Graphical Display of Search Trees for Transparent Robot Programming

Joaquin A. Pockels
Computer Engineering Department
Polytechnic University of Puerto Rico
San Juan, PR 00918
joaquin.pockels@gmail.com

Ashwin Iyengar
304 Le Roi Road
Pittsburgh, PA 15208
pghburger@gmail.com

David S. Touretzky
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
dst@cs.cmu.edu

Abstract

Search algorithms such as Rapidly-exploring Random Trees (RRTs) are common in robot programming. Including graphical representations of the output of these algorithms in a robotics framework can make the algorithms more accessible to students, and can also help programmers analyze and account for unexpected results. For this project, we used the Tekkotsu open source robot programming framework, available at Tekkotsu.org. We extended Tekkotsu's graphical user interface for displaying vision data and maps to also display the output of an RRT search. We created several demos using two types of searches: one from a navigation path planner, and one from an arm path planner. In some cases the search had no solution, and the graphical output helped to illustrate why. This confirms the utility of the RRT visualization for explaining unexpected search results. We expect that this tool will also contribute to improved student understanding of the search algorithm.

Introduction

The Rapidly-Exploring Random Tree (RRT) is a randomized algorithm used for single-query path planning problems, designed for efficiently searching non-convex high-dimensional spaces (LaValle 1998). On mobile robots with manipulators and sensors, these algorithms are used for complex path and motion planning problems. This involves randomly searching the system configuration space for collision-free paths while satisfying all the imposed constraints. Depending on the complexity of both the configuration space and the constraints, RRT searches may fail to find a solution, or may produce unexpected results. Finding the causes of such problems and then solving them could potentially be time consuming.

In this project we used Tekkotsu (see Tekkotsu.org), a modern, open source robot programming framework that uses RRT search for planning. The platform also includes a tool for displaying vision data and maps. We explore the possibility of helping programmers explain unexpected search results by extending this tool to graphically display the trees generated in the RRT search. This may also help students better understand how the algorithm works.

Framework and The Crew

Tekkotsu provides an intuitive set of primitives for perception, manipulation and attention control (Tira-Thompson and Touretzky 2011). The framework includes a group of interacting software components designed for high-level behavior programming called the "Crew" (Touretzky and Tira-Thompson 2010). Three of the Crew members are the Pilot, the Grasper, and the MapBuilder. The Pilot is responsible for localization, navigation and motion. The Grasper controls manipulation and arm path planning. The MapBuilder is responsible for vision and for creating Tekkotsu's internal representation of the world. It interacts with the Pilot and the Grasper via references to maps and shapes.

The Pilot and the Grasper both use a generic RRT planner based on the RRT-Connect algorithm. This modified version of the original RRT algorithm incrementally builds two RRTs, one rooted at the start configuration and the other at the goal, until the two trees meet (Kuffner and LaValle 2000). When this happens, a solution path is extracted by tracing the path from the meeting point back to the root of each tree. Finally, a smoothing operation is applied to simplify the resulting path. A user-specified limit on the maximum number of iterations allows the search to terminate for unsolvable problems, but may also cause some solutions to be missed.

The SketchGUI and Extensions

Tekkotsu includes a facility called the SketchGUI (refer to Figure 1) for visualizing a variety of information structures, including intermediate results and outputs of the vision system, in a camera-centric space, a robot-centered local map, and a global world map used for navigation (Touretzky et al. 2007). As seen in Figure 1, the SketchGUI displays both vision data (as pixel arrays) and maps represented as simple geometrical objects (points, lines, ellipses, polygons, etc.). As an example, the robot's location and heading are represented on the world map by an isosceles triangle. In addition, particles from the particle filter the Pilot uses for localization can be displayed as shapes on the world map (Watson and Touretzky 2011). We extended this tool by adding a new type of graphical object for efficiently representing complex

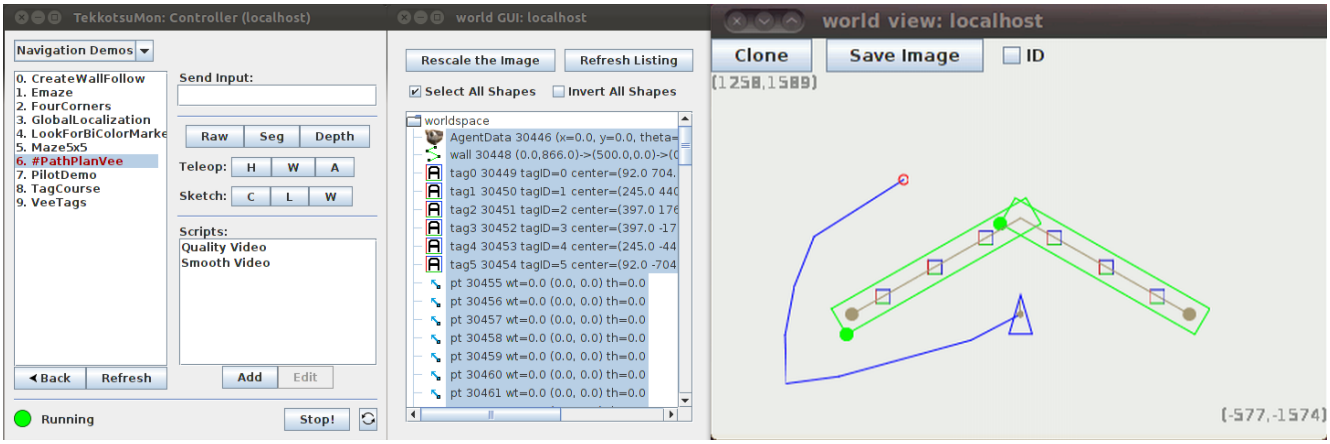

Figure 1: SketchGUI, ControllerGUI and world shape space panel. The ControllerGUI (left) is Tekkotsu's principal interface for controlling a robot. It allows the user to run system or user-defined behaviors, directly control the robot's effectors, and launch the SketchGUI. The SketchGUI (center) is used to view any of three coordinate spaces or maps, called Camera Space, Local Space and World Space (C, L, W buttons in the ControllerGUI). The figure shows the contents of the world shape space as a list of components. The world shape space (right) is where every geometric component is rendered.

structures, such as search trees, composed from simpler geometric primitives. We implemented a general drawing facility that can display points, lines, polygons, ellipses, and text objects in the selected coordinate space.

Applications

In this paper we show two applications of the facility for visualizing the results of RRT searches. In the first application, 2D navigation path planning, we use a set of lines to draw both the start and goal search trees directly onto the robot's world map, following the RRT-Connect algorithm as shown in Figure 2. As the robot executes the path, the Pilot updates the robot's position and the localization particle states on the world map, so deviations from the planned path are visually apparent.

In the second application, path planning for a three degree-of-freedom planar arm, the higher dimensional search space precludes our drawing the search tree directly. Instead, nodes in the tree are visualized by drawing the corresponding arm configuration, with each link of the arm depicted by its rectangular bounding box. The bounding boxes are automatically extracted from the robot's kinematic description. For easy interpretation, the two search trees are displayed in different colors, using the same color convention as the 2D navigational path planner. The varying colors are shown in Figure 5. These arm configurations are all superimposed on the world map so their relationships to the robot's body and environmental obstacles are apparent.

Demos and Robots

We created four demos to show the effectiveness of these visualizations. They varied in the complexity of the simulated environments and the difficulty of the search task. (The two are not equivalent: a narrow corridor is a simple environment but can be difficult to navigate using a random search algorithm.) Demos 1 through 3 involve the Pilot's navigation path planner, while Demo 4 uses the Grasper's arm path planner. To solve the problems we used two different robots designed at Carnegie Mellon University and fully supported in the Tekkotsu software framework:

- The Calliope Robot, an iRobot Create based robot capable of vision, navigation and manipulation (Touretzky et al. 2010).
- The planar Hand-Eye System, which includes a three degree of freedom planar arm and a vision system (Nickens et al. 2009).

Each demo was run several times, producing different results due to the random nature of RRT search. We used the graphical display of the search trees to analyze the outcomes and determine the reason why some searches failed.

The Demos

Demo 1 – Single-Exit Box for a Navigation Path Planning Problem

This demo uses a relatively simple path planning problem to illustrate the basic idea of the RRT-Connect algorithm. The robot starts out in the south end of a box-shaped room with a narrow doorway to the north. The walls of the box are obstacles that must be avoided. The goal location is a point to the south of the box, in the opposite direction of the exit. See Figure 2. A greedy algorithm that simply headed toward the goal would be trapped at the bottom of the box. A wavefront algorithm can solve this problem but would require a lengthy search to reach the goal. The RRT-connect solution is efficient and quick. To prevent the search trees from expanding infinitely, we limited the search space to a region slightly larger than the largest object in the world map. (We must provide some extra space for the robot to have enough room to maneuver around the outside of the box, but we don't

Figure 2: Demo 1. A simple navigation path planning problem: finding a collision-free path out of the box to the goal location shown as a red circle near the bottom of the figure. One search tree (shown in black) is grown from the robot's start point, and the other (shown in green) from the goal location. The trees connect just outside the doorway at the top of box. The final extracted and smoothed path is shown in blue.

want it wandering off into territory we know is empty.) Examining the resulting search tree, one can easily see how the solution to the task was found.

Demo 2 - Nested Single-Exit Boxes for a Harder Path Planning Problem

This demo illustrates what happens when the navigation path planner is given a more difficult problem. We nested two single-exit boxes and had their doorways point in opposite directions. In the first attempts, the RRT would rarely find a solution. By analyzing the graphical display of the search output (see the left half of Figure 3), we found that the search trees were successfully permeating the environment, passing through the doorways and proceeding along the corridors. This shows that enough collision-free space is available for the creation of new search points. But the trees were failing to meet. We determined that this was caused by too strict a limit on the number of iterations (4,000 by default). We made this limit user-modifiable so that it could be raised for harder problems, but kept small to allow the quick rejection of simple but unsolvable problems. When Demo 2 was retested with a limit of 10,000 iterations, the RRT always found a solution. A typical solution is shown in the right half of Figure 3. The shape of the solution varied depending on the location of the connected trees, which varied from one run to the next. So, for example, on some runs the solution involved a left turn after exiting the first box, while on other runs it involved a right turn.

Demo 3 - Nested Single-Exit Boxes with Narrow Corridors

This demo results in a path planning failure for reasons that are not visually obvious. We slightly modified the Demo 2 environment by narrowing down the vertical corridors of the nested boxes to the collision-free limits. This causes the navigation path planner to fail most of the time. The source of the problem is unclear when examining the environment alone. But when visualizing the search tree superimposed on the environment, the difficulty is revealed, as seen in Figure 4. The failure is caused by the limited collision-free space available in the narrow corridor, resulting in a scarcity of target points there. Increasing the number of iterations would be of little use, because only a narrow band of target points can produce progress through the corridors, and these are unlikely to be found by random search. Widening the vertical corridors is the appropriate remedy.

Demo 4 - Arm Configuration Setup for an Arm Path Planning Problem

In this demo, shown in Figures 5 and 6, Tekkotsu's arm path planner tries to plan a path for the robot's arm to move from the start configuration (shown in red) to the goal configuration (shown in magenta) while avoiding obstacles. The obstacles are represented in the world map as three small ellipses. A successful solution requires the arm to fold up by bending at the elbow, then sweep past the obstacles, and then unfold.

Since this is a higher dimensional search space, we cannot lay out the search tree in the plane. Instead we display each node by plotting the arm configuration it represents, taking the joint angles from the node and the fixed link lengths from the arm's kinematic description. This approach makes the proximity of a node to a collision visually obvious, at the cost of not being able to show the parent/child relationships between nodes.

As in the case of the navigation planning graphical display, visualizing the search allows the user to determine the cause of failures in non-obvious situations, such as when there is insufficient room for the arm to maneuver between two obstacles, or when the path around an obstacle would require one of the joints to exceed its limits of travel. With this application the user can also develop a better understanding of the arm's behavior, for example, by analyzing the frequency of similar configurations visible in the search space. However, depending on the problem's complexity, too many plotted arm configurations may occlude our vision of the search space, making it difficult to interpret. To solve this, we implemented a secondary display option to show only the smoothed intermediate arm configuration and both starting and ending configurations (Figure 6). In the problem shown here, the RRT-Connect algorithm found a solution in every run.

Use in Teaching

To determine the effectiveness of this tool in a learning environment, we showed it to a group of four high school students in Puerto Rico participating in a Saturday enrichment

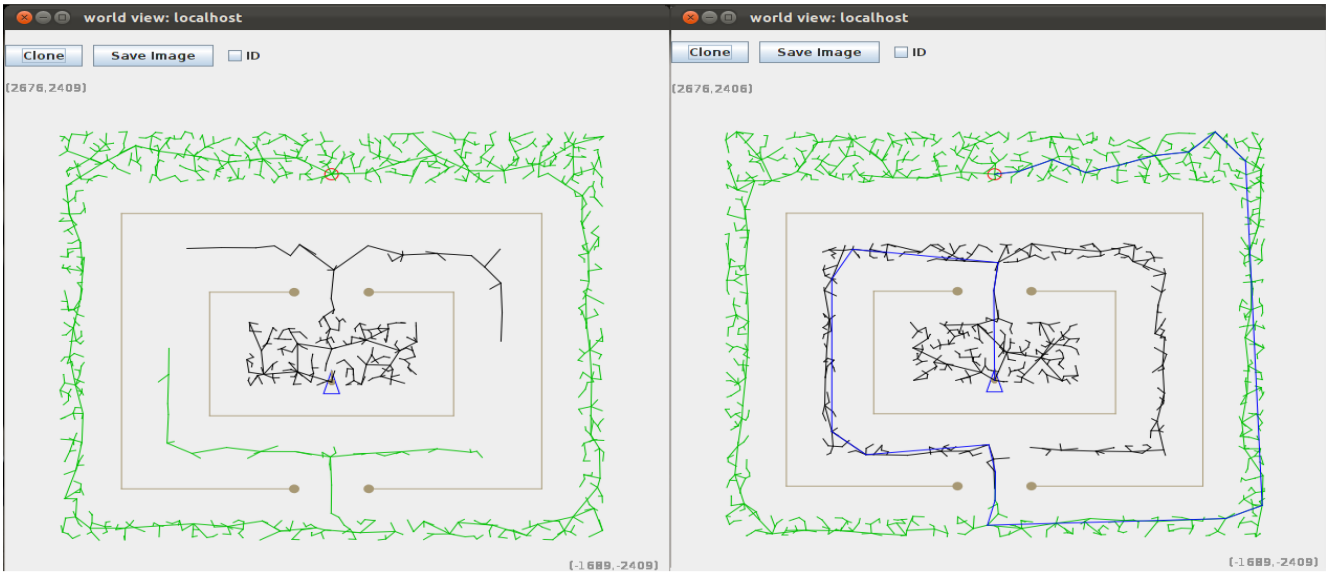

Figure 3: Demo 2. The left image shows the algorithm failing to solve a more difficult navigation path planning problem. Examining the graphical display lets the user determine the reason for the failure. On the right, a typical solution to the problem, found after raising the iteration limit.

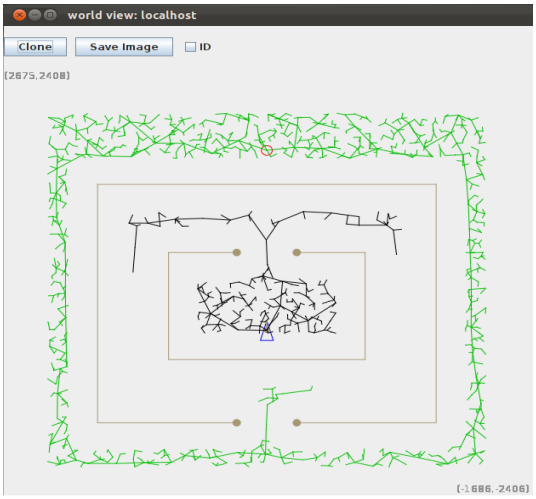

Figure 4: Demo 3. Failed navigation path planning problem. By analyzing the output of this search, specifically the lack of target points found within the vertical corridors, the user can infer that the limited space is impeding the creation of collision-free points.

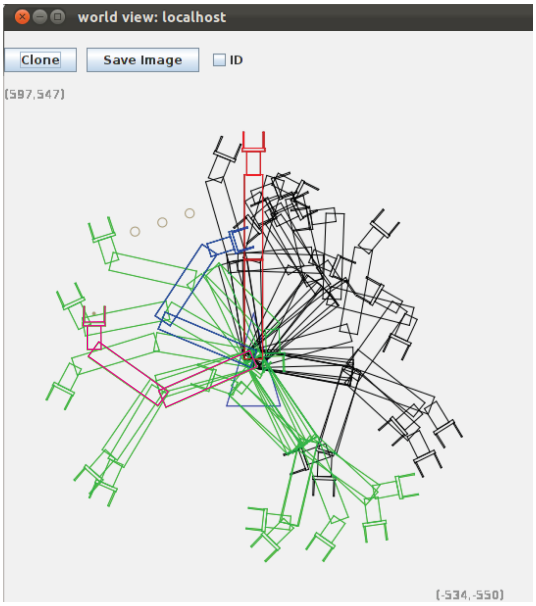

Figure 5: Demo 4. Solved arm path planning problem. The starting configuration is displayed in red and the goal configuration in magenta. An intermediate configuration on the solution path is shown in blue. A subset of nodes from each search tree are shown in black (start) and green (goal).

program where they were learning Tekkotsu programming. The graphical display was used to augment a lesson on RRT path planning. We started with Demo 1 because students were already familiar with robot navigation. Seeing the two search trees connect in the graphical display made it easy for the students to understand the algorithm. They were also shown Demo 4 to provide a different perspective on RRT search. It was simple enough that the search was understandable even though the students had not yet studied kinematics. The students were able to assimilate the lesson.

We are planning a more extensive test in the Spring 2012 semester at Carnegie Mellon, where undergraduates and MS students in a Cognitive Robotics course will be asked to experiment with the RRT-Connect algorithm and produce vi-

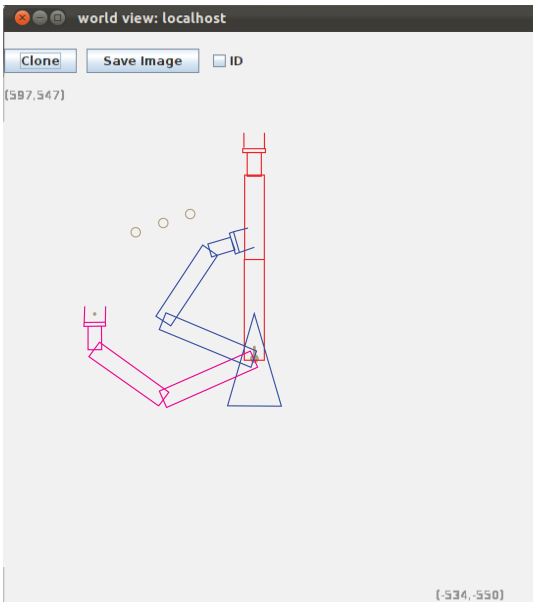

Figure 6: Demo 4. Unobstructed view of the solved arm path planning problem. The starting configuration is displayed in red and the goal configuration in magenta. An intermediate configuration on the solution path is shown in blue.

sualizations of their searches.

Conclusion

The primary innovation in this work is the integration of graphically displayed search trees with a robot-maintained description of the environment, including obstacle locations, the robot's own position and heading, and localization particle states. In some cases, such as Figures 3 and 4, the RRT-Connect search found no solution, and the graphical representations offered explanation. For example, if there were too few iterations, as in Demo 2, or too many unavoidable collisions, as in Demo 3, the trees would never meet. Visualization also helped explain some decisions made by the planner to go right or left, caused by the randomness of the RRT search. We have preliminary indications that beginning robotics programmers can use these visualizations to develop a better understanding of path planning even if they are not familiar with all the details of the RRT-Connect algorithm. These displays have certainly been useful in our own work.

Future Work

We don't believe we have exhausted all the potential applications for this tool. For example, in a game-playing application, possible moves might be visualized directly on the game board. If the robot is playing chess, it could indicate the most highly-rated moves it is considering by drawing arrows from each piece's starting position to its ending position on the board in the world map. Color could be used to indicate the relative rankings of the candidates.

Acknowledgments

We thank Owen Watson, Ethan Tira-Thompson and Ramon Cardona for helpful discussions. This work was supported in part by the National Science Foundation's Broadening Participation in Computing Program through awards 1042322 (ARTSI Alliance) and 0940522 (CCCE Alliance).

References

Kuffner, J. J., and LaValle, S. M. 2000. RRT-connect: an efficient approach to single-query path planning. In *ICRA'2000*.

LaValle, S. M. 1998. Rapidly-exploring random trees: A new tool for path planning. TR 98-11, Computer Science Dept., Iowa State University.

Nickens, G. V.; Tira-Thompson, E. J.; Humphries, T.; and Touretzky, D. S. 2009. An inexpensive hand-eye system for undergraduate robotics instruction. 423–427. Proceedings of the Fortieth SIGCSE Technical Symposium on Computer Science Education, Chattanooga, TN.

Tira-Thompson, E. J., and Touretzky, D. S. 2011. The Tekkots robotics development environment. In *Proceedings of ICRA-2011*.

Touretzky, D. S., and Tira-Thompson, E. J. 2010. The Tekkotsu "crew": Teaching robot programming at a higher level. In *Proceedings of EAAI-10: The First Symposium on Educational Advances in Artificial Intelligence*. Menlo Park, CA: AAAI Press.

Touretzky, D. S.; Halelamien, N. S.; Tira-Thompson, E. J.; Wales, J. J.; and Usui, K. 2007. Dual-coding representations for robot vision in Tekkotsu. *Autonomous Robots* 22(4):425–435.

Touretzky, D. S.; Watson, O.; Allen, C. S.; and Russell, R. 2010. Calliope: Mobile manipulation from commodity components. In Thomaz, A., and Anderson, M., eds., *Papers from the 2010 AAAI Robot Workshop*. Technical report WS-10-09. Menlo Park, CA: AAAI Press.

Watson, O., and Touretzky, D. S. 2011. Navigating with the Tekkotsu Pilot. In *Proceedings of FLAIRS-24*. AAAI Press.