A Computational Focus for Robotics Education

Zachary Dodds, Kristina Ming, Chris Eriksen,
Shih-Chieh Hsiung, Xin Huang, and Zakkai Davidson
Harvey Mudd College Computer Science 301 Platt Blvd. Claremont, CA 91711
contact: dodds@cs.hmc.edu

Abstract

This paper details two undergraduate robotics projects that highlight sophisticated, accessible interactions with spatial representations and computational reasoning for ground and aerial navigation. The results suggest that robotics education may now have the resources – both in hardware and software scaffolding – to bring computationally focused curricula to a wide audience. We argue that the field's computational facets will emerge its empowering force, and that emphasizing them at the undergraduate level will benefit students and professional practitioners alike.

Is Robotics CS?

Certainly not! Robotics depends on insights and practices from many fields. In fact, it is robotics's breadth that ensures the success of its precollege educational programs such as VEX and FIRST: there are many distinct niches in which students can build an identity. What's more, an even wider variety of research pursuits share the aegis of *robotics*, as evidenced by the annual programs at IROS, RSS, AAAI, and ICRA.

Yet precisely because of its integrative success, robotics today enjoys forward-facing vistas similar to computer science's in the mid-twentieth century. On one hand, robotics benefits from the ingenuity and maturity of device design and control; over time, more accessible, powerful, and robust software abstractions will continue to emerge from those efforts.

Just as hardware dominates the connotations of the word "computer," so does it underlie those of the word "robotics." Physical embodiment, however, does not diminish computation's importance in either case. Indeed, one might argue that, today, computation's influence far exceeds that of its enabling substrates.[1]

Drawing inspiration from this history, this paper argues that undergraduate robotics is poised to benefit from a focus on the computational facets of the field. Here, we present field-tested, cost- and curriculum-accessible resources to support precisely such a computational focus:

• with the scaffolding provided by ROS, students with one semester of programming background can create, deploy, and use substantive representations of *large, uncontrived, human-scale environments*.
• we highlight deterministic and probabilistic algorithms for spatial reasoning and task-level control *accessible for from-scratch implementation*
• we report, too, on recent platforms that make it possible to deploy these computational insights and skill-building *without sacrificing robotics's hands-on approach*.

These items do not push forward the boundaries of knowledge representation and reasoning (KRR) *per se*. After all, spatial representations accessible to first-year undergraduates must build upon the field's well-established foundations. Yet we believe that this work *does* contribute to the efforts of robotics in general, and KRR in particular, because it offers a compelling, readily reproduceable onramp for talent into the field. By curricularizing and field-testing the compelling facets of robots' spatial representations and reasoning through hands-on projects, this work strives to advance the state-of-the-art of *who* will contribute to KRR's goals -- and those of similar groups -- in the future.

In sum, although robotics is not CS, it is the field's *computational* and *representational* advances that will pay

[1] This holds, we believe, atop both artificial and human machines… .

the greatest future dividends. As a result, KRR will benefit from active partnerships and deliberate scaffolding within early-undergraduate CS curricula. This work provides one such scaffold, field tested with widely accessible resources.

New platforms, old scaffolding

Consider the ever-widening space of new robot platforms: novel mechanisms and form factors inspire both applications and imaginations. Comparing the space of new robot platforms and the space of new software for those platforms, however, evokes a stark contrast. Just as each advance in computer hardware enables an even larger cascade of computation, so too each new robot offers far greater opportunities to integrate its sensors, architect its behaviors, and interact with its environment.

Here, we highlight two platforms that enable a focus on computational representations and spatial reasoning. Both have software scaffolding via ROS drivers (Quigley et al., 2009) and cost less than $300. Though we used these robots in undergraduate classrooms and laboratories, their capabilities would pique the interest of some graduate students and their accessibility would draw pre-college enthusiasts.

Human-scale navigation with the Neato

Shown in Figure 1 (left), the Neato vacuum robot is a capable, task-specific device akin to its older cousin, the iRobot Roomba. Its <$250 price tag and on-board 360-degree laser range finder set it apart, however – especially as a basis for robotics education. A team of three first-year undergraduate students, with one Python course each, outfitted our Neato with an existing netbook, running ROS, and a shelf velcroed to the top of the platform.

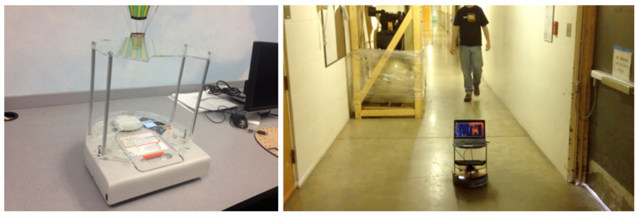


Figure 1. The knee-high sub-$300 Neato and shelf

So attired, the knee-high Neato becomes a general-purpose ground robot, programmable in any ROS-accessible language (we used Python), and capable of navigating large, unaltered human-scale spaces. Figure 1 (right) shows a snapshot of the robot near a passerby in a large labyrinth of hallways. The next section summarizes results and the resulting curriculum that uses this robot.

Aerial navigation with the AR.Drone2

The Parrot AR.Drone2 and newer, smaller Crazyflie quadcopters, at $300 and $180, respectively, offer cost-effective opportunities for robotics investigations to "get off the ground." With extant ROS drivers, both can be controlled simply and easily through Python or C++; we followed other projects' leads (Bills, Chen, and Saxena 2011) in using the AR.Drone2, i.e., the *drone*, because of its on-board cameras (Figure 2). This paper's penultimate section details our investigations in autonomous navigation for the drone.

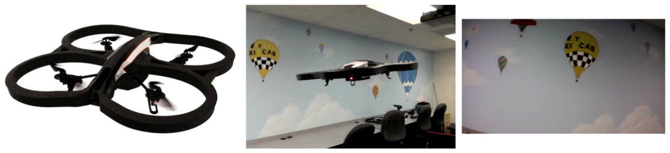


Figure 2. The ARDrone.2 and an example image

Building one's own navigation stack? Neato!

The past decade of ROS's influence on robotics software is difficult to overestimate (Cousins, 2011). Yet, from an educational standpoint, ROS's off-the-shelf algorithms are only a secondary contribution. Rather, it is ROS's consistent abstraction of sensors, actuators, and communications streams that has transformed robotics education. These are the computational building blocks that allow efficient composition of algorithmic procedures and representations, whether from scratch or using existing routines.

Why do early CS students continue to implement mergesort and quicksort, in light of the many already-optimized routines available at the push of a button? They do so because future computational creativity depends on ***two*** insights: "this is a subroutine I can use anywhere" and "this works because of specific structures and assumptions, and those may or may not apply in the future."

In this latter spirit, a summer '13 trio of first-year undergraduate students agreed to build a Neato navigation stack from scratch. This project sought to revise the curriculum for a CS2-level *Computational Robotics Lab*, five sections of which ran in 2012 with three sections on offer in Spring '14.

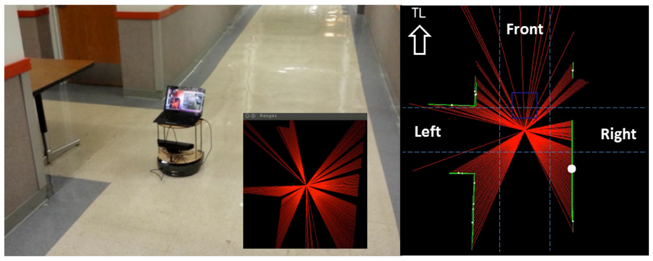


Figure 3. Raw and Hough-transformed laser data

With background of one Python-based programming course, the team accessed the Neato's low-level serial API and gathered its sensor data and control primitives into a single Python script within a week. Using ROS, they were able to visualize the laser-scan rays as the robot moved (Figure 3, inset).

Low-level reasoning

With these raw data available, the students leveraged OpenCV's (Pulli et al., 2012) built-in Hough transform to extract straight-line walls from the raw laser-scan rays (Figure 3, right). By correcting the robot's heading toward the most-parallel of the Hough-extracted walls and by avoiding obstacles too close to the front of the robot, the team had implemented its own low-level reactive corridor-follower.

Like many indoor environments, our academic corridors (the *Libra Complex*) alternate straight-line hallways with intersections of various sorts. By aligning to the dominant wall direction and measuring open space in each of four directions, the group succeeded in building their own intersection-recognizer that worked well over 95% of the time. Figure 3 shows the system identifying a three-way intersection whose branch extends to the left, i.e., a "TL."

Topological representation and reasoning

Robust intersection recognition can support further layers of capability, as long as the graph of hallways is known. A hand-built, topologically accurate but metrically approximate layout of the Libra Complex enabled the team to implement breadth-first search (BFS is an early CS topic) within that graph. BFS provides simple and optimal, at least in a graph-node-counting sense, path-planner, which can delegate progress from node to node to the low-level reasoning system. When faced with a choice, the axis-aligned topological map provides the basis for making the correct turn. Figure 4 shows three snapshots of a long run that encountered and avoided many unmodeled obstacles, static and dynamic, through the complex.

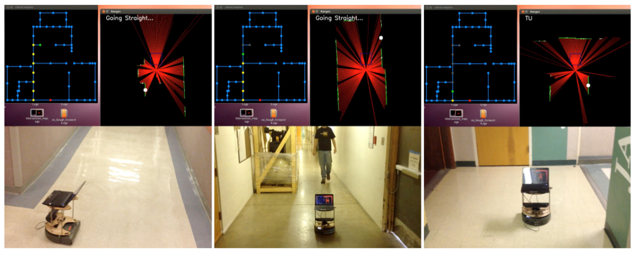


Figure 4. Navigation in a large, hand-built graphical map

Deliberative layers

That five of the ten weeks remained after building this reliable human-scale navigation system testifies to three things: ROS's excellent infrastructure, the Neato's powerful sensing, and Python's expressive power.[2] At this midpoint the team sought to improve their system in two ways: first, it should not need to know its starting point, i.e., it should localize itself and then path-plan to the goal; second, the robot should be as deliberate as possible in localizing itself efficiently.

The grid-aligned graph representation provides a natural foundation for localization: it contains a discrete set of all of the locations the robot *could be*. What remains is choosing actions in order to cull possibilities until the pose ambiguity disappears, i.e., classic Markov localization (Nourbakhsh, 1998) or Monte Carlo Localization with environmentally anchored pose populations (Thrun et al., 2001).

Again implementing from scratch, the team first tracked the possible locations of the robot as it made default intersection-choices through the map. With each new observation, the (uniformly weighted) viable hypotheses decreased until the robot knew its location; from there, it planned and executed a path using the graph-navigation routines. Figure 5 demonstrates one such run.

Yet, there remained (computational) degrees of freedom to exploit! Figure 5's localization routine used default turn options at each intersection: straight if possible, then right, then left, then turning around. With the environmental graph in hand, the team wanted to make a probabilistically better-informed turning choice during localization.

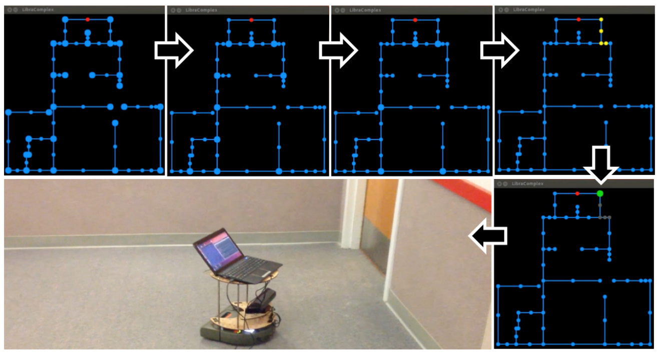


Figure 5. Markov localization culling the set of possible poses to one, then followed by navigation

To that end the group measured the Shannon entropy (Shannon, 1948) of the distributions of the possible

[2] The team's enthusiasm and ability are also part of this list, to be sure!

subsequent intersections for each of the robot's possible next turns. Figure 6 shows snapshots from the quicker alternative route that entropy provides for the same localization and navigation task run in Figure 5. The entropy metric offers at least two benefits: (1) insight into a fundamental, easily-implemented computational idea often missed in early CS curricula and (2) a sound basis for deliberative AI-level reasoning that few educational robots reach because of the overhead that can detract from computational considerations.

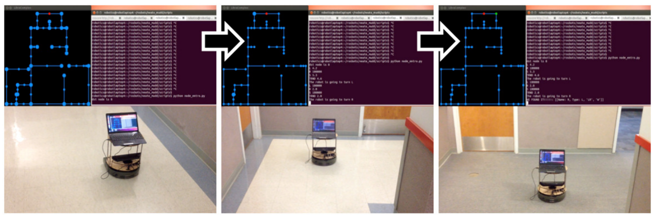


Figure 6. Entropy-based choices more quickly localize the robot and complete the navigation task.

Even building these (i) reactive, (ii) navigation, and (iii) deliberative layers from empty files, the team found it had its final two weeks to explore additional platforms (the CrazyFlie with LeapMotion sensor) and alternative predictive-path-planning strategies. In the latter, the system makes a probabilistic "best-guess" as to the direction of the goal and heads in that direction. This turns out to be a less effective overall approach because the cost of actually traversing those best-guess hallways is so dear in the cases when the guess is, in fact, incorrect.

Such accessibility suggests that, with the right resources, representation and spatial reasoning can productively anchor robotics education without sacrificing deep understanding or hands-on deployment. This paper's final section summarizes the past and present curricula that this Neato project validates.

2d-3d spatial coordination? Drone on!

In parallel with the ground-robot investigation, a team of three junior-level undergraduate students sought to investigate the extent to which an aerial platform, the drone, could leverage 3d spatial representations for task execution. The results demonstrate that, with appropriate curricular scaffolding, the sophistication of student-designed software can greatly exceed and extend that of the underlying robot hardware.

The team chose the task of *escaping the lab*, by which the drone would servo out of the door. They constructed a system to follow a straightforward error-reducing loop:

(0) First, use existing software (RGB-D SLAM) to build and access a 3d map of the lab

(1) Take an image from the drone's camera

(2) Localize by matching with the map's images

(3) Compute the heading from the pose estimate to the lab door and apply that velocity for two seconds

(4) If out of the lab, land; otherwise, hover and goto (1).

3d mapping on the shoulders of RGBD-SLAM

ROS's access to the community's codebase makes it possible to incorporate the results of systems too complicated to hand-build in a single term. For instance, the drone team started with the ROS-based RGB-D SLAM source code (Endres et al., 2012). Out of the box, the algorithm provided Kinect-based 3d point-cloud maps of the lab, known as the *hot-air lab*, for the balloons adorning its walls (Figure 7).

The team did adapt that source in order to preserve separately both the RGB images (40 of them) that texture the map and the relative positions of those images' viewpoints in the lab's coordinate frame.

Those 40 images, in turn, provide the raw material for the drone's localization. Each time the drone hovers, a novel image is taken and the system computes its closest match among the 40. The images' poses are known; thus, the location of the best-matching image provides an estimate of the drone's location.

Leveraging OpenCV for 2d-2d matching

In order to compute the best-matching of the map's 40 images to the drone's novel image, the team implemented a pipeline of 2d-2d matching routines from ROS's embedded OpenCV library. Figure 8 summarizes this pipeline, implemented using C++.

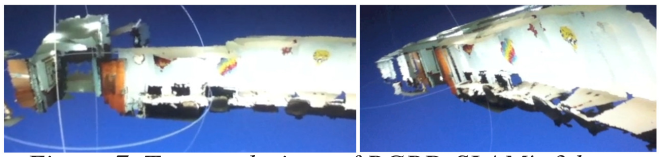


Figure 7. Two renderings of RGBD-SLAM's 3d map

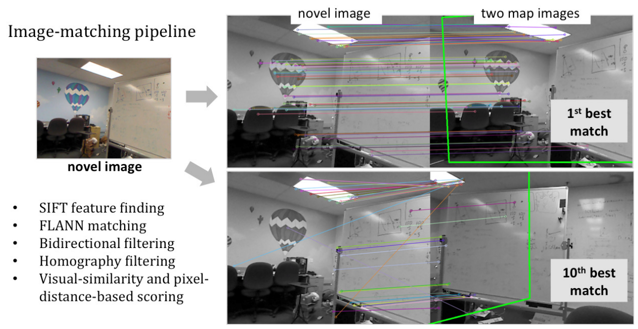
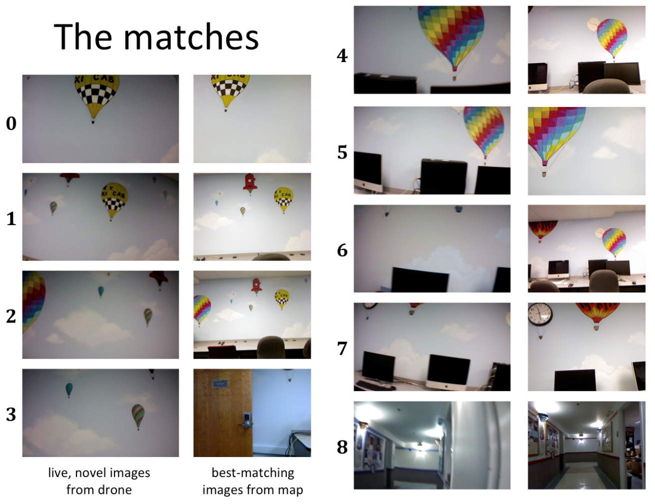


Figure 8. The drone's image-matching pipeline builds a confidence ranking for image-based localization after several layers of filtering SIFT features.



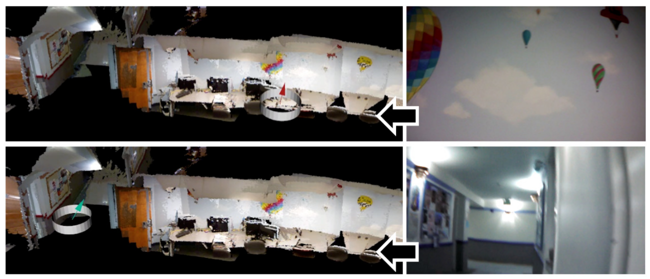


*Figure 9. (**left**) The nine matches made during one of the autonomous "escape" runs by the drone out of the lab. The wider images were taken live by the drone; the narrower ones were taken a week before and used by RGBD-SLAM to build the 3d map of the lab below. (**right**) Two 3d renderings of the drone's position (frame 2 and frame 8) during the same escape run.*

Computational Robot Lab

Weeks	Lab summaries
1	line following
2-3	ground visual servoing
4-5	low-level navigation
6-7	path-planned navigation
8	drone control w/kinect
9-10	aerial visual servoing
11 to 15	student-chosen tasks, such as these examples:

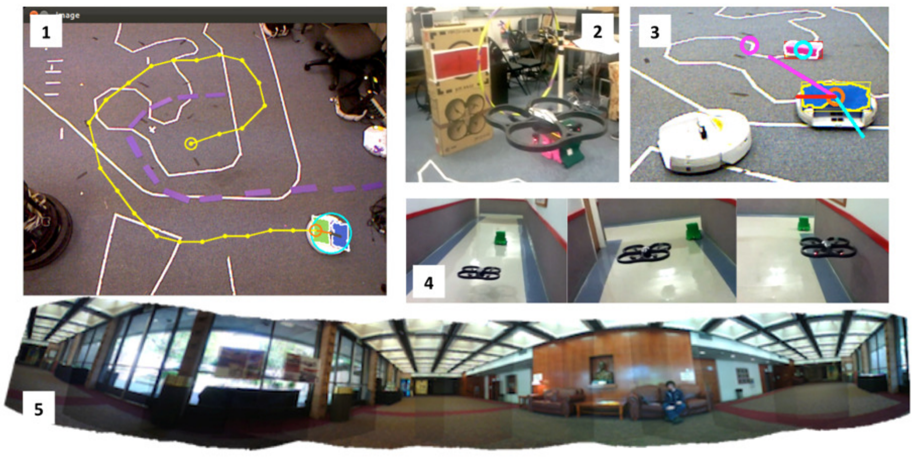


*Figure 10. (**left**) An overview of the field-tested computational robotics lab curriculum, designed for students with one semester of prior programming background (**right**) Student-proposed and –built capstone projects from that lab, including (1) A maze sensed and solved by a ground robot and offboard Kinect; (2) Autonomous flight through a hula-hoop via visual-servoing control; (3) forces superimposed on a robot seeking the magenta goal and avoiding the cyan obstacle using vision – the resulting control vector is in red; (4) A drone following an iCreate down a hallway; (5) a panoramic map created from an autonomous flight.*

Visualizing the results

An advantage of localization within a 3d map is that the system has a representation of the drone's position throughout its flight. Figure 9 (top) shows nine matches (including one mismatch) of novel drone images with map images through one lab-escaping run; at bottom are two snapshots from the rendering of the drone's path in 3d.

The group, with no prior experience in robotics, vision, or ROS, constructed a system that accomplished the *escape* task several times; improvements continue today. This result, produced without curricular support, suggests that – even in the actively developing areas of 3d maps, visual map-matching, and visual control of aerial robots – there exist untapped opportunities to build compelling curricula for a broad audience of students. What's more, those curricula do not have to sacrifice computational sophistication for hands-on deployment.

UG Robotics: computational *and* hands-on

So, how would such robotics curricula look? We have explored one possibility in 2012's offering of five sections of *computational robotics lab* (Dodds, 2013). In that course, pairs of students use both ground platforms and the aerial drones in developing a new autonomous capability each week or two; those experiences culminate in a month-long capstone projects of the students' own design. Despite the limited platforms available, the projects' creativity illustrates the power of a computation-first approach to the field. Figure 10 summarizes the semester-long curriculum and highlights several of those capstones.

That such hands-on labs require only one semester of computational background testifies to the remarkable resources available to robotics educators today. Yet, the projects detailed in this work show that even more may be possible with careful scaffolding. In the spring of 2014, at least three more sections of the lab will be offered; we anticipate that the Neato and AR.Drone2, both new additions, will enable spatial representations and algorithmic reasoning to feature even more centrally in this next iteration.

Right now, robotics education is dominated by wonderful programs whose emphases are the engineering facets of the discipline. Without taking anything away from the importance of those efforts, work in knowledge representation and spatial reasoning seeks to expand those computational underpinnings of the field *that outlast individual artifacts*. Our approach to robotics education seeks precisely that goal, which is why this work pushes the limits of $300 platforms instead of relying on the capabilities of $3,000 (or higher-cost) ones.

More broadly, we hope these efforts contribute to the ongoing development and maturation of robotics's curricular resources. To the extent that robotics education can embrace its computational core – i.e., building appropriate task representations and leveraging algorithms that support them -- we believe new students and seasoned researchers alike will benefit from the emphasis and interactions that naturally arise from such an approach.

References

Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R, and Ng, A. 2009. *ROS: an open-source Robot Operating System*. ICRA Workshop on Open Source Software, Kobe, Japan, 5/12-5/17/09.

Bills, C., Chen, J., and Saxena, A. 2011. *Autonomous MAV Flight in Indoor Environments using Single Image Perspective Cues*. ICRA Shanghai, China, 5/9-5/13/11.

Dodds, Z. 2013. www.cs.hmc.edu/~dodds/ROSatSIGCSE2013/

Endres, F., Hess, J., Engelhard, N., Sturm, J., Cremers, D., Burgard, W. 2012. *An Evaluation of the RGB-D SLAM System*. ICRA pp. 1691-1697, St. Paul, MN, 5/14 – 5/18/12.

Cousins, Steve. 2011. *Exponential Growth of ROS*. IEEE Robotics and Automation Magazine, pp. 19-20, March 2011.

Pulli, K, Baksheev, A., Kornyakov, K., and Eruhimov, V. 2012. *Real-time computer vision with OpenCV*. Communications of the ACM 55(6), pp.61-69, Jun 2012, ACM Press.

Shannon, C. 1948. *A Mathematical Theory of Communication*. Bell System Technical Journal 27(3): 379–423, July/October, 1948.

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

Noubakhsh, I. 1998. *Dervish: An office-navigating robot*. Artificial Intelligence and Mobile Robots, pp.73-90, MIT Press.