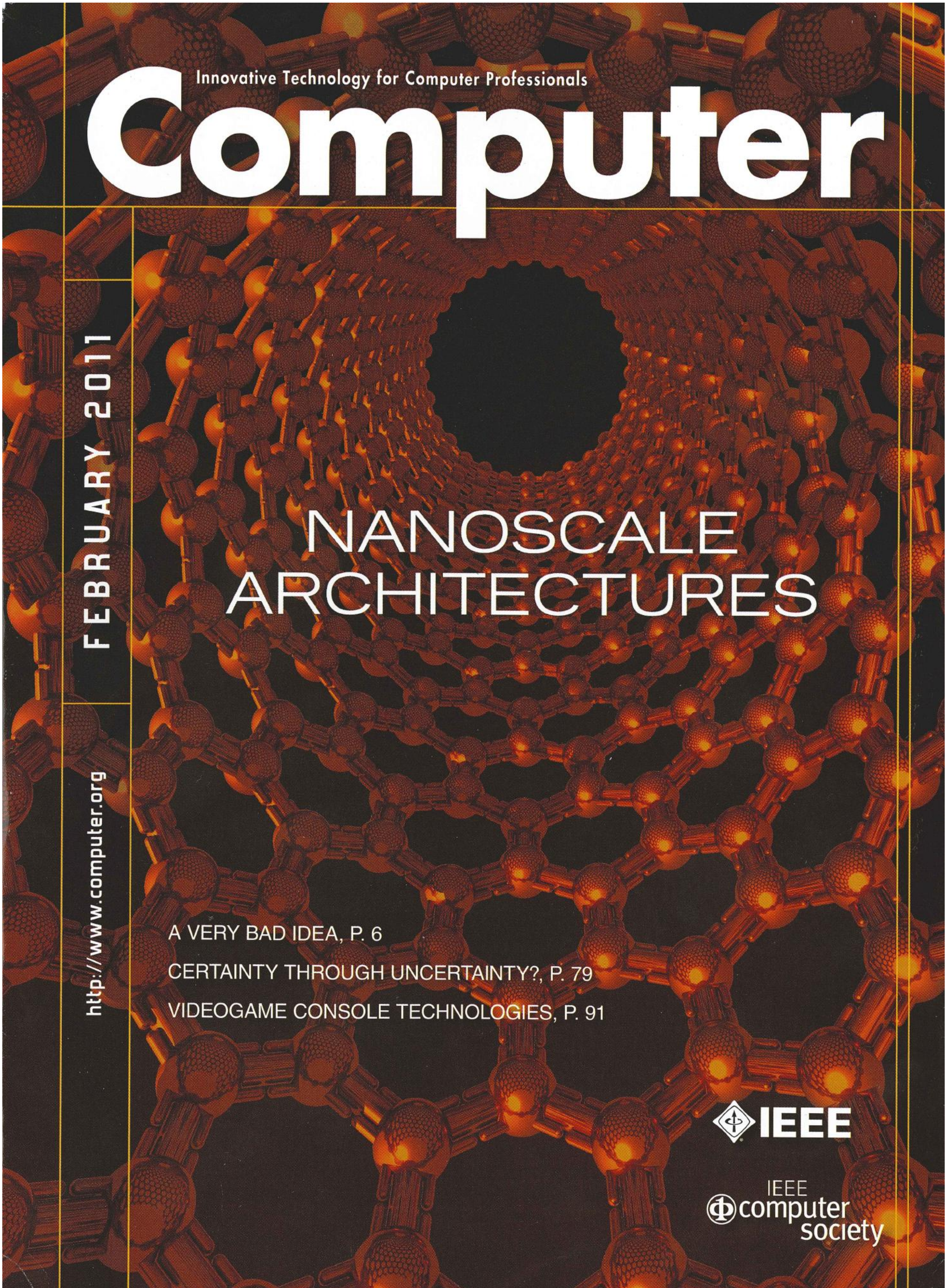


Innovative Technology for Computer Professionals

Computer

FEBRUARY 2011

http://www.computer.org

NANOSCALE ARCHITECTURES

IEEE

IEEE computer society

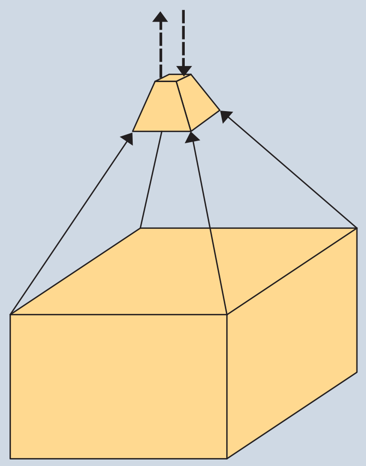


Figure 3. How learning occurs. Linear and nonlinear transformations cooperate to implement the learning that corresponds to long-term memory. Cog combines a linear transformation's current state, *W*, with the input, *x*, to produce *y*, a partial inference. It then combines *g*, *x*, and *W* to implement learning. The learning function, *f*, determines the type of learning that occurs.

transformations onto the regular computational engines (GPUs, Dendras), maps nonlinear transformations onto the CPU cores, and implements tensor field communication through local memory or through messages passed across the network.

None of these platform attributes are visible to users, which frees them to focus on their cognitive models. Adding computational resources either speeds up model execution or, in a real-time environment, increases the size of the model that Cog can execute.

These abstractions might seem distant from their biological counterparts, but a rough correspondence exists. Tensor fields moving along the graph edges are similar to the information that axon bundles convey in a human nervous system. Linear transformations are analogous to the computation performed in the dendritic trees of neuron populations, with the learning or adaptation analogous to the modifications of synaptic weights. This is the storage of long-term memory. Nonlinear transformations correspond to the nonlinear dynamics of populations of neuron bodies or somas—the storage of medium- and short-term memories.

Learning

Linear transformation adaptation generally occurs over a much longer time scale relative to nonlinear transformations, and this slower adaptation is what constitutes learning. As Figure 3 shows, feedback from a nonlinear transformation guides learning. Cog holds the actual learned state, *W*, within a linear transformation. It then convolves or correlates *W* with an input, *x* (part of a tensor field), to produce an output tensor field, *y*. We call this a *partial inference*. Cog uses the partial inference to drive a nonlinear transformation, which can respond by feeding back a learning field, *g*, to the linear transformation. The linear transformation uses *g*, *x*, and *W* (its current state) to update its learned state.

Through configuration and appropriate feedback,[6] we can implement a wide variety of classical learning laws, which fall into four categories:

- *Hebb rule derivatives*, including classic Hebbian, Hebb plus passive decay, presynaptically gated decay (outstar), postsynaptically gated decay (instar), Oja, dual OR, and dual AND;
- *threshold-based rules*, including Covariance 1, Covariance 2, BCM (Dayan and Abbott), original BCM (oBCM), IBCM, and Bienenstock, Cooper, and Munro (BCM) theory (Law and Cooper);
- *feedback-based rules*, including back-propagation, Harpur's rule, and contrastive divergence; and
- *temporal-trace-based rules*, including Rescorla Wagner, temporal difference, and Foldiak.

Development environment

Brains interact with environments, receiving information from sensors and conveying motor commands so that the bodies they govern can move about and affect the world around them. To make development easier, Cog offers a set of virtual environments with which brain models can interact. The environments, which vary in complexity, run synchronously with the brain model.

Because Cog is a synchronous, digital architecture, users can halt and restart model execution without perturbing the computation, so that they can peek inside their model and debug it if necessary. If enough computational resources are available, Cog can run faster than real time in a virtual environment.

Figure 4 is a screenshot of the graphical user interface (GUI) in the Cog debugger. The left side of the screen shows a graph of the network being debugged; in this case, a feed-forward network is implementing a simple form of boundary completion. The user can click on various points within that model to view its internal state, which the GUI displays on the right side.

BUILDING APPLICATIONS

Cog can implement a wide range of neuromorphic algorithms. To give an idea of Cog's capabilities, we describe four small applications that we built using our framework: contrast normalization, independent component analysis, learning of orientation maps with ocular dominance, and boundary completion.

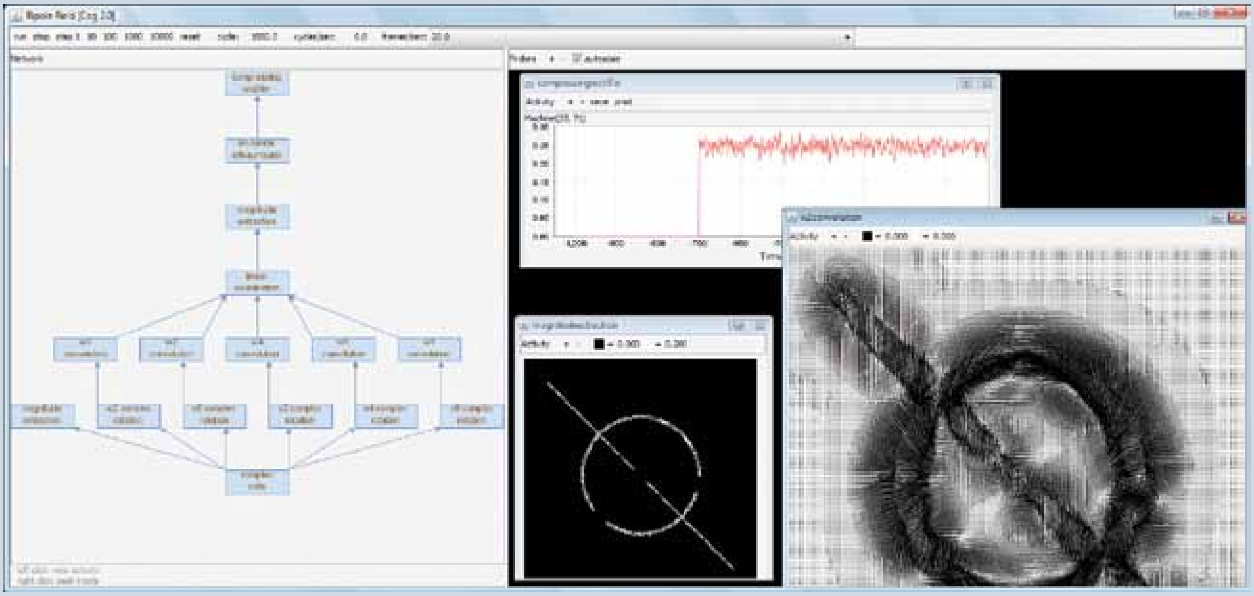


Figure 4. **Debugging with Cog. (left) The GUI displays the network structure, which the user is probing to display the internal state of some of the model's adaptive transformations. (right) Each window names the adaptive transformation being displayed. The bottom two windows show snapshots of the tensor fields generated and transmitted by two of the network transformations, while the top window displays a history of a single state variable within a third transformation.**

Contrast normalization

Figure 5 illustrates contrast normalization using the Retinex algorithm[7] as implemented with Cog. The image in the figure is of a parking garage, which contains a dynamic range that a camera cannot capture in all its detail, but that a human eye can. In Figure 5a, standard photographic compression loses detail in the brightest and darkest regions. However, one linear and one nonlinear transformation on Cog can easily implement the Retinex algorithm, which approximates retinal processing, to capture detail in both shadow and glare. Figure 5b shows the dramatic difference. Filtering is clearly nonlocal, since the brightest areas in the original image do not always correspond to the brightest areas in the processed image. This early preprocessing is essential for handling the real-world video streams that the brain model receives as input.

(a)

(b)

Figure 5. **Contrast normalization of a high-contrast scene in a parking garage. (a) Standard photographic compression (jpeg) of the scene loses details in deep shadow and bright regions. (b) The Retinex algorithm as implemented with Cog recaptures the details while maintaining local contrast. (Image data courtesy of R. Brinkworth and D. O'Carroll, "Robust Models for Optic Flow Coding in Natural Scenes Inspired by Insect Biology,"** ***PLoS Computational Biology*****, vol. 5, no. 11; www.ploscompbiol.org/article/info%3Adoi%2F10.1371%2Fjournal.pcbi.1000555;jsessionid=4BEF883BF3CAA0B158302B10E2E74AA1.ambra02).**

Independent component analysis

Scientists believe that most of the information in natural images is contained in the scene's edges[8] and that edge filters can capture and compress that information to simplify later processing. Figure 6 shows an example of a model built with Cog that uses BCM theory to learn the independent components of natural scenes.

In the application, a simple network implemented on Cog uses three transformations in series to approximate early processing in the visual system. The first transformation enhances edges using a difference-of-Gaussians

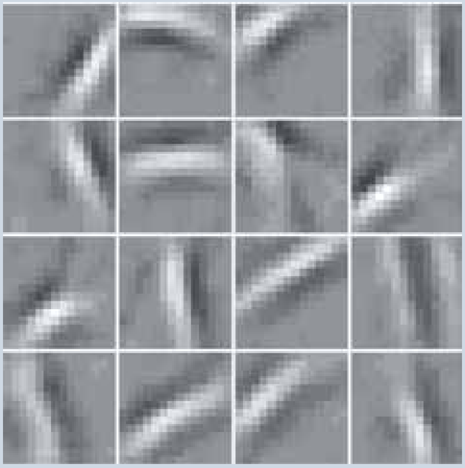
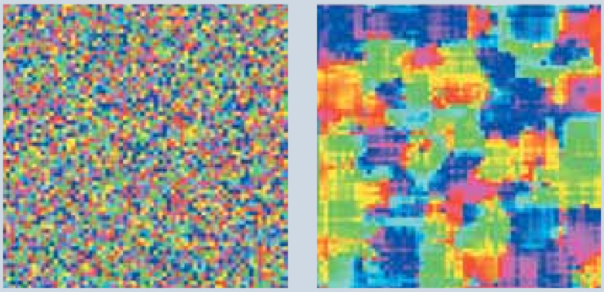


Figure 6. Learning independent components of natural scenes. A network built with Cog for learning independent components preprocessed random patches from natural scenes using a difference-of-Gaussians filter and then processed that output using the BCM neuron model combined with Hebbian learning. The result was the learning of the 16 Gabor-like edge filters shown.



(a) (b)

Figure 7. Topographic map of orientation selectivity. (a) Before learning, there are no identifiable clusters of orientation selectivity. (b) Clusters emerge after learning, with different colors denoting particular orientations.

filter. The second, a linear transformation, applies convolution with adaptive kernels that implement simple Hebbian learning. Finally, the third transformation implements a competitive field of BCM theory neurons.[9] Random patches from natural images (photographs of trees, grass, fields) drive the first layer, and the last two layers respond by adapting to their inputs. The resulting network self-organizes into a set of edge filters, shown in the figure, that react strongly to edges (the "independent components" of natural scenes) in visual inputs (approximating simple cells in the V1 cortical region of the vision system).

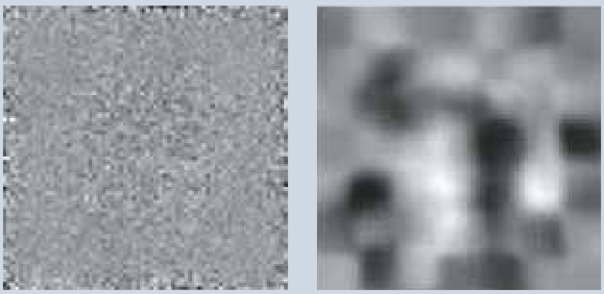
Orientation maps and ocular dominance

Experiments show that simple cells (neurons) in the V1 area of the visual cortex behave as edge filters and that the orientation of those edge filters varies smoothly over the surface of V1. Experiments also show that V1 cells receive visual input information from both eyes, but that signals from only one eye dominate any given cell. This ocular dominance organization occurs in clumps or bands, depending on the species.

Figures 7 and 8 illustrate a Cog model of the self-organization of V1 for both orientation and ocular dominance. In this example, we input a series of random images, heavily filtered to form blob-like images.[10] We then topographically connected two views of the image, simulating two eyes, to the simulated V1. Figure 7 shows the resulting self-organization of the orientation filters, and Figure 8 shows the resulting ocular dominance clumping.

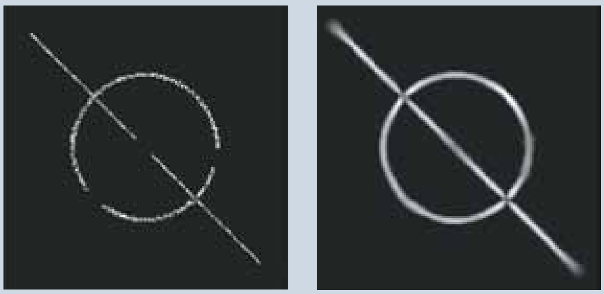
Boundary completion

A person can easily recognize visual objects even when noise or foreground objects obstruct or partially occlude them. People somehow infer the missing information and fill in the scene to form completed percepts—presumably to facilitate recognition in later cortical stages. Boundary completion, a simple form of this filling-in process, com-



(a) (b)

Figure 8. Topographic map of ocular dominance. (a) Before learning, there are no identifiable columns of ocular dominance. (b) Columns emerge after learning, with white denoting left eye, black denoting right eye, and gray denoting intermediate.



(a) (b)

Figure 9. Simple boundary completion. (a) A noisy figure with a broken line and broken circle becomes (b) smooth and filled in using a boundary completion algorithm implemented with Cog.

pletes broken and curved edges and lines using the Gestalt principles of proximity and good continuation.

Although a self-organizing boundary-completion model exists, it is computationally expensive to learn.[11] We found it much cheaper simply to prewire the mechanism into a Cog-built model. The model can then exploit a vast number of mathematical tricks, such as steerable filters, fast Fourier transform, tensor convolution, and data compression.[12] Relative to the traditional boundary completion model,[13] a Cog-built model with this mechanism can reduce implementation energy by at least four orders of magnitude.

Figure 9 shows the input and output of a Cog-built model that implements a simple form of boundary completion. In this example, the user has implemented the feed-forward network in Figure 4 to complete the noisy input image of a broken circle and broken line.

Although a general theory of cognition does not yet exist, researchers do recognize that platform flexibility is essential as they plow through the fog and uncertainty of learning to build intelligent machines. Cog has many features that offer this flexibility. Its all-digital hardware foundation reduces technological and fabrication risk. Its placement of memristive synaptic memory banks close to their associated processing elements reduces CV2f power losses by several orders of magnitude—a reduction critical in processing neuromorphic algorithms, which deeply entangle memory and computation.

Cog's tensor framework mechanisms are perhaps nonbiological, but they are well-matched to our underlying CMOS/memristive technology. The framework is also expressive, pulling in linear algebra, geometry, and analysis into a single foundation, and enables exploitation of much mathematical and engineering knowledge—for example, information and coding theory, digital signal processing, non-Euclidean coordinate systems, tensor convolution, normalized convolution, and fast Fourier transforms, among many others. The framework also supports a wide variety of learning laws and network models.

Perhaps the most important architectural attribute is the nearly complete decoupling of the software abstractions for building brains (tensor fields and adaptive transformations) from the underlying hardware platform. Not only does this provide portability among existing and future platforms, it allows us to quickly modify the software architecture to accommodate new or unexpected algorithmic problems as they arise. C

Acknowledgments

This work was partially funded by the DARPA SyNAPSE program, contract HR0011-09-3-0001. The views, opinions, and/or findings contained in this article are those of the authors and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense.

References

1. L.O. Chua and S.M. Kang, "Memristive Devices and Systems," *Proc. IEEE*, vol. 64, no. 2, 1976, pp. 209-223.
2. D.B. Strukov et al., "The Missing Memristor Found," *Nature*, 1 May 2008, pp. 80-83.
3. Q. Xia et al., "Memristor/CMOS Hybrid Integrated Circuits for Reconfigurable Logic," *Nano Letters*, vol. 9, no. 10, 2009, pp. 3640-3645.
4. B. Madappuram et al., "On Brain-Inspired Connectivity and Hybrid Network Topologies," *IEEE Symp. Nanoscale Architectures* (NANOARCH 08), IEEE Press, June 2008, pp. 54-61.
5. D.S. Basset et al., "Efficient Physical Embedding of Topologically Complex Information Processing Networks in Brains and Computer Networks," *PLoS Computational Biology*, Apr. 2010, e1000748.
6. A. Gorchetchnikov et al., "General Form of Learning Algorithms for Neuromorphic Hardware Implementation," *BMC Neuroscience*, vol. 11 (supp. 1), 2010, p. 91; www.biomedcentral.com/1471-2202/11/S1/P91.
7. D.J. Jobson, Z. Rahman, and G.A. Woodell, "Properties and Performance of a Center/Surround Retinex," *IEEE Trans. Image Processing*, Mar. 1997, pp. 451-462.
8. A. Bell and T. Sejnowski, "The 'Independent Components' of Natural Scenes Are Edge Filters," *Vision Research*, Dec. 1997; pp. 3327-3338.
9. N. Intrator and L. Cooper, "Objective Function Formulation of the BCM Theory of Visual Cortical Plasticity: Statistical Connections, Stability Conditions," *Neural Networks*, vol. 5, no. 1, 1992, pp. 3-17.
10. R. Mikkulainen et al., *Computational Maps in the Visual Cortex*, Springer, 2005.
11. S. Grossberg and J. Williamson, "A Neural Model of How Interlaminar Connections of Visual Cortex Develop into Adult Circuits that Carry Out Perceptual Grouping and Learning," *Cerebral Cortex*, vol. 11, no. 1, 2001, pp. 37-58.
12. E. Franken et al., "An Efficient Method for Tensor Voting Using Steerable Filters," LNCS 3954, Springer, 2006, pp. 228-240.
13. S. Grossberg and E. Mingolla, "Neural Dynamics of Perceptual Grouping—Textures, Boundaries, and Emergent Segmentations," *Perception and Psychophysics*, vol. 38, no. 2, pp. 141-171.

***Greg Snider** is a senior researcher at Hewlett-Packard Laboratories and the principal HP investigator in the DARPA SyNAPSE project. His research interests include hardware and software architectures. Snider received an MS in scientific instrumentation from the University of California, Santa Barbara, and an MSEE from Stanford University. Contact him at snider.greg@hp.com.*

***Rick Amerson** is a managing consultant for Hewlett-Packard Laboratories. His research interests include computer architecture, instruction sets, memory, computation, and algorithms. Amerson received an SM in management of* *technology from the Massachusetts Institute of Technology. Contact him at frederic.amerson@hp.com.*

***Dick Carter** is a consulting research scientist at Hewlett-Packard Laboratories. His research interests include parallel algorithms and GPU computing. Carter received an MSEE from Stanford University. He is a member of IEEE. Contact him at richard.carter@hp.com.*

***Hisham Abdalla** is a research associate at Hewlett-Packard Laboratories. His research interests include neuromorphic engineering, robotics, computer architecture, and floating-point gate array design. Abdalla received a PhD in electrical and computer engineering from the University of Maryland at College Park. Contact him at hisham.abdalla@hp.com.*

***Muhammad Shakeel Qureshi** is a researcher and circuit designer at Hewlett-Packard Laboratories. His research interests include low-power circuits, MEMS sensor interfaces, optoelectronics, and mixed-signal and radio-frequency circuits for communication blocks. Qureshi received a PhD in electrical and computer engineering from the Georgia Institute of Technology. Contact him at shakeel.qureshi@hp.com.*

***Jasmin Léveillé** is a postdoctoral associate in the Department of Cognitive and Neural Systems at Boston University. His research interests include human and computer vision and neural architectures for high-performance computing. Léveillé received a PhD in cognitive and neural systems from Boston University. Contact him at jasminl@cns.bu.edu.*

***Massimiliano Versace** is a senior research scientist in the Department of Cognitive and Neural Systems at Boston University and director of the university's Neuromorphics Laboratory. He is also codirector of technology outreach at NSF's Science of Learning Center's Center of Excellence for Learning in Education, Science, and Technology (CELEST). He is a principal investigator for the Boston University subcontract with Hewlett-Packard in the DARPA SyNAPSE project. Versace received a PhD in cognitive and neural systems from Boston University. Contact him at maxversace@gmail.com.*

***Heather Ames** is a research scientist in the Department of Cognitive and Neural Systems at Boston University, codirector of technology outreach at NSF's CELEST, and a member of its governing board to facilitate technology transfer from models to private industries and laboratories. Her research interests include speech modeling, analysis of learning and homeostasis in neural network systems, and the technology transfer of brain-based applications. Ames received a PhD in cognitive and neural systems from Boston University. Contact her at heather.m.ames@gmail.com.*

***Sean Patrick** is a doctoral student in the Department of Cognitive and Neural Systems at Boston University. His research interests include robotics and prosthetics; adaptive, neural, and evolutionary algorithms; and computational neuroscience. Patrick received a BS in electrical and computer engineering from the Worcester Polytechnic Institute. Contact him at sean.patrick.619@gmail.com.*

***Benjamin Chandler** is a doctoral student in the Department of Cognitive and Neural Systems at Boston University and ACES associate at the university's Center for Computational Science. His research interests include large-scale simulation, homeostatic plasticity, and neuromorphic computing. Chandler received a BS in cognitive science from Carnegie Mellon University. He is a student member of IEEE. Contact him at bchandle@gmail.com.*

***Anatoli Gorchetchnikov** is a research assistant professor in the Department of Cognitive and Neural Systems at Boston University and leader of the Modular Neural Exploring and Traveling Agent project within DARPA's SyNAPSE program. His research interests include the biologically detailed modeling of the hippocampus and related areas involved in spatial navigation and episodic memory, the application of biological models to robotic navigation, and the development of software tools to increase modeling productivity. Gorchetchnikov received a PhD in cognitive and neural systems from Boston University. Contact him at tangorn@gmail.com.*

***Ennio Mingolla** is a professor and chair, Department of Cognitive and Neural Systems, Boston University. His research interests include the development and empirical testing of neural network models of visual perception and the transition of these models to technological applications, visual psychophysics, and the computational modeling of brain processes. Mingolla received a PhD in experimental psychology from the University of Connecticut. Contact him at ennio@cns.bu.edu.*

cn **Selected CS articles and columns are available for free at http://ComputingNow.computer.org.**