THE EXPERIENCE OF TEACHING SOFTWARE DEVELOPMENT IN A ROBOTICS PROJECT COURSE

Nicholas Roy
Department of Aeronautics and Astronautics
Massachusetts Institute of Technology
Cambridge, MA 02139
nickroy@mit.edu

John Leonard
Department of Mechanical Engineering
Massachusetts Institute of Technology
Cambridge, MA 02139
jleonard@mit.edu

Una-May O'Reilly
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139
unamay@csail.mit.edu

Daniela Rus
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139
rus@csail.mit.edu

Seth Teller
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139
teller@csail.mit.edu

Abstract
We describe a curriculum for a two-semester course sequence in hardware-software development, taking students through a complete development cycle of the various capabilities for an autonomous mobile robot. The experience of these courses has led to insights in teaching large-scale system development, especially with respect to software-intensive courses.
Keywords: Lab-based courses, robotics, software

1. Introduction

In 2005, five faculty from three engineering departments at MIT launched a year-long sequence, Robotics: Science and Systems (R:SS) I and II. In the first semester, we introduced students to the general topics of robotics, including control, state estimation, planning, localization and manipulation. R:SS I introduced each topic in a week-long lab, and then asked the students in small teams

to integrate the various capabilities they had developed into a unified system, performing a mini "course challenge". In this manner, we took the students through a complete cycle of developing all the various capabilities necessary for an autonomous mobile robot. In the second semester, the students chose one capability of the robot and studied this capability in more depth. The stated objective at the end of the second semester was to have the students as a group (rather than individual teams) develop a single "Grand Challenge" system capable of performing some real-world task. The task we chose involved developing a robot that could move around campus, collect objects and build a small structure in the hangar.

We spent considerable time teaching sound principles of software design, development and testing at regular intervals in the course, in order to facilitate the integration of different capabilities. However, the experience of teaching these courses has led to certain key insights in teaching large-scale systems development, especially with respect to mixed hardware- and software-intensive courses. We have just begun our third offering of the course sequence, and in this paper we describe the course structure and our implementation decisions. Additionally, we discuss three main insights in teaching a robotics systems class to undergraduate students. The first lesson we learned is the importance of structure. We progressively allowed the students considerable freedom in their final Grand Challenge project, and we learned that specific forms of guidance were essential to success. Secondly, we re-learned a lesson about spiral project management, that we believe is worth reinforcing. Finally, we report lessons on integrating a strong communication component into the course, and the lessons we learned in doing this.

2. Course Organization

An overriding theme of the course sequence is the model of spiral development, with which we take the students through the development cycle of capabilities for an autonomous vehicle multiple times. For example, the pedagogical goal of the first semester is to introduce the students to a broad spectrum of the technical and systems challenges in mobile autonomous systems, leading to a very structured development of a vehicle with moderate capabilities for the initial course challenge. The pedagogical goal of the second semester is to allow the students to explore a single technical area in some depth, and to introduce them to the technical and managerial challenges of developing a single capability in the context of a larger system. The combination of the different capabilities lead to the Grand Challenge vehicle.

While the course contains hardware design issues that naturally follow from a robotics program, the majority of the conceive-design-implement-operate process is focussed on software development issues. In the first semester of the course, the software development is very structured and focussed within teams, which has led to successful deployments of limited capability systems. In the second semester, the entire class form a single team with a substantial amount of freedom to develop new robotic capabilities in conjunction with a research faculty advisor.

3. R:SS I

The first semester, Robotics: Science and Systems I, is designed to introduce the students to different autonomous capabilities and the basic principles that underlie different algorithms. In general, we expose students to the state of the art in the different areas, such as estimation, planning and control. The course has a formal set of learning objectives, but we generally expect at the end of the course, that

1. the students will be familiar with basic implementations of kinematics, control theory, state estimation and planning to implement controllers, estimators and planners that satisfy the requirements of specified task;
2. can specify the requirements for an integrated hardware and software design and implementation of an autonomous system performing a specified task;
3. can implement the necessary hardware and software components in an integrated system and operate the system for an extended and specified time.

The course is structured such that there are two 50-minute lectures and two 120-minute lab sessions per week. The lectures generally cover the following topics:

- Actuation
- Control
- Locomotion
- Sensing and Perception
- Camera Models
- Software Engineering
- Control Architectures
- Localization
- Map Construction
- Planning
- Grasping and Manipulation
- High-level Vision
- Simultaneous Localization and Mapping

3.1. Structured Laboratories

There are a series of milestones, in the form of labs, that the students must meet, but the principal outcome of the class is in the form of the "Course challenge", in which students must build upon the skills of the preceding labs to develop a complete working autonomous system. We divide the class into groups of 3-4 students, and the each group works in parallel. After each lab is complete, the student teams brief the lecturers in 10 minute presentations, to demonstrate their implementation and any additional analysis they have performed.

1. Schematics: Layout and Components

   The first lab is designed to familiarize the students with their hardware kits, and learn some basic hardware debugging skills. Students are required to learn basic division of capabilities between hardware and software, to learn to read and understand circuit schematics, and practice the use of basic hardware skills such as soldering and multimeter use. We issue a basic kit of robot components including a partially-populated microprocessor board shown in figure 1, and the students must complete and test their board.

Figure 1: The robot components issued in the first laboratory to begin the hardware and software design, implementation and testing.

2. Motor Characterization and Control

   The second lab is designed to familiarize the students with their software development practice. We introduce the software development environment and require the students to practice the use of software management tools such as version control systems. We pose this instruction in the context of the mixed hardware-software system by asking for a software-based characterization of the motor subsystem. Students are issued with the additional components shown in figure 2 including the drive motors, and required to complete the motor circuitry, implement a controller in software and evaluate their controller.

Figure 2: The robot components issued in the second laboratory to begin the implementation of motor characterization and closed-loop control.

3. Robot Chassis and Driving

   The third lab is the first lab designed to test the students in their understand basic navigation concepts. At this point, the students assemble their hardware kits into a basic robot chassis, and control their robot using simple feedback loops on the wheel odometers, essentially navigating using dead-reckoning. The students are issued with components to allow them to build the complete frame shown in figure 3. The robot is a basic differential-drive system, with the drive wheels at the front and simple caster wheels at the rear.

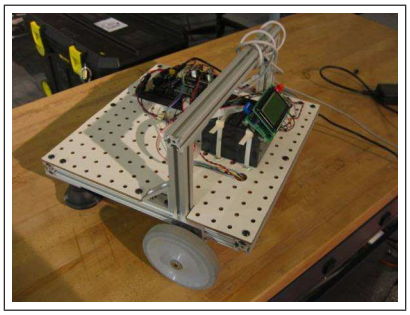

Figure 3: The robot components issued in the third laboratory to begin the implementation of basic mobility.

4. Light Sensors and Braitenberg Behaviors

   The fourth lab allows us to introduce sensing to the students, and implement feedback control loops around some environmental stimulus. We issue the students with light sensors, and require them to implement simple "Braitenberg" type light-following behaviours. One important lesson from this simple sensing modality is that the behaviour of the sensor changes substantially from night to day, and so the students quickly discover the need to recalibrate their systems regularly.

5. Software Engineering and Visual Servoing

   The fifth lab introduces some major changes to the robot. Until now, computation was largely performed off-board the robot with a serial cable to communicate with the on-board microprocessor. In this lab, we first issue the students with a laptop. However, we also provide the students with a camera (shown in figure 4, left) for more sophisticated sensing.

   Image processing is an intensive task, so we also introduce a large software framework called Carmen. Carmen is a publicly-available open-source robot control suite that facilitates distributed, networking computation. This allows the students to seamlessly parallelize operations between the on-board laptop and the off-board workstation. This software framework allows us to introduce new software development practices for embedded systems that the students may not have encountered previously.

   The goal of this fifth lab is for the students to implement a simple reactive ball follower by extracting recognizable colour features, and then using properties of the extracted features (e.g., blob sizes, positions in the image, as shown in figure 4, right) to generate control

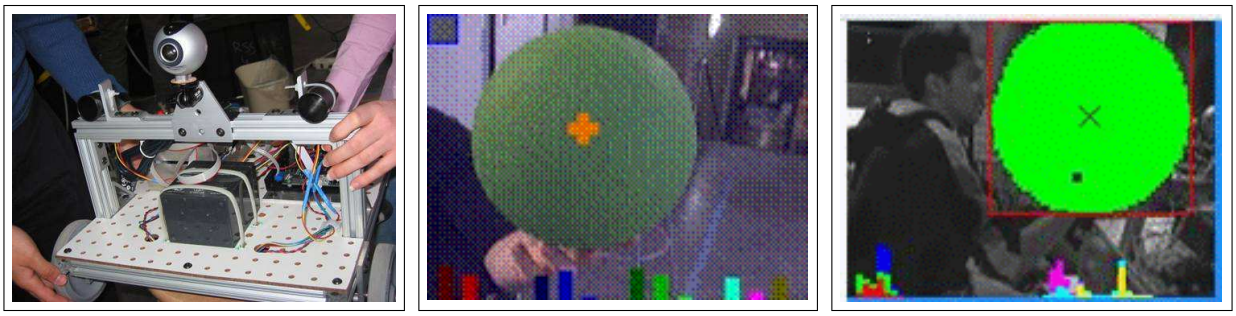

Figure 4: The students are issued with the camera shown mounted on the crossbar (left panel) for more sophisticated environmental sensing in the fifth lab. The camera is used to track a ball of a specified colour (middle panel). The students extract features such as blob size and location (right panel) in order to servo the robot to keep the ball in view and at a fixed stand-off distance.

commands to keep the ball centred in the image frame and at a specific distance away. The students again encounter the problems of sensor calibration and learn about different colour spaces.

6. Local Navigation and Environmental Modelling

   The sixth lab introduces the students both to additional sensors and to the concepts of a persistent environmental map. The students are issued with bump sensors and also sonar range sensors. The requirements is for the robot to detect an obstacle using the bump sensors, and then drive around the obstacle, building a map using the sonar sensors. An example map is shown figure 5. The sonar data is useful for range sensing but extremely noisy, and the students must learn more issues in dealing with real-world sensing.

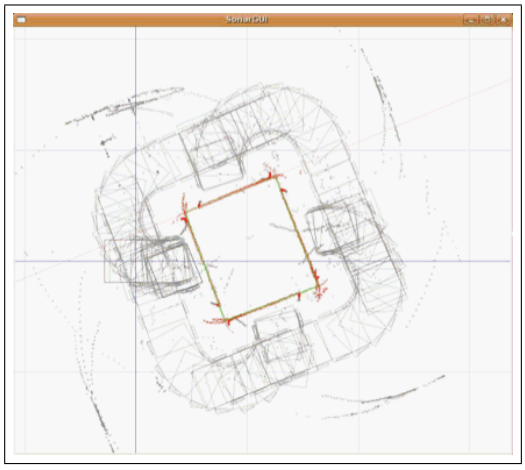

Figure 5: An example map of a square obstacle learned from sonar data during the sixth lab. The grey dots are the individual sonar returns (notice that substantial noise in the data), and the fitted green lines are the sides of the square obstacle. *Image courtesy of Silvia Baptista, Mark Vayngrib, Kevin Wang, Tina Wright.*

7. Motion Planning and Global Navigation

   The seventh lab introduces the students to motion planning and deliberate action. The students are issued with a map of a maze environment and must implement a simple planner. The students learn issues of representation (e.g., different ways of representing the search problem) and must also consider computational efficiency in the design and implementation of their algorithms.

8. Grasping and Object Transport

   The final lab introduces the students to a very different capability. They are issued with components of a robot arm and gripper shown in figure 6, which they must assemble and then control in order to grasp and pick up blocks. Additionally, the students are expected to use the camera to identify blocks for grasping, and ensure that the robot's gripper is within range of the block. As a result, this lab requires the students integrate their solution to the visual servoing lab from earlier in the semester with the grasping solution. By introducing the students to the integration of different subsystems into a single capability, we lay the ground work for the final month of the course.

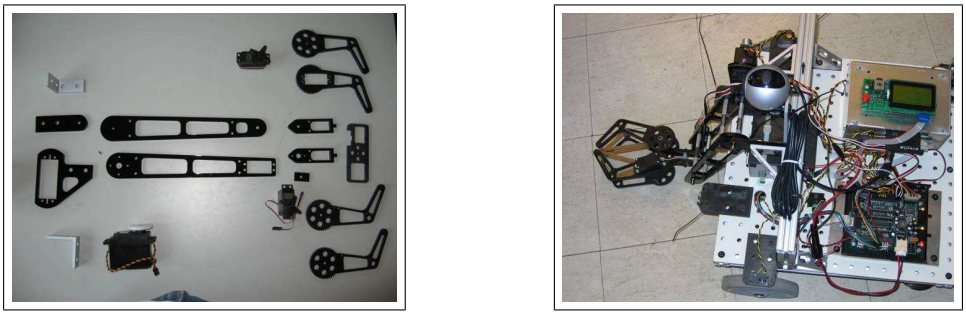

Figure 6: (Left) The robot components issued in the eighth laboratory that constitute a robot arm and gripper. The students mount the arm on the robot (right) to implement mobile manipulation algorithms.

3.2. Course Challenge

The last month of the course is the "Course Challenge", which to date has been to build a shelter on Mars. The objective is for the robot to explore, gather materials and build a structure in a dynamic partially-known environment. The robot will be given a partially specified map of this space containing obstacles and blocks. However, the environment will have dynamic obstacles whose location and behavior will not be known to the robot. The robot will have to pick a construction site (home, or at a given place, or by reasoning), identify as many building blocks as possible, bring them to construction site and build as much of a wall structure as possible at the site.

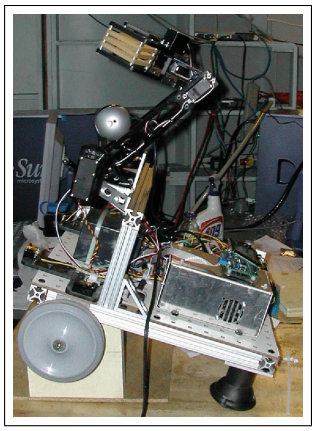

Figure 7: The complete robot at the end of the last lab.

At this point in the semester, the students should have a complete robot, such as the one shown in figure 7, with working implementations of all the necessary software subsystems to complete the challenge. The task itself is very much abstracted from a true robotic construction task, and could be completed using relatively unintelligent behaviour. However, the students are expected to leverage the skills and implementations from the preceding labs in order to implement a robot that operates in a deliberative manner. We have stressed the elegance of design and implementation as much as completion of the specific task.

We also encourage the students to think about the integrated hardware-software design problem. Although the course is software-intensive, some aspects of the course challenge can be simplified or addressed more easily through innovative hardware. In order to limit the amount of time that can be sunk into unstructured hardware modifications, we restrict the students to modifications that cost no more than $50. However, some innovative designs have emerged, including a beautiful hardware assembly that uses the shape of the building blocks to automatically form a structure as each block is collected. Some example hardware modifications are shown in figure 8

4. R:SS II

The second semester course, Robotics: Science and Systems II, is designed to allow the students to explore a topic in mobile autonomy in more depth. Building on the first semester, that students have built a relatively small robot that explores, gathers and builds a structure in a dynamic partially-known environment. This implementation requires understanding of the overall issues of robot control, visual servoing, motion planning, position estimation and manipulation in the laboratory setting, in a moderately controlled environment.

In the second semester, the class as a group scales this task up to something that approximates a real-world challenge task that we term a “Grand Challenge” problem, in the style of the DARPA

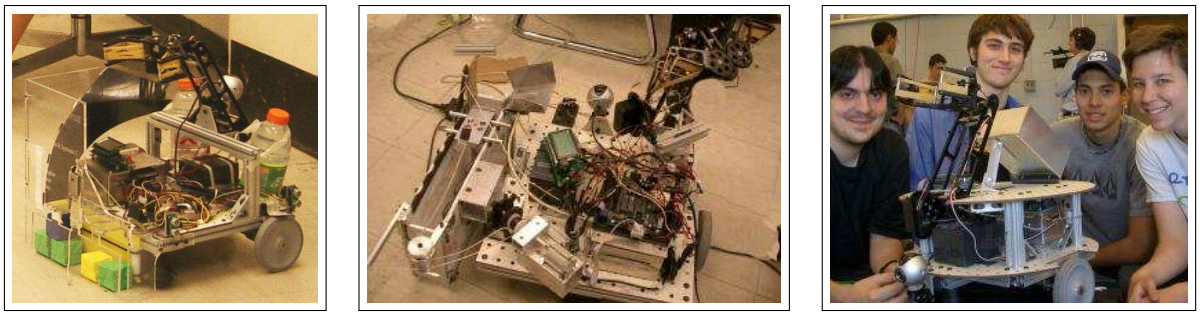

Figure 8: Some example hardware modifications that the students make to the basic frame we provide. On the left, the robot has a collection and release mechanism that automatically organizes the robots into a structure. In the middle, the robot has an entirely different collecion and release mechanism. On the right, the robot has a simple "dumping" mechanism, requiring the robot to place blocks together once collected. However, the robot frame has been replaced with a circle, allowing easier motion planning.

Grand Challenge. By scaling the problem up to real-world size, we ask the students to address the challenges of unstructured, outdoor environments, unknown world models, real-world navigation and locomotion, and real-world manipulation. These are all largely open research topics in the literature, and our goal is to get the class working at the cutting edge of robotics research, and hopefully generate publishable contributions to the state of the art. As in R:SS I, the course has a formal set of learning objectives, but we generally expect at the end of the course, that

1. the students will be able to critically evaluate different choices of subsystem designs and implementations;
2. the students will use their knowledge of autonomous control, planning or estimation to design and implement a new capability for an autonomous system;
3. and will integrate their new implementation into the complete system.

In addition to the research challenges in autonomy, the class must address the engineering challenges of developing in collaborating teams, developing across multiple platforms, and developing software that is reliable enough for others to use. The problem is too large to be tackled by small teams of 3-4 students; the class as a whole works on a single system. We provide a much larger, pre-built robot base with substantially more capabilities (i.e., larger arm, laser range finders, GPS, etc.).

4.1. Grand Challenge

The specific "Grand Challenge" problem is a large-scale remote autonomous construction problem. The robot should, over a long period of time, collect building components located on campus,

return them to the hangar, and and assemble as large a structure as possible in the hangar. The robot must have capabilities to do the following:

- navigate in a totally unknown environment
- locate good construction objects
- identify the location of the construction site
- retrieve, carry the objects to and place the components at the construction site
- create a structure at the construction site.

4.2. Lectures and Labs

Our approach in the R:SS II is to allow the students to self-organize. We impose little structure on the labs. We provide one lecture per week in a topic of the students choosing (although we provide an initial set of lectures to begin the semester), with the expectation that the student will request more instruction in different areas. During lab sessions, the students working on individual subsystems meet one-on-one with a faculty advisor. At the end of each week we hold design reviews, where a student from each subsystem group presents to the class the current state of the system and expected plans for the next week.

The class immediately is faced with not only a design problem but also an organization problem, and and must produce an architecture, design and engineering plan for their autonomous system within the first month of the course. The only additional structure we impose is a series of deliverables throughout the semester tied to capabilities of the robot, such as demonstrating navigation, demonstrating mobile manipulation, demonstrating exploration, etc.. Between the required deliverables, we encourage the students to set their own deadlines but do not require this.

5. Overall Lessons Learned

5.1. Successes

The major success of the course has been the student performance. Despite the fairly demanding pace of the first semester labs, we have a very high success rate in student teams completing all parts of the lab. We have never had more than one team in a semester (out of 6-8 teams per semester) not start the course challenge with a complete and working solution to each subsystem. This has been satisfying, because many of our students have had limited programming experience or limited experience with embedded systems. As a result, we had initial concerns about the ability of the students to understand and implement many of the concepts. Additionally, student feedback

indicates a general satisfaction with the coverage of the material. The students feel that they are getting a broad exposure to a range of issues without being overwhelmed.

We also feel that our efforts in team management and team balance issues have been a success. We draw students from three main departments: Electrical Engineering and Computer Science (EECS), Aeronautics and Astronautics (Aero/Astro) and Mechanical Engineering (MechE). At the beginning of each semester, we ask the students to complete questionnaires outlining their background and skill set. We then create teams by combining students with complementary backgrounds, for example, pairing a computer scientist strong in programming with a mechanical engineering and Aero/Astro student who are strong in control and systems engineering. In R:SS I, we also encourage students to work in areas that our outside their comfort zone, such as asking the non-computer-scientists to take on the bulk of the implementation initially, while encouraging the natural programmers to pick up the soldering iron. This has generally worked extremely well, and we have had a surprisingly small number of dysfunctional student teams. In the one case of a dysfunctional team, we have acted to re-organize the teams and this appeared to solve the teaming issues quickly. Additionally, the students report that they liked "the freedom", and the "actual engineering [practices] that were involved" [1]. However, as we discuss below, the teaching of organization and engineering practices in the R:SS II class will be improved.

Finally, we emphasize that there are rarely right answers to the design questions. To reinforce this point, the students are required to debate different design decisions, such as deliberative planning compared to reactive planning. Our experience has been that the students generally begin the debate sequence unsure of the point of the exercise, but quickly learn that the literature typically contains multiple, opposing views on most issues and that engineering decisions are often the subject for discussion. We have been very pleased at some of the student discussions that have resulted from the debates.

However, not every aspect of the course has been a complete success. In the following sections we list some specific lessons that we have learned in developing this course.

The importance of structure Firstly, while we provide the students with substantial instruction in systems development practices, we have imposed relatively little structure on the process itself, allowing the students to choose how to organize themselves. Concerns that the students would not understand the trade-off between organizational freedom and risk of failure have not been realized; post-hoc evaluation suggests that the students appreciate the substantial amount of freedom to manage the system development themselves. However, Tom Clay & Associaties provided an independent evaluation of the first instance of R:SS II. The Clay report [1] listed the following student concerns regarding the teaming freedom:

- "They worked ineffectively with teams."
- "They worked ineffectively across teams."
- "They did not develop sufficient or timely processes."

- "They assigned or took on responsibilities in ways that did not support individuals' learning."

In order to address these concerns, we have invited additional faculty with experience in working with large teams to lecture on engineering management practices.

The spiral development process We pursued a spiral development evaluation process throughout the course. However, both faculty and students re-learned the lesson that spiral development is essential across the entire system, not only within specific capabilities. Milestones were set on a per-capability basis, rather than on a complete-system performance basis, allowing teams to progress at different rates. A key insight (that has been learned elsewhere in the past) is that for software projects, no development within a capability should be permitted until the complete system is at the same readiness level.

The Clay report contained the suggestion of continuing the structure of R:SS I into R:SS II, that is structuring the course around a series of goals, instead of around a single "Grand Challenge" end-goal. This would encourage the spiral development model, and would also allow the students to fall short of the end goals without feeling like they had failed.

The students also felt (as reported in the final debriefing) that they were unable to get a sense of ownership of the system as a whole: they suggested making sure the teams were much more loosely organized, in order to allow them to get a sense of how to use all sub-components and would lead to faster testing and integration. This loose organization is probably unwieldy give the amount of development required, however, we have recognized the need to provide a strong sense of ownership of different components while building a larger system.

The important of communication deliverables Finally, a major component of the systems engineering management plan in early version of R:SS I required the students to present design reviews to each other regularly. The general upward trend in the presentation grades indicated that students were learning from seeing each other present. While the design reviews were not sufficient to catch all development errors, we consider the reviews to be successful in that most design or implementation errors were identified by the students themselves, rather than faculty advisors. We believe the in-class peer-to-peer design reviews were ultimately more useful than conventional presentations to external faculty reviewers would have been.

We have, however, introduced the presentation component into R:SS I as well, and students now brief the faculty after each lab is complete. This has greatly improved both the quality of the lab analysis and also the student timeliness of completing the labs. We have also begun working with the Writing Program to ensure that the communications deliverables are of high quality not only technically but also from a writing and communication standpoint.

6. Conclusion

This paper presented the first two iterations of a two-semester sequence of courses that introduces students from Aeronautics and Astronautics, Electrical Engineering and Computer Science, and Mechanical Engineering to the process of conceiving, designing, implementing and operating a mixed hardware-software system.

Acknowledgments

This course was funded by the School of Engineering at MIT. Additional support was provided by the d'Arbeloff foundation, Sun Microsystems and Intel Corporation. Their support is gratefully acknowledged.

References

[1] Tom Clay. Assessment of RSS-II. Technical report, Tom Clay and Associates, 2005.

Biographical Information

Nicholas Roy is the Boeing Assistant Professor in the Department of Aeronautics Astronautics at the Massachusetts Institute of Technology. He received his Ph. D. in Robotics from Carnegie Mellon University, Pittsburgh in 2003. He is a member of both the Autonomous Systems Laboratory and a member of the Computer Science and Artificial Intelligence Laboratory (CSAIL) at MIT. His research interests include autonomous systems, mobile robotics, human-computer interaction, decision-making under uncertainty and machine learning.

John J. Leonard is Associate Professor of Mechanical and Ocean Engineering at MIT and a member of the MIT Computer Science and Artificial Intelligence Laboratory (CSAIL). His research addresses the problems of navigation and mapping for autonomous mobile robots. He holds the degrees of B.S.E.E. in Electrical Engineering and Science from the University of Pennsylvania (1987) and D.Phil. in Engineering Science from the University of Oxford (formally 1994). He studied at Oxford under a Thouron Fellowship and Research Assistantship funded by the ESPRIT program of the European Community. Prof. Leonard joined the MIT faculty in 1996, after five years as a Post-Doctoral Fellow and Research Scientist in the MIT Sea Grant Autonomous Underwater Vehicle (AUV) Laboratory. He has participated in numerous field deployments of AUVs, including under-ice operations in the Arctic and several major experiments in the Mediterranean. He has served an associate editor of the IEEE Journal of Oceanic Engineering and of the IEEE

Transactions on Robotics and Automation. He is the recipient of an NSF Career Award (1998), an E.T.S. Walton Visitor Award from Science Foundation Ireland (2004), and the King-Sun Fu Memorial Best Transactions on Robotics Paper Award (2006).

Dr. Una-May O'Reilly is a principal research scientist at the Computer Science and Artificial Intelligence Lab, MIT. She leads EVO-DesignOpt, a group which focuses on the development and application of evolutionary algorithms in tandem with convex optimization and machine learning techniques. She is co-editor of Genetic Programming Theory and Practice II. She serves as associate editor of the journal Genetic Programming and Evolvable Machines and editor for the journal Evolutionary Computation. Dr. O'Reilly is a fellow of the International Society of Genetic and Evolutionary Computation.

Daniela Rus is professor in the EECS Department at MIT and a member of CSAIL at MIT. Previously, she was a professor in the computer science department at Dartmouth College. She holds a PhD degree in computer science from Cornell University. Her research interests include distributed robotics, mobile computing, and self-organization. She was the recipient of an NSF Career award and an Alfred P. Sloan Foundation Fellow. She is a class of 2002 MacArthur Fellow.

Seth Teller obtained a Ph.D. from U.C. Berkeley in 1992, focusing on accelerated rendering of complex architectural environments. After post-doctoral research at the Computer Science Institute of the Hebrew University of Jerusalem Institute of Computer Science, and Princeton University's Computer Science Department, in 1994 he joined MIT's Computer Science and Artificial Intelligence Laboratory, and Electrical Engineering and Computer Sciences Department, where he is now a Professor of Computer Science and Engineering and co-head of the Robotics, Vision and Sensor Networks group. Teller's research focuses on sensor networks, human-computer interaction, machine vision, and mobile robotics. He is the Perception Team lead for MIT's 2007 DARPA Grand Challenge team, which is developing a passenger vehicle capable of safe, autonomous driving in an urban environment.

Corresponding Author

Nicholas Roy
Massachusetts Institute of Technology
77 Massachusetts Ave., 33-315
Cambridge, MA
02139
nickroy@mit.edu