DOI:10.1145/1787234.1787244

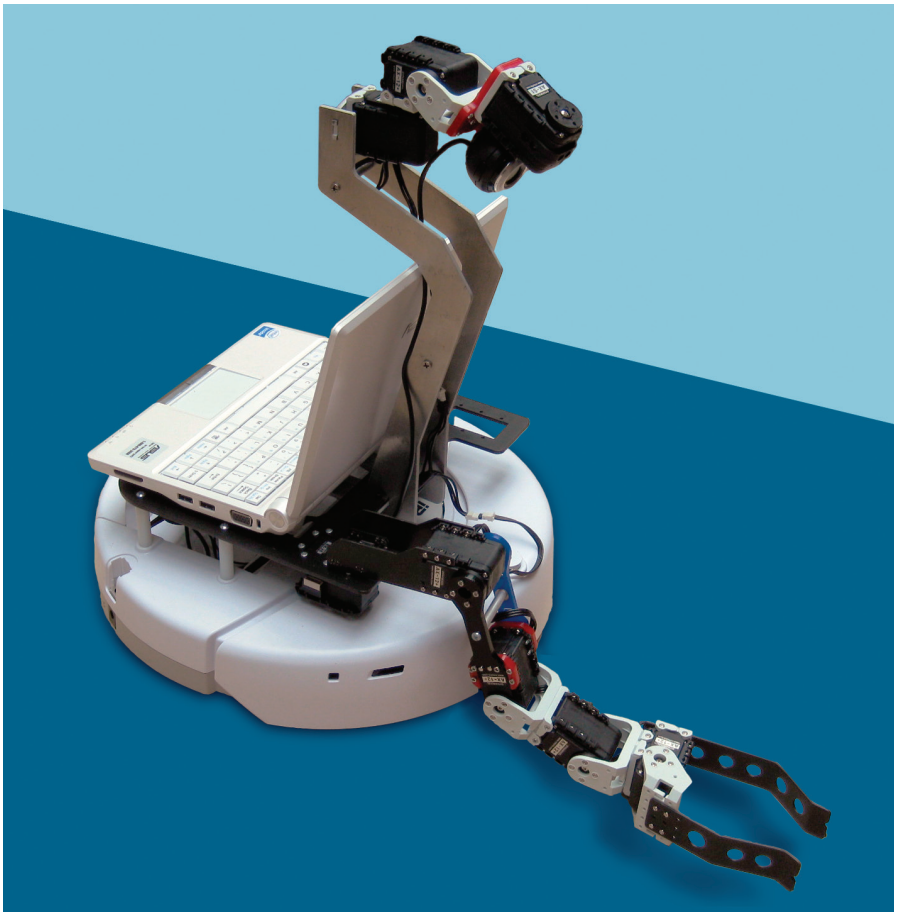
David S. Touretzky

Education

Preparing Computer Science Students for the Robotics Revolution

Robotics will inspire dramatic changes in the CS curriculum.

Beginning in the 1970s, a series of technological advances in computing has repeatedly reshaped the undergraduate computer science curriculum. Affordable bitmapped displays brought GUI interfaces into widespread use, gave us the new field of human-computer interaction, and led CS departments to introduce courses in computer graphics and HCI. The maturation of networking technology that led to the Internet and the Web also spawned a whole spectrum of new courses, from the nuts and bolts of network protocols to the social impacts of online communities. The microprocessor that launched the personal and then wearable computer revolutions, and in conjunction with the growth of wireless networks, produced new types of platforms that are always on and always with us, has led to courses targeting smartphones and PDAs instead of conventional computers. And when inexpensive graphics processors and sound cards grew electronic gaming into a multibillion-dollar business with revenues comparable to the film and music industries,[a] CS departments responded by introducing a variety of multidisciplinary courses in game design.[9]



PHOTOGRAPH BY ETHAN J. TIRA-THOMPSON/CARNEGIE MELLON

Calliope: A prototype Create/ASUS robot with a pan/tilt camera and gripper arm.

Robotics is the leading candidate for the next dramatic change in the CS curriculum. Advances in sensing, actuator, and power technologies are fueling an explosion in robotics comparable to what microprocessors did for computing three decades ago. In a 2007 *Scientific American* article, Bill Gates drew a parallel between today's robotics industry and the computing

a Estimated 2008 revenues from Hoovers.com: motion pictures $33 billion; music $15 billion; computer and electronic games $12 billion.

industry at the start of the PC revolution.[2] He compared today's state-of-the-art industrial robots—priced at tens to hundreds of thousands of dollars—to 1970s era mainframes, while consumer robots resemble 1970s microcomputers: crude, underpowered, and of interest mainly to hobbyists who enjoy tinkering with technology for its own sake. Today's consumer robots include a variety of kits (Lego Mindstorms, VEX Pro, Robotis Bioloid), limited but intriguing toys (Wowee's Robosapien, Pleo from Innvo Labs, Penbo from Bossa Nova Robotics, and a dozen others; many more in Japan), and one astonishingly successful vacuum cleaner: the Roomba; more than five million Roombas have been sold.

Advances in robotics are reported weekly at technology news sites such as Robots.net, while the popular magazines *Robot* and *Servo* are energizing the robotics hobbyist community the way *Byte* and *Dr. Dobbs' Journal* once nurtured amateur computing enthusiasts. Meanwhile, more than 40 nations now have military robotics programs.[3]

We see glimmers of our robotic future in today's self-parking cars, cameras that recognize human smiles, and flying devices ranging from micro-scale robot bees to the airliner-size Eitan UAV. But the robotics revolution will be farther ranging—and a lot more weird—than most of us can envision now. Who in 1971 would have looked at the first Intel microprocessor and predicted eBay, Wikipedia, Google Earth, or "sexting"?

Impediments to Progress

We can help speed the revolution by introducing our undergraduates to state-of-the-art robotics hardware and software. But three factors have stymied progress in robotics education for computer scientists. The first is misconceptions held by some about the nature of the subject. Robotics cannot be taught in CS1. The use of simple robots to teach basic programming concepts dates back to Papert's Logo turtle of the 1970s[5] and Pattis' Karel the (simulated) Robot in the 1980s.[4] More recent examples include a Python-based programming course using the Parallax Scribbler,[1] and a variant of Alice (http://www.alice.org) that can both simulate an iRobot Create and teleoperate a real one via a bluetooth dongle.[8] CS educators must understand that while it might be a good idea to use simple robots to teach students about variables, procedure calls, and while loops, this is not the same as teaching them robotics, any more than making a penguin move around in Alice counts as teaching computer graphics.

Real robotics involves deep, computationally demanding algorithms.

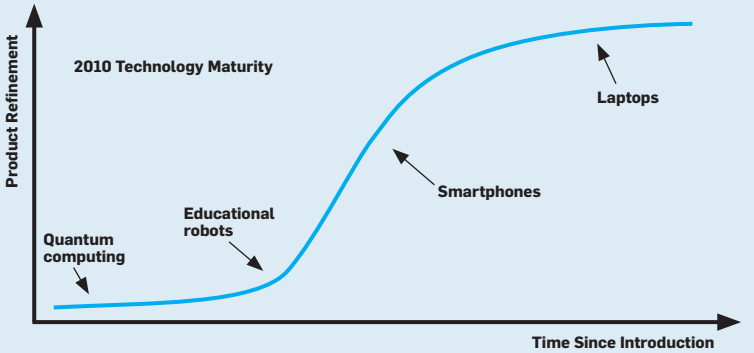
High school robotics contests such as US FIRST, which emphasize the mechanical engineering aspects of the field at the expense of computer science, are another source of misconceptions. The public doesn't always appreciate that the elaborate hardware platforms students construct must be primarily teleoperated because students aren't being taught the kind of software that would allow their robots to act autonomously.

Real robotics involves deep, computationally demanding algorithms. Machine vision, probabilistic localization and navigation, kinematics calculations, grasp and path planning, multi-robot coordination, and human-robot interaction (face tracking, speech and gesture recognition) are core technologies. Today these are found mainly in advanced research labs and graduate-level robotics courses, but they can be made accessible to undergraduates. The time to do that is now.

The second impediment to be overcome is the lack of suitable robot platforms for undergraduate instruction. The devices used in CS1 courses typically have no camera and can't even drive in a straight line without drifting. The Lego Mindstorms kits used in many current college courses are no better. On the other hand, the groundbreaking Sony AIBO robot dog was an excellent instructional platform due to its powerful MIPS processor, rich sensor suite (including a color camera, stereo microphones, and multiple IR range finders and accelerometers), and sophisticated servos with position and force feedback. The AIBO's $2,000 price tag was comparable to high-end laptops of its day. But when Sony abruptly exited the robotics market in 2006, the AIBO had not yet caught on as a teaching platform, and that market niche has yet to be filled.

In 2007 the RoboCup Federation, which oversees robotic soccer competitions worldwide, selected the Nao humanoid from Aldebaran Robotics to replace the AIBO in the Standard Platform League. A few schools are now teaching robotics courses using Naos, but at a retail cost of approximately $16,000, the Nao will remain out of reach for most educators. There are some who believe that educational robots should cost no more than a Lego Mindstorms kit: only a few hundred dollars. They're right, but it will be a while before the economies of mass production can do for AIBO- or Nao-type robots what they've already done for laptops and smartphones. Meanwhile, Mindstorms' widespread and growing use in high schools and

Technology maturation curve.



even middle schools underscores my point that undergraduates require something better. They need robots that can see, with processors that can run the sophisticated algorithms computer scientists should be studying.

As illustrated in the figure on the preceding page, educational robotics is entering the unstable region of the technology maturation curve. For just a few more years, computer scientists will build their own platforms for education and research. In less than a decade this will become infeasible, for the same reason that no individual today builds their own laptop or cellphone. But since highly capable robots are not yet mass-produced consumer products, today's educators must innovate. Several colleagues in the U.S.[b] have found a good solution by mounting a laptop or netbook atop an iRobot Create. The Create—a Roomba without the vacuum—provides an inexpensive mobile base with a few simple sensors, while the laptop provides a Webcam for vision, a WiFi connection, a speaker, and plenty of computing power. The total parts cost can be as low as $600. Anyone who thinks these platforms are too expensive should recall what schools were paying for workstations a few years ago. Readers who would like to put one of these robots together themselves, or purchase a pre-assembled version from a commercial vendor, can find all the necessary information at http://www.Chiara-Robot.org/Create. An enhanced version with a pan/tilt camera and an arm with gripper is presently under development (see the image on the first page of this column).

The final impediment to be overcome is the lack of easy-to-use software. The three major open source frameworks for robotics application development are Player/Stage (http://playerstage.sourceforge.net), ROS (http://www.ros.org), and Tekkotsu (http://www.tekkotsu.org). Player/Stage and ROS have similar philosophies. Both provide a general communication framework for a collection of independent software modules responsible for controlling various types of hardware or providing services such as localization. Both support a wide variety of platforms and devices. And both are designed primarily for research, although Player/Stage in particular has been widely used for education. Modules can be written in any of several languages; the frameworks themselves make no assumptions about representation.

Who in 1971 would have looked at the first Intel microprocessor and predicted eBay, Wikipedia, Google Earth, or "sexting."

Another Approach

Tekkotsu, developed with Ethan Tira-Thompson in my lab at Carnegie Mellon University, takes a different approach. It is implemented in C++ and makes heavy use of abstraction facilities such as templates, multiple inheritance, polymorphism, functors, and namespaces. It offers a common representation scheme for vision, navigation, and manipulation tasks.[6,7] The idea is to provide a unified framework that undergraduates can master in two-thirds of a semester and then move on to working on an interesting final project. Tekkotsu does not strive for universal hardware coverage; instead it provides well-tuned primitives for a small number of educational platforms, including the AIBO and the Create. Other groups are developing Tekkotsu support for additional platforms, including the Nao and various robot arms. But for all three frameworks, more work remains to be done to make advanced robotics technologies easy for non-experts to use.

Our Robotic Future

To predict the future of robot software, look at the history of graphics software. Early graphics programming was done by turning pixels on and off, just as early robot programming was done by turning motors on and off. But graphics has developed into a wonderfully rich field that includes specialties such as Web design, game design, and scientific visualization, where the focus is on principles of visual aesthetics or the graphical presentation of information, not low-level details of rendering algorithms or GPU programming. Web and game designers rely on computer scientists for the tools of their trade, but they have different skill sets and are not themselves computer scientists. The applications of computer graphics have outgrown the confines of a single discipline.

I believe our notions about robot programming will likewise broaden in the coming years. Our students will create the technologies that make this possible. Better algorithms for perception and manipulation, and high-level frameworks for robot instruction will enable robotics application development by a diverse population of users and innovators, some of whose job descriptions are as unforeseeable today as "Web designer" was in 1971. That will be one unmistakable sign that the robotics revolution has arrived. Let's get started. C

b Jeff Forbes at Duke University, Chad Jenkins at Brown University, Monica Anderson at the University of Alabama, and Zach Dodds at Harvey Mudd College have been at the forefront of this work.

References

1. Balch, T. et al. Designing personal robots for education: Hardware, software, and curriculum. *IEEE Pervasive Computing 7*, 2 (Feb. 2008), 5–9.
2. Gates, B. A robot in every home. *Scientific American* (Jan. 2007), 58–65.
3. Levinson, C. Israeli robots remake battlefield. *The Wall Street Journal* (Jan. 12, 2010), A12.
4. Pattis, R.E. *Karel the Robot: A Gentle Introduction to the Art of Programming, Second Edition*, Wiley, New York, 1995.
5. Solomon, C. Logo, Papert, and Constructionist Learning, 2010; http://logothings.wikispaces.com.
6. Touretzky, D.S. et al. Dual-coding representations for robot vision in Tekkotsu. *Autonomous Robots 22*, 4 (Apr. 2007), 425–435.
7. Touretzky, D.S., and Tira-Thompson, E.J. The Tekkotsu "crew": Teaching robot programming at a higher level. In *Proceedings of the AAAI Symposium on Educational Advances in Artificial Intelligence* (EAAI-10), (Atlanta, Georgia, July 13–14, 2010).
8. Wellman, B., Davis, J., and Anderson, M. Alice and robotics in introductory CS courses. In *Proceedings of the Fifth Richard Tapia Celebration of Diversity in Computing Conference* (Portland, OR, 2009), 98–102.
9. Zyda, M. Computer science in the conceptual age. *Commun. ACM 52*, 12 (Dec. 2009), 66–72.

David S. Touretzky (dst@cs.cmu.edu) is a research professor in the Computer Science Department and the Center for the Neural Basis of Cognition at Carnegie Mellon University in Pittsburgh, PA. He was named a Distinguished Scientist by ACM in 2006.

Research funded by National Science Foundation award DUE-0717705.