Roomba Pac-Man: Teaching Autonomous Robotics through Embodied Gaming

Brendan Dickinson, Chad Jenkins Advising
Department of Computer Science
Brown University
115 Waterman St.
Providence, RI, 02912-1910
bcd@cs.brown.edu

Abstract

We present an approach to teaching autonomous robotics to upper-level undergraduates through the medium of embodied games. As part of a developing course at Brown University, we have created the Roomba Pac-Man task to introduce students to different approaches to autonomous robot control in the context of a specific task. Roomba Pac-Man has been developed using commodity hardware from which students explore standard methods in robotics, namely subsumption, localization, and path planning. Our development of Roomba Pac-Man is founded upon grounding robotics in a compelling and accessible application in a non-contrived real-world environment in a manner than can be reproduced, giving students a sense of ownership.

Introduction

As the field of robotics advances, robotics education must adapt to incorporate both technical developments that become core topics and compelling new challenges of societal-level interest. Undergraduate autonomous robotics curricula have been adept at exposing students to relatively modern topics, such as behavior-based control (Arkin 1998) and Monte Carlo Localization (Thrun, Burgard, & Fox 2005). However, the impact of such coursework can be difficult to conceptually translate beyond the academic setting. While Lego Mindstorms and Pyro are excellent simplified platforms for teaching, the artifacts created with them are not directly applicable to real world applications. Such as setup makes the teaching of robotics easier, it does not ground robotics in the real-world for the students. On the other end of the spectrum are platforms such as Pioneer robots, which can be prohibitively expensive and are distant from deployment in society. We believe that Roombas may fit into a "sweet-spot" between educational platforms (Lego Mindstorms, Pyro) and highly sophisticated and expensive platforms (Pioneers). Our approach is to explore different approaches to autonomous control with focus on a specific task that is compelling, reproducible with inexpensive off-the-shelf hardware, and deployable in many environments. To this end, we have developed the *Roomba Pac-Man* task (RPM) as a central theme in the Brown course CS148 ("Building Intelligent Robots") for exploring topics in reactive and deliberative robot control. Working from the appeal of video games, RPM is an embodied version of the classic 1980s arcade game *Pac-Man*. In RPM, a virtual Pac-Man is replaced with a physically embodied iRobot Roomba vacuum equipped with a webcam and on-board computing. For two of the four projects in the course, the fifth floor of Brown's Computer Science building becomes a Pac-Man world complete with fiducialized versions of: "food pellets," "ghosts," and "power ups." Demos of these projects consist of student's robotic clients competing for high scores by vacuuming pellets and visiting power-ups within a fixed amount of time. The other two projects focus on developing understanding of methods to improve RPM performance (localization and path-planning). The first three projects cover subsumption (Arkin 1998), localization (Thrun, Burgard, & Fox 2005), and path planning (Choset *et al.* 2005) in the context of RPM, allowing for normalized comparison and appreciation for the relative strengths of the approaches. The final project is culmination of what has been learned in class, and offers the opportunity for innovative design.

RPM leverages Roombas as a cost-effective and deployable solution for teaching Robotics on real robots. Following the desire for reproducibility, students use the Player robot server and Gazebo simulation platform (collectively referred to as PSG) to develop control clients.

Roombas are used as the platform for the course because they offer the student a chance to interact with a robot that is actually used in the world. The Roombas provide a platform that is easy to interact with (no proprietary software/hardware). Via the PSG open-source project a wide variety tools may be used to acquire sensory information, and if something is not currently supported it can be reasonably added; this allows the student to attach virtually any real world sensor to the Roomba. In short, the Roombas are real robots, used in the real world, not some stripped down educational robot, and certainly not a toy.

To summarize, the Roomba Pac-Man task offers a compelling approach to teaching robotics because it is:

- applicable to the real world, allowing for straightforward manipulation (vacuuming).
- easily reproducible.

- contains a compelling task that helps to motivate the students beyond their own academic interest.
- reinforces various approaches to mobile robotics covered in class.

In the following sections present our ongoing work developing robotics curriculum around Roomba Pac-Man. The extensions to PSG to support RPM and the progression of lab exercises and projects throughout the course are presented.

Course Structure

Brown's CS 148 is geared toward undergraduates drawn mainly from CS, but also Engineering and Cognitive Sciences. All students are expected to have a minimum of two semesters of programming experience. While there are no required texts, recommended readings are drawn from (Thrun, Burgard, & Fox 2005), (Martin 2001), (Choset *et al.* 2005), and (Arkin 1998). The structure of Brown CS148 consists of three introductory labs, three control projects, and a final project all of which reinforce material presented in lecture. Labs are simple projects designed to give students an introduction to PSG and the Roomba hardware. The labs progress through basic reactive obstacle avoidance, blobfinding, object/fiducial seeking, and color calibration with a physically simulated robot. These labs are designed to be straightforward exercises (implementable within a given lab period) that give the students a chance to familiarize themselves with robotics.

The course projects are designed to explore reactive and deliberative approaches to robot control in the context of RPM. The first project, implementing a reactive subsumption controller to compete in Roomba Pac-Man, integrates all the topics covered previously in the labs. The next two projects focus on localization and its use for deliberative path planning. For the second project, students implement Monte-Carlo Localization (MCL) for a simulated Pioneer 2AT in PSG. The third project involves writing a path planner to play RPM, again in simulation, using the estimates from their MCL system from the second project. For final projects, undergrads develop robot clients using their MCL and planning systems developed in previous projects to compete in a final competition.

The projects involve two main deliverables, in-class competition/demonstration of the project and an electronic submission of the work (project write-up, source code, and other materials). The in-class competition involve a demonstration of the students' robot controllers and should show understanding of the specific aspects of the project (statelessness in the subsumption project, localization estimates for MCL for example). In the project write-ups students are asked to scientifically present the results of their implementations. Specifically, project write-ups should address the design choices relevant to individual projects and various strengths and weaknesses.

RPM Platform

We aimed for RPM to be cheap, reproducible, and usable in normal environments. For this purpose, the iRobot Roomba was a logical choice, being a cost effective solution with brand familiarity. Because of its popularity in society, students immediately see it not as a toy, but a device with real-world applicability. Each Roomba costs \$150. Standard Dell Dimension laptops, which cost \$500 each, control an individual Roomba. The Roombas are connected to the laptops via a Robo Dynamics Roo Stick, which can be purchased for \$25 each. PSG has basic support for the Roomba Serial Command Interface, which was extended to incorporate more of the Roomba's features (e.g., IR, vacuum, etc.). Finally, Logitech Communicate STX webcams were mounted on the Roombas, costing about \$30 each. Thus, for under \$700 one can procure, a functional, real world robot with the ability to manipulate objects (e.g., vacuum). As a result of the low cost of the set-up, there were nine Roombas available to students. An early version of RPM is shown in Figure 1.

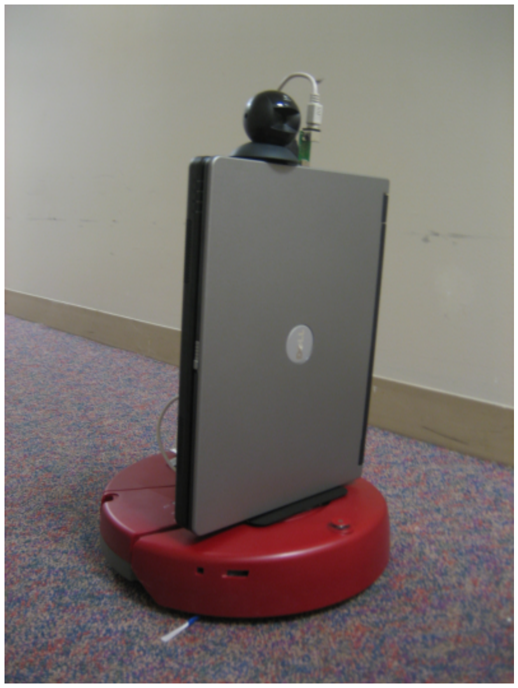
Figure 1: An early version of Roomba Pac-Man.

While cheap, our infrastructure is far from optimal due to the size and weight of the laptop and the sketchy nature of webcams. The setup presented in figure 1 proved infeasible because of the dramatic change in the center of gravity of the Roomba due to weight of the laptops. The laptops could not lay flat, as they would have extended beyond the radius of the Roomba. The final, non-optimal, solution was to tether the Roombas to laptops that students carried. Other efforts have explored better options such as Gumstix embedded boards (Gerkey 2006) and MacMinis (Dodds 2006), but were prohibitively expensive for our purposes. Tethering the Roomba to the laptop via the Roo Sticks was chosen rather than a Bluetooth or wireless connection because at the time of procurement the Roo Sticks appeared to be cheaper and easier for the students to implement. In reality, the tethering was awkward, and Roo Sticks proved to be unreliable (often melting after prolonged use). Bluetooth and wireless options will be explored for the next iteration of RPM. These problems aside, the platform is portable and flexible to the specifics of the robot hardware. If one needs better computation, one can buy a more powerful computer. If one needs better or different sensors, one can attach a suitable device and use/write the Player interface.

PSG and its Modifications

Much of our framework relies on the PSG platform (Gerkey *et al.* 2001). Player is a network server for robot control. Player runs on-board a single robot and provides a clean interface to the robot's sensors and actuators over an IP network. Gazebo is a 3D physics-based robot simulator suitable for smaller numbers of robots simulated at high fidelity. The physics for Gazebo is provided by the Open Dynamics Engine (ODE), which integrates physical dynamics for arbitrary kinematic structures through optimization.

PSG provides an infrastructure for developing robot controllers. Students write controllers as client programs that send control commands to and request information from a robot through its Player server. Stage and Gazebo can simulate various types of robot platforms (i.e., hardware) and populations. The same interface, provided by the Player robot server, is used to control a robot in the real world or its equivalent in a Stage/Gazebo simulation. Robot platforms that are not currently supported in PSG can be developed through implementing appropriate Player server interfaces and devices in Stage or Gazebo.

Devices (e.g., a laser, a camera, or a complete robot) are actual hardware in the real world or simulated hardware that exists in a virtual environment maintained by Stage or Gazebo. A robot server (e.g., Player) is the information interface between the robot and any program that requests information from or sends commands to the robot. Regardless of whether a device is real or simulated, the robot server provides the same interface to the robot for client programs. Thus, controllers developed on a simulated device will immediately run the equivalent real robot device given PSG support for the device.

Another advantage of Player as a robot server is its independence from a particular client-development language. The interaction between Player and a client program is done completely over a TCP/IP (or UDP/IP) network connection. Thus, any language with libraries that supports Player functionalities can be used to develop robot clients. The most supported client language are C and C++. Many other languages are supported including Python, Java, and GNU Octave.

Several changes were made to the Player in the development of RPM. The original Player Roomba support allowed only for position control and reading the bump sensor. The ability to read the other sensors on the Roomba was added, this included: the six infrared sensors including the IR-wall detector and the various buttons on top of the Roomba. This ability was added by utilizing the Proxy structure supplied by Player, adding only the "glue" to map the commands into the proxy functions. The ability to control more of the Roomba outputs besides just the wheel motors was also added. This included the ability to control the color and brightness of the LEDs on the Roomba and to turn on and off the vacuum through the gripper proxy.

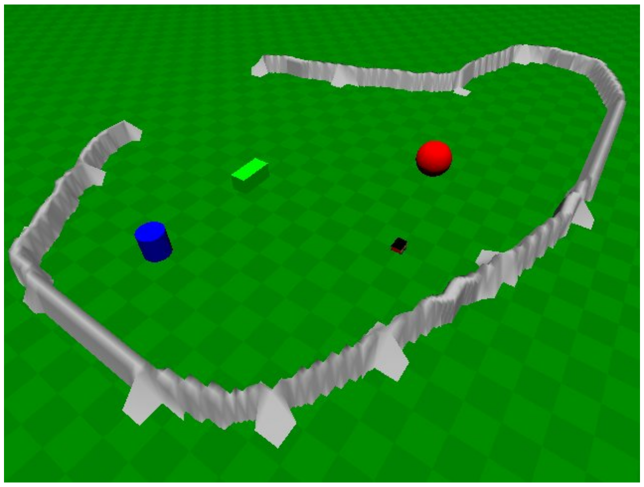

Figure 2: Simulated world in Gazebo for Labs 1 and 2

Several other modifications to Player were made in relation to the camera. The ability to auto-detect camera parameters was implemented to enable the webcams to function in Player. Modifications were also made to playercam, a PSG utility that streams the camera frames to the screen and overlays the blobfinder results in this image. This program was modified to report a range of YUV values when the user clicked and selected a rectangular region of interest in the image. This change was a tremendous help for camera calibration, which allows for more readily preparing Roombas to play in various lighting conditions. All of these changes have been submitted to the PSG developers, and all of the Roomba driver changes have been incorporated into the latest build of Player.

Labs and Projects

The work of the course consisted of three labs completed over the course of the first month of the semester. The four projects are larger undertakings, for which the students were given 3-4 weeks to work on.

Lab 1: Obstacle Avoidance

The first Lab is structured to acquaint students with the subtleties involved in using the PSG robot interface and simulation system. After a brief tutorial of libplayerc, the Player C client library, they are given the task of writing a reactive client for a simulated Pioneer 2AT to exit the enclosure shown in Figure 2. Students write wandering and obstacle avoidance routines using simulated SICK 2000 laser range finder.

Lab 2: Object Seeking

In Lab 2, students extend their obstacle avoidance client to perform an object seeking task. In this seeking task, the robot is to look for and drive to fiducials recognizable from blobfinding provided by CMVision (Bruce, Balch, & Veloso 2000). To accomplish this task, students use a simulated a Sony VID30 video camera. Given a world, the goal for this lab is to create a Player client containing a finite state machine that continually drives between two different fiducials (Figure 2). Students also experiment with positioning the light source to get a controlled sense of how lighting affects vision sensing.

Lab 3 and Project 1: Color Calibration and Reactive Roomba Pac-Man

Lab 3 serves a gateway into the first project, writing a subsumption client for the Roomba Pac-Man task. Lab 3 extends Lab 2's object seeking client to work with a physically embodied Roomba. Using the same Player proxies, the client controls a Roomba endowed with touch/bump, IR, and camera sensing. The camera sensing is accomplished by attaching a web-cam to the Roomba and having the client subscribe to the web-cam as a proxy. While the Lab 2 client could theoretically perform on the Roomba without modification, there are issues caused by the uncontrolled nature of the real world that must be addressed. Specifically, the blobfinder must be calibrated to recognize fiducial colors that vary under different lighting conditions, camera sensors, camera viewpoints, etc.

In Project 1, students compete for high scores in a game of Roomba Pac-Man on the fifth floor of Brown's CS department using a reactive subsumptive control policy. Lab 3 prepares students for this project by having them implement the following basic unprioritized functions:

- Fiducial attraction: same as in lab 2, except the sought cylindrical "Power up" fiducial will be composed of two colors, orange over green.(Figure 3(a))
- Fiducial avoidance: detect and avoid a green cylindrical "Ghost" fiducial by turning away from it. (Figure 3(b))
- Pellet consumption: detect and drive over a pile of orange colored "food pellets" on the floor. (Figure 3(c))
- Wander: wander around an environment to achieve "coverage."
- Wall avoidance: detect collisions with physical or virtual walls and move to avoid these contacts.

Project 2: Monte-Carlo Localization

Project 2 involves implementing Monte-Carlo Localization (MCL) with the goal of improving the performance of student's Project 1 by having a localization estimate. Students are given the fifth floor of Brown's Computer Science building as a Gazebo world file (Figure 4). This world file is a recreation of the world in which the students will compete in Deliberative Roomba Pac-Man in the final project. Fiducials of the same color are distributed throughout the world at known locations. Fiducials are used in the world so as to allow the students to write their MCL using a blobfinder. While a laser range finder may be more accurate and allow for a less contrived world, the goal of the project is to prepare the students for real-world implementations that will not have lasers. The fiducials have the same color in order to make it impossible to dead reckon off of a single fiducial forcing the students to maintain a probability distribution of hypothesises.

The goal for the student is to implement MCL on a simulated Pioneer 2AT using a blobfinder, bump/touch sensor, ir sensor, and odometry. The fact that the project is implemented in PSG makes it easier to deal with bugs and noise from the real world. For one, the lighting in PSG is constant and controllable. It is reasonable to ensure that the color of the fiducials only occur on fiducials. Furthermore, testing does not involve the set up of a lot of equipment, which means that bugs can be found and fixed expeditiously. Finally, the successful completion of project 2 allows student to concentrate their full attention to planning in project three.

Project 3: Path Planning for the Roomba Pac-Man

Project 3 uses the code developed in Projects 1 and 2 to create an effective deliberative robot control policy for the RPM task. The goal for this project is to create a control policy that uses a model of the world from a known map and state estimation to plan a path and execute it intelligently. While we initially planned to have the students implement this project in the real-world, students had a greater than anticipated amount of difficultly implementing vision-based MCL in Project 2. Thus we elected to allow the students to do Project 3 in simulation.

For this project, the students' clients deliberatively plan and navigate paths in order to for the robot to maximize the number of pellets eaten and power-ups visited while avoiding ghosts in a simulated environment. The student's clients used the pose estimates given by localization from Project 2 towards path planning. Students then must develop a planning algorithm to deliberatively control their Roomba. Students choose to implement a number of planning algorithms including: Dijkstra's algorithm (Cormen *et al.* 1990), wavefront planning, and potential fields (Thrun, Burgard, & Fox 2005). The algorithm, however, must take in to account the presence of ghosts and their random movements throughout the world.

Final Projects and Future Work: Making Robotics Relevant

CS148 concludes with final paper and project. The final project is an independently designed competitive RPM controller. Given what the students had learned over the course of the semester, students are tasked with developing the best controller possible for the Roomba Pac-Man task. Collectively, student clients are evaluated in a tournament-style RPM contest. The final paper is analysis of a fictional robot that discusses its technological feasibility (in terms of perception, decision making, motor control, and platform engineering) and possible means to develop innovations for realizing the robot. Additionally, the paper should try to answer

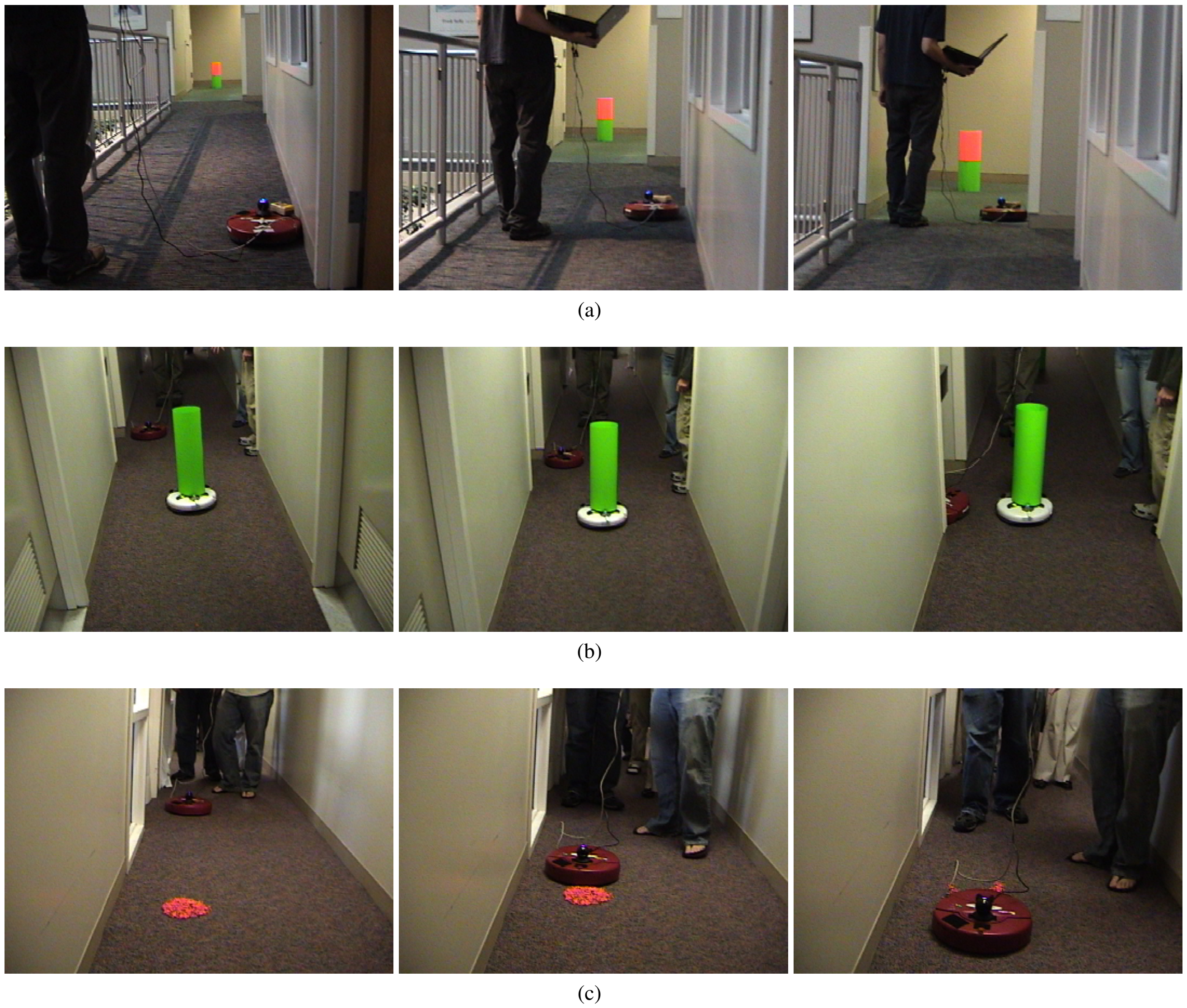

Figure 3: Examples of the robot driving to a fiducial (a), avoiding a ghost (b), and driving to food (c)

the following question: "What is the point of robotics?", specifically constructing an argument about most pertinent applications for robotics in society.

In future versions of CS148, we want to establish a stronger connection between human and robot decision making through embodied gaming. We are implementing an off-board, wireless tele-operation client that will allow a person to play RPM. Ideally, the tele-operator would observe only the perceptual features used by the robot (color blobs, IR, bump, and odometry). Such tighter human-robot interaction would help motivate the difficultly of developing robot control policies, provide students a baseline for their work, and make RPM even more fun.

Conclusion

On the whole, CS 148 was a success. In our observations and conversations with students they were motivated by the interesting task presented by RPM. Particularly successful were the first and third projects. In the first project a number of students wrote very effective reactive control policies for the RPM task. Students were clearly motivated by the embodied gaming aspect of the project. The open-ended nature of the third project, path-planning, resulted in the implementation of a number of different algorithms, which prompted interesting, well informed, class discussion about their relative strengths and weakness. However, there are a number of things we would change for future iterations of the course.

First, as mentioned earlier, we will explore options other than tethered connections between the Roomba and the laptop. This set-up was extremely awkward and made the Robots feel less "robot-like." Also, over the course of the semester 9 Roo-Sticks burned out, thus making then Roo-Sticks more expensive than initially thought. Another hardware issue encountered was the lackluster performance from

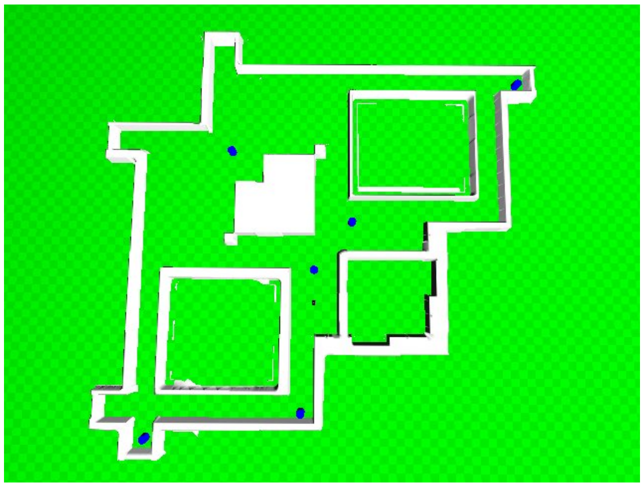

Figure 4: Map of the fifth floor of the Brown CS department

the webcams. The cameras automatically adjusted their white-balance, and, despite our best efforts, we were unable to disable this "functionality." The result was that when large amount of direct natural light was present the image from the camera would be very washed out.

Finally, robotics, in some ways, is more difficult for students than other disciplines of Computer Science, because, unlike other disciplines, the student must interact with the uncertainties of the real world. We found that some students became frustrated with the RPM task due to the fact that the sensors were not perfect.

We are currently debating whether the task as currently designed may be too difficult for an introductory robotics course given the fidelity of the sensors. While we found that some students became frustrated and struggled with the imperfect sensors, the success the course staff had implementing the projects within a much more limited time frame suggests that RPM should be within student's capabilities. The true problem may come down to properly motivating the students and preparing them for the noise inherent in real world sensors.

References

Arkin, R. C. 1998. *Behavior-Based Robotics*. Cambridge, Massachusetts, USA: MIT Press.

Bruce, J.; Balch, T.; and Veloso, M. 2000. Fast and inexpensive color image segmentation for interactive robots. In *In Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3.

Choset, H.; Lynch, K. M.; Hutchinson, S.; Kantor, G.; Burgard, W.; Kavraki, L. E.; and Thrun, S. 2005. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press.

Cormen, T.; Leiserson, C.; Rivest, R.; and Stein, C. 1990. *Introduction to Algorithms*. MIT Press.

Dodds, Z. 2006. Informal conversation with Z. Dodds of Harvey Mudd College.

Gerkey, B.; Vaughan, R.; Stoy, K.; Howard, A.; Sukhatme, G.; and Mataric, M. 2001. Most valuable player: A robot device server for distributed control. In *Proceedings of 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1226–1231.

Gerkey, B. 2006.

Martin, F. 2001. *Robotic Explorations: A Hands-On Introduction to Engineering*. Prentice-Hall.

Thrun, S.; Burgard, W.; and Fox, D. 2005. *Probabilistic Robotics*. MIT Press.