A Benchmark Suite for Motion Planning Algorthims

Latrelle Samuels Daniel Tomkins Nancy M. Amato

***Abstract*— The motion planning problem, is given the start and goal configurations of a robot in an environment, to find a valid path from the start to the goal. The motion planning problem can be solved by many different algorithms. The Parasol Laboratory Motion Planning Benchmark Suite includes narow passage, dis-assembly, surface, and folding problems. During the course of the summer, I improved and organized the benchmark suite in the Parasol Laboratory at Texas A&M University in College Station, Texas. I also used the benchmarks to evaulate several motion planning algorithms, comparing the overall time, collision detection calls, and shortest path generated in a given environment.**

I. INTRODUCTION

A benchmark is standard hard problem used to compare different solving techniques. By techniques, I mean motion planning algorithms, which will be discussed later in the paper. Benchmarks allow for comparison of several motion planning algorithms throughtout the laboratory. Research is an ongoing process in which we generally want to find news way of making things work better and faster. Implementing a new algroithm would generally be tested on a benchmark to determine it's efficiency. First, an older algorithm will be applied to a benchmark. Then, a new algorthim will be applied to compare the results of the two. The outcome should be that the newer algrothim would sove the problem faster and takes up less memory space. This is an example of how benchmarks are used.

II. RELATED WORK

In this section, we review relevant information to benchmarks discussed in this paper. We first highlight some work in motion planning, and then review some work on configuration space, probabilistic roadmaps, rapidly exploring random trees, and the Parasol Motion Planning Library (PMPL).

A. Motion Planning

Motion planning is a term used for the process of detailing a function into deteached motions. For example, consider the navigation of a mobile robot inside a building on a search-and-rescue mission. The robot is to leave it's start configuration and move through the buiding avoiding walls to get to a goal configuration or waypoint. A motion planning algorithm would take in information of the robot and the walls to return a valid path without collisions. Motion planning is applied throughout several application such as robotic sugery, video games, robot navigation, automation, and protein folding. Even on a day-to-day basis, we, as humans use motion planning when commuting to and from the workplace. It is essential for benchmarks, because it is the way how the problems need to be solved.

Fig. 1.

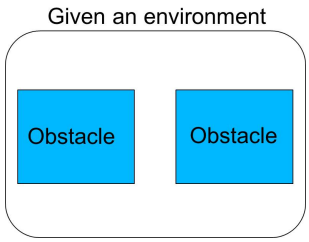
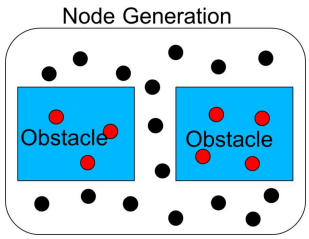


Fig. 2. Roadmaps that are discared are noted as red dotted-lines.



B. Configuration Space

A configuration describes the position of a robot and configuration space (C-Space) is the set of all possible configurations. One thing to remember in C-space is Degrees of Freedom (Dof). The Degrees of Freedom are the number of parameters that define it;s configuration. To solve a motion planning problem, algorithms must conduct a search in the C-space. The C-space provides an abstraction that converts the robot to a point. Each point in the C-space represents a different congfiguration of the robot. The set of configurations that avoids collisions with an obstacle is called free space and the set of configurations that make up the obstacle is called obstacle space.

C. Probabilistic Roadmaps

The probabilistic roadmap (PRM) planner is an algorithm that solves a motion planning problem given a start and goal configuration[**?**]. A PRM consists of two phases: a construction phase and a query phase. In the construction phase, given an environment, random nodes are generated (figures 1 & 2). Nodes that are generated on the obstacle are discared . Once the nodes are generated, a roadmap is created to connect the nearest nodes. Any roadmap that is connected through an obstacle is discared (figure 3). Finally, in the query phase, the start and goal configuration is added and a valid collision-free path is created (figure 4).

PRM's are general good for solving multi-query problems. Generally a query is a single path with a start and finish configuration and they are connected after some iterations.

Fig. 3. Nodes that are discared are denoted as red circles.

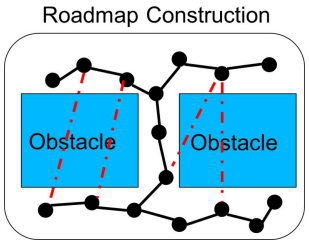
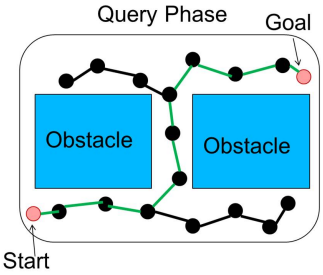


Fig. 4. Nodes that are discared are denoted as red circles.



D. Rapidly Exploring Random Trees

A Rapidly-exploring Random Tree (RRT) is a data structure algorithm that is desgined for efficiency search non-convex high-dimensional spaces [**?**]. RRT's are constructed incrementally in a way that quickly reduces the expected distance of a randomly chosen point to the tree. The direct advantage of this algorithm is that it can be applied to nonholonomic and kinodynamic planning.

Path planning is generaly viewed as a search in a metric space, *X*, for a continous path from and initial state, x_{init}(labeled "Start" on figure 5) to a goal x_{goal}. What happens, is a node is randomly generated named x_{random} (figure 6). The next step is that a new node, x_{new} is generated. Notice that the node is created in the direction of the emphx$_{random}$ but not on the obstacle. Once x$_{new}$ is plotted, another new node is created x$_{near}$ which is near the node recently plotted. This process keeps going until eventually we have reached the goal (figure 8).

III. METHODS

A. Organization & Upgrading

First, we organized the benchmarks according to the type of problems. Some benchmarks narrow passages while some were more global problems like get a robot to travel through several obstacles to reach a path. The benchamrks are organized into four categories: narrow passage, disassembly, folding, and surface. Narrow passage problems are hard because of the small areas between two or more obstacles. The benchmarks were then upgraded to current standards of the laboratory motion planning library.

Fig. 5.

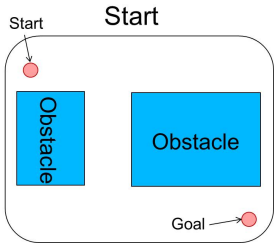


Fig. 6. X_{random} is generated

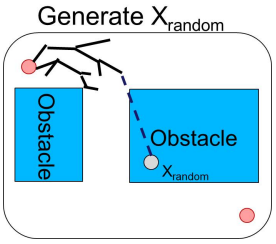
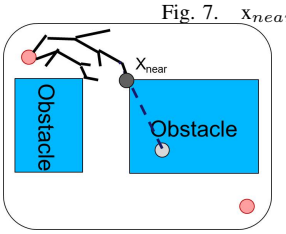


Fig. 7. x$_{near}$ is created from x$_{new}$



B. PMPL, STAPL, & VIZMO++

We implemented a x & y method in a C++ motion planning library, developed in the Parasol Laboratory at Texas A&M University, which was a distrubuted graph library(STAPL)[**?**]. The Standard Template Adaptive Parallel Library) is a framework for developing parallel programs in C++. It is desgined to work on both shared and distributed memory parallel computers. To actaully view a simulation of how the algorithms solve the benchmarks we used a visual tool developed by the Parasol Laborary called, VIZMO++. VIZMO++ is a 3D visualization/authoring tool for files provided/generated by OBPRM motion planning libray [**?**].

IV. EXPERIMENTS

A. PRM & OBPRM

To test the PRM and Obstacle-Based(OBPRM) algorthim, we used the s-tunnel benchmark(figure 9). The s-tunnel benchmark is simply a obstacle with a(n) s-shaped tunnel through it for a robot to travel through onto the other side. PRM's are commonly solving these type of problem where

Fig. 8. After some iterations, the start and goal states are connected

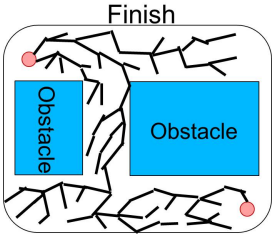


Fig. 9. S-Tunnel Benchmark

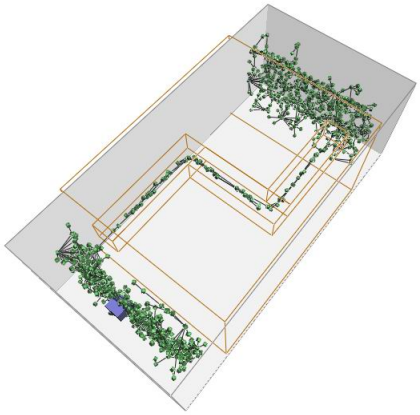
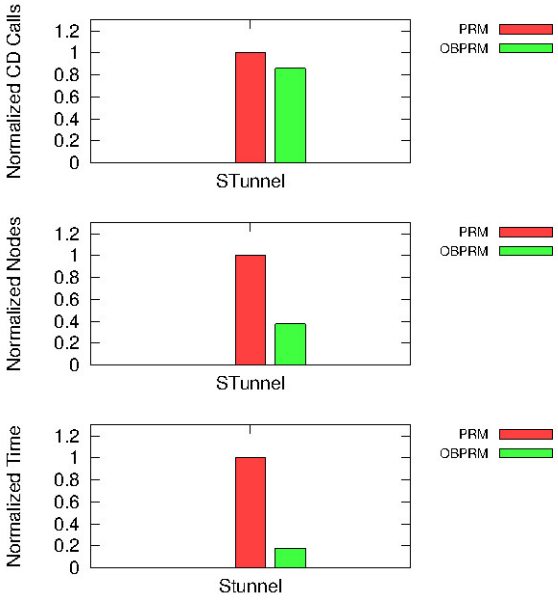


Fig. 10. S-Tunnel Benchmark after results from running PRM & OBPRM



there isn't neccesary a narrow passage for the robot to travel through. The s-shaped tunnel has enough space for the robot to travel through onto the other side to reach the goal. A total amouth of thirty test were run on each algrothim. The total time for all the test to finish were about thirty to forty minutes. In the results (figure 9), after running PRM & OBPRM, we obsrved that OBPRM generally worked very well compared to PRM. Some good metrics to compares these results are time and node generation. The OBPRM algorthim is great for the problem because it takes less time to solve the problem and less nodes. The more nodes that are generated, the more money that it cost and also more memory space is taken up on the computer.

B. RRT & OBRRT

We choose the flange problem as our testing benchmark for narrow passage problems. The overall goal of the flange problem is to get the pipe to manuever itself through the hole and eventually be free from the obstacle. RRT would

Fig. 11. Flange Benchmark

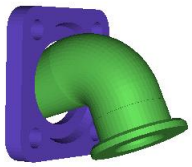
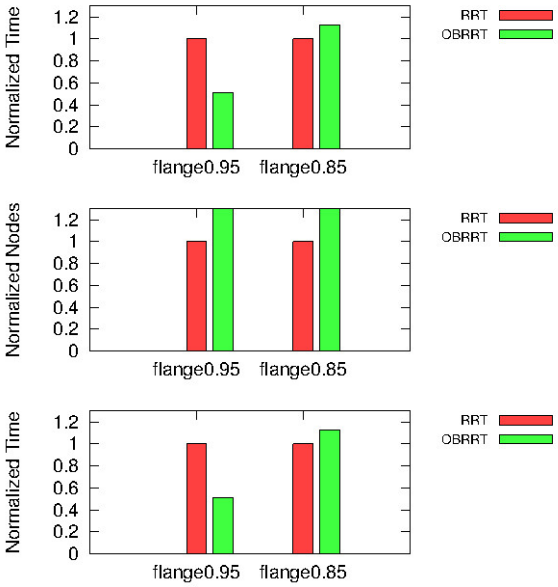


Fig. 12. Variations of Flange Benchmarks after results from running PRM & OBPRM



be good for solving this problem because of the narrow passage between the pipe robot and hole in the obstacle. A total amouth of thirty test were run on each algrothim. The total time for all the test to finish were about ten to fifteen minutes..

For the test of the flange benchmark (figure 11) we chose two variants of the problem. The difference between the two are the dimesions of the diameter at the end of the pipe. The higher the number, the bigger the end of the pipe is, thus creating less space for the pipe to manuever (narrow passage). The results from the experiments shows us that RRT work better for this benchmark. OBRRT is a form of RRT in which the increaments are created near the obstacle. The OBRRT algorithm creates more nodes then the RRT which is very costly and could take up more memory space than expected.

V. CONCLUSION

Motion planning is done everyday by humans without a single given thought. We are accustomed to simple things such as walking everyday once we learn it at an early age of life. While it may be easy to us, it is not the same for implementing these methods on robots. Configuration space will help us simplify the problem, by treating the robot as an abstraction; generally speaking, a point. We take that knowledge and figure out which motion planning algorithm

is best suited for solving the problem. PRM maybe good for multi-query problems while RRT"s are good for single query problems.

In conclusion, a benchmark is a standard hard problem used to compare different solving techniques. We use these benchmarks to compare between different motion planning algorithms in the Parasol Laboratory. These are very usual to us because they give us a since on making motion planning algorithms better to solve harder problems. Although there were not described in this paper, there are many other benchmark that are harder. Those benchmarks are included in the benchmarks suite that can be visited on the Parasol Laborartory webpage.

VI. ACKNOWLEDGEMENT

I would like to thank my faculty mentor, Dr. Nancy Amato, first, for the oppurtunity to come to Texas A&M University and participate in my first REU ever. I really enjoyed the lab atmosphere and learning about programming with C++ and using PMPL . Next, I would like to thank Daniel Tomkins, my graduate student mentor, for sticking with me for these long ten weeks and guiding me through the research. I also want to thank Jory Denny and the other graduate student in the Parasol Laboratory for answering many of the questions I had about PMPL and Computer Science.