

Enhance Students' Hands-On Experience With Robotics

Fang Tang

Computer Science Department
California State Polytechnic University, Pomona
Pomona, CA, 91768
Email: ftang@csupomona.edu

Abstract

This paper presents three different ways to include robotics into the undergraduate program. The three possible ways are robotics classes, AI classes, and programming classes. The author believes that using robots in classrooms can greatly motivate students' interests in robotics, AI, and more general, computer science. The paper first describes the appropriate hardware and software platforms for each class, and then discusses the challenging issues of using robots in classrooms.

Introduction

In recent years, the advances in robot hardware and software design have made it possible for bringing robots into the classroom, especially for undergraduate students. The introduction of robotics into undergraduate programs not only has the potential to enhance students' hands-on practices and real world experiences, but also motivates them for pursuing advanced education in AI and robotics, which in the long term benefits the whole AI and robotics society.

In general, robotics can be incorporated within the undergraduate curriculum in three different ways: (1) for robotics classes that are specifically designed to teach students how to implement the intelligent control of robots; (2) for traditional AI subjects, such as planning, genetic programming, learning, etc., where robotic platforms can be used to validate the approaches in a concrete way; and (3) for introductory level programming classes or other specific topics such as GUI design, software engineering, and compiler design, where robotic projects can be used to facilitate real world experiences for the students and motivate their interests in the various topics. The different emphases of the courses present many challenges to include robotics in classroom: selecting the appropriate robot hardware and software platforms based on different course requirements; balancing the efforts spent on programming robots and learning a specific subject, either an AI topic or a programming language, using robots as tools; evaluating the course outcome; minimizing the time that an instructor needs to maintain the programming platform; and designing the course based on the characteristics of the institution.

The rest of the paper is organized as follows. I first describe the three types of classes where robotics can be introduced. Then I discuss some challenging issues for using robots in a classroom.

Three Ways To Include Robots

Robotics classes

Robotics itself contains a lot of subfields, ranging from lower level control, such as sensory data processing, behavior-based control, to higher level control, such as planning, reasoning, etc. Within the domain of computer science, the students should be more concerned with the software components that are used to process the sensory data and to produce the motor control commands such that the robots could behave in an intelligent way. The mechanical or engineering aspects of the robotics are not the focuses of the discussion here. Ideally, to fully grasp the idea on how to program an intelligent robot, students need to first learn the lower level control mechanisms, such as building robot behaviors, and then learn more complex mechanism such as map building, reasoning, etc. The layered abstraction framework presented in (Crabbe 2006) gives some detailed explanation of the components that construct a survey type of robotics class. However, this is a very comprehensive list of subjects that are very difficult to cover in a quarter-based system. The objective would be for students to learn that there are different layers of control for a robot, and a subset of the layers across several levels can be presented without the loss of generality.

The theoretic basis and programming practices are equally important in a robotics class. Students are expected to understand the underline principles of controlling a robot or multiple robots, and to apply them on simulated or physical robots. Thus, students are also expected to learn various APIs (or even a new language) quickly and to develop libraries of code that they could use for further development. Pre-built robotic hardware and software platforms are recommended for the robotics class so that students can spend more time on applying the theory learned in class to real applications. When considering the hardware platform, we should provide students the opportunities to learn the functionalities of a broad range of sensors, instead of limiting on one or two. Of course, we also need to consider the bud-

get and the available working space. The K-Team Robots, such as the Khepera, have several selling points that make them stand out among the other robotic platforms presented in (Dodds *et al.* 2006). These points are: a good quality/price ratio; the option for a large set of sensors, which also make them suitable for research purposes; and the smaller size. In addition to physical robots, the simulation also plays an important role in developing robotic program. The simulation software should be easy to maintain and the software code developed in the simulation should be easily migrated to physical robots. The Player/Stage (Gerkey, Vaughan, & Howard 2003) and the 3-D Gazebo (Koenig & Howard 2004) are one of the good programming environments supporting Unix-flavor systems. The Cyberbotics Webots (Dodds *et al.* 2006) provides support for Windows systems. The above hardware and software systems should be sophisticated enough for students to learn the various aspects of robotic programming.

AI classes

Comparing with robotics, Artificial Intelligence is a much broader topic, and could be applied to many different areas. Robotics is one of the areas and provides a concrete way of using AI in real world applications. Although there are still a lot of concerns about using robotics to teach AI, many instructors have pointed out the benefits of including robotics: the uncertainty of robotic application helps students to understand the importance of certain AI technique (Blank *et al.* 2006; Veloso *et al.* 2006); and the interaction with physical agents motivates the students to understand hard AI topics and apply the techniques on real world problems (Greenwald *et al.* 2006). Some instructors have shown that by including robotics, the enrollment of the AI classes has increased and more students are willing to pursue a higher degree in AI and robotics (Klassner 2006). It is hard to tell whether including physical robots into the AI classroom is a good practice or not, what we can do is to carefully design the class with a good balance of theories and experiments, and to assess the learning outcome and student evaluations in the end.

The purpose of a general AI course is to cover the breadth of the field of AI, the topics that have emerged over the past fifty years of AI research, such as problem-solving, logic, planing, reasoning, learning, etc. It could also be a special topic on one of the subfields, such as machine learning. Students should first focus on understanding the various AI techniques and then apply these techniques to applications to observe the results. The design of the course should focus less on the low-level details of programming robots. Pre-built hardware and software platforms are highly recommended for AI classes. Many researchers have done further development on existing simulation software to make the programming of robots easier, such as the Pyro toolkit (Blank *et al.* 2006), hiding the low level details of programming robots; and the RCXLisp library (Klassner 2006), a programming library that is readily accessible to undergraduates and easily maintained by instructors. These advanced APIs provide a better environment for students to apply AI techniques in a fast and easy way, and achieve the effect of real world applications.

Programming classes and others

In addition to the above two types of courses where robotics naturally belongs to the curriculum, robots can also be used in programming classes to motivate the students' interests in learning abstract problem solving skills. Some instructors have started to use robots in their programming classes, especially introductory programming classes (ven Lent 2004). In introductory programming classes, for example, CS 140-141: Introduction to Programming and Problem Solving taught at California State Polytechnic University, Pomona (Cal Poly Pomona), the students often focus on the mechanics of the Java language for solving one or two specific problems and they find it difficult to see the importance of programming. Robotic platform provides a very interesting test bed for students to practice what they have learned in class to real world problems. Robotic examples are motivating since they are concrete examples, and students can easily see the impact of programming and solve the problems that appear during the execution. By doing it, students can also accumulate the problem solving skills and apply them for real world applications.

In a programming class, learning how to program robots is not our focus, instead, we should focus on using robots as tools to facilitate and enhance students' hands-on experience and problem-solving skills. All the traditional topics such as variables, control statements, arrays, I/O, etc., are still covered in class, with the introduction of a new set of programming exercises using robotic applications. To maximize the amount of time students spent on programming, the hardware and software platform needs to be pre-built and ready to use with little maintenance on both the teacher's side and students' side. The APIs should be easy to learn and use in various low level applications. Libraries of code that implement low level robot behaviors, such as "go to goal", "avoid obstacle" can be provided to students to hide the low level details of programming robots. The hardware platform for the introductory programming class does not require those very powerful or versatile robots, as long as the robots can interact with the world and possess some basic sensing capabilities, for example, Lego Mindstorms (Dodds *et al.* 2006) would work fine in these introductory programming classes. Although the simulation already provides a very interesting and interactive programming environment for the students, I believe that the introduction of physical robots would motivate the students more through physical interaction. It is also a good example to show the reality that the perfect program in simulation can sometimes result in errors in the real world.

In addition to the above introductory level programming classes, robots have also been used in the classroom to teach the designing of Graphical User Interface (Challinger 2005) and compiler construction (Xu & Martin 2006). For such topics that do not have a strong relation with robotics, more effort will be needed to prepare a ready-to-be-used robotic platform that hides the low level details of robot programming.

Challenging Issues In Using Robots In Classroom

Although robotics can be possibly included in the above three different types of classes. There are still many challenging issues.

Choosing the right platform

There are many hardware and software platforms available for developing robotic software. What is the appropriate choice? These are the factors that need to be considered when selecting the right platform:

- The different focus of a course requires different robotic software APIs. For example, the software API for a robotics class should be sophisticated enough to represent a broad range of robot sensor and effector capabilities. The software API for an introductory programming class should be general enough for students to quickly use for their programming tasks.
- As to robotic hardware, we need to consider the budget, the course requirements, the extendability to include more sensors and effectors, and their supporting software. For example, although ActivMedia Amigobots or Sony AIBOs have more sensing or computational power than smaller robots, they typically cost a lot more. The K-Team robots have smaller size, cost less, and have the potential to include more sensing capabilities as needed. It serves well in a department where experimental spaces are limited.
- Another hidden factor is whether the hardware or software platform can also serve research purposes. With a limited budget or space, how can a department maintain a small scale robotics lab that serve the purpose for both research and teaching?
- Simulation vs. physical robots. There are many tradeoffs when selecting the appropriate programming platform for learning purposes. Simulation enables a fast learning cycle (programming, debugging, testing) by assuming that robots work in an ideal environment. Physical robot applications enable more student interaction with physical entities which help them understand more about real world experiments and deal with issues such as dynamics of the environment, inaccuracy of sensor readings, etc. However, the maintenance requirement of such a robotics lab for teaching purposes is high and students must commit certain amount of time to work with the robots in the lab.

Course outcome evaluation

Course outcome evaluation is important for recognizing the benefits, identifying the deficiencies, and improving the course structure. There are still many concerns (Fagin & Klassner 2006) of whether the students' learning outcome outperforms the effort that is spent to include robots into traditional classroom. Through the evaluation, we should be able to assess students' attitudes towards using robots in class, whether it is a source of motivation, or it does not improve their learning process. Typical evaluation includes assessing students' homework, projects, and exams. We can

also get feedback from students through the use of questionnaires. These outcomes can be compared with the outcomes in the previous classes to see the differences. For the introductory classes, we can observe the percentage of students (freshman or sophomore) stay in the computer science program. For robotics or AI classes, we can also observe the number of students who are willing to pursue more advanced education or experience in robotics and AI.

Cal Poly Pomona characteristics

Cal Poly Pomona's hallmark is its learn-by-doing philosophy, thus it strongly supports faculty members to provide the opportunities for students to apply their knowledge on hands-on projects. However, we typically have a class size of 35 students and are based on a relatively short quarter system (11 weeks). These do bring some challenges because of the limited resource and budget. We may need to limit the class enrollment to a certain size, such that students can group together (3 maximum) to work collaboratively on the available resources. Because of the short quarter system, the robotics course should be divided into two levels: the introductory level which examines the basics of programming robots, and the advanced level which examines subjects such as planning, reasoning, learning and multi-robot systems. Students also have different programming and OS background, which may limit the types of simulation or robotic platform. Additionally, Cal Poly Pomona has a very dynamic community, many students work part-time while attending school. Thus it is often preferable that students could work from home for their class projects. This produces a great challenge for using physical robots in class. In the following section, I present a course proposal of an introductory robotics class that will be offered in 2007 at Cal Poly Pomona.

An introductory robotics course proposal

The introductory robotics course will focus on learning the basic building blocks for the development of intelligent robotics, which will examine a variety of algorithms for the control of autonomous mobile robots, exploring issues such as robot control paradigms, robotic behaviors, sensing, navigation, planning, and localization. The book "Introduction to AI Robotics" by Robin Murphy will be used and we will also reference some research papers or notes in robotics area as needed. The course will be project-oriented. Students will have the opportunity to work independently and with other teammates. For the first half of the quarter, three individual projects will be designed to help students be familiar with the robotic simulation platform, be able to program the building blocks, such as robotic behaviors, logical sensors, etc. For example, students will be asked to program the behaviors of a robot such that it can navigate from a certain starting position to a goal position and avoid any obstacle on the way; or program a logic sensor suite such that the robot can use both laser and sonar sensors to detect the nearest obstacle. Then, students will form groups, and select or propose an interesting project that is sophisticated enough for them to work on for the rest of the quarter. For example, map building, exploration, treasure hunt and tag game

could serve the purpose of a term project. In the end, each team will analyze their results and present their work either through formal presentation or poster presentation.

At Cal Poly Pomona, we are in the process of building an Intelligent Robotics Laboratory, which will not be ready by the time we offer the class. Additionally, this is the first time that we offer such a robotics class here, thus we prefer to start small. Player/Stage will serve as the simulation platform in this class. Although most students are not familiar with Unix/Linux systems, they should be able to possess basic working knowledge of the system in a short time. Player/Stage is selected for the simulation since it is a free software that supports a lot of robotic platforms and has been actively supported by many researchers. Linux (such as Knoppix) and Player/Stage can run completely from CD, thus save the students' effort in installing such a system at their home computers. With the simulation, students can choose to work at lab or work from home depending on their schedules.

In addition to the standard evaluation questions such as course organization, fairness in grading, etc, the course will also be evaluated in terms of: (1) stimulation of student interest with the robotic simulation, (2) whether the programming projects have facilitated their learning of robotics, (3) whether the use of physical robots would stimulate more interests, (4) whether they would take more advanced robotics or AI classes, and (5) whether they would recommend this class to other students.

Summary and Future Work

This paper has discussed three different ways to incorporate robotics into the undergraduate curriculum. The design of a course, such as its hardware and software platform, its balance of theories and programming practices, and its format, is determined by many factors, such as the contents, the budget, the time, the space, and the number of students. In the end, we should effectively evaluate the course results and improve the techniques as needed.

This paper presents some basic ideas to include robotics in an undergraduate program where robotics has not been introduced before. My future work includes the development of a robotics class for junior and senior students; expending the AI course here at Cal Poly Pomona to include robotics; and eventually bringing robots to the introductory courses of computer science.

References

- Blank, D.; Kumar, D.; Meeden, L.; and Yanco, H. 2006. The Pyro toolkit for AI and robotics. *AI magazine* 27(1).
- Challinger, J. 2005. Efficient use of robots in the undergraduate curriculum. In *Proceedings of SIGCSE Technical Symposium on Computer Science Education*, 436–440.
- Crabbe, F. 2006. Unifying undergraduate artificial intelligence robotics: Layers of abstraction over two channels. *AI magazine* 27(1).
- Dodds, Z.; Greenwald, L.; Howard, A.; Tejada, S.; and Weinberg, J. 2006. Components, curriculum, and commu-

nity: Robots and robotics in ungraduate AI education. *AI magazine* 27(1).

Fagin, M. M. M. G. B., and Klassner, F. 2006. Do LEGO MindStorms robots have a future in CS education? In *Proceedings of SIGCSE Technical Symposium on Computer Science Education*.

Gerkey, B.; Vaughan, R.; and Howard, A. 2003. The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the 11th International Conference on Advanced Robotics*, 317–323.

Greenwald, L.; Artz, D.; Mehta, Y.; and Shirmohammadi, B. 2006. Using educational robotics to motivate complete AI solutions. *AI magazine* 27(1).

Klassner, F. 2006. Launching into AI's October Sky with robotics and Lisp. *AI magazine* 27(1).

Koenig, N., and Howard, A. 2004. Design and use paradigms for Gazebo, an open-source multi-robot simulator. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Veloso, M.; Rybski, P.; Lenser, S.; Chernova, S.; and Vail, D. 2006. CMRoboBits: Creating an intelligent AIBO robot. *AI magazine* 27(1).

ven Lent, C. E. 2004. Using robot platforms to enhance concept learning in introductory CS courses. In *Proceedings of the AAAI Spring Symposium on Robotics in Education*.

Xu, L., and Martin, F. G. 2006. Chrip on chirkets: teaching compilers using an embedded robot controller. In *Proceedings of SIGCSE Technical Symposium on Computer Science Education*.