An Intensive Introductory Robotics Course Without Prerequisites

Julian Mason and Gavin Taylor
Duke University

Abstract

We introduce an introductory robotics course with no prerequisites which makes use of a novel programming package to explore modern robotic concepts without significant initial investment in teaching programming.

Introduction We present the details of a first course in robotics, targeted to the 11- to 16-year-old age group, and offered as part of a summer program for academically gifted students. The course features a fundamental emphasis on project-based and group learning, and assumes no student background in robotics or computer programming. For this reason, it focuses on solving challenging problems using simple robotic hardware and a novel high-level robot-control library called FLAIL. Course topics cover a variety of topics, and students are given the opportunity to explore a topic of personal interest during a final project. Despite their limited backgrounds, these final projects demonstrated substantial understanding and achieved success in robotics topics ranging from object recognition to game-playing algorithms.

Course Format This course was offered as part of the Duke Talent Identification Program (TIP), a nationally-recognized summer program for highly-gifted 7th through 10th grade students. The course is a three-week residential program, and the students are in class with the instructors for thirty-three hours a week. The students entered with a wide variety of backgrounds, although none had robotics experience. The vast majority also had no programming experience, and many students had not covered mathematics beyond algebra. Course offerings have averaged around twenty students.

The course also had a very limited budget of $86 per student. This is particularly restrictive given the need for robots and computers to control them.

The long blocks of uninterrupted class time, minimal student experience, and strict hardware limitations provided the framework on which we hung our course design.

The explicit goals of our course were to teach students the basics of robotics, and expose them to a large number of robotics topics despite the students' lack of background. The implicit goals were closely related to the age and intelligence of the students; we hoped to create an educational setting that was both challenging and social, pushing students to be engaged and develop the teamwork skills often lacking in young students accustomed to intellectual superiority.

To address these goals, we devoted as much of the course as possible to difficult projects performed in groups of two or three which would be appropriate for an undergraduate robotics course. The projects were made hard enough that no student could complete them without help from other students and their instructors, easing the fears of students unfamiliar with academic confusion. The projects were tied together with a stated theme of steadily increasing robot autonomy, which coincided nicely with increasing difficulty.

Hardware Any robotics course needs three hardware components: sensors, effectors, and a computational platform to control them. We used the iRobot Create, the Logitech QuickCam Pro 4000 USB camera, and assorted laptops. The reliability and battery life of our hardware proved extremely impressive. This course has now been taught six times (twice by an instructor not involved in the development of the course), for a total of more than 650 hours of class time, and not one robot, camera, or laptop has failed. These components were chosen for their low price and availability.

The use of generic webcams and laptops came as a surprise to many students; they had clearly not realized that such standard parts could be used in robotics applications. Students were also pleasantly surprised to learn that the Create is widely used in academic research. We believe that this helped make it clear that, while a simple robotics platform, the Create is also a serious robotics platform, not a "toy robot."

Software (FLAIL) Because this course is about robotics, not programming, we elected to teach programming concepts on an as-needed basis. For example, conditional statements were introduced early, to support keyboard control of a robot, while objects are the last programming concept introduced, to support game trees. In order to spend as little class time as possible on programming, it is nec-

essary to use a programming environment that is as simple as possible. To this end, we selected the Python programming language, and wrote our own software library, which we call FLAIL (for **FL**yweight **A**ctuator **I**nterface **L**ibrary). FLAIL aims only to hide the technical details of robot control and camera interaction; reasoning and software design are left entirely to the programmer. FLAIL consists of six components, `FLAILBot`, which provides robot control, `FLAILImage`, which provides an interface to a three-color-channel bitmap image, `FLAILCam`, which captures images from our camera, `FLAILKeys`, which provides unbuffered keyboard input, and `FLAILSimulator` and `SimulatedFLAILBot`, which together provide a two-dimensional graphical robot simulator.

Topics Covered Topics were covered roughly in order of increasing robot autonomy. Students were first introduced to FLAIL and robot control by creating a program for manual control to compete in a robot soccer game. We then introduced sensors, namely the Create's odometer and bump sensor. Students learned about the difficulties of sensors through dead-reckoning and wall-following projects.

We then introduced cameras. We painted cinder blocks with highly-saturated colors, and students programmed their robots to segment images based on color in Hue-Saturation-Value (HSV) space, and respond to this information by ramming bricks of a given color. Students then were given the color maze project, in which cinder blocks made up a "maze" with different rooms, each of which is painted a different color. Students then autonomously navigate from one end of the maze to the other.

Up to this point, student projects had focused on simple controllers operating in noisy but static environments. To continue our theme of increasing autonomy, we invert this, and focus on environments that are noise-free but highly dynamic and adversarial. Specifically, we introduce the perfect-information zero-sum games of Tic-Tac-Toe and Connect Four. Students were given partial implementations of these games, and completed the implementation and programmed a player based on minimax search. Tic-Tac-Toe has a very shallow game tree, so fast, unbeatable players were possible to implement. Connect Four's game tree is too deep for exhaustive search, requiring the introduction of a utility function and alpha-beta pruning. Student groups designed and implemented their own utility function, and we finished the project with a Connect Four tournament. Although the game-playing projects took place off the robots, the students needed no prompting to see the possible robotics applications.

Student Projects Final projects required students to extend a topic from the course of interest to them, and then formally present the results to the class and field questions. Due to the breadth of the course, student projects took on a range of topics, a few of which we detail here.

A common project was some variation on autonomous robot soccer. The location of the camera varied, from mounted on the robot to hanging from the ceiling for an overhead view, but in all cases, the camera was used to identify the robot, the ball, and the goal, and autonomously plan and perform a series of moves that would result in the scoring of the goal. Also in the area of robotic planning, a group used the simulator to build a maze, and constructed a graph between random vertices on the map, where edges corresponded to legal moves. Path planning could then be performed by calculating the shortest path on the graph. Other groups used the simulator to perform Monte Carlo localization.

Students interested in cameras explored a range of computer vision projects, from template-based object recognition to rangefinding using stereo vision or a laser pointer mounted at an angle and a camera to detect the laser dot. Other students continued on game theory through exploring game playing through case based reasoning or methods of ordering game tree exploration to increase the likelihood of pruning.

Conclusion This course met our goals of student engagement and growth using difficult, in-depth projects and FLAIL, a novel software package that minimized the difficulties of robot control for new programmers. It is important to consider the possibility of moving the course to other formats. By removing the later parts of the course (particularly the programming-intensive game theory projects), the course could be moved to a younger audience. FLAIL is well suited for the uninitiated; if the students are capable of understanding the very basics of programming, it is very easy to elicit complex behavior from a robot. Conversely, by presenting course topics more formally and in more detail, the course could be adapted to a lecture-and-lab format for undergraduate students. This course explored robotics by gradually increasing robot autonomy; if this were continued, more robotics topics would become relevant and interesting. As topics are introduced more formally, the course moves closer to a traditional "CS1 with robots."

Students continually demonstrated a great interest in the course, oftentimes preferring class to free time and rarely expressing boredom despite the amount of class each day. We believe the course's robot-first approach and FLAIL can be used as a blueprint for a introductory robotics course for a range of student demographics, from early middle school students to undergraduates.