Assistive Control for Smart Wheelchair Project with Autonomous Navigation

1st Vivek Sood
Robotics Engineering, ENPM
University of Maryland College Park, USA
vsood@umd.edu

2nd Rigved Rajesh Kulkarni
Robotics Engineering, ENPM
University of Maryland College Park, USA
rigvedk@umd.edu

3rd Yoseph Kebede
Robotics Engineering, ENPM
University of Maryland
College Park, USA
ykebede2@umd.edu

Abstract—Sampling based path planning algorithms have proven to be more efficient than action-based path planning algorithms since they reduce the time complexity of the algorithm as the graph or the configuration space complexity increases. However, the random sampling techniques employed commonly in the sampling based algorithms utilize the idea of pre-constructed structure of the obstacle. This restricts the algorithms to a huge extent when the problem at hand is extended in an unknown environment. Thus, inspite of showing incredible performances, the sampling based techniques have proved to be inefficient in unknown environments. In this paper, we implement a self-adjusting probabilistic roadmap algorithm which is proposed to deal with unknown obstacles quickly. This algorithm stored the generated samples in a grid structure which makes it easier for them to be checked with the obstacles when encountered. If any of the sample in the grid lies with the obstacle detected, a self-adjusting algorithm is employed to adjust the collision point so that we have samples free of obstacle collision. With this self-adjusting algorithm, the roadmap or the grid adjusts itself to have points in the obstacle free space whenever a change in environment is observed. Simulation of the same is carried out in the paper further.

Index Terms—Sampling based algorithms, roadmap, selfadjusting algorithm, Unknown environment

I. Introduction

Sampling based path planning algorithms were developed to be fast as they don't require a detailed map of the environment to function. However, this sometimes comes at the cost of optimality as they only need to check if a randomly selected configuration of the robot is in collision with any obstacle or not. They also use local planners using an average resolution interpolation between two configurations to decide if the configurations can be connected without collision or not.

Probabilistic roadmaps come under the class of multi query algorithms [2] which can be used for any pair of start and goal configurations. It discovers the collision free space by going through random configurations of the robot followed by constructing a graph formed by connecting these samples.

In this paper, a randomized path planning algorithm based on the probabilistic roadmap (PRM) planner is used to deal with the dynamically changing environment with unknown obstacles. The algorithm uses a grid based sample classification process and thus doesn't need to reconstruct the entire graph.

Sampling-based algorithms work in a way, wherein they represent the configuration space with a roadmap of sampled configurations. A basic algorithm samples N configurations in C (Configuration space), and retains those in C_{free} (obstacle free configuration space) to be considered in valid nodes, commonly referred to as 'milestones'. A roadmap is then constructed that connects two milestones P and Q if the line segment PQ is completely in C_{free}. Again, collision detection is used to test inclusion in C_{free}. To find a path that connects points S and G, they are added to the roadmap. If a path in the roadmap links points S and G, the planner succeeds, and returns that path. If not, the reason is not definitive: either there is no path in C_{free}, or the planner did not sample enough milestones. These algorithms work well for high-dimensional configuration spaces, because unlike combinatorial algorithms, their running time is not (explicitly) exponentially dependent on the dimension of C. They are also (generally) substantially easier to implement. They are probabilistically complete, meaning the probability that they will produce a solution approaches 1 as more time is spent.

The paper discussed two main classes of planners. In the first class, the algorithms are called multi-query as they can be used for any pair of start and goal configurations. These algorithms discover the collision-free space by selecting random configurations of the robot and then constructing a graph by connecting these samples. The resulting graph can be used for any planning query. The second class is, the constructed structure is mainly a tree which has been designed for a known planning query. These algorithms are called single-query as the tree is usually rooted at the start position of the given planning problem.

The paper states the common assumption in sampling-based algorithms - the environment is well defined such that the robot's location relative to obstacles is known. However, the operational environment of a mobile robot is rarely known, the encounter the robot has with the obstacle is not

known at any given point in time. The implementation of the paper was solely done by simulation and no sensor errors or characteristic physical error were taken into consideration.

Research in the field of predicting the path for the robot in an unknown environment or an uncertain environment is the topic of leading research, finding its application majorly in Autonomous self-driving cars. The project, following the paper, will leverage the advantage of sampling-based techniques to achieve speedy planning output and attempt at implementing a randomized path planning algorithm based on the probabilistic roadmap (PRM) planner.

II. Background

The project derives the concepts from some background algorithms and techniques implemented and deep-rooted in path planning literature. The background is as follows:

A. Probabilistic Roadmaps (PRM) Planner

The project leverages the advantage of the ability to learn the space without requiring a detailed map which is sometimes computationally impossible to get. Instead, it relies on two main components. The first component is able to determine if a given configuration is in collision with any obstacle or not, while the second component, i.e. the local planner, determines if the robot can freely move from one configuration to another without any collision. For example, let there be a set of two randomly generated points in the configuration space, say P and Q. The first component determines if either of the points lie in the obstacle space. If not, the local planner checks if the two points are connectable (which is determined by a check of some radius/distance betweeen the two points). If both the components are positive, the nodes are added in grid/the roadmap.

B. PRM Extensions in Unknown Environments

Conventional PRM and its extensions are not able to deal with unknown and changing environments because they require to learn the configuration space before the navigation starts. Few extensions of PRM have been proposed to deal with this situation. A real-time PRM has been introduced [4] which ignores the obstacles in the configuration space and generates a roadmap in the entire space and encodes the mapping from workspace cells to nodes and arcs in the graph. When the environemnt changes, this mapping is used to make appropriate modifications to the graph and paths can be planned over the modified roadmap. However, since this planner ignores every obstacle in the environment, when dealing with crowded environments or dense maps, the processing time will increase substantially in crowded environments. Another such PRMbased planner has been proposed that provides a framework to managing either roadmap under changing settings [5]. Once changes in the environment are detected, nodes in obstacle space and colliding paths are discarded. This leads to the emergence of separated roadmaps, or forests. However, this planner results in increasing the computational costs.

III. Methodology

A. Implementation of the Algorithm

The project has two parts to it in terms of practical implementation:-

1) Low-Dispersion PRM: Roadmap-based methods are generally used for multi-query purposes (catering to multiple problem scenarios), as they maintain a roadmap to answer multiple queries simultaneously. The main data structure used is a graph whose nodes are points in the configuration space. Edges in this graph exist between configurations that are close to one another, and the robot can move through these edges without collision. Edge is defined by a node which can be connected to another point in graph (the roadmap) as generated in figure 1. A typical PRM algorithm has two phases: a learning phase and a querying phase. In the learning phase, the roadmap is created. Collision-free configurations, called ‘samples’, are generated randomly which form the nodes of the roadmap (a subset of all the points in the configuration space). In the query phase, a number of iterations are conducted based on the number of sampled points and an attempt is made to connect these points based on a distance criteria to find its nearest neighbours, thus adding edges to the roadmap. To solve a particular query, the initial and goal configurations are added to the roadmap and a graph search algorithm is used to find a path. The efficiency of the algorithm depends on how well the roadmap can capture the connectivity of the configuration space [4]. In the implementation of the paper, ’n’ random number of sampled points are generated and a connectivity of atmost ’x’ nodes in the graph is established to define the dynamics of the entire graph structure. The Low-dispersion PRM algorithm is one such Roadmap-based path planning method which works similar to the original PRM with one major difference in the learning phase. This paper’s model is based on LD-PRM, the learning phase is different such that the generated samples need to fulfill an additional criterion in order to be included in the roadmap. This criterion implies that the samples are forbidden to be close to each other more than a predefined radius which proves instrumental in reducing the number of valid node points to be taken into consideration.

   The idea behind this formulation is that if we want to fit non-overlapping balls inside the configuration space,

the aggregate volume of these balls should be smaller than the volume of the free space.

Based on the Lebesgue measure (volume) of the collision-free configuration space, this radius can be defined as follows [1]:

$$R_S(n) = \left[L\left(Q_{\text{free}}\right)(n-\lambda)/\left(\pi n^2\right)\right]^{1/2} \quad (1)$$

where:-

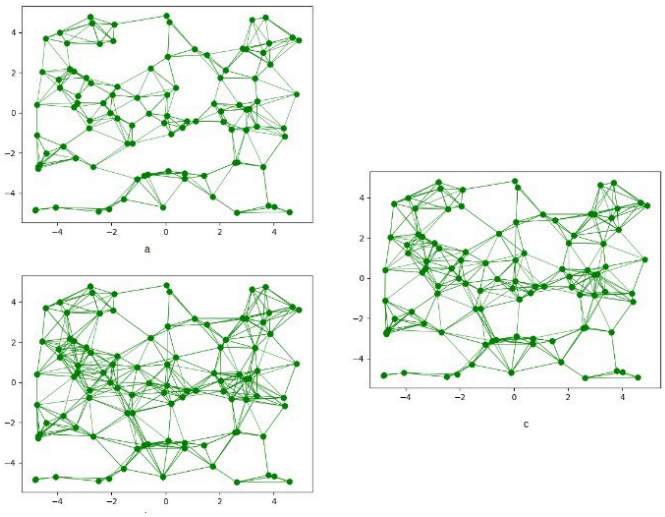

Fig. 1. Road map generated from Low-Dispersion PRM with a. atmost 5 viable connects, b. atmost 10 viable connects, c. atmost 8 viable connects

- Q_{free} is the free configuration space
- n is the number of samples (nodes) in the roadmap
- λ is a positive scaling scalar

We implement this so as to leverage PRM and reduce the valid nodes generation by implementing Lowdispersion PRM so as to achieve smaller graphs to solve for planning queries. The algorithm pseudo-code can be referred in figure 2.

Pseudo-code: LD-PRM

```
G (V, E) = ø
R = radius of the forbidden space
S = [point (1), point (2), ..., point (N)]
for i = 0 to N:
        current_point = S(i)
        distance_map = ø
        if current_point not in G (V, E):
                V ← current_point
                R =
                for j = 0 to N:
                        if S(j) ≠ current_point and d (current_point, S(j)) ≤ R:
                                distance = d (current_point, S(j))
                                distance_map = add (distance, S(j))
                sort (distance_map: distance)
                for all values in distance_map:
                        E ← values [1]
                        if |E| > 'x':
                                break
```

Fig. 2. Pseudo code for Algorithm 1. Radius of forbidden region is defined Eq.(1), 'x' is the number of permissible connections in the roadmap and G defines the roadmap

The algorithm starts with the generation of the graph that will in turn be used by the planner to find the optimal path using A* algorithm. For generating the road maps, the following parameters were taking into consideration:

a) Fixed number of nodes to be generated.
b) Minimum and maximum distance between two samples nodes (V) in the graph.
c) Maximum number of connections (E) that one node can have.
d) Nodes (V) should not lie in the obstacle space and the edge should not be intersecting the obstacle space.

2) Path-planner in Unknown Environment: After roadmap generation, optimal path was obtained using A* algorithm, the robot was made to follow the path along with continuously monitoring/polling the environment for dynamic injection of obstacles into the configuration space. The following steps were then followed:-

a) If an obstacle is detected, the following steps were taken:

i) Check whether the obstacle hinders the rest of the optimal path that the bot needs to follow. ii) If there is an obstruction present due to the new obstacle:

A) i. the algorithm finds the nodes and the edges in the remaining path that are in the obstacle space or intersecting with the obstacle space.
B) ii. the nodes in the path and the rest of the nodes in the sample space that lie in the obstacle's region are shifted to come

outside in free space. This is followed by the re-adjusting the edges/connection of the existing graph to accommodate the newly shifted points as well as remove any edges that intersect the obstacle space.

iii) After the adjusting the roadmap steps 3-5 are repeated till the robot reaches its goal position.

b) The robot is made to follow the newly reconfigured path from the point where it observed a new obstacle.

c) The robot resumes the same path, if the new obstacle does not obstruct the remaining path to be followed.

d) The computational speed of detecting the nodes that lie in the new obstacle space can be improved by mapping each node in the roadmap to a grid cell and the checking with grid cell/s are being occupied by the obstacle/s, followed by retrieving the corresponding nodes that were mapped to that particular grid.

B. Features of the Algorithm

- This algorithm stores the generated samples in a grid structure based on their position in the corresponding configuration space which makes it computationally affordable to check them against collision later.
- It also enables the path planner to include safety as a decision factor during the actual navigation.

- Next, only the occupied grid cells by the undetected obstacles and their corresponding samples will be checked and the roadmap responses to the changes in the environment as soon as they occur.
- Furthermore, the size of the graph is maintained, and the occupied nodes are pushed away from the obstacle rather than being removed from the set of samples.

IV. Results and Observations

To reproduce the results and achieve the working of the algorithms mentioned, The project was built by implementing algorithm 2 (refer. Section III, A.2) on the top of algorithm 1 (refer. Section III, A.1). Initially, with just one obstacle in the configuration space, the roadmap along with the best path along the roadmap was found as shown in figure 4, The blue path showing the path from start node to goal node along the roadmap.

New obstacle	Time taken to find path (seconds)	Time taken to readjust	Cost of path calculated cumulatively
No new obstacle	**0.007**	**0**	**11.8 units**
Obstacle 1 added	**0.0032**	**0.514**	**12.37 units**
Obstacle 2 added	**0.005**	**0.942**	**13.27 units**

Fig. 3. Table comparing time taken to find path, re-adjust and cumulative path cost

We employed the following configuration:

- Start Node := (-2.54, -3.0)
- Goal Node := (4.0, 4.0)
- Orientation := 20 (in deg.)
- Clearance := 0

When the obstacle is encountered by the robot in the configuration space, the roadmap is re-adjusted and the path is found again. Once the Path through the roadmap is found, the robot follows the path to the goal position, shown by orange color in figure 5

V. Conclusion

In this paper, we implemented the Self-Adjusting Roadmap strategy from scratch by leveraging upon the algorithm of PRM and implemented a path planning algorithm on the top of it such as, the A* algorithm to find the shortest and optimal path through the roadmap from the start position to the goal node. By implementing the algorithm suggested in the paper [1], we could achieve to establish a fast sampling based algorithm in an Unknown Environment. This algorithm would work in any

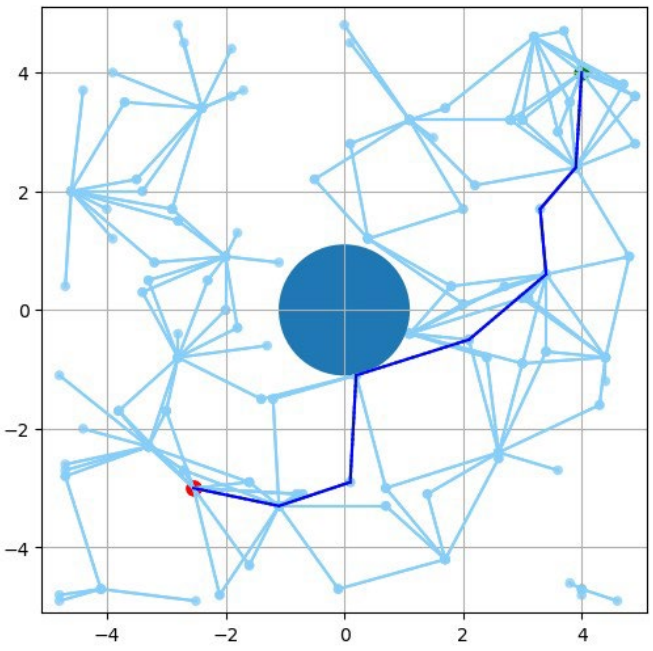

Fig. 4. Initial path obtained from the algorithm

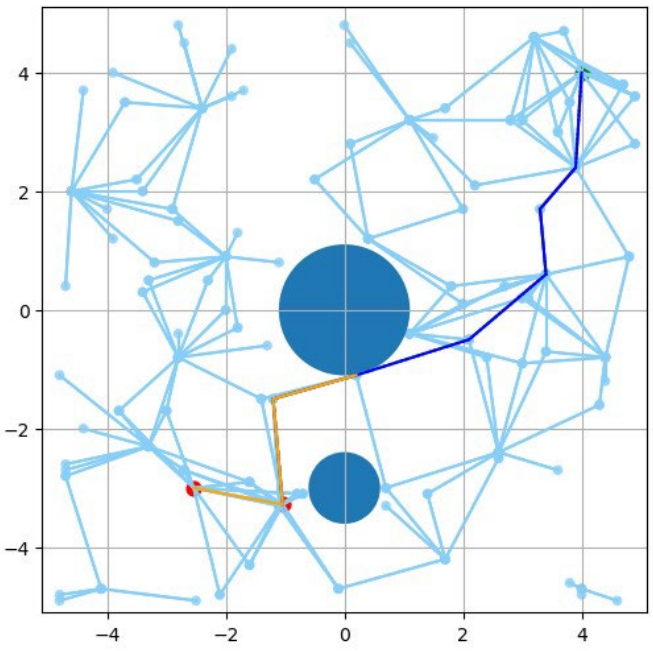

Fig. 5. Path obtained from the algorithm after two obstacles are introduced into the environment

unknown environment. It could be observed that by restricting the number of viable connectivity, the path discovered may not be optimal or the shortest, in most cases. However, with dense roadmap, the self-adjustment time may increase. It was observed that the sampling-based algorithm makes the path planning algorithms extremely fast, which

forms the basis of this paper. And thus, suitable results were obstained.

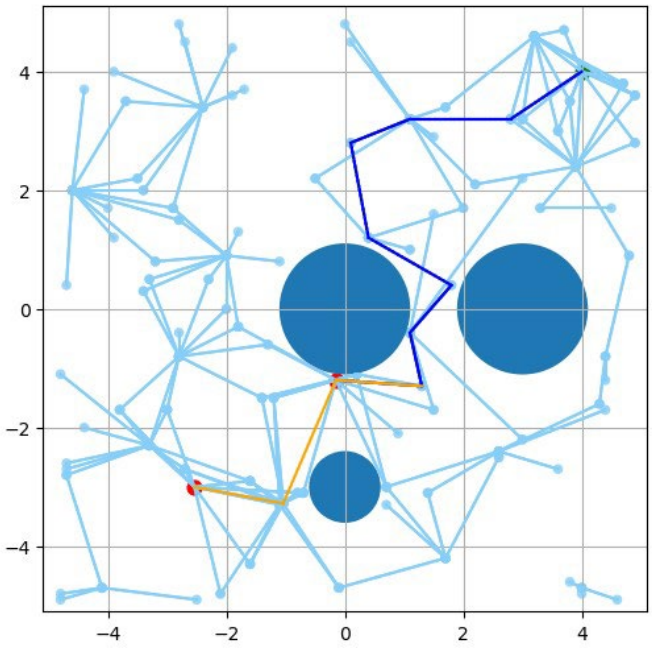

Fig. 6. Robot moves forward to the next node after third obstacle is introduced

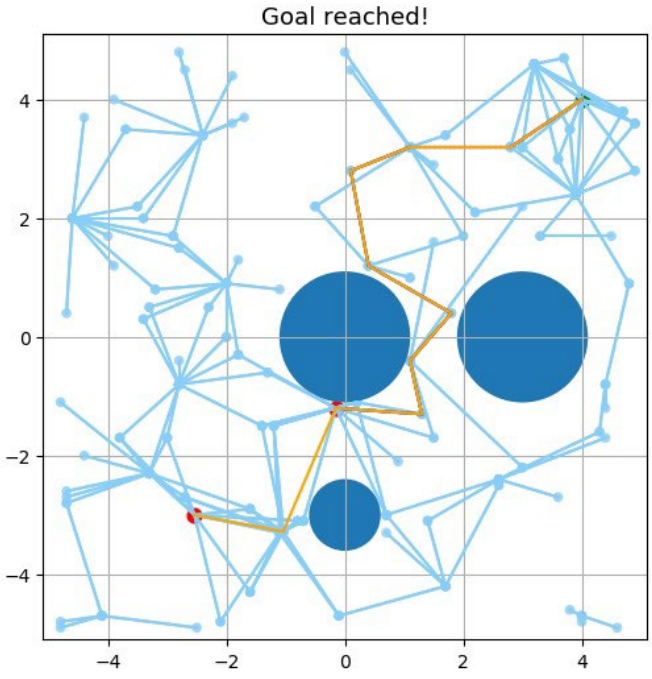

Fig. 7. Final path obtained from the algorithm after three obstacles are introduced into the environment

In future work, we intend to implement the algorithm in a real-

environments," in2018 IEEE International Conference on Robotics and Biomimetics (ROBIO), 2018,pp. 1094–1101.

[2] L. E. Kavraki, P. Svestka, J. Latombe, and M. H. Overmars,"Probabilistic roadmaps for path planning in high-dimensionalconfiguration spaces,"IEEE Transactions on Robotics and Automation, vol. 12, no. 4, pp. 566–580, 1996

[3] P. Leven and S. Hutchinson, "A frameworkfor realtime pathplanning in changing environments,"The International Journalof Robotics Research, vol. 21, no. 12, pp. 999–1030, 2002.DOI:10.1177/027836490202101200 1. eprint: https://doi.org/10.1177/027836490202101200 1. [Online]. Available: https : / / doi . org / 10 .1177/027836490202101200 1.

[4] W. Khaksar, T. Sai, M. Khaksar, and O. Motlagh, "A lowdispersion probabilistic roadmaps (ld-prm) algorithm for fast andefficient samplingbased motion planning,"International Journalof Advanced Robotic Systems, vol. 10, p. 1, Nov. 2013.DOI:10.5772/56973.

[5] T.-Y. Li and Y.-C. Shie, "An incremental learning approach to motion planning with roadmap management," in Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292), 2002, vol. 4, pp. 3411–3416 vol.4.

world robot such as a Turtlebot in an indoor home territory, by taking into account of all the sensing errors or calibration errors possible.

REFERENCES

[1] W. Khaksar, M. Z. Uddin, and J. Torresen, “Self-adjusting roadmaps: A fast sampling-based path planning algorithm fornavigation in unknown