Toys and Tools: Accessible Robotics via Laptop Computers

Morgan Conbere and Zachary Dodds

Harvey Mudd College Computer Science Department
301 Platt Boulevard
Claremont, CA 91711
mconbere, dodds@hmc.edu

Abstract

The ubiquity and capability of off-the-shelf laptop computers offer robotics and AI researchers a remarkable opportunity to reach into the broader computer science curriculum. At Harvey Mudd College we have developed two "lines" of laptop-controlled robots. The first, based on iRobot's vacuums, provides an inexpensive and autonomous platform suitable for indoor, human-scale environments. The second, based on PowerWheels toys, offers an inexpensive and capable platform for large, outdoor navigation and planning tasks. Both of these platforms enable cost- and time-effective undergraduate engagement in the ongoing community of robot-themed venues, exhibitions, contests, and conferences.

Overview

As a small undergraduate institution, Harvey Mudd College has sought inexpensive but powerful robotic platforms on which to base our AI curricula, independent student projects, and faculty-led research investigations. In the past decade the number of our students who use portable (laptop) computation has risen from almost zero to over 50%, and each incoming class seems to continue this trend. Leveraging students' computational resources for robotics projects offers us several pedagogical advantages over stand-alone platforms:

- Although machines that our department purchases grow older each year, students' laptops get newer (along with their owners).
- Students can develop their robotic software and interfaces anywhere: in their dorms, away from school, in the lab – whether or not the physical robot is present.
- Developing robot- and sensor-controlling software uses the same IDEs, software environments, and OSes as their other computational coursework.

The "catch" in the above list is, in fact, the robot itself. Until recently, the available mobile platforms could not provide a price/performance ratio that allowed broad deployment within small laboratories and institutions.

Through the 2006-2007 academic year, teams of faculty and students at Harvey Mudd College composed the interfaces necessary in order to create laptop-controlled platforms atop two new and promising educational platforms: the iRobot Create and FisherPrice's PowerWheels line of small vehicles. This paper summarizes these efforts, their results, and the benefits and drawbacks that come with laptop robotics.

iRobot's Create: Indoor Autonomy

The release of the iRobot Create in January, 2007 has set a new standard for the capabilities of a programmable, low-cost mobile platform. We used the Create as our default platform for an undergraduate robotics elective in spring '07. Pre-built and very rugged, it turned out to be an excellent foundation on which to investigate the spatial-reasoning algorithms that have become the foundation of computational robotics: Monte Carlo Localization, mapping, path planning, and navigation.

Positives: **Easy access, flexible interface**

The Create operates as a serial device, either tethered to a computer or via a Bluetooth wireless connector, the most reliable of which is ElementDirect's BAM module. The platform is thus accessible from any computational hardware that "speaks serial." This includes every computer in use today – what's more, the programming language needs only to be able to access the serial port. Again, this is universal.

Our students began familiarizing themselves with the Create and its Bluetooth interface first by developing a wandering algorithm similar to the robot's preloaded wandering routines. Programming the robot to wander in response to wall-bumps and odometric thresholds motivated students to familiarize themselves with the robot's interface while providing a concrete example of a behavior-based robotic system.

Students then implemented Monte Carlo Localization (Thrun at al., 2001) using the same two sensors – tactile and odometric – in order to maintain a particle-based probability distribution over the set of possible poses of the robot. That the sensing does not suffice to localize unambiguously, in fact, *better* conveys the power – as well as the limitations – of particle filters in comparison with parametric ones such as the Kalman filter. On platforms with laser range finders Monte Carlo Localization tends to converge on the correct location so quickly that the underlying strengths – and assumptions – of the algorithm could be taken for granted.

In order to visualize their algorithms' processing, students used and improved a compact, Python-based GUI that plotted the robot's, the particles', and the obstacles' poses in a global coordinate frame. Built atop the Pyro and, now, Myro set of Python resources (Blank et al. 2005), this visualizer provides a convenient interface for students to debug their localization routines. Figure 1 summarizes one team's submission.

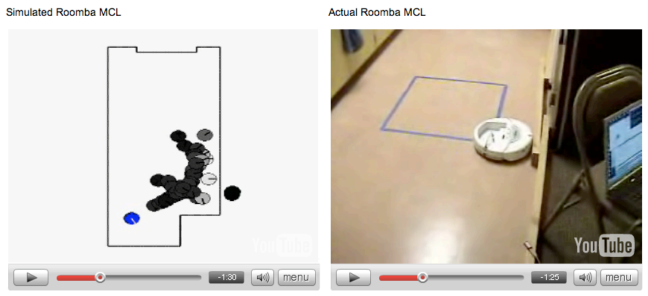


Figure 1 Snapshots of the visualizer (left) and the physical Create (right) during monte carlo localization. The white dot at left represents the most probable location of the robot, with darker pose hypotheses indicating lower likelihoods. Note that the best estimate is very close to the robot's real position, even as the odometry, shown as the blue dot, veers far from the true path.

Figure 1, shot in the kitchen area of one of our school's dormitories, emphasizes how the Create's accessibility extends beyond its low price of $150-$200 each with the power system and charger. Equally important, these robots are robust enough to be lent out to students during the semester: four teams brought them back to their rooms in order to develop there, instead of being tied to the labs on the academic side of our campus.

What's more, the Create's expandability led to several more ambitious projects. One pair of student created a robot whose motion was controlled by a freely spinning hamster ball; a second team programmed a trio of follow-the-leader Creates. A third group built a sketch-drawing robot, in which an off-board camera provided the localization accuracy that Create's odometry could not. The platform could then render human-specified strokes with a dry-erase marker on the hallway floors of our department. A small servo motor could deploy and lift the pen.

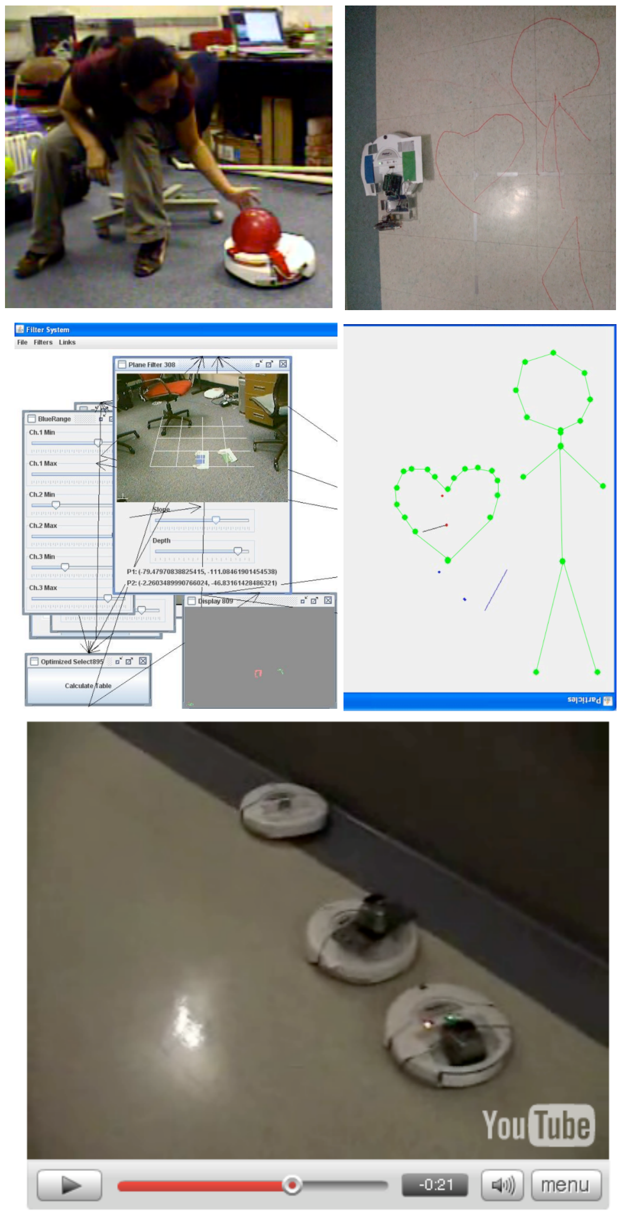


Figure 2 **(Top left)** A Create being controlled by a hamster ball positioned atop an inverted mouse. The large hamsterball replaced the mouse's trackball surprisingly well, and by simply reading mouse coordinates (all done within Python), it was possible to create a hamster-controlled vehicle. **(Top right)** Another team created a "graffitibot" that would render arbitrary images in dry-erase marker on the floors of the department. **(Middle)** The graffitibot used an off-board vision system and human-specified waypoints to determine its path. **(Bottom)** A third team created convoy behaviors and deployed them on a series of three Creates using the built-in infrared sensors and iRobot's virtual walls as IR emitters. The virtual wall on the center robot stands on a raised platform in order not to block the infrared receiver, which is located at the front of the platform. Nygaard 2007 has a complete description, along with video.

Negatives: local sensing, nonlocal computation

The Create has only local sensing built-in: two bump sensors, five cliff sensors, wheel encoders for odometry, and several measuring the system's electrical state. In our opinion, this lack of reach into the off-board world stands as the platform's biggest drawback. The "green box" offered by iRobot and known as the *command module*, provides a very small amount of computational capability, but does not extend the sensor reach *per se* beyond the perimeter of the platform.

The two most accessible sensing modalities that can compensate for this limitation are (1) sonar range sensing and (2) vision. Both require an on-board computer to collect and process the substantial streams of raw data involved. Other research groups have shown success in mounting special-purpose hardware to Creates for this purpose, e.g., the Gumstix computers advocated by USC's *Interaction Lab* (Mataric et al., 2007), the recently released recipe involving CMU's Qwerk board (Nourbakhsh et al., 2007), or the compact formfactor that characterizes Brown University's SMURV platform (Lapping-Carr et al., 2008). In contrast, we have experimented with approaches that use existing laptop computers to serve this purpose.

Figure 3 depicts two such COTS-laptop designs that we have publicly demonstrated. The top image shows a Create which uses vision to autonomously follow its sibling, an iRobot Roomba Red, as exhibited at AAAI 2007. The bottom image adds panning sonar sensors to the previous vision-only system. The robot shown competed at the Tapia Robotics Competition in Orlando, Florida in October, 2007.

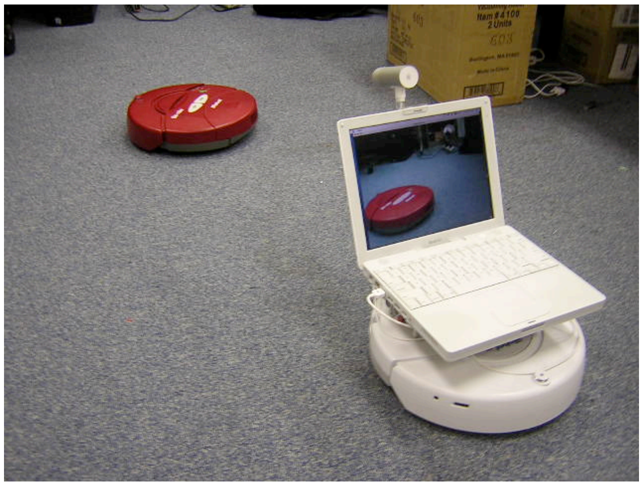


Figure 3a The Create chasing a Roomba Red based on color cues extracted from an onboard webcamera, exhibited at AAAI 2007

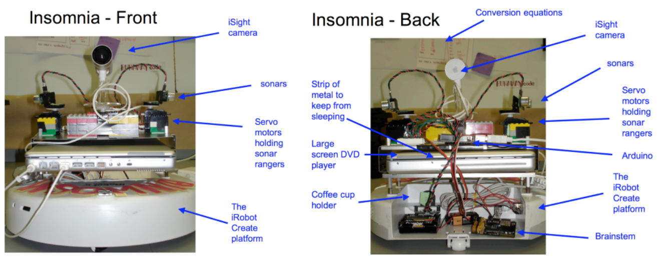

Insomnia - Front
iSight camera
sonars
Servo motors holding sonar rangers
The iRobot Create platform
Insomnia - Back
Conversion equations
iSight camera
sonars
Strip of metal to keep from sleeping
Servo motors holding sonar rangers
Large screen DVD player
Arduino
Coffee cup holder
The iRobot Create platform
Brainstem

Figure 3b Another design, revised from Figure 3a, using a closed laptop. Three sophomores used this platform as their entry to the Tapia's robotics competition in October, 2007.

In our experience, this laptop-based approach offers both plusses and minuses. Its low cost (the laptops already exist), familiar programming environments, wireless capabilities, and the resulting ease in adding sensors argue for placing laptops on board. Yet owners are also rightfully concerned about entrusting their computer to the fortunes of a possibly erratically moving vehicle. Also, although certainly powerful enough, the Create is really too small to comfortably contain the large desktop-replacement machines that many people prefer in a portable computer today. A computer any larger than a small Mac PowerBook will extend beyond the boundary of the Create, preventing the bumper from sensing obstacles in at least some orientations of the platform. Aesthetically, too, we have reservations about the designs we have tested to date: overall, they don't "sit well" or "fit well," despite their advantages for sensor and student accessibility.

PowerWheels for outdoor applications

This sense that the Create was too small of a platform to comfortably support a laptop led us to explore an alternative scaffolding for accessible robotics: the Fisher Price *PowerWheels* toys. Designed for small children, these plastic vehicles come with a rechargeable battery and motors that support well over an hour of operation and can carry 40-50 pounds of external load without difficulty. A bit large for indoor use, these vehicles are an excellent option for accessible outdoor robotics. Indeed, our designs derive directly from Bob Avanzato's work at Penn State Abington, where an annual *Mini Grand Challenge* contest challenges such vehicles to autonomously navigate the university's campus paths (Avanzato 2007).

To interact with motors and sensors, we have leveraged pair of well-supported microcontroller boards: the arduino and the brainstem. The process of altering the vehicle to run under laptop control requires less than an hour of student effort and approximately $100 in easily obtainable parts (Avanzato 2008). Figure four shows two of our vehicles.

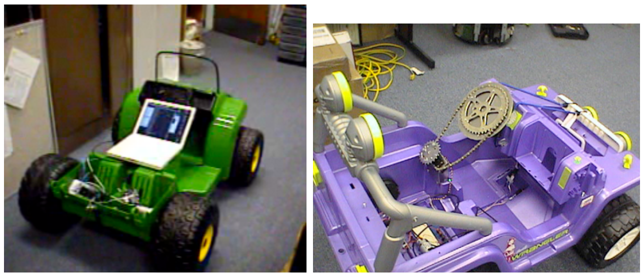


Figure 4 Two PowerWheels vehicles, adapted to run autonomously via an onboard laptop computer. Best suited for outdoor applications, these vehicles used vision, GPS, and sonar sensing in order to follow trails and navigate paths on campus.

Leveraging OpenCV on Mac OS X

Cameras for laptops are cheap and available sensors, but without software to interface with these webcams they can not help our robotics efforts. OpenCV, an open source computer vision library, provides the low level access to the hardware as well as providing higher level functionality in areas such as image processing and object recognition. Because OpenCV's support for MacOS X was quite limited, as part of this project we sought to extend OpenCV to take advantage of the native development environment and libraries available to all Macs.

This project began in order to free Open CV's Mac port from its many external dependencies, e.g. **`libjpeg`**, **`libtiff`**, **`ffmpeg`**, and several others. Indeed, the dependency on **`ffmpeg`** required a version older than the current release, which complicated the installation of the vision library. We succeeded in freeing the system from those dependencies, relying instead on Mac OS's wealth of built-in support for image and video formats. In addition, a **`make`** option now allows users to create a *Framework* build of OpenCV, Macs' native format for building software that can be installed via drag-and-drop. The resulting library retains the wealth of computer vision algorithms for which OpenCV is known, but now in a package that helps preserve the sanity of users whose operating system is Mac OS X. The new system has been folded into the Sourceforge repository (OpenCV, 2007).

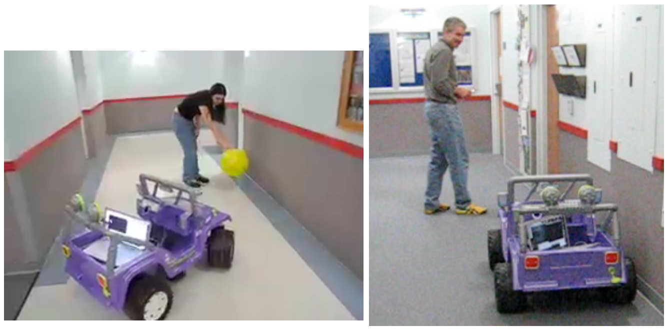


Figure 5 Color tracking – and mistracking – using OpenCV.

As examples of the visual processing that has leveraged OpenCV, one PowerWheels vehicle was programmed to approach and follow yellow objects: this successfully tracked our beachball both indoors and out, but occasionally got distracted, e.g., by an unusually bright pair of tennis shoes (Figure 6). Another application took advantage of the morphological operators OpenCV implements in order to support subtasks of the Mini Grand Challenge: finding cones, segmenting road, and mapping the environment.

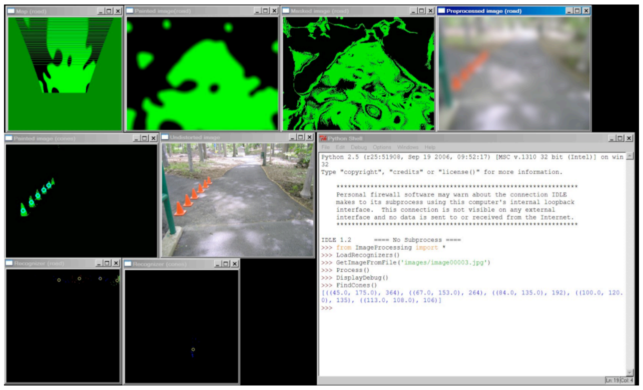
We briefly consider each subtask here:

To find cones, we first mask the input image using a cone color filter as described above. Once the image has been masked, we find connected components of marked pixels. We then decide that any component that is large enough must be a cone. This removes noise but does not account for other cone-colored objects in the image. Once the connected components representing cones have been found, we calculate the center and determine that to be the center of the cone in the image. This position will later be transformed and placed on our map.

Recognizing road by color is made difficult by the fact that roads are multi-colored and have a great deal of variation from pixel to pixel. We can account for things like light and shadow using the clusters in our color recognizer. To minimize the noise of pixel-to-pixel variations, we preprocess the image by applying a blur to the input image. This preprocessed image is then masked using the color filter. The masked image is then post-processed by alternately blurring and thresholding the masked image. This both fills in missing pixels in large blocks of included pixels and eliminates noise. This image (called 'Painted image') is then transformed onto the x-z plane to be passed into our mapping system.

The mapping system can determine the probable location of road with respect to our robot, given the results of the above processing steps. Specifically it transforms each pixel of the masked image down onto the x-y plane and then stores these values in a map image where each pixel corresponds with a box in space. Currently the mapping system simply returns this image so that it can be displayed as seen below in the top left corner of the screen shot. Note that this system is limited in so far as the "stretch" of the masked image produces the striations observed below. We plan on blurring the map image to eliminate those striations. Since we use brightness to indicate strength of knowledge this will result in those portions of the map simply having less weight.

Figure 6 summarizes these processing steps with a screenshot of the visual pipeline employed.



```
Python 2.5 (r25:51908, Sep 19 2006, 09:52:17) [MSC v.1310 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.

    ****************************************************************
    Personal firewall software may warn about the connection IDLE
    makes to its subprocess using this computer's internal loopback
    interface.  This connection is not visible on any external
    interface and no data is sent to or received from the Internet.
    ****************************************************************

IDLE 1.2      ==== No Subprocess ====
>>> from ImageProcessing import *
>>> LoadRecognizers()
>>> GetImageFromFile('images/image00003.jpg')
>>> Process()
>>> DisplayDebug()
>>> FindCones()
[((45.0, 175.0), 364), ((67.0, 153.0), 264), ((84.0, 135.0), 192), ((100.0, 120.
0), 135), ((113.0, 108.0), 106)]
>>>
```

Figure 6 A screenshot of the OpenCV-based cone- and road-finding software for our "Gator Jeep" PowerWheels platform. Color segmentation, morphological operators, region extraction and convolution are sequenced in order to stay on the path and away from obstacles (orange road-cones, in this case)

Perspective and Continuing Work

With iRobot's release of the Create and Bob Avanzato's pioneering work with PowerWheels vehicles, we have found that the distinction between educational platforms and research-ready platforms is rapidly fading. In both cases, the use of commercial, off-the-shelf computation in the form of students' laptop computers offers considerable advantages and flexibility:

- In each case, the focus is ***software and computation***, not hardware or assembly. Though we did spend a considerable time on the PowerWheels hardware, but with that hard-won experience, future development can focus on the computational facets of the field.

- iRobot's platforms are rugged, familiar, and are suited to exploration within ***indoor, human-scale environments***.

- The PowerWheels platforms are similarly robust, but extend our students' reach to ***outdoor environments*** -- in particular, to the Mini Grand Challenge at Penn State Abington or the International Ground Vehicle Competition in Rochester, Michigan (IGVC, 2008).

- Because they are ***peripheral to existing laptop computation***, both platforms leverage the sensors and computation that our department -- and our students -- already have.

- Both systems have ***low total cost***: $250-300 for a well-equipped iRobot system, and $450-500 for an autonomous PowerWheels platform.

The addition of an accessible version of the powerful OpenCV vision library for Mac OS X further eases the learning curve for student involvement with robotics algorithms, development, and the broader robotics community. We look forward to working with other educators and researchers to continue to make AI robotics a welcoming field in which undergraduates can quickly get up to speed and begin to make contributions. It is this active engagement within the larger community, we believe, that provides the most lasting benefit both to new students and established AI practitioners alike.

Acknowledgments

The authors gratefully acknowledge support from National Science Foundation DUE CCLI #0411176 and funds provided by Harvey Mudd College. Steven Wyckoff (HMC '07) and Mike Roberts (HMC '08) provided the mini grand challenge subtask descriptions and software.

References

Avanzato, R. (2007) Mini Grand Challenge Contest for Robot Education. In *Robots and Robot Venues: Resources for AI Education*, AAAI Technical Report SS-07-09, pp. 7-9, AAAI Press.

Avanzato, R. (2008) The Mini Grand Challenge website, www.ecsel.psu.edu/~avanzato/robots/contests/outdoor/

Blank, D.S., Kumar, D., Meeden, L., and Yanco, H. 2005) The Pyro toolkit for AI and robotics. *AI Magzine*. 27(1), pp. 39-50.

IGVC (2008) http://www.igvc.org/

Lapping-Carr, M., Jenkins, O. C., Hinkle, T., Schwertfeger, J., and Grollman, D. Wiimote Interfaces for Lifelong Robot Learning. In *Using AI to Motivate Greater Participation in Science*, AAAI Technical Report SS-08-08, to appear, AAAI Press

Mataric, M., Koenig, N., and Feil-Seifer, D. (2007) Materials for Enabling Hands-On Robotics and STEM Education. In *Robots and Robot Venues: Resources for AI Education*, AAAI Technical Report SS-07-09, pp. 99-102, AAAI Press.

Nourbakhsh, I., Hamner, E., Lauwers, T., DiSalvo, C., Bernstein, D. TeRK: A Flexible Tool for Science and Technology Education. In *Robots and Robot Venues: Resources for AI Education*, AAAI Technical Report SS-07-09, pp. 117-122, AAAI Press.

Nygaard, C., Pflueger, M., and Roberts, K. Multiple Coordinating Robots, at www.cs.hmc.edu/twiki/bin/view/Main/FinalReport (2007)

OpenCV (2007) http://sourceforge.net/projects/opencvlibrary/

Thrun, S., Fox, D., Burgard, W. and F. Dellaert, 2001. Robust Monte Carlo Localization for Mobile Robots, *Artificial Intelligence,* 128(1-2): pp. 99-141.