Motion Planning Test Suite

Leonel Pena, Marcos Pena, Brandon Martinez, Art Martinez, Jonathan Mulhern, Irving Solis, Shawna Thomas, Nancy M. Amato

***Abstract*— Motion planning is the problem of finding a valid path for a robot from a start to goal configuration. In addition to robotics, it has applications to other fields in computer science such as CAD software, autonomous vehicles, and artificial intelligence. Motion planning in robotics is difficult because of the large number of variables needed to account for a robots shape, size, dynamic environments, and forces such as gravity and friction. Weve designed a test suite to evaluate motion planning algorithms, that is integrated with Parasol Lab's C++ motion planning library (PMPL), visualization tool (Vizmo++), and a physically based robot simulator. In particular, weve developed benchmark scenarios to exercise motion planning algorithms in a variety of scenarios and for a range of robots. A representative set of problems were designed such as freebody scenarios that varied from two to three dimensions, a crane-like, robot with joints tasked with moving its arm from one location to another, and a car-like robot with different actuators that limited its movement to seem more car-like. These benchmark scenarios will enable the comparison and evaluation of future motion planning algorithms**

I. Introduction

Living in the 21st century we as a human race have continued to advance our technology in pursuit to always find a solution for whatever problem we face. Nowadays we revolve around computers and what we can create and compute from the machines, but the emphasis on technology couldn't be stressed enough as we are essentially stepping into the future. We are trying to be as efficient as we can be, and this also involves removing humans from an environment and replacing them with robots that are autonomous. However, before a robot can become autonomous it needs to be programmed to be able to navigate through environments and perform certain tasks. To be able to do this alone these robots need motion planning.

Motion Planning is a problem with many applications such as robotics and computational biology. Motion planning entails finding a valid path from a start configuration to an end configuration. This can be approached in many ways, such as having a direct path, a path with clearance, or a path that takes an agent near obstacles. This called for a new way to simulate these motion planning strategies as the method already in place did not take into account forces, the robots own shape and size, and dynamic environments.

Parasol labs has developed a C++ library, named PMPL (Parasol Motion Planning Library), that contains several algorithms and strategies for solving many different kinds of problems. Sampling methods within the library include, but are not limited to, PRM, MAPRM, OBPRM, RRT, etc. Basic probabilistic road map (PRM), medial-axis probabilistic road map (MAPRM), object-based probabilistic road map (OBPRM), and rapidly-exploring random tree (RRT). PRM is when there is a random sampling, nodes are randomly placed within the environment and then connected to make a road-map, and the most efficient and complete path is used for the agent to move across the solved path. MAPRM is when the sampler acquires equal distance from the obstacles and the edge of the environment to create an arbitrary skeleton so that an agent will essentially traverse this equal distanced path. OBPRM is when an environment is sampled with the intention to place nodes near obstacles so that the agent can maneuver around them. An RRT grows a tree rooted at the starting configuration by using random samples from the search space. As each sample is drawn, a connection is attempted between it and the nearest state in the tree until it reaches the goal configuration. After the process of finding a valid path for a robot to take place there is a visual tool that allows us to view a simulation of the problem being solved.

The PMPL simulator was developed to serve this very cause. Within it, motion planning strategies could be ran, tested, and simulated with all the desired variables. Beforehand, a motion planning strategy would still be ran and computed, now real world variables could be added to further realize the tests being run. So while before, a problem that would've been solved without accounting for forces such as gravity and friction will now face the same test in the test suite with the addition of these variables to see if the task would still be completed.

II. Related Work

In this section, we review existing methods and explain concepts and terms that we will use in the rest of the paper.

A. PMPL

As stated before, PMPL is a C++ library that has sampling-based algorithms developed by Parasol Labs, the use of PMPL correlates with motion planning strategies. It is capable of solving several motion planning problems with different strategies and approaches. Not only is PMPL an essential part of Parasol Labs, but it is always used for computations and simulations. With PMPL having many purposes, storing code, running simulations, and constantly testing, it is an important component that has been and will be used.

B. Vizmo

Within the existing work there is a software, Vizmo, that has been utilized to create environments, objects, and also used to view final paths a planner creates. Upon running experiments with Vizmo there are several types of files used; environment, query, path, and any object files that are created

This research supported in part by NSF awards CNS-0551685, CCF 0702765, CCF-0833199, CCF-1439145, CCF-1423111, CCF-0830753, IIS-0916053, IIS-0917266, EFRI-1240483, RI-1217991, by NIH NCI R25 CA090301-11,and by DOE awards DE-AC02-06CH11357, DE-NA0002376, B575363.

and incorporated. It doesn't take outside forces into account essentially making the space into a vacuum. For example, if gravity or friction were to be implemented it would mimic better how an agent or robot moves physically.

III. Methodology

The purpose of our work was to develop several benchmarks to help evaluate motion planning algorithms and therefore help the Parasol Lab to exercise their algorithms in a range of scenarios. Our goal was to create a set of comprehensive problems for future use. We created several scenarios that were quite varied, these included a set of 2D problems, a set of 3D problems, and a set of problems featuring manipulator robots all within the simulator. With the tools on hand, we also simulated a car-like robot which had limited movement imposed on it to better simulate a physical robot, which we then mirrored in the real world with actual physical robots.

In order to have fair results between all planners; such as, MAPRM, OBPRM, and PRM, the same scenarios were ran with all the strategies. This meant that some strategies would be more suited to some tasks than others, and this would be easily visualized in the motion planning test suite. Our approach throughout the summer involved several steps to ensure that we were achieving viable results.

1) An environment is created by adding several objects and obstacles and setting their positions. We determine whether it is a 2D or a 3D environment depending on the problem.
2) A robot is created through the use of triangulating vertices's and making faces. The robot then moves along within the environment.
3) A query is created by determining a start and goal configuration within the environment. The query is only valid if it is within the environment and does not collide with obstacles.
4) A motion planning strategy is applied so that the query is solved and a path is then formed for the robot to follow.
5) The robot follows the final path within the simulation and several files are created once it has completed solving.

   For Physical Robots:
6) Define controls for the robot through the definitions of actuators. It essentially gives limits to how the robot can move forwards and backwards at a certain speed.
7) Connection is established between Robot and Host through the creation of a server that allows for the passing of information and instructions.
8) Robot follows the path as it would within a simulation; however, the robot moving in the real world is different depending on different factors.

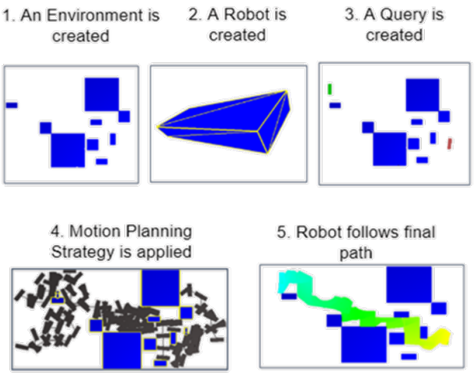

Fig. 1. Motion Planning

IV. Experiments

A. Scenarios

1) 2D Freebody Translational Robot: Within this scenario, the square robot was tasked with translating through two obstacles without rotating or changing orientation to reach an end goal on a planar surface. *See Fig. 5 for results*

2) 2D Freebody Rotational Robot: Within this scenario, the square robot was tasked with translating and rotating through nine obstacles to reach an end goal on a planar surface. *See Fig. 6 for results*

3) 3D Freebody Translational Robot: Within this scenario, the cuboid robot was tasked with translating through two obstacles without rotating or changing orientation to reach an end goal on a three dimensional space. *See Fig. 7 for results*

4) 3D Freebody Rotational Robot: Within this scenario, a rectangular robot was trapped between two obstacles and the only way to escape and reach an end goal was to rotate on a three dimensional space. *See Fig. 8 for results*

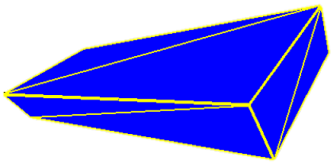

Fig. 2. Robot used in the previous scenarios

5) Arm Manipulator Translational Robot: Within this scenario, the crane-like robot was tasked with translating through multiple obstacles without rotating or changing orientation to reach an end goal on a planar surface. *See Fig. 9 for results*

6) Arm Manipulator Rotational Robot: Within this scenario, the crane-like robot was fixed to a location but the arm was free to move and rotate through multiple obstacles to

reach its end goal in a three dimensional space. *See Fig. 10 for results*

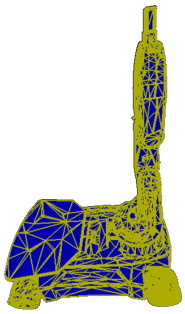

Fig. 3. Robot used in the previous scenarios

7) Rotational Translational Car-Like Robot: Within this scenario, a simulated robot was tasked with translating and rotating to the end goal around a mock up of the lab we worked in. Additionally, the simulation was mirrored by a physical robot conducting the very same commands in the real world.

8) Reverse Car-Like Robot: Within this scenario, a simulated robot was tasked with translating and rotating to the end goal around a mock up of the lab we worked in by reversing out of its start position, correcting itself, and moving towards its end goal. Additionally, the simulation was mirrored by a physical robot conducting the very same commands in the real world.

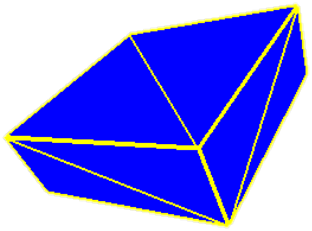

Fig. 4. Robot used in the previous scenarios

B. Statistics

In the simulator, when we run the tests, we are given a set of information on how the planner performed through the test suites. With this information, Parasol Labs will be able to have a deeper understanding of their Motion Planning strategies to further improve their planning techniques with a tool that is able to simulate other variables such as collisions, forces, and self- limitations.

In the tables following, we show information gathered from the simulator which is an average of 10 different runs; strategy used, number of nodes, time to solve, and final path length.

- Strategy Used
  - We used nine of the Motion Planning strategies that Parasol Labs has in their PMPL.
  - These perform differently because they are built with different focuses.
- Number of Nodes
  - This is the average number of nodes used to solve the problem.
- Time to Solve
  - This is the average time taken to solve the problem.
- Final Path Length
  - This is the average distance that the robot took to solve the problem.

As for the MAPRM results for the manipulator arm robots, we received no results because of the increasing complexity involved with robots with high degrees of freedom. Our manipulator arm robot had eight degrees of freedom when translating, and five when it was fixed. Since MAPRM has to find the medial axis for *all* the degrees of freedom, this takes an extraordinary amount of time to complete, rendering it basically unusable.

Strategy	# Nodes	Time	Path Length
PRM	21	0.139 s	76.5
Gauss	69	0.456 s	63.4
OBPRM	122	2.69 s	67.65
MAPRM	21	0.175 s	76.5
Bridge	10	0.105 s	230.5
TogglePRM	83	0.766 s	98.5
RRT	60	0.0509 s	61.39
RRT*	300	1.22 s	61.82
OBRRT	250	0.342 s	87.52

Fig. 5. *2D Translational Robot Strategy Results*

Strategy	# Nodes	Time	Path Length
PRM	412	25s	136.45
Gauss	199.8	8.6s	141.15
OBPRM	494.05	58.7s	143.05
MAPRM	247.2	103.5s	133.55
Bridge	129.6	25.2s	229.85
TogglePRM	1353	150.74s	169.15
RRT	1104.9	3.3s	152
RRT*	2020.2	28.9s	126.68
OBRRT	170.6	.45s	209

Fig. 6. *2D Freebody Rotational Robot Strategy Results*

Strategy	# Nodes	Time	Path Length
PRM	42	0.06477 s	107.15
Gauss	341	15.43 s	63.4
OBPRM	776	16.99 s	93.4
MAPRM	28	0.4953 s	79.25
Bridge	15	52.37 s	153.5
TogglePRM	460	1.066 s	173.65
RRT	149	0.04160 s	4.3906
RRT*	1994	21.2 s	76.762
OBRRT	79.5	0.11368 s	7.77236

Fig. 7. *3D Freebody Translational Robot Strategy Results*

Strategy	# Nodes	Time	Path Length
PRM	33	0.23s	79.2
Gauss	33.5	0.05s	32.6
OBPRM	60.2	0.06s	43.8
MAPRM	141.9	3.49s	84.1
Bridge	86.5	1.09s	164.6
TogglePRM	290.4	1.23s	122.9
RRT	78	0.06s	60.6
RRT*	49	0.08s	50.7
OBRRT	50	0.08s	63.8

Fig. 8. *3D Freebody Rotational Robot Strategy Results*

Strategy	# Nodes	Time	Path Length
PRM	31.1	18.22s	133.70
Gauss	50.7	01.27s	139.47
OBPRM	79.7	08.49s	146.18
MAPRM	NA	NA	NA
Bridge	26.8	00.92s	137.67
TogglePRM	87.7	00.27s	143.47
RRT	78.6	00.27s	006.32
RRT*	288.3	08.47s	104.07
OBRRT	135.2	00.86s	058.70

Fig. 9. *Manipulator Translational Robot Strategy Results*

Strategy	# Nodes	Time	Path Length
PRM	10.00	1.11540 s	1.1
Gauss	10.80	.04402 s	1.1
OBPRM	17.90	.07824 s	1.1
MAPRM	11.67	.0154 s	1.2
Bridge	10.10	.13782 s	3
TogglePRM	10.00	.766 s	1.1
RRT	4.80	.00405 s	1.581798
RRT*	7.80	.01236 s	1.03777
OBRRT	5.90	.01593 s	2.51708

Fig. 10. *Fixed Manipulator Rotational Robot Strategy Results*

V. Conclusion

To conclude this paper, we successfully created and tested the motion planning test suite to analyze the performance and help Parasol Labs exercise their algorithms in a range of scenarios. With our environments, robots, and queries we have successfully demonstrated the capabilities of the test suite and aided fellow researchers in their future work. All of the results from the experiments proved to be quite satisfying as the percentage error was less than one percent, and the standard deviation wasn't noticeably diverging therefore validating the results. The overall purpose of the motion planning test suite is to have a tool for motion planning developers that will help analyze their strategies in a realistic scenario thus allowing them to further improve their algorithms.

VI. Acknowledgements

We want to give great thanks to the DREU, USRG, Texas A&M, and parasol labs for hosting us this summer. We also want to give great thanks to our mentors, Nancy Amato and Irving Solis, alongside the many people working and helping us in parasol labs. This research was supported by the Department of Computer Science and Engineering of Texas A&M University College Station.