Robotics Across the Curriculum

Elizabeth Sklar[1,2], **Simon Parsons**[1,2] **and M Q Azhar**[2]
[1]Dept of Computer and Information Science
Brooklyn College, City University of New York
2900 Bedford Ave, Brooklyn, NY, USA 11210
[2]Dept of Computer Science
Graduate Center, City University of New York
365 5th Avenue, New York, NY, USA 10016
sklar,parsons@sci.brooklyn.cuny.edu
mazhar@gc.cuny.edu

Abstract

We describe a comprehensive program using educational robotics as a hands-on, constructionist learning environment, integrated into teaching across the undergraduate computer science curriculum. Five courses are described in detail. For the three courses which have been offered multiple times, evaluations were conducted to assess students' attitudes towards the robotics-based curriculum. These results are presented here. Lessons learned are shared, and new directions for the future are highlighted.

Introduction

For the past six years, we have been bringing LEGO robots into university classrooms to enhance courses on introductory programming and computer science (both for computer science majors and non-majors), object-oriented programming, artificial intelligence, embodied agents and multiagent systems. We have also experimented with the use of Sony AIBO robots and are currently investigating other platforms for teaching.

There is a rich history of instructors bringing robotics into undergraduate classrooms over the last 10 years as a means to teach a range of topics. Early work explored the use of robotics for teaching introductory programming (Stein 1996; Meeden 1996; Lawhead *et al.* 2002; Gossett & Flowers 2003). Other efforts examined the use of robotics for interdisciplinary subjects and for advanced undergraduate computer science topics such as artificial intelligence. Beer, Chiel, & Drushel (1999) used autonomous robots to teach an interdisciplinary science and engineering, attracting students from a wide range of majors (from biology and neuroscience to physics). Klassner (2002) outlines early experiences using LEGO robotics in artificial intelligence classes, pointing out the lack of a LISP-based programming environment, which the author later addressed in (Klassner & Anderson 2003). Kumar (2004) describes the integration and evaluation of robotics into artificial intelligence curriculum for undergraduates, weighing the increase in student excitement against the increased preparation time for instructors.

Others have detailed specific experiences, hardware and software problems and solutions for integrating LEGO Mindstorms into a range of undergraduate classrooms (Martin 1994; Fagin 2003; Mayer, Weinberg, & Yu 2004; Jacobsen & Jadud 2005; Sklar, Parsons, & Stone 2004). Some of the issues highlighted are lack of a simulator that students can use for debugging when they are not in the lab, longer preparation time for instructors (not only designing new curricula and projects, but also organizing and maintaining robotics equipment), increased time spent away from the curriculum for students—particularly time spent learning how to program the robots in non-programming classes (like AI) or time spent fixing hardware problems in non-hardware classes.

Several approaches have been taken to address these issues by providing programming interfaces that are designed for easy debugging and offer simulation capabilities. One is *Pyro* (Blank, Meeden, & Kumar 2003), based on Python, which provides solutions for several of these problems, moving beyond LEGO and defining a universal programming interface for several robotics platforms, e.g., the AIBO, Khepera and Pioneer robots, and includes a simulation environment. Another is *Tekkotsu* (Touretzky & Tira-Thompson 2005), based in C++, which uses an object-oriented, event-passing architecture and was designed to work with the Sony AIBO, though it can also be compiled for other platforms.

This paper is organized as follows. The first section describes the courses that we have developed for teaching with robotics across the curriculum. The second section presents results of evaluations we have conducted in the three courses that have been taught more than once. Finally, we conclude with a summary of lessons learned and future directions.

It should be noted that all the courses described have been taught at Brooklyn College, one of 19 campuses that comprise the City University of New York, an urban university with 400,000 students. Approximately 12,000 students are enrolled in undergraduate programs at Brooklyn College, a public liberal arts school that primarily attracts working-class students; many are the first in their families to attend college and most hold part-time jobs while they go to school. The student body is 60% female; and nearly half the students are ethnic minorities (30.7% African American and 11% Hispanic).

Teaching

Our university teaching experiences with robotics began in Spring 2001 and have grown from one introductory robotics

course for non-engineering computer science students to encompass a spectrum of seven courses, ranging from exploring robotics for non-majors to introductory programming for majors and advanced artificial intelligence for graduate students. Five of these courses have been taught at the time of writing: *Exploring Robotics* (for non-majors), *Computing: Nature, Power and Limits* (for non-majors), *Object-Oriented Programming* (for intermediate majors), *Artificial Intelligence* (for advanced majors), and *Introduction to Robotics* (for advanced majors). The middle three of these have been taught several times. The remaining two courses are scheduled to be taught in Spring and Fall 2007; these are: *Introduction to Computing Using C++* (for first-semester majors) and *Advanced Programming Techniques* (for second-semester majors). This section describes each of the five courses which have already been offered.

Exploring Robotics

This course provides an introduction to robotics for advanced students who are not computer science majors, through the use of case studies and project-based activities. Students work together in small groups on a series of two-week creative projects, using robots to address meaningful and socially important issues, such as urban search and rescue or elder care. Along the way, students are introduced to the fundamentals of robotics (including aspects of mechanical design) and elementary programming within a graphical environment called RoboLab[1] (Erwin, Cyr, & Rogers 2000). A series of seven scaffolded units build in complexity in terms of the robot solution, the task environment and the task(s) to be accomplished. Each unit is accompanied by a case study, with which to situate the technical material being introduced. Following is a brief outline of each unit:

1. *Introduction to Robotics:* this unit outlines basic robot construction and uses the "BigDog" project (Hambling 2006) as a case study.
2. *Simple Go-bot:* this unit introduces students to basic control ideas; the case study is the DARPA Grand Challenge (Thrun *et al.* forthcoming; Gutierrez *et al.* 2005).
3. *Dancing Go-bot:* this unit brings in touch sensors and the programming concept of *iteration*, using robotic dance as a case study.
4. *Home-helper Go-bot:* this unit explains the programming concept of *branching* and the notion of event-driven programs; the case study presents the Roomba[2].
5. *Robot Teams:* this unit discusses multiple robots operating in a complex, dynamic environment and uses RoboCup Soccer as the case study; the technical challenge is RoboCupJunior soccer, which only requires a light sensor to perform the task of finding the ball, localizing, and kicking towards the goal.
6. *Search-and-rescue Go-bot:* this unit combines touch and light sensors, making more sophisticated use of the light sensors to recognize multiple light levels. The case study is Urban Search and Rescue (USAR) robotics (Kleiner 2006), and the labs use the RoboCupJunior Rescue challenge (Sklar 2004) in which robots attempt to locate dummy victims in a mock collapsed building.
7. *State-of-the-art Robotics:* this unit presents exciting new topics in the field of robotics; the case study currently being used is in the area of evolutionary robotics (Zykov *et al.* 2005).

This course has become quite popular, and currently (Fall 2006) we are offering five sections of the course with a total enrollment of 88 students.

core technical topic(s)	*case study*
Introduction to Computers and Networks	tele-operated robots
Algorithms and Computer Languages	dancing robots
Machine Architecture, Data Representation and Storage	self-reproducing machines
Event-driven Programming	home-helper robots
Solvability and Feasibility	urban search and rescue robots
Programmer-defined functions	evolutionary robotics
Security, Privacy, Encryption and Plagiarism	security robots

Table 1: Topics covered in Computing: Nature, Power and Limits.

Computing: Nature, Power and Limits

This course offers an introduction to computer science and programming through the use of project-based educational robotics activities for beginning college students who have not yet declared a major. The course is part of the *core curriculum* required of all undergraduates at Brooklyn College, and our department is experimentally offering several "flavors" of the course to provide a variety of interdisciplinary, applied, context-based entries into the world of computing, as part of a larger project that is attempting to broaden the demographics of students pursuing careers in computer science, particularly aiming to attract female and minority students[3]. The course is organized as above, into seven curricular units, where each unit explores a technical topic and is framed with a case study and application area for hands-on laboratory work. The curricular areas are defined by the core course, and robotics topics provide the flavor for this particular section. The areas are shown in table 1.

We are currently (Fall 2006) offering one section of this course, with an enrollment of 22 students. A formal evaluation is being conducted, with pre- and post- attitudinal surveys. In addition, a standard academic assessment for this flavor of the course will be compared with that of the other

[1] `http://www.ceeo.tufts.edu/robolabatceeo/`
[2] `http://www.irobot.com/`
[3] `http://bridges.brooklyn.cuny.edu`

flavors and the non-flavored course; altogether, there are 29 sections of the course (three of which are flavored, and one is the robotics flavor) with a total enrollment of approximately 600 students. Formal evaluation results are forthcoming.

Object-Oriented Programming

This course introduces object-oriented programming using Java, with robotics used as a supplemental educational tool. This "flavored" section of this course has been offered four times since Fall 2004, employing LEGO Mindstorms robots and the Java-based leJOS[4] programming environment. Initially (i.e., during Fall 2004), we introduced educational robotics activities as a project toward the end of the semester, and feedback from the students was positive. Some students said that we should introduce robotics from the beginning of the semester, and since we agreed that time was short for robotics, during Spring 2004, we started integrating robotics activities throughout the course, eventually replacing significant portions of the take-home exercises (from non-robotics offerings of the course) with hands-on lab activities. Every semester, we have modified our curriculum based on our classroom experience and students' feedback.

All of our labs have pre-lab and post-lab activities. The idea of the pre-lab is to make the students' in-class lab activities more efficient. During the lab session, we give the students problems such as line-following, search and rescue, and RoboCupJunior soccer. The post-lab activities give closure to the lab. In addition, we let students come to the lab outside of class time, so they can finish work they did not complete in class. This also gives students more time for problem-solving. We also incorporated a "Showcase" in Fall 2005 where students can show off their projects to their peers and to spectators. This event makes students more engaged in the curriculum (Beer, Chiel, & Drushel 1999). Finally, students work in groups of two or three during lab activities which increases students' collaborative skills.

Artificial Intelligence

The metaphor of intelligent agents is a way of bringing together the many strands of work carried out under the banner of AI and presenting them to students in a convincing way. The topics in our AI syllabus include: agency, control architectures, search, knowledge representation, logic, and planning. Students engage in two robotics projects during the term, using LEGO Mindstorms and the Not Quite C (NQC) language. The first project is based on RoboCupJunior rescue (Sklar 2004), expanded to incorporate climbing and descending a ramp or detecting and avoiding obstacles, in order to make the task harder. In the second project, students are confronted with a "grid world" delineated with black lines where some of the squares contain colored figures. The challenge is to survey the grid, identifying the positions of the figures, and then re-position the robot (at the start) and move to the figures in a pre-specified order in the lowest possible time. The idea behind the challenge is to bring in some of the concepts related to *search* that the students have covered in the course, combining these with the reactive techniques from the first project (which are still required to move around the grid). Since the robots cannot localize, this is a hard challenge, but it is within the capabilities of the more able students.

[4] `http://lejos.sourceforge.net/`

Introduction to Robotics

This course is intended as the capstone of our robotics curriculum for undergrduates. Unlike the other courses we have described here, this course not just uses robots, but is actually about robots. It is intended as a broad introduction to the field, covering topics such as locomotion, kinematics, perception, localization and navigation. This theoretical background is accompanied by extensive practical work, with at least one hour of lab time for every hour in a conventional lecture. The idea of the practical work is to reinforce the main lessons explained in the theoretical work (the difficulty of navigating by dead-reckoning for example) as well as giving the students a feel for the kind of work involved in robotics research.

In previous offerings of the course, taught to graduate students, we ran the first few practicals using LEGO Mindstorms, as a simple platform that the students could easily master, before moving onto the more challenging AIBO. In these offerings we used the LEGO robots to do a set of increasingly complex tasks—a race that involved line-following, some simple flocking that involved heading towards a light source, and a contest that involved line-following plus a pursuer-evader segment—before moving to the AIBO for work on navigation. Our current offering includes students who have already taken courses that use the Mindstorms and jumps right into using the AIBOs, ending with a multi-week project.

Evaluation

From the start of this effort, we have been keen to evaluate the impact of the robotics curriculum. While it is clear that some students are excited by the possibilities of robotics, and welcome the chance to work on robots as part of the classes we have decribed above, we have felt it was important to get as much data as possible on students' attitudes toward the robot work. This has given us feedback which has been important in improving the courses, and has also given us hard data on our efforts which have convinced us that what we are doing has value[5].

As an example of the results that we have obtained, in this section we present data from several offerings of the "Object-oriented Programming" and "Computing: Nature, Power and Limits classes. The results of this evaluation for the Artificial Intelligence course have been presented elsewhere (Sklar, Parsons, & Stone 2004), and the remaining classes are, in their current form, being offered for the first time this semester, so we have no data on them as yet.

[5] This data has also helped us to persuade our department, and the administration generally, to provide continued support for our robotics efforts.

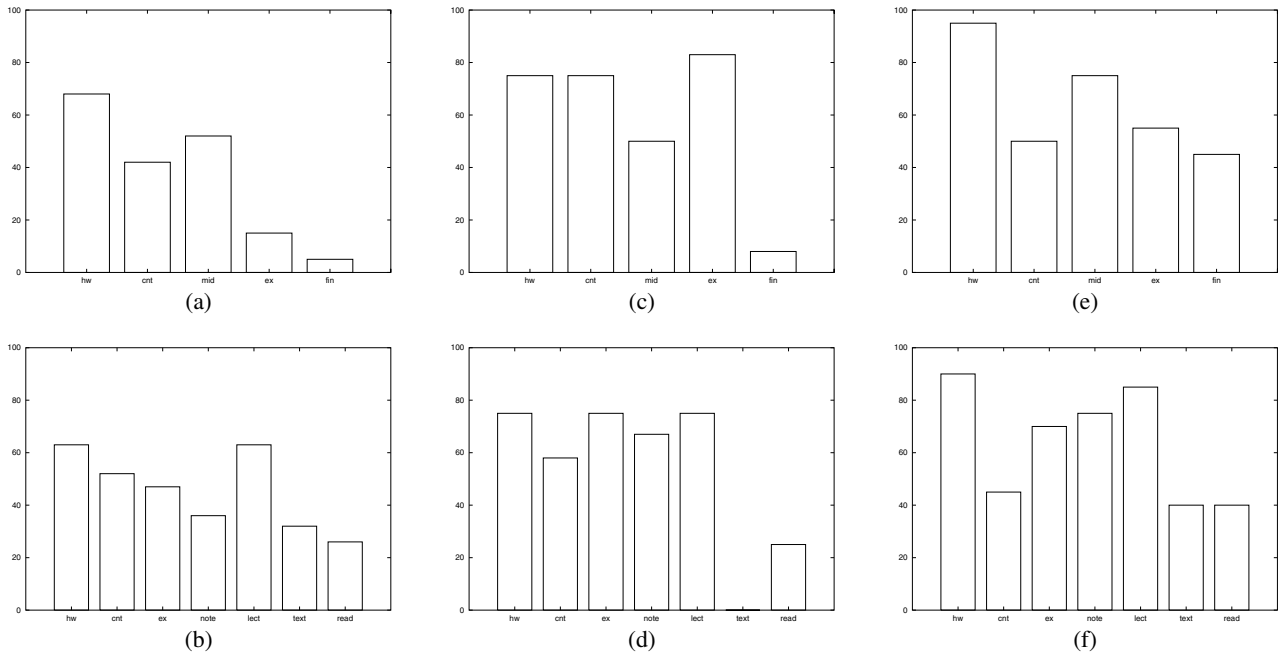

Figure 1: Evaluation results from object oriented programming. See text for explanation.

Object-oriented Programming

Figure 1 shows some of the data collected from questionnaires administered to sections of the object-oriented programming class that used robotics. Figure 1 (a) and (b) are from Spring 2005, (c) and (d) are from Fall 2005 and (e) and (f) are from Spring 2006. These graphs show the proportion of students who indicated various elements of the course were either helpful in *demonstrating* their understanding of programming concepts (graphs (a), (c) and (e)) and helpful in *learning* programming concepts (graphs (b), (d) and (f)). This information was obtained by asking students to suggest which of the listed elements they found helpful. The columns in (a), (c) and (e) are, from left to right, those for homework, robot contest, midterm, the classroom exercises with robots, and the final. The columns in (b), (d) and (f) are, from left to right homework, robot contest, classroom exercises with robots, lecture notes, lectures, textbook, and additional reading material.

Based on the feedback we received from students, we changed the course considerably between Spring 2005 and Fall 2005. In the spring, the robotics component, though present throughout the semester[6] was an addition to the course rather than an integrated part of it—preparation for the labs that used the robots (the "pre-labs" described in the previous section) was assigned in addition to the regular homework, the lab was physically separate from the usual classroom, and students could only work on the robots during class time. Since Spring 2005, we integrated the preparatory work for the labs into the homework, made the lab available outside class hours, and introduced a "robotics showcase" event at which students from all the sections of classes in which we were using robots were able to compete together. As a result, students' appreciation of the robot contest and the classroom exercises grew, in particular, the appreciation of the exercises grew greatly.

Other indications of student attitudes come from the open questions on the questionnaire. One such question was "Did the experience with robotics help you at all". Fully 100% of the 12 students in Fall 05 for whom we have data said "yes", and 70% of the 20 students for whom we have data in Spring 06 said "yes" (only three, 15%, said "no", the others gave ambiguous answers). This compares with only 10 of the 19 for whom we have data in Spring 2005 (52%).

The following comments are taken from the answers to the same question, giving an indication of the reasons behind the positive responses:

- "It makes it more fun to learn Java because you see that you actually accomplish something if your robot does what you need it to do."
- "It made programming a lot more interesting."
- "It was fun seeing computer science in action outside the narrow screen of the monitor. Actually was an example of real-world programming scenarios."

Those students who felt the robotics elements were not helpful seem to be students who would have been happy with a more traditional course. The following is typical: "It was a bit of a distraction from learning OOP and the Java language. The details of the robotics segment didn't help me to remember the details of OOP."

[6] As described above, there was one previous offering of the course, in Fall 2004, in which the robots were brought in only at the end of the course and so were even less integrated.

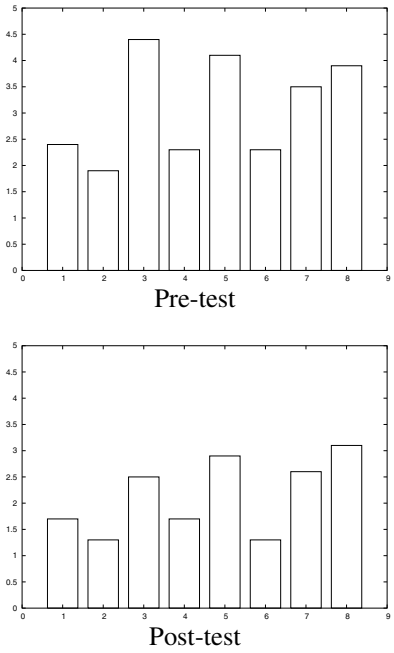

Figure 2: Evaluation results from Computing: Nature, Power and Limit, Spring 2006. See text for explanation.

Computing: Nature, Power and Limits

For the introductory computing class, we adminstered pre- and post-tests to the students. These tests contained a range of questions, some of which asked students to rate the extent to which they agreed with the following statements using a five point scale (ranging from 1=Strongly disagree to 5=Strongly agree):

1. Computer programming is hard for me.
2. I will not use a lot of computer science when I get out of college.
3. I can get good grades in computer science.
4. I am no good in science.
5. I study computer science because I know how useful it is.
6. I will choose a career in computer science.
7. A degree in computer science will allow me to obtain a well-paying job.
8. Web design is fun.

The average scores given to these questions on each test are given in Figure 2[7]. It seems a reasonable assumption that students with a postive view of computer science should report high values for questions 3, 5, 6, 7, 8 and low values for questions 1, 2 and 4.

The results we have are ambiguous. The scores that we would expect to be lower due to a positive experience in computer science (those for questions 1, 2 and 4) do indeed decrease. However, the scores that we would expect to increase due to a postive experience in computer science (questions 3 and 5 through 8), also decline. Thus, while students, on average, are less inclined to believe that computer programming is hard after the course, they are also less inclined to choose a career in computer science.

[7]The questions were asked in different ways on the two tests—each question was asked once in a positive way and once in a negative way—and the scores reported correct for this.

Section	Average	stdev
Fall 2005 without robotics	54	22
Fall 2005 with robotics	57	16
Spring 2006 with robotics	65	8

Table 2: Student marks for the core computing course.

More encouraging are the results from the final exam for the class. Here we have data from three sections of the class, two in Fall 2005, and one in Spring 2006. All sections were taught by the same instructor. as described above, one of the sections in Fall 2005 was taught using robotics (14 students enrolled) and one was not (16 students). The section in Spring 2006 was taught using robotics (18 students). The average marks for the final exam and the standard deviations of those marks, are given in Table 2. These are the marks for the parts of the exam that are common to all the sections (in other words the bits that are not due to the robotics material), in order to avoid any bias due to the robotics material being easier than the material it replaced. The marks are lowest for the section without robotics—both sections that used robotics are rather higher. The results are not statistically significant, as can be seen from the standard deviations, but the trend is worth noting.

Summary

We have evolved over the past five years a comprehensive set of courses that allow us to integrate educational robotics as a hands-on learning environment across the undergraduate computer science curriculum. Students with no programming background and those almost finished with their major have had opportunities to take these courses, and we have conducted both quantitative and qualitative evaluations with each course that we have offered. The results have shown that the students, for the most part, find the experiences with robotics to be enjoyable and motivating. As instructors, we have found that the course preparation time increases and that we have to design our course materials carefully to ensure that the curricular material we need to cover is properly presented and that the link between the curriculum and the robots is explicit.

Our most recent work involves development of a universal interface and simulator for educational robotics. While the use of low-cost robotics platforms in the classroom has many attractive features, there are still several shortcomings that must be overcome in order to realize the full potential of educational robotics as a practical learning environment. Particularly since time for "practice" is limited, there is a need to reduce debugging time when using robots in instructional settings. Most robotics programming interfaces are

designed for university-level or late high school students and are implemented as extensions to existing languages (such as C/NQC, C++/Brickos, Java/leJOS and Python/Pyro). We have been developing an agent-oriented, behavior-based interface framework targetting students with no programming experience and designed to ease them from a graphical interface into a text-based structured language (Azhar, Goldman, & Sklar 2006). This framework has the capability to interact with multiple agent platforms and a simulator through an XML-based agent behavior language. Our longterm goal is to create a standard middle ground that can act as a "magic black box", providing a seamless transition between simulator-based learning and debugging environments and a range of robotic platforms.

Acknowledgements

This work was partially supported by NSF ITR #02-19347, NSF IIS #03-29037, NSF BPC #05-40549, and DOE GEAR-UP #P334A050232.

References

Azhar, M. Q.; Goldman, R.; and Sklar, E. 2006. An agent-oriented behavior-based interface framework for educationa robotics. In *Agent-Based Systems for Human Learning (ABSHL) Workshop at Autonomous Agents and MultiAgent Systems (AAMAS-2006)*.

Beer, R. D.; Chiel, H. J.; and Drushel, R. F. 1999. Using autonomous robotics to teach science and engineering. *Communications of the ACM* 42(6).

Blank, D.; Meeden, L.; and Kumar, D. 2003. Python robotics: an environment for exploring robotics beyond lego. In *Proceedings of the 34th SIGCSE technical symposium on Computer science education*, 317–321.

Erwin, B.; Cyr, M.; and Rogers, C. B. 2000. Lego engineer and robolab: Teaching engineering with labview from kindergarten to graduate school. *International Journal of Engineering Education* 16(3).

Fagin, B. 2003. Ada/mindstorms 3.0: A computational environment for introductory robotics and programming. *IEEE Robotics and Automation Magazine* 10(2):19–24.

Gossett, K. A., and Flowers, T. R. 2003. Using robots and simulation to teach problem solving in an introductory course in computing and information technology. In *Proceedings of the 2003 Military, Government, and Aerospace Simulation Symposium*.

Gutierrez, A.; Galatai, T.; Gonzalez, J. P.; Urmson, C.; and Whittaker, W. L. 2005. Preplanning for high performance autonomous traverse of desert terrain exploiting a priori knowledge to optimize speeds and detail paths. Technical Report CMU-RI-TR-05-54, Robotics Institute, Carnegie Mellon University.

Hambling, D. 2006. Robotic 'pack mule' displays stunning reflexes. *NewScientist.com news service*.

Jacobsen, C. L., and Jadud, M. C. 2005. Towards concrete concurrency: occam-pi on the lego mindstorms. *SIGCSE Bulletin* 37(1):431–435.

Klassner, F., and Anderson, S. 2003. Lego mindstorms: Not just for k-12 anymore. *IEEE Robotics and Automation Magazine* 10(2):12–18.

Klassner, F. 2002. A case study of lego mindstorms suitability for artificial intelligence and robotics courses at the college level. In *Proceeding of the 33rd SIGCSE Technical Symposium on Computer Science Education*, 8–12.

Kleiner, K. 2006. Better robots could help save disaster victims. *NewScientist.com*.

Kumar, A. N. 2004. Three years of using robots in an artificial intelligence course: lessons learned. *Journal on Educational Resources in Computing (JERIC), Special Issue on robotics in undergraduate education, part I* 4(3).

Lawhead, P. B.; Duncan, M. E.; Bland, C. G.; Goldweber, M.; Schep, M.; Barnes, D. J.; and Hollingsworth, R. G. 2002. A road map for teaching introductory programming using lego mindstorms robots. In *ITiCSE-WGR '02: Working group reports from ITiCSE on Innovation and technology in computer science education*, 191–201.

Martin, F. 1994. Circuits to Control: Learning Engineering by Designing LEGO Robots. Ph.D. Dissertation, MIT.

Mayer, G.; Weinberg, J. B.; and Yu, X. 2004. Teaching deliberative navigation using the LEGO RCX and standard LEGO components. In *Accessible Hands on Artificial Intelligence and Robotics Education: Working Papers of the 2004 AAAI Spring Symposium Series*, 30–34.

Meeden, L. 1996. Using robots as introduction to computer science. In *Proceeding of the Ninth Florida Artificial Intellegence Research Symposium (FLAIRS)*, 473–477.

Sklar, E.; Parsons, S.; and Stone, P. 2004. Using RoboCup in university-level computer science education. *Journal on Educational Resources in Computing (JERIC), Special Issue on robotics in undergraduate education, part I* 4(2).

Sklar, E. 2004. A long-term approach to improving human-robot interaction: RoboCupJunior Rescue. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*.

Stein, L. A. 1996. Rethinking cs101: Or, how robots revolutionize introductory computer programming. *Computer Science Education*.

Thrun, S.; Montemerlo, M.; Dahlkamp, H.; Stavens, D.; Aron, A.; Diebel, J.; Fong, P.; Gale, J.; Halpenny, M.; Hoffman, G.; Lau, K.; Oakley, C.; Palatucci, M.; Pratt, V.; Stang, P.; Strohband, S.; Dupont, C.; Jendrossek, L.-E.; Koelen, C.; Markey, C.; Rummel, C.; van Niekerk, J.; Jensen, E.; Alessandrini, P.; Bradski, G.; Davies, B.; Ettinger, S.; Kaehler, A.; Nefian, A.; and Mahoney, P. forthcoming. Stanley, the robot that won the DARPA grand challenge. *Journal of Field Robotics*.

Touretzky, D. S., and Tira-Thompson, E. J. 2005. Tekkotsu: A framework for AIBO cognitive robotics. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*. Menlo Park, CA: AAAI Press.

Zykov, V.; Mytilinaios, E.; Adams, B.; and Lipson, H. 2005. Self-reproducing machines. *Nature* 435(7038).