Leveraging Laptops: Resources for Low-Cost Low-Level AI

Zachary Dodds

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

Abstract

The ubiquity and capability of off-the-shelf laptop computers offer AI educators remarkable opportunities to reach broadly across the subfields of AI. By leveraging student laptops, instructors can make so-called *low-level* subfields of artificial intelligence – for example, computer vision and robotics -- almost as accessible as investigations into higher-level topics that might rely less on sensors and actuators. 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, built atop PowerWheels toys, offers a capable, low-cost base for large, outdoor navigation and planning tasks. Both platforms enable cost- and time-effective undergraduate engagement in the ongoing community of robot- and vision-themed venues, exhibitions, contests, and conferences.

Overview

As a small undergraduate institution, Harvey Mudd College has sought inexpensive but powerful robotics and vision 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 purchased by our department grow older each year, students' laptops get newer and more capable, 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 chassis is present.
- Developing robot- and sensor-controlling software uses the same IDEs, software environments, and OSes as their other computational coursework.
- Increasingly, laptops carry their own suite of sensors, such as cameras, accelerometers, and GPS, all available to students programmatically.

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

Through the 2006-2007 academic year, teams of faculty and students at Harvey Mudd College composed laptop-controlled interfaces robots atop two new and promising educational platforms: the iRobot Create and FisherPrice's PowerWheels line of small vehicles. In addition, we undertook an effort to make the OpenCV computer vision library more accessible to students. 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 used needs only to be able to access the serial port. Again, this is universal.

Our students accessed the Create's sensors and actuation through a 100% Python library we have dubbed *Erdos* (Erdos 2007). Other libraries are also freely available, e.g., as part of the Microsoft Robotics Studio (MRS 2007) and the Player/Stage/Gazebo robot server software (Gerkey, Vaughan, and Howard 2003). With the interface in place, students begin familiarizing themselves with the Create 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 motivates students to learn the system while implementing a concrete example of a behavior-based robotic system.

Other low-cost platforms offer similar access to reactive-style robotics investigations. The presence of a laptop, however, enables students to push beyond reactive control to explicit spatial reasoning and environmental estimation. Students first implement Monte Carlo Localization (Thrun at al., 2001) by 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 (Figure 1). That this sensor suite 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, students used and extended 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 highlights one team's submission.

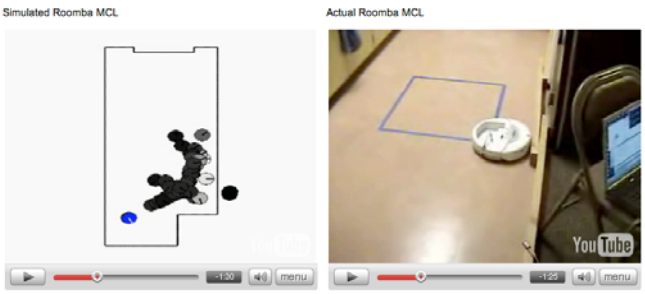

Figure 1 Snapshots of the visualizer (left) and the physical Create (right) during monte carlo localization. The white dot at left represents the most probable location of the robot, with darker pose hypotheses indicating lower likelihoods. Note that the best estimate is very close to the robot's real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

Because its images were shot in the kitchen area of one of our school's dormitories, Figure 1 emphasizes how the Create's accessibility extends beyond its low price of $150-$200 each with the power system and charger. 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 ambitious student-initiated projects. One pair of students 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 building. A small servo motor deployed and lifted the pen.

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 others 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 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.

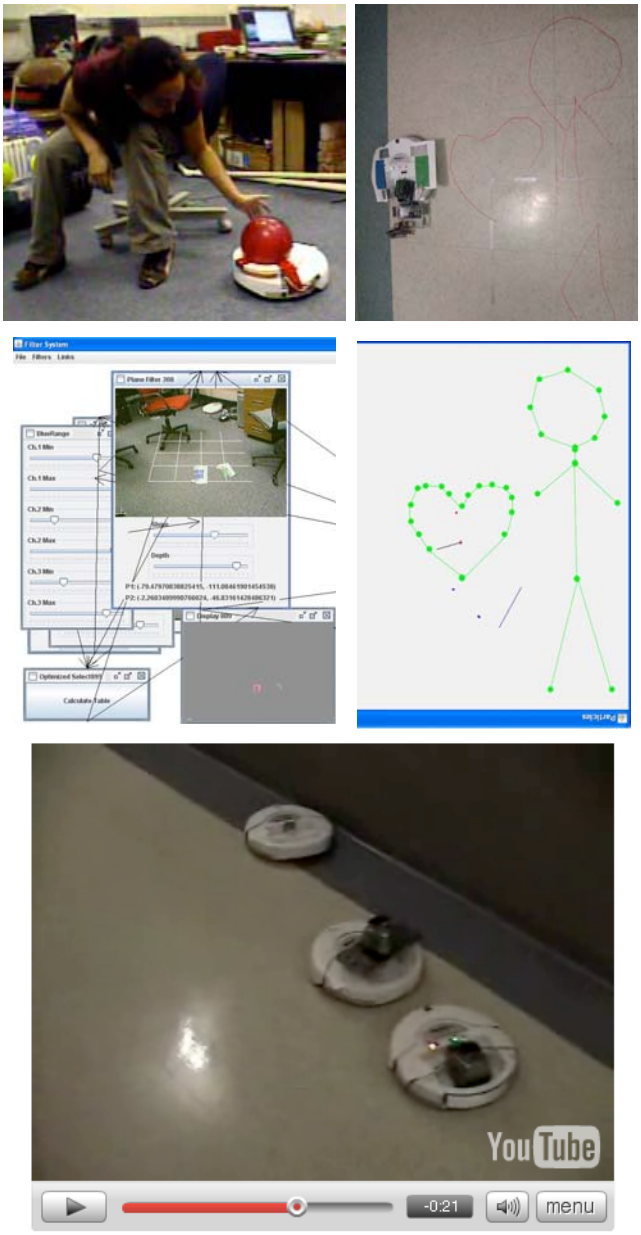

Figure 2 **(Top left)** A Create being controlled by a hamster ball positioned atop an inverted mouse. The large hamsterball replaced the mouse's trackball surprisingly well, and by simply reading mouse coordinates (all done within Python), it was possible to create a hamster-controlled vehicle. **(Top 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.

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

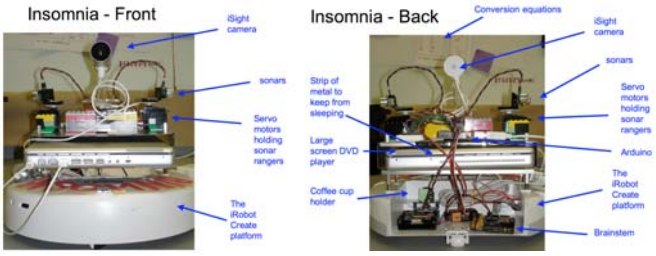

Figure 3b Another design, revised from Figure 3a, using a closed laptop. Three sophomores used this platform as their entry to the Tapia robotics competition in Orlando, FL in October, 2007.

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 too small to comfortably contain the 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 to comfortably support a laptop led us to explore an alternative scaffolding: Fisher Price *PowerWheels* toys. Designed for

small children, these plastic vehicles come with a rechargeable battery and motors that support 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* challenges such vehicles to autonomously navigate the university's campus paths (Avanzato 2007).

To interact with motors and sensors, we have used a pair of well-supported microcontroller boards: the Arduino (Arduino 2007) and the Acroname Brainstem (BrainStem 2007). Each of these sports a USB interface programmable by sending and receiving strings via an ordinary serial port. Both support infrared and sonar ranging sensors, as well. 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 4 shows two of our vehicles after being outfitted for autonomous operation.

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

Webcameras for laptops are cheap and available sensors, but software that provides access to the incoming raw pixels is neither easy to find nor to use. OpenCV, an open source computer vision library, provides low-level access to the hardware, as well as higher-level functionality in areas such as image processing and object recognition. Because OpenCV's support on the educationally ubiquitous MacOS X system lagged considerably behind Windows and Linux support, 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. With this native format for Mac libraries, OpenCV can now be installed via drag-and-drop. The library retains its wealth of computer vision algorithms for which it is known, but now in a package that helps preserve the sanity of users whose operating system is Mac OS X. Our Mac-capable system has been folded back into the main branch in the Sourceforge repository (OpenCV, 2007). On the Windows side, the release of the free Visual Studio Express compiler means that all partisans in the OS wars can now access pixels on their laptops, free of charge.

Figure 5 (**top**) Color tracking, and mistracking, using OpenCV. (**bottom**) Students with laptops – both Windows and Mac alike – are now able to prototype and test their software without the robot because OpenCV installs and compiles against student-developed code freely and easily. Porting to the robot involves nothing more than placing the laptop and the camera there.

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 5, top). Perhaps the most important capability enabled by OpenCV and laptop-based robotics is that students can test their algorithms without the underlying platform. Once the routines are up to snuff, getting them onboard the vehicle requires nothing more than placing the laptop there.

The Mini Grand Challenge offers three more examples of applications students have developed off-board and then placed onto a PowerWheels vehicle. They took advantage of OpenCV's morphological operators in order to support

the competition's subtasks of finding cones, segmenting the road, and mapping the environment.

We briefly consider each of these subtasks here:

To find cones, we first mask the input image using a straightforward color filter. We then find connected components of marked pixels. Connected components larger than a predetermined threshold are considered a cone. Note that this means that distant cones will not b categorized as such until the platform approaches them. This removes noise but does not account for other cone-colored objects in the image. Shape parameters, e.g., the fit of the outline to a triangle, remove many false positives. Once the connected components representing cones have been found, we calculate the region's centroid 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 is then transformed onto the x-z plane to be passed into our mapping system.

To provide an even more flexible classification system for distinguishing road from no-road surfaces, we have developed a backpropagation network that interfaces easily with the visual acquisition and motor control of the robot – precisely because of the uniformity and support of the COTS laptop interface. Initial tests using raw pixel values failed to converge, but preprocessed images yield powerful classifiers far more robust to lighting and texture variations than our hand-designed ones.

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 in Figure 6 Note that this system is limited in so far as the "stretch" of the masked image produces the striations observed below. Blurring the map image reduces those striations at the expense of fine image detail. We use brightness to indicate confidence in the map's representation; darker portions of the map simply indicate less certainty in the texture classification.

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

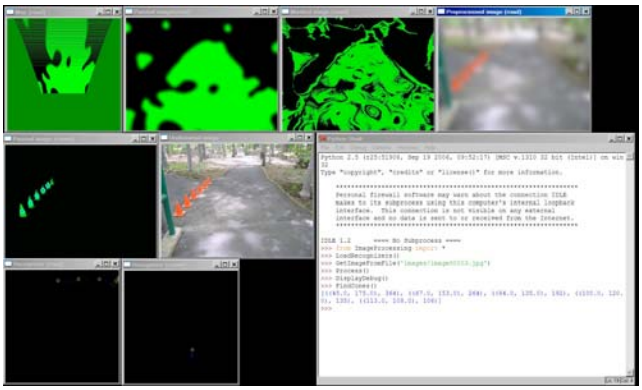

Figure 6 A screenshot of the OpenCV-based cone- and road-finding software for our "Gator Jeep" PowerWheels platform. Color segmentation, morphological operators, region extraction and convolution are sequenced in order to stay on the path and away from obstacles (orange road-cones, in this case)

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 low-level AI topics such as robotics and computer vision as welcoming as AI's less sensor-dependent subfields. In all cases, we feel that these new resources have enabled undergraduates to get up to speed quickly – to the point at which they can participate in and contribute to the ongoing AI robotics and vision communities. It is this active engagement within the larger community, we believe, that provides the most lasting benefits both to introductory 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

Arduino (2007), accessed 11/07 at http://www.arduino.cc/

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.

BrainStem (2007) The Acroname BrainStem Microcontroller, at http://www.acroname.com/brainstem/brainstem.html

ERDOS (2007) http://www.cs.hmc.edu/~dodds/erdos/

Gerkey, B., Vaughan, R. T., and Howard, A. (2003) "The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems". In *Proceedings of the 11th International Conference on Advanced Robotics* (ICAR '03), pp. 317-323, Coimbra, Portugal, June 2003.

IGVC (2008) accessed 11/07 at 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.

MRS (2007) Microsoft Robotics Studio, accessed 11/20/2007 at http://msdn2.microsoft.com/en-us/robotics/default.aspx

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.