Robotics in Early Undergraduate Education

David L. Duke, Justin Carlson and Chuck Thorpe
Carnegie Mellon University, Qatar Campus
{dduke|justinca|thorpe}@qatar.cmu.edu

Abstract

Introduction to Mobile Robot Programming is a new project-based course taught at Carnegie Mellon University in Qatar to undergraduates early in the curriculum. We describe the course details and relate our experiences and observations. We believe that this course is a valuable introductory computer science course; however, its success requires a significant time commitment from the instructors. The value lies in this course's ability to add breadth and computational thinking skills that will aid the students as they progress.

Introduction

At the new Qatar branch campus of Carnegie Mellon University, a new robotics based course is being offered which is targeted at undergraduates early in their course of study. This course is designed as a departure point for building breadth and computational thinking skills at a very early point in the curriculum. Mobile robotics provides a rich source of opportunities to apply creative thinking and analytical skills to tangible problems. Additionally, relatively simple platforms and low level APIs keep barriers to entry low, making it possible to expose students to nontrivial problems very early in their undergraduate careers.

In this paper, we will share our experiences in the design and teaching of this course through its first two iterations. We will outline the structure of the course, describe the hardware and software systems used and developed, give an overview of the curriculum and logistics, and outline our observations throughout the process.

Background

In 1999, the Qatar Foundation launched an initiative to bring high-quality, western-style postsecondary education to the Arabian peninsula. The result, Education City, houses branch campuses of several American universities serving a growing undergraduate population drawn primarily from the region.

Carnegie Mellon began providing undergraduate education programs in computer science and business at Education City in the fall of 2004. Carnegie Mellon University in Qatar (CMU-Q) currently has an undergraduate population of approximately 120, with a longer term plan to grow to approximately 400 students. The startup of a new program provides an intimate, fast-paced environment which is conducive to innovative ideas in teaching and curriculum.

Figure 1: Robot Platform

Course Context

Introduction to Mobile Robot Programming is offered to second-semester freshmen. The requirements for enrollment are minimal: one semester of programming as a prerequisite, and another as a corequisite. In addition to the majority of computer science students, the course is taken by a substantial proportion of business students. The course is not a robotics course in the traditional sense; instead, it is a computer science course which uses robotics as a platform for a project-based approach to early-stage computer science education. The curriculum was initially designed by adapting an existing course pioneered by Illah Nourbaksh (Lalonde, Bartley, & Nourbakhsh 2006).

One of the major goals of the computer science curriculum at CMU is the development of the skills needed to apply computational power to a wide variety of problems. This course helps students towards that goal in a number of ways. Teamwork, abstraction, complexity management, design, debugging, testing, and large project management are all required to succeed. The project nature of the course means that at the end of the semester students have built all the significant parts of a large software system which can

play a nontrivial competitive game in the real world.

Hardware

The robotic platform used is the AmigoBot from ActivMedia. This is a low-profile, two-wheeled differential drive robot, with a ring of 8 sonars to sense its environment. Integrated odometry is provided via encoders on the drive wheels. An example of the platform can be seen in Figure 1.

A standard Linux-based laptop is connected to the robot via an RS232 serial link, and rides atop the chassis on a custom metal platform. The computational burden of the tasks covered in the semester requires a very modest amount of processing power. While the hardware used is not as inexpensive as some alternatives (Wolz 2001), the cost per laptop/robot combination is not prohibitive.

The robot used has some significant limitations. The most significant is the coarse granularity of commands for setting the velocities of the wheels independently. This is not a hardware limitation; rather, it is a protocol limitation which could be addressed with a firmware update. However, we have been unsuccessful in persuading the vendor to address this issue.

Additionally, sonars have a relatively slow update rate and the firing order cannot be easily configured. The control loop over the serial link runs at 10 Hz. In each iteration of this loop, 2 sonars are updated, meaning the entire sonar ring update takes .4 seconds.

The possibility of using vision as a primary sensing modality for this course was considered, but eventually was explicitly rejected. Although it is possible to present high-level vision APIs, the desire to have students implement as much of the systems as possible guided us to simpler sensing solutions.

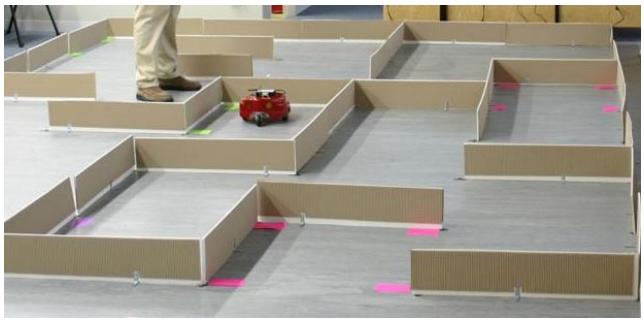

Figure 2: Robot solving a maze

Software

Currently, introductory programming classes at CMU-Q are taught using Java and the Eclipse IDE; the same platforms are used in this course, albeit on top of Linux instead of Windows.

Students communicate with the robot via a simplified API which allows them to query the state of the sonars and encoders and give linear or angular velocity commands to the robot at 10 Hz. The most significant calls to the API are listed in figure 4.

The run-time interface presented to the students provides them with visual feedback about the current state of the robot, and allows them to run their code by selecting assignments from a menu.

Lab assignments are typically implemented by adding a method with a specified prototype to a given file. In the instances where input is required of the user, the input and parsing mechanisms are given and the sanitized input is given to the student's code. This is an intentional design decision; although we would prefer to give the students the flexibility to implement a user interface how they see fit, the substantial majority have not seen any significant GUI programming before.

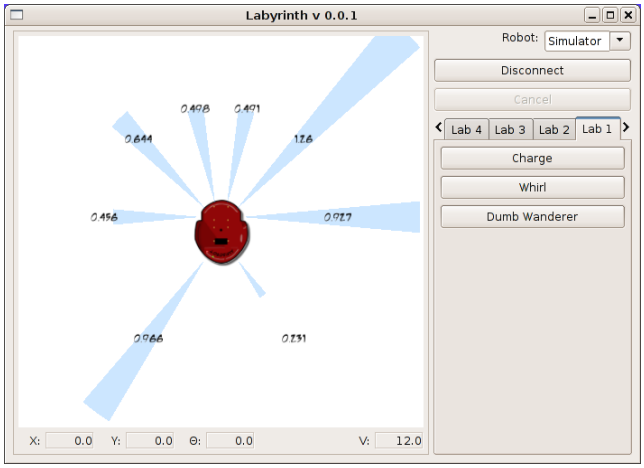

Figure 3: Software Frontend

For prototyping and experimentation, students are also provided with a simulated robot. This simulator is presented as a swappable backend to the API; students can run their programs on the simulator with no modification, though one of the constant surprises to an undergraduate is that the results of their programs can be significantly different on the physical robot than in the idealized simulator. Almost all lab work requires demonstration on a physical robot, forcing students to cope with the challenges not modeled in simulation.

Curriculum

The emphasis of the course is on laboratory programming assignments at one-week intervals. Each week the students demonstrate and explain their work to instructors for evaluation and advice.

The first half of the semester is fairly nonlinear in construction; each assignment is relatively independent of previous work. Topics covered include robot characterization, reactive control, open- and closed-loop control, precision movement, and learning via operant conditioning. Throughout this portion of the course, the environment in which the robot operates is left relatively unstructured; the ground is guaranteed to be flat, but obstacles are multiple shapes and sizes.

```
// Return the sonar and  odometry state
public AmigoBotState getState();

// Set the velocity of the wheels, in m/sec
public void setVel2(float lVel, float rVel);

// Rotate in place at rvel degrees/sec
public void setRVel(float rvel);

// Move in a straight line at vel m/sec
public void setVel(float vel);
```

Figure 4: Selected functions from the software API

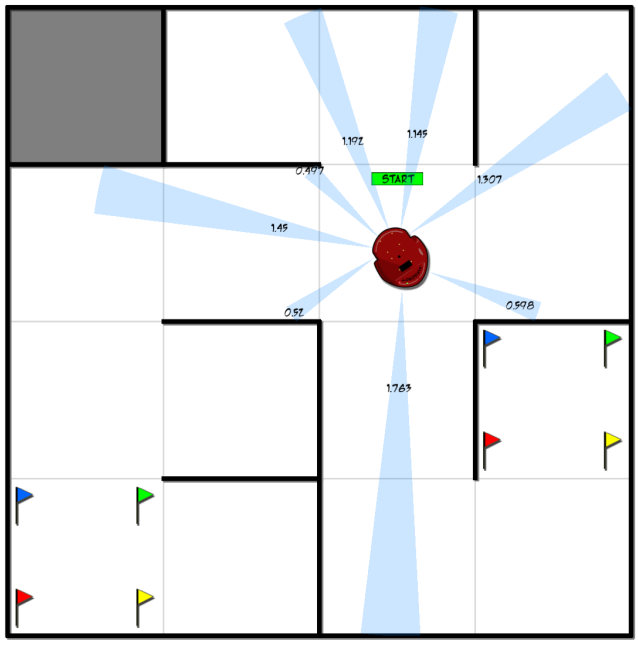

Figure 5: Simulator in a maze environment

The second half of the semester is more progressive, building towards the final tournament. At this point in the semester the robots operate in a more structured rectangular maze environment. All walls of the maze are orthogonal and of a fixed known length. We begin by having the students build primitives to map precision movement into movement commands useful for sensing and navigating a maze and bounding error accumulation. Using these primitives, students solve carefully constructed mazes using hand-generated universal plans, and are introduced to the idea of handling a case wherein the robot does not know where it begins. The next step is sequential planning, in which the robot must generate and execute its own plan for solving an arbitrary maze given a known starting point, goal and maze. Finally, information is hidden, and mazes must be solved both in the case where the starting position of the robot is unknown as well as in the case where the maze is not known.

The capstone of the class is the final challenge, in which robots race to gather goals in a networked interactive environment. It is designed to be achievable yet is complex enough that strategies for improving performance can be (and generally are) extremely diverse.

The challenge is a competitive game, with two teams playing at a time in identical (but separate) mazes. When the game begins, each team receives the outline of an empty maze, their starting position relative to that outline, and a set of shared goal locations. The number of goals is constant, though their locations change as teams claim them. When a team claims a goal it scores a point, and the goal is moved elsewhere in the maze for both teams.

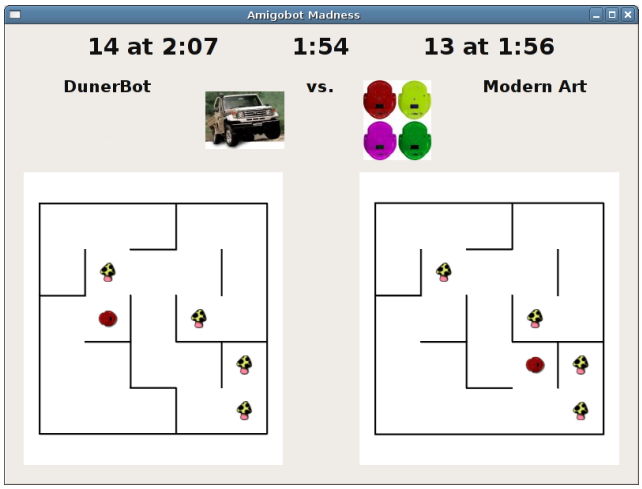

Figure 6: Head-to-head final competition

Logistics

Introduction to Mobile Robot Programming is taught with one instructor and two TAs. The most recent class size was 26 students divided into 13 groups of two. Each team had access to one laptop, and all teams shared time on 8 robots.

The class operates on a weekly cycle. At the start of the week, students are given a detailed laboratory assignment listing the skills their robot must demonstrate. When possible, the assignment includes a performance-based grading rubric to help students evaluate their progress.

Lectures are sparse, covering the basic material necessary for the completion of the lab as well as related topics. We often discuss different approaches to the problems presented, but generally do not give detailed guidance. Many of the projects include a competition component to help motivate the students. The weekly cycle ends with evaluations of the implementations. Significantly, all students can observe demonstrations for all groups.

Students work in groups of two to complete the labs, and after each week the students change partners. Each group is required to submit all underlying code when they demonstrate their implementation. The code is then placed into a code repository which is opened after all groups demonstrate the lab. Groups are allowed to access and build on any code from previous weeks placed in the code repository. Late in

the semester, students are allowed to form their own teams which are maintained through the last few lab assignments and through the final challenge.

Although the emphasis of the course is on the weekly laboratory projects, there are also homework assignments, exams, and some writing requirements which help to evaluate students progress outside of the team context.

Observations

This is a relatively new course, but we can offer some qualitative observations.

One of our key observations is that robotics is a fertile field for generating interesting computational problems which are interesting but do not require extensive computer science backgrounds to approach. All of the lab assignments can be solved using standard Java code with extremely simple data structures, yet students found most of the assignments quite challenging. From the comments of our students, we believe that seeing physical results from writing programs can be a powerful motivator.

Competition is good. Students are more motivated by assignments which have some aspect of intraclass competition, even when this is unrelated to their grade. In many cases competitions have motivated students to implement solutions that far exceed the requirements. In particular, top students are inspired to push themselves by friendly competition with the instructors, especially in the final challenge of the semester when their work is on display to the wider university community.

For many of our students this was their first experience working in a team. Keeping teams fluid for much of the semester improved the students' teamwork skills by letting them work with a variety of personalities and coding styles. As with any course involving teamwork, there are some problems with nonparticipation and group conflict which need appropriate handling. In particular, at the freshman level sometimes students need more guidance on how to effectively work in less-than-ideal team situations.

In the course from which this offering is adapted, teams of three students are formed at the beginning of the semester and are kept for the duration. In the first iteration of this course, the same approach was taken. The varied skill levels of team members combined with the expected group conflicts exacerbated problems of nonparticipation in many cases, causing us to move to the current system.

Making the code repository available is a valuable method for teaching the students how to read, understand, and debug code, and to find examples of design patterns which work well. It also helps to keep students from falling irreparably behind as lab assignments begin to build on previous work. Students often downloaded code written by a different group that needed to be debugged. As instructors we intentionally gave limited help, leaving the responsibility of understanding the code to the students.

One lesson that is constantly reinforced in the lab assignments is analysis. Particularly when debugging, students are forced to come up with strategies for making behavior triggers visible and breaking down the problem into smaller parts.

Finally, to be effective, teaching this course requires a significant amount of time invested in interactions between instructors and small groups. The nature of the course is such that individualized instruction and help is indispensable.

Conclusion

We have presented a new robotics course offered to undergraduates early in their coursework. This course leverages robotics as a platform to teach analytical skills, and gives students early exposure to teamwork and large-scale projects, and culminates in students building all the major components to solve a nontrivial task on real hardware.

The course is well received by students, most of whom find interactions with robots and competition with their peers to be good motivators to explore and invent. We look forward to continuing to refine the presented course and to exploring more options for using robotics in the curriculum.

References

Danyluk, A. P. 2005. Using robotics to motivate learning in an ai course for non-majors. In *In Accessible Hands-on Artificial Intelligence and Robotics Education (AAAI Spring Symposium)*.

Dodds, Z.; Greenwald, L.; Howard, A.; Tejada, S.; and Weinberg, J. 2006. Components, curriculum, and community: Robots and robotics in undergraduate ai education. *AI Magazine* 27(1):11–22.

Lalonde, J.-F.; Bartley, C.; and Nourbakhsh, I. 2006. Mobile robot programming in education. In *International Conference on Robotics and Automation (ICRA)*.

Maxwell, B. A., and Meeden, L. A. 2000. Integrating robotics research with undergraduate education. *IEEE Intelligent Systems* 15(6):22–27.

Rosenblatt, M., and Choset, H. 2000. Designing and implementing hands-on robotics labs. *IEEE Intelligent Systems* 15(6):32 – 39.

Wolz, U. 2001. Teaching design and project management with lego rcx robots. In *SIGCSE '01: Proceedings of the thirty-second SIGCSE technical symposium on Computer Science Education*, 95–99. New York, NY, USA: ACM Press.