Performance Analysis of Trajectory Planning Algorithms for Mobile Robotic Navigation

Hersch Nathan[1], Biyun Xie[1]

[1]Department of Electrical and Computer Engineering, University of Kentucky

Intelligent Robotic Arm Lab

Pigman College of Engineering

Department of Electrical and Computer Engineering

Chellgren Center

Student Fellow

Introduction

During major disasters there is a need to utilize robots to transverse complex environments. These autonomous systems will encounter environments with different densities of impassible obstacles. In this project we explore a static environment with random obstacles in different configurations. We conduct a performance analysis of a collection of motion planning algorithms to enable the selection of an optimal algorithm for a specific application.

Background Information

For motion planning algorithms considered there are two methods of representing space. One method applies graph theory to create a discrete space of node connected to one another via edges [1]. These nodes can be arranged in any format desired, which allows these algorithms to be used in a verity of problems. Commonly for trajectory planning these nodes are arrange in a graph where each node is connected on the edges and possibly the corners. The other method applies a continuous space and uses probability to randomly pull a space to create the node network [2].

Methods

We select a set of 7 trajectory planning algorithms from the Python Robotics library [3] for comparison. We benchmarked the performance in 2 different environments. The performance criteria that was used to compare the motion planning algorithms were as follows: path length, computational time, and fail rate for the comparisons. The environments we explore were: Cluttered (60-80% density of obstacles), and Sparse area (less than 40% density). For the sampling based algorithms we tried 20 times per each environment.

Algorithm	Type	Informed	Optimization Ability	Advantages	Disadvantages
Dijkstra's	Graph Based	Uninformed	Optimal	Simplicity and Effiency	Only Works well in sparse enviroments
A Star (A*)	Graph Based	Informed	Optimal	Complete and optimal	Requires an effienct and effective heuristic
Breadth First Search (BFS)	Graph Based	Uninformed	Optimal	Good for unweighted graphs	Does not work well for large graphs
Depth First Search (DFS)	Graph Based	Uninformed	Not Optimal	Good for deep paths	Is not complete nor optimal
RRT	Sampling Based	Uninformed	Not Optimal	Bias toward unexplored regions	Not optimal and had hard time with narrow passages
RRT Star (RRT*)	Sampling Based	Informed	Optimal	all the advantages of RRT with optimality	complex implementation and computationally
Informed RRT Star	Sampling Based	Informed	Optimal	Computationally More Efficient	More complex implementation

Fig 1: Table of Algorithms compared

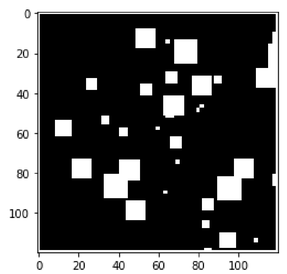

Fig 2: Sparse Grid Based Map

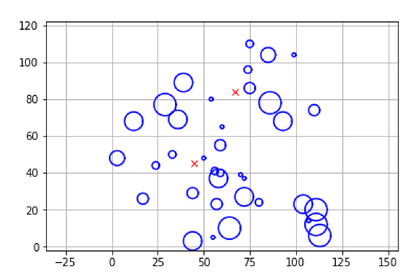

Fig 3: Sparse Continuous Map

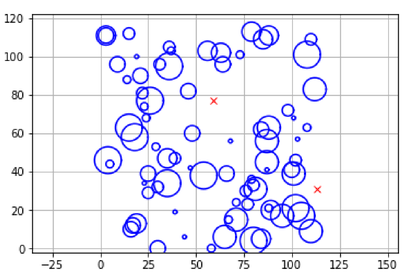

Fig 4: Cluttered Continuous Map

Results

We found that in the variety of environments we tested the sampling based algorithms were less computationally intensive than graph-based algorithms. The graph-based algorithms were generally able to find shorter paths. BFS and DFS preformed almost identically. RRT* was able to improve on RRT.

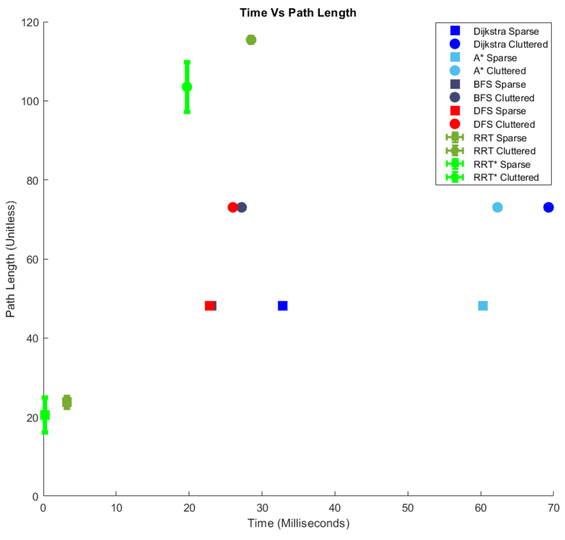

Fig 7: Graph of Algorithm Performance

	Median Time (Sec)	Time Standard Deviation	Median Length	Length Standard Deviation	Median Time (Sec)	Time Standard Deviation	Median Length	Length Standard Deviation
	Sparse				Cluttered			
Dijkstra's	0.0328		48.1127		0.0693		73.0538	
A Star (A*)	0.0603		48.1127		0.0623		73.0538	
Breadth First Search (BFS)	0.0231		48.1127		0.0272		73.0538	
Depth First Search (DFS)	0.0228		48.1127		0.026		73.0538	
RRT	0.0032	0.0008	23.7329	1.3717	0.0285	0.0085	115.436	0.7417
RRT Star (RRT*)	0.0002	0.0006	20.4628	4.4222	0.0197	0.0327	103.5294	6.2578
Informed RRT Star	0.3625	0.0935	23.5309	0.7654	1.649	0.4986	114.2331	1.0571

Fig 6: Table of Algorithm Performance

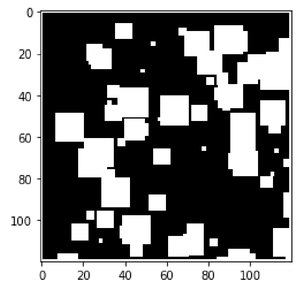

Fig 5: Cluttered Grid Based Map

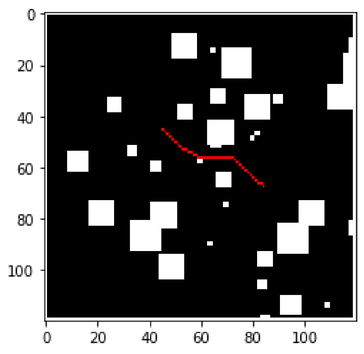

Fig 8: A* Sparse Environment

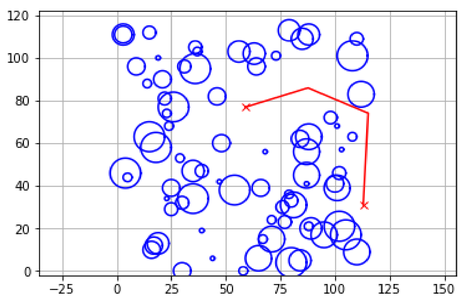

Fig 9: RRT* Cluttered Environment

Acknowledgements

This research was partly supported through an NSF Grant 2205292 and through the University of Kentucky Department of Electrical and Computer Engineering's Undergraduate Research Fellow Program.

References

[1] E. W. Dijkstra, "A note on two problems in connexion with graphs," in Edsger Wybe Dijkstra: His Life, Work, and Legacy, 2022, pp. 287–290

[2] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodyamic planning," The international journal of robotics research, vol. 20, no. 5, pp. 378–400, 2001.

[3] A. Sakai, D. Ingram, J. Dinius, K. Chawla, A. Raffin, and A. Paques, Pythonrobotics: A python code collection of robotics algorithms, 2018. arXiv: 1808.10703 [cs.RO].