamma\}$ set of decomposition coefficients, such that
  $\mathbf{x} = \sum_{\gamma\in\Gamma}\alpha_\gamma\mathbf{g}_\gamma + \mathbf{r}^n$
```

GREEDY METHODS FOR SPARSE APPROXIMATIONS

Greedy methods [24] are based on a simple divide and conquer principle: they select one atom, subtract its contribution, and iterate on the residual. The efficiency of these methods arise from the fact that, to select only one atom ($K = 1$), the direct sparse approximation problem is easily solved: the parameter γ_{opt} and the corresponding coefficient $\alpha_{\gamma_{\text{opt}}}$ that minimize $\|\mathbf{x} - \alpha_\gamma\mathbf{g}_\gamma\|_2$ are obtained by selecting the best orthogonal projection on individual atoms, i.e., by simple scalar products (correlation) between the signal and the atoms: $\gamma_{\text{opt}} = \text{argmax}_{\gamma\in\Gamma}|\langle \mathbf{x}, \mathbf{g}_\gamma\rangle|$, and $\alpha_{\gamma_{\text{opt}}} = \langle \mathbf{x}, \mathbf{g}_{\gamma_{\text{opt}}}\rangle$. This is the basis for the MP algorithm [25], whose pseudocode is given in Algorithm 1. Basically, it can be described as follows: at each iteration, find in the dictionary $\mathcal{D}$ the unit-norm atom best correlated with the signal (with the correlation computed as a scalar product, atom selection stage), subtract its contribution by (least-squares) orthogonal projection, and reiterate. There are more elaborate strategies such as orthogonal MP [26], low-complexity orthogonal MP [27], relaxed greedy algorithm [28], or gradient pursuit [29] that find a better minimizer of the error by considering the whole set of already selected atoms, from previous iterations. For the sake of simplicity, we do not consider them in this study.

Besides giving good approximate solutions of the sparse decomposition problem, MP is very appealing: first, its design is extremely simple, offering flexibility to adapt to the problem at hand; and second, it is scalable in complexity: at any given time, every subsequent iteration adds a vector and reduces the approximation error. The algorithm can be stopped at any time, depending on the available resources and/or the target precision. In a basic, most-general implementation, two steps can potentially be computational bottlenecks: the computation of the scalar products and the atom selection stage (respectively marked with (*) and (**) in Algorithm 1).

TO ACHIEVE HIGHER SPARSITY, THE KEY IS TO USE DECOMPOSITION SPACES THAT HAVE MORE BASIS VECTORS THAN ORTHONORMAL BASES, AND THEREBY MORE FLEXIBILITY.

As described above, these greedy iterative methods are essentially sequential. In the most general case, atoms have the same length as the signal itself, and any iteration is based on the residual of the previous one. Therefore, the potential gain in parallelizing is weak, and limited to subtasks such as the update of the scalar products. In practice, atoms with arbitrary shape and support are barely used, as the cost of updating the scalar products is often prohibitive (updating the scalar product of the length-N signal with P atoms requires in general $N \times P$ multiplications): one prefers fast algorithms such as the (local) FFT or the Mallat's pyramidal algorithm for the discrete wavelet transform. When such structured time-localized atoms are used, only a local update of the scalar products must be performed, considerably speeding up the process. For 1-D signals, there are now a number of flexible, optimized packages for MP, such as the open-source MP ToolKit (MPTK) [30]. However, for large signals, highly redundant dictionaries and/or high precision (i.e., large number of iterations), the decomposition time can still be large. As reported above, computation times are in the order of one hour for the processing of one image in [4] and of typically 100 times the duration of the audio piece in [5].

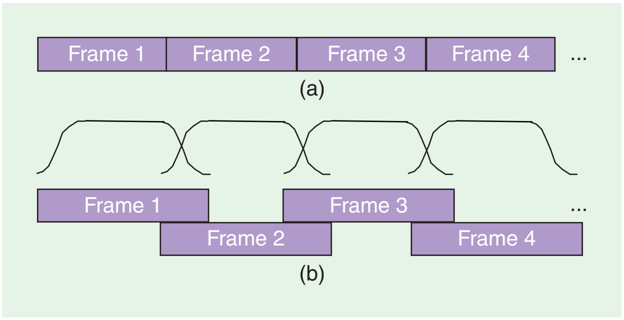
The goal of this research is to show how such principles can be generalized to "parallelize" MP or more generally, any greedy pursuit algorithm. The main idea is to break down the problem into a number of threads that could be handled by different processor cores working on different portions of the same signal. It should be emphasized that our approach is fundamentally different from the previously published work on parallel MP [31]–[33], that present efficient multicore implementations of MP by optimally distributing the computationally intensive steps of MP (atom selection stage, update of the scalar products), hence minimizing the message passing between subtasks. Instead, in the current work, we present a "suboptimal" version of MP that allows a straightforward parallelization of the MP algorithm: different threads work on different portions of the signal. The first benefit is that there is no message passing between tasks, which may be a limiting factor when scaled to a very large number of tasks, and may lead to a performance strongly dependent on the architecture. Our approach is fully scalable, in the way that is adapts to the processing power at hand (basically, dividing the load by the number of cores with a negligible master process), and does not require any platform-dependent implementation or parameter optimization. The second benefit is that there is no need to know, or load into memory, the entire signal. For a typical five-minute long song of CD-quality with more than 10^7 samples, there is no strong penalty in working locally on time frames whose size match the largest coherent "objects" in our signals (e.g., musical notes), with a typical duration of 0.1 s. This opens a perspective on "real-time" processing of an incoming stream of audio data—though with a delay that can become significant. The penalty to pay is that we are no longer guaranteed to make optimal choices at every iteration as in the plain MP. We shall see that, for typical signals, a smart algorithm design can not only reduce this "penalty," but even take advantage of it, at high number of iterations. Throughout this article, all the examples will be audio signals, as 1-D signals are the easiest way to present such algorithms, however, similar principles could potentially apply to other signals with similar scaling structures.

FIXED VERSUS SLIDING LOCAL MP

As a first approach, we divide the computational burden of MP by working locally on fixed adjacent segments ("frames") of the signal (see Figure 2). The simplest strategy is to use block-based frames: the signal is simply divided into equal-length frames [Figure 2(a)] and a local MP is applied on each of them. Although this strategy may be acceptable for some signals, for the audio signals studied here it provides sharp blocking effects: transients at edges, varying quality across edges.

An alternate strategy is to use overlapping frames with smooth windows [Figure 2(b)]. It is important to mention that the window is not applied on the signal but is a reweighting of the scalar products at the atom selection stage (step (**) in Algorithm 1), reducing the likeliness of choosing atoms at the side of the frames. This weighted matching pursuit (WMP) is described in Algorithm 2.

The weights w_γ are chosen according to the center times of the atoms: large weights for atoms centered around the middle of the frame, small weights on the side. More precisely, if t_γ is the centre time of the atom g_γ, then $w_\gamma = w(t_\gamma / L)$,



[FIG2] Parallel processing with (a) fixed frame-based segmentationß or (b) fixed smooth overlapping windows.

where L is the frame size and w any smooth tapering window defined on $[0, 1]$ such that $w(0) = w(1) = 0$, for instance a Hanning window (preliminary experiments indicate that the regularity of w is important but its exact design has little influence on the final performance). With this modification, the boundary effects are significantly reduced, but, as seen in Figure 2(b), adjacent overlapping windows share a portion of the signal. If an atom is selected in this zone, then a message has to be passed to its neighbor, indicating that the signal there has been updated locally. Due to the tapering window, these events should be relatively rare, but their existence impose stringent parallel programming constraints with message passing and synchronization.

GREEDY METHODS ARE BASED ON A SIMPLE DIVIDE AND CONQUER PRINCIPLE: THEY SELECT ONE ATOM, SUBTRACT ITS CONTRIBUTION, AND ITERATE ON THE RESIDUAL.

The simplest and more efficient strategy that we propose is to use instead sliding frames, such as the one displayed in Figure 3: a given core acts on a (windowed) frame of the signal, that after a number of iterations moves forward. An alternate view of this problem is shown on Figure 4, where multiple cores act on adjacent frames: after a number N_0 of iterations, the signal is shifted by h samples, with h a fraction of the frame size L. It is important to note that now, there is no overlap between adjacent frames, and therefore no need for message passing. This way, it is possible to use multiple core processors performing with the highest precision while guaranteeing real-time processing of the data, though with significant delay. If f_s is the signal sampling frequency, and $N_{\text{iter/core}}$ the guaranteed number of MP iterations/core/s (after initialization, the computational cost per iteration remains roughly constant), N_0 is simply given by $N_0 = (N_{\text{iter/core}} h)/(L_f f_s)$.

The pseudocode for this PLoMP algorithm is given in Algorithm 3. Apart from file input/output, PLoMP perfectly spreads the computational load onto the available computational resource, by making at each iteration K independent calls to the WMP.

RESULTS

We have simulated these approaches for representing audio signals on a redundant dictionary of local cosines. We first used a test signal of a sum of three constant-amplitude sinusoids with well-separated frequencies, a signal considered as very sparse, with added white noise. The energy of the residual is plotted on Figure 5, as a function of the total number of iterations. The following four different PLoMP configurations were tested and compared to the standard MP algorithm:

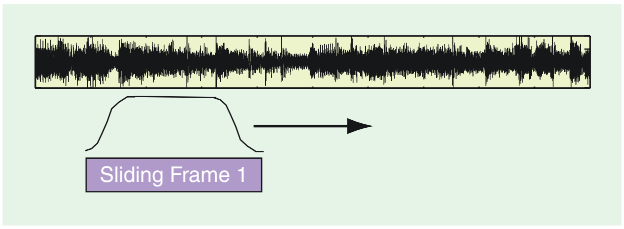
1) one-pass PLoMP (one core, working locally on the signal)
2) a two-pass PLoMP ($K = 2$ cores working locally on the signal)

[ALGORITHM 2] WEIGHTED MATCHING PURSUIT.

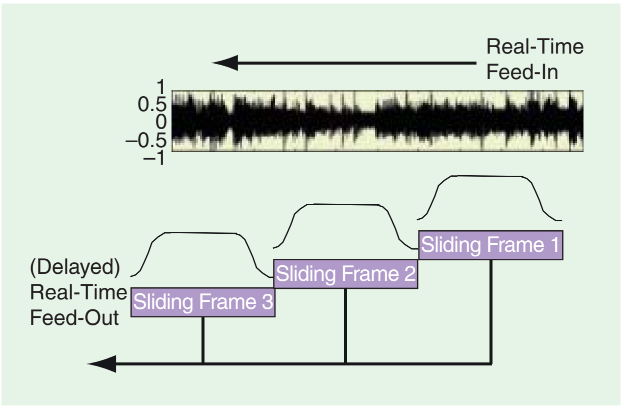
```
Require: signal $\mathbf{x} \in \mathbb{R}^N$, dictionary $\mathcal{D} = \{\mathbf{g}_\gamma, \gamma \in \Gamma\}$, maximum number of
    iterations $N_0$ set of weights $\{w_\gamma, \gamma \in \Gamma\}$.
Ensure: $\{\alpha_\gamma, \gamma \in \Gamma\}$ set of coefficients
  $n \leftarrow 0$ index of iteration
  $\mathbf{r}^0 \leftarrow x$ residual at initialization
  $\alpha_\gamma \leftarrow 0, \ \forall\gamma \in \Gamma$
  repeat
    $c_\gamma = \langle \mathbf{r}^n, \mathbf{g}_\gamma \rangle, \ \forall\gamma \in \Gamma$ scalar products computation
    $\gamma_{opt} \leftarrow \text{argmax}_{\gamma\in\Gamma} |w_\gamma c_\gamma|$ weighted atom selection stage
    $\mathbf{r}^{n+1} \leftarrow \mathbf{r}^n - c_{\gamma_{opt}} \mathbf{g}_{\gamma_{opt}}$ residual update
    $\alpha_{\gamma_{opt}} \leftarrow \alpha_{\gamma_{opt}} + c_{\gamma_{opt}}$
    $n \leftarrow n + 1$
  until $n = N_0$ iterations performed or required precision reached
  Return $\{\alpha_\gamma, \gamma \in \Gamma\}$ set of decomposition coefficients, such that
  $\mathbf{x} = \sum_{\gamma\in\Gamma} \alpha_\gamma \mathbf{g}_\gamma + \mathbf{r}^n$
```

[ALGORITHM 3] PARALLEL LOCAL MATCHING PURSUIT.

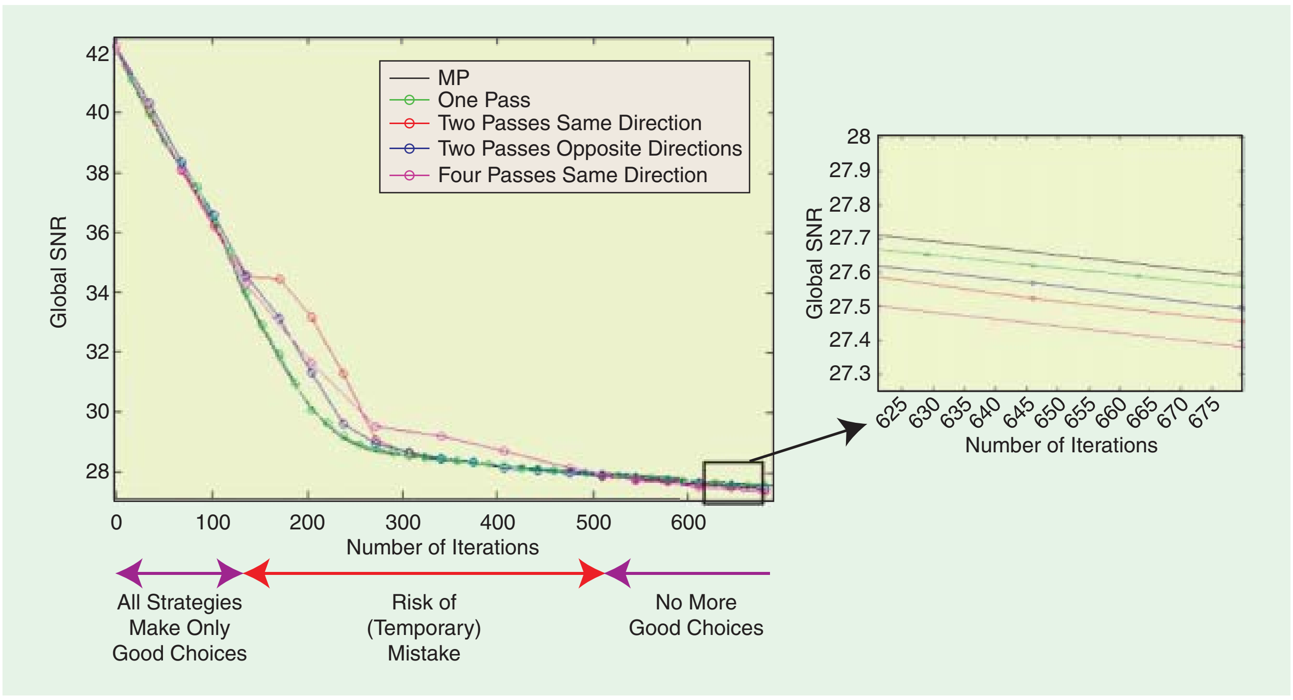
```
Require: incoming signal $\mathbf{x}$, frame length $L$, size-$L$ local dictionary
    $\mathcal{D} = \{\mathbf{g}_\gamma, \gamma \in \Gamma\}$, set of weights $\{w_\gamma, \gamma \in \Gamma\}$, frame hop size $h$,
    number of cores $K$, number of iterations per core $N_0$.
Ensure: $\{\alpha_\gamma, \gamma \in \Gamma\}$ set of coefficients
  $\mathbf{x}_{\text{local}} \leftarrow$ first $K * L$ coefficients of $\mathbf{x}$
  repeat
    equally divide $\mathbf{x}_{\text{local}}$ into $K$ nonoverlapping frames
    parallel process each frame with WMP, $N_0$ iterations
    store results in $\{\alpha_\gamma, \gamma \in \Gamma\}$
    load next $h$ samples of $\mathbf{x}$ and shift $\mathbf{x}_{\text{local}}$
  until no more incoming signal
  Return $\{\alpha_\gamma, \gamma \in \Gamma\}$ set of decomposition coefficients, such that
  $\mathbf{x} = \sum_{\gamma\in\Gamma} \alpha_\gamma \mathbf{g}_\gamma + \mathbf{r}^n$
```



[FIG3] One sliding frame with a smooth tapering window.



[FIG4] Parallel processing PLoMP with adjacent smooth windows. The incoming signal is fed through the successive local processors, each one making just as many operations as to guarantee real-time-processing.



[FIG5] Decay of the energy of the residual as a function of the number of iterations, for a signal being a sum of three constant sinusoids and the dictionary a union of local cosines with different scales. The plain black line is the reference global MP, other colors are local PLoMP with different strategies (different number of windows/number of iterations at a given position).

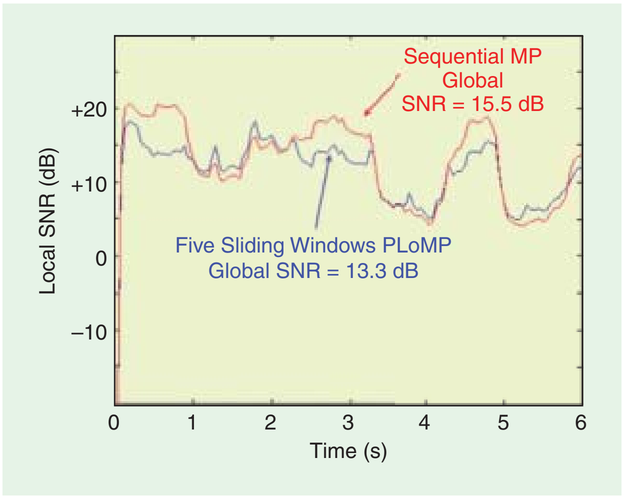
3) two one-pass PLoMP, while the second time the signal is entered in a time-reversed fashion, attempting to reduce the effect of time asymmetry in PLoMP

4) a four-pass PLoMP ($K = 4$ cores).

Interestingly, for a small number of iterations per core, all parallel or sequential strategies are equivalent, corresponding to "good" choices: selected atoms remove energy only from the sinusoids. For a range of subsequent iterations, parallel strategies sometimes make "mistakes," choosing atoms in the noise or side lobes. However, at high precision, all the sinusoidal components have been removed and the performance of all strategies are similar. Note that in this regime, the parallel local MP strategies even obtain a slightly better asymptotic behavior than global MP (see inset of Figure 5)! It should also be noted that the third strategy, using two passes in alternate directions, results in slightly better results in the intermediate regime than the $K = 2$ PLoMP (second strategy) (it can be guessed that some of the "bad choices" are a consequence of the time asymmetry of the local MP algorithm), but does not lead to any gain in the high-precision, asymptotic regime.

We then performed similar simulations on a real audio excerpt chosen for its large dynamics (jazz trio with loud piano notes/quiet double-bass + drums), and compared the standard MP to a $K = 5$ cores PLoMP, for a total number of iterations where the sound quality was deemed acceptable. For the same number of total iterations, the global signal-to-noise ratio (SNR) of the standard, sequential MP was 15.5 dB; while PLoMP resulted in 13.3 dB SNR.

However, looking at the global SNR may not be the only criteria to consider. Figure 6 shows the local SNR (computed on sliding windows of length 16,384 samples, i.e., about 370 ms at 44.1 sampling rate), for the same total number of iterations, between the global MP and PLoMP. Although the average SNR is higher in the case of the global MP, the situation can be the opposite locally: PLoMP has a steadier local SNR. This can be beneficial from a perceptual point of view, and this is confirmed by listening to the soundfiles: in PLoMP, the bass has significantly more presence, while it has almost disappeared in the



[FIG6] Parallel versus sequential processing on a real audio signal. Local SNR for sequential MP (red) and PLoMP (blue), for the same number of iterations.

global, sequential MP. Corresponding sound files can be downloaded at http://old.lam.jussieu.fr/src/Membres/Daudet/SPM/.

CONCLUSION

The widely used MP algorithm is intrinsically a sequential algorithm. We have shown that it can be modified to work locally, for carefully chosen frame duration and window shape, and is therefore particularly well suited to multicore processing. Simulations show that this comes at a usually small penalty in performance, if any. For signals with large dynamics, it may even provide a better adaption to the specificities of the signal. Although still at a preliminary stage, this study paves the way to what is so far considered as intrinsically impossible: an "online" sparse solver for live multimedia continuous data streams. For high-quality audio, this would typically require tens of cores!

ACKNOWLEDGMENTS

The author would like to thank Bob L. Sturm for proofreading and the anonymous reviewers for their insightful comments. Part of this work was done during an audio research residency at the Banff Centre of the Arts, Canada.

AUTHOR

Laurent Daudet (laurent.daudet@espci.fr) is a professor at Paris Diderot University-Paris 7, France. After a physics education at the Ecole Normale Supérieure in Paris, he received a Ph.D. degree in applied mathematics from the Université de Provence, Marseille, France, in 2000. In 2001 and 2002, he was a EUMarie Curie post-doctoral Fellow at the Centre for Digital Music at Queen Mary University of London, United Kingdom. From 2002 to 2009, he worked as an assistant/associate professor at UPMC-Paris 6 University in the Musical Acoustics Laboratory (LAM), now part of the D'Alembert Institute for Mechanical Engineering. Since September 2009, he has been a professor at the Paris Diderot University-Paris 7. He is also a visiting senior lecturer at Queen Mary University of London. He is an author or coauthor of over 80 publications on various aspects of audio digital signal processing, such as audio coding with sparse decompositions.

REFERENCES

[1] M. Adams, *The JPEG-2000 Still Image Compression Standard, ISO/IEC JTC 1/SC 29/WG 1*, vol. 2412, 2001.

[2] M. Bosi, K. Brandenburg, S. Quackenbush, L. Fielder, K. Akagiri, H. Fuchs, M. Dietz, J. Herre, G. Davidson, and Y. Oikawa, "ISO/IEC MPEG-2 advanced audio coding," *J. Audio Eng. Soc.*, vol. 45, no. 10, pp. 789–814, 1997.

[3] I. Selesnick, R. Baraniuk, and N. Kingsbury, "The dual-tree complex wavelet transform," *IEEE Signal Processing Mag.*, vol. 22, no. 6, pp. 123–151, 2005.

[4] L. Peotta, L. Granai, and P. Vandergheynst, "Image compression using an edge adapted redundant dictionary and wavelets," *EURASIP Signal Process. J.*, vol. 86, no. 3, pp. 444–456, 2006.

[5] E. Ravelli, G. Richard, and L. Daudet, "Union of MDCT bases for audio coding," *IEEE Trans. Audio Speech Lang. Proc.*, vol. 16, no. 8, pp. 1361–1372, 2008.

[6] R. Pichevar, H. Najaf-Zadeh, and L. Thibault, "A biologically-inspired low-bitrate universal audio coder," in *Proc. Audio Engineering Soc. 122th Convention*, Vienna, Austria, 2007.

[7] R. Neff and A. Zakhor, "Matching pursuit video coding. I. Dictionary approximation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 1, pp. 13–26, 2002.

[8] M. Plumbley, T. Blumensath, L. Daudet, R. Gribonval, and M. Davies, "Sparse representations in audio and music: From coding to source separation," *Proc. IEEE*, to be published.

[9] P. Leveau, E. Vincent, G. Richard, and L. Daudet, "Instrument-specific harmonic atoms for mid-level music representation," *IEEE Trans. Audio Speech Lang. Process.*, vol. 16, no. 1, p. 116, 2008.

[10] E. Ravelli, G. Richard, and L. Daudet, "Audio signal representations for indexing in the transform domain," *IEEE Trans. Audio Speech Lang. Process.*, to be published.

[11] D. Malioutov, M. Cetin, and A. Willsky, "A sparse signal reconstruction perspective for source localization with sensor arrays," *IEEE Trans. Signal Process.*, vol. 53, no. 8, pt. 2, pp. 3010–3022, 2005.

[12] R. Gribonval, "Sparse decomposition of stereo signals with matching pursuit and application to blind separation of more than two sources from a stereo mixture," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, 2002, vol. 3, pp. 3057–3060.

[13] A. Bronstein, M. Bronstein, M. Zibulevsky, and Y. Zeevi, "Sparse ICA for blind separation of transmitted and reflected images," *Int. J. Imaging Syst. Technol.*, vol. 15, no. 1, pp. 84–91, 2005.

[14] D. Gabor, "Acoustical quanta and the theory of hearing," *Nature*, vol. 159, no. 4044, pp. 591–594, 1947.

[15] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. Signal Process.*, vol. 54, no. 11, p. 4311, 2006.

[16] S. Chen, D. Donoho, and M. Saunders, "Atomic decomposition by basis pursuit," *SIAM Rev.*, vol. 43, no. 1, pp. 129–159, 2001.

[17] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *Ann. Stat.*, vol. 32, no. 2, pp. 407–451, 2004.

[18] E. Candes and T. Tao, "The Dantzig selector: Statistical estimation when p is much larger than n," *Ann. Stat.*, vol. 35, no. 6, pp. 2313–2351, 2007.

[19] E. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. Inform. Theory*, vol. 52, no. 2, pp. 489–509, 2006.

[20] D. Donoho, "Compressed sensing," *IEEE Trans. Inform. Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[21] A. Borghi, J. Darbon, S. Peyronnet, T. Chan, and S. Osher, "A simple compressive sensing algorithm for parallel many-core architectures," Univ. of California, Los Angeles, CA, Computational and Applied Mathematics Tech. Rep. (08–64), 2008.

[22] S. Lee and S. Wright, "Implementing algorithms for signal and image reconstruction on graphical processing units," Comp. Sci. Dept., Univ. Wisconsin-Madison, Tech. Rep., 2008.

[23] M. Andrecut. (2009). "Sparse approximation of computational time reversal imaging," [Online]. Available: http://arxiv.org/pdf/0904.3396

[24] J. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Trans. Inform. Theory*, vol. 50, no. 10, pp. 2231–2242, 2004.

[25] S. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Trans. Signal Process.*, vol. 41, no. 12, pp. 3397–3415, 1993.

[26] Y. Pati, R. Rezaiifar, and P. Krishnaprasad, "Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition," in *Proc. 27th Asilomar Conf. Signals, Systems and Computers*, 1993, pp. 40–44.

[27] B. Mailhé, R. Gribonval, F. Bimbot, and P. Vandergheynst, "A low complexity orthogonal matching pursuit for sparse signal approximation with shift-invariant dictionaries," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, 2009, pp. 3445–3448.

[28] A. Barron, A. Cohen, W. Dahmen, and R. DeVore, "Approximation and learning by greedy algorithms," *Ann. Stat.*, vol. 36, no. 1, pp. 64–94, 2008.

[29] T. Blumensath and M. Davies, "Gradient pursuits," *IEEE Trans. Signal Process.*, vol. 56, no. 6, pp. 2370–2382, 2008.

[30] S. Krstulovic and R. Gribonval, "MPTK: Matching pursuit made tractable," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, 2006, vol. 3, pp. 496–499.

[31] G. Dodero, V. Gianuzzi, M. Moscati, and M. Corvi, "A scalable parallel algorithm for matching pursuit decomposition," in *Proc. High Performance Computing and Networking Conf. (HPCN'98)*, 1998, pp. 458–466.

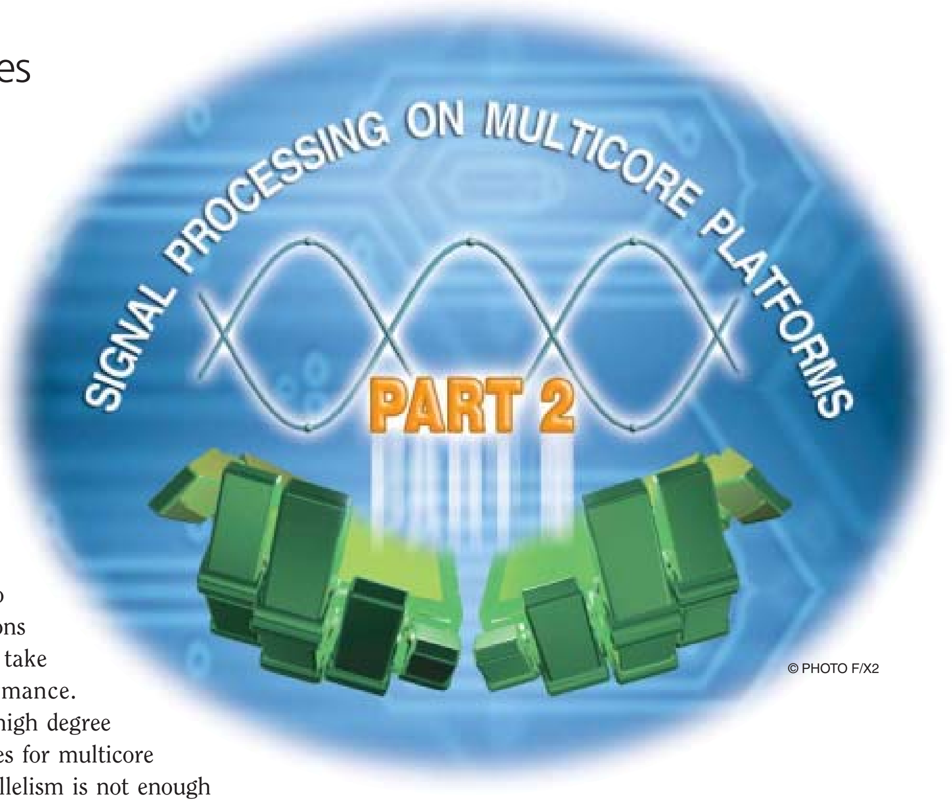
[32] A. Bultan and O. Arikan, "A parallelized matching pursuit algorithm for the four-parameter chirplet decomposition," in *Proc. Int. Symp. Time-Frequency Time-Scale Analysis*, Pittsburgh, PA, 1998, pp. 421–424.

[33] H. Feichtinger, A. Turk, and T. Strohmer, "Hierarchical parallel matching pursuit," *Proc. SPIE Int. Soc. Opt. Eng.*, vol. 2302, pp. 222–232, 1994. [SP]

[Daehyun Kim, Victor W. Lee, and Yen-Kuang Chen]

Image Processing on Multicore x86 Architectures

[Optimization techniques and examples]



© PHOTO F/X2

As multicore architectures overtake single-core architectures in today's and future compute systems, traditional applications with sequential algorithms can no longer rely on technology scaling to improve performance. Instead, applications must switch to parallel algorithms to take advantage of multicore system performance. Image processing applications exhibit a high degree of parallelism and are excellent candidates for multicore systems. However, simply exploiting parallelism is not enough to achieve the best performance. Optimization must take into account underlying architecture characteristics such as wide vector and limited bandwidth. This article illustrates techniques that can be used to optimize performance for multicore x86 systems on three key image processing kernels: fast Fourier transform, convolution, and histogram.

INTRODUCTION

Two major factors that affect application performance are the performance of the system that the application runs on and the underlying algorithms used in the application. For many years, applications could simply rely on system performance improvements from advances in semiconductor manufacturing and single-thread architecture. Recently, the power and the thermal wall began to limit further improvement. Industry switched to the more energy efficient design of multicore architectures. Today, nearly all major microprocessor vendors offer multicore processors. Instead of scaling performance with increased frequency, multicore processors offer higher performance via more processing cores. Multicore processors differ from traditional processors in many ways, such as higher core counts, simpler core architecture, and more elaborate on-chip interconnect. The increase in core count and compute density is the most notable difference between multicore architectures and traditional single-thread architectures. A critical implication of

Digital Object Identifier 10.1109/MSP.2009.935384

this change to applications is that they must be parallelized to take advantage of multicore architectures.

To fully utilize the computational capability of multicore platforms, it is very important to exploit the parallelism in applications. There is significant work on application parallelization. Lin [16] summarizes some general principles underlying parallel computation and clarifies why they represent opportunities or barriers to successful parallel programming. Breshears [6] gives eight simple rules on how to create scalable multithreaded applications. For example, one rule is choosing an alternative algorithm for a better chance of concurrency. The goal is to remind us that the best sequential algorithm may not be the best algorithm on multicore processors. While the best serial algorithm may have the theoretically lowest complexity, it may not be amenable to be parallelized. A suboptimal serial algorithm that is easier to parallelize may offer better performance for multicore architectures. While Breshears's rules demonstrate a high-level picture of parallel programming, we show practical programming details to help guide people to efficiently parallelize signal processing applications on multicore systems.

Furthermore, simply exploiting parallelism is not enough to provide high performance. Architecture-specific algorithm optimization may be required. Ryoo [21] presents the optimization principles on a GPU with Compute Unified Device Architecture (CUDA). The key idea is to use massive multithreading to utilize the large number of cores and hide global memory latency through striking a balance between each thread's resource usage and the number of simultaneously active threads. Some of their suggestions are also useful for general-purpose multicore processors, e.g., optimizing use of on-chip memory to reduce bandwidth usage and redundant execution. However, most of their suggestions are constrained to the target GPU platform. For instance, GPU can leverage its zero-overhead thread scheduling to hide memory latency, but not all processors have built-in hardware thread scheduling. Thus, the optimization principles on one multicore architecture cannot be directly applied to other multicore architectures.

Auto-tuning frameworks are proposed to allow applications to adapt to platform changes. There are many pieces of work about automatic tuning of fast Fourier transform (FFT) on multicore platforms [2], [11], [12], [20]. Automatic tools are particularly good at exploring the design space when we do not have enough knowledge of the platforms. Auto tuners fail when drastic platform changes occur, such as the recent switch from the single monolithic complex core design to the many simpler cores design. To achieve the best performance on a new multicore architecture, it is still critical to hand optimize the code. Similarly, for an automatic tool to consider all new architecture features, the work is similar to hand optimizing the code.

[A SET OF OPTIMIZATION TECHNIQUES IS PRESENTED TO HELP GUIDE PROGRAMMERS TO WORK WITH INTEL'S MULTICORE x86 SYSTEMS.]

This article discusses the necessity of architecture-aware optimizations. A set of optimization techniques is presented to help guide programmers to work with Intel's multicore x86 systems. Three detailed examples of optimizing FFT, convolution, and histogram are used to illustrate the principles behind the techniques.

FFT, convolution, and histogram are three important kernels that form the building blocks of many image processing applications. Optimizing these kernels will provide performance benefit to all applications that utilize them. A number of open-source or commercial packages exist to provide preoptimized versions of these kernels for specific platforms [11][12][14]. However, as the platforms evolve, existing optimizations may not apply any more and new optimization packages are required.

The contributions of this article include:

1) providing a set of optimization techniques to guide programmers in architecture-aware optimization

2) providing detailed examples illustrating the use of the techniques on a multicore x86 architecture.

OPTIMIZATION TECHNIQUES

Multicore processors provide performance through parallel computation instead of running single-thread fast. As more processors are being integrated on to the same die, there are a few architecture implications. First, the memory bandwidth per core is reduced and that potentially makes some workloads bandwidth bound. Second, due to power limitation, the more processors are integrated, the less power each processor can consume. One consequence is that future multicore processor may employ a much simpler core, or more energy efficiency architecture features, such as, wider SIMD [24].

We introduce a set of techniques to optimize the three image processing kernels on multicore x86 systems, which is summarized in Table 1. Important optimization topics are identified so that programmers can use it as guidance for their optimization work.

The first step is to identify whether a program is compute-bound or memory-bound. This helps set the optimization goal. By understanding the program's compute and memory requirements and matching it to the capability of a given multicore system, we can project the potential performance. The second step is to optimize single-core efficiency. Since a core's computation capability largely stems from its wide vector

[TABLE 1] OPTIMIZATION TECHNIQUES.

STEP	DESCRIPTION
1) WORKLOAD CHARACTERISTICS	— IDENTIFYING COMPUTE OR MEMORY BOUND
2) SINGLE-CORE OPTIMIZATION	— VECTORIZING
3) MULTICORE OPTIMIZATION	— DATA PARTITIONING — LOAD BALANCING
4) MEMORY OPTIMIZATION	— CACHE BLOCKING — DOUBLE BUFFERING — DATA PREFETCHING

units, vectorization is the key for good single-core performance. The third step is to achieve high multicore scalability. Good parallelization should include intelligent data partitioning that minimizes intercore communication as well as balanced load distribution among cores. The fourth and last step is memory optimization. Memory optimization should focus more on saving bandwidth than hiding latency. One of the biggest architectural changes in multicore chips from traditional parallel architectures is in the memory hierarchy; multicore chips provide fast communication among on-die cores, but all the cores share the same off-chip bandwidth. We observe that the off-chip bandwidth is the most precious resource in multicore systems. We use various techniques like cache blocking, double buffering, and data prefetching to use the memory bandwidth efficiently.

FAST FOURIER TRANSFORM

FFT is an algorithm to implement discrete Fourier transform. It improves performance from O(N^2) to O(NlogN). FFTs have been studied exhaustively. Good algorithmic overviews are provided in [9], [15], and [17] and some representative implementations are published in forms of general libraries [12], [14], [19] or architecture specializations [7], [13], [18], [23]. The Cooley-Tukey algorithm [8] provides a theoretical basis, and it is the most common among various FFT algorithms. Our FFT optimization starts with a baseline Cooley-Tukey algorithm. The same optimization techniques can also be applied to other complicated implementations. The main scope of this article is performance optimization rather than algorithmic discussion.

The Cooley-Tukey FFT algorithm consists of logN stages of butterfly operation followed by a bit-reverse permutation. Arithmetic computations are simple floating-point multiply-add, but data access patterns are nontrivial, which leads to optimization issues. Figure 1 illustrates our optimization strategy. It implements a 16-point radix-2 algorithm on a two-core system with a two-wide vector. We use this simplified example for brevity. Later we will address key optimization concepts such as higher radix algorithm, larger data size, wider vector, and more cores.

SINGLE-CORE OPTIMIZATION: VECTOR DECIMATION

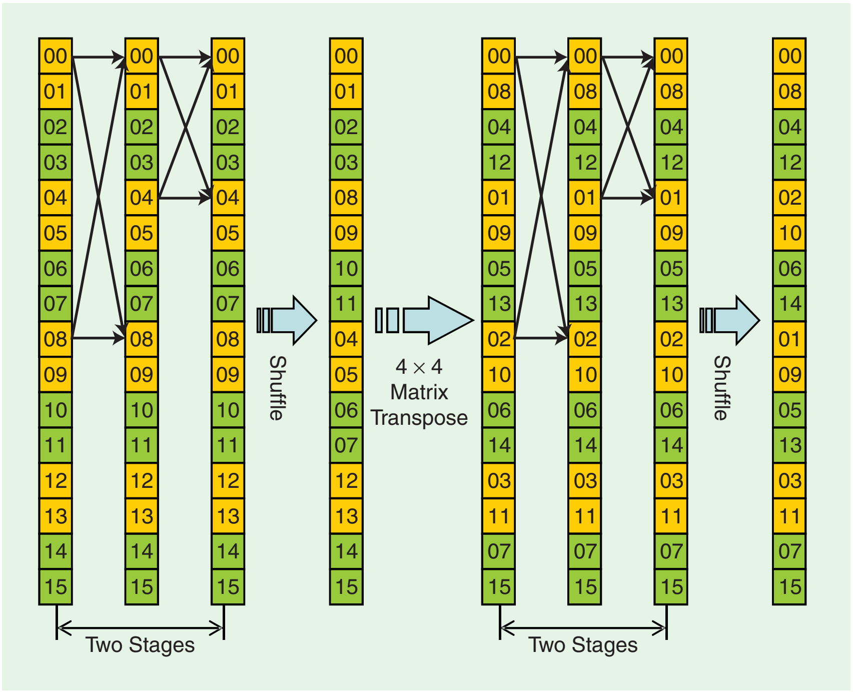
A simple vectorization scheme is to group consecutive data elements to one vector. However, it becomes problematic when the butterfly stride becomes smaller than the vector width (two in our example). When the butterfly stride is greater than the vector width, computation is performed with full vector width and the vector efficiency is 100%. As the butterfly stride becomes shorter than the vector width, vector efficiency decreases and ultimately only one element can be processed.

Various vectorized FFT algorithms such as [3] and [10] have been studied. The main idea is summarized here for reference. The solution is to perform a matrix transpose before the butterfly stride becomes shorter than the vector width. The transpose reorders the memory layout so that the butterfly stride can always be greater than the vector width.

For the 16-point radix-2 example, the initial butterfly stride is eight elements, so we can fully utilize the vector width of two. However, after two stages of the butterfly operations, the stride distance is one and we cannot perform vector operation. Using the vectorized FFT algorithm (as shown in Figure 1), if a 4 × 4 matrix transpose is performed after the second stage, the full vector execution can be performed throughout the stages.

MULTICORE OPTIMIZATION: INTERCORE COMMUNICATION

The right parallelization strategy depends on application scenarios. We classify the usage model based on two workload characteristics: the size of data and the number of FFTs to be performed. First, if the data size is too small, parallelizing it to multiple cores will be fruitless because the communication overhead will outweigh the parallelization benefit. Therefore, multicore parallelization is only considered for the big size FFTs. Second, if the target application performs many independent FFTs, parallelization can be performed at the individual FFT level. Since no communication is required among the



[FIG1] FFT optimization example: a 16-point radix-2 algorithm on a two-core system with a two-wide vector (elements are color-coded: different colors are assigned to different cores, and two consecutive elements are grouped into one vector).

individual FFTs, each can be assigned to different cores and executed independently.

Parallelization of large size FFT has been studied well in such as the six-step algorithm [4]. The idea is to partition data intelligently to minimize data communication. One naïve scheme is to partition data contiguously. To parallelize a size-N FFT on C cores, the first N/C elements are assigned to Core 1, the next N/C to Core 2, and so on. However, this scheme requires a lot of intercore communication because data elements participating to the same butterfly computation are distributed across cores. A more intelligent scheme will take advantage of the butterfly access pattern. The first butterfly stride is N/2, so the first and N/2th elements are allocated to the same core. Because the second butterfly stride is N/4, the N/4th element is also allocated at the same core, and so on. We continue until all the data are assigned to cores evenly.

Figure 1 shows an example of the proposed scheme. For two cores, yellow boxed elements are assigned to Core 1 and green boxed elements are assigned to Core 2. We observe no intercore communication during the first two butterfly stages and also the last two. Intercore communication only occurs between the second and the third stages during the matrix transpose, which is unavoidable to parallelize the codes on two cores.

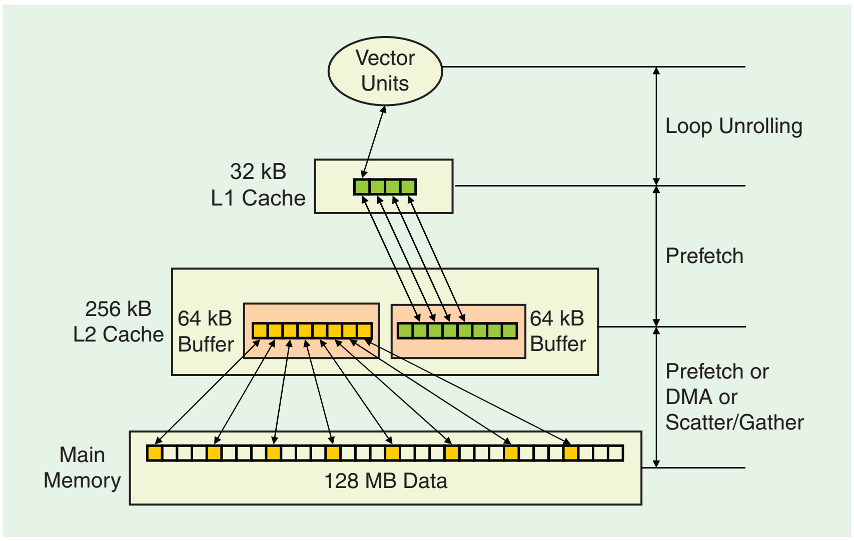
MEMORY MANAGEMENT: BANDWIDTH EFFICIENCY

Due to the stride access pattern, FFT implementations are memory-bound in many cases. This makes the large compute capability provided by the multicore systems useless unless an efficient memory management scheme is used. We achieve high off-chip bandwidth efficiency by combining multiple memory operations into one. The memory operations involved in FFT are the matrix transpose for vectorization, the intercore communication due to parallel processing, and the bit-reverse permutation. If we perform each memory operation separately, it requires multiple memory accesses. Worse yet, if the data size does not fit to the cache, it results in multiple trips to main memory and this is a poor way of utilizing limited off-chip memory bandwidth. In the worst case, a naïve implementation requires an entire data sweep per butterfly stage.

Our optimization combines multiple memory operations into one to increase off-chip bandwidth efficiency. In Figure 1, at the first butterfly stage, data blocks (yellow elements for Core 1 and green elements for Core 2) are loaded from the memory to the scratchpad (caches/buffers/registers). Then, the first two stages are executed without intercore communication or memory accesses. After finishing the two stages, each core writes the results to the main memory. At the same time, it also performs the bit-reverse shuffling and matrix transposing together. The remaining two stages are executed in the same manner. Note that bit-reverse shuffle, matrix transpose, and intercore communication are combined into one memory stage, which saves off-chip memory bandwidth significantly. Otherwise, we should sweep the entire data through the main memory whenever we perform such operations.

Another important memory optimization is multiple buffering. It allows overlapping computation and memory transfer. While arithmetic units perform computations on one buffer, memory units prepare data for another buffer. Both units work concurrently instead of one unit waiting for the other.

Figure 2 shows an example of double buffering. It performs an FFT on 128 MB data, which can be a big three-dimensional (3-D) FFT or hundreds of small one-dimensional (1-D) FFTs. Using the vectorization and parallelization techniques discussed earlier, we can divide the whole FFT into multiple blocks. The block size should be chosen based on the cache size. For a cache size of 256 KB, we choose the block size of 64 KB for a double buffering implementation. Two buffers will occupy 128 KB and the remaining space will be used by constant values like twiddle factors. Note that 32 KB L1 cache is too small to maintain the double buffering. While the vector units are executing a FFT on one buffer, the data for the next FFT block are fetched to the second buffer. Once the operations are done, the two buffers are switched. To achieve a good computation and memory efficiency, the block computation time and the block communication time should be matched, which is another factor to decide the block size.



[FIG2] **FFT double buffering scheme.**

HIGHER RADIX ALGORITHM

For brevity, we discussed a radix-2 algorithm so far. But in practice, higher radix algorithms usually provide better performance. Table 2 illustrates computation requirements to perform a 4,096-point complex FFT with a radix-2, -4, and -8 implementations. Higher radix algorithms save arithmetic and memory operations significantly. However, we also need to consider architectural features. First, higher radix algorithms require more internal storage. For example, in our x86 implementations, a radix-2 implementation uses four variables, while a radix-4 implementation needs 14 variables, and a radix-8 implementation needs 20 variables. Since only 16 SIMD registers are available for x86 SSE, a radix-8 implementation requires register spilling and filling, which will degrade performance. Second, higher radix algorithms can be used only for specific data sizes. For example, a radix-8 algorithm is used for power-of-eight FFT. So, if we perform a 1,024-point FFT, we should mix it with lower radix stages (two radix-4 stages and two radix-8 stages). Generally, the best-known-method is to use mixed radix algorithms that are composed of as high radix algorithms as possible for the microprocessor architecture and lower radix algorithms to match the given data size. Automatic code generation tools such as [12] and [19] can be very helpful because they can search the design space automatically.

MULTIDIMENSIONAL FFT

A multidimensional FFT is a collection of 1-D FFTs. It performs multiple 1-D FFTs with respect to each dimension. However, it creates long-stride memory accesses. Because data is stored in memory contiguously along with one dimension, memory accesses in other dimensions always result in long stride. For example, in Figure 3, row-wise FFTs access data contiguously, while column-wise FFTs access data in 2K stride. Long-stride memory accesses cause two problems. 1) Vectorizing is difficult because data is not located contiguously, and 2) cache behavior suffers from conflicts especially if the stride is power-of-two.

One solution is to perform a matrix transpose between dimension switch. Once row-wise FFTs are done, we transpose the matrix to switch rows and columns, then perform another row-wise FFTs. Another technique is to execute column-wise FFTs of the vector width, shown in Figure 3. After finishing row-wise FFTs, it performs column-wise FFTs directly without the transpose. Instead, it computes multiple columns of the vector width (W) simultaneously.

The first approach incurs matrix transpose penalty. But once the matrix is transposed, its performance is always optimized as the row-wise FFT. The second approach allows vectorization easily. But it suffers from column-wise memory accesses, resulting in much slower column-wise FFTs. The tradeoff between the matrix transpose and the column-wise accesses is data and machine dependent. It is related to data size and data dimension, as well as cache size and memory latency. In practice, we found that the optimal design point changes from one data set to another. To choose the right scheme for a given multidimensional FFT, we should profile each implementation and compare its actual performance. Automatic tools can help decide the right choice.

MATRIX TRANSPOSE

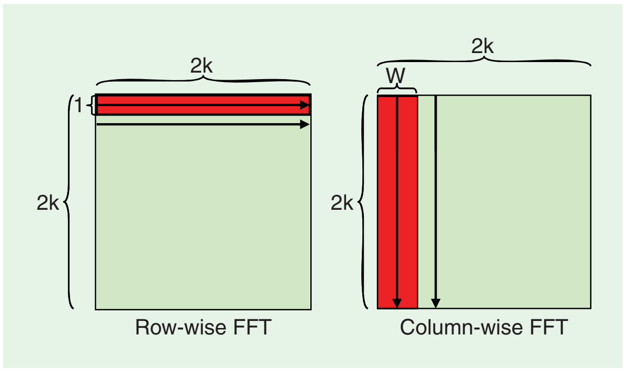
Matrix transpose is a crucial memory operation in FFT to enable vectorization and parallelization. We discuss a cache efficient parallel transpose algorithm. In Figure 4, we divide the entire matrix into smaller blocks denoted by A1, A2, and so on. Each block performs a matrix transpose A1 to A1′, A2 to A2′, etc. Assume Core 1 and Core 2 share the cache. Core 1 is now transposing A1 to A1′. If we assign B1 to Core 2, it will suffer from cache conflicts in writing to B1′ because Core 1 is writing to A1′. Remember that column-wise accesses cause cache conflicts. If we assign C1 to Core 2, it will also exhibit cache conflicts because Core 1 is reading from A1. To avoid cache conflicts, Core 2 should be working on A2. Once the cores are done with A1, and A2, they will move to B1 and B2, which are also conflict free. Therefore, the right policy is to assign blocks to cores with staggering.

[TABLE 2] COMPUTATION REQUIREMENTS FOR A 4K SINGLE-PRECISION COMPLEX FFT.

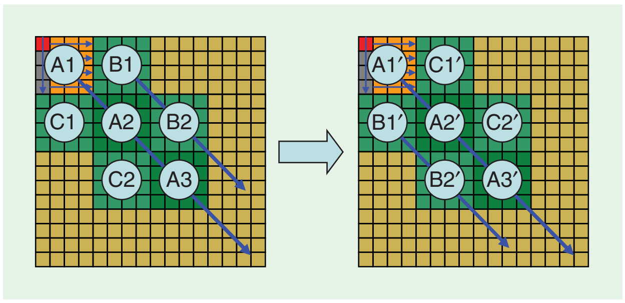
	LOADS/STORES	ADDS/SUBS	MULTIPLIES
RADIX-2	196,608	147,456	98,304
RADIX-4	98,304	135,168	73,728
RADIX-8	65,536	102,400	32,768

CONVOLUTION

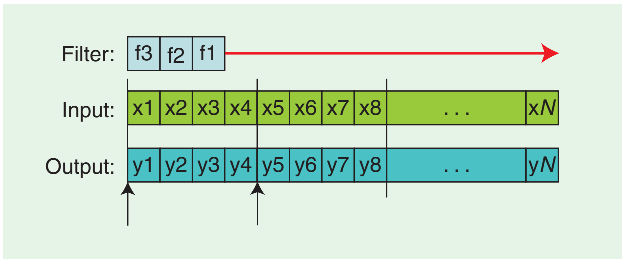
Convolution is used for many image filters such as blur, emboss, and sharpen. Figure 5 illustrates an example of a convolution. It overlays an n-tap filter on a set of n input pixels, multiplies the corresponding values, and accumulates the results into an output pixel. In most cases (except small-size 1-D filters) its compute-to-memory ratio is high and exhibits very regular operations. The arithmetic operations are multiply-add, and the data access patterns are streaming. However, because the convolution filter slides from the first pixel to the last, it has a memory alignment issue.



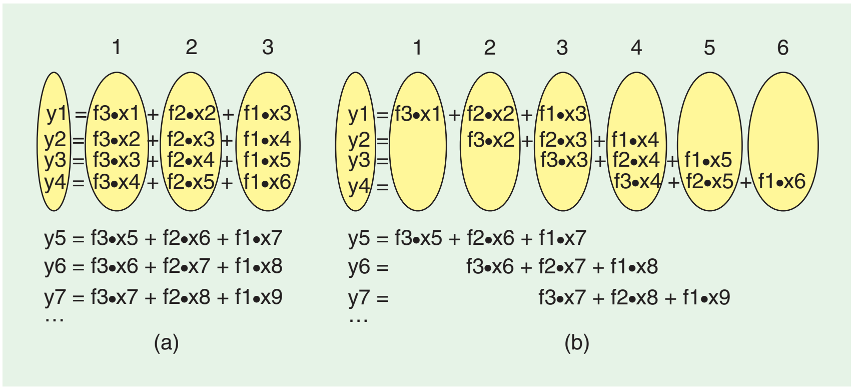
[FIG3] Column-wise FFT for a 2K × 2K 2-D FFT.



[FIG4] Cache conflict-free matrix transpose.


Filter:
f3 f2 f1
Input:
x1 x2 x3 x4 x5 x6 x7 x8 . . . xN
Output:
y1 y2 y3 y4 y5 y6 y7 y8 . . . yN

[FIG5] Memory alignment in a three-tap 1-D convolution (arrows represents memory alignment points for four-wide vector).


1 2 3
y1 = f3•x1 + f2•x2 + f1•x3
y2 = f3•x2 + f2•x3 + f1•x4
y3 = f3•x3 + f2•x4 + f1•x5
y4 = f3•x4 + f2•x5 + f1•x6
y5 = f3•x5 + f2•x6 + f1•x7
y6 = f3•x6 + f2•x7 + f1•x8
y7 = f3•x7 + f2•x8 + f1•x9
...
(a)
1 2 3 4 5 6
y1 = f3•x1 + f2•x2 + f1•x3
y2 = f3•x2 + f2•x3 + f1•x4
y3 = f3•x3 + f2•x4 + f1•x5
y4 = f3•x4 + f2•x5 + f1•x6
y5 = f3•x5 + f2•x6 + f1•x7
y6 = f3•x6 + f2•x7 + f1•x8
y7 = f3•x7 + f2•x8 + f1•x9
...
(b)

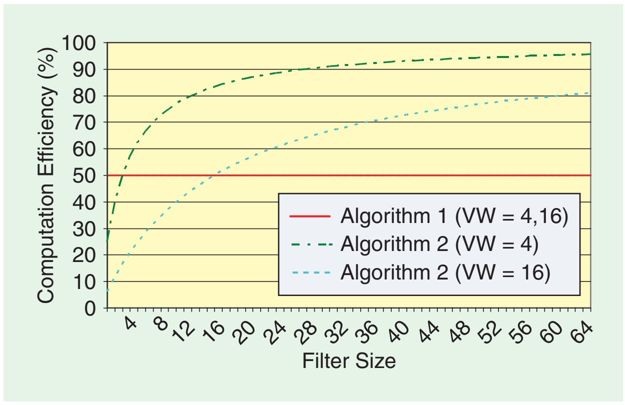
[FIG6] Vectorizing algorithms for convolution (a circled computation is performed by one vector operation): (a) Algorithm 1 and (b) Algorithm 2.

SINGLE-CORE OPTIMIZATION: MEMORY ALIGNMENT

Memory alignment is the biggest hurdle in vectorizing convolution. In many vector architectures like x86 SSE, data should be aligned in memory to take advantage of vector instructions. For example, in Figure 5 the input and output are located in memory so that their first elements are aligned. We can access a four-wide vector for [x1, x2, x3, x4] fast, but accesses to a vector [x2, x3, x4, x5] will pay unalignment penalty. The convolution filter slides over the input one-by-one. So, it might access an aligned data at some point, but the next access will be unaligned. To calculate output y1, we overlay the filter on input [x1, x2, x3] that is aligned. But, for output y2, we need to read unaligned [x2, x3, x4].

We discuss two vectorization algorithms to address the memory alignment. The first one is shown in Figure 6(a). To calculate an output vector [y1, y2, y3, y4], it performs three vector multiplications and sums them up: [f3, f3, f3, f3] * [x1, x2, x3, x4], [f2, f2, f2, f2] * [x2, x3, x4, x5], and [f1, f1, f1, f1] * [x3, x4, x5, x6]. It contains one aligned and two unaligned reads. Writes to the output is always aligned. The filter coefficients are also aligned because they can be preallocated into the right format. Figure 6(b) shows the second algorithm. It executes six vector operations to calculate the same output. It requires more steps because it does not utilize the vector fully. For example, the first calculation performs only one multiplication (f3 * x1) out of four-wide vector wasting the remaining three computing capability. Though the first algorithm may look better at glance, the second one is better in practice. The unaligned memory overhead causes more performance degradation than the vector inefficiency.

Figure 7 shows the comparison of the computation efficiency of the two algorithms. The filter size varies from one to 64 for the vector width four and 16, assuming the unaligned vector loads are on average 2x more expensive than aligned ones. The first algorithm (Algorithm 1) remains at low efficiency about 50% throughout the range, while the second algorithm (Algorithm 2) achieves much higher efficiency. Especially, as the filter size gets larger, its unused vector capacity decreases, thus its efficiency improves approaching 100%.



[FIG7] Efficiency comparison between the two vectorized convolution algorithms for various filter size and vector width.

MULTICORE EFFICIENCY: THREADING OVERHEAD

Parallelizing convolution to multiple cores is easy because arithmetic computations and memory operations are very regular. We can distribute the data evenly across the cores. Data sharing between cores occurs at partition boundaries. It is a read sharing, not a true data communication, thus resulting in very small performance loss.

Overall, we achieve a good scalability on a Core i7 system as shown in Figure 8. However, if the data size is too small, the multicore scalability starts saturating as the 128 × 128 image on four cores. Thread management overhead, such as thread creation and termination, increases as the number of core increases. Even though the main computation scales linearly, the overall performance is saturated if the parallel task is too small for the threading overhead.

MEMORY MANAGEMENT: CACHE BLOCKING AND DATA PREFETCHING

The basic memory management strategy is to stream the input and output only once and maintain all the other data in the cache or register. Convolution filters are usually smaller than the cache and in many cases smaller than the register file. However inputs and outputs are huge, so we should stream them from the memory. To compute an output, we need to read multiple inputs. There are data reuses, which we can take advantage of. Once the data is read from memory, we should keep them in the cache as long as they are required.

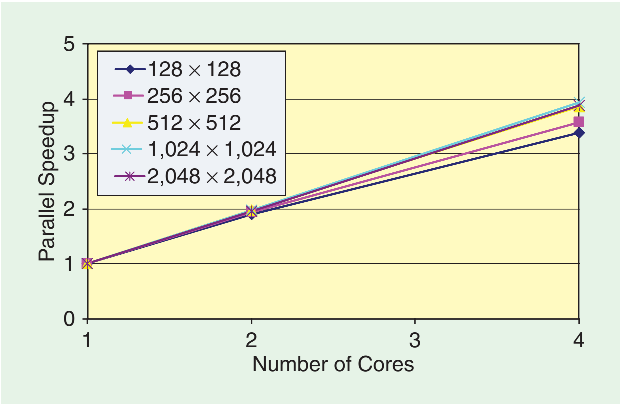
Figure 9 shows a 3 × 3 convolution. To calculate an output, it reads the surrounding nine inputs. Scheme 1 is cache ignorant. It computes an output and moves on to the next until it reaches the end of the row. Then, it returns back to the next row. Though consecutive two rows share some of the inputs, Scheme 1 cannot exploit it. When calculating the first output of a row, it reads a 3 × 3 input block to the cache. But, when it reaches at the end of the row, the first 3 × 3 inputs will be probably kicked out (especially if the image size is big). When it calculates the next row, it should reload all the 3 × 3 inputs again from the memory, though 2 × 3 inputs of which have been already read before. Scheme 2 solves the problem by cache blocking. Instead of moving to the end of a row, it stops at some point and moves down to the next row. It can reuse the inputs from the previous row before they are replaced from the cache.

The table in Figure 9 compares the two schemes in our Core i7 implementation. It shows the speedup of the cache blocking over the cache ignorance. For a small data (1,024 × 1,024 image), they are almost identical. However, as the data size gets bigger (2,048 × 2,048 image), the cache blocking is about 1.14x faster than the cache ignorance.

Data prefetching is another important memory optimization. Modern microprocessors are equipped with sophisticated hardware prefetchers. However, multicore systems tend to adopt simpler cores that cannot afford such complex hardware prefetchers. Therefore, software prefetching can improve performance significantly. In addition, software prefetching is even more important if data structures are complex (to handle multidimensional matrix) and memory patterns are nonlinear (to implement cache blocking). Though software prefetching is difficult in general, because memory access patterns in convolution are predetermined, programmers can insert software prefetches easily.

HISTOGRAM

Histogram is a statistical method to calculate the frequency of events. It is commonly used as a preprocessing step for other image filters. Its main operation is a table update. For each input value, it calculates the table index and updates the table entry. The most common index function and update operator are a range calculation and frequency increment. In practice, various index functions and update operators are used. The input access pattern is a simple read streaming. However, a table update causes a data-dependent indirect scattered writes, which makes vectorization and parallelization difficult. We cover the best known methods: serialization and privatization. We also look into a large histogram case that traditional solutions cannot handle well.



[FIG8] Multicore scalability of a 5 × 5 convolution on Core i7 four cores.

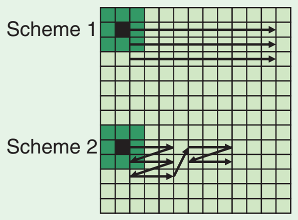
SINGLE-CORE OPTIMIZATION: ELEMENT ALIAS AND SCATTERED INDEX

Vectorizing histogram is difficult due to the two problems: element alias and scattered index. Element alias means that two or more elements within a vector point to the same table entry. We should resolve the index conflict as the original nonvector codes intend to. For example, if two elements point to the same table entry and the table update operator is addition, the entry should be incremented by two. However, conventional vector add instructions do not provide ordering within a vector. Instead of adding twice (one-after-another), it will update the entry only once. The second problem is scattered index. Because the table entry indices are calculated from the input, their values are arbitrary. As a result, elements within a vector can point to noncontiguous table entries. This causes performance loss, because normal vector operations execute only on contiguous memory locations.

A simple solution for element alias is serialization, i.e., elements are serially processed one by one. In other words, no vectorization is done and we lose the benefit of vector computation. A more sophisticated method is to detect index conflict. The indices within one vector are compared with each other to identify conflicts. For example, if two elements of value "1" conflict with the same index, we combine them into one element of value "2."

Scattered index problems can be also solved by serialization. Because it uses only scalar instructions, a scattered index is not a problem. However, some systems support gather/scatter

Image Size	Speedup (Scheme 2 Over Scheme 1)
1,024 × 1,024	0.96×
2,048 × 2,048	1.14×



[FIG9] Cache blocking of a 3 × 3 convolution and its performance impact on Core i7 one core.

operations, which handle noncontiguous memory accesses for vector. Gather/scatter instructions grab scattered data into a contiguous memory location. Instead of normal vector loads/stores we can use gathers/scatters, which will solve the scattered index problem.

We showed two vectorization approaches: serialization and conflict detection + gather/scatter. Note that conflict detection and gather/scatter should be used together. Neither one of them can provide vectorization alone. Basically, serialization is not using vectors at all, thus the latter vectorization technique will provide better performance. However, it is not always true in reality. Conflict detection and gather/scatter instructions are not free, and actually very expensive in today's platforms. We observe that their overheads outweigh their benefits; thus serialization is faster than vectorization for today's systems. However, future multicore architectures will address the performance of those operations, and then we will be able to use the vectorization technique discussed here.

MULTICORE OPTIMIZATION: ATOMIC REDUCTION AND LARGE TABLE ENTRY

When parallelizing histogram, we should solve the parallel reduction problem. If two or more cores try to update the same table entry, each update should be atomic. In Figure 10(b), two cores update the same entry whose current value is two. If their operations are additions, the correct result will be four. However, if atomicity is not guaranteed, the result can be three. When one core reads the current value, another core may also read the same value. If both of them write the data, one increment operation is lost. For correctness, we should provide atomicity during the "read-modify-write" operation.

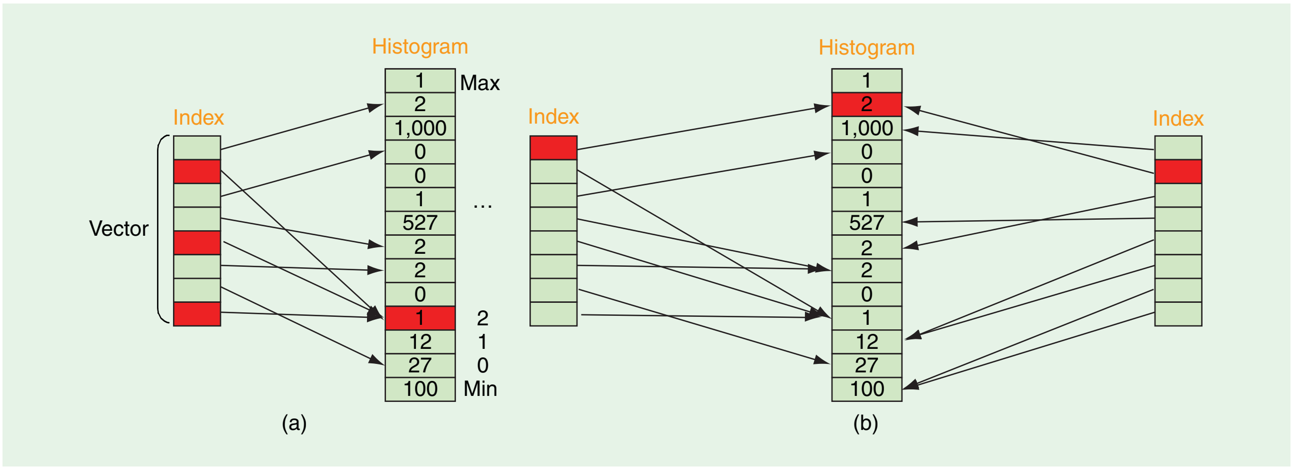
Traditionally there are two solutions: locking and privatizing. Locking is to reserve a memory location. A core locks a table entry, updates it, and unlocks it. While the table entry is locked, other cores cannot access it. Many multicore systems provide hardware support for locking only for simple operations like addition and subtraction. If the table update operators are complicated, we may need to implement software locking. Another solution is privatizing. Instead of updating the global table simultaneously, each core maintains its "private" copy of histogram. Because the private table is accessed by the core exclusively, atomicity is guaranteed. However, this solution requires merging all private copies back to the global table at the end.

The two solutions have pros and cons. Locking is expensive even with hardware support. Privatizing requires global reduction. Worse yet, both solutions do not scale well with increasing number of cores. In today's systems, the cost of locking overhead is usually higher than privatizing overhead, so privatizing is widely adopted as a general solution. However, future systems will provide better hardware primitives for locking, and locking will therefore become an alternate solution.

Parallelizing a large histogram is a difficult problem. It exhibits very different behavior from small histograms, thus it requires special optimization. Both the locking and privatizing are not effective. Especially, in the privatizing, the global reduction overhead is proportional to the table size and it gets worse as we increase the number of cores. (Each private table is the same size as the global histogram and one private table per each core is needed.)

Figure 11 compares the locking and privatizing for a large histogram. For small number of cores, the privatizing is better because the locking suffers from its fixed synchronization overhead. As the number of cores increases, the locking becomes better because the privatizing is hit by the large table size. In a Core i7 system, we scale only up to four cores, thus we do not see such a situation. However, where we scale to many cores, we might see a cross-over point that the locking becomes better than the privatizing. In any case, overall scalability of both techniques is poor.

We propose a hybrid algorithm that outperforms the locking and privatizing. Its performance is also shown in Figure 11. Our technique combines both techniques and, in addition, exploits the cache. It is based on the observation that nonzero entries are sparse though the table may be large. If we privatize only the nonzeros, we can eliminate a lot of overhead.



[FIG10] **Challenges in vectorizing and parallelizing the histogram: (a) element alias and scatter index and (b) atomic reduction.**

Figure 12 shows an overview of the proposed algorithm. We privatize the histogram table, where we effectively use the cache. Though the table size is large, only nonzero elements are cached, thus its working set will fit into the cache. Once the local table updates are done, we perform the global reduction. The original privatizing scheme reduces the entire private tables blindly. However, we only reduce nonzero entries. The original scheme does not require an atomic update because each core is assigned to different histogram entries so that there will be no conflict. However, our scheme needs an atomic update because nonzero elements of each core can be mapped to the same global entry. We implement an atomic update using the locking scheme. The proposed scheme works only if nonzero elements are sparse. We observe many sparse histograms are used in practice, for which our algorithm provide a reasonable solution.

SINCE A CORE'S COMPUTATION CAPABILITY LARGELY STEMS FROM ITS WIDE VECTOR UNITS, VECTORIZATION IS THE KEY FOR GOOD SINGLE-CORE PERFORMANCE.

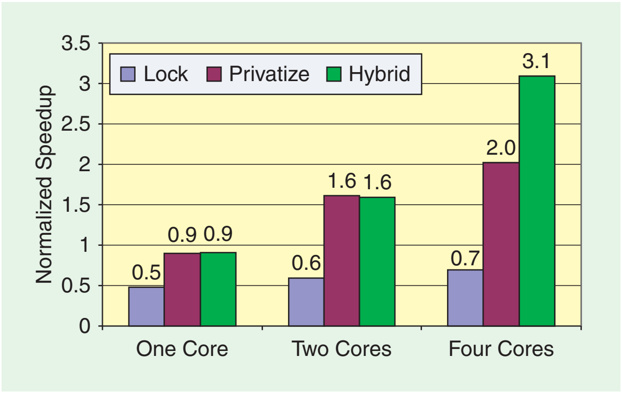
IMPACT OF OPTIMIZATION

We illustrate the impact of architecture-aware optimization by showing the performance analysis of an FFT implementation on an Intel's Core i7 system. We implement a 2K × 2K FFT, and compare the performance of two different optimization approaches.

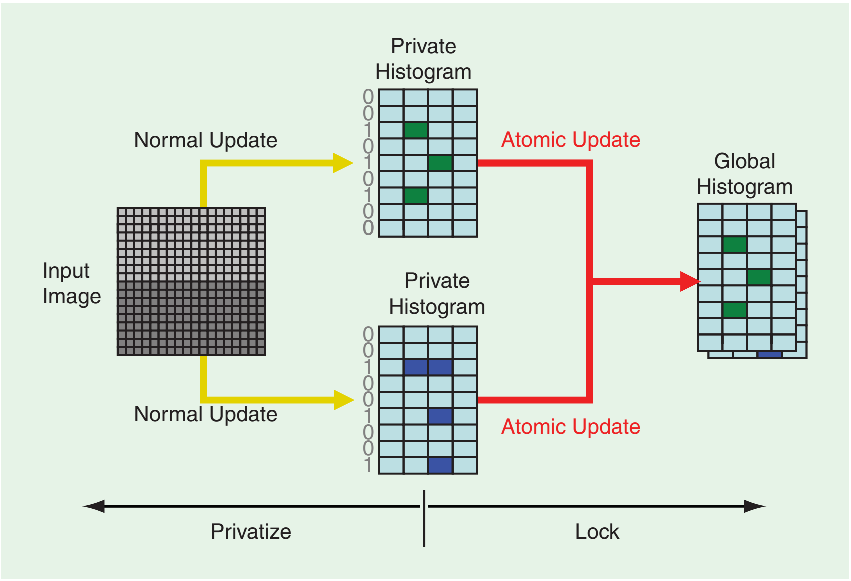
The first implementation step is to choose an algorithm. Assume that a naïve approach uses a radix-2 algorithm, but an optimized approach takes a radix-4 algorithm. Our hand-optimized implementations of both algorithms show that the radix-4 is 1.72x faster than the radix-2. The second decision is which vectorization scheme to use. The naïve one stops vector execution when the butterfly stride becomes less than four (the vector width of x86 SSE), while the optimized one keeps vectorizing to the final stage with the matrix transpose technique. In our implementations, full vectorization provides 1.18x speedup. The third decision is which parallelization method to use. We use the same parallelization scheme for both, because a trivial parallelization (assigning independent FFTs to each core) works well. However, the optimized approach uses a dynamic task scheduling for better load balancing while the naïve one use a static partitioning. Both schemes show almost identical performance for small number of cores (four cores). The fourth decision is which memory management scheme to use. The naïve implementation does not use any intelligent method and depends on the hardware prefetcher. The optimized one uses the double-buffering technique, which results in 1.14x speedup.

The final decision is which column-wise FFT scheme to use. The naïve scheme accesses the column-wise data directly, while the optimized one performs a matrix transpose before executing column-wise FFTs. Our implementation shows that the matrix transpose scheme is 1.16x faster than the direct access scheme. Overall, the optimized approach [30.1 giga floating-point operations per second (Gflops)] can achieve 2.68x speedup over the naïve approach (11.2 Gflops), which is summarized in Table 3.

We do not show a detailed performance analysis of convolution or histogram. Instead, Figure 13 summarizes our performance optimization results. It compares naïve implementations and our optimizations. The naïve approach uses only basic vectorization and parallelization, while we take advantage of architecture-aware optimizations discussed in this article. We



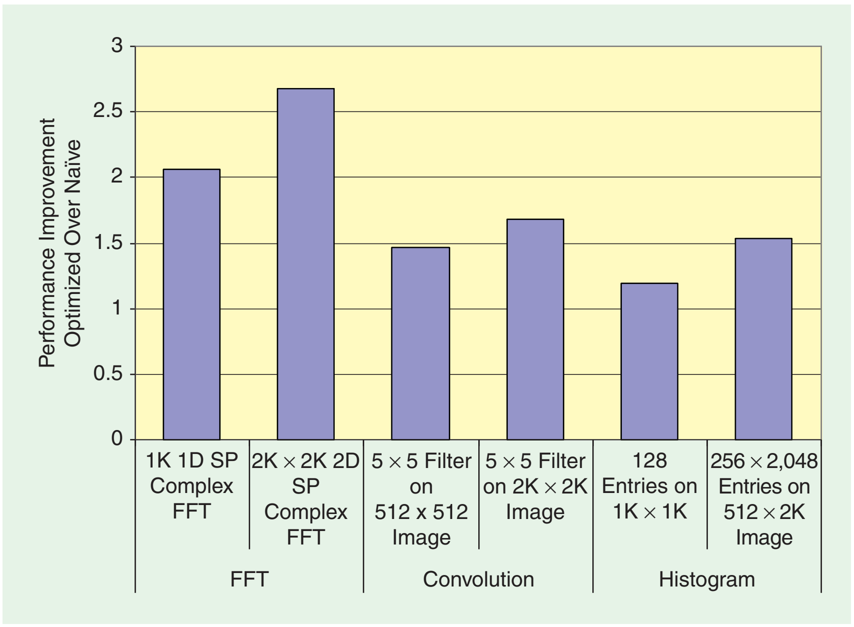
[FIG11] Multicore scalability of the three algorithms for a large histogram (256 × 2048 entries) on Core i7 four cores normalized to the sequential code.



[FIG12] Hybrid algorithm for a large histogram.

[TABLE 3] PERFORMANCE ANALYSIS OF A 2K × 2K SINGLE-PRECISION COMPLEX FFT BETWEEN A NAÏVE AND OPTIMIZED APPROACHES ON CORE I7 FOUR CORES.

	NAÏVE	OPTIMIZED	SPEEDUP (OPTIMIZED OVER NAÏVE)
ALGORITHM	RADIX-2	RADIX-4	1.72X
VECTORIZATION	PARTIAL VECTORIZING	FULL VECTORIZING	1.18X
PARALLELIZATION	STATIC PARTITIONING	DYNAMIC PARTITIONING	1.00X
MEMORY MANAGEMENT	HARDWARE PREFETCHING	DOUBLE BUFFERING	1.14X
COLUMN-WISE FFT	DIRECT COLUMN-WISE ACCESS	MATRIX TRANSPOSE	1.16X
OVERALL	11.2 GFLOPS	30.1 GFLOPS	2.68X

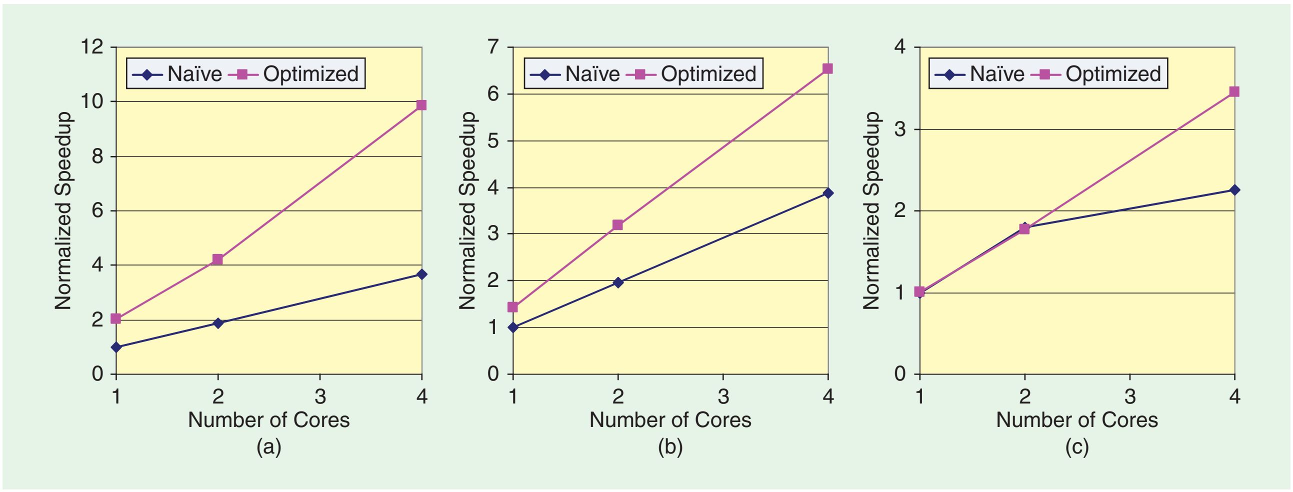


[FIG13] Performance improvement of our optimization over naïve implementation for selected image processing kernels on Core i7 four cores.

observed that performance optimization achieves from 1.19x to 2.68x speedup in various usage scenarios.

Figure 14 presents the multicore performance of the optimized and naïve implementations. It shows the normalized speedup with respect to one-core naïve performance. Initially, our optimization achieves higher single-core efficiency. Then, our optimization improves multicore scalability. Overall, combining the improvements in single-core efficiency and multicore scalability, our optimization accomplishes 9.8x (FFT), 6.5x (convolution), and 3.4x (histogram) speedup over one-core naïve implementations.

Relevance of optimization techniques discussed in this article is expected to grow as more cores are integrated into the multicore platform (e.g., Intel Larrabee whose architecture and vector instruction set are summarized in [22] and [1]). We discuss how our optimization techniques can be applied. First, since Larrabee architecture provides a natural extension to the conventional x86 programming model, the same optimization techniques can be applied to both Core i7 and Larrabee. Second, when optimizing for single core performance, wider (512-b) SIMD emphasizes the importance of vectorized algorithms. One should also note the growing SIMD width of traditional x86 multicores from 128b SSE to 256b AVX [24]. Third, since cores are connected with a high-speed on-die



[FIG14] Multicore performance of selected image processing kernels on Core i7 four cores (speedups are normalized to one-core naïve performance).

interconnect in Larrabee, lower costs in thread communication, and synchronization help multicore scalability. Lastly, optimizing for off-chip memory bandwidth becomes more important as on-die compute-density is expected to grow faster than external memory bandwidth. Traditional blocking techniques can therefore be expected to offer better performance return taking advantage of lower latency on-die caches and high-bandwidth on-die interconnect. In summary, our techniques allow the signal processing algorithms to achieve optimal performance in today's platforms and enable them to scale forward to optimal performance in tomorrow's platforms.

CONCLUSION

The computing industry is entering a new era of multicore architectures. Future multicore platforms will provide order-of-magnitude higher computational power than traditional single-core systems. Architecture-aware optimization allows us to approach optimal application performance on multicore systems. This article demonstrates the benefits of architecture-aware optimization. We define and then apply a set of optimization techniques to three common image processing kernels on Intel's x86 multicore processors. Architecture-blind naïve implementations cannot realize the full capability of the multicore system. Our optimizations help improve performance by 1.19x to 2.68x on a four-core Core i7 system through better vector utilization, higher multicore scalability, and more efficient bandwidth usage. We hope the optimization techniques and illustrative examples in this article provide a tutorial to aid developers of image processing applications to achieve optimal performance on future x86 multicore systems.

AUTHORS

Daehyun Kim (daehyun.kim@intel.com) received the Ph.D. degree in electrical and computer engineering from Cornell University in New York. He is a senior research scientist with Intel Throughput Computing Labs. His research interests include parallel computer architectures, intelligent memory systems, and emerging workloads. His current research interests in throughput computing involve analyzing future parallel applications and designing multicore microprocessors.

Victor W. Lee (victor.w.lee@intel.com) received his M.S. degree from Massachusetts Institutes of Technology. He is a senior staff research scientist with Intel Throughput Computing Labs. His research interests are emerging applications and their implications to computer architecture. He is currently involved in defining next-generation chip-multiprocessors architecture.

Yen-Kuang Chen (y.k.chen@ieee.org) received the Ph.D. degree from Princeton University in New Jersey. He is a principal research scientist at Intel Labs. His research interests include developing innovative multimedia applications, studying the performance bottleneck in current computer platforms, and designing next-generation processor/platforms with multiple cores. He is an associate editor of four journals and transactions, a lead guest editor of six special issues, a member of four IEEE technical committees, and a program committee member of over 35 international conferences on multimedia, video compression/communication, image/signal processing, VLSI circuits and systems, parallel processing, and software optimization. He is a Senior Member of the IEEE.

REFERENCES

[1] M. Abrash. A first look at the Larrabee new instructions (LRBni) [Online], White Paper. Available: http://www.gpucomputing.org/drdobbs_042909_final.pdf

[2] A. Ali, L. Johnsson, and J. Subhlok, "Scheduling FFT computation on SMP and multicore systems," in *Proc. Int. Conf. Supercomputing*, 2007, pp. 293–301.

[3] D. Bailey, "A high-performance FFT algorithm for vector supercomputers," *Int. J. Supercomput. Appl.*, vol. 2, no. 1, pp. 82–87, 1988.

[4] D. Bailey, "FFTs in external or hierarchical memory," *J. Supercomput.*, vol. 4, no. 1, pp. 23–35, 1990.

[5] G. Blake, R. Dreslinski, and T. Mudge, "A survey of multicore architectures," *IEEE Signal Processing Mag.*, vol. 26, no. 6, pp. 26–37, 2009.

[6] C. Breshears (2007). "8 simple rules for designing threaded applications [Online]. Available: http://softwarecommunity.intel.com/articles/eng/1607.htm

[7] A. Chow, G. Fossum, and D. Brokenshire, [Online]. Available: https://www-01.ibm.com/chips/techlib/techlib.nsf/techdocs/0AA2394A505EF0FB872570AB005BF0F1 "A programming example: Large FFT on the cell broadband engine," *IBM*, 2005.

[8] J. Cooley and J Tukey, "An algorithm for the machine calculation of complex Fourier series," *Math. Computat.*, vol. 19, no. 90, pp. 297–301, 1965.

[9] P. Duhamel and M. Vetterli, "Faster Fourier transforms: A tutorial review and a state of the art," *Signal Process.*, vol. 19, no. 4, pp. 259–299, 1990.

[10] F. Franchetti and M. Puschel, "Short vector code generation for the discrete Fourier transform," in *Proc. Int. Parallel and Distributed Processing Symp.*, 2002, pp. 20–26.

[11] F. Franchetti, M. Puschel, Y. Voronenko, S. Chellappa, and J. Moura, "Discrete Fourier transform on multicore," *IEEE Signal Processing Mag.*, vol. 26, no. 6, pp. 90–102, 2009.

[12] M. Frigo and S. Johnson, "The design and implementation of FFTW3," *Proc. IEEE*, vol. 93, no. 2, pp. 216–231, 2005.

[13] J. Greene and R. Cooper, "A parallel 64K complex FFT algorithm for the IBM/Sony/Toshiba cell broadband engine processor," in *Proc. Int. Conf. Supercomputing*, 2009, pp. 26–35.

[14] R. Rahman, "Fast Fourier transforms in the Intel math kernel library," Intel Corp., 2008.

[15] J. Johnson, R. Johnson, D. Rodriguez, and R. Tolimieri, "A methodology for designing, modifying, and implementing Fourier transform algorithms on various architectures," *IEEE Trans. Circuits Syst.*, vol. 9, no. 4, pp. 449–500, 1990.

[16] C. Lin and L. Snyder, *Principles of Parallel Programming*. Boston, MA: Pearson/Addison Wesley, 2009.

[17] C. Loan, *Computational Framework of the Fast Fourier Transform*. Philadelphia: SIAM, 1987.

[18] K. Moreland and E. Angel, "The FFT on a GPU," in *Proc. SIGGRAPH/EUROGRAOHCIS Workshop on Graphics Hardware*, 2003, pp. 112–119.

[19] M. Püschel, J. Moura, J. Johnson, D. Padua, M. Veloso, B. Singer, J. Xiong, F. Franchetti, A. Gacic, Y. Voronenko, K. Chen, R. Johnson, and N. Rizzolo, "SPIRAL: Code generation for DSP transforms," *Proc. IEEE*, vol. 93, no. 2, pp. 232–275, 2005.

[20] M. Ren, J. Park, M. Houston, A. Aiken, and W. Dally, "A tuning framework for software-managed memory hierarchies," in *Proc. Int. Conf. Parallel Architectures and Compilation Techniques*, 2008, pp. 280–291.

[21] S. Ryoo, C. Rodrigues, S. Baghsorkhi, D. Kirk, and W. Hwu, "Optimization principles and application performance evaluation of a multithreaded GPU using CUDA," in *Proc. 13th ACM SIGPLAN Symp. Principles and Practice of Parallel Programming*, 2008, pp. 73–82.

[22] L. Seiler, D. Carmean, E. Sprangle, T. Forsyth, M. Abrash, P. Dubey, S. Junkins, A. Lake, J. Sugerman, R. Cavin, R. Espasa, E. Grochowski, T. Juan, and P. Hanrahan, "Larrabee: A many-core x86 architecture for visual computing," in *Proc. SIGGRAPH*, 2008, pp. 1–15.

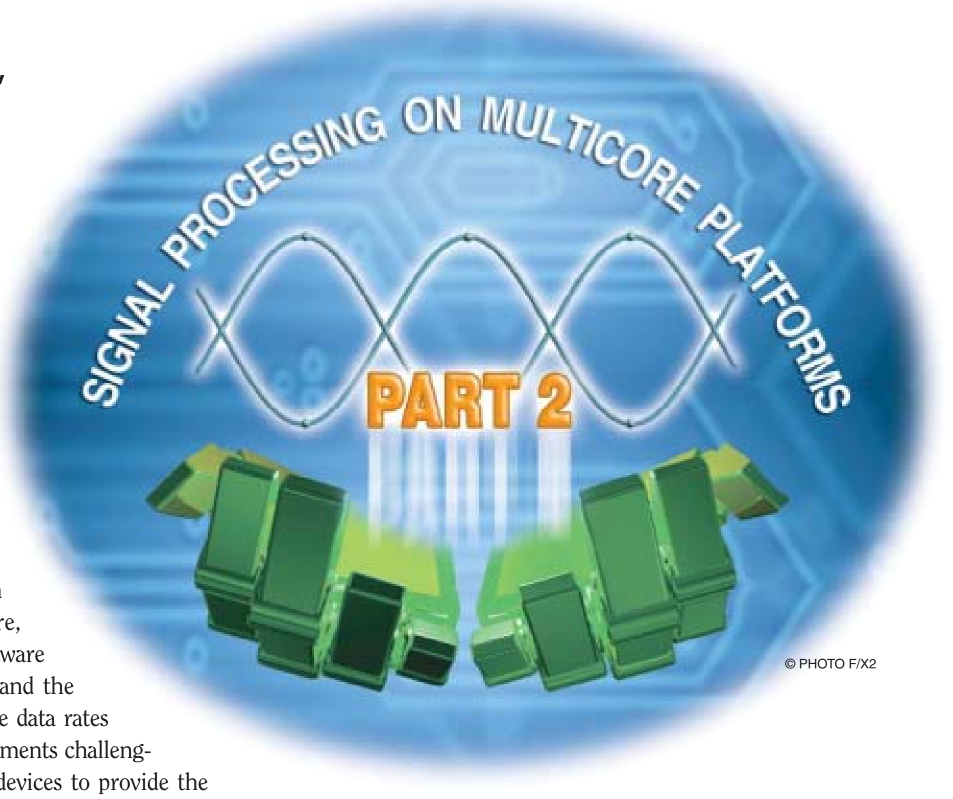
[23] V. Volkov and B. Kazian. *Fitting FFT onto the G80 architecture* [Online]. Available: http://www.cs.berkeley.edu/~kubitron/courses/cs258-S08/projects/reports/project6_report.pdf

[24] Intel Corp., *Intel Advanced Vector Extensions Programming*, Ref. No. 319433-006, July 2009. [Online]. Available: http://software.intel.com/file/21558 [SP]

[Rob V. van Nieuwpoort and John W. Romein]

Building Correlators with Many-Core Hardware

[A look at performance, optimization, and programmability]



© PHOTO F/X2

Radio telescopes typically consist of multiple receivers whose signals are cross-correlated to filter out noise. A recent trend is to correlate in software instead of custom-built hardware, taking advantage of the flexibility that software solutions offer. Examples include e-VLBI and the low frequency array (LOFAR). However, the data rates are usually high and the processing requirements challenging. Many-core processors are promising devices to provide the required processing power. In this article, we explain how to implement and optimize signal-processing applications on multicore CPUs and many-core architectures, such as the Intel Core i7, NVIDIA and ATI graphics processor units (GPUs), and the Cell/BE. We use correlation as a running example. The correlator is a streaming, possibly real-time application, and is much more input/output (I/O) intensive than applications that are typically implemented on many-core hardware today. We compare with the LOFAR production correlator on an IBM Blue Gene/P (BG/P) supercomputer. We discuss several important architectural problems which cause architectures to perform suboptimally, and also deal with programmability.

The correlator on the BG/P achieves a superb 96% of the theoretical peak performance. We show that the processing power and memory bandwidth of current GPUs are highly imbalanced. Because of this, the correlator achieves only 16% of the peak on ATI GPUs, and 32% on NVIDIA GPUs. The Cell/BE processor, in contrast, achieves an excellent 92%. Many of the insights we discuss here are not only applicable to telescope correlators, but are valuable when developing signal-processing applications in general.

Digital Object Identifier 10.1109/MSP.2009.935385

INTRODUCTION

Radio telescopes produce enormous amounts of data. LOFAR [1], for instance, will produce some tens of petabits per day, and the Australian square kilometer array pathfinder (ASKAP) will even produce over six exabits per day [2]. These modern radio telescopes use many separate receivers as building blocks and combine their signals to form a single large and sensitive instrument.

To extract the sky signal from the system noise, the correlator coordinates the signals from different receivers, and integrates the correlations over time to reduce the amount of data. This is a challenging problem in radio astronomy, since the data volumes are large and the computational demands grow quadratically with the number of receivers. Correlators are not limited to astronomy but are also used in geophysics [3], radar systems [4], and wireless networking [5].

Traditionally, custom-built hardware, and later field programmable gate arrays (FPGAs), were used to correlate telescope signals. A recent development is to use a supercomputer [6]. Both approaches have important advantages and disadvantages. Custom-built hardware is efficient and consumes modest amounts of power but is inflexible, expensive to design, and has a long development time. Solutions that use a supercomputer are much more flexible, but are less efficient and consume more power. Future instruments, like the square kilometer array (SKA), need several orders of magnitude more computational resources. It is likely that the requirements of the SKA cannot be met by using current supercomputer technology. Therefore, it is important to investigate alternative hardware solutions.

General-purpose architectures no longer achieve performance improvements by increasing the clock frequency but by adding more compute cores and by exploiting parallelism. Intel's recent Core i7 processor is a good example of this. It has four cores and supports additional vector parallelism. Furthermore, the high-performance computing community is steadily adopting clusters of GPUs as a viable alternative to supercomputers, due to their unparalleled growth in computational performance, increasing flexibility and programmability, high power efficiency, and low purchase costs. GPUs are highly parallel and contain hundreds of processor cores. An example of a processor that combines GPU and central processing unit (CPU) qualities into one design is the Cell Broadband Engine (Cell/BE) [7]. The Cell/BE consists of an "ordinary" PowerPC core and eight powerful vector processors that provide the bulk of the processing power. Programming the Cell/BE requires more effort than programming an ordinary CPU, but various studies showed that the Cell/BE performs well on signal-processing tasks like fast Fourier transforms (FFTs) [8].

In this article, we explain how many-core architectures can be exploited for signal-processing purposes. We give insights into their architectural limitations, and how to best cope with them. We treat five different, popular architectures with multiple cores: the Cell/BE, GPUs from both NVIDIA and ATI, the Intel Core i7 processor, and the BG/P supercomputer. We discuss their similarities and differences, and how the architectural differences affect optimization choices and the eventual performance of a correlator. We also discuss the programmability of the architectures. We focus on correlators but many of the findings, claims, and optimizations hold for other signal-processing algorithms as well, both inside and outside the area of radio astronomy. For instance, we discussed radio-astronomy imaging (another signal processing algorithm) on many-core hardware in previous work [9].

In this article, we use the LOFAR telescope as a running example and use its production correlator on the BG/P as a comparison. This way, we demonstrate how many-core architectures can be used in practice for a real application. For educational purposes, we made the correlator implementations for all architectures available online. They exemplify the different optimization choices for the different architectures. The code may be reused under the GNU public license. We describe and analyze the correlator on many-core platforms in much more detail in [10].

TRENDS IN RADIO ASTRONOMY

During the past decade, new types of radio-telescope concepts emerged that rely less on concrete, steel, and extreme cooling techniques, but more on signal-processing techniques. For example, LOFAR [1], Karoo Array Telescope (MeerKAT) [11], and ASKAP [2] are distributed sensor networks that combine the signals of many receiver elements. All three are pathfinders for the future SKA [12] telescope, which will be orders-of-magnitude larger. These instruments combine the advantages of higher sensitivity, higher resolution, and multiple concurrent observation directions. However, they require huge amounts of processing power to combine the data from the receiving elements.

The signal-processing hardware technology used to process telescope data also changes rapidly. Only a decade ago, correlators required special-purpose application-specific integrated circuits (ASICs) to keep up with the high data rates and processing requirements. The advent of sufficiently fast FPGAs significantly lowered the developments times and costs of correlators and increased the flexibility substantially. LOFAR requires even more flexibility to support many different processing pipelines for various observation modes, and uses FPGAs for on-the-field processing and a BG/P supercomputer to perform real-time central processing. We describe LOFAR in more detail below.

THE LOFAR TELESCOPE

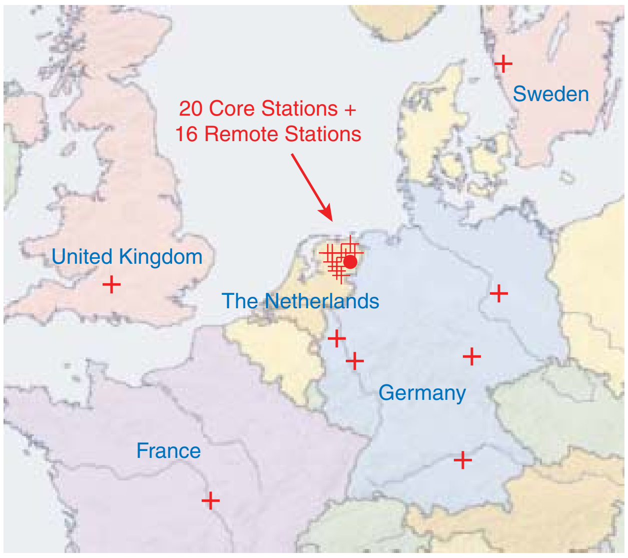
LOFAR is an aperture array radio telescope operating in the 10–250 MHz frequency range [1]. It is the first of a new generation of radio telescopes that breaks with the concepts of traditional telescopes in several ways. Rather than using large, expensive dishes, LOFAR uses many thousands of simple antennas that have no movable parts (see Figure 1). Essentially, it is a distributed sensor network that monitors the sky and combines all signals centrally. This concept requires much more signal processing, but the costs of the silicon for the processing are much lower that the costs of steel that would be needed for dishes. Moreover, LOFAR can observe the sky in many directions concurrently and switch directions instantaneously. In several ways, LOFAR will be the largest telescope of the world. The antennas are simple, but there are a lot of them—44,000 in the full LOFAR design. To make radio pictures of the sky with adequate resolution, these antennas are to be arranged in clusters. In the rest of this article, we call a cluster of antennas "receivers." The receivers will be spread out over an area of ultimately 350 km in diameter. This is shown in Figure 2.

[FIG1] A field with LOFAR antennas (photo courtesy of ASTRON).

Data transport requirements are in the range of many terabits/s and the processing power needed is tens of tera-ops.

Another novelty is the elaborate use of software to process the telescope data in real time. LOFAR thus is an IT-telescope. The cost is dominated by the cost of computing and will follow Moore's law; becoming cheaper with time and allowing increasingly large telescopes to be built.

LOFAR will enable exciting new science cases. First, we expect to see the epoch of reionization, the time that the first star galaxies and quasars were formed. Second, LOFAR offers a unique possibility in particle astrophysics for studying the origin of high-energy cosmic rays. Third, LOFAR's ability to continuously monitor a large fraction of the sky makes it uniquely suited to find new pulsars and to study transient sources. Since LOFAR has no moving parts, it can instantaneously switch focus to some galactic event. Fourth, deep extragalactic surveys will be carried out to find the most distant radio galaxies and study star-forming galaxies. Fifth, LOFAR will be capable of observing the so far unexplored radio waves emitted by cosmic magnetic fields. For a more extensive description of the astronomical aspects of the LOFAR system, see [13].



[FIG2] LOFAR layout.

A global overview of the LOFAR processing is given in Figure 3. The thickness of the lines indicates the size of the data streams. Initial processing is done in the field, using FPGA technology. Typical operations that are performed there include analog-to-digital conversion, filtering, frequency selection, and combination of the signals from the different antennas. Next, the data is transported to the central processing location in Groningen, The Netherlands, via dedicated optical wide-area networks.

The real-time central processing of LOFAR data is done on a BG/P supercomputer. There, we filter the data and perform phase-shift and bandpass-corrections. Next, the signals from all receivers are cross correlated. The correlation process performs a data reduction by integrating samples over time. Finally, the data is forwarded to a storage cluster, where results can be kept for several days. After an observation has finished, further processing, such as radio frequency interference (RFI) removal, calibration, and imaging is done offline on commodity cluster hardware. In this article, we focus on the correlator step (the highlighted part in the red box in Figure 3), because it must deal with the full data streams from all receivers. Moreover, its costs grow quadratically with the number of receivers, while all other steps have a lower time complexity.

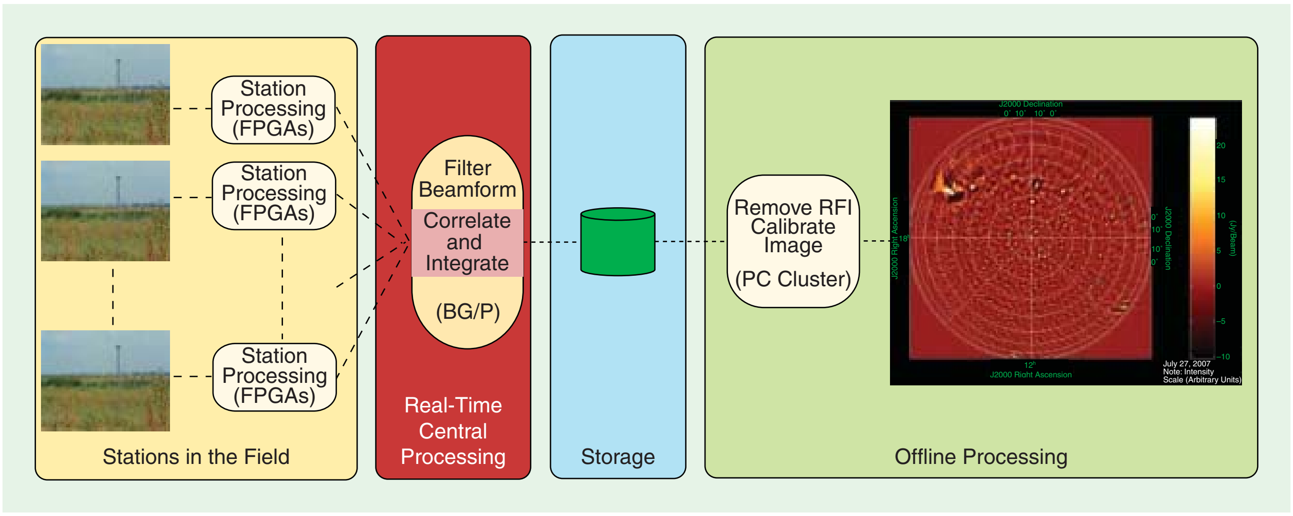
CORRELATING SIGNALS

LOFAR's receivers are dual-polarized; they take separate samples from orthogonal (X and Y) directions. The receivers support 4-, 8-, and 16-b integer samples, where the normal mode of operation uses the 16-b samples to help mitigate the impact of strong RFI. The smaller samples are important for observations that require larger sky coverage. Before filtering and correlating, the samples are converted to single-precision floating point, since all architectures support this well. This is accurate enough for our purposes. From the perspective of the correlator, samples thus consist of four 32-b floating-point numbers: two polarizations, each with a real and an imaginary part.

LOFAR uses an FX correlator: it first filters the different frequencies and then correlates the signals. This is more efficient than an XF correlator for larger numbers of receivers. In FX and XF, F means frequency separation and X means correlation (multiplication of the signal).

Prior to correlation, the data that comes from the receivers must be reordered: each input carries the signals of many frequency bands from a single receiver, but the correlator needs data from a single frequency of all inputs. Depending on the data rate, switching the data can be a real challenge. The data reordering phase is outside the scope of this article, but a correlator implementation cannot ignore this issue. The LOFAR BG/P correlator uses the fast three-dimensional torus for this purpose; other multicore architectures need external switches.

The received signals from sky sources are so weak, that the antennas mainly receive noise. To see if there is statistical coherence in the noise, simultaneous samples of each pair of receivers are correlated, by multiplying the sample of one receiver with the complex conjugate of the sample of the other receiver. To reduce the output size, the correlations are integrated over time, by


Station Processing (FPGAs)
Station Processing (FPGAs)
Station Processing (FPGAs)
Stations in the Field
Filter
Beamform
Correlate and Integrate
(BG/P)
Real-Time Central Processing
Storage
Remove RFI
Calibrate
Image
(PC Cluster)
Offline Processing
July 27, 2007
Note: Intensity Scale (Arbitrary Units)

[FIG3] **A simplified overview of the LOFAR processing.**

accumulating all products. Therefore, the correlator is mostly multiplying and adding complex numbers. Both polarizations of a station A are correlated with both polarizations of a station B, yielding correlations in XX, XY, YX, and YY directions. The correlator algorithm itself thus is straightforward, and can be written in a single formula

$$C_{s_1,s_2\geq s_1,p_1\in\{X,Y\},p_2\in\{X,Y\}} = \sum_t Z_{s_1,t,p_1} * Z^*_{s_2,t,p_2}.$$

The total number of correlations we have to compute is $(nr\text{Receivers} \times (nr\text{Receivers}+1))/2$, since we need each pair of correlations only once. This includes the autocorrelations (the correlation of a receiver with itself), since we need them later in the pipeline for calibration purposes. The autocorrelations can be computed with fewer instructions. We can implement the correlation operation very efficiently, with only four fused-multiply-add (FMA) instructions, doing eight floating-point operations in total. For each pair of receivers, we have to do this four times, once for each combination of polarizations. Thus, in total we need 32 operations. To perform these operations, we have to load the samples generated by two different receivers from memory. As explained above, the samples each consist of four single-precision floating-point numbers. Therefore, we need to load eight floats or 32 B in total. This results in exactly one floating-point operations per second (FLOP)/byte. We will describe the implementation and optimization of the correlator on the many-core systems in more detail in the section "Implementation and Optimization," but first, we explain the architectures themselves.

MANY-CORE ARCHITECTURES

In this section, we explain key properties of five different architectures with multiple cores and the most important differences between them. Table 1 shows the most important properties of the different many-core architectures.

GENERAL-PURPOSE MULTICORE CPUs (INTEL CORE i7)

As a reference, we implemented the correlator on a multicore general-purpose architecture, in this case an Intel Core i7. The theoretical peak performance of the system is 85 gflops, in single precision. The parallelism comes from four cores with hyperthreading. Using two threads per core allows the hardware to overlap load delays and pipeline stalls with useful work from the other thread. The SSE4 instruction set provides single instruction multiple data (SIMD) parallelism with a vector length of four floats.

IBM BG/P SUPERCOMPUTER

The IBM BG/P [14] is the architecture that is currently used for the LOFAR correlator. Four PowerPC processor cores are integrated on each BG/P chip. Each core is extended with two floating-point units (FPUs) that provide the bulk of the processing power. The BG/P is an energy-efficient supercomputer. This is accomplished by using many small, low-power chips, at a low clock frequency.

ATI GPUs

ATI's GPU with the highest performance is the Radeon 4870 [15]. The chip contains 160 cores, with 800 FPUs in total, and has a theoretical peak performance of 1.2 teraflops. The board uses a

[TABLE 1] PROPERTIES OF THE DIFFERENT MANY-CORE PLATFORMS.

ARCHITECTURE	INTEL CORE i7	IBM BG/P	ATI 4870	NVIDIA TESLA C1060	STI CELL/BE
GFLOPS PER CHIP	**85**	**13.6**	**1200**	**936**	**204.8**
CLOCK FREQUENCY (GHz)	2.67	0.850	0.75	1.296	3.2
CORES × FPUS PER CORE = **TOTAL FPUs**	4 × 4 = **16**	4 × 2 = **8**	160 × 5 = **800**	30 × 8 = **240**	8 × 4 = **32**
REGISTERS PER CORE × REGISTER WIDTH	16 × 4	64 × 2	1024 × 4	2048 × 1	128 × 4
TOTAL DEVICE RAM BANDWIDTH (GB/s)	N.A.	N.A.	115.2	102	N.A.
TOTAL HOST RAM BANDWIDTH (GB/s)	**25.6**	**13.6**	**4.6**	**5.6**	**25.8**

PCI-express 2.0 interface for communication with the host system. The GPU has 1 GB of device memory onboard. It is possible to specify if a read should be cached by the texture cache or not. Each streaming processor also has 16 KB of shared memory that is completely managed by the application. On both ATI and NVIDIA GPUs, the application should run many more threads than the number of cores. This allows the hardware to overlap memory load delays with useful work from other threads.

NVIDIA GPUs

NVIDIA's Tesla C1060 contains a GTX 280 GPU with 240 single-precision and 30 double-precision FPUs [16]. The GTX 280 uses a two-level hierarchy to group cores. There are 30 independent multiprocessors that each have eight cores. Current NVIDIA GPUs have fewer cores than ATI GPUs, but the individual cores are faster. The theoretical peak performance is 933 gflops. The number of registers is large: each multiprocessor has 16,384 32-b floating-point registers that are shared between all threads that run on it. There is also 16 KB of shared memory per multiprocessor. Finally, texture-caching hardware is available. The application can specify which area of device memory must be cached, while the shared memory is completely managed by the application.

THE CELL/BE

The Cell/BE [7] is a heterogeneous many-core processor, designed by Sony, Toshiba, and IBM (STI). The Cell/BE has nine cores: one power processing element (PPE), acting as a main processor, and eight synergistic processing elements (SPEs) that provide the real processing power. A SPE contains a reduced instruction set computing (RISC) core, a 256 KB local store (LS), and a direct memory access (DMA) controller. The LS is an extremely fast local memory for both code and data and is managed entirely by the application with explicit DMA transfers to and from main memory. The LS can be considered the SPU's (explicit) L1 cache. The Cell/BE has a large number of registers: each SPU has 128, which are 128-b (four floats) wide. The SPU can dispatch two instructions in each clock cycle using the two pipelines designated even and odd. Most of the arithmetic instructions execute on the even pipe, while most of the memory instructions execute on the odd pipe. For the performance evaluation, we use a QS21 Cell blade with two Cell/BE processors. The eight SPEs of a single chip in the system have a total theoretical single-precision peak performance of 205 gflops.

MAPPING SIGNAL-PROCESSING ALGORITHMS ON MANY-CORE HARDWARE

Many-core architectures derive their performance from parallelism. Several different forms of parallelism can be identified: multithreading (with or without shared memory), overlapping of I/O and computations, instruction-level parallelism, and vector parallelism. Most many-core architectures combine several of these methods. Unfortunately, an application has to handle all available levels of parallelism to obtain good performance. Therefore, it is clear that algorithms have to be adapted to efficiently exploit many-core hardware. Additional parallelism can be obtained by using multiple processor chips. In this article, however, we restrict ourselves to single chips for simplicity.

FINDING PARALLELISM

The first step is to find parallelism in the algorithm, on all different levels. Basically, this means looking for independent operations. With the correlator, for example, the thousands of different frequency channels are completely independent, and they can be processed in parallel. But there are other, more fine-grained sources of parallelism as well. The correlations for each pair of receivers are independent, just like the four combinations of polarizations. Finally, samples taken at different times can be correlated independently, as long as the subresults are integrated later. Of course, the problem now is how to map the parallelism in the algorithm to the parallelism provided by the architecture. We found that, even for the relatively straightforward correlator algorithm, the different architectures require very different mappings and strategies.

OPTIMIZING MEMORY PRESSURE AND ACCESS PATTERNS

On many-core architectures, the memory bandwidth is shared between the cores. This has shifted the balance between computational and memory performance. The available memory bandwidth per operation has decreased dramatically compared to traditional processors. For the many-core architectures we use here, the theoretical bandwidth per operation is three to ten times lower than on the BG/P, for instance. In practice, if algorithms are not optimized well for many-core platforms, the achieved memory bandwidth can easily be ten to 100 times lower than the theoretical maximum. Therefore, we must treat memory bandwidth as a scarce resource, and it is important to minimize the number of memory accesses. In fact, one of the most important lessons of this article is that on many-core architectures, optimizing the memory properties of the algorithms is more important than focusing on reducing the number of compute cycles that is used, as is traditionally done on systems with only a few or just one core.

WELL-KNOWN MEMORY OPTIMIZATION TECHNIQUES

The insight that optimizing the interaction with the memory system is becoming more and more important is not new. The book by Catthoor et al. [17] is an excellent starting point for more information on memory-system related optimizations.

We can make a distinction between hardware and software memory-optimization techniques. Examples of hardware-based techniques include caching, data prefetching, write combining, and pipelining. The software techniques can be divided further into compiler optimizations and algorithmic improvements. The distinction between hardware and software is not entirely black and white. Data prefetching, for instance, can be done both in hardware and software. Another good example is the explicit cache of the Cell/BE processor. This is an architecture where the programmer handles the cache replacement policies instead of the hardware.

Many optimizations focus on utilizing data caches more efficiently. Hardware cache hierarchies can, in principle, transparently improve application performance. Nevertheless, it is important to take the sizes of the different cache levels into account when optimizing an algorithm. A cache line is the smallest unit of memory than can be transferred between the main memory and the cache. Code can be optimized for the cache line size of a particular architecture. Moreover, the associativity of the cache can be important. If a cache is N-way set associative, this means that any particular location in memory can be cached in either of N locations in the data cache. Algorithms can be designed such that they take care that cache lines that are needed later are not replaced prematurely. In addition, write combining, a technique that allows data writes to be combined and written later in burst mode, can be used if the ordering of writes is not important. Finally, prefetching can be used to load data into caches or registers ahead of time.

Many cache-related optimization techniques have been described in the literature, both in the context of hardware and software. For instance, an efficient implementation of hardware-based prefetching is described in [18]. As we will describe in the section "Implementation and Optimization," we implemented prefetching manually in software, for example by using multibuffering on the Cell/BE, or by explicitly loading data into shared memory or registers on the GPUs. A good starting point for cache-aware or cache-oblivious algorithms is [19]. An example of a technique that we used to improve cache efficiencies for the correlator is the padding of multidimensional arrays with extra "dummy" data elements. This can be especially important if memory is accessed with a stride of a (large) power of two. This way, we can make sure that cache replacement policies work well, and subsequent elements in an array dimension are not mapped onto the same cache location. This well-known technique is described, for instance, by Bacon et al. [20]. Many additional data access patterns optimization techniques are described in [17].

Many memory-optimization techniques have been developed in the context of optimizing compilers and run-time systems (e.g., efficient memory allocators). For instance, a lot of research effort has been invested in cache-aware memory allocation; see e.g., [21]. Compilers can exploit many techniques to optimize locality by applying code and loop transformations such as interchange, reversal, skewing, and tiling [22]. Furthermore, compilers can optimize code for the parameters and sizes of the caches by carefully choosing the placement of variables, objects, and arrays in memory [23].

The memory systems of the many-core architectures are quite complex. GPUs, for instance, have banked device memory, several levels of texture cache, in addition to local memory, application-managed shared memory (also divided over several banks), and write combining buffers. There also are complex interactions between the memory system and the hardware thread scheduler. GPUs literally run tens of thousands of parallel threads to overlap memory latencies, trying to keep all functional units fully occupied. We apply the techniques described above in software by hand, since we found that the current compilers for the many-core architectures do not (yet) implement them well on their complex memory systems.

APPLYING THE TECHNIQUES

So, the second step of mapping a signal-processing algorithm to a many-core architecture is optimizing the memory behavior. We can split this step into two phases: an algorithm phase and an architectural phase. In the first phase, we identify algorithm-specific, but architecture-independent optimizations. In this phase, it is of key importance to understand that, although a set of operations in an algorithm can be independent, the data accesses may not be. This is essential for good performance, even though it may not be a factor in the correctness of the algorithm. The number of memory accesses per operation should be reduced as much as possible, sometimes even at the cost of more compute cycles. An example is a case where different parallel operations read (but not write) the same data. For the correlator, the most important insight here is a technique to exploit date reuse opportunities, reducing the number of memory loads. We explain this in detail in the section "Architecture-Independent Optimizations."

The second phase deals with architecture-specific optimizations. In this phase, we do not reduce the number of memory loads, but think about the memory access patterns. Typically, several cores share one or more cache levels. Therefore, the access patterns of several different threads that share a cache should be tailored accordingly. On GPUs, for example, this can be done by coalescing memory accesses. This means that different concurrent threads read subsequent memory locations. This can be counter-intuitive, since traditionally, it was more efficient to have linear memory access patterns within a thread. Table 2 summarizes the differences in memory architectures of the different platforms. Other techniques that are performed in this phase include optimizing cache behavior, avoiding load delays and pipeline stalls, and exploiting special floating-point instructions. We explain several examples of this in more detail in the section "Architecture-Specific Optimizations."

A SIMPLE ANALYTICAL TOOL

A simple analytic approach, the Bound and Bottleneck analysis [24], [25], can provide more insight on the memory properties of an algorithm. It also gives us a reality check and calculates what the expected maximal performance is that can be achieved on a particular platform. The number of operations that is performed per byte that have to be transferred (the flop/byte ratio) is called the arithmetic intensity (AI) [24]. Performance is bound by the

[TABLE 2] DIFFERENCES BETWEEN MEMORY ARCHITECTURES.

FEATURE	CELL/BE	GPUs
ACCESS TIMES	UNIFORM	NONUNIFORM
CACHE SHARING LEVEL	SINGLE THREAD (SPE)	ALL THREADS IN A MULTIPROCESSOR
ACCESS TO OFF-CHIP MEMORY	THROUGH DMA ONLY	SUPPORTED
MEMORY ACCESS OVERLAPPING	ASYNCHRONOUS DMA	HARDWARE-MANAGED THREAD PREEMPTION
COMMUNICATION	DMA BETWEEN SPEs	INDEPENDENT THREAD BLOCKS AND SHARED MEMORY WITHIN A BLOCK

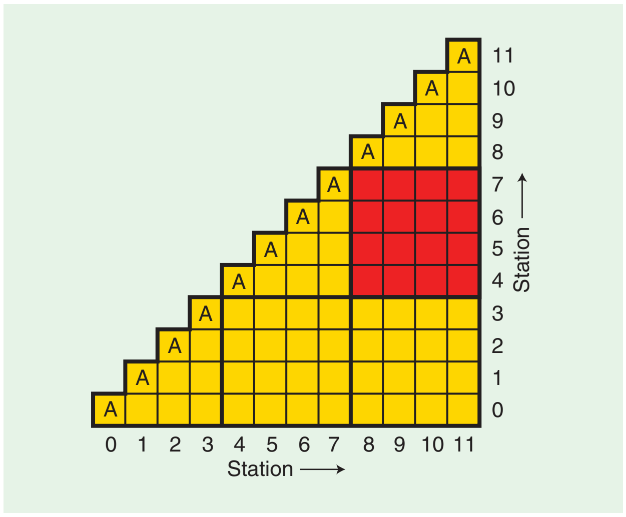
product of the bandwidth and the AI:perf$_{\max}$= AI $\times$ bandwidth. Several important assumptions are made with this method. First, it assumes that the bandwidth is independent of the access pattern. Second, it assumes a complete overlap of communication and computation, i.e., all latencies are completely hidden. Finally, the method does not take caches into account. Nevertheless, it gives a rough idea of the performance than can be achieved.

It is insightful to apply this method to the correlator on the GPUs. We do it for the NVIDIA GPU here, but the results for the ATI hardware is similar. With the GPUs, there are several communication steps that influence the performance. First, the data has to be transferred from the host to the device memory. Next, the data is read from the device memory into registers. The host-to-device bandwidth is limited by the low PCI-express throughput, 5.6 GB/s in this case. We can easily show that this is a bottleneck by computing the AI for the full system, using the host-to-device transfers. (The AI can also be computed for the device memory.)

As explained in the section "Correlating Signals," the number of flops in the correlator is the number of receiver combinations times 32 operations, while the number of bytes that have to be loaded in total is 16 B times the number of receivers. The number of combinations is $(nr\text{Receivers} \times (nr\text{Receivers} + 1))/2$ (see the section "Correlating Signals"). If we substitute this, we find that the AI $= nr\text{Receivers} + 1$. For LOFAR, we can assume 64 receivers (each in turn containing many antennas), so the AI is 65 in our case. Therefore, the performance bound on NVIDIA hardware is $65 \times 5.6 = 364$ gflops. This is only 39% of the theoretical peak. Note that this even is optimistic, since it assumes perfect overlap of communication and computation.

COMPLEX NUMBERS

Support for complex numbers is important for signal processing. Explicit hardware support for complex operations is preferable, both for programmability and performance. Except for the BG/P, none of the architectures support this. The different architectures require two different approaches of dealing with this problem. If an architecture does not use explicit vector parallelism, the complex operations can simply be expressed in terms of normal floating-point operations. This puts an extra burden on the programmer, but achieves good performance. The NVIDIA GPUs work this way. If an architecture does use vector parallelism, we can either store the real and complex parts alternatingly inside a single vector, or have separate vectors for the two parts. In both cases, support for shuffling data inside the vector registers is essential, since complex multiplications operate on both the real and imaginary parts. The architectures differ considerably in this respect. The Cell/BE excels; its vectors contain four floats, which can be shuffled around in arbitrary patterns. Moreover, shuffling and computations can be overlapped effectively. On ATI GPUs, this works similarly. The SSE4 instructions in the Intel CPUs do not support arbitrary shuffling patterns. This has a large impact on the way the code is vectorized.



[FIG4] **An example correlation triangle.**

IMPLEMENTATION AND OPTIMIZATION

In this section, we explain the techniques described above by applying them to the correlator for all different architectures.

ARCHITECTURE-INDEPENDENT OPTIMIZATIONS

An unoptimized correlator would read the samples from two receivers and multiply them, requiring two sample loads for one multiplication. We can optimize this by reusing a sample as often as possible, by using it for multiple correlations (see Figure 4). The figure is triangular, because we compute the correlation of each pair of receivers only once. The squares labeled *A* are autocorrelations. For example, the samples from receivers 8, 9, 10, and 11 can be correlated with the samples from receivers 4, 5, 6, and 7 (the red square in the figure), reusing each fetched sample four times. By dividing the correlation triangle in 4×4 tiles, eight samples are read from memory for sixteen correlations, reducing the amount of memory operations by a factor of four. The maximum number of receivers that can be simultaneously correlated this way (i.e., the tile size) is limited by the number of registers that an architecture has. The samples and accumulated correlations are best kept in registers, and the number of required registers grows rapidly with the number of receiver inputs. The example above already requires 16 accumulators. To obtain good performance, it is important to tune the tile size to the architecture. There still is opportunity for additional data reuse between tiles. The tiles within a row or column in the triangle still need the same samples. In addition to registers, caches can thus also be used to increase data reuse.

ARCHITECTURE-SPECIFIC OPTIMIZATIONS

We will now describe the implementation of the correlator on the different architectures, evaluating the performance and optimizations needed in detail. For comparison reasons, we use the performance per chip for each architecture. The performance results are shown in Figure 5.

INTEL CPUs

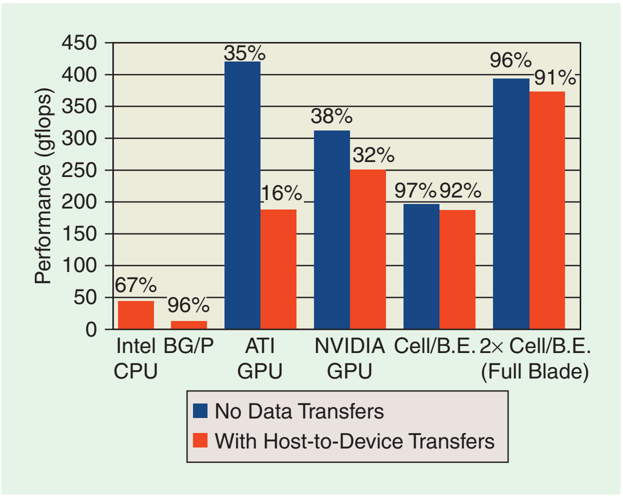
The SSE4 instruction set can be used to exploit vector parallelism. Unlike the Cell/BE and ATI GPUs, a problem with SSE4 is the limited support for shuffling data within vector registers. Computing the correlations of the four polarizations within a vector is inefficient, and computing four samples with subsequent time stamps in a vector works better. The use of SSE4 improves the performance by a factor of 3.6 in this case. In addition, multiple threads should be used to utilize all four cores. To benefit from hyperthreading, twice as many threads as cores are needed. For the correlator, hyperthreading increases performance by 6%. Also, the number of vector registers is small. Therefore, there is not much opportunity to reuse data in registers, limiting the tile size to 2 × 2; reuse has to come from the L1 cache.

THE BG/P SUPERCOMPUTER

We found that the BG/P is extremely suitable for our application, since it is highly optimized for processing of complex numbers. However, the BG/P performs all floating-point operations in double precision, which is overkill for our application. Although the BG/P can keep the same number of values in register as the Intel chip, an important difference is that the BG/P has 32 registers of width two, compared to Intel's 16 of width four. The smaller vector size reduces the amount of shuffle instructions needed. In contrast to all other architectures we evaluate, the problem is compute bound instead of I/O bound, thanks to the BG/P's high memory bandwidth per operation, which is three to ten times higher than for the other architectures.

ATI GPUs

The ATI architecture has several important drawbacks for data-intensive applications. First, the host-to-device bandwidth is a bottleneck. Second, overlapping communication with computation does not work well. We observed kernel slowdowns of more than a factor of two due to asynchronous transfers in the background. This can clearly be seen in Figure 5. Third, the architecture does not provide random write access to device memory, but only to host memory. The correlator reduces the data by a large amount, and the results are never reused by the kernel. Therefore, they can be directly streamed to host memory. Nevertheless, in general, the absence of random write access to device memory significantly reduces the programmability and prohibits the use of traditional programming models. ATI offers two separate programming models at different abstraction levels [15]. The low-level programming model is called Compute Abstraction Layer (CAL). It provides communication primitives and an assembly language, allowing fine tuning of device performance. For high-level programming, ATI provides Brook+. We implemented the correlator with both models. In both cases, the programmer has to do the vectorization, unlike with NVIDIA GPUs. CAL provides a feature called swizzling, which is used to select parts of vector registers in arithmetic operations. We found this improves readability of the code. However, the programming tools still are unsatisfactory. The high-level Brook+ model does not achieve acceptable performance. The low-level CAL model does, but it is difficult to use. The best-performing implementation uses a tile size of 4 × 3, thanks to the large number of registers. Due to the low I/O performance, we achieve only 16% of the theoretical peak.



[FIG5] **Achieved performance on the different platforms: percentages are the fraction of the peak for that architecture.**

NVIDIA GPUs

NVIDIA's programming model is called Compute Unified Device Architecture (CUDA) [16]. CUDA is relatively high level, and achieves good performance. An advantage of NVIDIA hardware, in contrast to ATI, is that the application does not have to do vectorization. This is thanks to the fact that all cores have their own address-generation units. All data parallelism is expressed by using threads. When accessing device memory, it is important to make sure that simultaneous memory accesses by different threads are coalesced into a single memory transaction. In contrast to ATI hardware, NVIDIA GPUs support random write access to device memory. This allows a programming model that is much closer to traditional models, greatly simplifying software development. It is important to use shared memory or the texture cache to enable data reuse. In our case, we use the texture cache to speedup access to the sample data. CUDA provides barrier synchronization between threads within a thread block. We exploit this feature to let the threads that access the same samples run in lock step. This way, we pay a small synchronization overhead, but we can increase the cache hit ratio significantly. We found that this optimization improved performance by a factor of two. This optimization is a good example that shows that, on GPUs, it is important to optimize memory behavior, even at the cost of additional instructions and synchronization overhead.

We also investigated the use of the per-multiprocessor shared memory as an application-managed cache. Others report good results with this approach [26]. However, we found that, for our application, the use of shared memory only led to performance degradation compared to the use of the texture caches.

Registers are a shared resource. Using fewer registers in a kernel allows the use of more concurrent threads, hiding load delays. We found that using a relatively small tile size (3 × 2) and many threads increases performance. The kernel itself, without host-to-device communication achieves 38% of the theoretical peak performance. If we include communication, the performance drops to 32% of the peak. Just like with the ATI

[TABLE 3] STRENGTHS AND WEAKNESSES OF THE DIFFERENT PLATFORMS FOR SIGNAL-PROCESSING APPLICATIONS.

INTEL CORE i7 920	IBM BG/P	ATI 4870	NVIDIA TESLA C1060	STI CELL/BE
+ WELL KNOWN	+ L2 PREFETCH UNIT	+ LARGEST NUMBER OF CORES	+ RANDOM WRITE ACCESS	+ POWER EFFICIENCY
– FEW REGISTERS	+ HIGH MEMORY BANDWIDTH	+ SWIZZLING SUPPORT	+ CUDA IS HIGH LEVEL	+ RANDOM WRITE ACCESS
– NO FMA INSTRUCTION	+ FAST INTERCONNECTS	– LOW PCI-e BANDWIDTH	– LOW PCI-e BANDWIDTH	+ SHUFFLE CAPABILITIES
– LIMITED SHUFFLING	– DOUBLE PRECISION ONLY	– TRANSFER SLOWS DOWN KERNEL		+ EXPLICIT CACHE (PERFORMANCE)
	– EXPENSIVE	– NO RANDOM WRITE ACCESS		– EXPLICIT CACHE (PROGRAMMABILITY)
		– BAD PROGRAMMING SUPPORT		– MULTIPLE PARALLELISM LEVELS

hardware, this is caused by the low PCI-e bandwidth. With NVIDIA hardware, significant performance gains can be achieved by using asynchronous host-to-device I/O.

THE CELL/BE

With the Cell/BE it is important to exploit all levels of parallelism. Applications deal with task and data parallelism across multiple SPEs, vector parallelism inside the SPEs, and multibuffering for asynchronous DMA transfers [7]. Acceptable performance can be achieved by programming the Cell/BE in C or C++, while using intrinsics to manually express vector parallelism. Thus, the programmer specifies which instructions have to be used but can typically leave the instruction scheduling and register allocation to the compiler.

A distinctive property of the architecture is that cache transfers are explicitly managed by the application using DMA. This is unlike other architectures, where caches work transparently. Communication can be overlapped with computation, by using multiple buffers. Although issuing explicit DMA commands complicates programming, we found that this usually is not problematic for signal-processing applications. Thanks to the explicit cache, the correlator implementation fetches each sample from main memory only exactly once. The large number of registers allows a big tile size of 4×3, leading to a lot of data reuse. We exploit the vector parallelism of the Cell/BE by computing the four polarization combinations in parallel. We found that this performs better than vectorizing over the integration time. This is thanks to the Cell/BE's excellent support for shuffling data around in the vector registers. Due to the high memory bandwidth and the ability to reuse data, we achieve 92% of the peak performance on one chip. If we use both chips in a cell blade, we still achieve 91%. Even though the memory bandwidth per operation of the Cell/BE is eight times lower than that of the BG/P, we still achieve excellent performance, thanks to the high-data reuse factor.

COMPARISON AND EVALUATION

Figure 5 shows the performance on all architectures we evaluated. The NVIDIA GPU achieves the highest absolute performance. Nevertheless, the GPU efficiencies are much lower than on the other platforms. The Cell/BE achieves the highest efficiency of all many-core architectures, close to that of the BG/P. Although the theoretical peak performance of the Cell/BE is 4.6 times lower than the NVIDIA chip, the absolute performance is only 1.6 times lower. If both chips in the cell blade are used, the Cell/BE also has the highest absolute performance. For the GPUs, it is possible to use more than one chip as well, for instance with the ATI 4870x2 device. However, we found that this does not help, since the performance is already limited by the low PCI-e throughput, and the chips have to share this resource. In Table 3 we summarize the architectural strengths and weaknesses that we discussed.

PROGRAMMABILITY OF THE PLATFORMS

The performance gap between assembly and a high-level programming language is quite different for the different platforms. It also depends on how much the compiler is helped by manually unrolling loops, eliminating common subexpressions, and the use of register variables up to a level that the C code becomes almost as low level as assembly code. The difference varies between only a few percent to a factor of ten.

For the BG/P, the performance from compiled C++ code was by far not sufficient. The assembly code is approximately ten times faster. For both the Cell/BE and the Intel Core i7, we found that high-level code in C or C++ in combination with the use of intrinsics to manually describe the SIMD parallelism yields acceptable performance compared to optimized assembly code. Thus, the programmer specifies which instructions have to be used, but can typically leave the instruction scheduling and register allocation to the compiler. On NVIDIA hardware, the high-level CUDA model delivers excellent performance, as long as the programmer helps by using SIMD data types for loads and stores, and separate local variables for values that should be kept in registers. With ATI hardware, this is different. We found that the high-level Brook+ model does not achieve acceptable performance compared to hand-written CAL code. Manually written assembly is more than three times faster. Also, the Brook+ documentation is insufficient.

CONCLUSIONS

Radio telescopes require large amounts of signal processing and have high computational and I/O demands. We presented general insights on how to use many-core platforms for signal-processing applications, looking at the aspects of performance, optimization, and programmability. As an example, we evaluated the extremely data-intensive correlator algorithm on today's many-core architectures.

The many-core architectures have a significantly lower memory bandwidth per operation compared to traditional architectures.

This requires completely different algorithm implementation and optimization strategies: minimizing the number of memory loads per operation is of key importance to obtain good performance. A high memory bandwidth per operation is not strictly necessary, as long as the architecture (and the algorithm) allows efficient data reuse. This can be achieved through caches, shared memory, LSs, and registers. It is clear that application-level control of cache behavior (either through explicit DMA or thread synchronization) has a substantial performance benefit, and is of key importance for signal-processing applications.

We demonstrated that the many-core architectures have very different performance characteristics, and require different implementation and optimization strategies. The BG/P supercomputer achieves high efficiencies thanks to the high memory bandwidth per operation. The GPUs are unbalanced: they provide an enormous computational power but have a relatively low bandwidth per operation, both internally and externally (between the host and the device). Because of this, many data-intensive signal-processing applications will achieve only a small fraction of the theoretical peak. The Cell/BE performs excellently on signal-processing applications, even though its memory bandwidth per operation is eight times lower than the BG/P. Applications can exploit the application-managed cache and the large number of registers. For the correlator, this results in optimal reuse of all sample data. Nevertheless, it is clear that, for signal-processing applications, the recent trend of increasing the number of cores will not work indefinitely if I/O is not scaled accordingly.

ACKNOWLEDGMENTS

This work was performed in the context of the NWO STARE AstroStream project. We gratefully acknowledge NVIDIA, and in particular Dr. David Luebke, for freely providing some of the GPU cards used in this work.

AUTHORS

Rob V. van Nieuwpoort (nieuwpoort@astron.nl) is a postdoctoral researcher at the Vrije Universiteit, Amsterdam and ASTRON. His current research interests focus on many-core systems and radio astronomy. He got his Ph.D degree at Vrije Universiteit, Amsterdam on efficient Java-centric grid computing. He has designed and implemented the Manta, Ibis, Satin, and JavaGAT systems and worked on the GridLab and Virtual Labs for E-science projects. At ASTRON, he works on the central, real-time data processing of LOFAR. His research interests include high-performance computing, parallel and distributed algorithms, networks, programming languages, and compiler construction.

John W. Romein (romein@astron.nl) is a senior system researcher of high-performance computing at ASTRON, where he is responsible for the central, real-time data processing of LOFAR. He obtained his Ph.D. degree on distributed search algorithms for board-game playing at Vrije Universiteit, Amsterdam. As a postdoctoral researcher, he solved the game of Awari using a large computer cluster and did research on parallel algorithms for bioinformatics. His research interests include high-performance computing, parallel algorithms, networks, programming languages, and compiler construction.

REFERENCES

[1] M. de Vos, A. W. Gunst, and R. Nijboer, "The LOFAR telescope: System architecture and signal processing," *Proc. IEEE*, vol. 97, no. 8, Aug. 2009, pp. 1431–1437.

[2] S. Johnston, R. Taylor, M. Bailes, N. Bartel, C. Baugh, M. Bietenholz, C. Blake, R. Braun, J. Brown, S. Chatterjee, J. Darling, A. Deller, R. Dodson, P. Edwards, R. Ekers, S. Ellingsen, I. Feain, B. Gaensler, M. Haverkorn, G. Hobbs, A. Hopkins, C. Jackson, C. James, G. Joncas, V. Kaspi, V. Kilborn, B. Koribalski, R. Kothes, T. Landecker, A. Lenc, J. Lovell, J.-P. Macquart, R. Manchester, D. Matthews, N. McClure-Griffiths, R. Norris, U.-L. Pen, C. Phillips, C. Power, R. Protheroe, E. Sadler, B. Schmidt, I. Stairs, L. Staveley-Smith, J. Stil, S. Tingay, A. Tzioumis, M. Walker, J. Wall, and M. Wolleben, "Science with ASKAP. The Australian square-kilometre-array pathfinder," *Exp. Astron.*, vol. 22, no. 3, pp. 151–273, 2008.

[3] W. M. Telford, L. P. Geldart, and R. E. Sheriff, *Applied Geophysics*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 1991.

[4] J. D. Taylor, *Introduction to Ultra-Wideband Radar Systems*. Boca Raton, FL: CRC, 1995.

[5] P. Chandra, A. Bensky, R. Olexa, D. M. Dobkin, D. A. Lide, and F. Dowla, *Wireless Networking*. Burlington, MA: Newnes Press, 2007.

[6] J. W. Romein, P. C. Broekema, J. D. Mol, and R. V. van Nieuwpoort. (2010, Jan.). The LOFAR correlator: Implementation and performance analysis. *Proc. 15th ACM SIGPLAN Symp. Principles and Practice of Parallel Programming (PPoPP 2010)*, Bangalore, India [Online]. Available: http://www.astron.nl/~romein/papers/

[7] M. Gschwind, H. P. Hofstee, B. K. Flachs, M. Hopkins, Y. Watanabe, and T. Yamazaki, "Synergistic processing in cell's multicore architecture," *IEEE Micro*, vol. 26, no. 2, pp. 10–24, 2006.

[8] D. A. Bader and V. Agarwal, "FFTC: Fastest Fourier transform for the IBM cell broadband engine," in *Proc. 14th IEEE Int. Conf. High Performance Computing*, 2007, pp. 172–184.

[9] A. S. van Amesfoort, A. L. Varbanescu, H. J. Sips, and R. V. van Nieuwpoort, "Multi-core platforms for HPC data-intensive kernels," in *Proc. ACM Computing Frontiers*, Ischia, Italy, 2009, pp. 207–216.

[10] R. V. van Nieuwpoort and J. W. Romein, "Using many-core hardware to correlate radio astronomy signals," in *Proc. ACM Int. Conf. Supercomputing*, New York, NY, June 2009, pp. 440–449.

[11] Karoo array telescope (MeerKAT) [Online]. Available: http://www.ska.ac.za/

[12] R. T. Schilizzi, P. E. F. Dewdney, and T. J. W. Lazio, "The square kilometre array," in *Proc. SPIE*, July 2008, vol. 7012.

[13] M. P. van Haarlem. (2005). Lofar: The low frequency array. *Eur. Astron. Soc. Pub. Series* vol. 15, pp. 431–444 [Online]. Available: http://dx.doi.org/10.1051/eas:2005169

[14] IBM Blue Gene team, "Overview of the IBM Blue Gene/P Project," *IBM J. Res. Develop.*, vol. 52, no. 1/2, 2008, pp. 199–220.

[15] Advanced Micro Devices Corp., *AMD Stream Computing User Guide Revision 1.1*. Sunnyvale, CA, Aug. 2008.

[16] NVIDIA Corp., *NVIDIA CUDA Programming Guide Version 2.0*. Santa Clara, CA, July 2008.

[17] F. Catthoor, K. Danckaert, K. K. Kulkarni, E. Brockmeyer, P. G. Kjeldsberg, T. van Achteren, and T. Omnes, *Data Access and Storage Management for Embedded Programmable Processors*. Norwell, MA: Kluwer, 2002.

[18] T.-F. Chen and J.-L. Baer, "Effective hardware-based data prefetching for high-performance processors," *IEEE Trans. Comput.*, vol. 44, no. 5, pp. 609–623, 1995.

[19] U. Meyer, P. Sanders, and J. Sibeyn, Eds., *Algorithms for Memory Hierarchies* (Lecture Notes in Computer Science, vol. 2625). Berlin: Springer-Verlag, 2003.

[20] D. F. Bacon, J.-H. Chow, D.-C. R. Ju, K. Muthukumar, and V. Sarkar, "A compiler framework for restructuring data declarations to enhance cache and TLB effectiveness," in *Proc. 1994 Conf. Centre for Advanced Studies on Collaborative Research*, Toronto, ON, Canada, IBM Press, 1994, pp. 270–282.

[21] P. R. Wilson, M. S. Johnstone, M. Neely, and D. Boles, "Dynamic storage allocation: A survey and critical review," in *Proc. Int. Workshop on Memory Management (Lecture Notes in Computer Science,* vol. 986), Kinross, Scotland. Springer-Verlag, 1995, pp. 1–116.

[22] M. E. Wolf and M. S. Lam, "A data locality optimizing algorithm," in *Proc. ACM SIGPLAN 1991 Conf. Programming Language Design and Implementation (PLDI)*, Toronto, ON, Canada, 1991, pp. 30–44.

[23] P. R. Panda, N. D. Dutt, and A. Nicolau, "Memory data organization for improved cache performance in embedded processor applications," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 2, no. 4, pp. 384–409, 1996.

[24] E. D. Lazowska, J. Zahorjana, G. S. Graham, and K. C. Sevcik, *Quantitative System Performance, Computer System Analysis Using Queueing Network Models*. Englewood Cliffs, NJ: Prentice-Hall, 1984.

[25] S. Williams, A. Waterman, and D. Patterson, "Roofline: An insightful visual performance model for floating-point programs and multicore architectures," *Commun. ACM*, vol. 52, no. 4, pp. 65–76, 2009.

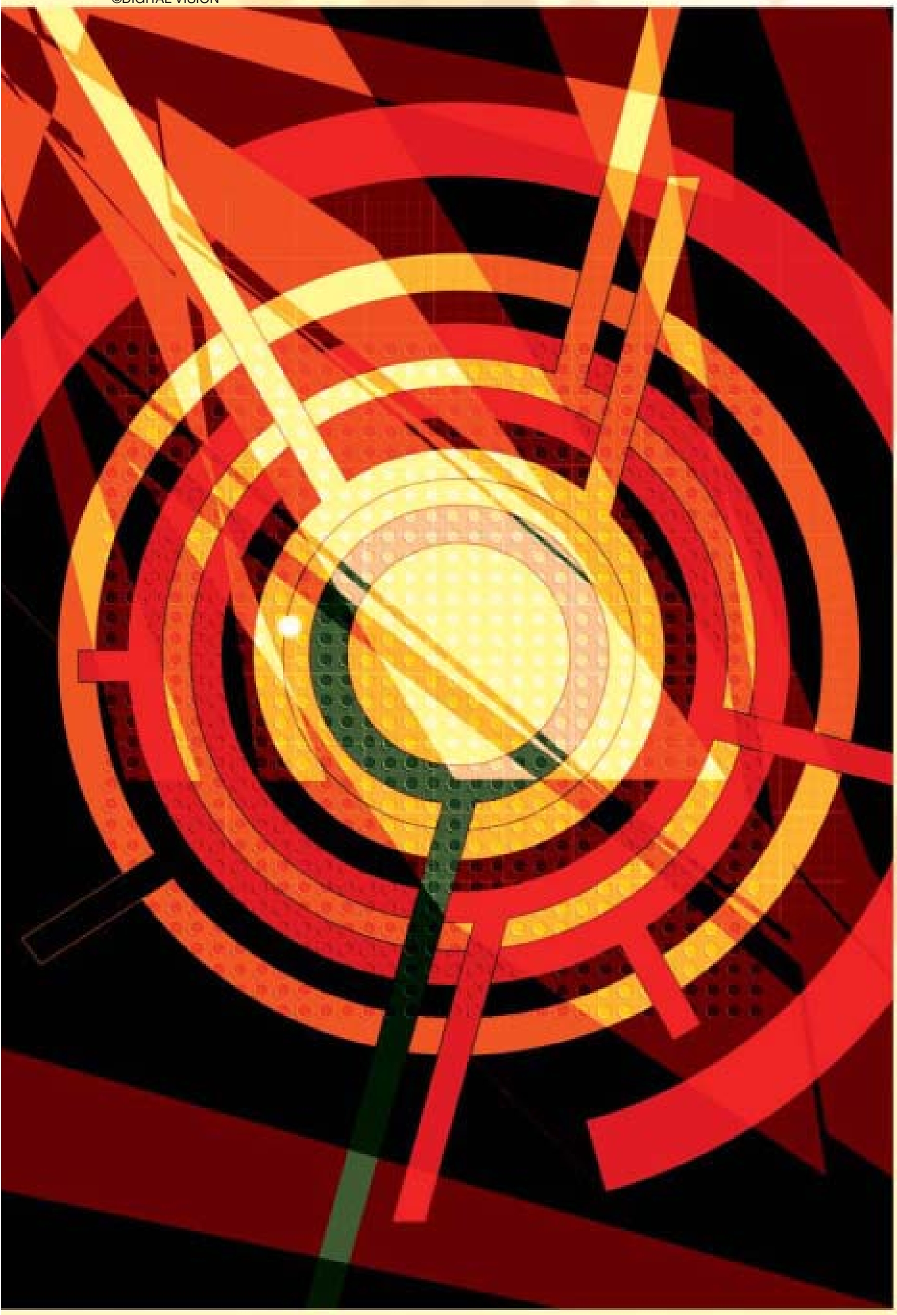
[26] M. Silberstein, A. Schuster, D. Geiger, A. Patney, and J. D. Owens, "Efficient computation of sum-products on GPUs through software-managed cache," in *Proc. 22nd ACM Int. Conf. Supercomputing*, June 2008, pp. 309–318. [SP]

Jianwei Ma and Gerlind Plonka

The Curvelet Transform

A review of recent applications

©DIGITAL VISION



Multiresolution methods are deeply related to image processing, biological and computer vision, and scientific computing. The curvelet transform is a multiscale directional transform that allows an almost optimal nonadaptive sparse representation of objects with edges. It has generated increasing interest in the community of applied mathematics and signal processing over the years. In this article, we present a review on the curvelet transform, including its history beginning from wavelets, its logical relationship to other multiresolution multidirectional methods like contourlets and shearlets, its basic theory and discrete algorithm. Further, we consider recent applications in image/video processing, seismic exploration, fluid mechanics, simulation of partial different equations, and compressed sensing.

INTRODUCTION

Most natural images/signals exhibit line-like edges, i.e., discontinuities across curves (so-called line or curve singularities). Although applications of wavelets have become increasingly popular in scientific and engineering fields, traditional wavelets perform well only at representing point singularities since they ignore the geometric properties of structures and do not exploit the regularity of edges. Therefore, wavelet-based compression, denoising, or structure extraction become computationally inefficient for geometric features with line and surface singularities. In fluid mechanics, discrete wavelet thresholding often leads to oscillations along edges of the coherent eddies and, consequently, to the deterioration of the vortex tube structures, which in turn

Digital Object Identifier 10.1109/MSP.2009.935453

can cause an unphysical leak of energy into neighboring scales producing an artificial "cascade" of energy.

Multiscale methods are also deeply related with biological and computer vision. Since Olshausen and Field's work in *Nature* [55], researchers in biological vision have reiterated the similarity between vision and multiscale image processing. It has been recognized that the receptive fields of simple cells in a mammalian primary visual cortex can be characterized as being spatially localized, oriented, and bandpass (selective to structure at different spatial scales). However, wavelets do not supply a good direction selectivity, which is also an important response property of simple cells and neurons at stages of the visual pathway. Therefore, a directional multiscale sparse coding is desirable in this field.

One of the primary tasks in computer vision is to extract features from an image or a sequence of images. The features can be points, lines, edges, and textures. A given feature is characterized by position, direction, scale, and other property parameters. The most common technique, used in early vision for extraction of such features, is linear filtering, which is also reflected in models used in biological visual systems, i.e., human visual motion sensing. Objects at different scales can arise from distinct physical processes. This leads to the use of scale-space filtering and multiresolution wavelet transform in this field. An important motivation for computer vision is to obtain directional representations that capture anisotropic lines and edges while providing sparse decompositions.

To overcome the missing directional selectivity of conventional two-dimensional (2-D) discrete wavelet transforms (DWTs), a multiresolution geometric analysis (MGA), named curvelet transform, was proposed [7]–[12]. In the 2-D case, the curvelet transform allows an almost optimal sparse representation of objects with singularities along smooth curves. For a smooth object f with discontinuities along C^2-continuous curves, the best N-term approximation $\widetilde{f}_N$ that is a linear combination of only N elements of the curvelet frame obeys $\|f - \widetilde{f}_N\|_2^2 \leq CN^{-2}\,(\log N)^3$, while for wavelets the decay rate is only N^{-1}. Combined with other methods, excellent performance of the curvelet transform has been shown in image processing; see e.g., [49], [45], [60], and [61]. Unlike the isotropic elements of wavelets, the needle-shaped elements of the curvelet transform possess very high directional sensitivity and anisotropy (see Figure 1 for the 2-D case). Such elements are very efficient in representing line-like edges. Recently, the curvelet transform has been extended to three dimensions by Ying et al. [7], [68].

Let us roughly compare the curvelet system with the conventional Fourier and wavelet analysis. The short-time Fourier transform uses a shape-fixed rectangle in frequency domain, and conventional wavelets use shape-changing (dilated) but area-fixed windows. By contrast, the curvelet transform uses angled polar wedges or angled trapezoid windows in frequency domain to resolve directional features.

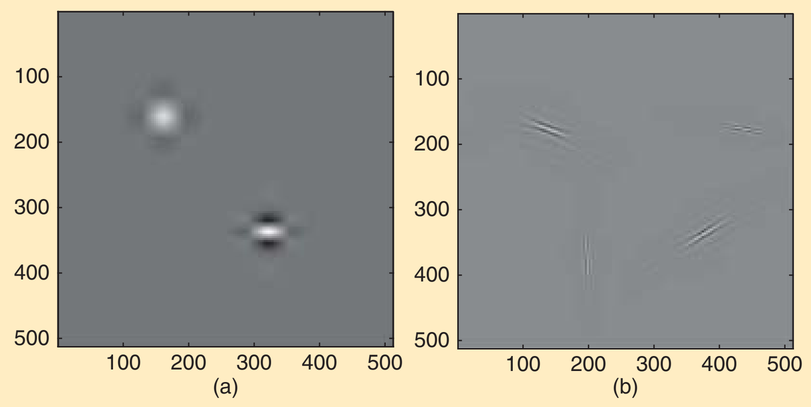
The theoretic concept of curvelets is easy to understand, but how to achieve the discrete algorithms for practical applications is challenging. In this article, we first address a brief history of curvelets starting from classical wavelets. We also mention some other wavelet-like constructions that aim to improve the representation of oriented features towards visual reception and image processing. Then we shall derive the discrete curvelet frame and the corresponding fast algorithm for the discrete curvelet transform in the 2-D case. Finally, we show some recent applications of the discrete curvelet transform in image and seismic processing, fluid mechanics, numerical treatment of partial differential equations, and compressed sensing.

FROM CLASSICAL WAVELETS TO CURVELETS

As outlined in the introduction, although the DWTs has established an impressive reputation as a tool for mathematical analysis and signal processing, it has the disadvantage of poor directionality, which has undermined its usage in many applications. Significant progress in the development of directional wavelets has been made in recent years. The complex wavelet transform is one way to improve directional selectivity. However, the complex wavelet transform has not been widely used in the past, since it is difficult to design complex wavelets with perfect reconstruction properties and good filter characteristics [29], [53]. One popular technique is the dual-tree complex wavelet transform (DT CWT) proposed by Kingsbury [37], [38], which added (almost) perfect reconstruction to the other attractive properties of complex wavelets, including approximate shift invariance, six directional selectivities, limited redundancy and efficient $\mathcal{O}(N)$ computation.

The 2-D complex wavelets are essentially constructed by using tensor-product one-dimensional (1-D) wavelets. The directional selectivity provided by complex wavelets (six directions) is much better than that obtained by the classical DWTs (three directions), but is still limited.

In 1999, an anisotropic geometric wavelet transform, named ridgelet transform, was proposed by Candès and Donoho [4], [8]. The ridgelet transform is optimal at representing straight-line



[FIG1] The elements of (a) wavelets and (b) curvelets on various scales, directions, and translations in the spatial domain. Note that the tensor-product 2-D wavelets are not strictly isotropic but prefer axes directions.

singularities. Unfortunately, global straight-line singularities are rarely observed in real applications. To analyze local line or curve singularities, a natural idea is to consider a partition of the image, and then to apply the ridgelet transform to the obtained subimages. This block ridgelet-based transform, which is named curvelet transform, was first proposed by Candès and Donoho in 2000, see [9]. Apart from the blocking effects, however, the application of this so-called first-generation curvelet transform is limited because the geometry of ridgelets is itself unclear, as they are not true ridge functions in digital images. Later, a considerably simpler second-generation curvelet transform based on a frequency partition technique was proposed by the same authors; see [10]–[12]. Recently, a variant of the second-generation curvelet transform was proposed to handle image boundaries by mirror extension (ME) [19]. Previous versions of the transform treated image boundaries by periodization. Here, the main modifications are to tile the discrete cosine domain instead of the discrete Fourier domain and to adequately reorganize the data. The obtained algorithm has the same computational complexity as the standard curvelet transform.

The second-generation curvelet transform [10]–[12] has been shown to be a very efficient tool for many different applications in image processing, seismic data exploration, fluid mechanics, and solving partial different equations (PDEs). In this survey, we will focus on this successful approach and show its theoretical and numerical aspects as well as the different applications of curvelets. From the mathematical point of view, the strength of the curvelet approach is their ability to formulate strong theorems in approximation and operator theory. The discrete curvelet transform is very efficient in representing curve-like edges. But the current curvelet systems still have two main drawbacks: 1) they are not optimal for sparse approximation of curve features beyond C^2-singularities, and 2) the discrete curvelet transform is highly redundant. The currently available implementations of the discrete curvelet transform (see www.curvelet.org) aim to reduce the redundancy smartly. However, independently from the good theoretical results on N-term approximation by curvelets, the discrete curvelet transform is not appropriate for image compression. The question of how to construct an orthogonal curvelet-like transform is still open.

RELATIONSHIP OF CURVELETS TO OTHER DIRECTIONAL WAVELETS

There have been several other developments of directional wavelet systems in recent years with the same goal, namely a better analysis and an optimal representation of directional features of signals in higher dimensions. None of these approaches has reached the same publicity as the curvelet transform. However, we want to mention shortly some of these developments and also describe their relationship to curvelets.

Steerable wavelets [28], [59], Gabor wavelets [40], wedgelets [23], beamlets [24], bandlets [51], [54], contourlets [21], shearlets [39], [31], wave atoms [20], platelets [67], and surfacelets [42] have been proposed independently to identify and restore geometric features. These geometric wavelets or directional wavelets are uniformly called X-lets.

The steerable wavelets [28], [59] and Gabor wavelets [40] can be seen as early directional wavelets. The steerable wavelets were built based on directional derivative operators (i.e., the second derivative of a Gaussian), while the Gabor wavelets were produced by a Gabor kernel that is a product of an elliptical Gaussian and a complex plane wave. In comparison to separable orthonormal wavelets, the steerable wavelets provide translation-invariant and rotation-invariant representations of the position and the orientation of considered image structures. This feature is paid by high redundancy. Applications of Gabor wavelets focused on image classification and texture analysis. Gabor wavelets have also been used for modeling the receptive field profiles of cortical simple cells. Applications of Gabor wavelets suggested that the precision in resolution achieved through redundancy may be a relevant issue in brain modeling, and that orientation plays a key role in the primary visual cortex. The main differences between steerable wavelets/Gabor wavelets and other X-lets is that the early methods do not allow for a different number of directions at each scale.

Contourlets, as proposed by Do and Vetterli [21], form a discrete filter bank structure that can deal effectively with piecewise smooth images with smooth contours. This discrete transform can be connected to curvelet-like structures in the continuous domain. Hence, the contourlet transform [21] can be seen as a discrete form of a particular curvelet transform. Curvelet constructions require a rotation operation and correspond to a partition of the 2-D frequency plane based on polar coordinates; see the section "The Discrete Curvelet Frame." This property makes the curvelet idea simple in the continuous case but causes problems in the implementation for discrete images. In particular, approaching critical sampling seems difficult in discretized constructions of curvelets. For contourlets, critically sampling is easy to implement. There exists an orthogonal version of the contourlet transform that is faster than current discrete curvelet algorithms [7]. The directional filter bank, as a key component of contourlets, has a convenient tree-structure, where aliasing is allowed to exist and will be canceled by carefully designed filters. Thus, the key difference between contourlets and curvelets is that the contourlet transform is directly defined on digital-friendly discrete rectangular grids. Unfortunately, contourlet functions have less clear directional geometry/features than curvelets (i.e., more oscillations along the needle-like elements) leading to artifacts in denoising and compression.

Surfacelets [42] are 3-D extensions of the 2-D contourlets that are obtained by a higher-dimensional directional filter bank and a multiscale pyramid. They can be used to efficiently capture and represent surface-like singularities in multidimensional volumetric data involving biomedical imaging, seismic imaging, video processing and computer vision. Surfacelets and the 3-D curvelets (see the section "Three-Dimensional Curvelet Transform") aim at the same frequency partitioning, but the two transforms achieve this goal with different approaches as we described above in the 2-D case. The surfacelet transform is less redundant than the 3-D curvelet transform, and this advantage is payed by a certain loss of directional features.

Unlike curvelets, the shearlets [39], [31] form an affine system with a single generating mother shearlet function parameterized by a scaling, a shear, and a translation parameter, where the shear parameter captures the direction of singularities. It has been shown that both the curvelet and shearlet transforms are (at least theoretically) similarly well suited for approximation of piece-wise smooth images with singularities along smooth curves [12], [31]. Indeed, using the fast curvelet transform based on transition to Cartesian arrays, described in the section "Transition to Cartesian Arrays," the discrete implementations of the two transforms are very similar [7].

The bandlet transform [51], [54] is, in contrast with the previously mentioned transforms, based on adaptive techniques and has a good performance for images with textures beyond C^2-singularities, but it has to pay much higher computational cost for its adaptation.

In this article, we are not able to give a more detailed overview on all these approaches and refer to the given references for further information.

THE DISCRETE CURVELET FRAME

In this section, we want to consider the construction of a discrete curvelet frame. Recalling the structure of 1-D wavelets being well localized in frequency domain, we consider the question how these ideas can suitably be transferred to construct a curvelet frame that is an (almost) rotation-invariant function frame in two dimensions. Finally, we summarize the properties of the obtained curvelet elements.

WHAT CAN BE LEARNED FROM THE 1-D WAVELET TRANSFORM?

Using the 1-D dyadic wavelet transform in signal analysis, one considers a family of dilated and translated functions $\{\psi_{j,k} := 2^{j/2}\psi(2^j \cdot - k) : j, k \in \mathbb{Z}\}$, generated by one mother wavelet $\psi \in L^2(\mathbb{R})$, and being an orthonormal basis of $L^2(\mathbb{R})$. Then, each signal $f \in L^2(\mathbb{R})$ can be uniquely represented in a wavelet expansion

$$f = \sum_{j,k} c_{j,k}(f)\, \psi_{j,k},$$

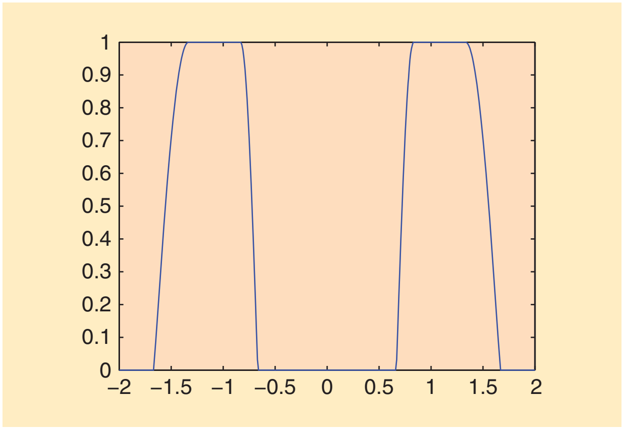
where $c_{j,k}(f) := \langle f, \psi_{j,k}\rangle$ are the wavelet coefficients. Here $\langle \cdot,\cdot \rangle$ denotes the scalar product in $L^2(\mathbb{R})$. Observe that the Fourier transformed elements of the wavelet basis have the form

$$\hat{\psi}_{j,k}(\xi) = 2^{-j/2}\, e^{-i2^{-j}\xi k}\, \hat{\psi}(2^{-j}\xi),$$

i.e., dilation by 2^j in space domain corresponds to dilation by 2^{-j} in frequency domain, and the translation corresponds to a phase shift.

For a good frequency localization of the wavelet basis, the main idea is to construct a wavelet basis that provides a partition of the frequency axis into (almost) disjoint frequency bands (or octaves). Such a partition can be ensured if the Fourier transform of the dyadic wavelet $\hat{\psi}$ has a localized or even compact support and satisfies the admissibility condition

$$\sum_{j=-\infty}^{\infty} |\hat{\psi}(2^{-j}\xi)|^2 = 1, \quad \xi \in \mathbb{R}\ a.e.. \qquad (1)$$



[FIG2] Plot of a Meyer wavelet $\hat{\psi}(\xi)$ in frequency domain.

This admissibility condition also ensures the typical wavelet property $\hat{\psi}(0) = \int_{-\infty}^{\infty}\psi(x)\, dx = 0$.

A particularly good frequency localization is obtained, if $\hat{\psi}$ is compactly supported in $[-2, -1/2] \cup [1/2, 2]$. Such a construction has been used for Meyer wavelets (see Figure 2). Obviously, the dilated Meyer wavelets $\hat{\psi}(2^{-j}\xi)$ generate a tiling of the frequency axis into frequency bands, where $\hat{\psi}(2^{-j}\xi)$ has its support inside the intervals $[-2^{j+1}, -2^{j-1}] \cup [2^{j-1}, 2^{j+1}]$. In this case, for a fixed $\xi \in \mathbb{R}$, at most two wavelet functions in the sum (1) overlap. We remark that the condition (1) implies even more! It ensures that the function family $\{\psi_{j,k} : j, k \in \mathbb{Z}\}$ forms a tight frame of $L^2(\mathbb{R})$ (see e.g., [50, Theorem 5.1]).

Finally, a localization property of the dyadic wavelet transform in space domain is guaranteed if also ψ is localized, i.e., if $\hat{\psi}$ is smooth.

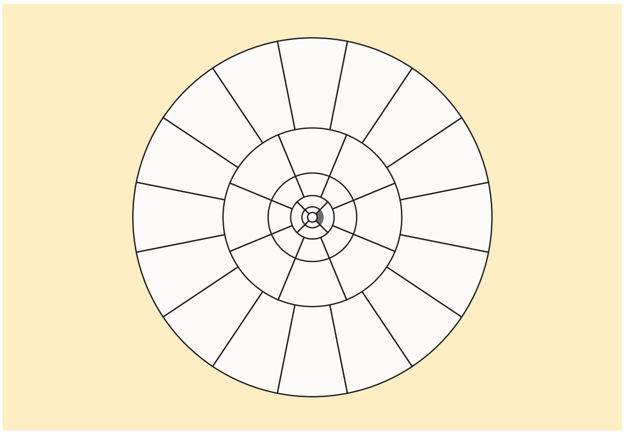
HOW TO TRANSFER THIS IDEA TO THE CURVELET CONSTRUCTION

We wish to transfer this construction principle to the 2-D case for image analysis and incorporate a certain rotation invariance. So, we wish to construct a frame, generated again by one basic element, a basic curvelet ϕ, this time using translations, dilations, and rotations of ϕ. Following the considerations in the 1-D case, the elements of the curvelet family should now provide a tiling of the 2-D frequency space. Therefore the curvelet construction is now based on the following two main ideas [11].

1) Consider polar coordinates in frequency domain.
2) Construct curvelet elements being locally supported near wedges according to Figure 3, where the number of wedges is $N_j = 4 \cdot 2^{\lceil j/2 \rceil}$ at the scale 2^{-j}, i.e., it doubles in each second circular ring. (Here $\lceil x \rceil$ denotes the smallest integer being greater than or equal to x.)

Let now $\boldsymbol{\xi} = (\xi_1, \xi_2)^T$ be the variable in frequency domain. Further, let $r = \sqrt{\xi_1^2 + \xi_2^2}$, $\omega = \arctan \xi_1/\xi_2$ be the polar coordinates in frequency domain. For the "dilated basic curvelets" in polar coordinates we use the ansatz

$$\hat{\phi}_{j,0,0}(r,\omega) := 2^{-3j/4}\, W(2^{-j}r)\tilde{V}_{N_j}(\omega), \quad r \ge 0,\ \omega \in [0, 2\pi), j \in N_0, \qquad (2)$$



[FIG3] **Tiling of the frequency domain into wedges for curvelet construction.**

where we use suitable window functions W and $\widetilde{V}_{N_j}$, and where a rotation of $\hat{\phi}_{j,0,0}$ corresponds to the translation of a 2π-periodic window function $\widetilde{V}_{N_j}$. The index N_j indicates the number of wedges in the circular ring at scale 2^{-j} (see Figure 3). To construct a (dilated) basic curvelet with compact support near a "basic wedge" (see e.g., the gray wedge in Figure 3 for $j = 0$), the two windows W and $\widetilde{V}_{N_j}$ need to have compact support. The idea is to take $W(r)$ similarly as in the 1-D case, to cover the interval $(0, \infty)$ with dilated curvelets, and to take $\widetilde{V}_{N_j}$ such that a covering in each circular ring is ensured by translations of $\widetilde{V}_{N_j}$. Here, we have $r \in [0, \infty)$, therefore we cannot take the complete Meyer wavelet to determine W, but only the part that is supported in $[1/2, 2]$ (see Figure 2). Then the admissibility condition (1) yields

$$\sum_{j=-\infty}^{\infty} |W(2^{-j}r)|^2 = 1 \qquad (3)$$

for $r \in (0, \infty)$. For an explicit construction of W, see "Window Functions."

Further, for the tiling of a circular ring into N wedges, where N is an arbitrary positive integer, we need a 2π-periodic nonnegative window $\widetilde{V}_N$ with support inside $[-2\pi/N, 2\pi/N]$ such that

$$\sum_{l=0}^{N-1} \widetilde{V}_N^2\left(\omega - \frac{2\pi l}{N}\right) = 1 \quad \text{for all } \omega \in [0, 2\pi)$$

is satisfied. Then only two "neighbored" translates of $\widetilde{V}_N^2$ in the sum overlap. Such windows $\widetilde{V}_N$ can be simply constructed as 2π-periodizations of a scaled window $V(N\omega/2\pi)$, where V is given in "Window Functions."

In this way we approach the goal to get a set of curvelet functions with compact support in frequency domain on wedges, where in the circular ring that corresponds to the scale 2^{-j} the sum of the squared rotated curvelet functions depends only on $W(2^{-j}r)$, i.e., it follows that

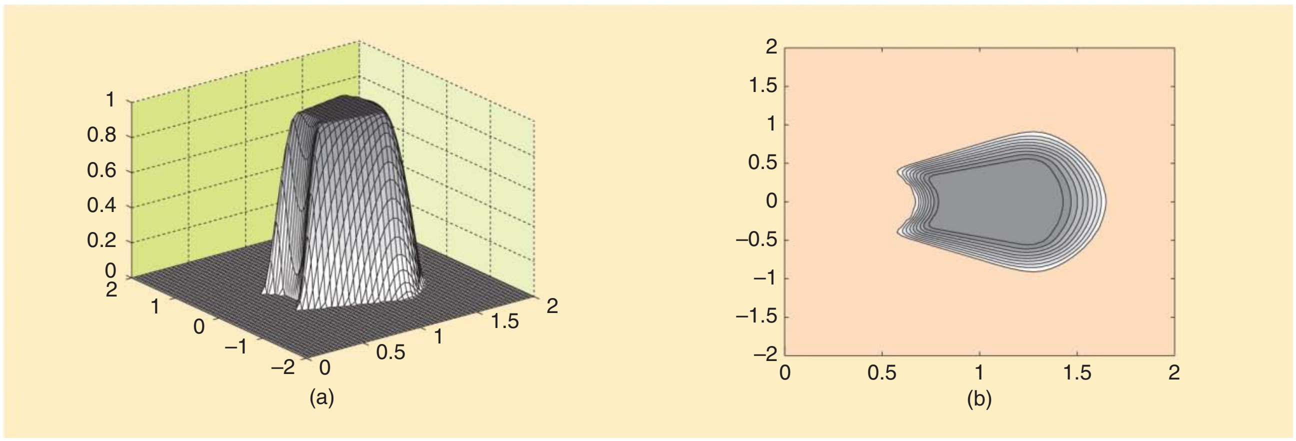
$$\sum_{l=0}^{N_j-1} \left| 2^{3j/4}\, \hat{\phi}_{j,0,0}\left(r, \omega - \frac{2\pi l}{N_j}\right)\right|^2 = |W(2^{-j}r)|^2 \sum_{l=0}^{N_j-1} \widetilde{V}_{N_j}^2\left(\omega - \frac{2\pi l}{N_j}\right)$$
$$= |W(2^{-j}r)|^2. \qquad (4)$$

Together with (3), that means that using the rotations of the dilated basic curvelets $\hat{\phi}_{j,0,0}$, we are able to guarantee an admissibility condition similar to (1) for the 1-D wavelet frame. Remember that the translates of $\phi_{j,0,0}$ have no influence here, since translates in space domain correspond to phase shifts in frequency domain that do not change the support of the Fourier transformed curvelets. The basic curvelet $\hat{\phi}_{j,0,0,0}$ is illustrated in Figure 4.

There is a last point we have to attend to, namely the "hole" that arises in the frequency plane around zero, since the rotations of the dilated basic curvelets work only in the scales 2^{-j} for $j = 0, 1, 2, \ldots$. Taking now all scaled and rotated curvelet elements together with $N_j := 4 \cdot 2^{\lceil j/2 \rceil}$ we find for the scales 2^{-j}, $j = 0, 1, \ldots$ with (3) and (4)

$$\sum_{j=0}^{\infty} \sum_{l=0}^{N_j-1} \left| 2^{3j/4}\, \hat{\phi}_{j,0,0}\left(r, \omega - \frac{2\pi l}{N_j}\right)\right|^2 = \sum_{j=0}^{\infty} |W(2^{-j}r)|^2,$$

and this sum is only for $r > 1$ equal to one. For a complete covering of the frequency plane, we therefore need to define a low-pass element



[FIG4] **Basic curvelet $\hat{\phi}_{0,0,0}$ in the (a) frequency domain and (b) its support.**

WINDOW FUNCTIONS

For constructing the curvelet functions we shall use the following special window functions. Let us consider the scaled Meyer windows (see [18, p. 137])

$$V(\omega) = \begin{cases} 1 & |\omega| \le 1/3, \\ \cos\left[\frac{\pi}{2}\,\nu(3|\omega| - 1\right] & 1/3 \le |\omega| \le 2/3, \\ 0 & \text{else,} \end{cases}$$

$$W(r) = \begin{cases} \cos\left[\frac{\pi}{2}\nu(5-6r)\right] & 2/3 \le r \le 5/6, \\ 1 & 5/6 \le r \le 4/3, \\ \cos\left[\frac{\pi}{2}\nu(3r-4)\right] & 4/3 \le r \le 5/3, \\ 0 & \text{else,} \end{cases}$$

where ν is a smooth function satisfying

$$\nu(x) = \begin{cases} 0 & x \le 0, \\ 1 & x \ge 1, \end{cases} \qquad \nu(x) + \nu(1-x) = 1, \quad x \in \mathbb{R}.$$

For the simple case $\nu(x) = x$ in $[0, 1]$, the window functions $V(\omega)$ and $W(r)$ are plotted in Figure S1. To obtain smoother functions W and V, we need to take smoother functions ν. We may use the polynomials $\nu(x) = 3x^2 - 2x^3$ or $\nu(x) = 5x^3 - 5x^4 + x^5$ in $[0, 1]$, such that ν is in $C^1(\mathbb{R})$ or in $C^2(\mathbb{R})$. An example of an arbitrarily smooth window ν is given by

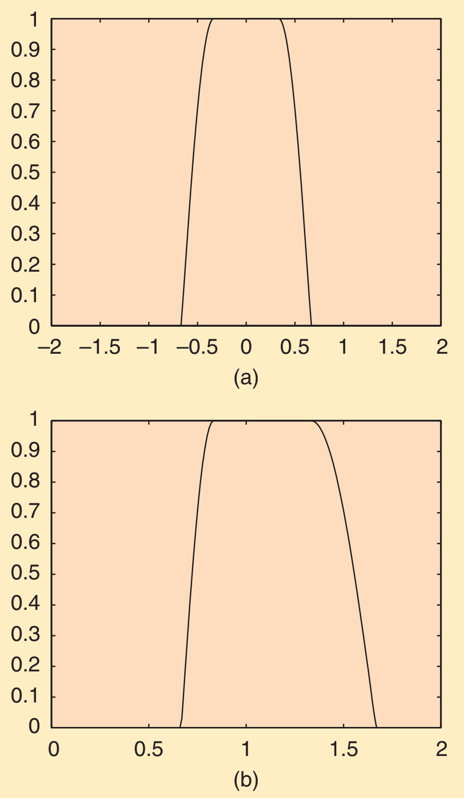
$$\nu(x) = \begin{cases} 0 & x \le 0, \\ \dfrac{s(x-1)}{s(x-1)+s(x)} & 0 < x < 1, \\ 1 & x \ge 1 \end{cases}$$

$$\text{with } s(x) = e^{-\left(\frac{1}{(1+x)^2} + \frac{1}{(1-x)^2}\right)}.$$

The above two functions $V(t)$ and $W(r)$ satisfy the conditions

$$\sum_{l=-\infty}^{\infty} V^2(\omega - l) = 1, \quad t \in \mathbb{R}, \tag{S1}$$

$$\sum_{j=-\infty}^{\infty} W^2(2^j r) = 1, \quad r > 0. \tag{S2}$$



[FIGS1] Plot of the (a) windows $V(\omega)$ and (b) $W(r)$.

In particular, the 2π periodic window functions $\tilde{V}_N(\omega)$ needed for curvelet construction, can now be obtained as 2π-periodization of $V(N\omega/2\pi)$.

$$\hat{\phi}_{-1}(\xi) := W_0(|\xi|) \quad \text{with} \quad W_0^2(r)^2 := 1 - \sum_{j=0}^{\infty} W(2^{-j} r)^2 \tag{5}$$

that is supported on the unit circle, and where we do not consider any rotation.

HOW MANY WEDGES SHOULD BE TAKEN IN ONE CIRCULAR RING?

As we have seen already in Figure 3 and postulated in the last subsection, for the curvelet construction, there are $N_j = 4 \cdot 2^{\lceil j/2 \rceil}$ angles (or wedges) chosen in the circular ring (with radius $2^{j-1/2} \le r \le 2^{j+1/2}$) corresponding to the 2^{-j}th scale, see [11]. But looking at the above idea to ensure the admissibility condition for a tight frame, one is almost free to choose the number of wedges/angles in each scale. Principally, the construction works for all ratios of angles and scales. In fact this is an important point, where curvelets differ from other constructions.

1) If we take the number of wedges in a fixed way, independent of the scale, we essentially obtain steerable wavelets.

2) If the number of wedges increases like 1/scale (i.e., like 2^j), then we obtain tight frames of ridgelets.

3) If the number of wedges increases like $\sqrt{1/\text{scale}}$ (i.e., like $2^{j/2}$), the curvelet frame is obtained. This special anisotropic scaling law yields the typical curvelet elements whose properties are considered next.

WHAT PROPERTIES DO THE CURVELET ELEMENTS HAVE?

To obtain the complete curvelet family, we need to consider the rotations and the translations of the dilated basic curvelets $\phi_{j,0,0}$. We choose

- an equidistant sequence of rotation angles $\theta_{j,l}$,

$$\theta_{j,l} := \frac{\pi l\, 2^{-\lceil j/2 \rceil}}{2} \quad \text{with } l = 0, 1, \ldots, N_j - 1$$

- the positions $\mathbf{b}_k^{j,l} = \mathbf{b}_{k_1,k_2}^{j,l} := \mathbf{R}_{\theta_{j,l}}^{-1}\left((k_1/2^j)(k_2/2^{j/2})\right)^T$ with $k_1, k_2 \in \mathbb{Z}$, and where $\mathbf{R}_\theta$ denotes the rotation matrix with angle θ.

Then the family of curvelet functions is given by

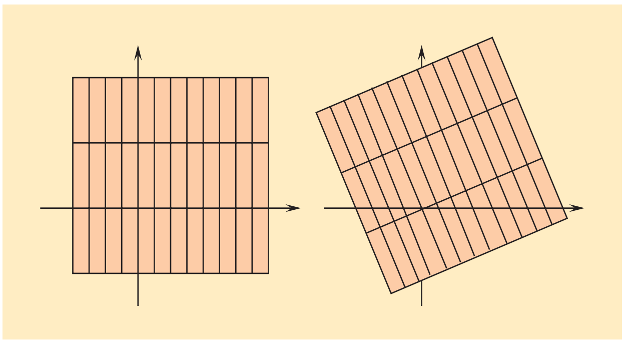
$$\phi_{j,\mathbf{k},l}(\mathbf{x}) := \phi_{j,0,0}(\mathbf{R}_{\theta_{j,l}}(\mathbf{x} - \mathbf{b}_{\mathbf{k}}^{j,l})) \tag{6}$$

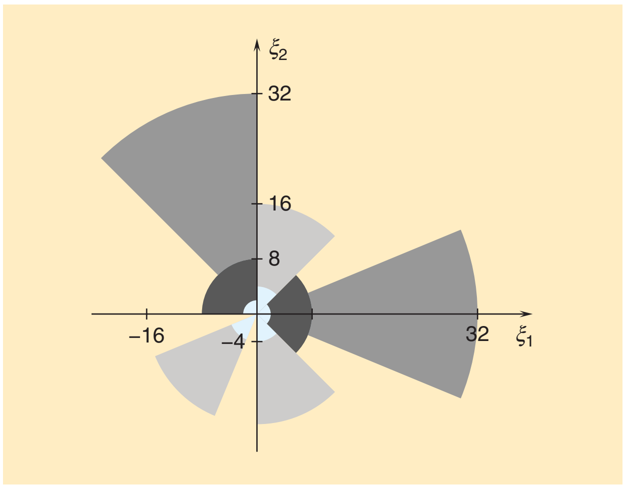
with indices $j \in \mathbb{N}_0$ and $\mathbf{k} = (k_1, k_2)$, l as above.

One should note that the positions $\mathbf{b}_{\mathbf{k}}^{j,l}$ are on different regular grids for each different rotation angle, and these grids have different spacing in the two directions being consistent with the parabolic scaling (i.e., with the ratio of angles and scales). This choice will lead to a discrete curvelet system that forms a tight frame, i.e., every function $f \in L^2(\mathbb{R})$ will be representable by a curvelet series, and hence the discrete curvelet transform will be invertible.

For example, for $j = 0$ we consider the angles $\theta_{0,l} = \pi l/2$, $l = 0, 1, 2, 3$ and the positions $\{\mathbf{b}_{\mathbf{k}}^{0,l}\}_{\mathbf{k}\in\mathbb{Z}^2, l=0,1,2,3} = \mathbb{Z}^2$. For $j = 4$, the angles $\theta_{4,l} = \pi l/8$, $l = 0, \ldots, 15$ occur, and, depending on the angles $\theta_{4,l}$, eight different grids for translation are considered, where rectangles of size $1/16 \times 1/4$ are rotated by $\theta_{4,l}$, $l = 0, \ldots, 7$, see Figure 5. In particular, the choice of positions yields a parabolic scaling of the grids with the relationship length $\approx 2^{-j/2}$ and width $\approx 2^{-j}$.

The underlying idea for the choice of the translation grids is as follows. Considering a band-limited function f, where $\hat{f}$ has its support on a single wedge (e.g., in the scale 2^{-j}; see Figure 6), one can determine a rotation angle and a translation to map this wedge into the center of the frequency plane, then find a rectangle of size $2^j \times 2^{j/2}$ to cover the wedge, and finally use the Shannon sampling theorem to fix the needed sampling rate for covering f. All sampling rates that are obtained in this way have to be taken, and thus one finds the needed positions as above.


[FIG5] Grid for $\theta_{4,0} = 0$ and for $\theta_{4,1} = \pi/8$.



[FIG6] Maximal supports of $\hat{\phi}_{2,\mathbf{k},0}$ and $\hat{\phi}_{2,\mathbf{k},5}$ (dark grey); of $\hat{\phi}_{3,\mathbf{k},3}$ $\hat{\phi}_{3,\mathbf{k},6}$ and $\hat{\phi}_{3,\mathbf{k},13}$ (light grey); and of $\hat{\phi}_{4,\mathbf{k},0}$ and $\hat{\phi}_{4,\mathbf{k},11}$ (grey). The translation index $k \in \mathbb{Z}^2$ does not influence the support of the curvelet elements.

SUPPORT IN FREQUENCY DOMAIN

In frequency domain, the curvelet function $\hat{\phi}_{j,\mathbf{k},l}$ is supported inside the polar wedge with radius $2^{j-1} \le r \le 2^{j+1}$ and angle $2^{-\lfloor j/2 \rfloor}\pi(-1-l)/2 < \omega < 2^{-\lfloor j/2 \rfloor}\pi(1-l)/2$. The support of $\hat{\phi}_{j,\mathbf{k},l}$ does not depend on the position $\mathbf{b}_{\mathbf{k}}^{j,l}$. For example, $\hat{\phi}_{2,\mathbf{k},l}(r,\omega)$ is supported inside the wedge with $2 \le r \le 8$ and $(-1-l)\pi/4 \le \omega \le (1-l)\pi/4$, $l = 0, \ldots, 7$; see Figure 6. (Here we have used supp $\tilde{V}_{N_j} \subset [-2\pi/N_j, 2\pi/N_j]$ and supp $W \subset [1/2, 2]$.)

SUPPORT IN TIME DOMAIN AND OSCILLATION PROPERTIES

In time domain, things are more involved. Since $\hat{\phi}_{j,\mathbf{k},l}$ has compact support, the curvelet function $\phi_{j,\mathbf{k},l}$ cannot have compact support in time domain. From Fourier analysis, one knows that the decay of $\phi_{j,\mathbf{k},l}(\mathbf{x})$ for large $|\mathbf{x}|$ depends on the smoothness of $\hat{\phi}_{j,\mathbf{k},l}$ in frequency domain. The smoother $\hat{\phi}_{j,\mathbf{k},l}$, the faster the decay.

By definition, $\hat{\phi}_{j,0,0}(\boldsymbol{\xi})$, $j \in \mathbb{N}_0$, is supported away from the vertical axis $\xi_1 = 0$ but near the horizontal axis $\xi_2 = 0$; see Figure 6. Hence, for large $j \in \mathbb{N}_0$ the function $\phi_{j,0,0}(\mathbf{x})$ is less oscillatory in x_2-direction (with frequency about $2^{-j/2}$) and very oscillatory in x_1-direction (containing frequencies of about 2^{j-1}). The essential support of the amplitude spectrum of $\phi_{j,0,0}$ is a rectangle of size $[-\pi 2^{j-1}, \pi 2^{j-1}] \times [-\pi 2^{j/2}, \pi 2^{j/2}]$, and the decay of $\phi_{j,0,0}$ away from this rectangle essentially depends on the smoothness of $\hat{\phi}$ respectively, the windows V and W. From (6), we simply observe that the essential support of $\phi_{j,\mathbf{k},l}$ is the rectangle rotated by the angle $\theta_{j,l}$ and translated by $\mathbf{R}_{\theta_{j,l}}\mathbf{b}_{\mathbf{k}}^{j,l}$.

Remark

The concept "essential support" of a function f with good decay properties is used in literature without rigorous definition but with the following intuitive meaning: the essential support is a finite region that contains the most important features of the function. Outside this support, the graph of f consists mainly of asymptotic tails that can be neglected in certain considerations.

TIGHT FRAME PROPERTY

The system of curvelets

$$\{\phi_{-1,\mathbf{k},0} : \mathbf{k} \in \mathbb{Z}^2\} \cup \{\phi_{j,\mathbf{k},l} : j \in \mathbb{N}_0,$$
$$l = 0, \ldots, 4 \cdot 2^{\lceil j/2 \rceil} - 1, \mathbf{k} = (k_1, k_2)^T \in \mathbb{Z}^2\}$$

satisfies a tight frame property. That means, every function $f \in L^2(\mathbb{R}^2)$ can be represented as a curvelet series

$$f = \sum_{j,\mathbf{k},l} \langle f, \phi_{j,\mathbf{k},l} \rangle\, \phi_{j,\mathbf{k},l}, \tag{7}$$

and the Parseval identity

$$\sum_{j,\mathrm{k},l} |\langle f, \phi_{j,\mathrm{k},l}\rangle|^2 = \|f\|^2_{L^2(\mathbb{R}^2)}, \quad \forall f \in L^2(\mathbb{R}^2)$$

holds. For a proof, we refer to [11]. The terms $c_{j,\mathrm{k},l}(f) := \langle f, \phi_{j,\mathrm{k},l}\rangle$ are called curvelet coefficients. In particular, we obtain by Plancherel's Theorem for $j \geq 0$

$$c_{j,\mathrm{k},l}(f) := \int_{\mathbb{R}^2} f(\mathrm{x})\, \overline{\phi_{j,\mathrm{k},l}(\mathrm{x})}\, d\mathrm{x} = \int_{\mathbb{R}^2} \hat{f}(\boldsymbol{\xi})\, \overline{\hat{\phi}_{j,\mathrm{k},l}(\boldsymbol{\xi})}\, d\boldsymbol{\xi}$$
$$= \int_{\mathbb{R}^2} \hat{f}(\boldsymbol{\xi})\, \hat{\phi}_{j,0,0}(\mathrm{R}_{\theta_{j,l}}\boldsymbol{\xi})\, e^{i\langle \mathrm{b}_{\mathrm{k}}^{j,l}, \boldsymbol{\xi}\rangle}\, d\boldsymbol{\xi}. \qquad (8)$$

THE FAST CURVELET TRANSFORM

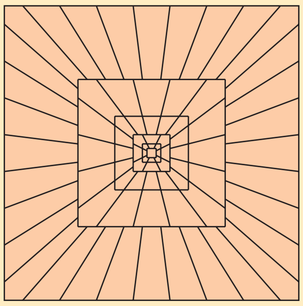
TRANSITION TO CARTESIAN ARRAYS

In practical implementations, one would like to have Cartesian arrays instead of the polar tiling of the frequency plane. Cartesian coronae are based on concentric squares (instead of circles) and shears (see Figure 7). Therefore, a construction of window functions on trapezoids instead of polar wedges is desirable. Hence, we need to adapt the discrete curvelet system as given in the section "What Properties Do the Curvelet Elements Have?" suitably. Let us remark that the frequency tiling into shears, as given in Figure 7, has been similarly used for the construction of contourlets [21] by a pyramidal directional filter bank. However, the tiling for the contourlet transform is slightly more flexible by allowing that the number of directions need not to be doubled at each second scale, see [21].

For the transition of the basic curvelet according to the new tiling, where rotation is replaced by shearing, we use the ansatz

$$\hat{\tilde{\phi}}_{j,0,0}(\boldsymbol{\xi}) := 2^{-3j/4} W(2^{-j}\xi_1)\, V\!\left(\frac{2^{\lfloor j/2\rfloor}\xi_2}{\xi_1}\right)$$

with the window function W as in the section "How to Transfer This Idea to the Curvelet Construction" and with a nonnegative window V with compact support in $[-2/3, 2/3]$; see "Window Functions." This adapted scaled basic curvelet $\hat{\tilde{\phi}}_{j,0,0}$ in Figure 8 is the Cartesian equivalent to $\hat{\phi}_{j,0,0}$ in (2) (see Figure 4).



[FIG7] Discrete curvelet tiling with parabolic pseudopolar support in the frequency plane.

Observe that the support of $V_j(\xi) := V(2^{\lfloor j/2\rfloor}\xi_2/\xi_1)$ is now inside the cone $K_1 := \{(\xi_1, \xi_2): \xi_1 > 0, \xi_2 \in [-2\xi_1/3, 2\xi_1/3]\}$. Hence the adapted basic curvelet $\tilde{\phi}_{j,0,0}$ determines the frequencies in the trapezoid

$$\left\{(\xi_1,\xi_2): 2^{j-1} \leq \xi_1 \leq 2^{j+1},\ -2^{-\lfloor j/2\rfloor}\cdot\frac{2}{3} \leq \xi_2/\xi_1 \leq 2^{-\lfloor j/2\rfloor}\cdot\frac{2}{3}\right\}.$$

To replace rotation of curvelet elements by shearing in the new grid, we need to consider the eastern, western, northern, and southern cone separately (see Figure 9 for the eastern cone). Let us only consider the shearing in the eastern cone $K = \{(\xi_1, \xi_2)^T: \xi_1 > 0,\ -\xi_1 < \xi_2 \leq \xi_1\}$, for the other cones, suitable curvelet elements are then obtained by rotation by $\pm\,\pi/2$ radians and reflection.

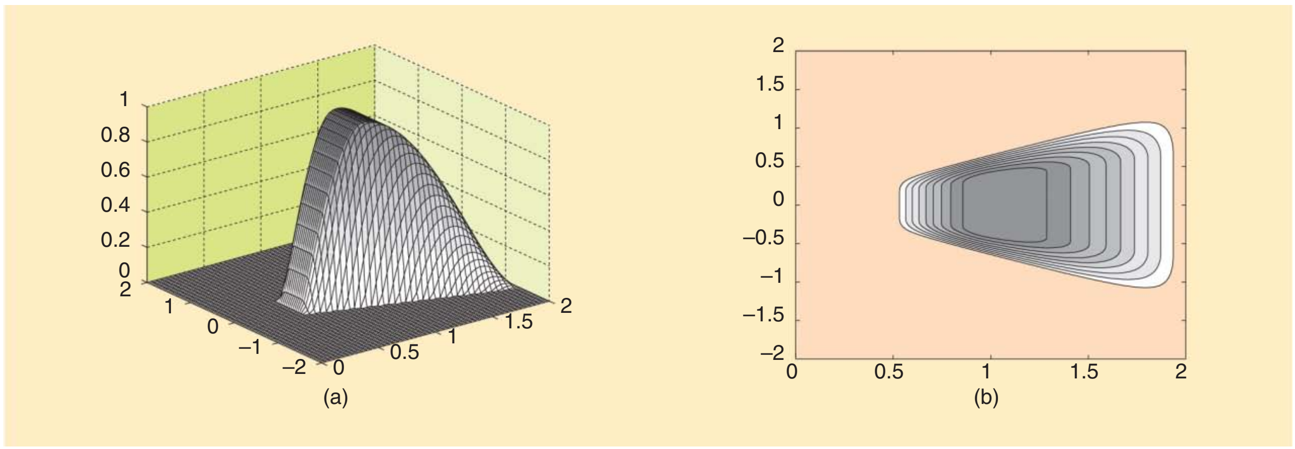
Instead of equidistant angles, we define a set of equispaced slopes in the eastern cone

$$\tan\theta_{j,l} := l\, 2^{-\lfloor j/2\rfloor}, \quad l = -2^{\lfloor j/2\rfloor}+1, \ldots, 2^{\lfloor j/2\rfloor}-1.$$

Observe that the angles $\theta_{j,l}$, which range between $-\pi/4$ and $\pi/4$, are not equispaced here, while the slopes are.

Now, let the curvelet-like functions be given by

$$\tilde{\phi}_{j,\mathrm{k},l}(\mathrm{x}) := \tilde{\phi}_{j,0,0}(\mathrm{S}^T_{\theta_{j,l}}(\mathrm{x} - \tilde{\mathrm{b}}_{\mathrm{k}}^{j,l})), \qquad (9)$$



[FIG8] (a) Basic curvelet $\hat{\tilde{\phi}}_{0,0,0}$ and (b) its support adapted to the Cartesian arrays in frequency domain.

being the Cartesian counterpart of $\phi_{j,\mathrm{k},l}$ in (6), with the shear matrix

$$\mathrm{S}_\theta = \begin{pmatrix} 1 & 0 \\ -\tan\theta & 1 \end{pmatrix},$$

and where $\tilde{\mathrm{b}}_{\mathrm{k}}^{j,l} := \mathrm{S}_{\theta_{j,l}}^{-T}(k_1 2^{-j}, k_2 2^{-\lfloor j/2 \rfloor}) =: \mathrm{S}_{\theta_{j,l}}^{-T}\mathrm{k}_j$ denotes the position of $\tilde{\phi}_{j,\mathrm{k},l}$ in space domain. Let us have a closer look at the functions $\tilde{\phi}_{j,\mathrm{k},l}$. The Fourier transform gives by $\mathrm{S}_{\theta_{j,l}}^{-1}\boldsymbol{\xi} = (\xi_1, \xi_1 \tan\theta_{j,l} + \xi_2)^T$

$$\begin{aligned}\hat{\tilde{\phi}}_{j,\mathrm{k},l}(\xi) &= e^{-i\langle \tilde{\mathrm{b}}_{\mathrm{k}}^{j,l}, \xi\rangle} \hat{\tilde{\phi}}_{j,0,0}(\mathrm{S}_{\theta_{j,l}}^{-1}\boldsymbol{\xi}) \\ &= e^{-i\langle \tilde{\mathrm{b}}_{\mathrm{k}}^{j,l}, \xi\rangle} 2^{-3j/4} W(2^{-j}\xi_1)\, V(2^{\lfloor j/2 \rfloor}\xi_2/\xi_1 + l).\end{aligned}$$

Hence, $\hat{\tilde{\phi}}_{j,\mathrm{k},l}$ is compactly supported on sheared trapezoids.

Let us for example examine $\hat{\tilde{\phi}}_{4,\mathrm{k},l}$. For $j = 4$, we consider the angles $\tan\theta_{4,l} = l/4$, $l = -3, \ldots, 3$. The support of $\hat{\tilde{\phi}}_{j,\mathrm{k},0}$ is symmetric with respect to the ξ_1 axis, and for $j = 4$ we have

$$\operatorname{supp} \hat{\tilde{\phi}}_{4,\mathrm{k},0} = \left\{(\xi_1, \xi_2)^T: 8 \le \xi_1 \le 32, -\frac{1}{6} \le \frac{\xi_1}{\xi_2} \le \frac{1}{6}\right\}.$$

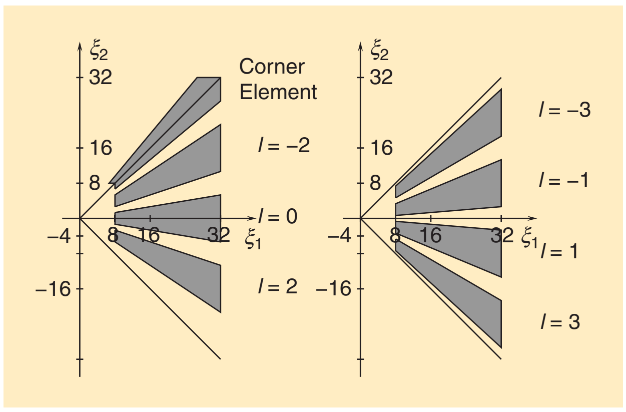
The supports of $\hat{\tilde{\phi}}_{4,\mathrm{k},l}$ with $l = -3, \ldots, 3$ in the eastern cone are now sheared versions of this trapezoid (see Figure 9).

The set of curvelets $\tilde{\phi}_{j,\mathrm{k},l}$ in (9) needs to be completed by symmetry and by rotation by $\pm\pi/2$ radians to obtain the whole family. Moreover, as we can also see in Figure 9, we need suitable "corner elements" connecting the four cones (north, west, south, and east). In [7], it is suggested to take a corner element as the sum of two half-part sheared curvelet functions of neighboring cones as indicated in Figure 9 (on the left).

Finally, the coarse curvelet elements for low frequencies are needed, and we take here

$$\tilde{\phi}_{-1,\mathrm{k},0}(\mathrm{x}) := \tilde{\phi}_{-1}(\mathrm{x} - \mathrm{k}), \quad \mathrm{k} \in \mathbb{Z}^2,$$

where $\hat{\tilde{\phi}}_{-1}(\boldsymbol{\xi}) := W_0(\xi_1)W_0(\xi_2)$ [with W_0 in (5)] has its support in $[-1, 1]^2$. For this construction of curvelet-like elements one can show the frequency tiling property



[FIG9] Supports of the functions $\hat{\tilde{\phi}}_{4,\mathrm{k},l}$ for $l = -3, \ldots, 3$, and one corner element.

$$\hat{\tilde{\phi}}_{-1}(\xi) + \sum_{j=0}^{\infty} \sum_{l=-2^{\lfloor j/2 \rfloor}}^{2^{\lfloor j/2 \rfloor}} 2^{3j/4}\, \hat{\tilde{\phi}}_{j,0,l}(\xi) = 1$$

for all $\boldsymbol{\xi}$ in the eastern cone $K = \{\boldsymbol{\xi} = (\xi_1, \xi_2)^T: \xi_1 > 0, \xi_2 \in [-\xi_1, \xi_1]\}$, where we have taken also the two corner elements in the sum. Similarly, this assertion is true for the rotated functions in the other three cones.

THE ALGORITHM

We find the Cartesian counterpart of the coefficients in (8) by

$$\begin{aligned}\tilde{c}_{j,\mathrm{k},l}(f) = \langle f, \tilde{\phi}_{j,\mathrm{k},l}\rangle &= \int_{\mathbb{R}^2} \hat{f}(\boldsymbol{\xi})\, \hat{\tilde{\phi}}_{j,0,0}(\mathrm{S}_{\theta_{j,l}}^{-1}\boldsymbol{\xi})\, e^{i\langle \tilde{\mathrm{b}}_{\mathrm{k}}^{j,l}, \boldsymbol{\xi}\rangle} d\boldsymbol{\xi} \\ &= \int_{\mathbb{R}^2} \hat{f}(\mathrm{S}_{\theta_{j,l}}\xi)\, \hat{\tilde{\phi}}_{j,0,0}(\xi)\, e^{i\langle \mathrm{k}_j, \xi\rangle} d\xi \qquad (10)\end{aligned}$$

with $\mathrm{k}_j = (k_1 2^{-j}, k_2 2^{-\lfloor j/2 \rfloor})^T$, $(k_1, k_2)^T \in \mathbb{Z}^2$.

The forward and the inverse fast discrete curvelet transform as presented in [7] have a computational cost of $\mathcal{O}(N^2 \log N)$ for an $(N \times N)$ image, see e.g., CurveLab (http://curvelab.org) with a collection of MATLAB and C++ programs. The redundancy of that curvelet transform implementation is about 2.8 when wavelets are chosen at the finest scale, and 7.2 otherwise (see e.g., [7]); see the "Forward Algorithm," which uses formula (10).

For the inverse curvelet transform, one applies the algorithm in each step in reversed order. Observe that in the second step, a suitable approximation scheme has to be applied in the forward transform and in the inverse transform.

THREE-DIMENSIONAL CURVELET TRANSFORM

For three-dimensional (3-D) data, a generalization to 3-D multiscale geometric methods is of great interest. So far, only a few papers have been concerned with applications of the 3-D

[FORWARD ALGORITHM]

1) Compute the Fourier transform of f by means of a 2-D FFT.
Let f be given by its samples $f((n_1/N),(n_2/N))$ $n_1, n_2 = 0, \ldots, N-1$, where N is of the form $N = 2^J$, $J \in N$. Suppose, that f can be approximated by a linear combination of bivariate hat functions. Let $\tilde{s}(x) = s(x_1)\, s(x_2)$ with $s(x_1) := (1 - |x_1|)\, \chi_{[-1,1]}(x_1)$ and

$$f(x) = \sum_{n_1=0}^{N-1}\sum_{n_2=0}^{N-1} f\left(\frac{n_1}{N}, \frac{n_2}{N}\right) \tilde{s}(Nx_1 - n_1, Nx_2 - n_2).$$

With $\hat{\tilde{s}}(\xi) = (\operatorname{sinc} \xi_1/2)^2 (\operatorname{sinc} \xi_2/2)^2$ it follows that

$$\hat{f}(\xi) = \sum_{n_1=0}^{N-1}\sum_{n_2=0}^{N-1} f\left(\frac{n_1}{N}, \frac{n_2}{N}\right) e^{-i(n_1\xi_1 + n_2\xi_2)/N}\, \hat{\tilde{s}}\left(\frac{\xi}{N}\right),$$

and the 2-D FFT of length N gives us the samples $\hat{f}(2\pi n_1, 2\pi n_2)$, $n_1, n_2 = -N/2, \ldots, (N/2) - 1$.

2) Compute $\hat{f}(\mathrm{S}_{\theta_{j,l}}\xi)$ by interpolation.
Fix the scales to be considered, say $j_0 \le j \le J$. The support of $\hat{\tilde{\phi}}_{j,0,0}$ is contained in the rectangle $R_j = [2^{j-1}, 2^{j+1}] \times [-2^{\lfloor j/2 \rfloor}, 2^{\lfloor j/2 \rfloor}]$. For each pair (j, l) compute now $\hat{f}(2\pi n_1, 2\pi n_2 - 2\pi n_1 \tan\theta_{j,l})$ for $2\pi(n_1, n_2) \in R_j$.

3) Compute the product $\hat{f}(\mathrm{S}_{\theta_{j,l}}\boldsymbol{\xi})\, \hat{\tilde{\phi}}_{j,0,0}(\boldsymbol{\xi})$.
For each pair (j, l) compute the product $\hat{f}(2\pi n_1, 2\pi n_2 - 2\pi n_1 \tan\theta_{j,l})\, \hat{\tilde{\phi}}_{j,0,0}(2\pi n_1, 2\pi n_2)$.

4) Apply the inverse 2-D FFT to obtain the discrete coefficients $\tilde{c}_{j,\mathrm{k},l}^{D}(f)$ that are an approximation of the coefficients in (10).

curvelet transform to 3-D turbulence [2], [47] and 3-D seismic processing [52].

The idea of the 3-D curvelet transform on Cartesian arrays can be carried out analogously as done in the section "Transition to Cartesian Arrays" for the 2-D case. This time, one considers curvelet functions being supported on sheared truncated pyramids instead of sheared trapezoids. The 3-D curvelet functions then depend on four indices instead of three; the scale, the position and two angles; and for the shearing process, one can introduce 3-D shear matrices.

A fast algorithm can be derived similarly as in the section "The Algorithm" for the 2-D case. The computational complexity of the 3-D discrete curvelet transform based on FFT algorithms is $\mathcal{O}(n^3 \log n)$ flops for $n \times n \times n$ data [7]. For further details, we refer to [7] and [68].

RECENT APPLICATIONS

In this section, we shall review applications of the curvelets in image processing, seismic exploration, fluid mechanics, solving of PDEs, and compressed sensing, to show their potential as an alternative to wavelet transforms in some scenarios.

IMAGE PROCESSING

In 2002, the first-generation curvelet transform was applied for the first time to image denoising by Starck et al. [60], and by Candès and Guo [13]. The applications of the first-generation curvelets were extended to image contrast enhancement [62] and astronomical image representation [61] in 2003, and to fusion of satellite images [17] in 2005. After the effective second-generation curvelet transform [12] had been proposed in 2004, the applications of curvelets increased quickly in many fields involving image/video presentation, denoising, and classification. For instance, Ma et al. applied the second-generation curvelets for motion estimation and video tracking of geophysical flows [45] and deblurring [43]. Ma and Plonka presented two different models for image denoising by combining the discrete curvelet transform with nonlinear diffusion schemes. In the first model [49], a curvelet shrinkage is applied to the noisy data, and the result is further processed by a projected total variation diffusion to suppress pseudo-Gibbs artifacts. In the second model [56], a nonlinear reaction-diffusion equation is applied, where curvelet shrinkage is used for regularization of the diffusion process. Starck et al. [63], [3] applied curvelets for morphological component analysis. Recently, B. Zhang et al. [69] used curvelets for Poisson noise removal in comparison with wavelets and ridgelets. In [70], C. Zhang et al. successfully applied the multiscale curvelet transform to multipurpose watermarking for content authentication and copyright verification. Jiang et al. [36] considered structure and texture image in painting with the help of an iterative curvelet thresholding method. Tessens et al. [66] proposed a new context adaptive image denoising by modeling of curvelet domain statistics. By performing an intersubband statistical analysis of curvelet coefficients, one can distinguish between two classes of coefficients: those that represent useful image content, and those dominated by noise. Using a prior model based on marginal statistics, an appropriate local spatial activity indicator for curvelets has been developed that is found to be very useful for image denoising, see [66]. Geback et al. [30] applied the curvelets for edge detection in microscopy images.

Interestingly, the pure discrete curvelet transform is less suitable for image compression and for image denoising. The reason may be the redundancy of the curvelet frame. Most successful approaches related with the discrete curvelet transform are hybrid methods, where curvelets are combined with another technique for image processing. These methods usually can exploit the ability of the curvelet transform to represent curve-like features.

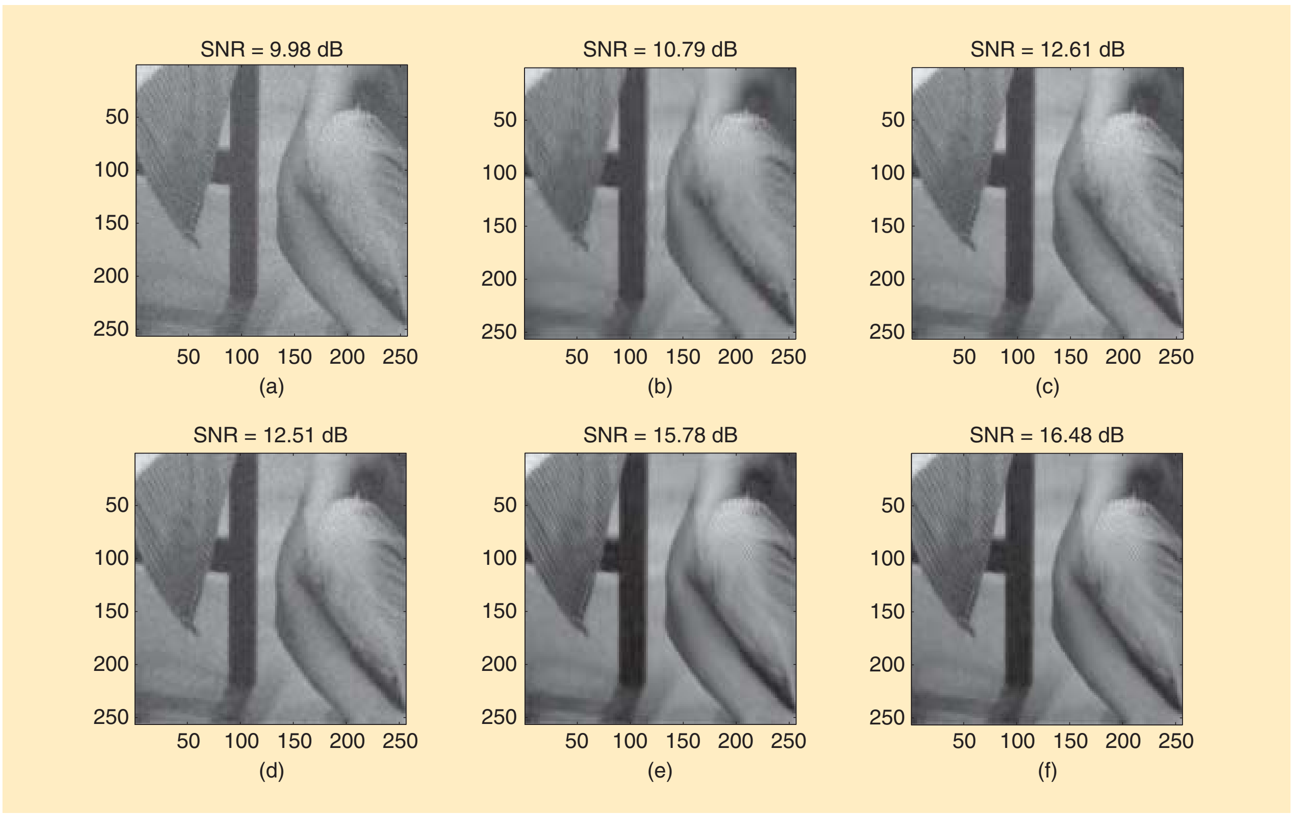
Let us give one example of image denoising [49], where curvelet shrinkage is combined with nonlinear anisotropic diffusion. Figure 10(a) shows a part of noisy Barbara image. Figure 10(b)–(f) present the denoising results by using tensor-product Daubechies's DB4 wavelets, TV diffusion, contourlets, curvelets, and TV-combined curvelet transform [49], respectively. The curvelet-based methods preserve the edges and textures well.

SEISMIC EXPLORATION

Seismic data records the amplitudes of transient/reflecting waves during receiving time. The amplitude function of time is called seismic trace. A seismic data or profile is the collection of these traces. All the traces together provide a spatio-temporal sampling of the reflected wave field containing different arrivals that respond to different interactions of the incident wave field with inhomogeneities in Earth's subsurface. Common denominators among these arrivals are wave fronts (as shown in Figure 11(a) for a real seismic profile), which display anisotropic line-like features, as edges and textures in images. They basically show behaviors of C^2-continuous curves. The main characteristic of the wave fronts is their relative smoothness in the direction along the fronts and their oscillatory behavior in the normal direction. A crucial problem in seismic processing is to preserve the smoothness along the wave fronts when one aims to remove noise. From a geophysical point of view, curvelets can be seen as local plane waves. They are optimal to sparsely represent the local seismic events and can be effectively used for wave front-preserving seismic processing. Therefore, the curvelet decomposition is an appropriate tool for seismic data processing.

Figure 11 shows a denoising of a real seismic data set by curvelets, in comparison to wavelets. Five decomposing levels are used in both transforms. Figure 12 shows the comparison of subband reconstruction in the first three levels; from coarse scale to fine scale. It can be seen clearly that the curvelets perform much better than wavelets to preserve the wave fronts/textures in multiscale decomposition and denoising. We also observe that the curvelet transform can achieve an almost complete data reconstruction if used without any thresholding for coefficients (reconstructed signal-to-noise ratio (SNR) = 310.47 and error = 2.9770e-010).

So far, curvelets have been applied successfully in seismic processing. Hennenfent and Herrmann [32] suggested a nonuniformly sampled curvelet transform for seismic denoising. Neelamani et al. [52] proposed a 3-D curvelet-based effective approach to attenuate random and coherent noise in a 3-D data



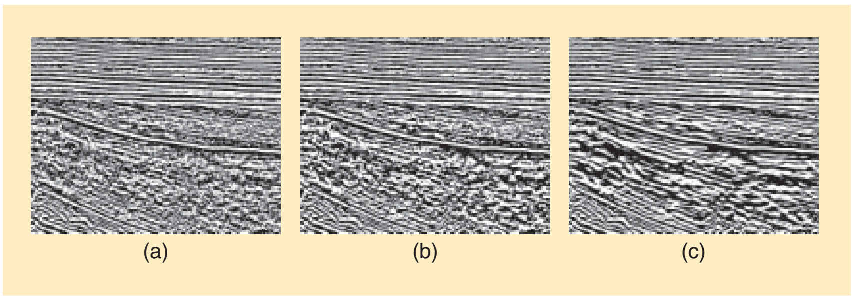
[FIG10] Image denoising: (a) noisy image, (b) wavelet denoising, (c) TV-diffusion denoising, (d) contourlet denoising, (e) curvelet denoising, and (f) TV-combined curvelet denoising.

set from a carbonate environment. Comparisons of wavelets, contourlets, curvelets, and their combination for denoising of random noise have been also investigated in [58]. Douma and de Hoop [25] presented a leading-order seismic imaging by curvelets. They show that using curvelets as building blocks of seismic data, the Kirchhoff diffraction stack can (to leading order in angular frequency, horizontal wave number, and migrated location) be rewritten as a map migration of coordinates of the curvelets in the data, combined with an amplitude correction. This map migration uses the local slopes provided by the curvelet decomposition of the data. Chauris and Nguyen [16] considered seismic demigration/migration in the curvelet domain. The migration consists of three steps: decomposition of the input seismic data (e.g., common offset sections) using the curvelet transform; independent migration of the curvelet coefficients; and inverse curvelet transform to obtain the final depth migrated image. Currently, they concentrate on a ray-based type of prestack depth-migration (i.e., common-offset Kirchhoff depth migration) with respect to heterogeneous velocity models. It turns out that curvelets are almost invariant under the migration operations. The final objective is to be able to derive a formulation and build an efficient algorithm for the full waveform inversion in the curvelet domain.

In addition, curvelet-based primary-multiple separation [35], extrapolation [41], and seismic data recovery [34], [33], [65] have been also proposed by Herrmann et al.

TURBULENCE ANALYSIS IN FLUID MECHANICS

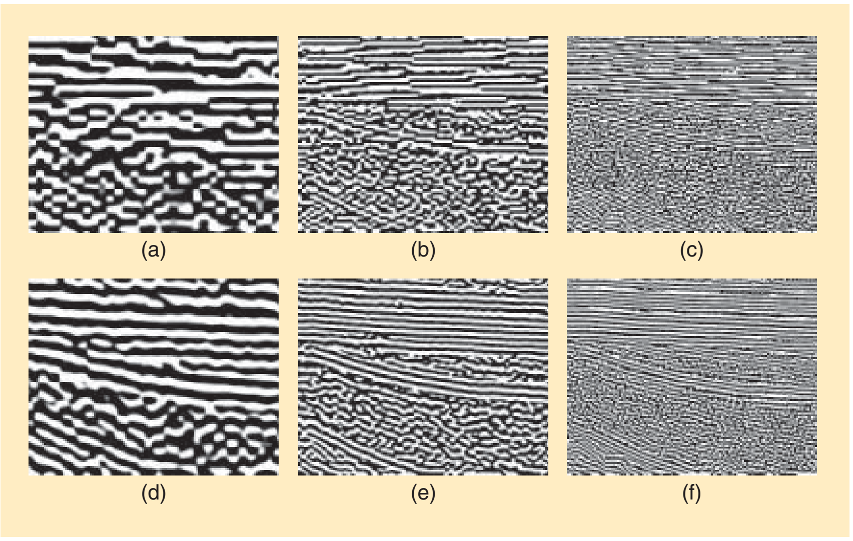
Turbulence has been a source of fascination for centuries because most fluid flows occurring in nature, as well as in engineering applications, are turbulent. Fluid turbulence is a paradigm of multiscale phenomena, where the coherent structures evolve in an incoherent random background. Turbulence is difficult to approximate and analyze mathematically or to calculate numerically because of its range of spatial and temporal scales. The geometrical representation of flow structures might seem to be restricted to a well-defined set of curves along which the data are singular. As a consequence, the efficient compression of a flow field with minimum loss of



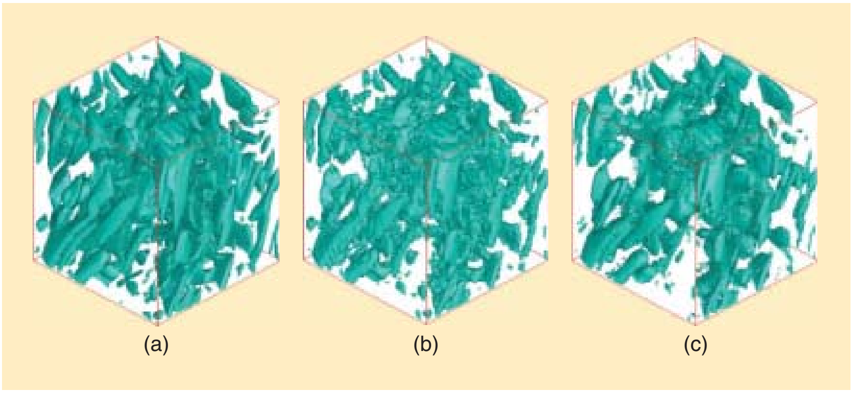
[FIG11] Comparison of seismic denoising: (a) original data, (b) wavelet denoising, and (c) curvelet denoising.

the geometric flow structures is a crucial problem in the simulation of turbulence. The development of appropriate tools to study vortex breakdown, vortex reconnection, and turbulent entrainment at laminar-turbulent interfaces, is imperative to enhance our understanding of turbulence. Such tools must capture the vortical structure and dynamics accurately to unravel the physical mechanisms underlying these phenomena.

Recently, the curvelets have been applied to study the nonlocal geometry of eddy structures and the extraction of the coherent vortex field in turbulent flows [2], [47], [48]. Curvelets start to influence the field of turbulence analysis and have the potential to upstage the wavelet representation of turbulent flows addressed in [26] and [27]. The multiscale geometric property, implemented by means of curvelets, provides the framework for studying the evolution of the structures associated to the main ranges of scales defined in Fourier space, while keeping the localization in physical space that enables a geometrical study of such structures. Such a geometrical characterization can provide a better understanding of cascade mechanics and dissipation-range dynamics. Moreover, curvelets have the potential to contribute to the development of structure-based models of turbulence fine scales, subgrid-scale models for large-eddy simulation, and simulation methods based on prior wavelet transforms [2].



[FIG12] Comparisons of subband reconstruction in the first three levels from coarse scale to fine scale by (a)–(c) wavelet transform and (d)–(f) curvelet transform.



[FIG13] Extraction of coherent fields from turbulent flows: (a) original flow, (b) coherent components by wavelets, and (c) curvelets.

Figure 13 gives an example of the extraction of coherent fields from turbulent flows. The curvelet method preserves the edges and structures better than wavelet methods. The results of multiscale turbulence analysis depend on the threshold or shrinkage. The question of how to find the optimal threshold to separate coherent fields and incoherent random fields still remains open.

SOLVING OF PDEs

Candès and Demanet [5], [6] have shown that curvelets essentially provide optimally sparse representations of Fourier integral operators. While the wavelet transform is optimal for solving elliptical PDEs, the motivation to use the curvelet transform is that for a wide class of linear hyperbolic differential equations, the curvelet representation of the solution operator is both optimally sparse and well organized. Sparsity means that the matrix entries decay nearly exponentially fast, and they are well organized in the sense that very few nonnegligible entries occur near a few shifted diagonals. Wave fronts of solutions can be also sparsely represented in curvelet domain [6]. Some updated results for hyperbolic evolution equations with limited smoothness have been obtained by Andersson et al. [1]. The key idea of the existing methods is first to decompose the initial fields by the curvelet transform, and then to compute the rigid motions of the significant curvelet coefficients along Hamiltonian ray flows at each scale. Finally, one needs to reconstruct the evolution coefficients at all scales by an inverse curvelet transform and obtains an approximate wave field $u(x,t)$ at a given time t. The theory is quite elegant but still far away from practical applications. The papers cited above show the potential of curvelets for solving of PDEs from the point of view of mathematical analysis and raise the hope to achieve fast algorithms for the solution of hyperbolic PDEs using curvelets.

Let us consider a wave equation with the associated Cauchy initial value problem

$$\frac{\partial^2 u}{\partial t^2}(\mathrm{x},t) = v^2\,\Delta u(\mathrm{x},t) \quad u(\mathrm{x},0) = u_0(\mathrm{x}),\ \ \frac{\partial u}{\partial t}(\mathrm{x},0) = u_1(\mathrm{x}). \tag{11}$$

For simplicity, assume that v is a constant wave speed, and $\Delta u(\mathbf{x}, t) = \partial^2/x_1^2\, u(\mathbf{x}, t) + (\partial^2)/(x_2^2) u(\mathbf{x}, t)$ denotes the usual Laplace operator. Its solution can be written as $u(\mathbf{x}, t) = F(\mathbf{x}, t) u_0(\mathbf{x}) + G(\mathbf{x}, t) u_1(\mathbf{x})$, with suitable solution operators $F(\mathbf{x}, t)$ and $G(\mathbf{x}, t)$ (involving Green's functions) that can be sparsely represented in curvelet domain.

Curvelet-based finite difference schemes for seismic wave equations have been studied in [64]. The goal is to construct a fast adaptive scheme for numerical modeling of wave propagation. Similarly as with prior wavelet-based finite difference schemes, one crucial problem is to explore how the differential operator Δ (or ∂_{x_i}) can be computed by the curvelet transform in an efficient way. The 2-D wave field u can be transformed into curvelet domain by $u(x_1, x_2, t) = \sum_{\boldsymbol{\mu}} c_{\boldsymbol{\mu}}(t) \phi_{\boldsymbol{\mu}}(x_1, x_2)$. Here, we have used the tight frame property (7) with the short notation $\boldsymbol{\mu} = (j, \mathbf{k}, l)$, and $c_{\boldsymbol{\mu}}(t)$ denotes the μth curvelet coefficient of u at time t. A possible way to compute the curvelet coefficients of Δu is

$$c_{\boldsymbol{\mu}}^{\triangle} := c_{\boldsymbol{\mu}}(\Delta u) := \int \Delta u(\mathbf{x}, t)\, \overline{\phi_{\boldsymbol{\mu}}(\mathbf{x})}\, d\mathbf{x} = \int \widehat{\Delta u}(\boldsymbol{\xi}, t)\, \overline{\hat{\phi}_{\boldsymbol{\mu}}(\boldsymbol{\xi})}\, d\boldsymbol{\xi}$$
$$= \int (-\xi_1^2 - \xi_2^2)\, \hat{u}(\boldsymbol{\xi}, t)\, \overline{\hat{\phi}_{\boldsymbol{\mu}}(\boldsymbol{\xi})}\, d\boldsymbol{\xi}.$$

Using the definition of the curvelet coefficients in (10), we obtain with $\mathrm{S}_{\theta_{j,l}} \boldsymbol{\xi} = (\xi_1, \; -\xi_1 \tan\theta_{j,l} + \xi_2)^T$

$$c_{\boldsymbol{\mu}}^{\triangle} = \int [\,-(1 + \tan^2\theta_{j,l})\, \xi_1^2 - \xi_2^2 + 2(\tan\theta_{j,l})\, \xi_1 \xi_2\,]$$
$$\times\, \hat{u}(\mathrm{S}_{\theta_{j,l}} \boldsymbol{\xi})\, \overline{\hat{\phi}_{j,0,0}(\boldsymbol{\xi})}\, e^{i\langle \mathbf{k}_j, \boldsymbol{\xi}\rangle} d\boldsymbol{\xi}$$
$$= 4^j (1 + \tan^2\theta_{j,l}) \frac{\partial^2 c_{\boldsymbol{\mu}}}{\partial k_1^2} + 4^{\lfloor j/2 \rfloor} \frac{\partial^2 c_{\boldsymbol{\mu}}}{\partial k_2^2} - 2^{j+1}\, 2^{\lfloor j/2 \rfloor} \tan\theta_{j,l} \frac{\partial^2 c_{\boldsymbol{\mu}}}{\partial k_1 \partial k_2}.$$

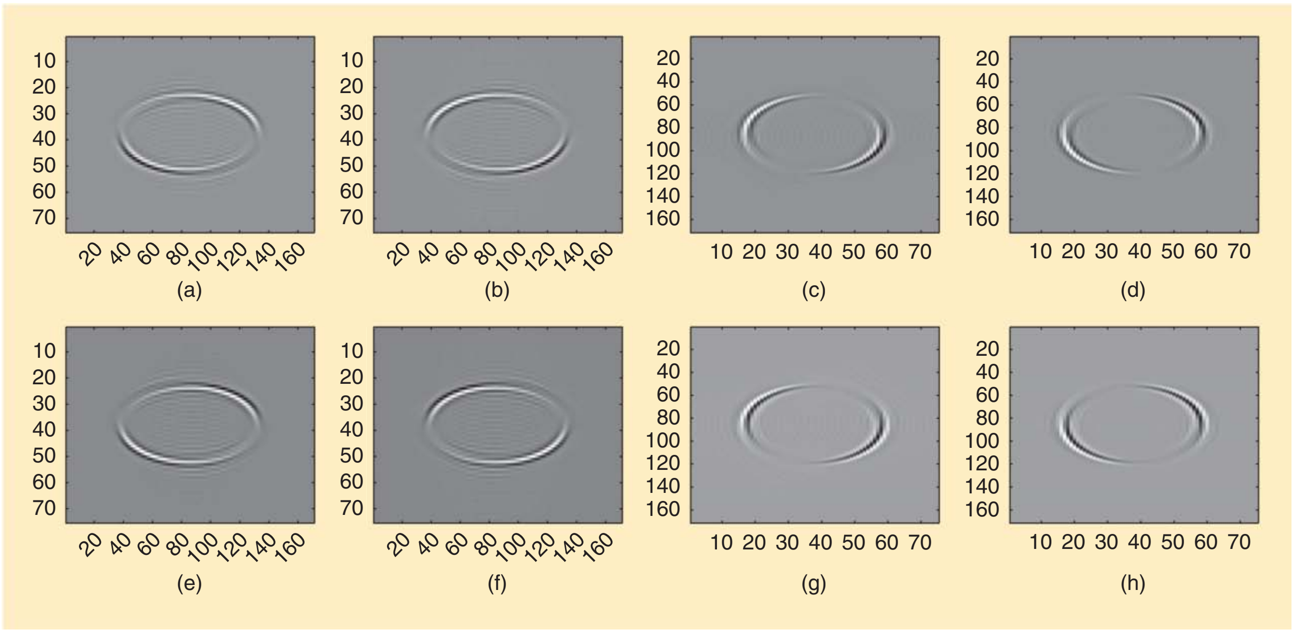
Here we recall that $\mathbf{k} = (k_1, k_2)^T \in \mathbb{Z}^2$ and $\mathbf{k}_j = (k_1/2^j, k_2/2^{\lfloor j/2 \rfloor})^T$. That means, we can obtain the curvelet coefficients of Δu by using the coefficients of the instant wave field u. Thus, we can rewrite the wave equation in coefficient domain by

$$\frac{\partial^2 c_{\boldsymbol{\mu}}}{\partial t^2} = v^2 \Bigg(4^j (1 + \tan^2\theta_{j,l}) \frac{\partial^2 c_{\boldsymbol{\mu}}}{\partial k_1^2} + 4^{\lfloor j/2 \rfloor} \frac{\partial^2 c_{\boldsymbol{\mu}}}{\partial k_2^2} - 2^{j+1}\, 2^{\lfloor j/2 \rfloor} \tan\theta_{j,l} \frac{\partial^2 c_{\boldsymbol{\mu}}}{\partial k_1 \partial k_2} \Bigg). \tag{12}$$

Figure 14 shows an example of curvelet coefficients of an instant wave field at the coarsest curvelet detail scale, by implementing the computation in curvelet domain as given in (12). For details of this approach we refer to [64]. Using suitable thresholding, one can implement a fast adaptive computation for the wave propagation. Unfortunately, due to the redundancy of the current discrete curvelet algorithm, the curvelets have not performed at the level that we expected. The matrices are not as sparse as the estimates promise. The efficient numerical treatment of PDEs using curvelets is still a challenging problem.

COMPRESSED SENSING

Finally, we mention a new direction of applications of the curvelet transform to the so-called compressed sensing or compressive sampling (CS), an inverse problem with highly incomplete measurements. CS [14], [15], [22] is a novel sampling paradigm that carries imaging and compression simultaneously. The CS theory says that a compressible unknown signal can be recovered by a small number of random measurements using sparsity-promoting nonlinear recovery algorithms. The number of necessary measurements is considerably smaller than the number of needed traditional measurements that satisfy the Shannon/Nyquist sampling theorem, where the sampling rate has to be at least twice as large as the maximum frequency of the signal. The CS-based data acquisition depends on its sparsity



[FIG14] Curvelet coefficients of an instant wave field at the coarsest curvelet detail scale. (a)–(h) denotes eight different directional subbands in this curvelet scale.

rather than its bandwidth. CS might have an important impact for designing of measurement devices in various engineering fields such as medical magnetic resonance (MRI) imaging and remote sensing, especially for cases involving incomplete and inaccurate measurements limited by physical constraints, or very expensive data acquisition.

Mathematically, we handle the fundamental problem of recovering a signal $\mathbf{x} \in \mathbb{R}^N$ from a small set of measurements $\mathbf{y} \in \mathbb{R}^K$. Let $\mathbf{A} \in \mathbb{R}^{K\times N}$ be the so-called CS measurement matrix, where $K \ll N$, i.e., there are much fewer rows in the matrix than columns. The measurements can be described as [14]

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \boldsymbol{\epsilon}. \tag{13}$$

Here $\boldsymbol{\epsilon}$ denotes possible measurement errors or noise. It seems to be hopeless to solve this ill-posed underdetermined linear system since the number of equations is much smaller than the number of unknown variables. However, if the $\mathbf{x}$ is compressible by a transform, as e.g., $\mathbf{x} = \mathbf{T}^{-1}\mathbf{c}$, where $\mathbf{T}$ denotes the discrete curvelet transform, and the sequence of discrete curvelet coefficients $\mathbf{c} = (c_\mu)$ is sparse, then we have $\mathbf{y} = \mathbf{A}\mathbf{T}^{-1}\mathbf{c} + \boldsymbol{\epsilon} = \tilde{\mathbf{A}}\mathbf{c} + \boldsymbol{\epsilon}$. If the measurement matrix $\mathbf{A}$ is not correlated with $\mathbf{T}$, the sparse sequence of curvelet coefficients $\mathbf{c}$ can be recovered by a sparsity-constraint l_1-minimization [14]

$$\min_{\mathbf{c}} \|\mathbf{y} - \tilde{\mathbf{A}}\mathbf{c}\|_{l_2} + \lambda \|\mathbf{c}\|_{l_1}.$$

The second term is a regularization term that represents the a priori information of sparsity. To solve the minimization, an iterative curvelet thresholding (ICT) can be used, based on the Landweber descent method (see, e.g., [33])

$$\mathbf{c}_{p+1} = S_\tau(\mathbf{c}_p + \tilde{\mathbf{A}}^T(\mathbf{y} - \tilde{\mathbf{A}}\mathbf{c}_p)),$$

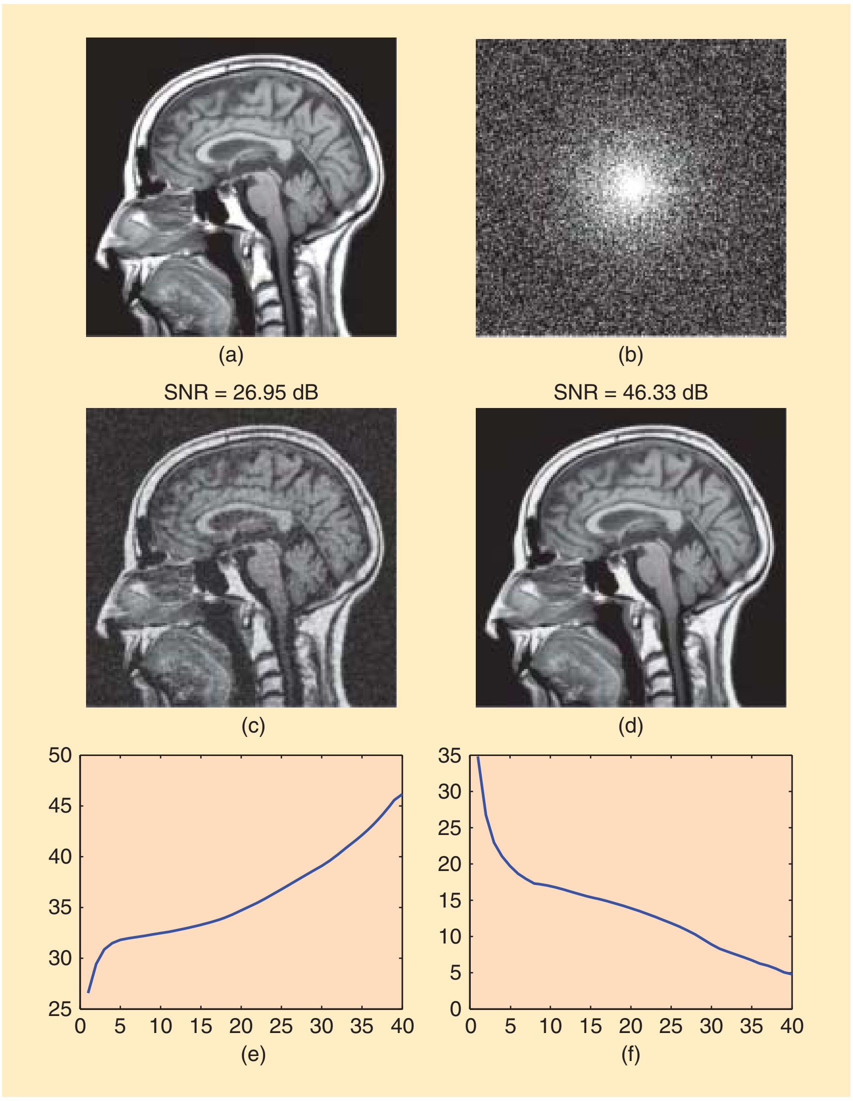
until $\|\mathbf{c}_{p+1} - \mathbf{c}_p\| < \varepsilon$, for a given error ε. Here the (soft) threshold function S_τ, given by

$$S_\tau(x) = \begin{cases} x - \tau, & x \geq \tau, \\ x + \tau, & x \leq -\tau, \\ 0, & |x| < \tau, \end{cases}$$

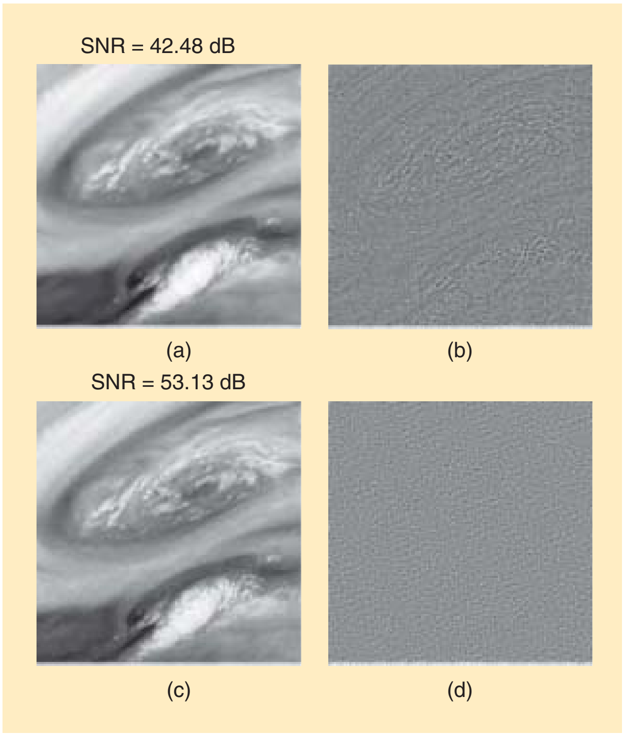
is taken component wisely, i.e., for a sequence $\mathbf{a} = (a_\mu)$ we have $S_\tau(\mathbf{a}) = (S_\tau a_\mu)$.

Figure 15 shows an example of compressed sensing with 25% Fourier measurements. Here the operator A is obtained by a random subsampling of the Fourier matrix. Figure 15(b) shows the 25% samples in Fourier domain, Figure 15(c) is the recovering result by zero-filling reconstruction, and Figure 15(d) is the result found by ICT. Figure 15(e) and (f) denotes the changes of the SNR and errors of the recovered images as the number of iterations increases. The unknown MRI image can be obtained by using highly incomplete measurements, which can reduce the online measurement time and thus lessen the pain of a patient.

The motivation of applying the curvelet thresholding method is that most natural images are compressible by the curvelet transform. Currently, a few researchers have applied the ICT method to compressed sensing in seismic data recovery [33], [34], [65], and remote sensing [44], [46]. Variant ICT methods (see e.g., [57]) have been also proposed for compressed sensing.



[FIG15] Compressed sensing in Fourier domain for medical imaging: (a) original MRI image, (b) pseudorandom Fourier sampling, (c) recovery by zero-filling reconstruction, (d) recovery by ICT, (e) SNR (in dB) of the recovered image versus the number of iterations for the ICT, and (f) recovery error versus the number of iterations for the ICT.



[FIG16] Compressed sensing in remote sensing: (a) recovery by iterative wavelet thresholding, (b) recovery error by the wavelet method, (c) recovery by ICT, and (d) recovery error by the curvelet method.

Figure 16 shows an example for the curvelet-based compressed sensing in remote sensing [44], [46]. It can be seen that the curvelet method is superior to the wavelet method to recover the edges.

FUTURE WORK

The multiresolution geometric analysis technique with curvelets as basis functions is verified as being effective in many fields. However, there are some challenging problems for future work.

1) The computational cost of the curvelet transform is higher than that of wavelets, especially in terms of 3-D problems. However, the theory and application of the 3-D curvelets are burgeoning areas of research, and it is possible that more efficient curvelet-like transforms will be developed in the near future. Currently, a fast message passing interface-based parallel implementation can somewhat reduce the cost [68]. How to build a fast orthogonal curvelet transform is still open.

2) The issue of how to explore suitable thresholding functions that incorporate and exploit the special characteristics of the curvelet transform is very important for curvelet applications involving edge detection, denoising, and numerical simulation.

ACKNOWLEDGMENTS

Jianwei Ma was supported by the NSFC under grants (40704019), CACS Innovation Fund (20094030085), TBRF (JC2007030), and PetroChina Innovation Fund (060511-1-1). Gerlind Plonka was supported by the German Research Foundation within the projects PL 170/11-2 and PL 170/13-1. This support is gratefully acknowledged.

AUTHORS

Jianwei Ma (jma@tsinghua.edu.cn) received the Ph.D. degree in solid mechanics from Tsinghua University in 2002. He has been a visiting scholar, postdoctoral research fellow, and guest professor with the University of Cambridge, the University of Oxford, the University of Huddersfield, the University of Grenoble (INRIA), and the University of Duisburg-Essen. He was an assistant professor and is now an associate professor with the School of Aerospace as well as a research director with the Institute of Seismic Exploration, Tsinghua University. From June to August 2006, he was a visiting professor at the Swiss Federal Institute of Technology (EPFL). In 2007, he was a visiting professor at Florida State University and the University of Colorado at Boulder. In 2008 and 2009, he was an invited professor at INRIA-Grenoble and Ecole des Mines de Paris. His research interests include wavelets, curvelets, image processing, compressed sensing, remote sensing, and seismic exploration. He is an editor of *International Journal Artificial Intelligence*.

Gerlind Plonka (gerlind.plonka@uni-due.de) received the Diploma degree in mathematics from the University of Rostock, Germany, in 1990, and the Ph.D. degree in mathematics as well as the Habilitation degree from the University of Rostock, in 1993 and 1996, respectively. Since 1998, she has been an associate professor for applied analysis at the University of Duisburg-Essen. Her current research interests include wavelet and frame theory, Fourier analysis, nonlinear diffusion, and applications in signal and image processing.

REFERENCES

[1] F. Andersson, M. de Hoop, H. Smith, and G. Uhlmann, "A multi-scale approach to hyperbolic evolution equations with limited smoothness," *Commun. Partial Differ. Equ.*, vol. 33, no. 6, pp. 988–1017, 2008.

[2] I. Bermejo-Moreno and D. Pullin, "On the non-local geometry of turbulence," *J. Fluid Mech.*, vol. 603, pp. 101–135, 2008.

[3] J. Bobin, J. Starck J. Fadili, Y. Moudden, and D. Donoho, "Morphological component analysis: An adaptive thresholding strategy," *IEEE Trans. Image Processing*, vol. 16, no. 11, pp. 2675–2681, 2007.

[4] E. Candès, "Harmonic analysis of neural networks," *Appl. Comput. Harmon. Anal.*, vol. 6, no. 2, pp. 197–218, 1999.

[5] E. Candès and L. Demanet, "Curvelets and Fourier integral operators," *C. R. Math. Acad. Sci. Paris*, vol. 336, no. 5, pp. 395–398, 2003.

[6] E. Candès and L. Demanet, "The curvelet representation of wave propagators is optimally sparse," *Commun. Pure Appl. Math.*, vol. 58, no. 11, pp. 1472–1528, 2005.

[7] E. Candès, L. Demanet, D. Donoho, and L. Ying, "Fast discrete curvelet transforms," *Multiscale Model. Simul.*, vol. 5, no. 3, pp. 861–899, 2006.

[8] E. Candès and D. Donoho, "Ridgelets: A key to higher-dimensional intermittency?," *Philos. Trans. R. Soc. London A, Math. Phys. Eng. Sci.*, vol. 357, no. 1760, pp. 2495–2509, 1999.

[9] E. Candès and D. Donoho, "Curvelets—A surprisingly effective nonadaptive representation for objects with edges," in *Curves and Surface Fitting: Saint-Malo 1999*, A. Cohen, C. Rabut, and L. Schumaker, Eds. Nashville: Vanderbilt Univ. Press, 2000, pp. 105–120.

[10] E. Candès and D. Donoho, "Continuous curvelet transform. I. Resolution of the wavefront set," *Appl. Comput. Harmon. Anal.*, vol. 19, no. 2, pp. 162–197, 2005.

[11] E. Candès and D. Donoho, "Continuous curvelet transform. II. Discretization and frames," *Appl. Comput. Harmon. Anal.*, vol. 19, no. 2, pp. 198–222, 2005.

[12] E. Candès and D. Donoho, "New tight frames of curvelets and optimal representations of objects with piecewise singularities," *Commun. Pure Appl. Math.*, vol. 57, no. 2, pp. 219–266, 2004.

[13] E. Candès and F. Guo, "New multiscale transforms, minimum total variation synthesis: Applications to edge-preserving image reconstruction," *Signal Process.*, vol. 82, no. 11, pp. 1519–1543, 2002.

[14] E. Candès, J. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Commun. Pure Appl. Math.*, vol. 59, no. 8, pp. 1207–1233, 2006.

[15] E. Candès and T. Tao, "Decoding by linear programming," *IEEE Trans. Inform. Theory*, vol. 51, no. 12, pp. 4203–4215, 2005.

[16] H. Chauris and T. Nguyen, "Seismic demigration/migration in the curvelet domain," *Geophysics*, vol. 73, no. 2, pp. S35–S46, 2008.

[17] M. Choi, R. Kim, M. Nam, and H. Kim, "Fusion of multispectral and panchromatic satellite images using the curvelet transform," *IEEE Geosci. Remote Sens. Lett.*, vol. 2, no. 2, pp. 136–140, 2005.

[18] I. Daubechies, *Ten Lectures on Wavelets*. Philadelphia, PA: SIAM, 1992.

[19] L. Demanet and L. Ying, "Curvelets and wave atoms for mirror-extended images," in *Proc. SPIE Wavelets XII*, San Diego, Aug. 2007, vol. 6701, p. 67010J.

[20] L. Demanet and L. Ying, "Wave atoms and sparsity of oscillatory patterns," *Appl. Comput. Harmon. Anal.*, vol. 23, no. 3, pp. 368–387, 2007.

[21] M. Do and M. Vetterli, "The contourlet transform: An efficient directional multiresolution image representation," *IEEE Trans. Image Processing*, vol. 14, no. 12, pp. 2091–2106, 2005.

[22] D. Donoho, "Compressed sensing," *IEEE Trans. Inform. Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[23] D. Donoho, "Wedgelets: Nearly minimax estimation of edges," *Ann. Statist.*, vol. 27, no. 3, pp. 859–897, 1999.

[24] D. Donoho and X. Huo, "Beamlets and multiscale image analysis," in *Multiscale and Multiresolution Methods* (Springer Lecture Notes in Computer Science Engineering, vol. 20), T. Barth, T. Chan, and R. Haimes, Eds. Berlin: Springer, 2002, pp. 149–196.

[25] H. Douma and M. de Hoop, "Leading-order seismic imaging using curvelets," *Geophysics*, vol. 72, no. 6, pp. S231–S248, 2007.

[26] M. Farge, N. Kevlahan, V. Perrier, and E. Goirand, "Wavelets and turbulence," *Proc. IEEE*, vol. 84, no. 4, pp. 639–669, 1996.

[27] M. Farge, G. Pellegrino, and K. Schneider, "Coherent vortex extraction in 3D turbulent flows using orthogonal wavelets," *Phys. Rev. Lett.*, vol. 87, no. 5, pp. 45011–45014, 2001.

[28] W. Freeman and E. Adelson, "The design and use of steerable filters," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 13, no. 9, pp. 891–906, 1991.

[29] X. Gao, T. Nguyen, and G. Strang, "A study of two-channel complex-valued filter banks and wavelets with orthogonality and symmetry properties," *IEEE Trans. Signal Processing*, vol. 50, no. 4, pp. 824–833, 2002.

[30] T. Geback and P. Koumoutsakos, "Edge detection in microscopy images using curvelets," *BMC Bioinformatics*, vol. 10, no. 75, 2009.

[31] K. Guo and D. Labate, "Optimally sparse multidimensional representation using shearlets," *SIAM J. Math. Anal.*, vol. 39, no. 1, pp. 298–318, 2007.

[32] G. Hennenfent and F. Herrmann, "Seismic denoising with nonuniformly sampled curvelets," *Comput. Sci. Eng.*, vol. 8, no. 3, pp. 16–25, 2006.

[33] F. Herrmann and G. Hennenfent, "Non-parametric seismic data recovery with curvelet frames," *Geophys. J. Int.*, vol. 173, no. 1, pp. 233–248, 2008.

[34] F. Herrmann, P. Moghaddam, and C. Stolk, "Sparsity- and continuity-promoting seismic image recovery with curvelet frames," *Appl. Comput. Harmon. Anal.*, vol. 24, no. 2, pp. 150–173, 2008.

[35] F. Herrmann, D. Wang, and D. Verschuur, "Adaptive curvelet-domain primary-multiple separation," *Geophysics*, vol. 73, no. 3, pp. A17–A21, 2008.

[36] L. Jiang, X. Feng, and H. Yin, "Structure and texture image inpainting using sparse representations and an iterative curvelet thresholding approach," *Int. J. Wavelets Multiresolut. Inform. Process.*, vol. 6, no. 5, pp. 691–705, 2008.

[37] N. Kingsbury, "Image processing with complex wavelets," *Philos. Trans. R. Soc. London A, Math. Phys. Sci.*, vol. 357, no. 1760, pp. 2543–2560, 1999.

[38] N. Kingsbury, "Complex wavelets for shift invariant analysis and filtering of signals," *Appl. Comput. Harmon. Anal.*, vol. 10, no. 3, pp. 234–253, 2001.

[39] D. Labate, W.-Q. Lim, G. Kutyniok, and G. Weiss, "Sparse multidimensional representation using shearlets," in *Proc. SPIE Wavelets XI*, San Diego, CA, 2005, vol. 5914, pp. 254–262.

[40] T. Lee, "Image representation using 2D Gabor wavelets," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 18, no. 10, pp. 1–13, 2008.

[41] T. Lin and F. Herrmann, "Compressed extrapolation," *Geophysics*, vol. 72, no. 5, pp. SM77–SM93, 2007.

[42] Y. Lu and M. N. Do, "Multidimensional directional filter banks and surfacelets," *IEEE Trans. Image Processing*, vol. 16, no. 4, pp. 918–931, 2007.

[43] J. Ma, "Deblurring using singular integrals and curvelet shrinkage," *Phys. Lett. A*, vol. 368, pp. 245–250, 2007.

[44] J. Ma, "Single-pixel remote sensing," *IEEE Geosci. Remote Sens. Lett.*, vol. 6, no. 2, pp. 199–203, 2009.

[45] J. Ma, A. Antoniadis, and F.-X. Le Dimet, "Curvelet-based multiscale detection and tracking for geophysical fluids," *IEEE Trans. Geosci. Remote Sensing*, vol. 44, no. 12, pp. 3626–3637, 2006.

[46] J. Ma and F.-X. Le Dimet, "Deblurring from highly incomplete measurements for remote sensing," *IEEE Trans. Geosci. Remote Sensing*, vol. 47, no. 3, pp. 792–802, 2009.

[47] J. Ma and M. Hussaini, "Three-dimensional curvelets for coherent vortex analysis of turbulence," *Appl. Phys. Lett.*, vol. 91, no. 18, p. 184101, 2007.

[48] J. Ma, M. Hussaini, O. Vasilyev, and F.-X. Le Dimet, "Multiscale geometric analysis of turbulence by curvelets," *Phys. Fluids*, vol. 21, p. 075104, 2009.

[49] J. Ma and G. Plonka, "Combined curvelet shrinkage and nonlinear anisotropic diffusion," *IEEE Trans. Image Processing*, vol. 16, no. 9, pp. 2198–2206, 2007.

[50] S. Mallat, *A Wavelet Tour of Signal Processing*. San Diego, CA: Academic, 1999.

[51] S. Mallat and G. Peyré, "A review of bandlet methods for geometrical image representation," *Numer. Algorithms*, vol. 44, no. 3, pp. 205–234, 2007.

[52] R. Neelamani, A. Baumstein, D. Gillard, M. Hadidi, and W. Soroka, "Coherent and random noise attenuation using the curvelet transform," *The Leading Edge*, vol. 27, no. 2, pp. 240–248, 2008.

[53] J. Neumann and G. Steidl, "Dual–tree complex wavelet transform in the frequency domain and an application to signal classification," *Int. J. Wavelets Multiresolut. Inform. Process.*, vol. 3, no. 1, pp. 43–66, 2005.

[54] E. Le Pennec and S. Mallat, "Sparse geometrical image approximation with bandlets," *IEEE Trans. Image Processing*, vol. 14, no. 4, pp. 423–438, 2005.

[55] B. Olshausen and D. Field, "Emergence of simple-cell receptive filed properties by learning a sparse code for natural images," *Nature*, vol. 381, no. 6583, pp. 607–609, 1996.

[56] G. Plonka and J. Ma, "Nonlinear regularized reaction-diffusion filters for denoising of images with textures," *IEEE Trans. Image Processing*, vol. 17, no. 8, pp. 1283–1294, 2008.

[57] G. Plonka and J. Ma, "Curvelet-wavelet regularized split Bregman method for compressed sensing," *Int. J. Wavelets Multiscale Inform. Processing,* to be published.

[58] H. Shan, J. Ma, and H. Yang, "Comparisons of wavelets, contourlets, and curvelets for seismic denoising," *J. Appl. Geophys.*, vol. 69, no. 2, pp. 103–115, 2009.

[59] E. Simoncelli, W. Freeman, E. Adelson, and D. Heeger, "Shiftable multiscale transforms," *IEEE Trans. Inform. Theory*, vol. 38, no. 2, pp. 587–607, 1992.

[60] J. Starck, E. Candès, and D. Donoho, "The curvelet transform for image denoising," *IEEE Trans. Image Processing*, vol. 11, no. 6, pp. 670–684, 2002.

[61] J. Starck, E. Candès, and D. Donoho, "Astronomical image representation by the curvelet transform," *Astron. Astrophys.*, vol. 398, pp. 785–800, 2003.

[62] J. Starck, F. Murtagh, E. Candès, F. Murtagh, and D. Donoho, "Gray and color image contrast enhancement by the curvelet transform," *IEEE Trans. Image Processing*, vol. 12, no. 6, pp. 706–717, 2003.

[63] J. Starck, M. Elad, and D. Donoho, "Image decomposition via the combination of sparse representation and a variational approach," *IEEE Trans. Image Processing*, vol. 14, no. 10, pp. 1570–1582, 2005.

[64] B. Sun, J. Ma, H. Chauris, and H. Yang, "Solving the wave equation in the curvelet domain: A multiscale and multidirectional approach," *J. Seismic Exploration*, vol. 18, no. 4, pp. 385–399, 2009.

[65] G. Tang, R. Shahidi, J. Ma, and F. Herrmann, "Two dimensional stochastic sampling schemes for curvelet based seismic data recovery," submitted for publication.

[66] L. Tessens, A. Pizurica, A. Alecu, A. Munteanu, and W. Philips, "Context adaptive image denoising through modeling of curvelet domain statistics," *J. Electron. Imaging*, vol. 17, no. 3, pp. 03021:1–03021:17, 2008.

[67] R. Willett and K. Nowak, "Platelets: A multiscale approach for recovering edges and surfaces in photon-limited medical imaging," *IEEE Trans. Med. Imaging*, vol. 22, no. 3, pp. 332–350, 2003.

[68] L. Ying, L. Demanet, and E. Candès, "3D discrete curvelet transform," in *Proc. SPIE Wavelets XI*, San Diego, CA, 2005, vol. 5914, p. 591413.

[69] B. Zhang, J. Fadili, and J. Starck, "Wavelets, ridgelets, and curvelets for Poisson noise removal," *IEEE Trans. Image Processing*, vol. 17, no. 7, pp. 1093–1108, 2008.

[70] C. Zhang, L. Cheng, Z. Qiu, and L. Cheng, "Multipurpose watermarking based on multiscale curvelet transform," *IEEE Trans. Inform. Forensics Security*, vol. 3, no. 4, pp. 611–619, 2008.

SP

Greg Slabaugh, Richard Boyes,
and Xiaoyun Yang

Multicore Image Processing with OpenMP

One of the recent innovations in computer engineering has been the development of multicore processors, which are composed of two or more independent cores in a single physical package. Today, many processors, including digital signal processors (DSPs), mobile, graphics, and general-purpose central processing units (CPUs) [1] have a multicore design, driven by the demand of higher performance. Major CPU vendors have changed strategy away from increasing the raw clock rate to adding on-chip support for multithreading by increases in the number of cores; dual- and quad-core processors are now commonplace. Signal and image processing programmers can benefit dramatically from these advances in hardware, by modifying single-threaded code to exploit parallelism to run on multiple cores.

This article describes the use of open multiprocessing (OpenMP) to multithread image processing applications to take advantage of multicore general-purpose CPUs. OpenMP is an extensive and powerful application programming interface (API), that supports many functionalities required for parallel programming. The purpose of this article is to provide a high-level overview of OpenMP and present simple image processing operations to demonstrate the ease of implementation and effectiveness of OpenMP. More sophisticated applications could be built on similar principles.

OPENMP

Historically, a key challenge in parallel computing has been the lack of a broadly supported, simple-to-implement parallel programming model. As a result, numerous vendors provided different models, with often mixed degrees of complexity and portability. Software programmers subsequently found it difficult to adapt applications to take advantage of multicore hardware advances.

OpenMP was designed to bridge this gap, providing an industry standard, parallel programming API for shared memory multiprocessors, including multicore processors. A vendor-independent OpenMP Architecture Review Board, which includes most of the major computer manufacturers, oversees the OpenMP standard and approves new versions of the specification. Support for OpenMP is currently available in most modern Fortran and C/C++ compilers as well as numerous operating systems, including Microsoft Windows, Linux, and Apple Macintosh OS X. Version 1.0 of OpenMP was released in 1997. The latest version, 3.0, was released in 2008. Please see the official OpenMP Web site [2] for the full specification, list of compilers supporting the standard, and reference documents.

We should note that OpenMP is certainly not the only way of achieving parallelism on multicore systems. Other implementation models, such as CILK, Pthreads, and MPI [3], [4] exist and may be a good choice depending on the hardware, application, and the preference of the programmer. In our experience, OpenMP has the advantages of being exceedingly simple to learn and implement, in addition to being powerful and well suited to modern processor architectures.

USING OPENMP

OpenMP works as a set of preprocessor directives, run-time library routines, and environment variables provided to the programmer, who instructs the compiler how a section of code can be multithreaded. In Fortran, the directives appear as comments, while in C/C++ they are implemented as pragmas. In this way, compilers that do not support OpenMP will automatically ignore OpenMP directives, while compilers that do support the standard will process and potentially optimize the code based on the directives. Since the OpenMP API is independent of the machine/operating system, properly written OpenMP code for one platform can easily be recompiled and run on another platform. However, in this article, we present C++ code examples that were compiled using Microsoft Visual C++ 2005 and executed in Windows XP. Using this compiler, one can enable OpenMP in the project settings (Configuration Properties → C/C++ → Language → OpenMP Support) and include omp.h.

An OpenMP application always begins with a single thread of control, called the master thread, which exists for the duration of the program. The set of variables available to any particular thread is called the thread's execution context. During execution, the master thread may encounter parallel regions, at which the master thread will fork new threads, each with its own stack and execution context. At the end of the parallel region, the forked threads will terminate, and the master thread continues execution. Nested parallelism, for which forked threads fork further threads, is supported.

LOOP-LEVEL PARALLELISM

As mentioned above, parallelism is added to an application by including pragmas, which, in C++, have the following form:

```
#pragma omp <directive>
        [clauses]
```

There are numerous directives, but this article focuses on the `parallel for` directive, which offers a simple way to

Digital Object Identifier 10.1109/MSP.2009.935452

achieve loop-level parallelism, often existing in signal- and image-processing algorithms. The optional clauses modify the behavior of the directive.

The parallelization of loops is the most common use of OpenMP. Consider the following code, which computes a sine wave with amplitude A and frequency w

```
for (int n=0; n<N; n++)
   x[n] = A*sin(w*n);
```

With OpenMP, this code can be trivially parallelized as

```
#pragma omp parallel for
 for (int n=0; n<N; n++)
     x[n] = A*sin(w*n);
```

Here, the directive instructs the compiler that the next `for` loop is to be parallelized. The compiler will then distribute the work among a set of forked threads. If we assume that there are four threads forked, and $N = 100$, then the iterations may be spread among the processors such that iterations $0-24$ are given to Thread 1, iterations $25-49$ are given to Thread 2, iterations $50-74$ are given to Thread 3, iterations $75-99$ are given to Thread 4. Such allocation of work assumes static scheduling, which will be discussed below. The four threads will run simultaneously. If a forked thread completes its work before any other forked thread, it will block. Once all forked threads complete their work, the master thread then resumes execution. Note the simplicity of using OpenMP—the loop was parallelized with a single line of code.

The code in the above example was easily parallelized because it does not contain loop dependencies, which means the compiler can execute the loop in any order. Consider a modification of this loop

```
x[0] = 0;
for (int n=1; n<N; n++)
    x[n] = x[n-1] + A*sin(w*n);
```

The loop is no longer trivially parallelized, as the computation of the `x[n]` now depends on `x[n-1]`. Thread 2 may start by computing `x[25]`, which depends on `x[24]`. However, Thread 1 might not yet have computed `x[24]`, resulting in a runtime error. The programmer must be careful to ensure the parallelized loop is free of loop dependencies, as the compiler does not check this. As a result of such dependencies, some algorithms require additional coding to remove the dependencies [4] to render them amenable to parallelization.

VARIABLE SCOPE

As mentioned above, every thread has its own execution stack that contains variables in the scope of the thread. When parallelizing code, it is very important to identify which variables are shared between the threads, and which are private. In the parallelized sine wave example above, the variables x, A, w, and N were shared, while n was private; that is, each thread has its own n but shares all the other variables.

OpenMP provides explicit constructs to specify shared and private variables in the execution stack. By default, all variables are shared, except

1) the loop index.
2) variables local (declared within) the loop.
3) variables listed in private clauses.

One can explicitly assign shared and private variables using directive clauses

```
#pragma omp parallel for \
private(n)  \
shared(A, x, w)
for (int n = 0; n < N; n++)
    x[n] = A * sin(w * n);
```

Copies of the variables in the private clause will be placed into each thread's execution context. Note, however, that any variable in a private clause is initially undefined. This can lead to coding errors. For example, consider

```
int x = 5;
#pragma omp parallel for
   private(x)
for (int n = 0; n < N; n++)
    // The value of x is
    undefined
```

in this case, the `firstprivate` clause can be used to copy the value of a variable to the execution stack of each thread

```
int x = 5;
#pragma omp parallel for
firstprivate(x)
for (int n = 0; n < N; n++)
    // The value of x is 5
    for each thread
```

SCHEDULING

Earlier, we mentioned static scheduling, which divides work of the loop evenly among the different threads. Static scheduling is the default work sharing construct and works well for balanced loops that have a relatively constant cost per iteration. However, some loops are unbalanced, with some iterations taking much longer than others. For such loops, static scheduling is not ideal, as fast threads will complete their work early and block until slow threads have completed their work. With OpenMP, it is possible to specify the scheduling mechanism (for example, using the `static` or `dynamic` clause). In a dynamic schedule, the number of iterations for each thread can vary depending on the workload. When free, each thread requests more iterations until the loop is complete. Dynamic scheduling is more flexible but does add additional overhead in coordinating the distribution of work amongst the threads. Later, we will show an example where dynamic scheduling makes a significant difference in run time of a parallelized image processing algorithm.

By default, the system will decide how many threads to fork during run time. The number of spawned threads can be retrieved using `integer omp_get_num_threads(void)`. In addition, the number of threads may be set using `omp_set_num_threads(integer)` or by using an environment variable, `OMP_NUM_THREADS`.

APPLICATIONS OF IMAGE PROCESSING USING OPENMP

In the previous section, we provided an introduction to OpenMP and a short description of how one may achieve loop-level parallelization using the `parallel for` pragma. In this section, we demonstrate examples that show the ease and power of OpenMP for image processing. The examples are, by design, simple so that the principles can be easily demonstrated.

IMAGE WARPING

An image warp is a spatial transformation of an image and is commonly found in photo-editing software as well as registration algorithms. In this example, we apply a "twist" transformation of the form

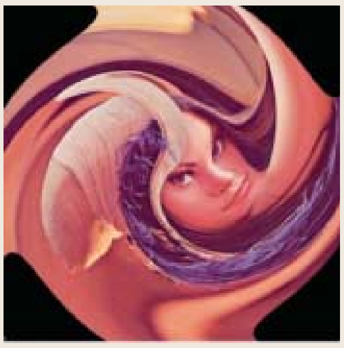
$$x' = (x - c_x)\cos\theta + (y - c_y)\sin\theta + c_x$$
$$y' = -(y - c_y)\sin\theta + (y - c_y)\cos\theta + c_y, \quad (1)$$

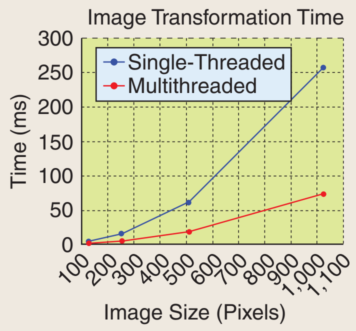
where $[c_x, c_y]^T$ is the rotation center, $\theta = r/2$ is a rotation angle that increases with radius r, $[x, y]^T$ is the point before transformation, and $[x', y']^T$ is the point after transformation. The effect of this image transformation in shown in Figure 1(b).

We implemented this transformation using OpenMP; the code listing appears in Listing 1. In the code, both the original image and transformed image are shared variables, along with the width and height of the image. The code loops over the pixels $[x', y']^T$ of the transformed image, mapping them back to the original image using the inverse transform of (1). In the original image, the pixels are bilinearly interpolated. Each thread requires its own variables for `x`, `y`, `index`, `radius`, `theta`, `xp`, and `yp`. Since these variables are initialized within the parallelized code, we use the shared clause. OpenMP will multithread the outer loop (over `yp`) using static scheduling.

On a 512 × 512 image, and using a quad-core 2.4 GHz CPU, the single-threaded code requires 62.2 ms to process the image, while the multithreaded code requires 17.7 ms, corresponding to a 3.5× speedup. In Figure 1(c), we present a plot showing the speedup as a function of image size (measured in one dimension of the square image). The code, very easily multithreaded, achieves an excellent speedup when executed on multiple cores.

(a)


(b)



(c)

[FIG1] (a)–(c) show an example of a twist transformation applied to an image.

[LISTING 1] PARALLELIZED CODE FOR THE IMAGE WARP.

```
int index, xp, yp, tx = width / 2, ty = height / 2;
float x, y, radius, theta, PI = 3.141527f, DRAD = 180.0f / PI;
#pragma omp parallel for \
shared(inputImage, outputImage, width, height) \
private(x, y, index, radius, theta, xp, yp)
for (yp = 0; yp < height; yp++) {
  for (xp = 0; xp < width; xp++) {
    index = xp + yp * width;
    radius = sqrtf((xp - tx) * (xp - tx) + (yp - ty) * (yp - ty));
    theta = (radius / 2) * DRAD;
    x = cos(theta) * (xp - tx) - sin(theta) * (yp - ty) + tx;
    y = sin(theta) * (xp - tx) + cos(theta) * (yp - ty) + ty;
    outputImage[index] = BilinearlyInterpolate(inputImage, width, height,
    x, y);
  }
}
```

MATHEMATICAL BINARY MORPHOLOGY

Mathematical morphology was originally developed for binary images and later was extended to grayscale images. Morphological operations are widely used in image segmentation, noise removal, and many other applications and employ a structuring element to probe an image and create an output image [5]. At its core, mathematical morphology has two basic operations: erosion and dilation. Erosion and dilation of an image/set X by a structuring element B are defined in (2)

$$\text{Erosion}_B(X) = \{x | B_x \subseteq X\}$$
$$\text{Dilation}_B(X) = \{x | B_x \cap X \neq \emptyset\}, \quad (2)$$

where B represents the structuring element and B_x denotes B centered at x. The result of erosion by B can be explained as the locus of points hit by the center of B when B moves entirely inside X. The result of dilation by B are the locus of the points covered by B when the center of B moves inside X. Other operations can be defined using combinations of erosion and dilation. For example, a closing is defined as a dilation followed by an erosion.

We implemented binary morphology erosion and dilation using OpenMP; the erosion code is listed in Listing 2 and the dilation code is similar. At each pixel, the function `FullFit` checks if the structuring element entirely fits into foreground binary region of the input image, as defined by (2). Here, OpenMP will multithread the outer loop (over `y`) using dynamic scheduling.

On a 512 × 512 image, and using a quad-core 2.4 GHz CPU, for a closing operation by a disk-structure element with radius 15 pixels, the single-threaded code requires 79.4 ms to process the image, while the multithreaded code requires 33.9 ms, corresponding to a 2.34× speedup. Figure 2(a) and (b)

[LISTING 2] PARALLELIZED CODE FOR BINARY EROSION.

```
int x, y;
#pragma omp parallel for \
shared(inputImage, outputImage, structuringElement, width, height) \
private(x, y) schedule(dynamic)
for (y = 0; y < height; y++) {
  for (x = 0; x < width; x++) {
       int index = x + y*width;
       if (inputImage[index]) {
          if (FullFit(inputImage, x, y, structuringElement))
             outputImage[index]=1;
          else
             outputImage[index]=0;
       }
   }
}
```

illustrates the input image and the closed result, respectively. In Figure 2(c), we present a plot showing the computation time as a function of structure element size (in radial pixels).

Binary morphology is a good example where the dynamic scheduling is helpful, as the same example scheduled statically requires 95.1 ms. In this example, the workload for a thread is unbalanced by the shape of the input. There may be large differences in the number of foreground pixels within the part of the image allocated to each thread, meaning dynamic scheduling is a better choice for how to distribute the work.

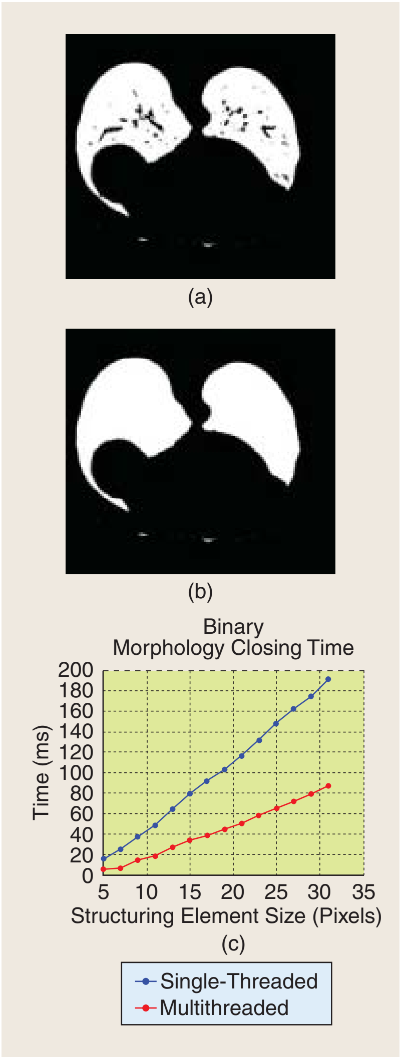
MEDIAN FILTERING

Median filtering is a commonly applied nonlinear filtering technique that is particularly useful in removing speckle and salt-and-pepper noise [6]. Simply put, an image neighborhood surrounding each pixel is defined, and the median value of this neighborhood is calculated and is used to replace the original pixel in the output image

$$I_{\mathrm{med}}[x,y] = \mathrm{median}(I_{\mathrm{orig}}[i,j], i,j \in n\mathrm{bor}[x,y]). \quad (3)$$

In this example, we choose a square neighborhood around each pixel, defined using the halfwidth of the neighborhood, i.e., for a halfwidth of n, the number of pixels in the neighborhood would be $(2n+1)^2$. At each pixel, the function GetNbors retrieves the neighbors; any neighbors that lie outside the image domain are assigned to be that of the nearest pixel within the image boundary. These neighbors are then sorted using the C++ STL sort function and the median selected.

On a 512 × 512 medical image, and using a quad-core 2.4 GHz CPU, we show the result of median filtering using a half width of three, i.e., the number of neighbors $=(2\times3+1)^2= 49$ in Figure 3(a) and (b). In Figure 3(c), we demonstrate the linear acceleration of the median filtering using different numbers of threads on different sizes of neighborhoods.



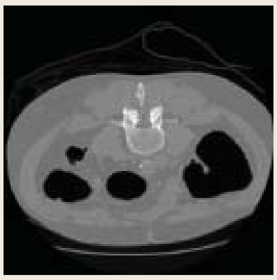
[FIG2] (a)–(c) show an example of a morphological closing applied to an image.

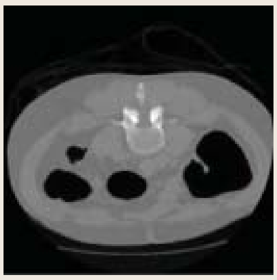
[LISTING 3] PARALLELIZED CODE FOR MEDIAN FILTERING USING A HALF WIDTH OF THREE.

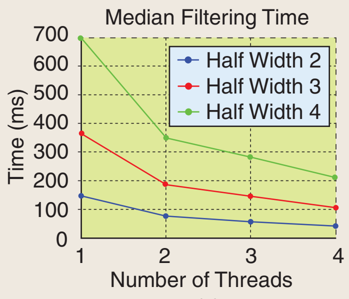
```
int x, y, halfWidth, nborSize;
PixelType nbors[MAX_NBOR_SIZE];
halfWidth = 3;
nborSize = 2*halfWidth + 1;
nborSize *= nborSize;
#pragma omp parallel for \
shared(inputImage, outputImage, structuringElement, width, height) \
private(x, y, nbors) firstprivate(halfWidth, nborSize) schedule(static)
for (y = 0; y < height; y++) {
   for (x = 0; x < width; x++) {
        GetNbors(inputImage, x, y, width, height, halfWidth, nbors);
        sort(&nbors[0], &nbors[nborSize]);
        int index = x + y*width;
        outputImage[index] = nbors[nborSize/2];
   }
}
```

NORMALIZATION

Normalization is a process whereby the pixel intensities are scaled linearly. The linear scale factors can include those that will give the normalized image a prescribed minimum and maximum, or, say, a new intensity average. This is usually performed to bring the intensities into a standard range. In our example, we wish to alter the pixel


(a)


(b)



(c)

[FIG3] **(a)–(c) show an example of median filtering applied to an image.**

[LISTING 4] PARALLELIZED CODE FOR RESCALING DATA.

```
int index;
int n = width*height;
float mean = 0.0, var = 0.0, svar, std;
// Calculate the mean of the image intensities
#pragma omp parallel for \
shared(inputImage, n) reduction(+:mean) \
private(index) schedule(static)
for (index = 0; index < n; index++) {
   mean += (float)(inputImage[index]);
}
mean /= (float)n;

// Calculate the standard deviation of the image intensities
#pragma omp parallel for \
shared(inputImage, n) reduction(+:var) \
private(index, svar) schedule(static)
for (index = 0; index < n; index++) {
   svar = (float)(inputImage[index]) - mean;
   var += svar*svar;
}
var /= (float)n;
std = sqrtf(var);
// Rescale using the calculated mean and standard deviation
#pragma omp parallel for \
shared(inputImage, outputImage, n) private(index) \
firstprivate(mean, std) schedule(static)
for(index = 0; index < n; index++) {
   outputImage[index] = (inputImage[index] - mean)/std;
}
```

intensities so that they have a mean intensity of zero and a standard deviation of unity, which is common when analyzing images from a statistical viewpoint. This is achieved by first calculating the mean and standard deviation of the pixel intensities, and then scaling them as

$$I_{\text{new}} = \frac{(I_{\text{orig}} - \mu)}{\sigma}, \tag{4}$$

where μ and σ are the mean and standard deviation of the image intensities. To parallelize the estimation of μ and σ, we use an OpenMP reduction variable, which indicates that a variable has to be accumulated from all the threads in some way within the parallel loop. In our example, the reduction performs a sum $(+)$, although there are several other types, including subtraction $(-)$, product $(*)$, and bit-wise and logical operations. Note that in Fortran, minimum and maximum can also be used in reduction clauses as they are built in functions, whereas in C++ they are not.

To test the code, we renormalized the image from Figure 3(a) for different threads, and the processing took 5.6 ms for one thread, and 1.6 ms for four threads, demonstrating a near factor of four acceleration.

OUTLOOK

This article has only scratched the surface of the capabilities of OpenMP for parallelized signal and image processing on multicore machines. More advanced features, such as synchronization and parallel regions, extend the basic functionalities described here. However, with simple use of the OpenMP `parallel for` directive, it is remarkably easy for the signal processing programmer to achieve loop level parallelism. As general-purpose CPUs continue to advance, OpenMP will continue provide an uncomplicated way to harness the increasing power of multicore architectures.

AUTHORS

Greg Slabaugh (greg.slabaugh@medicsight.com) is the head of research and development at Medicsight PLC.

Richard Boyes (richard.boyes@medicsight.com) is a scientific research and development engineer at Medicsight PLC.

Xiaoyun Yang (xiaoyun.yang@medicsight.com) is a senior scientific research and development engineer at Medicsight PLC.

REFERENCES

[1] G. Blake, R. G. Deslinski, and T. Mudge, "A survey of multicore processors: A review of their common attributes," *IEEE Signal Processing Mag.*, vol. 26, no. 6, pp. 26–37, 2009.
[2] OpenMP Web site. [Online]. Available: http://www.openmp.org
[3] H. Kim and R. Bond, "Multicore software technologies: A survey," *IEEE Signal Processing Mag.*, vol. 26, no. 6, pp. 80–89, 2009.
[4] R. Chandra, L. Dagum, D. Kohr, D. Maydan, J. McDonald, and R. Menon, *Parallel Programming in OpenMP*, 1st ed. San Mateo, CA: Morgan Kaufmann, 2001.
[5] P. Soille, *Morphological Image Analysis*, 2nd ed. New York: Springer-Verlag, 2003.
[6] A. Jain, *Fundamentals of Digital Image Processing*, 1st ed. Englewood Cliffs, NJ: Prentice-Hall, 1989.
[SP]

Tae Hyun Yoon and Eon Kyeong Joo

A Flexible Window Function for Spectral Analysis

"DSP Tips and Tricks" introduces practical design and implementation signal processing algorithms that you may wish to incorporate into your designs. We welcome readers to submit their contributions. Contact Associate Editors Rick Lyons (R.Lyons@ieee.org) or C. Britton Rorabaugh (dspboss@aol.com).

Regarding window functions used in spectral analysis, the most important performance measures are 3-dB bandwidth and sidelobe attenuation. For many window functions, Hanning and Hamming for example, we have no control over a window's 3-dB bandwidth and sidelobe attenuation for a given window length. For other window functions—Kaiser, Gaussian, and Chebyshev—we can reduce those windows' 3-dB bandwidth to get improved spectral resolution. However, with these later window functions (what we refer to as "conventional windows"), spectral resolution improvement comes at the expense of sidelobe attenuation reduction that degrades our ability to avoid undesirable spectral leakage. Likewise we can increase those windows' sidelobe attenuation, but only by sacrificing desirable spectral resolution. This article describes a novel window function that enables us to control both its 3-dB bandwidth (spectral resolution) and sidelobe attenuation (spectral leakage) independently.

The 3-dB bandwidth, sidelobe attenuation, and roll-off rate are used to measure the performance of windows for power spectral density (PSD) estimation [1]–[3]. Improved frequency resolution of the estimated PSD can be obtained if we reduce a window's 3-dB bandwidth. The sidelobe attenuation means the difference between magnitude of the mainlobe and the maximum magnitude of the sidelobes. The sidelobe roll-off rate is the asymptotic decay rate of sidelobe peaks. Undesirable spectral leakage [4]–[6] can be reduced by increasing sidelobe attenuation and roll-off rate. Therefore, an ideal window for PSD estimation has zero bandwidth and infinite sidelobe attenuation such as an impulse function in frequency domain.

The conventional windows are able to control 3-dB bandwidth or sidelobe attenuation by only one parameter in general [1], [7]–[12]. Thus, they cannot control these two characteristics independently. In other words, if we reduce a window function's 3-dB bandwidth, the sidelobe attenuation is also reduced, and vice versa [5], [6]. This behavior is the cause of the tradeoff problem between good frequency resolution and acceptable spectral leakage in the estimated PSD. The Butterworth window does not have this problem because it allows control of the 3-dB bandwidth and sidelobe attenuation independently.

Butterworth windows are used as antialiasing filters to reduce the noise in the reconstructed image in previous research [13]. They are also used to remove the edge effect of the matched filter output in pattern matching algorithm [14]. The transfer function of a Butterworth filter is adopted as a window in those applications. However, in this article, a portion of the impulse response of a Butterworth filter is called the Butterworth window and its characteristics in PSD estimation are analyzed.

BUTTERWORTH WINDOW

The Butterworth window can be obtained by the standard Butterworth filter design procedure. Important to us is the frequency magnitude response $|H(f)|$ of a Butterworth filter, denoted by [2] and [6]

$$|H(f)| = 1/\sqrt{1+\left(\frac{f}{f_c}\right)^{2N}}, \qquad (1)$$

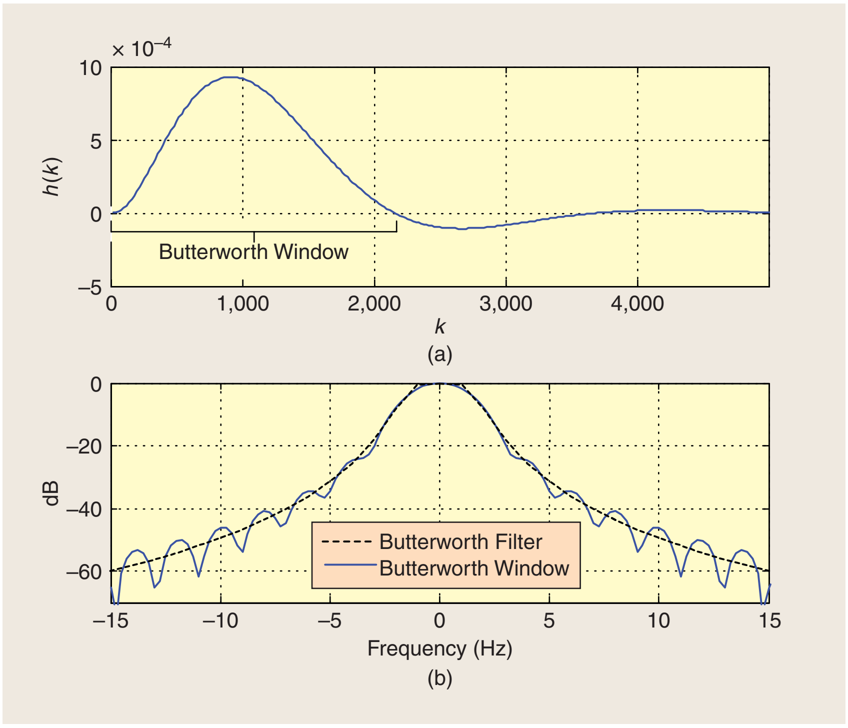
where f is frequency in hertz.

The Butterworth filter is characterized by two independent parameters, 3-dB cutoff frequency f_c and filter order N. The cutoff frequency and order of the Butterworth filter serve as parameters that control the bandwidth and sidelobe attenuation of the Butterworth window. The cutoff frequency of a filter has the identical meaning with the bandwidth of a window. However, the cutoff frequency is represented as a half of the bandwidth since the bandwidth of a window refers to two-sided frequency from negative to positive, while the cutoff frequency of a filter refers to only one-sided positive frequency. Our desired window spectrum is identical to the frequency response of the Butterworth filter. Thus, the inverse Fourier transform is applied to the Butterworth filter's frequency response, in (1), to obtain the filter's impulse response, and a portion of that response becomes the Butterworth window in the time domain.

SIMULATION AND PERFORMANCE ANALYSIS

In our simulation, the frequency and impulse responses of Butterworth filters are investigated to design the Butterworth window by varying the cutoff frequency f_c and filter order N. The sampling frequency f_s is set to 2,048 Hz. The magnitude levels of the impulse response of a

Digital Object Identifier 10.1109/MSP.2009.935422



[FIG1] **Butterworth filter and window: (a) filter impulse response and window function and (b) the filter magnitude response and window spectrum.**

Butterworth filter are nearly zero after a certain point—almost all information that determines the filter's frequency response is in the portion before that zero-magnitude point of the impulse response. Therefore, it is expected that the suitable length of a Butterworth window can be determined by only a part of the infinite-time duration impulse response of a Butterworth filter.

Figure 1(a) shows the time-domain impulse response $h(k)$ of a unity-gain low-pass Butterworth filter when $f_c = 0.75$ Hz and $N = 3$. In that figure, we show the initial positive-only portion of the impulse response that becomes our desired Butterworth window. The 2,139th sample of $h(k)$ is the point that the magnitude of the impulse response of the Butterworth filter becomes zero for the first time.

The solid curve in Figure 1(b) is the frequency spectrum of the 2,139-sample Butterworth window. The 3-dB bandwidth and sidelobe attenuation of this window are 1.3 Hz and 24.3 dB, respectively. In Figure 1(b), for comparison, we show the frequency magnitude response of the Butterworth filter as the dashed curve. We see that there is no significant difference between the magnitude response of the Butterworth filter and the spectrum of the Butterworth window. Therefore, a suitable length of the Butterworth window may be considered to be up to the point where the magnitude of the impulse response of filter becomes zero for the first time.

Based on the order N, the sampling frequency f_s, and the cutoff frequency f_c of the Butterworth filter, we have empirically determined the suitable lengths of the Butterworth windows to be those given in Table 1. Here the $\lfloor x \rfloor$ notation means the integer part of x.

The frequency characteristics of Butterworth windows with $f_c = 0.75$ Hz are shown in Table 2. The sidelobe attenuation is increased from ten to 30.4 dB as the filter order is increased from one to five, while the 3-dB bandwidth is fixed at about 1.5 Hz.

The PSD of an example signal is estimated by Butterworth windows to confirm the performance. The signal $x(t)$ used for our simulation is

$$\begin{aligned} x(t) = {} & 0.84\cos(2\pi \cdot 52 \cdot t) \\ & + 0.8\cos(2\pi \cdot 65.5 \cdot t) \\ & + 0.3\cos(2\pi \cdot 85 \cdot t) \\ & + 1.1\cos(2\pi \cdot 105 \cdot t) \\ & + 0.35\cos(2\pi \cdot 140 \cdot t) \\ & + 0.98\cos(2\pi \cdot 159 \cdot t) \\ & + 0.6\cos(2\pi \cdot 174 \cdot t) \\ & + 0.8\cos(2\pi \cdot 190 \cdot t) \\ & + \cos(2\pi \cdot 205 \cdot t). \end{aligned} \tag{2}$$

The solid lines in Figure 2(a) show the ideal PSD of $x(t)$. The dotted curve in Figure 2(a) shows the estimated PSD of a 2,139-sample rectangular windowed $x(t)$, using Welch's method [15], where that window's insufficient sidelobe attenuation (spectral leakage) produces

[TABLE 1] SUITABLE LENGTHS OF BUTTERWORTH WINDOWS (F_S = SAMPLING FREQUENCY, F_C = CUTOFF FREQUENCY).

FILTER ORDER, *N*	WINDOW LENGTH (SAMPLES)
1	$\left\lfloor 0.660 \cdot \frac{f_s}{f_c} \right\rfloor$
2	$\left\lfloor 0.705 \cdot \frac{f_s}{f_c} \right\rfloor$
3	$\left\lfloor 0.784 \cdot \frac{f_s}{f_c} \right\rfloor$
4	$\left\lfloor 0.890 \cdot \frac{f_s}{f_c} \right\rfloor$
5	$\left\lfloor 1.005 \cdot \frac{f_s}{f_c} \right\rfloor$

[TABLE 2] FREQUENCY CHARACTERISTICS OF THE BUTTERWORTH WINDOWS.

FILTER ORDER	3-DB BANDWIDTH	SIDELOBE ATTENUATION	ROLL-OFF RATE
1	1.5 Hz	10.0 dB	
2	1.4 Hz	18.2 dB	
3	1.3 Hz	24.3 dB	−12 dB/OCT.
4	1.3 Hz	28.8 dB	
5	1.3 Hz	30.4 dB	

false spectral components, particularly on either side of a relatively high-level ideal PSD spectral component.

Figure 2(b) shows the estimated PSDs of $x(t)$ using various 2,139-sample Butterworth windows, where we see that reducing the cutoff frequency and increasing the order can reduce spectral leakage without the undesirable spectral mainlobe broadening (loss of resolution) experienced by the conventional window functions. It means the tradeoff problem between resolution and spectral leakage is solved. This beneficial behavior is illustrated in Table 3, where Butterworth windows are compared to the conventional window functions. In that table, we see that Butterworth windows can increase their sidelobe attenuation without the undesirable mainlobe broadening exhibited by the conventional window functions. Reference [16] provides spectral plots comparing Butterworth windows to the conventional window functions in Table 3.

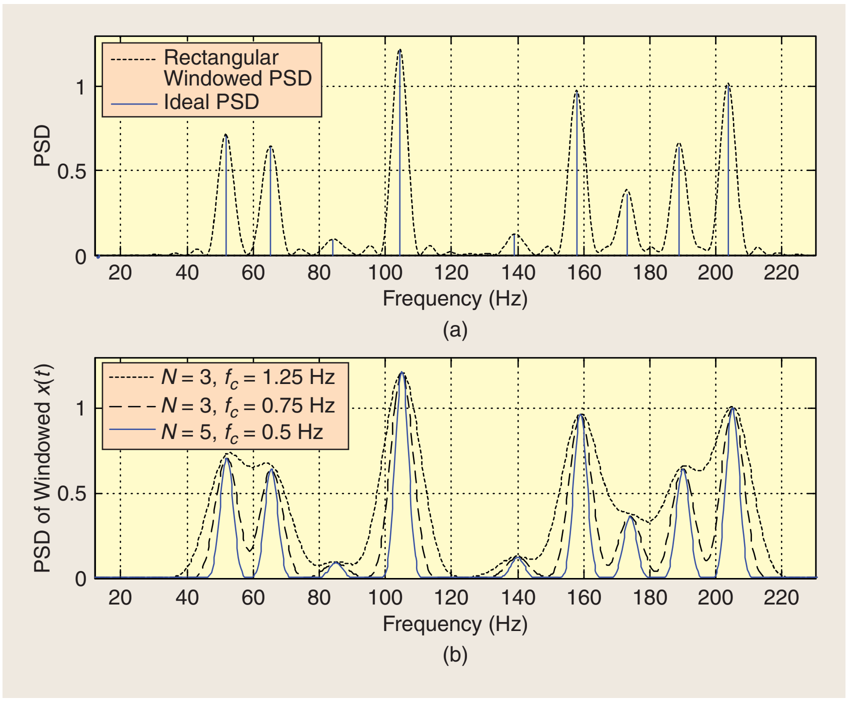
IMPLEMENTATION ISSUES

- The computational time of PSD estimation is not related to window type, but rather the window length and estimation method. So Butterworth windows have the same computational workload as the conventional window functions.
- To use the computationally efficient radix-2 fast Fourier transform algorithm, we suggest that the time-domain samples of Butterworth window should be zero padded to make the window length an integer power of two. As an alternative to zero padding, we can restrict the Butterworth window's f_c cutoff frequency to be

$$f_c = \frac{Kf_s}{2^{M+1}}, \quad (3)$$

which leads to Butterworth windows that are 2^M in length, where K is one of the scaling constants from Table 1, and M is an integer.

- Because Butterworth windows are not symmetrical, any specialized spectral analysis scheme that requires the imaginary part of a window function's spectrum to be all zero will not work with the Butterworth windows.



[FIG2] Ideal and windowed PSD: (a) rectangular windowed and ideal PSDs and (b) PSDs using various Butterworth windows.

[TABLE 3] COMPARISON OF FREQUENCY CHARACTERISTICS.

WINDOW		3-DB BANDWIDTH	SIDELOBE ATTENUATION
RECTANGULAR		0.879 Hz	13.3-dB
TRIANGULAR		1.270 Hz	26.5 dB
HANNING		1.367 Hz	31.3-dB
KAISER	$\alpha = 2$	0.980 Hz	18.5 dB
	$\alpha = 4$	1.172 Hz	30.4 dB
CHEBYSHEV	$\beta = 1$	0.890 Hz	20.1 dB
	$\beta = 2$	1.172 Hz	40.5 dB
BUTTERWORTH ($f_c = 0.439$ Hz)	$N = 2$	0.793 Hz	18.2 dB
	$N = 3$	0.740 Hz	24.3-dB
	$N = 4$	0.731 Hz	28.8 dB
BUTTERWORTH ($N = 4$)	$f_c = 0.439$ Hz	0.731 Hz	28.8 dB
	$f_c = 1.500$ Hz	2.815 Hz	28.8 dB
	$f_c = 2.500$ Hz	4.720 Hz	28.8 dB

CONCLUSIONS

We've shown that the Butterworth window can be obtained by the conventional Butterworth filter design procedure. This window is able to control the 3-dB bandwidth and sidelobe attenuation independently by two parameters, the cutoff frequency and the order of the filter. As such, the sidelobe attenuation can be varied even if the 3-dB bandwidth is fixed, and vice versa. Therefore the tradeoff problem between the frequency resolution and spectral leakage in the estimated PSD, unavoidable with the conventional windows, can be solved by the Butterworth window.

ACKNOWLEDGMENT

The authors thank Associate Editor Richard (Rick) Lyons for his kind help in preparation of this article.

AUTHORS

Tae Hyun Yoon (thyoon@ee.knu.ac.kr) is a Ph.D. candidate at the School of Electrical Engineering and Computer Science, Kyungpook National University, Daegu, Korea.

Eon Kyeong Joo (ekjoo@ee.knu.ac.kr) is a professor at the School of Electrical Engineering and Computer Science,

Kyungpook National University, Daegu, Korea.

REFERENCES

[1] F. J. Harris, "On the use of windows for harmonic analysis with the discrete Fourier transform," *Proc. IEEE*, vol. 66, no. 1, pp. 51–83, Jan. 1978.

[2] S. Kay and S. Marple, "Spectrum analysis-a modern perspective," *Proc. IEEE*, vol. 69, no. 11, pp. 1380–1419, Nov. 1981.

[3] M. H. Hayes, *Statistical Signal Processing and Modeling*. New York: Wiley, 1996.

[4] B. P. Lathi, *Modern Digital and Analog Communication Systems*, 3rd ed., New York: Oxford Univ. Press, 1998.

[5] N. Geckinli and D. Yavuz, "Some novel windows and a concise tutorial comparison of window families," *IEEE Trans. Acous. Speech Signal Processing*, vol. ASSP-26, no. 6, pp. 501–507, Dec. 1978.

[6] S. Orfanidis, *Introduction to Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1996.

[7] S. Mitra and J. Kaiser, *Handbook for Digital Signal Processing*. New York: Wiley, 1993.

[8] S. Bergen and A. Antoniou, "Design of ultraspherical window functions with prescribed spectral characteristics," *EURASIP J. Appl. Signal Process.*, vol. 2004, no. 13, pp. 2053–2065, Oct. 2004.

[9] A. Nuttall, "Some windows with very good sidelobe behavior," *IEEE Trans. Acous. Speech Signal Processing*, vol. 29, no. 1, pp. 84–91, Feb. 1981.

[10] G. Andria, M. Savino, and A. Trotta, "Windows and interpolation algorithms to improve electrical measurement accuracy," *IEEE Trans. Instrum. Meas.*, vol. 38, no. 4, pp. 856–863, Aug. 1989.

[11] G. Andria, M. Savino, and A. Trotta, "FFT-based algorithms oriented to measurements on multifrequency signals," *Measurement*, vol. 12, no. 1, pp. 25–42, Dec. 1993.

[12] J. Gautam, A. Kumar, and R. Saxena, "On the modified Bartlett–Hanning window (family)," *IEEE Trans. Signal Processing*, vol. 44, no. 8, pp. 2098–2102, Aug. 1996.

[13] H. Baghaei, W. Wong, H. Li, J. Uribe, Y. Wang, M. Aykac, Y. Liu, and T. Xing, "Evaluation of the effect of filter apodization for volume PET imaging using the 3-D RP algorithm," *IEEE Trans. Nucl. Sci.*, vol. 50, no. 1, pp. 3–8, Feb. 2003.

[14] S. Su, C. Yeh, and C. Kuo, "Structural analysis of genomic sequences with matched filtering," in *Proc. 25th Ann. Int. Conf. IEEE Engineering in Medicine and Biology Society*, Sept. 2003, vol. 3, pp. 2893–2896.

[15] P. Welch, "The use of fast Fourier transform for the estimation of power spectra : A method based on time averaging over short, modified periodograms," *IEEE Trans. Audio Electroacous.*, vol. AU-15, no. 2, pp. 70–73, June 1967.

[16] T. Yoon and E. Joo, "Butterworth window for power spectral density estimation," *ETRI J.*, vol. 31, no. 3, pp. 292–297, June 2009.

[SP]

[spotlight REPORT] continued from page 15

of which are donating hardware and some funding. "We have no intellectual property. Everything is open-source. We're not allowed to patent anything. Companies like that; it makes it easier to collaborate with them."

WHERE IS EVERYBODY?

Has anyone found anything? Nothing has been confirmed, but there have been some false terrestrial alarms and artificially produced extraterrestrial signals.

Probably the best known signal picked up by SETI researchers so far was at Ohio State in 1977 when they received what became known as the "Wow!" signal. As an Ohio State astronomer was checking printouts, he found a sharp, clear signal that even turned on and off during the period it was observed. The astronomer wrote "Wow!" in the margin of the printout. But the signal was never heard again.

Why wouldn't aliens respond to the signals we have already been broadcasting for years?

Today, some SETI scientists are less certain about the narrowband approach to the search for alien civilizations.

Shostak says that several years ago, when wideband techniques, such as spread spectrum, were coming into wide use, SETI scientists began to wonder if aliens were sending out spread-spectrum signals. "In which case, we're not going to find them." While the wideband issue continues to be a concern, he thinks there might still be a very narrowband component to the signal just to get your attention.

Shostak notes that we have been broadcasting seriously into space for maybe 70 years and that means those early broadcasts are 70 light years out. "So, if you're running SETI experiments today and looking for responses, then the aliens can't be more than 35 light years away. You need enough time for 'Howdie Doody' to get to them and for them to get back to us." Shostak estimates those parameters give us access to a couple of thousand stars at most—out of a couple of hundred billion stars in the galaxy.

And while most astronomers are interested in finding other planets, Shostak believes that recent reports from European astronomers identifying 32 new planets orbiting stars outside our solar system doesn't improve the chances that we'll find intelligent life on those planets. "The number of stars in the Milky Way is on the order of 100 billion. So another 32 planets doesn't mean much to the SETI community."

"What you want to know," he says, "is what fraction of these planets are most like Earth. Unfortunately, these planets are never like Earth because those are hard to find. It will take a couple of years to find Earth-like planets, but within about a thousand days [by approximately the end of 2012), we will know what percent of stars have planets that are sort of like Earth, with liquid water and atmospheres. That's what the Kepler mission is designed to do—find Earth-like planets"

WHAT'S NEXT?

The search goes on. In addition to the vastly improved radio searches—mainly the ATA—SETI is looking for signals that might be sent at visible wavelengths or in the infrared. Experiments at UC-Berkeley at Santa Cruz, Berkeley, and Harvard are using relatively large, conventional mirror telescopes to hunt for very brief flashes of light (presumably from high-powered lasers) that other civilizations might be beaming our way.

At the same time, Dr. Jill Tarter, the director of the SETI Institute and the winner of the 2009 Technology, Entertainment, and Design Award, says she plans to open-source the ATA'a detection algorithms to advance the search so engineers around the world can help improve them and, in the process, develop a new generation of SETI enthusiasts.

In fact, Tarter has a wish list that includes a massive outreach to grow the TeamSETI network, including finding and recruiting engineers with expertise in digital signal processing. Another item high on her list is funding—to acquire additional telescopes for the Allen array.

[SP]

Jian Li and Petre Stoica

The Phased Array Is the Maximum SNR Active Array

A "folk theorem" in the radar community states that the phased array (PA) maximizes the signal-to-noise ratio (SNR) at a given focal point over the class of possible active arrays. In this article, we prove this theorem under various conditions.

RELEVANCE

Active arrays are widely used in many sensing applications, including radar, sonar, and medical imaging. It is often stated without proof that the PA maximizes the SNR at a given focal point over the class of possible active arrays. The proof here of this statement, under various conditions, provides new insight into the use of active arrays in diverse sensing applications. Courses that may benefit from this article include array signal processing, radar, sonar, and medical imaging.

PREREQUISITES

It is assumed that the reader is familiar with basic linear algebra and optimization principles and has some array signal processing background.

PROBLEM STATEMENT

Consider a transmit array with N sensors in an active sensing application. Without loss of generality, we focus on radar applications with antennas as sensors. We wish to determine the transmit strategy that maximizes the power of the signal at a focal point under the constraint that the maximum power each antenna can transmit is c.

NARROWBAND SCENARIO WITHOUT PROPAGATION ATTENUATION

Let

$$\mathbf{a} = [e^{-j\phi_1} \ \cdots \ e^{-j\phi_N}]^T \tag{1}$$

Digital Object Identifier 10.1109/MSP.2009.935418

be the steering vector of the array for the given focal point of interest, which determines the phase shifts $\{\phi_n\}_{n=1}^N$, where $(\cdot)^T$ denotes the transpose. Let $x_n(t)$ denote the signal emitted by the nth antenna ($n = 1, \ldots, N$). Then the power of the signal at the focal point is

$$\mathbf{a}^*\mathbf{R}\mathbf{a}, \tag{2}$$

where $(\cdot)^*$ denotes the conjugate transpose and the (k, n)th element of $\mathbf{R}$ is defined as

$$R_{kn} = E[x_k(t)x_n^*(t)]. \tag{3}$$

So maximizing the SNR under the elemental power constraint is equivalent to

$$\max_{\mathbf{R}\geq 0} \ \mathbf{a}^*\mathbf{R}\mathbf{a} \ \text{ s.t. } \ R_{nn} \leq c, \ n = 1, \ldots, N, \tag{4}$$

where c denotes the maximum elemental power allowed and $\mathbf{R} \geq 0$ means that $\mathbf{R}$ is a positive semidefinite matrix. (We could also consider a total power constraint, such as $\text{tr}(\mathbf{R}) \leq Nc$, where $\text{tr}(\cdot)$ denotes the trace of a matrix, but the elemental power constraint is more practical.)

To solve (4), we note that

$$\mathbf{a}^*\mathbf{R}\mathbf{a} \leq \lambda_{\max}(\mathbf{R})\|\mathbf{a}\|^2 \leq \text{tr}(\mathbf{R})N \leq N^2c, \tag{5}$$

where $\|\cdot\|$ denotes the Euclidean norm of a vector. The upper bound in (5) is achieved by

$$\mathbf{R} = c\,\mathbf{a}\mathbf{a}^*, \tag{6}$$

which satisfies the constraint ($R_{nn} = c, \forall n$) and therefore is the optimal design. Because (6) corresponds to a PA, the proof of the theorem is concluded. Furthermore, it follows from (5) that for a maximum-SNR design, all equalities in (5) must hold; in particular, we must have $\lambda_{\max}(\mathbf{R}) = \text{tr}(\mathbf{R})$ so that $\text{rank}(\mathbf{R}) = 1$. This observation implies that the optimal-SNR PA design in (6) is the *unique* solution to (4) (modulo a common phase shift of all elements of $\mathbf{a}$).

WIDEBAND SCENARIO WITHOUT PROPAGATION ATTENUATION

In this case, the signal at the focal point is

$$x_1(t-\phi_1) + \cdots + x_N(t-\phi_N), \tag{7}$$

where, for a calibrated array and a specified focal point, the delays $\{\phi_n\}_{n=1}^N$ are known. Therefore, the power at the focal point is

$$E\left[\left|\sum_{n=1}^N x_n(t-\phi_n)\right|^2\right]. \tag{8}$$

Maximizing (8) with respect to $\{x_n(t)\}$, under the constraint that $E[|x_n(t)|^2] \leq c$, looks like a rather different problem from (4), but in actuality it is quite similar. To see this, let

$$\mathbf{u} = [1 \ \cdots \ 1]^T, \tag{9}$$

and

$$\Gamma_{kn} = E[x_k(t-\phi_k)x_n^*(t-\phi_n)]. \tag{10}$$

Then the problem is

$$\max_{\mathbf{\Gamma}\geq 0} \ \mathbf{u}^*\mathbf{\Gamma}\mathbf{u} \ \text{ s.t. } \ \Gamma_{nn} \leq c, \ \ n = 1, \ldots, N, \tag{11}$$

which has the same form as (4). Consequently, the solution to (11) is $\mathbf{\Gamma} = c\mathbf{u}\mathbf{u}^*$ or in terms of the signals

$$x_n(t) = s(t+\phi_n), \ \ n = 1, \ldots, N, \tag{12}$$

where $s(t)$ is any stationary signal with power equal to c. This proves the fact that, once again, the PA is SNR optimal; the maximum achievable SNR is still N^2c.

NARROWBAND SCENARIO WITH PROPAGATION ATTENUATION

In this case,

$$\mathbf{a} = [\rho_1 e^{-j\phi_1} \ \cdots \ \rho_N e^{-j\phi_N}]^T, \quad (13)$$

where $\rho \in \mathbb{C}$ $(n = 1, \ldots, N)$ are the attenuation coefficients, and the optimization is still (4). Let $\mathbf{Q}$ be a square root of $\mathbf{R}$, that is

$$\mathbf{R} = \mathbf{Q}^*\mathbf{Q}. \quad (14)$$

Using $\mathbf{Q}$, we can reformulate the problem in (4) as

$$\max_{\mathbf{Q}} \|\mathbf{Qa}\|^2, \ \|\mathbf{q}_n\|^2 \le c, \quad n = 1, \ldots, N, \quad (15)$$

where $\{\mathbf{q}_n\}$ are the columns of $\mathbf{Q}$:

$$\mathbf{Q} = [\mathbf{q}_1 \ \cdots \ \mathbf{q}_N]. \quad (16)$$

Because

$$\begin{aligned}\|\mathbf{Qa}\| &= \left\|\sum_{n=1}^{N} \rho_n e^{-j\phi_n}\mathbf{q}_n\right\| \\ &\le \sum_{n=1}^{N} |\rho_n| \, \|\mathbf{q}_n\| \\ &\le \sqrt{c}\sum_{n=1}^{N}|\rho_n|, \end{aligned} \quad (17)$$

it follows that

$$\mathbf{a}^*\mathbf{Ra} \le c\left(\sum_{n=1}^{N}|\rho_n|\right)^2. \quad (18)$$

The upper bound in (18) is achieved at

$$\mathbf{R} = c\begin{bmatrix} e^{-j(\phi_1+\psi_1)} \\ \vdots \\ e^{-j(\phi_N+\psi_N)}\end{bmatrix}\left[e^{j(\phi_1+\psi_1)} \ \cdots \ e^{j(\phi_N+\psi_N)}\right], \quad (19)$$

where $\rho_n = |\rho_n|e^{-j\psi_n}$, $n = 1, \ldots, N$. Furthermore, the optimal $\mathbf{R}$ above is unique, which can be seen by using the fact that the first inequality in (17) becomes equality if and only if the vectors $\{\mathbf{q}_n\}$ are parallel to one another.

Evidently, the result proved for this narrowband with attenuation case is more general than the one presented for the narrowband without attenuation scenario, to which it reduces for $\rho_n = 1$ $(n = 1, \ldots, N)$. However, we preferred to prove these results separately because the proof for the narrowband without attenuation case is simpler, yet stronger than the one above: in particular, the proof for the narrowband without attenuation case can deal with the constraint $\mathrm{tr}(\mathbf{R}) \le Nc$ without any modification, unlike the proof above.

WIDEBAND SCENARIO WITH PROPAGATION ATTENUATION

The power of the signal at the given focal point is now

$$E\left[\left|\sum_{n=1}^{N}\rho_n x_n(t-\phi_n)\right|^2\right]. \quad (20)$$

Therefore, the maximum-SNR design problem is, analogously to (11)

$$\max_{\Gamma \ge 0} \mathbf{v}^*\boldsymbol{\Gamma}\mathbf{v} \ \ \text{s.t.} \ \ \boldsymbol{\Gamma}_{nn} \le c, \quad n = 1, \ldots, N, \quad (21)$$

where

$$\mathbf{v} = [\rho_1 \ \cdots \ \rho_N]^*. \quad (22)$$

The proof for the narrowband with attenuation case then implies that the maximum possible power is equal to

$$c\left(\sum_{n=1}^{N}|\rho_n|\right)^2, \quad (23)$$

and that this power is achieved by the PA

$$x_n(t) = e^{j\psi_n}s(t+\phi_n), \quad n = 1, \ldots, N, \quad (24)$$

where $E[|s(t)|^2] = c$ (otherwise the stationary signal $s(t)$ can be arbitrarily chosen).

WHAT WE HAVE LEARNED

In many active sensing applications, such as in radar, the SNR is critical. The analysis above sheds some light on the usefulness of PA when SNR is the main concern. In the presence of strong clutter and jammer, however, the SNR may not be the most critical factor and multiple input multiple output radar, which can transmit via its antennas multiple probing signals that are quite different from each other, is likely to play an important role in performance enhancement [1]–[3].

ACKNOWLEDGMENTS

This work was supported in part by the Office of Naval Research under grant N00014-07-1-0293, the Swedish Research Council, and the European Research Council. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

AUTHORS

Jian Li (li@dsp.ufl.edu) received the M.Sc. and Ph.D. degrees in electrical engineering from The Ohio State University (OSU), Columbus, in 1987 and 1991, respectively. In 1991, she was an adjunct assistant professor with the Department of Electrical Engineering at OSU. From 1991 to 1993, she was an assistant professor with the Department of Electrical Engineering, University of Kentucky, Lexington. Since 1993, she has been a professor in the Department of Electrical and Computer Engineering, University of Florida, Gainesville. Her research interests include spectral estimation, statistical and array signal processing, and their applications. She is a Fellow of the IEEE.

Petre Stoica (ps@it.uu.se) received the D.Sc. degree in automatic control from the Polytechnic Institute of Bucharest (BPI), Romania, in 1979 and an honorary doctorate degree in science from Uppsala University (UU), Sweden, in 1993. He is a professor of systems modeling with the Division of Systems and Control, the Department of Information Technology, UU. He was a professor of system identification and signal processing with the Faculty of Automatic Control and Computers, BPI. His scientific interests are in the areas of system identification, time series analysis and prediction, statistical signal and array processing, spectral analysis, wireless communications, and radar signal processing. He is a Fellow of the IEEE.

REFERENCES

[1] J. Li and P. Stoica, "MIMO radar with colocated antennas: Review of some recent work," *IEEE Signal Processing Mag.*, vol. 24, pp. 106–114, Sept. 2007.

[2] A. H. Haimovich, R. S. Blum, and L. J. Cimini, "MIMO radar with widely separated antennas," *IEEE Signal Processing Mag.*, vol. 25, pp. 116–129, Jan. 2008.

[3] J. Li and P. Stoica, Eds., *MIMO Radar Signal Processing*. Hoboken, NJ: Wiley, 2009.

[SP]

H.J. de Wind, J.C. Cilliers, and
P.L. Herselman

DataWare: Sea Clutter and Small Boat Radar Reflectivity Databases

Please send suggestions for Web resources of interest to our readers, proposals for columns, as well as general feedback, by email to Dong Yu ("Best of the Web" associate editor) at dongyu@microsoft.com.

In this issue, "Best of the Web" presents online radar reflectivity databases, which are free of charge to the international research community. Databases such as these are used for the characterization of interference (including noise, sea clutter and land clutter, among others) in radar systems, and the development of signal processing techniques that reduce the detrimental effects of this unwanted interference. The data can also be used for the development and evaluation of signal processing techniques aimed at detecting and tracking man-made objects by separating targets from interference based on Doppler and amplitude characteristics. These signal processing techniques include moving target indication (MTI), pulse-Doppler, space-time adaptive processing, track-before-detect (TkBD), and constant false alarm rate (CFAR) processing.

The databases described in this column are specifically designed to support continued development of statistically accurate sea clutter models and radar detection and tracking schemes used in a maritime environment. Statistical models of sea clutter as well as the performance of the detection and tracking schemes are highly dependent on environmental conditions, the geometry of the radar deployment site relative to the area of interest, and the radar system operating parameters. Consequently, it is of paramount importance that the required database contains sea clutter data recorded with varying geometries, under varying environmental conditions, with a range of radar system parameters and targets of interest performing various maneuvers in this environment.

Two databases that meet the above-mentioned requirements are available to the international research community. The McMaster Intelligent PIXel Processing Radar (IPIX) database was published in 2001 and the Council for Scientific and Industrial Research (CSIR) database, published in 2007. Although these databases still only cover a limited set of environmental conditions, relative geometries, radar characteristics, and targets of interest, they already contain a significant amount of suitable data.

IPIX DATABASE

In 2001, McMaster University in Hamilton, Canada published a set of sea clutter radar data on the Web. This database can be found at the following two addresses:
http://soma.crl.mcmaster.ca/ipix/
http://soma.mcmaster.ca/ipix.php.

MOTIVATION FOR RECORDING THE DATA

The study conducted by the Cognitive Systems Laboratory at McMaster University involved the effects of sea state and radar orientation with respect to wave direction on amplitude and frequency modulation and on the width and shape of its short-time Fourier transform [1].

MEASUREMENT TRIALS

The available data were recorded during two measurement trials. The first was in November 1993 near Dartmouth, Nova Scotia, and the second was in 1998 in Grimsby, on the shore of Lake Ontario. The first trial focused on obtaining data sets of sea clutter only. During the second trial, the focus was specifically on the presence of known floating objects of various sizes. The data recorded during both trials were measured with the McMaster IPIX radar, a fully coherent X-band radar with features such as dual transmit/receive polarization, frequency agility, and stare/surveillance mode [2]. This database has been used extensively for sea-clutter characterization [3] by the international radar community, with the number of publications that reference this database exceeding 200.

CURRENT LIMITATIONS

As mentioned earlier, the IPIX database, however, only covers a limited set of environmental conditions, relative geometries, and radar characteristics. The range from the radar is, for example, limited to 500–8,000 m and the grazing angle is limited to 0.2–3.5°. Another limitation to the development of detection and tracking algorithms is the lack of radar reflectivity data and geometric information for maneuvering maritime vessels. The environmental conditions, such as the wind and wave conditions, are only supplied for one measurement trial and are only provided in raw format.

SUPPORTING RESOURCES ON THE WEB SITE

The Web site does not only contain the data, but it also has supporting information, such as the radar parameters, a radar systems tutorial, a list of publications that references the database, and a page for frequently asked questions.

Digital Object Identifier 10.1109/MSP.2009.935415

Helping to improve the radar detection accuracy. Cartoon by Tayfun Akgul (tayfun.akgul@ieee.org).

Figures that give a quick overview of the content of the data set are provided for each data set. The recorded environmental conditions are provided in raw format, but only for the first measurement trial. The Web site also contains maps and pictures of the deployment sites.

CSIR DATABASE

In 2006 and 2007 the Defence, Peace, Safety, and Security Unit of the CSIR in Pretoria, South Africa, conducted two sea-clutter measurement trials to address some of the limitations of the IPIX database. These data were made available on the Web at the end of 2007 for the first trial and in the middle of 2009 for the second trial as an additional data source to the IPIX database. This database can be found at

http://www.csir.co.za/small_boat_detection/.

MOTIVATION FOR RECORDING THE DATA

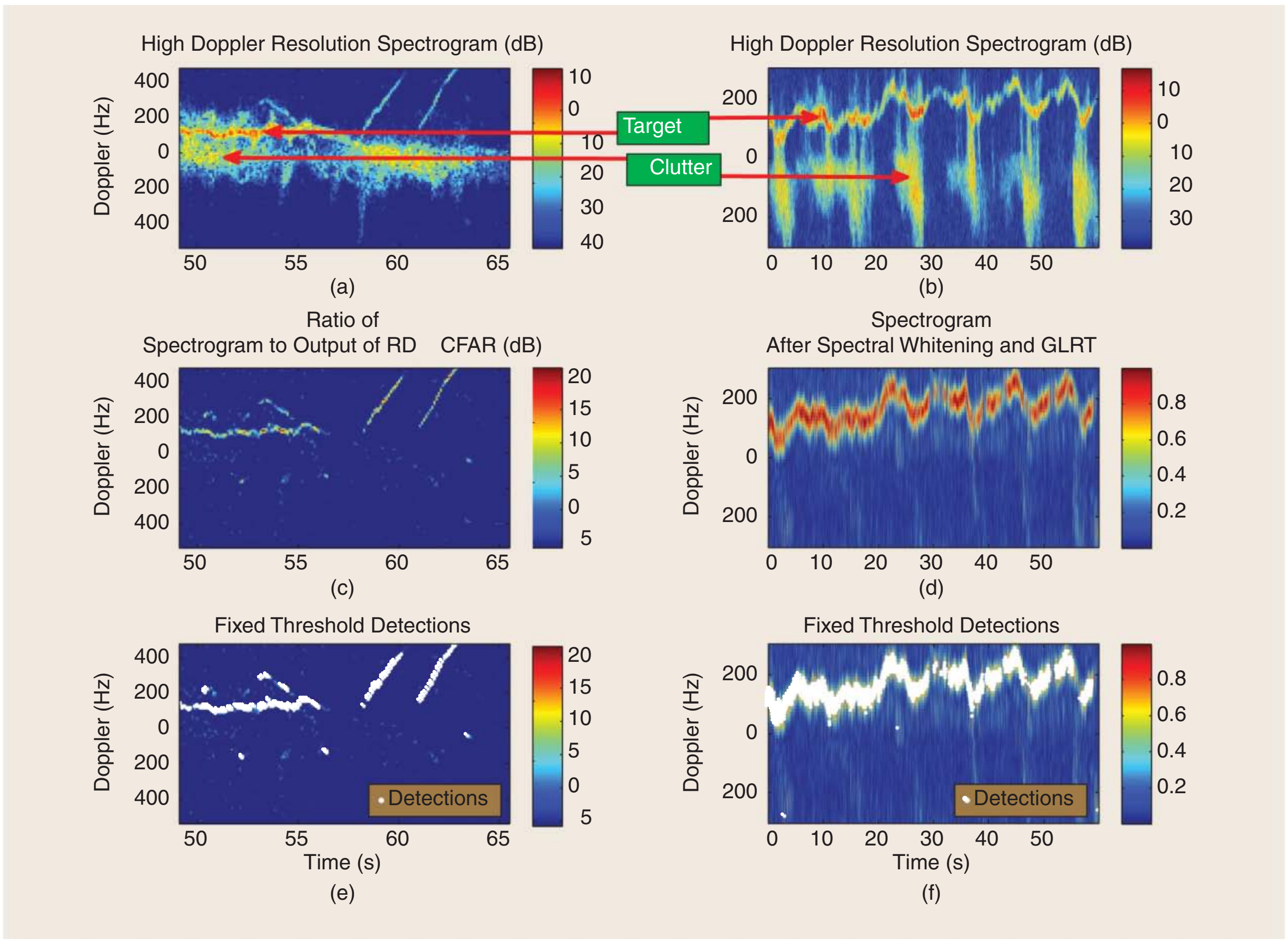
The data available in the CSIR database were recorded with the purpose of aiding in the development of a persistent ubiquitous maritime surveillance system using radar as the primary sensor. This database can be used to address several of the radar-related signal processing topics mentioned earlier, but more specifically those related to the detection and tracking of small boats in a maritime environment, including the littoral. To develop detection and tracking techniques applicable to the maritime environment, it is essential to characterize the sea clutter and to characterize the return from objects of interest. Estimation theory can be applied, for example, to determine the amplitude statistics and Doppler characteristics of the sea clutter and to develop a process to perform these estimations in real time. Detection theory can be applied to discern between the return from sea clutter and the return from other objects. These objects may include targets of interest, such as small boats, or objects that are not of interest (such as birds). In a nonstationary environment, detectors are designed to carry out this process adaptively to obtain a CFAR. Declared detections that are deemed to be of interest (based on position, radial speed, or amplitude, for example) may be extracted from the output of the detection algorithm whereby spurious and interfering signals are discarded. Tracks are then formed on the detections of interest. This tracking involves the association of detections with existing tracks or creating new tracks with detections that cannot be associated with any existing track, smoothing/filtering of all the tracks, estimating the speed and heading of the targets, predicting the new position, and providing estimates on the errors in these predictions. Another approach that can be evaluated on the available data is the concept of tracking a signal before it is declared as a target. In this approach, called TkBD, the sensor data originating from a tentative target are integrated over time. This integration may yield detections in cases where the amplitude return from the target at any particular time instance is such that, when compared to the sea clutter amplitude return, detections will not be obtained with more prevalent detection algorithms.

The foregoing discussion demonstrated that a large number and variety of signal processing techniques [4] can be applied to and evaluated on the data available in the CSIR database. Example images of detection techniques evaluated on data sets contained in the CSIR database are shown in Figure 1.

MEASUREMENT TRIALS

The data contained in the CSIR database were recorded during two measurement trials. Both were in the Western Cape, South Africa. The first was in July 2006 at the Overberg Test Range near Arniston [5] and the second was on Signal Hill in Cape Town [6]. A significant library of both sea clutter data and radar reflectivity data from various small maritime vessels were recorded. The radar used during the first trial was a coherent radar cross section measurement facility with an operating frequency from 6.5 GHz to 17.5 GHz transmitting pulsed continuous wave waveforms, with a pulse-to-pulse frequency agility bandwidth of 500 MHz. An experimental X-band, pulse-Doppler radar was used during the second trial.

In addition to the radar data, the weather data (wind, temperature, and rain) for the area of interest were obtained from the South African weather services and were recorded at a local weather sensor at the deployment site (Table 1). The local wave direction, significant wave height, and wave period were logged with wave buoys deployed close to the deployment site. A differential processing global positioning system (GPS) receiver (3–5 m absolute accuracy) was installed on the cooperative vessels used during the measurement trials, to enable the estimation of ground truth tracks of these vessels for use in processing, specifically for the evaluation of detection and



[FIG1] In (a)–(f), detections obtained with range-Doppler CFAR and spectral whitening detectors are given.

tracking algorithms. Video cameras were mounted alongside the radar antenna. These cameras were aligned with the boresight of the radar antenna. The video output of both cameras (wide and narrow fields of view) was recorded to complement the radar reflectivity data, especially when used as an aid in explaining phenomena observed in the radar data. All the above-mentioned peripheral data were included in the data sets available in the database (video available for both, but only published for the second trial).

CURRENT LIMITATIONS

The data available from this database is a useful addition to the IPIX database as it extends the set of environmental conditions covered (higher sea states and wind speeds), the maximum range, and grazing angles. In addition, data are also available of cooperative maneuvering maritime vessels (with recorded GPS track) and not only of floating objects.

Even this large database only covers a limited set of environmental conditions, relative geometry and radar characteristics. The data in this database were for example only recorded with vertical polarization on both transmit and receive (VV). The Defence, Peace, Safety and Security (DPSS) operating unit of the CSIR is therefore planning more measurement trials and will continue to update this database as new data become available.

[TABLE 1] SUMMARY OF RELATIVE GEOMETRY AND ENVIRONMENTAL CONDITIONS DURING CSIR MEASUREMENT TRIALS.

	OVERBERG	SIGNAL HILL
GEOMETRY		
RADAR HEIGHT (AMSL)	67 M	294 M
DISTANCE FROM COASTLINE	1.2 KM	1.25 KM
AZIMUTH COVERAGE OF SEA	90° N TO 225° N	240° N TO 20° N
GRAZING ANGLES	3° TO 0.3°	10° TO 0.3°
MAXIMUM RANGE	15 KM	60 KM
ENVIRONMENTAL CONDITIONS		
AVERAGE WIND SPEED	0–20 KTS	0–40 KTS
MAXIMUM WIND GUST	40 KTS	60 KTS
PREDOMINANT WIND DIRECTION	180° N–270° N	130° N–140° N AND 320° N–330° N
SIGNIFICANT WAVE HEIGHT	1 M–3.8 M	1 M–6 M
MAXIMUM WAVE HEIGHT	7.31 M	11.26 M
SWELL DIRECTION	135° N–180° N	230° N–270° N

Even though both databases have data sets of antenna scans over a given sector, neither have data suitable for the development of track-while-scan algorithms for surveillance radars. It is possible to extract sections of data that will emulate the effect of scanning, but it will not include effects such as inter clutter modulation and the influence of adjacent azimuth directions on the estimation and detection processes.

SUPPORTING RESOURCES ON THE WEB SITE

The Web site provides supporting resources, such as information on the deployment sites, radar parameters, images of the boats that were used as cooperative targets, example data sets, and answers to frequently asked questions. A summary of each data set is also provided for quick reference. These summaries consist of an overview plot of the data set, a map of the GPS track of the cooperative target where applicable, a summary of the environmental conditions, radar setup and viewing geometry and the date, time, and duration of the data set.

CONCLUSION

The continuing development of models, detection schemes, and tracking algorithms for the maritime environment is highly dependent on the availability of suitable data. Hopefully, in the future, the free exchange of data from initiatives such as the IPIX and the CSIR databases will support further research and broaden the understanding of the complex subjects contained in this field.

AUTHORS

H.J. de Wind (rdewind@csir.co.za) is a researcher with the CSIR Defence, Peace, Safety, and Security Unit, Pretoria, South Africa.

J.C. Cilliers (jcilliers@csir.co.za) is a principal researcher with the CSIR Defence, Peace, Safety, and Security Unit, Pretoria, South Africa.

P.L. Herselman (paulh@rrs.co.za) is a senior systems engineer with Reutech Radar Systems, Stellenbosch, South Africa.

REFERENCES

[1] S. Haykin, R. Bakker, and B. Currie, “Uncovering nonlinear dynamics—The case study of sea clutter,” *Proc. IEEE,* vol. 90, no. 5, pp. 860–881, 2002.

[2] S. Haykin, C. Krasnor, T. Nohara, B. Currie, and D. Hamburger, “A coherent dual-polarized radar for studying the ocean environment,” *IEEE Trans. Geosci. Remote Sens.ing, vol.* 29, no. 1, pp. 189–191, 1991.

[3] M. Greco, F. Gini, A. Younsi, M. Rangaswamy, and A. Zoubir, “Non-stationary sea clutter: Impact on disturbance covariance matrix estimate and detector CFAR,” in *Proc. 2008 Int. Conf. Radar,* 2008, pp. 558–562.

[4] P. L. Herselman and H. J. de Wind, “Improved covariance matrix estimation in spectrally inhomogeneous sea clutter with application to adaptive small boat detection,” in *Proc. 2008 Int. Conf. Radar,* 2008, pp. 94–99.

[5] P. L. Herselman and C. J. Baker, “Analysis of calibrated sea clutter and boat reflectivity data at C- and X-band in South African coastal waters,” in *Proc. 2007 Int. Conf. Radar*, 2007.

[6] P. L. Herselman, C. J. Baker, and H. J. De Wind, “An analysis of X-band calibrated sea clutter and small boat reflectivity at medium-to-low grazing angles,” *Int. J. Navig. Observ.*, vol. 2008, 2008. [SP]

continued from page 2

multidisciplinary field. Together with the contributions from other disciplines, signal processing forms an integral yet fundamental tool for developing rich sets of scientific and engineering techniques to address a wide range of critical societal needs including health care, energy systems, sustainability, transportation, visualization, entertainment, education, communication, collaboration, defense, and security. The techniques established are frequently inspired by diverse scientific disciplines. As I was contemplating the content of this editorial, I was also attending the Neural Information Processing Systems (NIPS) Workshop, listening to a keynote speech by a prominent signal processing expert. The topics were cognitive radar/radio and nonlinear filtering, pertinent to how human brains (visual, auditory, and motor cortex) perform similar styles of computations. The cognitive radar/radio system design as presented also demonstrated the leading role that signal processing plays in fostering optimal design methodologies of complex engineering systems with “perception,” adaptation, action, and feedback from the environment. While NIPS is known to be a premier conference for machine learning and neural network researchers, signal processing is a frequent subject throughout the conference as I noted, not only as a technical methodology but also as an important application area covering diverse information sources such as audio, speech, image, video, and biomedical signals.

In light of the expanded focus of interest that considerably spans multidisciplinary areas in terms of both “signal” types and “processing” styles, this magazine has a more challenging role to play for educating the readers in bridging the knowledge gap across various technical subfields. In addition, because the contributions of signal processing are often in conjunction with other scientific and engineering disciplines, bridging the gap with technical fields not traditionally associated with signal processing is also becoming important. To address this challenge, the Publications Board of our Society recently established overview articles in each of the Society’s transactions while mandating *IEEE Signal Processing Magazine* with the major role of publishing tutorial articles educating the readers with little background in a given field but with main background in the related fields. The magazine’s tutorial articles serve the purpose of what can be best characterized as cross-fertilization over multiple technical subareas.

We warmly welcome contributions from you with the tutorial articles meeting the new challenges in face of the new, expanded focus of interest of our Society. [SP]

[illegible]

Gene Frantz, Jörg Henkel,
Jan Rabaey, Todd Schneider,
Marilyn Wolf, and Umit Batur

Ultra-Low Power Signal Processing

This *IEEE Signal Processing Magazine (SPM)* forum discusses the latest advances and challenges in ultra-low power (ULP) signal processing (SP). The forum members bring their expert insights to issues such as design requirements and future applications of ULP SP systems. The invited forum members are Gene Frantz (Texas Instruments), Jörg Henkel (Karlsruhe Institute of Technology), Jan Rabaey (University of California at Berkeley), Todd Schneider (ON Semiconductor), and Marilyn Wolf (Georgia Institute of Technology). The moderator of the forum is Umit Batur (Texas Instruments). Our readers may agree or disagree with the ideas discussed next. In either case, we invite you to share your comments with us by e-mailing batur@ti.com or spm.columns.forums@gmail.com.

Moderator: Let's first start by defining our topic. What does "ULP SP" mean? What specific requirements should a signal processing system satisfy to be called an "ULP" system?

G. Frantz: This is a fun question as it means different things to different people. I wrote an internal white paper on this topic a couple of years ago so we could, as a company, minimize the confusion on what it means. Here is an excerpt from the paper:

> The easy way to differentiate ULP from other power aware concepts is to create a chart on the priority order of the basic aspects: performance, price and power dissipation. Here is a description of my version of that chart.
>
> - *High-performance* devices are those where performance is the primary priority, if not the only priority.
> - *Low-power* devices are those where, given no performance is sacrificed, the focus is on how much the power dissipation can be reduced.
> - *ULP* devices are those where, given the absolute minimal power dissipation is achieved, how much performance is left? Then the focus is on performance.

So, I see ULP as a design philosophy rather than a specification.

J. Henkel: An ULP signal processing system should address low power consumption at all levels of abstraction i.e., it should apply the latest power efficient silicon technology to start with, it should utilize the most advanced power management techniques at OS-level, and, don't forget, the application itself should be trimmed to low power consumption. Especially algorithmic transformations at the application level can be very power efficient in signal processing systems. In summary, exploiting the potentials in power savings at each (or at least many) level makes to my mind a real ULP system. Therefore, designing an ULP signal processing system is a combined hardware/software problem.

G. Frantz: All aspects of a system should be involved: process, transistor, gate, hardware architecture, software, and system.

T. Schneider: We, at ON Semiconductor, design single-chip digital SP (DSP) systems that are almost always battery powered, so for us "ULP signal processing" means meeting demanding battery lifetime constraints while simultaneously accomplishing a demanding signal processing task. Thus, the prime metric is really one of efficiency: in a given, real world application, minimize the power consumed by the DSP system for a given task. We use *m*W/MIPS, where MIPS are the actual instructions being executed for application and *m*W is the measured power consumption of the DSP system.

Specific applications have constraints driven by desired battery lifetimes and batteries that are used. As an example, a power-consumption constraint of 1 *m*W or less is common for hearing aids. Designers strive to get as much signal processing and the best overall audio quality they can within this power budget. Within this power budget, developers are expected to implement features like adaptive feedback reduction, dynamic range compression, noise reduction, and parameter selection based on environmental monitoring. For some smaller hearing aid applications, this power constraint can drop to as low as 500 *u*W.

The extreme end of the ULP signal processing we address is signal processing in implantable systems such as pacemakers, implantable defibrillators, cochlear implants, and neurostimulators. Typical functions included in an implantable device are low-pass filtering, level measurement, and threshold detection. Some implantable devices are designed for battery lifetimes of ten years with an average power consumption of less than 30 *u*W, including the therapy. This corresponds to a current of 10 *u*A from a typical battery. With the current required to deliver therapy, only a few *u*A average current remains for all of the electronics. Of course, static current is also key in implantable applications and minimizing this requires a tradeoff to be made between threshold voltage, supply voltage, and operating frequency.

To achieve the lowest possible dynamic power consumption, we strive

Digital Object Identifier 10.1109/MSP.2009.935417

to operate the signal processor(s) at the lowest possible operating voltage. This typically means minimizing the clock rate, which necessitates application driven design.

In our experience, the largest power savings can be derived from clever design of the signal processing algorithm(s) and from clever design of a system architecture onto which this algorithm can be efficiently mapped. In the late 1990s, we pioneered a novel approach that combines a flexible, software-programmable DSP with a more fixed function, microcoded "accelerator." This architecture recognizes that many DSP algorithms consist of a "vector number crunching" portion and a "control path/side-chain" portion. By partitioning an algorithm into these two portions with an efficient mapping, clock rates can be minimized. This allows the supply voltage to be minimized, which reduces power consumption. Properly executed, this approach provides an approximately linear reduction in power consumption. This approach makes a tradeoff between flexibility and power consumption, which strikes a compromise between "hard code everything in logic" (lowest possible power with zero flexibility) and "make everything programmable" (highest power and most flexibility). To fully realize the benefits of this approach, the two processing units must be run in parallel, which can increase programming complexity. Of course, there are applications (like implantables) where the power constraints are so stringent that there is no choice but to implement signal processing as fixed-function blocks.

In summary, for us, ULP signal processing means a focus on efficient algorithms and application-driven architectures both seeking to minimize the clock rate and operating voltage and thus realizing the lowest *m*W/MIPs in the intended application.

J. Rabaey: I tend to take a somewhat different tack, and tend to classify systems by their energy source or provision.

- *ULP systems*: self-contained–these systems either live off a single battery charge for the lifetime of the product, or scavenge energy
- *Low-power systems*: battery operated–need occasional battery replacement or recharge
- *Powered systems*: Performance is dominant. Energy-efficiency is important for either heat management or for cost reduction.

The actual boundaries between these different classes are variable depend upon the size of the node and the usage patterns. Observe that this definition is generic and extends beyond the signal processing label.

M. Wolf: I like Jan's definition, but what about devices like radio-frequency identification (RFID) that receive energy from an antenna?

J. Rabaey: Devices like RFID that receive energy from an antenna fall clearly under the ULP class. They are "perpetual" and harvest electromagnetic or magnetic energy.

G. Frantz: I think there is a whole family of products of which I call "perpetual devices" that beg the same question. But I think Jan covers it in his definition. What I read his definition to say is the "power source lasts longer than the useful life of the product."

J. Rabaey: I am perfectly in line with Gene on this one.

T. Schneider: We have an internal definition (more of a joke really) that we call "infinite battery life" that is along the same line as Gene's comment above. If the device has replaceable batteries and the user cannot remember the last time they replaced the batteries then the device has, in effect, infinite battery life.

Reading these definitions, which are more general than mine, I realize that many of the applications we address have a power source that is predetermined by size, reliability, availability, or the fact that it has been historically used. This is a hard constraint and ULP also means cramming as much processing capability in as possible while living within this constraint.

G. Frantz: That is why I call it a philosophy rather than a specification. But, back to Jan's "infinite battery life" rather than my "perpetual device," the problem with mine is you can't patent anything that starts with the word "perpetual."

Moderator: Which signal processing applications are driving the need for ULP? What are the most important tradeoffs in these ULP applications?

J. Rabaey: This is a tough one to answer, as there are many options. My top choice would be medical applications. Various wireless monitoring and implanted devices are pushing the limits on what ULP design is all about. In the future, I can see various immersed media devices becoming crucial as well. How about virtual reality on a mobile?

J. Henkel: I agree with Jan's points of medical and wireless but want to pick up his last point and extend it a bit to mobile multimedia in general. In a research project we are currently conducting, we are exploring the low-power (I hesitate to call it ULP) design space of a mobile multimedia device with respect to impacts on both the hardware architecture and the software architecture. So far, we ended up with a new hardware architecture that uses dynamic reconfiguration. On the other side, we had to heavily redesign the software architecture (in that case, an H.264) to exploit the hardware architecture's low power capabilities by, for instance, exploiting a high inherent degree of instruction-level parallelism. Coming back to the question on the tradeoffs, in our case the price to pay is to completely redesign the software architecture and to develop a novel hardware architecture.

M. Wolf: I think that sensor networks/cyber-physical systems are an important category. For example, people don't want to crawl through a building or bridge every year to replace the batteries in the sensors, which is what they have to do right now.

G. Frantz: So, to follow the thread above, Jan's definition (or my rewording) of ULP describes what markets it will be used. Those markets for which the batteries should last longer than the useful life of the product.

I was talking to a friend of mine in the world of implants. I suggested that implantables needed to be perpetual as they needed to last longer than the life of the individual. He said that it wasn't true, implants only needed to last ten years; not for the life of the human but to outlast the obsolescence of the technology. His point was that, independent of the battery life, technology needed to be replaced at least every ten years, if not sooner.

Marilyn's example of changing batteries in a bridge monitoring application gives a different perspective of the market. I argue there is another market where the product has been buried or placed in an unreachable location. For example, load cells at the bottom of structural columns, under highways, or in nuclear plants.

But to keep it simple, when I am asked the question about what market/product can use ULP, I explain that I don't know. It's kind of like fishing in a new hole: you don't know what bait to use and you don't know what you're going to catch, but it's exciting. I believe ULP will address new markets we have never thought of and that is what makes it exciting.

Now to discuss compromises: To guarantee the lowest possible power dissipation, one must tradeoff price and performance. Which begs the questions "what value does ULP bring to the party?" "Will people pay for it?" "How much performance will I need to eliminate?"

J. Henkel: I disagree that ULP is necessarily a question of the price one is willing to pay. You can get ULP devices quite cheap. An example: a few years back I talked to the developers of Texas Instruments' MSP430. Embedded in a system it can run for ten to 15 years with one battery i.e., without any recharge! This, of course, depends on the application and assumes that the processor only wakes up from time to time to, for instance, gather some data and to do some simple calculations and finally storing some of the data. A simple system built with this (or a comparable) processor is quite cheap, but according to some previous definitions in this discussion, it is an ULP design. Anyway, I agree that one needs to tradeoff performance when ULP is required.

G. Frantz: I understand questioning price. But my experience has been if you choose to make one variable fixed you need to tradeoff the other two. In this case, we choose to not compromise power dissipation, we will find we either need to reduce performance at the same price or increase price to increase performance. I get this from the idea that what Moore's law will continue to give us is more transistors. We can either increase performance or decrease power dissipation by adding more transistors.

Now having said that, Jörg's prime example of where that isn't the case is a good one. And there are many more. So, I'll be glad to not push it any further but ask a more interesting question: does ULP have perceived value in a product? Can a premium be placed on a product with a longer battery life?

T. Schneider: I agree with Jan's point about medical, especially implantable and body worn (e.g., hearing aids), applications. However, in applications that use wireless, the power savings gained through ULP signal processing can sometimes be overshadowed by the power consumption of the wireless subsystem. Still, short range wireless applications like body sensor networks are likely to be a future application where ULP signal processing will make a difference. This could evolve into a more general category of wearable or (semi) implantable electronics for both medical and recreational purposes.

FUTURE ULP SYSTEMS NEED TO BE ABLE TO FLEXIBLY ADAPT AT RUN TIME TO ALWAYS (OR MOST OF THE TIME) RUN AT MINIMUM POSSIBLE POWER.

In the multimedia area, some portable, body-worn audio designs (e.g., headsets and mobile phones) require ULP signal processing to make the design viable. In these cases, significant DSP resources are devoted to echo cancellation and noise reduction to compensate for the mechanical aspects of the device (e.g., a small form factor that locates the speaker close to a microphone or a design that is relies on DSP to compensate for other mechanical or acoustic features). Pushing more and more functionality into smaller and smaller devices and having them used in a wide range of environments is likely to place higher demands on ULP signal processing in these types of applications.

The most significant tradeoff in my experience is flexibility (and therefore programmability). Reduced programmability generally means less memory is used, which means smaller die. This drives lower cost at volume.

Jörg's point about dynamically reconfiguring hardware is an interesting one. This is one way to gain some flexibility in a power-efficient way without resorting to a fully software programming system.

In many ULP signal processing applications, performance tradeoffs are also made. These applications will often have "just enough performance." There is little or no "headroom" for new features or product expansion.

J. Rabaey: Just to address some of Todd's comments: It is indeed true that the wireless communication overshadows the signal processing budget. However, signal processing can help substantially to reduce the wireless communication power either by reducing the amount of data to be transmitted, or by making the RX/TX more efficient (in short distance wireless, the RX/TX power is way larger than the communication power).

With respect to the programmability aspect, I fully agree. Fully programmable solutions and ULP do not go together (unless you want to run extremely slow, such as the Michigan subthreshold processor). A combination of accelerators, reconfigurable/parameterizable modules and a rarely used simple core is the best option.

T. Schneider: Jan's point regarding the power reduction via ULP signal processing for wireless links is good and I completely agree. I was thinking more of standards based approaches like LE Bluetooth and Zigbee, but these can hardly be classified as ULP.

G. Frantz: We took a chart from Berkeley several years ago and added one point. The chart showed the amount of energy it took to transmit 1-b of information through various ways. We added a point (as I remember) to the chart. That point was how much energy it took to execute one instruction. The thought behind it was that it should be lower power to compress a bit then to transmit a bit. As I remember, the cost of compressing a bit was several orders of magnitude less then to transmit a bit. We did this in 2002, so the data is old. But I think the concept is still there. What I have tried to do is to change the idea from analog to digital (A/D) to analog to information (A/I). As we compress (convert) data into information, it will always be cheaper to transmit the information rather than the data. For example, in a security system at my home, the system doesn't need to transmit a picture of me as I walk through the house, it just needs to send a minimum set of information saying what I did.

Moderator: What are the most important ULP signal processing design techniques used today in semiconductor device fabrication, and processor, system, and software design?

M. Wolf: If one is designing an ULP system, one arguably doesn't want any software on the ULP device itself.

G. Frantz: On the contrary, the only way it will work in the IC world we are creating is for it to be programmable, or at least configurable, to be viable. I would agree, given today's memory technology, you are correct. So I would change your words to "how do we make memory ULP and nonvolatile." In fact, there are some technologies that might make this possible in the research labs now. One example is the FERAM.

So, let me go back to my first comment on the IC world. At each node we are making it more difficult to design and tool a new device. Once done, it is virtually free to manufacture devices. Jeff Bier has predicted that companies like Texas Instruments would only be able to create a handful of digital chips per year due to the prohibitive cost of design and tooling. That will force us to a few devices that can serve thousands of markets. My conclusion is that this results in programmable platforms on which innovation takes place.

I will make a bold statement and say that "ULP devices are not the innovations. They are the platforms on which innovation will happen to create a whole new world of products."

With that behind, the areas that most need to be dealt with for success in the product are: zero power memories, ULP wireless communications, energy scavenging, and analog.

M. Wolf: Memory isn't the only power hog in a programmable system—there's the I-box and the datapath. Of course, programmability doesn't necessarily mean Turing machine...

G. Frantz: Very good points. I also swept by, far too quickly, the concept of programmability. There is a spectrum of programmability from fixed function to infinitely programmable. Somewhere between these two extremes we find concepts like configurable and accelerator. Even a fixed function processor has two instructions (on and off), or should have. A lot of fun to be had by all.

J. Rabaey: I tend to agree with Gene that for ULP signal processing to be successful, a "reusable" platform needs to be created. The ad hoc approach does not fly. This means that we need to start thinking design methodology, libraries, reuse strategies, and common concepts. Innovation is cool for a while, but does not lead to mass impact. This is where the true challenge is. Gene's list is very good—I would add some: mechanical computing, passive computing, and bio-inspired processing (just to make it more exciting).

M. Wolf: Reusability can come from a combination of the host and sensor sides. Fancy signal processing, for example, can be performed at the host after basic data reduction is done at the sensor.

T. Schneider: Very interesting answers so far... the scope has broadened to include economic considerations as well as methodology and system aspects.

I agree with Jan that the design approach is key. I also agree with the points made by Gene regarding the economic considerations. These together imply that one of the most important aspects is the partitioning and design of the system. What portions of the device get "hard coded" permanently in hardware (for the lowest possible power)? Which portions of the device will retain some flexibility, through free programmability on a processor or via lower-power options like microcoding an accelerator in a ROM? Wise choices will lead to a system that will find broad application (that as Gene says "will serve thousands of markets"); whereas poor choices will result in a system that will not become a widely used platform.

An additional area that I see as important is software/configuration tools that provide an efficient and effective means of programming deeply embedded parallel systems in the absence of an operating system (in my opinion, an OS and ULP just don't go together). Ideally, these tools must work with a multiprocessor, heterogeneous system, and provide close to the same efficiency as one could achieve with hand coding. This is a challenge because both partitioning and scheduling across multiple compute units must be addressed. Without tools like this, building complex ULP systems will remain a challenge.

Moderator: It seems that hardware/software partitioning and codesign present important tradeoffs when building ULP systems. How do you expect ULP design techniques to evolve in the near future to address these trade-offs more effectively?

J. Henkel: I agree that HW/SW partitioning is a crucial decision to make when it comes to ULP. This decision is done at design time. I want to go a step further and claim that future ULP systems need to be able to flexibly adapt at run time to always (or most of the time) run at minimum possible power. There are scenarios that simply cannot be predicted at design time/compile time and if the system would not be able to adapt appropriately, it would give away some of the potentials in power savings. So, my statement is: future design techniques for ULP systems need (more than nowadays) to provide the ULP system with the capability of more run-time flexibility (even though classic design paradigms teach us that flexibility costs power consumption).

G. Frantz: Good point, a significant tradeoff will be flexibility versus power consumption. But let me replace flexibility with a different concept: time to market. I tell people that my customer wants four things from a vendor; a solution that has good enough performance, low enough cost, low enough power dissipation, and the ability to get them first to market.

In fact, forget the first three, "help me get to market first and give me a roadmap to cheap." Note that I said "roadmap to cheap" rather than "roadmap to ULP." At the end of the day, the system designer need not have ULP, but low enough power to lead the market. Flexibility is the key to leading a market.

T. Schneider: Perhaps I am being too optimistic, but I believe that with all of the activity we see in mainstream multiprocessor systems, it is only a matter of time before ULP design techniques evolve to support high-level assessment of different multiprocessor/computational unit/reconfigurable accelerator design concepts. These tools will allow assessment of the tradeoffs that are made between communications/data transfer overhead and the power consumption reductions realized through parallelization.

To extend Gene's point, ULP done properly is lower cost because an efficient design will typically have fewer gates, use less memory, and is therefore likely to consume less power and less silicon area. Flexibility is good, but optimizing the approach for very high volume ULP applications is the key for realizing the cost targets that are typically required.

Moderator: How mature are the ULP design tools that are needed to efficiently and effectively explore all tradeoffs we mentioned so far?

G. Frantz: I think this is a simple answer: They don't yet exist. But in case I am wrong, can we list the ones we need and the ones we already have?

J. Rabaey: Dismal is the right answer. There is progress in modeling and simulation (including statistical analysis). Logic synthesis tools can be adapted to serve for instance subthreshold logic without to much of a challenge. The rest is mostly spreadsheet. We really need good exploration tools.

T. Schneider: I agree on all points. I've seen some tools (and been involved in a research project) that studied exploration tools. They worked, but the models used for power consumption were too simplistic to make them useful in real applications. For now, experienced systems/chip designers, a good methodology for exploring design concepts, and spreadsheets are the best we've got.

G. Frantz: Let me give a real example of the state of our power evaluation tools in IC design. We just introduced (in the last month or so) a new device that was designed specifically for low power. Once we got silicon and tested it we found that the silicon's power dissipation was off by a factor of two from the design tool estimate. Fortunately it was off by two in the right direction. That means we have a lower power part than we expected. Although this sounds like great news, had we known that earlier we might have changed how we marketed it and given our system customers a greater head start on taking advantage of the lower power.

[ULP DONE PROPERLY IS LOWER COST BECAUSE AN EFFICIENT DESIGN WILL TYPICALLY HAVE FEWER GATES, USE LESS MEMORY, AND IS THEREFORE LIKELY TO CONSUME LESS POWER AND LESS SILICON AREA.]

Moderator: What are the implications of analog versus digital implementations of SP algorithms in ULP designs?

G. Frantz: I don't know that analog versus digital changes much at a high level. But it does once under the hood. ULP digital will get a great portion of its advancing by way of lowering the operating voltage, running slower, and using parallel concepts. Analog will not have as much of an advantage with lower voltage but will get its advantage by using analog concepts to do the intensive math operations. But the old thought that we would be able to virtually eliminate analog and do everything in digital is dying away and we now have the advantage of making tradeoffs between digital and analog implementation for SP.

As an aside, we have seen digital concepts used to enhance the analog to digital conversion accuracy. In the same way we will begin to see analog concepts used to enhance the performance of digital circuits. This will be relatively independent of whether the task is to increase performance or lower power dissipation. Of course, this leads to the question of what and how do we prepare students for this new world of ULP SP .

J. Rabaey: I essentially agree with Gene. It seems that for a number of SP functions (such as spectrum analysis), analog, and "mechanical" implementations can be more efficient than digital as long as the required accuracy is low (the number that I have often seen is around 6-b of accuracy as the transition point). When you need higher resolution/accuracy, you need to go digital. We have gradually migrated towards believing that digital is always the better way, but that is simply not true. Analog can often be very efficient, but don't ask accuracy.

Passive solutions don't take any energy at all!

Now here is where the opportunity lays, as Gene has perfectly pointed out: a combination of low-accuracy analog with higher resolution providing digital. The latter can still scale for a bit in terms of energy efficiency. The former benefits from the complex functionality that can be realized with few devices. And yes, this begs the next question.

T. Schneider: I agree with the points made above. In my experience, if you can "tolerate" an analog solution (pun intended) it will generally be lower power. Analog circuits also have the advantage of offering lower input-output delay and this can make them preferred (even required) in some application. Analog solutions may also bring in an additional degree of freedom. In some applications you can tradeoff size for reduced noise.

J. Henkel: I agree with most that has been said about analog. At the same time I want to point out that analog techniques for low power are by far not as advanced as digital ones. For example, transistor sizing in digital is highly optimized for low power. Techniques for optimizing sizing for analog transistors is often rather ad hoc and hence less power efficient. However, the whole picture in a DSP system might be more complex. Let's assume that an analog signal is processed without the need to convert it to digital. This saves the power the ADC/DAC consume and might compensate for the effect I pointed out above.

Moderator: Let's now look at ULP from an algorithm designer's point of view. What should SP algorithm designers focus on to enable ULP implementations of their algorithms?

J. Henkel: One important decision an algorithmic designer has to make is to determine the numerical representation of data in a DSP application i.e., to choose fixed point or floating point representation. This decision will heavily impact the power consumption since not only the fixed point and floating point calculations differ widely in power consumption but also the power related to storing/moving the respective data in/to/from memory can differ by many x. The ULP algorithmic designer should therefore consciously and carefully weigh the use of floating point versus fixed point.

T. Schneider: Algorithm designers should focus on clever optimizations that retain (or alternatively tradeoff as little as possible) algorithm performance, while minimizing the computation load (and hence the clock speed). In our experience, the biggest gains are realized by efficient SP algorithms. Doing this allows operation at reduced voltage that will deliver the reduced power. In many situations, partitioning an algorithm into elements that can be run in parallel across multiple computational units is an effective way to reduce power consumption, provided communication overhead does not consume the power reduction that can be gained via this approach. If the parallel computation units can be made reconfigurable or hard coded (in hardware), additional efficiencies can also be gained through more efficient utilization of hardware resources. I would also argue that in the vast majority of applications, floating-point and ULP don't mix. If you want the lowest possible power, you require a deep understanding of the algorithm and the minimum precision required to realize the required levels of performance.

PANELISTS

Gene Frantz (genf@ti.com) is the principal fellow at Texas Instruments and has been with Texas Instruments for more than 30 years. He holds 40 patents in memories, speech, consumer products, and DSP. He has written more than 50 papers and articles. He is an IEEE Fellow.

Jörg Henkel (henkel@kit.edu) is the chair for Embedded Systems at Karlsruhe Institute of Technology, Germany. He was previously with NEC Laboratories in Princeton, New Jersey. His current research is focused on design and architectures for embedded systems with focus on low power and reliability. He received the 2008 DATE Best Paper Award and the 2009 IEEE/ACM William J. McCalla ICCAD Best Paper Award. He is the chair of the IEEE Computer Society, Germany Section, and the editor-in-chief of *ACM Transactions on Embedded Computing Systems (ACM TECS)*. He is the coordinator and a founder of the German national research focal program "Design and Architecture for Dependable Embedded Systems." He holds ten U.S. patents.

Jan Rabaey (jan@eecs.berkeley.edu) is the Donald O. Pederson Distinguished Professor in the Electrical Engineering and Computer Science Department at the University of California, Berkeley. He is the scientific codirector of the Berkeley Wireless Research Center, as well as the director of the GigaScale Systems Research Center. He received numerous scientific awards, including the 1985 IEEE Transactions on Computer Aided Design Best Paper Award (Circuits and Systems Society), the 1989 Presidential Young Investigator Award, the 1994 Signal Processing Society Senior Award, and the 2002 ISSCC Jack Raper Award. He serves on the technical advisory board of a wide range of companies. He is an IEEE Fellow.

Todd Schneider (Todd.Schneider@onsemi.com) is the vice president of the Medical Diagnostics, Monitoring and Therapy product line at ON Semiconductor in Waterloo, Ontario, Canada. He holds BASc. and a MASc. degrees from the University of Waterloo, with a specialization in DSP. In 1998, he cofounded Dspfactory Ltd. He has published numerous refereed papers and conference proceedings. He also holds a number of U.S. and international patents in the field of DSP.

Marilyn Wolf (marilyn.wolf@ece.gatech.edu) is the Rhesa "Ray" S. Farmer, Jr. distinguished chair in embedded computing systems, and the Georgia Research Alliance Eminent Scholar at the School of Electrical and Computer Engineering at the Georgia Institute of Technology. She was with AT&T Bell Laboratories in Murray Hill, New Jersey from 1984 to 1989. She was with Princeton University from 1989 until 2007. She is a cofounder of Verificon Corporation. She helped to start several technical conferences, including CODES and MPSoC. She has written four textbooks.

MODERATOR

Umit Batur (batur@ti.com) received his B.S. degree in electrical engineering from Bilkent University, Turkey in 1998, and the M.S. and Ph.D. degrees in electrical engineering from Georgia Institute of Technology, Atlanta in 2000 and 2003, respectively. In 2003, he joined the Digital Signal Processing R&D Center of Texas Instruments in Dallas, where he is currently the manager of the imaging R&D branch. [SP]

[dates AHEAD]

Please send calendar submissions to:
Dates Ahead, c/o Jessica Barragué,
IEEE Signal Processing Magazine
445 Hoes Lane,
Piscataway, NJ 08854 USA,
e-mail: j.barrague@ieee.org
(Colored conference title indicates SP-sponsored conference.)

2010

[APRIL]

The 9th ACM/IEEE Conference on Information Processing in Sensor Networks (IPSN'10)
12–16 April, Stockholm, Sweden.
URL: http://ipsn.acm.org/2010

2010 IEEE International Symposium on Biomedical Imaging (ISBI 2010)
14–17 April, Rotterdam, The Netherlands.
General Chair: Wiro Niessen
URL: http://www.biomedicalimaging.org
E-mail: isbi2010@bigr.nl

[JUNE]

The 5th IEEE International Conference on Cognitive Radio Oriented Wireless Networks and Communications (CrownCom 2010)
9–11 June, Cannes, France.
Chairs: Erik G. Larsson and Aawatif Hayar
URL: http://www.crowncom2010.org/

Digital Object Identifier 10.1109/MSP.2009.935381

The 2nd International Workshop on Cognitive Information Processing (CIP 2010)
14–16 June, Elba Island (Tuscany), Italy.
General Cochairs: Fulvio Gini and Sergios Theodoridis
URL: http://www.conference.iet.unipi.it/cip2010/

The 11th IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAW 2010)
20–23 June, Marrakech, Morocco.
General Cochairs: Mounir Ghogho and Ananthram Swami
URL: http://www.spawc2010.org/

[JULY]

The IEEE International Conference on Multimedia & Expo (ICME 2010)
19–23 July, Singapore.
General Chairs: Yap-Peng Tan and Oscar C. Au
URL: http://www.icme2010.org

[AUGUST]

The 2010 International Workshop on Machine Learning for Signal Processing (MLSP 2010)
29 August–1 September, Kittilä, Finland.
General Chair: Erkki Oja
URL: http://mlsp2010.conwiz.dk/

[SEPTEMBER]

2010 International Conference on Image Processing (ICIP 2010)
26–29 September, Hong Kong.
General Chair: Wan-Chi Siu
URL: http://www.icip2010.org/icip2010.htm

[OCTOBER]

2010 IEEE Workshop on Signal Processing Systems (SiPS 2010)
6–10 October, San Francisco, California.
General Cochairs: Shuvra Battacharyya and Jorn Janneck
URL: http://www.sips2010.org/

2010 IEEE International Symposium on Phased Array Systems and Technology (ARRAY'10)
12–15 October, Waltham, Massachusetts.
Conference Chair: Mark Russell
URL: http://www.array2010.org/

[reader's CHOICE]

continued from page 19

TITLE, AUTHOR, PUBLICATION YEAR	ABSTRACT	RANK IN IEEE TOP 100 (MAY–OCT 2009)						*N* TIMES IN TOP 100 SINCE JAN 2006
IEEE SPS JOURNALS		OCT	SEP	AUG	JUL	JUN	MAY	
ACTIVE CONTOURS WITHOUT EDGES Chan, T.F.; Vese, L.A. *IEEE Transactions on Image Processing*, vol. 10, no. 2, Feb. 2001, pp. 266–277	This paper proposes a new model for active contours to detect objects in a given image, based on techniques of curve evolution, Mumford-Shah functional for segmentation and level sets.				100			7
REPRODUCIBLE RESEARCH IN SIGNAL PROCESSING Vandewalle, P. ; Kovacevic, J.; Vetterli, M. *IEEE Signal Processing Magazine*, vol. 26, no. 3, May 2009, pp. 37–47	This article suggests some practices for raising the quality of signal processing publications to an even higher level.						70	2
REMOVAL OF CORRELATED NOISE BY MODELING THE SIGNAL OF INTEREST IN THE WAVELET DOMAIN IMAGE Goossens, B.; Pizurica, A.; Philips, W. *IEEE Transactions on Image Processing*, vol. 18, no. 6, June 2009, pp. 1153–1165	This paper proposes a new denoising method for the removal of correlated noise, by modeling the significance of the noise-free wavelet coefficients in a local window using a new significance measure.						85	1
IEEE SPS CONFERENCES								
ENVIRONMENTAL ROBUSTNESS IN AUTOMATIC SPEECH RECOGNITION Acero, A.; Stern, R.M. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 2, Apr. 1990, pp. 849–852	This paper proposes two novel methods that are based on additive corrections in the cepstral domain in order to deal with differences in noise level and spectral tilt between close-talking and desk-top microphones.						47	1

continued from page 160

PHOTO BY MITRA OSQUI

[FIG2] The distinguished panel for the session in honor of Sanjoy Mitter (MIT). From left to right: Tom Kailath (Stanford), Dimitri Bertsekas (MIT), Y.C. (Larry) Ho (Harvard), Pravin Varaiya (University of California, Berkeley), Jan Willems (Catholic University of Louvain), Roger Brockett (Harvard), Petar Kokotovic (University of California, Santa Barbara), and Karl Åström (Lund Institute).

The "center of gravity" of Friday afternoon's sessions included systems, control, and optimization. The session was chaired by LIDS Co-Associate Director Munther Dahleh (MIT), with a panel consisting of principal speaker Keith Glover (Cambridge) and panelists Albert Benveniste (INRIA), Vincent Blondel (Catholic University of Louvain), Stephen Boyd (Stanford), Jonathan How (MIT), Richard Murray (California Institute of Technology), and Pablo Parrilo (MIT).

Keith Glover gave an overview of the field and its central elements, including feedback, dealing with uncertainty, approximation, and verification and certification of performance. He also discussed areas appropriate for academic research, which ranged from design methodologies for particular applications to development of verification tools. The other presentations in the session spanned topics that included a discussion of the central role that computational methods play in our field (and in particular, in redefining what we mean by a "solution"); a presentation of grand challenges (including robust and certifiably correct control of networked systems such as smart grids, the need for learning algorithms that lead to safe performance in a nonstationary world, and NASA's Green Flight Challenge); a discussion of the challenges in controlling complex systems, with examples including the DARPA Urban Challenge and the stunningly robust, computationally limited, and slow control system that allows a fly to maintain stable flight in the presence of sudden changes such as wind gusts; and a presentation on the challenges of "componentizing" control systems as is generally specified in the system engineering of complex and often safety-critical man-made systems, and the clear need for those in systems and control to contribute to the overall system-wide issues as well as to the components.

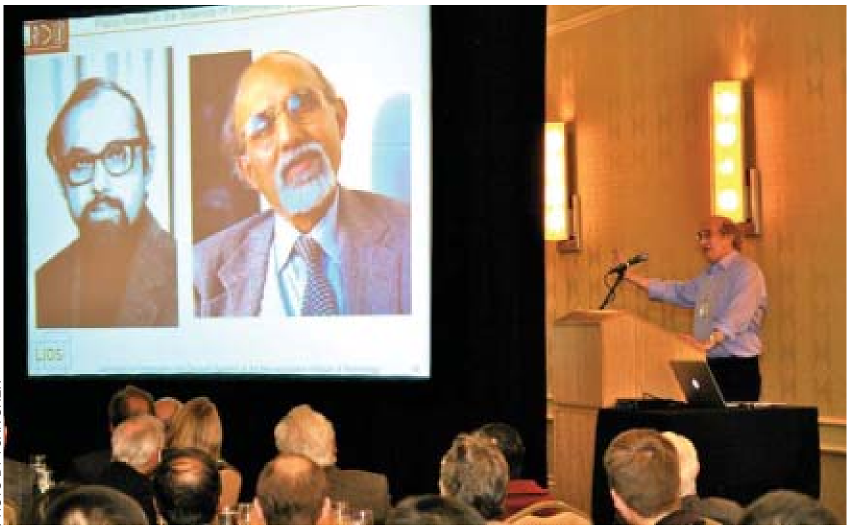


PHOTO BY YUAN SHEN

[FIG3] Alan Willsky (MIT) gives a talk "LIDS Through the Years" at the banquet. A pair of photos of Sanjoy Mitter (MIT) are in the background.

The symposium banquet was held Friday evening and included a talk by Alan Willsky on the long and celebrated history of LIDS, beginning with its days as the Servomechanism Laboratory extending back to the period prior to the Second World War until now (see Figure 3). The talk recognized the major figures whose contributions fueled the laboratory's major role in academia and society, as well as highlighting many of the contributions over the years, which included:

- the development of high-performance fire control systems
- the invention of magnetic core memory
- major advances in numerically controlled machines
- some of the earliest efforts in CAD and database systems
- a leadership role in the development of modern control and the development of robust control methods
- the broadening of its agenda to include networked systems ranging from communication and transportation networks to power grids
- the expansion of efforts in statistical signal processing and learning
- continuing advances in optimization algorithms
- a strong record of theoretical advances, influential texts, and an

impressive array of former students, colleagues, and visitors.

A plenary talk "System Theory: A Retrospective and Prospective Look" by Sanjoy Mitter was given Saturday morning. This far-reaching lecture provided both concrete and philosophical remarks about revolutionary science and argued that such a revolution took place in system theory in the 1960s, with its key elements being the emerging central role of computation, a new language leading to state space models, and the exploitation of this language to gain a far deeper understanding of systems as well as powerful new methods. Mitter argued that the challenges of today, in particular networked systems, might require some new elements and lead to new structural insights and methods. The talk also touched on pattern recognition and artificial intelligence and their close intellectual ties to information and decision systems as well as ties of Bayesian inference to statistical mechanics, a topic that resonates with the role of physics in understanding some of the core models and methods in machine learning. He discussed the challenges and opportunities that arise when one brings the constraints of communication systems into the design of control systems and closed with a list of challenges that could, by itself, fuel the field for a very long time.

The Saturday morning panel discussion had networks and information (broadly defined) as its center of gravity (see Figure 4). This session was chaired by LIDS Co-Associate Director John Tsitsiklis (MIT). John Doyle (Cal Tech) was the lead speaker in the session, together with a panel consisting of P.R. Kumar (Illinois), Asuman Ozdaglar (MIT), H. Vincent Poor (Princeton), Balaji Prabhakar (Stanford), Jeff Shamma (Georgia Tech), and David Tse (University of California, Berkeley). In his presentation, John Doyle gave a far-reaching discourse on networks, layered systems, their fragility and challenges in their design, as well as a contrasting view of some man-made networks (such as the Internet and power grids) and biological systems (e.g., bacteria), pointing out similarities, differences, and challenges for those of us in the information and decision sciences. The presentations of other panelists included

- an examination of application and domain challenges (including wireless security and multimedia communications) to "pull" the development of methodology and the "push" of specific technical challenges (e.g., in information theory and finite-block-length capacity)
- an examination, through example, of why it is worthwhile to continue examining very hard problems and looking for ways in which to reformulate them creatively in ways that overcome technical difficulties and lead to new results and insights
- an examination of the serious challenges in interplay of networks and information (including control of distributed systems over unreliable networks, methods for verifying performance, and distributing information processing as a problem blending computation, communication, and inference)
- the design of incentive systems for complex transportation networks to influence behavior and reduce congestion
- challenges and opportunities in network games and in understanding dynamics, learning, and decision-making in social and economic networks.

The afternoon panel discussion, focusing on signal processing, inference, and learning, was chaired by Alan Willsky (see Figure 5). The lead speaker in this session was Michael Jordan (University of California, Berkeley), who was joined by Alfred Hero (Michigan), Sanjeev Kulkarni (Princeton), Robert Nowak (Wisconsin), Pietro Perona (Cal Tech), Devavrat Shah (MIT), and Martin Wainwright (University of California, Berkeley). Jordan's presentation provided an overview of the broad area of machine learning and its ties to problems in a vast array of fields. He provided a view of current trends in machine learning including nonparametric Bayesian methods (with applications in signal and image processing highlighted), the challenges that the availability of massive data sets presents to those in learning and modeling; the investigation of "Objective Bayes" methods, which provide a unifying blend between Bayesian and frequentist views of statistics, with many ties to information theory; the great interest in methods that capture or recover "sparsity" in one form or another; and the challenge of bringing control and statistics together in the same synergistic way as optimization and statistics. Other presentations provided discussions of machine learning challenges in computer vision (e.g., so that one can search on parts of images or so that we can capture a human's ability to recognize new objects quickly); the challenging dynamic learning problems embedded in the operation of engineered networks (e.g., medium access control) and the role of so-called message passing algorithms; the challenges and opportunities in confronting increasingly high-dimensional data sets (with applications in learning graphical models) and the "blessings" as well as the well-known curses of dimensionality (with applications in sparse reconstruction and the uncovering of scaling laws) as well as

PHOTO BY MITRA OSQUI

[FIG4] The Saturday morning panel focused on networks and information. From left to right: Asuman Ozdaglar (MIT), P.R. Kumar (Illinois), H. Vincent Poor (Princeton), John Doyle (Cal Tech), Balaji Prabhakar (Stanford), David Tse (University of California, Berkeley), and Jeff Shamma (Georgia Tech).

[FIG5] The afternoon panel discussion focused on inference, signal processing, and learning. From left to right: Alan Willsky (MIT), Mike Jordan (University of California, Berkeley), Sanjeev Kulkarni (Princeton), Devavrat Shah (MIT), Martin Wainwright (University of California, Berkeley), Pietro Perona (Cal Tech), and Al Hero (Michigan). (One of the panelists, Rob Nowak (Wisconsin) was unable to attend.)

the posing of a question seen in other sessions as well, namely the tradeoff between computational effort and performance; the challenges in distributed or networked learning, and the fusion or aggregation of heterogeneous and nontraditional signal and data sources (ranging from sensor outputs to written text to forecasts of multiple agents); integrative modeling, prediction, and uncertainty assessment with predictive health and disease detection as a motivating application and challenge, characterized by heterogeneous data and diverse outputs (ranging from individual predictions to drug effectiveness assessment); and the use of feedback in sensing systems, i.e., the control or selection of measurements to be taken driven by the information state resulting from data already collected.

The meeting attracted a substantial number of researchers from around the world, leading to lively discussions that prompted our inviting participants to continue this conversation and to provide short perspective and position papers through the end of 2009. The Web site for this meeting http://paths.lids.mit.edu includes not only a statement of purpose, agenda, and list of sponsors, but also a complete collection of files generated by this meeting. This includes a) video of the entire meeting; b) all panelist slides; c) short perspectives and position papers submitted by attendees; and d) a summary document produced by Munther Dahleh, John Tsitsiklis, and Alan Willsky.

AUTHOR

Alan S. Willsky (willsky@mit.edu) is the Edwin Sibley Webster Professor of electrical engineering and computer science and director of the Laboratory for Information and Decision Systems at MIT. He was a founder of Alphatech, Inc. and chief scientific consultant, a role in which he continues at BAE Systems Advanced Information Technologies. From 1998 to 2002, he served on the U.S. Air Force Scientific Advisory Board. He has received several awards including the 1975 American Automatic Control Council Donald P. Eckman Award, the 1979 ASCE Alfred Noble Prize, the 1980 IEEE Browder J. Thompson Memorial Award, the 1988 IEEE Control Systems Society Distinguished Member Award, the 2004 IEEE Donald G. Fink Prize Paper Award, and the 2005 Doctorat Honoris Causa from Université de Rennes. He is coauthor of *Signals and Systems*. [SP]

advertisers INDEX

The Advertisers Index contained in this issue is compiled as a service to our readers and advertisers: the publisher is not liable for errors or omissions although every effort is made to ensure its accuracy. Be sure to let our advertisers know you found them through *IEEE Signal Processing Magazine*.

COMPANY	PAGE#	URL	PHONE
ESC	5	www.embedded.com/esc/sv	
IASTED	158	www.iasted.org/conferences/home-710.html	+1 403 288 1195
John Wiley & Sons	7	www.wiley.com	+1 877 762 2974
Mathworks	CVR 4	www.mathworks.com/connect	+1 508 647 7040
Mini-Circuits	CVR 2, 3, CVR 3	www.minicircuits.com	+1 718 934 4500

advertising SALES OFFICES

James A. Vick
Staff Director, Advertising
Phone: +1 212 419 7767;
Fax: +1 212 419 7589
jv.ieeemedia@ieee.org

Marion Delaney
Advertising Sales Director
Phone: +1 415 863 4717;
Fax: +1 415 863 4717
md.ieeemedia@ieee.org

Susan E. Schneiderman
Business Development Manager
Phone: +1 732 562 3946;
Fax: +1 732 981 1855
ss.ieeemedia@ieee.org

Product Advertising

MIDATLANTIC
Lisa Rinaldo
Phone: +1 732 772 0160;
Fax: +1 732 772 0164
lr.ieeemedia@ieee.org
NY, NJ, PA, DE, MD, DC, KY, WV

NEW ENGLAND/ EASTERN CANADA
Jody Estabrook
Phone: +1 774 283 4528;
Fax: +1 774 283 4527
je.ieeemedia@ieee.org
ME, VT, NH, MA, RI, CT
Canada: Quebec, Nova Scotia, Newfoundland, Prince Edward Island, New Brunswick

SOUTHEAST
Thomas Flynn
Phone: +1 770 645 2944;
Fax: +1 770 993 4423
tf.ieeemedia@ieee.org
VA, NC, SC, GA, FL, AL, MS, TN

MIDWEST/CENTRAL CANADA
Dave Jones
Phone: +1 708 442 5633;
Fax: +1 708 442 7620
dj.ieeemedia@ieee.org
IL, IA, KS, MN, MO, NE, ND, SD, WI, OH
Canada: Manitoba, Saskatchewan, Alberta

MIDWEST/ ONTARIO, CANADA
Will Hamilton
Phone: +1 269 381 2156;
Fax: +1 269 381 2556
wh.ieeemedia@ieee.org
IN, MI. Canada: Ontario

SOUTHWEST
Shaun Mehr
Phone: +1 949 923 1660;
Fax: +1 775 908 2104
sm.ieeemedia@ieee.org
AR, LA, OK, TX

WEST COAST/ NORTHWEST/ WESTERN CANADA
Marshall Rubin
Phone: +1 818 888 2407;
Fax: +1 818 888 4907
mr.ieeemedia@ieee.org
AZ, CO, HI, NM, NV, UT, AK, ID, MT, WY, OR, WA, CA. Canada: British Columbia

EUROPE/AFRICA/MIDDLE EAST
Heleen Vodegel
Phone: +44 1875 825 700;
Fax: +44 1875 825 701
hv.ieeemedia@ieee.org
Europe, Africa, Middle East

ASIA/FAR EAST/PACIFIC RIM
Susan Schneiderman
Phone: +1 732 562 3946;
Fax: +1 732 981 1855
ss.ieeemedia@ieee.org
Asia, Far East, Pacific Rim, Australia, New Zealand

Recruitment Advertising

MIDATLANTIC
Lisa Rinaldo
Phone: +1 732 772 0160;
Fax: +1 732 772 0164
lr.ieeemedia@ieee.org
NY, NJ, CT, PA, DE, MD, DC, KY, WV

NEW ENGLAND/EASTERN CANADA
John Restchack
Phone: +1 212 419 7578;
Fax: +1 212 419 7589
j.restchack@ieee.org
ME, VT, NH, MA, RI. Canada: Quebec, Nova Scotia, Prince Edward Island, Newfoundland, New Brunswick

SOUTHEAST
Cathy Flynn
Phone: +1 770 645 2944;
Fax: +1 770 993 4423
cf.ieeemedia@ieee.org
VA, NC, SC, GA, FL, AL, MS, TN

MIDWEST/TEXAS/CENTRAL CANADA
Darcy Giovingo
Phone: +1 847 498 4520;
Fax: +1 847 498 5911
dg.ieeemedia@ieee.org;
AR, IL, IN, IA, KS, LA, MI, MN, MO, NE, ND, SD, OH, OK, TX, WI. Canada: Ontario, Manitoba, Saskatchewan, Alberta

WEST COAST/SOUTHWEST/ MOUNTAIN STATES/ASIA
Tim Matteson
Phone: +1 310 836 4064;
Fax: +1 310 836 4067
tm.ieeemedia@ieee.org
AZ, CO, HI, NV, NM, UT, CA, AK, ID, MT, WY, OR, WA. Canada: British Columbia

EUROPE/AFRICA/MIDDLE EAST
Heleen Vodegel
Phone: +44 1875 825 700;
Fax: +44 1875 825 701
hv.ieeemedia@ieee.org
Europe, Africa, Middle East

Digital Object Identifier 10.1109/MSP.2009.935380

Alan S. Willsky

Paths Ahead in the Science of Information and Decision Systems

On 12–14 November 2009, a significant meeting, the symposium on "Paths Ahead in the Science of Information and Decision Systems" was held at the Massachusetts Institute of Technology (MIT). This meeting was organized and run by MIT's Laboratory for Information and Decision Systems (LIDS), the oldest continuing laboratory at MIT. LIDS has played (and continues to play) a major role in the development of our field, responding to critical national and societal needs; developing fundamental and path-breaking advances in theory, methodology, and practice; and serving as a focal point for activities involving the best across MIT, the nation, and the world.

The science of information and decision systems encompasses a substantial and exceptionally pervasive set of interrelated disciplines, ranging from signal and image processing; to embedded control systems; to the analysis, design, and optimization of complex distributed systems and networks. Thanks both to the richness of the challenges throughout engineering and the physical, biological, and social sciences, as well as the continuing developments of the foundations of our disciplines, the information and decision sciences stand today as an exciting, continually evolving, and critical domain of intellectual inquiry.

Consistent with that history and mission, LIDS organized the Paths Ahead Symposium, bringing together leading researchers from all around the world who have been influential in shaping the vision of and leading this broad field. Sponsored by MIT's School of Engineering as well as by a number of private companies, laboratories, and by the National Science Foundation, the Air Force Office of Scientific Research, and the Army Research Office, the meeting consisted of several panel-oriented sessions, providing both context and history as well as a look across disciplines for challenges and opportunities for the future. While each of these sessions had a specific theme, an overall objective of each session was to look across disciplines for challenges and opportunities across disciplines.

Digital Object Identifier 10.1109/MSP.2009.935454

[FIG1] The audience listens at the opening session on 13 November 2009.

The meeting, which attracted 340 registrants, began with a reception on 12 November at the MIT Museum. Technical sessions began on 13 November with welcoming remarks given by Alan Willsky, the symposium general chair and LIDS director (see Figure 1).

The morning session on Friday was organized in honor of Sanjoy Mitter, a major leader in the field and former director of LIDS, who recently retired from MIT (although one would not know that from his continued presence). This session was chaired by Thomas Magnanti (MIT), former dean of engineering at MIT and long-time collaborator with Mitter. The panelists were Karl Johan Åström (Lund Inst.), Dimitri Bertsekas (MIT), Roger Brockett (Harvard), Y.C. (Larry) Ho (Harvard), Thomas Kailath (Stanford), Petar Kokotovic (University of California, Santa Barbara), Pravin Varaiya (University of California, Berkeley), and Jan Willems (Catholic University of Louvain) (see Figure 2). The presentations and discussion in this section included personal perspectives on the past history of research in this broad field, some challenges and exciting opportunities that are before us, and the challenge of educating our students in a field with the breadth that the information and decision sciences possess.

Some of the challenges that were brought up by the panel were those central to the social agenda and the exciting areas of research of today, including energy, transportation, biology, and health care. Intellectual challenges, including the development of new methods to deal with compositional descriptions of complex systems and the broad area of networks, information, and control were also discussed, as were continuing areas such as robotics, embedded systems, and autonomy. The intellectual vibrancy of our field and its ability to move with agility into new domains were evident throughout.

(continued on page 156)

1053-5888/10/$26.00©2010IEEE

WORLD'S WIDEST SELECTION

VCOs

RoHS models available, consult factory.

10 to 6840 MHz from $11.95 ea. (qty. 5)

Want a miniature surface mount, shielded plug-in, or rugged coaxial voltage controlled oscillator with the right stuff for your project? Contact Mini-Circuits! From custom designs to standard catalog models ***always in stock***, we'll supply extra robust, 100% tested VCO solutions you need at a price you can afford. Choose from narrow to broad to octave band widths. Select linear tuning, low phase noise, and 5V models optimized for PLLs and synthesizers. And pick from an innovative array of miniature SM packages as small as 0.370" square for a variety of designs and applications. You can quickly find the model you need using "The YONI2 Search Engine" at the Mini-Circuits web site. Just enter your specifications into YONI2...click...and immediately start evaluating suggested VCO solutions using the *actual measured performance data* displayed. But perhaps you need a custom design. Not a problem! Contact us for our lightning fast response, low prices, and quick turnaround. Give the competition *real competition*... specify Mini-Circuits VCOs!

Mini-Circuits...Your partners for success since 1969

Agilent Technologies

For high reliability, all Mini-Circuits VCOs are tested with the Agilent E5052B Signal Source Analyzer.
www.agilent.com/find/ssa

Mini-Circuits®
ISO 9001 ISO 14001 AS 9100 CERTIFIED

P.O. Box 350166, Brooklyn, New York 11235-0003 (718) 934-4500 Fax (718) 332-4661

YONI2™ Patent Pending ***The Design Engineers Search Engine*** *finds the model you need, Instantly* • For detailed performance specs & shopping online see minicircuits.com

IF/RF MICROWAVE COMPONENTS

359 rev N

Agilent

Tektronix

LeCroy

Rohde & Schwarz

National Instruments

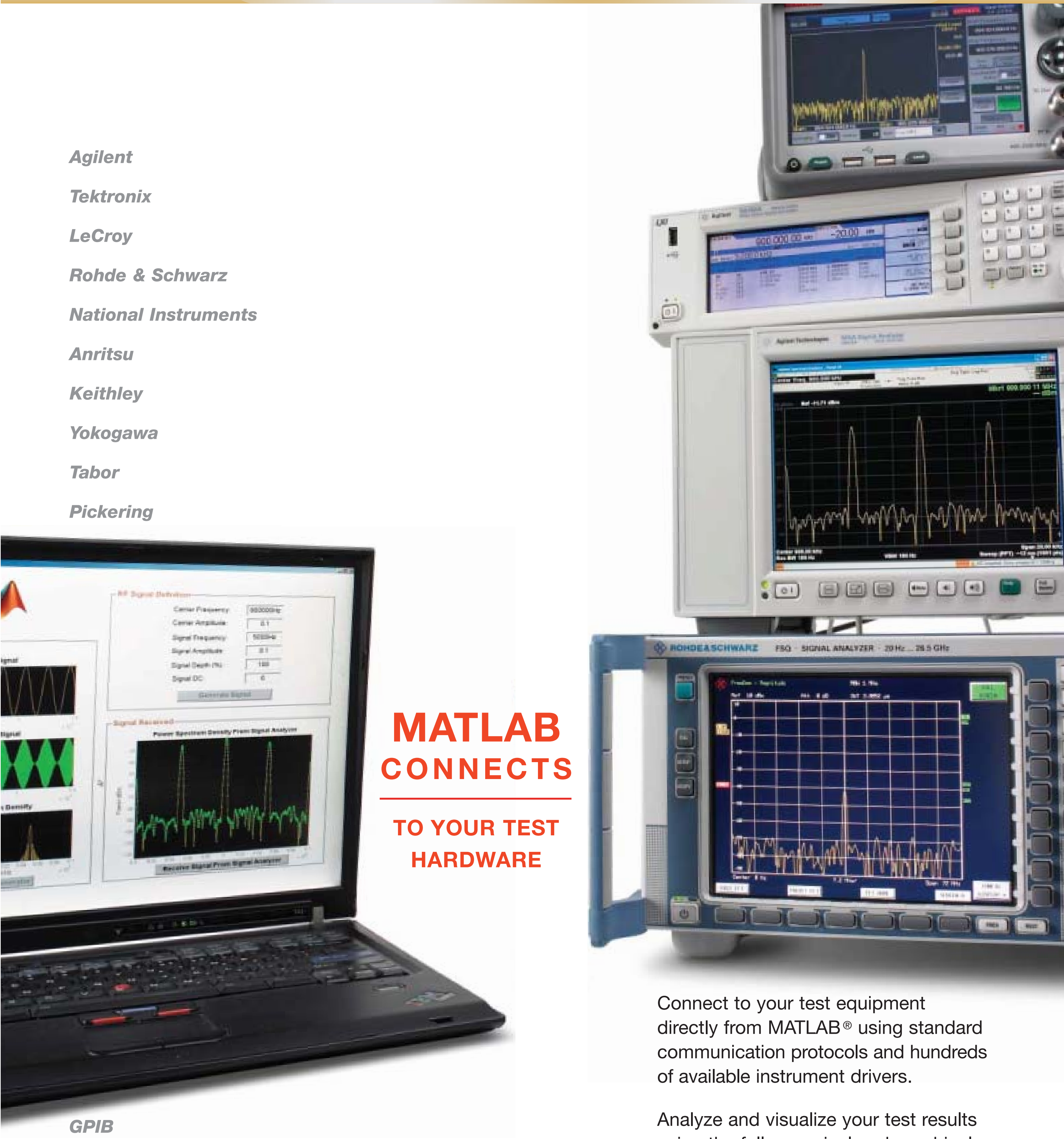
Anritsu

Keithley

Yokogawa

Tabor

Pickering



MATLAB CONNECTS

TO YOUR TEST HARDWARE

GPIB

LXI

IVI

TCP/IP

VISA

USB

UDP

RS-232

Connect to your test equipment directly from MATLAB® using standard communication protocols and hundreds of available instrument drivers.

Analyze and visualize your test results using the full numerical and graphical power of MATLAB.

For more information on supported hardware, visit www.mathworks.com/connect

The MathWorks™