Roomba Pac-Man: Teaching Autonomous Robotics through Embodied Gaming

Brendan Dickenson **Odest Chadwicke Jenkins** **Mark Moseley**
David Bloom **Daniel Hartmann**
Department of Computer Science
Brown University
115 Waterman St.
Providence, RI, 02912-1910
{bcd | cjenkins | mmoseley | dbloom | dhartman }@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 an compelling and accessible application in a noncontrived 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 and Monte Carlo Localization. However, the impact of such coursework can be difficult to conceptually translate beyond the academic setting. Tangible artifacts produced in robotics courses are often overly structured (e.g., simulations, toy-level robots), difficult to reproduce (e.g, expensive equipment), or are distant from deployment in society. Our approach is to explore different approaches to autonomous control with focus on a specific task that is compelling, reproduceable 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 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 classic 1980s arcade game *Pac-Man*, where the player-controlled agent must navigate a maze to consume "dots" while avoiding "ghosts". In RPM, a virtual Pac-Man is replaced with a physically embodied iRobot Roomba vacuum equipped with a webcam and onboard computing. Students perform three introductory projects that cover subsumption (Arkin 1998), localization (Thrun, Burgard, & Fox 2005), and path planning in the context of RPM, allowing for normalized comparison and appreciation for the relative strengths of both approaches. RPM leverages Roombas as a cost-effective and deployable solution for teaching Robotics on real robots. Following the desire for reproduceability, students use the Player robot server and Gazebo simulation platform (collectively referred to as PSG) to develop control clients. RPM is placed in typical human environments containing fiducialized versions of "food pellets," "ghosts," and "power ups." Student's control clients score points by vacuuming pellets and visiting power-ups within a fixed amount of time.

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 striped down educational robot, and certainly not a toy.

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

Course Structure

The structure of Brown CS148 consists of 3 introductory labs and 3 control projects to reinforce material presented in lecture. Labs are simple projects that are designed to give students an introduction to using PSG and the Roomba hardware. The labs progress through basic reactive obstacle avoidance, blobfinding, object/fiducial seeking , and color calibration with 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. These labs build on one another to yield a robot client that is the foundation for the first project of RPM.

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, integrates all the topics covered previously in the labs. The final 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 in the real world, using the estimates from their MCL system from the second project. For final projects, undergrads develop robot clients of their own design to compete in a final competition. Students are encouraged to either strengthen their existing code or blend their creativity with concepts from the course, such as learning policies from demonstration or performing SLAM.

Our approach offers a logical transition through the material. Starting with three straightforward lab assignments (two of which are in simulation) allows the students to gain a mastery of the platforms as well as a basic understanding of fundamentals of robotics (odometry for instance), before dealing with more complex ideas or issues with real world implementations. The third lab provides a transition from simulation to real world implementation. The second project gives the students the chance to successfully implement MCL in simulation where debugging and lighting conditions are much easier before being forced to make it work in the real world.

RPM Platform

We aimed for RPM to be cheap, reproduceable, 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. We use standard Dell Dimension laptops which cost \$500 each to 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 we have extended to incorporate more of the Roomba's features (e.g., IR, vacuum, etc.). Finally, we mounted Logitech Communicate STX webcams on the Roombas, costing about \$30 dollars each. Thus, for under \$700 a robot, we have a very functional, real world robot with the ability to manipulate objects (i.e., vacuum). An early version of RPM is shown in Figure 1.

While cheap, our infrastructure is far from optimal due to their size and weight of the laptop and the sketchy nature of webcams. Other efforts have explored better options such as Gumstix embedded boards and MacMinis. The primary strength of these approach follows from the philosophy of PSG itself in that it is portable and flexible to the specific of the robot hardware. If one needs better computation, one can buy a more powerful computer. If better or different sensors are needed, attach a suitable device and use or write the Player interface.

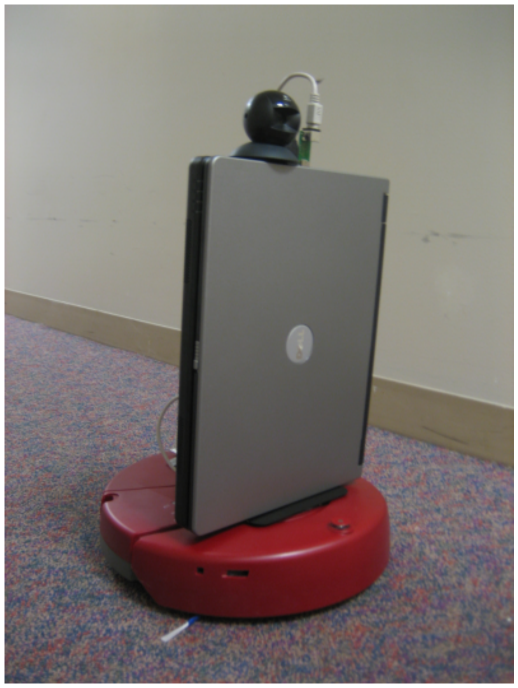


Figure 1: An early version of Roomba Pac-Man.

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 onboard 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

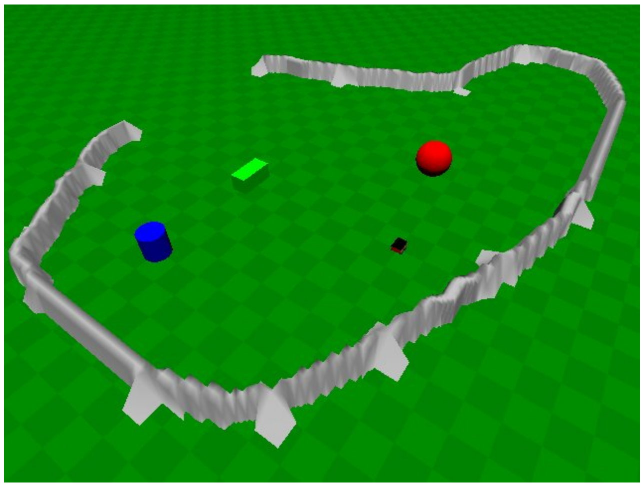


Figure 2: Simulated world in Gazebo for Labs 1 and 2

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.

We made several changes to player in the development of RPM. The original Player Roomba support allowed only for position control and reading the bump sensor. We added the ability to read the other sensors on the Roomba, including six infrared sensors including the IR-wall detector and the various buttons on top of the Roomba. We were able to add this ability by utilizing the Proxy structure supplied by Player, adding only the ”glue” to map the commands into the proxy functions. We also added the ability to control more of the Roomba outputs besides just the wheel motors. 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.

We made several other modifications to Player in relation to the camera. We added the ability to auto-detect camera parameters, enabling the webcams we had purchased 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. We hope to commit all of these changes to the Player project in the very near future.

Labs and Projects

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.

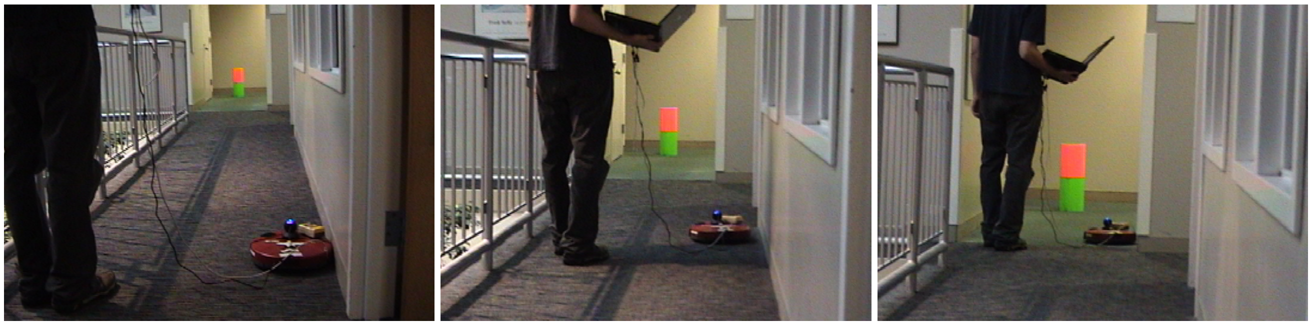
Lab 2: Object Seeking

In lab two, 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). 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). The robot has one bit of state indicating the current object of interest. The robot must seek, identify, and drive (as close as possible) to the the current object of interest without hitting it. Upon arriving to the current object, the state bit is flipped and the process continues to the next fiducial. Students also experiment with positioning the light source to get a controlled sense of how lighting affects vision sensing.

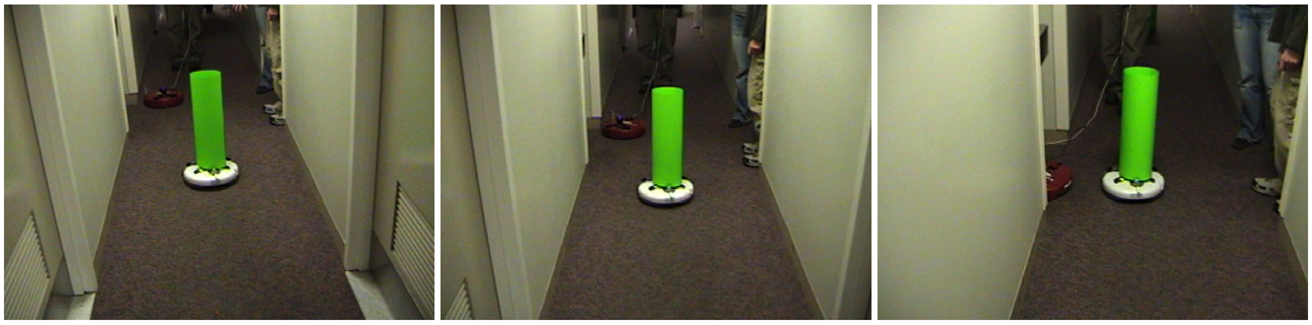
Lab 3 and Project 1: Color Calibration and Reactive Roomba Pac-Man

Lab 3 serves a gateway into the first project, writing a subsumption client for the Roomba Pac-Man task. Lab three extends Lab two's object seeking client to work with a physically embodied Roomba. Using the same Player proxies, the client (running on a supplied laptop) controls a Roomba endowed with touch/bump, IR, and camera sensing. The camera sensing is accomplished by attaching a web-cam to Roomba and having the client subscribe to the web-cam as a proxy. While the lab two 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. The goal for Project 1 is to play Roomba Pac-Man with a subsumption control policy. Lab 3 prepares students for this project by having them implement the following basic unprioritized functions:

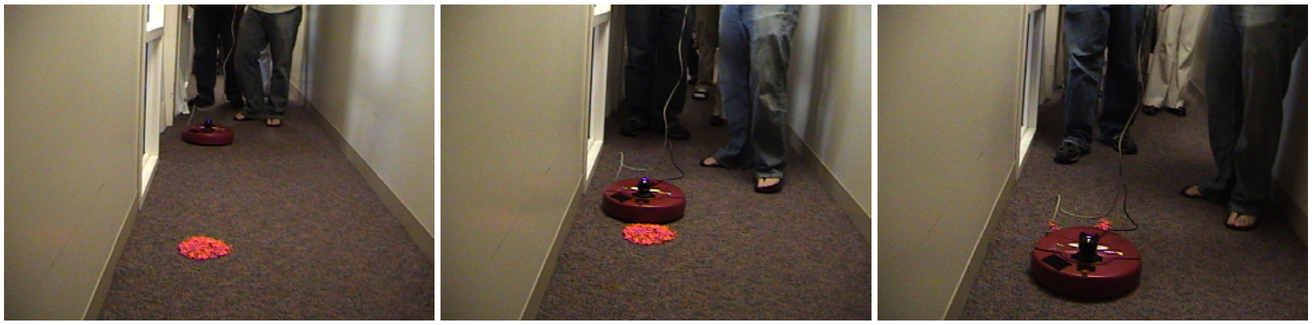
- Fiducial attraction: same as in lab 2, except the sought cylindrical “Power up” fiducial will be composed of two colors, orange over green.(Figure 3(a))
- Fiducial avoidance: detect and avoid a green cylindrical “Ghost” fiducial by turning away from it. (Figure 3(b))
- Pellet consumption: detect and drive over a pile of orange colored “food pellets” on the floor. (Figure 3(c))
- Wander: wander around an environment without an objective.



(a)



(b)



(c)

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

- Wall avoidance: detect collisions with physical or virtual walls and move to avoid these contacts.

Project 1 involves two main deliverables, a demonstration of the subsumption client and electronic submission of the work (project write-up, source code, and other materials). In the demonstration, we look for understanding of the subsumption architecture (prioritization and statelessness), fiducial recognition, responsiveness to obstacles and performance in the Roomba Pac-Man task (ghost avoidance, food pellets vacuumed). Project write-up should address the design choices and implementation approach for the subsumption architecture and calibration procedures.

Project 2: Monte-Carlo Localization

Project 2 involves implementing Monte-Carlo Localization (MCL) in preparation for the third project: Deliberative Roomba Pac-Man. Students are given the fifth floor of Brown's Computer Science building as a Gazebo world file (Figure 4). This world file is a functionally exact recreation of the world in which the students will compete in Deliberative Roomba Pac-Man in the third 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 the third project which is implemented in the real world. As we do not have enough laser range finders for all the Roombas, web-cams and blobfinders must be used. 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 for RPM in project three.

Project 3: Deliberative 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. In order to create an effective Roomba Pac-Man player, this project forces the students to pull knowledge from all the areas of robotics including but not limited to: sensing, in order to get meaningful data about the world; perception, in order to maximize the information gleaned from the senses; decision making, how to make smart decisions under uncertainty; and motion control, in order to maximize speed to and from objects.

Improving the sensory inputs is a way for students to improve the data they are using to perceive the world. Students must ensure that their color calibration for the blobfinder has optimal thresholds in order to get back usable data. Furthermore, they are free to modify the hardware (within reason) to increase the amount of sensory data they receive. For instance modifying the Roomba to have two web-cams could potentially yield a bountiful increase in sensory data.

Students must also tackle a wide variety of issues in perception. There are many ways to glean information from a single camera frame. The PSG blobfinder detects blobs, nothing else. However, extending this blobfinder to determine distances and angles to blobs should not be very difficult. Filtering out noise from the images is a critical task, and one that plagues robotics to the highest levels. Efficiently maximizing how much information is leveraged from single frame is very important. Furthermore, students must figure out how to estimate the state of the world. The natural state estimation technique would be to use the MCL from project two, as, thanks to PSG, it will port directly to the Roombas.

Students then must develop a planning algorithm to deliberatively control their Roomba. This can be a simple as using Dijkstra's to calculate the shortest paths to fiducials. The algorithm, however, must take in to account the presence of ghosts and their random movements throughout the world.

Finally, students must also develop a motion model for controlling their Roombas. While PSG nicely abstracts the commanding of the motors, it leaves room for development of different control loops. P, PD, and PID are all viable control options, but it will be important to maximize one's speed in order to get the highest score.

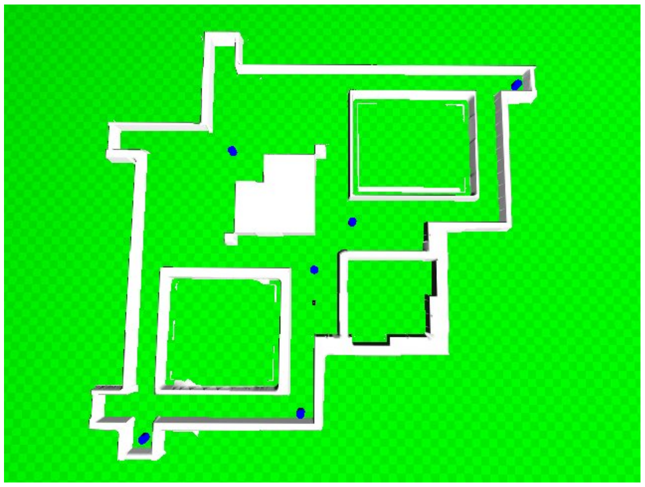


Figure 4: Map of the fifth floor of the Brown CS department

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. 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 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 teleoperation client that will allow a person to play RPM. Ideally, the teleoperator 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.

References

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

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.

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