In: *Computers in Education (Special Issue on Novel Approaches to Robotics Education)*, 20(3):50–59, 2010.

Teaching Robot Motion Planning

Mark Moll[1], Janice Bordeaux[2] and Lydia E. Kavraki[1]
[1] Department of Computer Science, Rice University, Houston, TX 77005, USA
[2] George R. Brown School of Engineering, Rice University, Houston, TX 77005, USA
{mmoll,jbordeau,kavraki}@rice.edu

Abstract

Robot motion planning is a fairly intuitive and engaging topic, yet it is difficult to teach. The material is taught in undergraduate and graduate robotics classes in computer science, electrical engineering, mechanical engineering and aeronautical engineering, but at an abstract level. Deep learning could be achieved by having students implement and test different motion planning strategies. However, a full implementation of motion planning algorithms by undergraduates is practically impossible in the context of a single class, even by students proficient in programming. By helping undergraduates grasp motion planning concepts in series of courses designed for increasing advanced levels, we can open the field to young and enthusiastic talent. This cannot be done by asking students to implement motion planning algorithms from scratch or access thousands of lines of code and just figure out how things work. We present an ongoing project to develop microworld software and a modeling curriculum that supports undergraduate acquisition of motion planning knowledge and tool use by computer science and engineering students.

1 Introduction

Robots have fascinated people for generations. Today, robots are built for applications as diverse as exploring Mars and the Earth's deep seas, demining war zones, cleaning toxic waste, assembling cars, inspecting pipes in industrial plants, driving autonomously in off-road and urban environments, and mowing lawns. Robots are also interacting with humans in a variety of ways: robots are museum guides, robot pets entertain, and robots assist surgeons in life threatening operations. By integrating science, technology, engineering, and mathematics (STEM disciplines) in a unique way, the field of robotics studies both the design of new mechanisms and the algorithms and frameworks that make robots useful in the physical world.

This project aims to provide software tools and applications that will fill some of the existing curricular gaps in college level robotics education. Robotics is not exploited for its fascinating real-world examples and applications in STEM disciplines. In many high-schools, robotics competitions are used to draw students into college and STEM disciplines. Each year the FIRST competitions[1] attract hundreds of students from grades 1 to 12 who build robots and compete in well-defined contests. Lego MindStorms [1] have provided another appealing entry to the basics of robotics in high school and undergraduate education. Students can quickly assemble a robot using the basic Lego bricks and special bricks for sensors and actuators. Also, the robot can be programmed through a regular PC. Although students spend time calibrating the robots and repeat the design-test-modify loop many times to achieve a result [2], there is no doubt that such efforts have attracted many students and particularly women to science and engineering. At the freshman level in colleges, robotics is used to attract students to STEM fields. Because of the interdisciplinary

[1] FIRST (For Inspiration and Recognition of Science and Technology) http://www.usfirst.org/who/default.aspx?id=34

nature of robotics, but also the excitement of building a robot that works and does something useful, colleges across the country strive to have a design class on robotics to convince students that STEM disciplines are challenging and fun.

A Gap in College-Level Robotics Education Exists After an initial intro-level course on robotics, students typically have to wait till their senior year to take another robotics course. As a result senior courses are extremely demanding. The reasons are multiple. Robotics offered at the senior or beginning graduate level tend to include many topics in an effort to teach the students as much as possible about an extremely rich subject. Typically, course instructors are faced with the difficult task of covering breadth and depth while keeping the class manageable and exciting. Robotics textbooks [3–5] cover many topics and challenge the student and the instructor to keep up with the material.

Goals of this Project This project concentrates on one key concept in robotics: motion planning. Motion planning is a central topic in robotics and deals with finding feasible collision-free paths that take a robot from an initial to a final state. Motion capabilities are intrinsically related to robots and tend to occupy a significant part of upper-level robotics classes. Due to significant advances over the last decade, motion planning is a very mature field [3–5]. Of the two most recent textbook in robotics, one concentrated entirely on motion planning [5] and it occupied almost half of the other book [3]. However, teaching motion planning is difficult, in part because a considerable mathematical and algorithmic background is needed to comprehend critical ideas. As a result, either the module is over by the time students are ready for the hands-on experience, or students lack the the programming background to implement an interesting project.

The project described in this paper aims to fill the gap in robotics curriculum by developing a teaching module that includes an extensible software tool and related assignments. Also, we aim to engage a broad spectrum of faculty and students, and develop a community focused on motion planning learning. The effectiveness of our motion planning curriculum for diverse learners will be assessed at Rice University and other partner institutions. The teaching module's software tool described in this paper has an easy-to-use interface and mitigates many of the problems mentioned above. A set of assignments is developed, implemented and assessed at Rice University and other interested partner institutions. A feedback loop will be implemented to improve the curriculum so that it can be used by other institutions. Also, we aim to engage a broad spectrum of faculty and students and promote the growth of a user community that will make the efforts self-sustainable in the long run.

2 Background

A Brief History of Motion Planning The general motion planning problem has been studied extensively in the past. The classical case is the general mover's problem, which is posed for polyhedral robots operating in a polyhedral workspace. This problem was shown to be PSPACE-complete in [6]. Advances in robotics and modeling pressures the development of practical algorithms for robots with many degrees of freedom. A new generation of sampling-based motion planning algorithms emerged that preprocesses a robot's configuration space to construct a graph representation, which approximates the connectivity of the configuration space. See figure 1 for a high-level overview of sampling-based planning algorithms. The predominant family of these algorithms was pioneered with the introduction of the Probabilistic Roadmap Method (PRM) [7]. PRM techniques make some sacrifices over exact (exponential-time) algorithms. For example, path non-existence cannot be proven using PRM. However, a weaker completeness result has been proven: if a path exists then a PRM planner will eventually find it [8].

PRM provides a good solution to a broad class of motion planning problems where the robot operates in a known environment for a good deal of time. Many times one is interested in getting to the goal as soon as possible without exploring

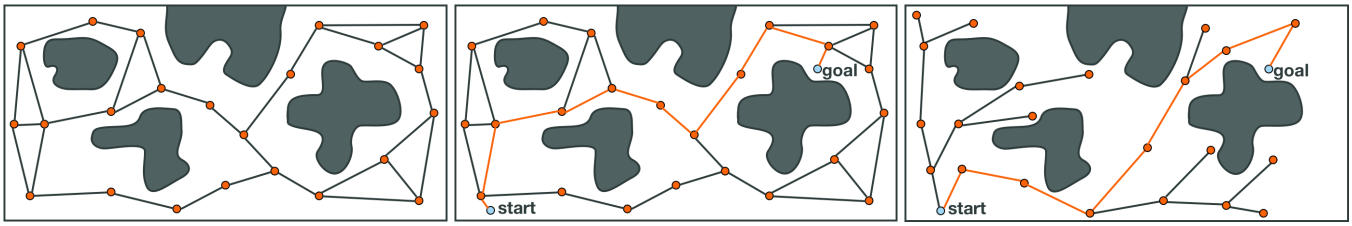

Figure 1: **Sampling-based motion planning.** Each red point represents a robot configuration, while the dark gray areas represent 'forbidden' areas corresponding to, e.g., collisions. The dimensionality of a robot's configuration space is typically much larger than the two shown here (a free moving 3D rigid body already has a 6-dimensional configuration space: 3 for translation and 3 for rotation.) *Left:* configurations are sampled, checked for collisions, and connected to nearby other configurations when possible. This results in a graph called the *roadmap*. *Middle:* to find a path between two goal configurations, they are first connected to the roadmap, and a path is found through a graph search. *Right:* tree-based algorithms "grow" a tree from the start configuration by iteratively adding tree nodes and edges, often biasing the tree growth towards the goal.

the environment. This led to the development of tree-based planners [9–11]. Tree-based planners operate by constructing a tree in the configuration space. Tree-based methods have some advantages over a roadmap method: they concentrate on one or two connected components of the space and primarily connect configurations by integrating the controls of the system. This novel way of generating configurations and paths is critical in planning for systems with motion constraints [9, 12, 13].

Recent Books In the last few years several robotics textbooks have been published [3, 5, 14]. These books are currently used in upper level undergraduate classes or introductory graduate level classes. The first two textbooks cover motion planning extensively. The material is also covered in the Handbook of Discrete and Computational Geometry [15], the Handbook of Robotics [16], and a recent review article [17].

Available Software If students had to write motion planning algorithm implementations from scratch, much of their time would be spent on writing the low-level data structures and not on understanding the higher-level algorithm. There exist a few motion planning software packages such the Motion Strategies Library (MSL) [18], the Motion Planning Kit (MPK) [19], and OpenRAVE [20], but they contain only a limited number of algorithms or have not been developed for several years (or both). KineoWorks [21] provides commercial motion planning software for academic research and industrial applications. Recently, a new motion planning package called Vizmo++ [22] was introduced for possible educational use, but it appears to have been abandoned (source code or binaries of the software were never released). In 2007, our group released the Object-Oriented Programming System for Motion Planning (OOPSMP) [23]. Although initially released as a research tool, through continued development it has become viable to use as a teaching tool as well.

3 Development of Motion Planning Software

3.1 Software Overview

The software used for the teaching module is based on what is currently offered by the Object-Oriented Programming System for Motion Planning (OOPSMP, http://oopsmp.kavrakilab.org). It is a software system that provides easy access to many state-of-the-art sampling-based motion planning algorithms and underlying data structures. Many motion planning algorithms share many similar components, and OOPSMP allows students to quickly design their own planner by combining existing components. We have spent considerable effort in adapting OOPSMP for educational purposes. This is an ongoing process. We will first give an overview of existing functionality:

Table 1: **Motion planning algorithm components.** Variants of motion planning algorithms can be composed by choosing different options in each column.

sampling strategy:	**connection strategy:**	**path generation:**	**state space:**
• uniform	• random nearest neighbors	• geodesics ("straight lines")	• rotation+translation 2D
• Gaussian	• approx. nearest neighbors	• best or random control	• rotation+translation 3D
• obstacle-based	• exact nearest neighbors	• approx. steer	• controller: car, diff. drive, unicycle
• bridge test		• collision check: incremental, subdivision	

- OOPSMP contains implementations of popular motion planning algorithms such as Probabilistic Roadmap Methods (PRM) [7], Rapidly-exploring Random Trees (RRT) [13], Expansive Spaces Trees (EST) [24], bi-directional tree planners, and others. There are different sampling strategies (uniform, Gaussian [25], obstacle-based [26], bridge test [27]) that can be used with any of the motion planning strategies.
- Each algorithm can produce solutions for problems involving one or multiple robots. Each robot can rotate and translate, but a user can also impose constraints on the kinematics and dynamics that restrict the possible motions. OOPSMP implements several kinodynamic models of cars, differential drives, and unicycles, and makes it easy to add new models of different robotic systems.
- OOPSMP also contains many general purpose functions and data structures for:
  Linear algebra: low-dimensional vector and matrix operations.
  Topology: rigid body topology in 2D and 3D, such as SO(2) and SE(3) (rotations in 3D as quaternions).
  Numerical integration: several fixed-step and adaptive-step methods up to 8th order.
  Data structures and algorithms: sets, disjoint sets, update-able heaps, graphs, depth-first search, breadth-first search, Dijkstra's shortest path, A$*$ search.
  Proximity algorithms: nearest neighbors for metric spaces.
- OOPSMP can be run as command line tool or run interactively with a simple graphical interface. Figure 2 shows some example problems in 2D and 3D, along with solution paths. It runs on Microsoft Windows, Linux, and Mac OS X.

OOPSMP implements motion planners in a modular, and objected-oriented fashion. This is done in a plug-and-play fashion: users can easily create new components as plugins and have their functionality immediately available at runtime without having to recompile any part of OOPSMP. Table 1 gives an overview of the common components of a motion planning algorithm and which options are available for each component in OOPSMP. Each motion-planning component (data structures and low-level algorithms) can be combined and plugged into each motion planner giving rise to a combinatorially large number of possibilities. This provides a great degree of flexibility in evaluating the impact of different components and comparing different motion planners.

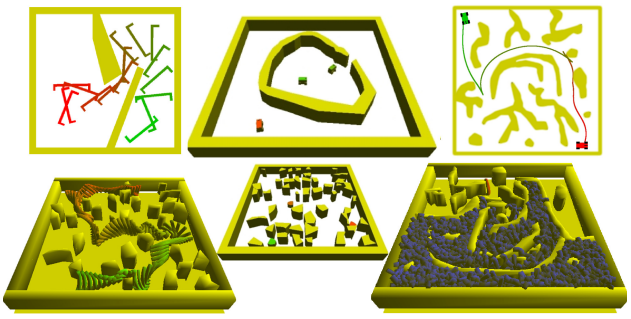

Figure 2: Screenshots of the graphical version of OOPSMP.

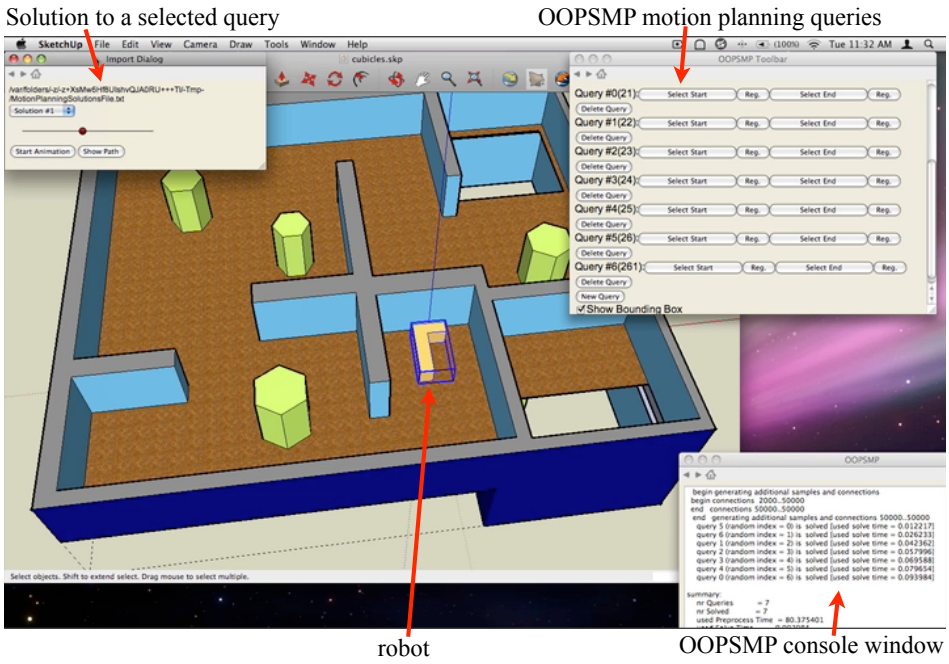

Figure 3: The SketchUp interface to the OOPSMP software.

3.2 Education-Focused Software Development

We have developed a new graphical user interface to OOPSMP that relies on Google Sketchup, a popular, free 3D modeling program. This makes the tool more accessible to students with little programming experience, but—just as importantly—it also allows students to explore the high-level behavior of different motion planning algorithms on different problems without getting bogged down in writing low-level code. Through a plugin for SketchUp a student can define and solve motion planning programs using OOPSMP. The plugin generates the input files for OOPSMP, runs the OOPSMP program in the background, and reads back the output. Instead of running OOPSMP inside SketchUp, one can also export motion planning problems and run the OOPSMP program outside of SketchUp. This is useful for more complex problems that may take a long time to solve. The SketchUp plugin makes it much easier to try out different planning algorithms and see the effect of changing algorithm parameters. Figure 3 shows the interface of the SketchUp plugin. For a "robot" in the form of a free-flying L-shaped block several motion planning queries have been defined. Each query consists of a start and goal pose. After OOPSMP has been called from SketchUp to solve the queries, the solutions can be shown in animation. SketchUp provides access to the online Google 3D Warehouse, a repository where people around the world deposit geometric models of everything ranging from simple household objects to realistic car models to complete buildings. This makes it very simple to create realistic environments and robots to test motion planning algorithms. The significance of this is not to be underestimated. In the past, many motion planning algorithms could only be demonstrated on simple toy examples, because it would take an inordinate amount of effort to create complex environments.

4 Motion Planning Curriculum Development

We have developed a series of assignments that gradually introduce students to motion planning algorithms. The lecture material necessary to complete these assignments is described in at least two robotics textbooks currently on the market [3, 5]. The assignments will be given to students taking robotics courses at Rice. Formal assessment will be done and feedback will be solicited from the in-

structors and the students to continuously improve the assignments which will use OOPSMP. Since the courses will be taught several times, laying the basis for correct assessment is important for the success of this project.

With the enhancements described above, it will be possible to cover a broad range of problems in assignments. For instance, in a robotics class that is more tailored towards mechanical engineering students, example exercises can be used that emphasize control aspects of motion planning. For computer science students assignments can focus on the complexity of motion planning for high-dimensional systems.

4.1 Assignments

The assignments are set up so that each assignment builds on the previous assignment, but without directly depending on it. Furthermore, for each assignment we have developed several versions that vary in depth. By varying the depth of each assignment and the number of assignments instructors can adjust the assignments to the level of the students and the amount of time they want to allocate to the motion planning module in their robotics class. We have initially divided the assignments into four initial categories:

Getting familiar with motion planning algorithms The students will use the self-paced tutorials from the OOPSMP handbook to get familiar with the basic workflow of solving motion planning problems: (a) define an environment, (b) define a robot, (c) define motion planning queries by specifying pairs of start/goal configurations, (d) select a motion planning algorithm and its parameters, and (e) solve the problem by running the algorithm. This assignment can be completed using the SketchUp interface to OOPSMP.

Develop a 'naïve' motion planning algorithm In these assignments students have to write some code that implements a simplified version of one of the algorithms contained in OOPSMP. In one assignment a code template is given and students have to "fill in the blanks." This will teach students the basic code structure. OOPSMP consists of thousands of lines of C++ code. This can be overwhelming at first, so an assignment that illustrates which classes need to be implemented and how they are connected is useful. A subsequent assignment will ask the students to implement a variant of an algorithm that already exists in OOPSMP. An important part of this assignment is the performance evaluation of an algorithm. It is not always easy to decide which algorithm is better. Run-time matters, but the number of vertices and edges created in the roadmap or tree are also important since they can indicate how the algorithm performance would scale to more complex problems. Many motion planning algorithms rely on random sampling, so runs need to be repeated to obtain a reliable estimate of expected performance. Another complication in performance measurement is that there really is no standard benchmark problem. Any algorithm needs to be evaluated on a set of various benchmarks to get an understanding of its strengths and weaknesses.

Comparison of algorithms The comparison of motion planning algorithms requires critical thinking. Students learn that determining which algorithm is 'better' is non-trivial. Besides run-time, many other performance metrics are important, such as the number of collision checks or the memory use. The selection of good test cases is also critical. Considering both performance metrics and problem selection will help students generalize specific results and evaluate which algorithms are more likely to scale up to more complex systems.

Replace a core component with an alternative implementation This assignment will help students acquire a deeper understanding of the nuts and bolts of motion planning. At Rice University, where OOPSMP has already been used in a robotics class taught by Kavraki, students could choose projects such as: (a) implement a better collision checker, (b) write a different proximity data structure, or (c) implement a new motion planning algorithm that is not a simple variant of what is available in OOPSMP.

Open-ended projects At Rice, we have also developed several ideas for projects that can be implemented at varying levels of difficulty. Examples include path optimization (of, e.g., path length, smoothness, energy consumption, etc.), path clustering, dynamic manipulation with a planar manipulator (with kinodynamic constraints and possibly under-actuated joints), and formation planning.

4.2 Documentation

To complete the assignments students need documentation that explains in detail what functionality is available and how to use it. Currently, the documentation consists of a high-level description of the code structure, basic usage instructions as well as a large collection of linked web pages that are automatically extracted from the extensive comments in the code. As part of this project we will develop a comprehensive reference manual that provides the background and conceptual framework that undergraduates for a robotics course are expected to have. To get started with OOPSMP, though, a reference manual is not always what the students need most. Initially, they will need a handbook that explains at a high level how all the pieces of the program fit together and includes tutorials with worked-out examples and screencasts. Such a handbook is virtually nonexistent at this point. We will also link the lecture material to assignments by inserting references to a robotics textbook [3] in all documentation. This book is used in many robotics classes around the country (see the web site for the book at http://motionplanning.com).

We have a developed a tutorial for OOPSMP that consists of slides, movies, and code. The tutorial materials are available online at http://www.kavrakilab.org/OOPSMPtutorial. The tutorial was held in September 2008 at the IEEE/RSJ International Conference on Intelligent Robots and Systems, one of the major conferences in robotics. Faculty can use the tutorial as-is, or just use selected material in their own slides.

4.3 Assessment

Systematic, formal outcomes assessment will be conducted to evaluate the effectiveness of the technical, curricular and community building activities. Both formative and summative evaluation methods are included. For example, the new technology will be pilot-tested with undergraduates and the feedback used to inform design decisions on critical components and functionality. Also, evidence of student learning outcomes will be collected at Rice University and other partner institutions on an ongoing basis, strengths and weakness identified, and the results used to improve curriculum materials and teaching methods. In contrast, the project's success will be determined at the end of the study by evaluating the overall level of achievement attained by student participants. In addition, student and instructor surveys designed to capture quantitative data (e.g., the amount of independent exploration) and qualitative data (e.g., student suggestions for ways to improve the curriculum or technology) will augment measures of student learning. An independent expert will be hired 1–2 weeks each year to analyze and help evaluate the results.

5 Dissemination

Our work can be used to broaden the dialog between students and faculty and beyond institutional boundaries by creating a worldwide community of OOPSMP users. In the long run, we aim for the OOPSMP project to be self-sustaining. Over 200 people from around the world have already registered as OOPSMP users. We currently have an official mailing list for users to discuss OOPSMP. This is mostly used for announcements. Since many people are already inundated with email these days, we therefore plan to make official announcements available through both a blog/RSS feed and the mailing list.

Our work can also be used to form partnerships with faculty members who are planning to use our teaching module in their classes. In exchange for faculty and student participation in surveys, we

would be able to assist in the design and selection of assignments tailored for the specific needs of a class. Anyone can download and use the teaching module without this additional overhead. We will keep track of the number of instructors including this module in their courses. The growth and engagement of the OOPSMP user community into a collaborative educational resource will be tracked through the project's blog/RSS feed and email list. User community responses to a brief online user registration form and basic web analytics will document community size and user characteristics, along with the development of community collaborative activities.

6 Conclusion

This project aspires to transform how college students learn robotics by offering a motion planning curriculum that enhances deep learning and is supported by an integrated software environment. Students are challenged to work on real-world robotics problems, and develop deeper knowledge by reflecting on and formally evaluating their results. We scaffold learning by freeing students of tedious details and heavy programming and help them to develop critical thinking within and outside robotics through a hands-on problem-based learning approach. In the long run, the products of this project can strengthen other courses in computer science and engineering, and motivate younger students to pursue STEM careers. The work will be widely disseminated through the National Science Digital Library (NSDL).

Acknowledgements

This project is funded by NSF CCLI 0920721. The authors are indebted to Kostas Bekris, Erion Plaku and the other members of the Kavraki Lab for developing the initial version of OOPSMP. They are also grateful for the work done by Nick Bridle and Nicholas Feltman that resulted in the OOPSMP plugin for Google SketchUp.

References

[1] Lego mindstorms. URL http://mindstorms.lego.com.

[2] M. McNally, M. Goldweber, B. Fagin, and F. Klassner. Do Lego MindStorms robots have a future in CS education? In *Proceedings of the 37th SIGCSE technical symposium on Computer Science Education*, pp. 61–62. ACM New York, NY, USA, 2006.

[3] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, 2005.

[4] J.-C. Latombe. *Robot Motion Planning*, chapter 7, pp. 295–353. Kluwer, Dordrecht; Boston, 1991.

[5] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[6] J. Canny. *The Complexity of Robot Motion Planning*. MIT Press, Cambridge, MA, 1987.

[7] L. E. Kavraki, P. Švestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. on Robotics and Automation*, 12(4):566–580, August 1996.

[8] A. Ladd and L. E. Kavraki. Generalizing the analysis of PRM. In *IEEE Intl. Conf. on Robotics and Automation*, volume 2, pp. 2120–2125, 2002.

[9] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock. Randomized kinodynamic planning with moving obstacles. *Intl. J. of Robotics Research*, 21(3):233–255, March 2002.

[10] J. Kuffner and S. M. LaValle. RRT-connect: An efficient approach to single-query path planning. In *Proc. 2000 IEEE Intl. Conf. on Robotics and Automation*, pp. 995–1001, San Francisco, CA, April 2000.

[11] G. Sánchez and J.-C. Latombe. A single-query bi-directional probabilistic roadmap planner with lazy collision checking. *Intl. J. of Robotics Research*, pp. 403–407, 2003.

[12] R. Kindel, D. Hsu, J.-C. Latombe, and

S. Rock. Kinodynamic motion planning amidst moving obstacles. In *Proc. 2000 IEEE Intl. Conf. on Robotics and Automation*, volume 1, pp. 537–543, April 2000.

[13] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. *Intl. J. of Robotics Research*, 20(5):378–400, May 2001.

[14] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. Intelligent Robotics and Autonomous Agents. MIT press, Cambridge, MA, 2005.

[15] D. Halperin, L. Kavraki, and J.-C. Latombe. Robotics. In G. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 48, pp. 1065–1094. CRC Press, second edition, 2004.

[16] L. E. Kavraki and S. M. LaValle. Motion planning. In B. Siciliano and O. Khatib, editors, *Handbook of Robotics*, chapter 5. Springer Verlag, 2008.

[17] K. I. Tsianos, I. A. Şucan, and L. E. Kavraki. Sampling-based robot motion planning: Towards realistic applications. *Computer Science Review*, 1(1):2–11, August 2007.

[18] Motion strategies library. URL http://msl.cs.uiuc.edu/msl.

[19] F. Schwarzer, M. Saha, and J.-C. Latombe. MPK — motion planning kit. URL http://robotics.stanford.edu/~mitul/mpk/.

[20] R. Diankov and J. Kuffner. OpenRAVE: A planning architecture for autonomous robotics. Technical Report CMU-RI-TR-08-34, Robotics Institute, Carnegie Mellon University, July 2008. URL http://openrave.programmingvision.com/.

[21] Kineoworks. URL http://www.kineocam.com/kineoworks-library.php.

[22] A. V. Estrada, J. M. Lien, and N. M. Amato. VIZMO++: a visualization, authoring, and educational tool for motion planning. In *Proc. 2006 IEEE Intl. Conf. on Robotics and Automation*, pp. 727–732, 2006.

[23] E. Plaku, K. E. Bekris, and L. E. Kavraki. OOPS for motion planning: An online open-source programming system. In *Proc. 2007 IEEE Intl. Conf. on Robotics and Automation*, pp. 3711–3716, Rome, Italy, 2007.

[24] D. Hsu, J.-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. *Intl. J. of Computational Geometry and Applications*, 9(4-5):495–512, 1999.

[25] V. Boor, M. H. Overmars, and A. F. van der Stappen. The Gaussian sampling strategy for probabilistic roadmap planners. In *Proc. 1999 IEEE Intl. Conf. on Robotics and Automation*, volume 2, pp. 1018–1023, 1999.

[26] N. Amato, O. Bayazit, L. Dale, C. Jones, and D. Vallejo. OBPRM: An obstacle-based PRM for 3D workspaces. In P. K. Agarwal, L. E. Kavraki, and M. T. Mason, editors, *Robotics: The Algorithmic Perspective*, pp. 155–168. A.K. Peters, 1999.

[27] D. Hsu, T. Jiang, J. Reif, and Z. Sun. The bridge test for sampling narrow passages with probabilistic roadmap planners. *Proc. 2003 IEEE Intl. Conf. on Robotics and Automation*, 3:4420–4426, September 2003.

Biographical Information

Mark Moll is a research scientist in the Physical and Biological Computing group at Rice University. Previously, he has was a research scientist at the Information Sciences Institute at the University of Southern California. Before that, he was a post-doctoral research associate at Rice University in the Physical and Biological Computing Group. He a obtained a Ph.D. degree in Computer Science from Carnegie Mellon University in 2002 and a M.S. degree in Computer Science from the University of Twente in the Netherlands in 1995. His current research interests include motion planning, and geometric problems in robotics and computational structural biology.

Janice Bordeaux, Associate Dean of the George R. Brown School of Engineering and a Licensed Psychologist, is an expert on learning in higher education, educational program improvement processes, and outcomes assessment. In the last nine years Bordeaux has collaborated on numerous federally and privately funded educa-

tional projects including interdisciplinary, multi-institutional, and web-based initiatives for educational communities.

Lydia E. Kavraki received the Ph.D. degree in computer science from Stanford University, Stanford, CA, U.S.A. She is currently the Noah Harding Professor of computer science and bioengineering with Rice University, Houston, TX. She is the author or coauthor of more than 150 technical papers and a robotics textbook: Principles of Robot Motion (MIT Press, 2005). Her current research interests include motion planning for continuous and hybrid systems, connection of formal methods and task planning with motion planning, manipulation, networked multiagent systems, and applications of robotics methods to biology.

Prof. Kavraki is a Fellow of the Association for the Advancement of Artificial Intelligence and the American Institute for Medical and Biological Engineering. She is the recipient of the Association for Computing Machinery Grace Murray Hopper Award and the Early Academic Career Award from the IEEE Robotics and Automation Society. Information about her work can be found at http://www.kavrakilab.org.