# Dynamic Obstacle Avoidance & Path Planning in a Hospital Environment

Dominic Ferro
*MS in Robotics Engineering*
*Worcester Polytechnic Institute*
Worcester, MA
drferro@wpi.edu

Thabsheer Jafer Machingal
*MS in Robotics Engineering*
*Worcester Polytechnic Institute*
Worcester, MA
tmachingal@wpi.edu

Girivaasan Chandrasekaran
*MS in Robotics Engineering*
*Worcester Polytechnic Institute*
Worcester, MA
gchandrasekaran@wpi.edu

***Abstract*—The safe and efficient mobility of mobile robots in a hospital environment is crucial for the prompt delivery of supplies, equipment, and medication. Hospital surroundings, however, are dynamic and complicated, with obstacles like equipment, medical staff, and patients. Due to their assumptions about static impediments and inability to manage non-holonomic restrictions, conventional path planning techniques like Rapidly-exploring Random Trees (RRT) or Probabilistic Roadmap (PRM) may not be suitable for such situations. To address the dynamic and changing nature of hospital environments while taking into account the non-holonomic limits of mobile robots, path planning algorithms efficient in dynamic environments must be implemented. In an attempt to learn and understand the nature of mapping methods that work well in a complex environment such as a hospital environment, we implemented D*, Hybrid Potential Based Probabilistic Roadmap Algorithm(HPPRM), and Spline based Dynamic Window Approach(Splines+DWA). Our comparative analysis of these algorithms highlights their strengths and limitations, providing valuable insights for development of safer and more efficient mobile robots in hospital and complex environments.**

***Keywords* — Rapidly-exploring Random Trees, Probabilistic Roadmap, non-holonomic, path planning algorithms, dynamic environments**

## I. Introduction

Determining a secure and efficient way for a robot to move through an environment with obstacles is known as path planning, and it is a crucial task in the field of robotics. Path planning algorithms have historically relied on mathematical models that assume a predetermined environment and set of constraints. Yet, these traditional approaches frequently find it difficult to deal with the complexity of real-world surroundings, including dynamic obstacles or ambiguous terrain.

Sampling-based path planning algorithms have emerged as a promising solution to overcome these challenges. These algorithms work by randomly sampling the environment to build a graph of possible paths. Three popular sampling-based path planning algorithms are D*, Splines+DWA, and HPPRM.

D* is a dynamic path planning algorithm that updates the path as the environment changes. This algorithm is particularly useful for mobile robots that need to navigate through unknown or partially known environments. D* has been shown to be more efficient and robust than classical path planning algorithms.

Splines+DWA is a path planning algorithm that uses a combination of cubic splines and distance-weighted graphs (DWA) to generate smooth and efficient paths. This algorithm can handle complex environments and has been shown to be faster and more accurate than other sampling-based path planning algorithms.

The Hybrid Potential-based Probabilistic Roadmap Algorithm (HPPRM) is based on a combination of two popular methods for path planning in robotics: the artificial potential field method and the probabilistic roadmap method. The artificial potential field method generates attractive and repulsive forces to guide the robot towards the goal while avoiding obstacles. The HPPRM algorithm uses a hybrid potential function that integrates both attractive and repulsive forces into a single function to better guide the robot towards the goal while avoiding obstacles. The probabilistic roadmap method generates a graph of feasible paths that the robot can use to navigate the environment. The HPPRM algorithm introduces a new sampling strategy that generates more samples in areas of the configuration space that are more likely to lead to a successful path.

## II. Related work

The D* path planning algorithm has been researched to increase its reliability and computational cost. D* improves upon A* by utilizing the edges and vertices of cells instead of relying on the center of cells for the node location. While this may not seem like a big improvement, it allows the robot to travel at different angles other than 45 degrees. D* also has the edge on A* in dynamic or unknown environments, by not recalculating the entire path, just the changed portion, significantly decreasing computational costs. D* Lite provides the same path as D* but utilizes a different algorithm to cut down on computational complexity even further [1]. Instead of building off of A*, D* Light builds off of Lifelong Planning A* to utilize a completely different algorithm. LPA* only recalculates start node costs to nodes that are relevant to finding the shortest path. By decreasing the number of nodes that need to be recalculated, the overall computational cost is decreased. The authors [1] propose several implementations of pseudo-code and show experimental results to prove the efficiency and optimality of their new D* Lite method.

Solving foresighted navigation is essential to introduce robots in collaborative environments like hospitals. This paper[2] introduces a dynamic collision model that is capable of predicting future collisions. Dynamic window approach[3] is a motion planning algorithm used in robotics and autonomous systems. It is an algorithm that generates a real-time trajectory for a robot to follow while avoiding obstacles in its environment. The algorithm uses a dynamic window to represent the possible motion of the robot and a cost function to evaluate the feasibility of different trajectories. The DWA algorithm is known for its ability to handle non-holonomic constraints, making it a popular choice for real-time motion planning problems.

This article [4] proposes a novel approach to motion planning and navigation of mobile robots that combines spline-based motion planning with Dynamic WIndow Approach for velocity control. The authors demonstrate the ability of the algorithm for improved motion control. The authors of this[5] paper propose an improved dynamic window approach, that is based on the robot dynamics model, reducing computational complexity and improving real-time performance.

The HPPRM algorithm [6] generates a roadmap by discretizing the free space of the environment and selecting random points. Then, it applies potential fields to the generated roadmap to create a smooth and obstacle-free path. The algorithm also uses probabilistic roadmap methods to generate alternative paths in case the primary path is blocked. One of the strengths of HPPRM is its ability to adapt to changes in the environment in real-time. The algorithm can update the roadmap dynamically and recalculate the path if there is a change in the environment, such as the appearance of a new obstacle. This feature makes HPPRM suitable for mobile robots that need to operate in dynamic and unpredictable environments. The authors provide simulation results that demonstrate the effectiveness of HPPRM compared to other path planning algorithms. The simulation results show that HPPRM generates paths that are smoother, shorter, and less likely to collide with obstacles than other methods. The authors also provide a detailed analysis of the computational complexity of HPPRM and show that the algorithm is efficient and scalable, making it suitable for practical applications. However, the authors also acknowledge the limitations of the proposed algorithm, including the need for further testing and refinement in real-world scenarios. The authors suggest that future work could include optimizing the algorithm parameters, implementing hardware experiments, and testing the algorithm in various environments.

The proposed algorithm [7] combines the potential field method and the probabilistic roadmap method to generate a safe and optimal path for a robot to navigate through a complex environment with multiple obstacles. The algorithm integrates a virtual force field generated by the potential field method to guide the search process of the probabilistic roadmap method. Simulation experiments show that the proposed algorithm outperforms other path planning algorithms in terms of path length, computation time, and success rate. The paper provides a valuable contribution to the field of robotics and offers a promising approach to path planning in complex environments.

The APBPRM algorithm [8] combines two popular methods: the artificial potential fields method and the probabilistic roadmap method, to generate feasible paths for mobile robots in dynamic environments. The artificial potential fields method is used to generate attractive and repulsive forces to guide the robot towards the goal while avoiding obstacles. The probabilistic roadmap method generates a graph of feasible paths that the robot can use to navigate the environment. The authors introduce a bias term to the probabilistic roadmap method to improve the efficiency of the algorithm by guiding the sampling towards areas of the configuration space that are more likely to lead to a successful path. The authors provide experimental results that demonstrate the effectiveness of the APBPRM algorithm in improving the path planning capabilities of mobile robots, including its ability to handle dynamic environments and its efficiency in generating feasible paths. The paper also compares the APBPRM algorithm to other popular path planning algorithms, such as A* and D*, and shows that it outperforms them in terms of success rate, efficiency, and optimality. The proposed algorithm has potential applications in a variety of fields, including robotics, automation, and unmanned vehicles. Overall, the paper presents a novel approach to path planning for mobile robots that combines two popular methods and improves the efficiency of the algorithm. The experimental results demonstrate the effectiveness of the proposed algorithm and its potential applications in various fields.

## III. Proposed Method

The hospital environments are dynamic with patients, workers, equipment and other obstacles, the safe and efficient mobility of robots is crucial for the prompt delivery of essential supplies, equipment and medication.

We propose testing and analysis of three motion planning algorithms that are well suited for complex dynamic environments with obstacles. The proposed methods include D*, spline based trajectory generation for Dynamic Window approach (Splines+DWA), and Hybrid Potential based Probabilistic Roadmap (HPPRM). D* is an incremental search algorithm that efficiently re-plans a path in a graph when encountering unknown changes. Spline based DWA is a real-time path planning and obstacle avoidance algorithm, and HPPRM can generate smooth, collision-free paths for robots in complex environments with obstacles.

We aim to investigate each algorithm on different performance metrics and try to elucidate the best algorithm, given a scenario in a hospital environment.

### A. *D* Algorithm*

D*(Dynamic A*) Algorithm is an an extension of A*, and is designed specifically for dynamic environments. D* begins at the goal and works its way back to the starting position. This allows the algorithm to adapt more efficiently to the changes in the environment since it can update the cost values for the affected areas more easily. When the robot is encountered with an obstacle or the environment changes, D* updates the cost values and recalculates the optimal path. This incremental approach makes D* more computationally efficient.

#### a. *Initialization*

Just like A* search, D* maintains a priority queue with nodes to be explored in it. It starts by initializing the goal node's cost(g-value) to 0 and that of all other nodes to infinity. The queue is initially populated with the goal node.

#### b. *Reverse Search*

D* begins at the goal node and works its way back to the starting position. It iteratively explores the nodes with lowest priority (based on the sum of g-values and heuristic estimates) until it reaches the start node. During this process, it constructs a tree of back-pointers from the starting node to the goal node, representing the optimal path at each stage.

c. *Path following*

The robot follows the optimal path from the starting node towards the goal node. As it moves, it may encounter a change in the environment, which calls for replanning.

d. *Updating costs*

When the environment changes or is encountered with an obstacle, the algorithm updates the costs of the affected nodes. The costs of neighboring nodes are also updated to account for the new information. Nodes that require updating are added back to the priority queue.

e. *Replanning*

The algorithm continues the reverse search, exploring the nodes with the lowest priority in the updated priority queue. This process continues until the starting node is reached again, and a new optimal path is found.

## B. *Hybrid Potential based PRM (HPPRM)*

The HRM algorithm is a combination of the Artificial Potential field and the Probabilistic Roadmap Algorithm. The APF works with two important parameters that are 'Influence Coefficient' and 'Repulsive range'. The influence coefficient represents how the obstacles on the map influence the robot's path to the goal position and the repulsive range represents the range in which an obstacle can repulse a path.

a. Repulsive Potential

The number of sampling points is reduced by the adaptive sampling point method based on obstacle density to match workplace obstacle density, preventing an excessive number of sampling points from leading to an ineffective roadmap construction. Additionally, too many sampling points will increase the computational complexity, which will lead to poor performance.

The repulsive potential U rep (q) produced by obstacle q o for each location q on the map is calculated using

$$u_{\text{rep}}(q) = \begin{cases} \frac{1}{2} k_{\text{rep}} \left( \frac{1}{q - q_0} - \frac{1}{p_0} \right)^2 & q - q_0 \leqslant p_0 \\ 0 & q - q_0 > p_0 \end{cases} \quad (1)$$

where, $p_o$ is the repulsion field range of the obstacle.

Equation (1) was used to compute the repulsive potential for each location in an open section of the map. Open area and obstacle region sampling points can be scattered over the map by using bounding range. By guaranteeing a high density of sample points around the obstructions that generate narrow paths, this distribution technique overcomes the PRM constraint and enhances the connectivity of the graph while providing excellent connectivity of narrow paths.

b. Path generation

The sampling points generated from APF are passed on to the PRM algorithm that generated the sampling coordinates. The algorithm then checks for obstacles, stores an array of coordinates that are free of obstacles, and connects the nearest neighbor of the collision free path using the Nearest neighbor library. In addition, the PRM uses Dijkstra [9] to find the shortest path between the start and goal from the collision free coordinates generated by the PRM.

## C. *Splines+DWA*

DWA is a real-time planning algorithm used for mobile robots in dynamic and uncertain environments, which makes the method suitable for local planning in complex hospital environments. Unlike other methods tested in this project, DWA does not require a map or occupancy grid of the environments or information about the obstacle. The algorithm chooses the best trajectory from potential candidates based on a cost function.
At each time step , the algorithm evaluates each potential trajectory within a predefined "dynamic window" that takes into account the robot's maximum velocity and acceleration.

a. *Kinematic model of the robot*

The kinematic model of a mobile robot can be represented as follows:

$$x(t_n) = x(t_0) + \int_{t_0}^{t_n} v(t) \cdot cos\theta(t) dt \quad (2)$$

$$y(t_n) = y(t_0) + \int_{t_0}^{t_n} v(t) \cdot sin\theta(t) dt \quad (3)$$

$$\theta(t_n) = \theta(t_0) + \int_{t_0}^{t_n} \omega(t) \cdot dt \quad (4)$$

Where pose of the mobile robot with 3-DOF is *(x,y,Θ)*. v(t) is the translational velocity and $\omega$(t) is the angular velocity of the robot with respect to time, t.

b. *The Dynamic Window Approach*

The search for commands controlling the robot is carried out directly in the velocity space. The dynamics of the robot is incorporated into the algorithm by reducing the search space into velocities reachable under dynamic constraints. In addition to this, only velocity that does not collide with obstacles are considered.

The DWA considers only circular trajectories uniquely determined by pairs $(v, \omega)$of translational and rotational velocities, this gives 2D search space.

The use of only admissible velocities ensures safe trajectories are considered. A pair $(v, \omega)$ is considered admissible, if the robot can stop at a safe distance from obstacles.

The dynamic window can ensure the selection of velocities to those that can be reached within a given time interval(dynamic window) and given acceleration limits.

*c. Prediction of the future positions*

The algorithm generates a set of possible velocity controls, and for each velocity control, it predicts the robot's position and orientation at some future time , t+*T,* where T is the planning horizon (steps to plan ahead). The predicted pose can be predicted using the kinematic model of the robot.

*d. Cost function*

The cost function balances the robot's progress towards the goal and its proximity to the obstacle. The cost function can be expressed as follows:

$$G(v,\omega) = \sigma(\alpha \cdot heading(v,\omega) + \beta \cdot dist(v,\omega) + \gamma \cdot vel(v,\omega)) \qquad (5)$$

Where $\alpha$ and β, $\gamma$ are the weight for each term.
**Target heading (*heading*)** is a measure of the progress towards the goal location. It is maximum if the robot moves directly towards the target.
**Clearance (*dist*)** is the distance to the closest obstacles on the trajectory.The smaller the distance to the closest obstacle the higher the robot can move around.
**Velocity(*vel*)** is the forward velocity of the robot and this term supports fast movements.
We tested this cost function, and noticed that sometimes, the robot gets stuck in local minima. Hence we tested different combinations of cost functions from other papers (*see the section IV.B*).

*e. Spline Optimization*

Combining DWA with spline based trajectory generation can produce smoother paths and better understanding of the robot's kinematic constraints.
For this project, we've tried out B-splines and cubic splines for trajectory generation.
B-splines(Basis Splines) generate smooth continuous paths or trajectories. These are piecewise defined polynomial functions that allow for local control and smooth transitions between control points. The B-spline curve is defined by a set of control points and a degree.
A cubic spline is a piecewise cubic polynomial function that passes through a set of control points, with smooth transition between the polynomial segments. Cubic splines are used for interpolation and pass through all control points, while B-splines are generally used for approximation and do not necessarily pass through the control points(except for interpolating B-splines). We have used interpolating B-splines and unlike cubic splines, these can have any number of degrees.

## IV. Experiments

To validate and compare the proposed methods outlined in section III (Proposed Method) each of the motion planning algorithms will be tested in a simulation depicting a real life application. The robot platform that is being simulated is a nursing assistants robot with a mobile robot base. The physics simulation software being used to check collisions and receive sensor data from is Unity. Both the simulation software and the robot extend upon research done by WPI's Human-Inspired Robotics Lab.

### A. *Dynamic window approach as a complete planner*

DWA is a real time local planner. Through this experiment we attempted to identify the challenges for DWA to become a complete planner. The research question for this experiment was that, "Can Splines+DWA can plan without a global planner? ".

We attempted to fine-tune the cost functions with different weight values. The original objective function, trades-off between the goal heading and distance from obstacles. The common problems with DWA was that it get stuck in local minima with either of the four following reasons:

*a. Forward heading outweighs velocity*

Optimizing hyperparameters can be challenging. The environment we choose for this project, a hospital environment, is complex even with static obstacles, like hospital equipment. If the goal is just behind a wall-likeFig 11. Path generated by spline based DWA in a toy problem with dynamic obstacles. obstacle, the robot gets stuck because the forward heading function is maximized(fig1.a).

*b. Obstacle cost is much higher*

If the obstacle cost is large, the robot finds it harder to squeeze through narrow gaps or doors.

*c. Obstacle cost is lower*

It is generally not ideal to lower the obstacle cost below a certain limit, this would cause the robot to run over an obstacle in simulation. This is not a practical scenario and the planner would fail in the real world(fig1.b).

*d. Velocity cost is higher*

If velocity cost is higher, the robot would not get stuck, but rather tries to move out of local minima by choosing the trajectory with highest velocity and hence it would move around in a neighborhood of the local minima(fig1.c).

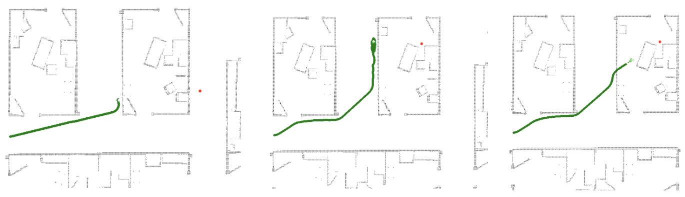

Fig1**.** The figure depicts three scenarios in which the DWA fails without a global planner. (a) Forward heading is higher (b) Velocity cost is higher (c) Obstacle cost is lower.

We optimized each parameter, and found that the Algorithm was not complete. In some cases a certain hyperparameter should be greater than the other.

## B. *Objective function optimization*

The original cost function weights the heading towards the goal and obstacle cost, to prevent the robot from colliding with obstacles. In addition to the original cost function or objective function used in the DWA paper, we tried three different functions in combination with it. The experiment was done without a global planner inorder to test different cost functions without bias and identify the reason for the failure of these cost functions.

| *Cost Function* | *Percentage Completeness* |
|---|---|
| OG | 30 |
| OG- HT | 20 |
| OG+VD | 20 |
| OG-OD | *5* |

**Table 1:** The results from different objective function testing, please note that each function is tested without a global planner in the complex hospital environment. The results are from trials of 20 different local environments in the hospital. HT, VD, OD are the angle heading term, velocity difference, and Obstacle density respectively.

In Table1, OG is the original cost function, other terms are either added or subtracted from the original cost dispensing on the nature of their usefulness. Surprisingly, the original objective function works better, when used alone. This might be because the hospital environment is complex and has many local minima. We noticed that each term has its own advantage depending on the nature of the local minima. For example, Obstacle density works better if the local minima is in an obstacle dense region and the goal position is in a free space.

Angle heading term(HT) measures the angle between the current heading of the robot and target heading. It gives an angular difference between robot and goal. Velocity difference(VD) measures the absolute difference between left and right wheel velocities, it gives a sense if the robot is stuck or not. Obstacle density(OD) checks if the closest obstacle to the robot is in an obstacle dense area.

## C. *Simulation*

The robot will be placed in a simulation hospital outfitted with a variety of static obstacles, like beds and chairs, and dynamic obstacles, like people. The robot will be tasked to move from one room in the hospital to another with various stages of obstacles. Each stage will increase in difficulty to find if different methods perform better under different parameters. First the robot will move through a known static environment, followed by a partially known static environment. In the second layout the robot may know the layout of the walls, but not where the furniture is, mirroring a scenario where a building staff likes to redecorate often. Two more environment configurations will be used that are the same as the first two, but include dynamic obstacles as well. The final two environments bring the simulation closer to a real-life application as people are often in a hospital and are known to walk around.

## D. *Evaluation*

Through each of the tasks outlined above, the proposed methods will be compared in order to rank each of them comparatively. The metrics we will compare are: optimality, completeness, safety, computational efficiency, and robustness. Optimality will be compared by comparing the robot's travel time across runs as well the generated path's length. Riskier paths that cut corners to decrease a path length may trigger recovery behavior that lengthen the robots driving time. Some of the proposed algorithms also create smoother paths that allow the robot to travel faster, so both the time to follow a path and the overall length must be considered for path optimality. Completeness can be determined mathematically before planning any paths, but in practice algorithms that are probabilistically complete may not always find a path. Algorithms that depend on random sampling or any type of random number generation can get lucky, so they must be tested for repeatability and completeness. In a hospital environment, it is important that the robot does not cause injuries or knock over equipment, so the number of collisions the robot has should be zero for any task. The computation time of the algorithm is less important with known static obstacles, but will be an increasingly interesting metric as dynamic obstacles become present. The Robustness of the robot will be checked by running the different tasks. Some algorithms may do better with static or dynamic obstacles, but the best one for the proposed purpose must be the best overall.

| | **D*** | **HPPRM** | **Splines+DWA** |
|---|---|---|---|
| Optimality | Yes (number of nodes) | Yes (Needs to be smooth) | No |
| Completeness | Yes (depends on resolution) | Yes | No/depends on global planner |
| Safety | Yes | Yes | Yes |
| Computational efficiency | No | No | Yes |
| Robustness | Yes | No | Yes |

**Table 2:** Evaluation table

## V. Results

In the process of setting up the simulation software, several packages were installed for localization, mapping and path planning. The motions planning, while naive, was able to show a reasonable baseline for automated robot movement. Utilizing the ROS package move_base, a target pose for the robot can be chosen on the map of the hospital in RVIZ. The move_base path planner uses a global planner based on heuristics and cost to create the initial path. The path avoids obstacles by creating a boundary layer around each obstacle to define the configuration space. The move_base path planner then uses a local planner that creates a cost based path to stay as close as possible to the global path. A local planner allows the robot to encounter unknown or dynamic obstacles and have a chance to recover. However, even with static and known obstacles, the move_base path planner has intersected into the bounding area it set around obstacles and had to enter recovery behavior. These preliminary tests highlight the need to simulate path planning methods. While the optimal path was found by move_base, path feasibility can affect the robots performance.

D* was implemented in the Unity simulation. While the robot was moving through the simulation, however, the computational cost of the simulator was slowing down the computer and causing ROS messages to build up. The robot soon began to give control messages that were delayed from the robots true position and caused the robot to crash into walls. Due to the problems in computation power presented by the unity simulation, a complex 2D problem was created to mimic the real life conditions and provide a path that the move_base package could utilize. In Figure 2 the path generated by D* in the hospital environment can be seen to successfully reach the goal target in red from the robots starting position in green. That simulation included unknown static obstacles and dynamic obstacles. The robot was successfully able to navigate around these obstacles without crashing and reach the goal position.

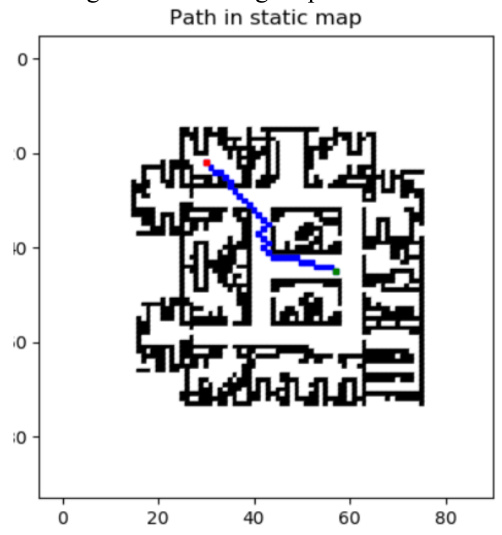

Fig 2. path generated by D* in the hospital

Through implementing D* in a realistic environment such as the hospital found in the Unity setup, we have come to realize the limitations of grid based search algorithms. The hospital covers a large area and contains many doorways and hallways that are narrow passages for the robot to traverse through. In order to keep the optimality and completeness of D*, the resolution of the map of the world has to be precise enough to measure these narrow passages. Overall assuming that the resolution parameters have been tuned correctly, D* is both complete and optimal when path planning. To test the completeness of the algorithm, the robot was given a goal to each of the separate rooms in the hospital. Out of the 14 different rooms it was able to find a path to all of them. Through our experiments we found that D* was able to find a path in all the scenarios tested, but the paths found were not always optimal. While D* found a path in the least number of nodes, the paths would sometimes be 'wiggly' and take winding diagonals. By running the algorithm 10 times at different resolutions with the same start and end goal the following table was made.

| Resolution | # Nodes | Optimal # Nodes | Distance | Optimal Distance |
|---|---|---|---|---|
| 0.10 m /grid cell | 50±0 | 50 | 6.1±0.4m | 5.646m |
| 0.25 m/ grid cell | 19±0 | 19 | 5.8±0.3m | 5.536 |
| 0.45 m /grid cell | 13±0 | 13 | 6.9±0.45 m | 6.518m |
| 0.50 m /grid cell | 11±0 | 11 | 6.4±0.4m | 6.035 m |

**Table 3**: D* resolution against path optimality in terms of node length and distance in meters

The safety of D* was mainly determined by the local path planner as well as the tuning of the configuration space along with the map resolution. Since the algorithm does not inherently determine its own safety rating it should be regarded as less safe than the local path planner. While in the experiments D* did crash, it was due to a poor local planner and ROS message buildup. Overall, as a global path planner instead of a local path planner, it should be regarded as less safe than other algorithms that take local path planning into consideration.

In order to have the best odds of optimality, completeness, and safety, the resolution has to be high. The result of a high resolution on a large map is a computationally expensive search algorithm. The first stages of D* involves creating a value for each square on the grid that acts as a gradient towards the optimal path. While this helps it rewire and repair paths in the presence of unknown and dynamic obstacles, it dramatically increases computational time when looking at a large workspace. As seen in Table 4, as the map resolution gets finer to more accurately depict obstacles, the time to calculate the path gets exponentially longer. Through testing the algorithm for completeness, it was found that the computational time did not significantly change based on the length of the desired path. Overall as a grid based path planner, D* is not suited towards applications that require short computational times without adjusting the workspace area or the algorithms efficiency.

| Grid Resolution | Path computation times |
|---|---|
| 0.05 m /grid cell | 3 hours 4 minutes |
| 0.10 m /grid cell | 35 minutes 29 seconds |
| 0.25 m/ grid cell | 1 minute 34 seconds |
| 0.45 m /grid cell | 17 seconds |
| 0.50 m/grid cell | 15 seconds |

**Table 4**: D* initial path computation times for each resolution tested

Robustness was measured through testing the algorithm in several different environments. The algorithms were tested in a static environment with all obstacles known, a static environment with unknown obstacles, a dynamic environment with known obstacles, and a dynamic environment with unknown obstacles. Through testing the different environments we were able to see the algorithms performance in the face of known and unknown dynamic obstacles. D* was able to recover when faced with unknown obstacles and efficiently reroute the robot on the next most efficient path. This change was able to happen quickly and efficiently due to the preloading of a large majority of the computational effort. Overall D* was able to find paths in all four of the different environments, showing good robustness.

For the HPPRM algorithm, as the map began to get more complex and introduced several narrow passages, the computation complexity increased. However, when trying to navigate through a map of the WPI campus, the algorithm was able to find a path, but after a good amount of time. The map resolution played an important role for computational efficiency. When the same map was resized to a lower resolution, the algorithm was able to compute a path faster. When the algorithm was tested with a sample map that was provided with the Gopher ROS-Unity repository, it didn't perform as expected. The map had thin walls that resulted in thinner obstacles when mapped. The path that was generated went through the walls. Therefore the algorithm failed in generating a path in that map. Figure 3 shows the incorrect path that was generated for the sample hospital map.

Then the algorithm was tuned with parameters like repulsion range and influence potential that affects the reactability towards obstacles and then got a better result. The same sample map was then tested and a correct path was generated as shown in Figure 5. The algorithm was then tested with the same hospital map but with added circular obstacles that were assumed to be humans. The algorithm was still able to find an optimal path to the goal as shown in Figure 7. The hospital maps were tested with the start point to be at [50,10] and the goal to be at [50, 200]. The map was modified again by adding rectangular obstacles as shown in Figure 8, that were assumed to be the hospital beds and couches in each room. The algorithm was able to find a feasible path as shown in Figure 9.

Optimality and Robustness of the path generated was promising for all the maps, but smoothness of the path was not satisfactory. Further works in developing this algorithm include generation of smooth paths and reconfiguring the algorithm for dynamic environments.

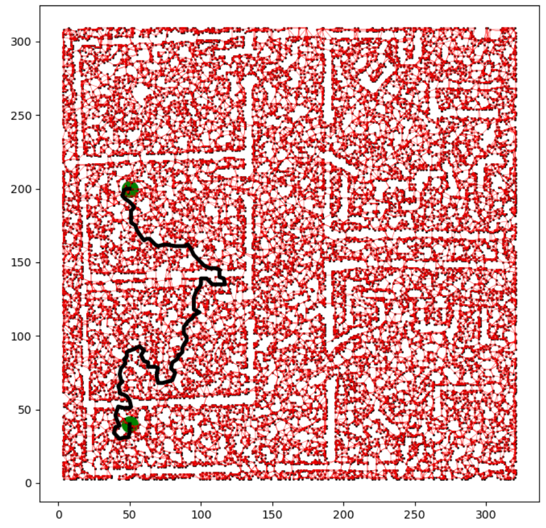
Fig 3. The incorrect path generated by HPPRM

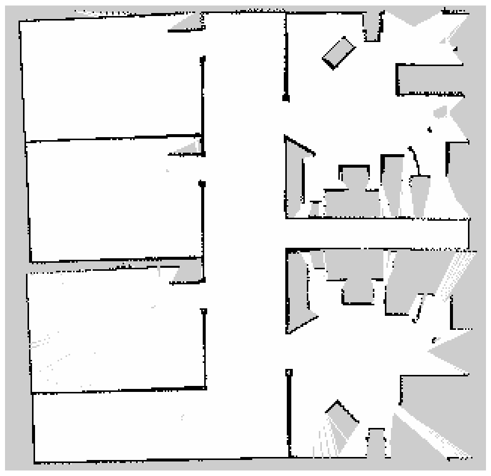
Fig 4. Sample hospital map

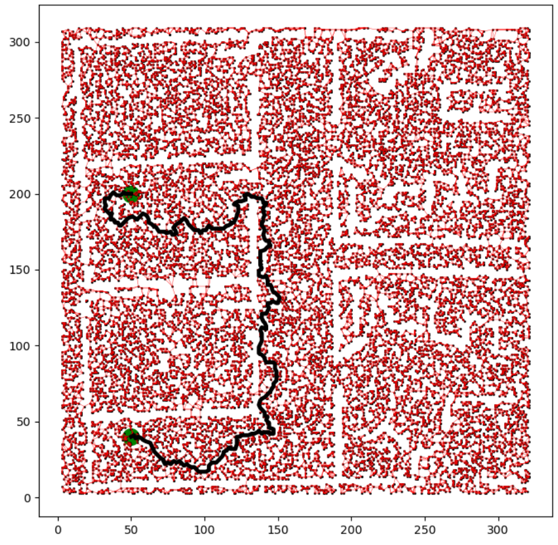
Fig 5. Correct path generated from start to goal.

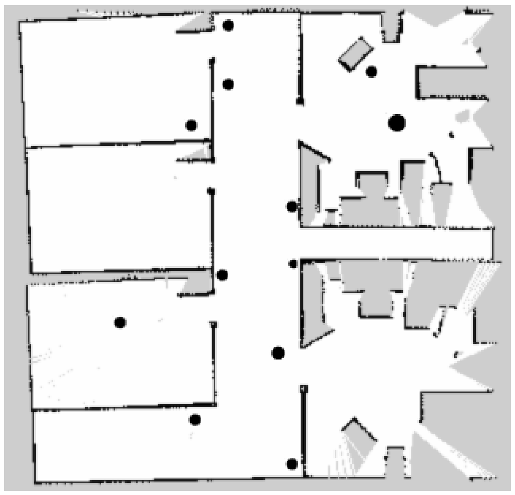
Fig 6. Sample hospital map with circular obstacles

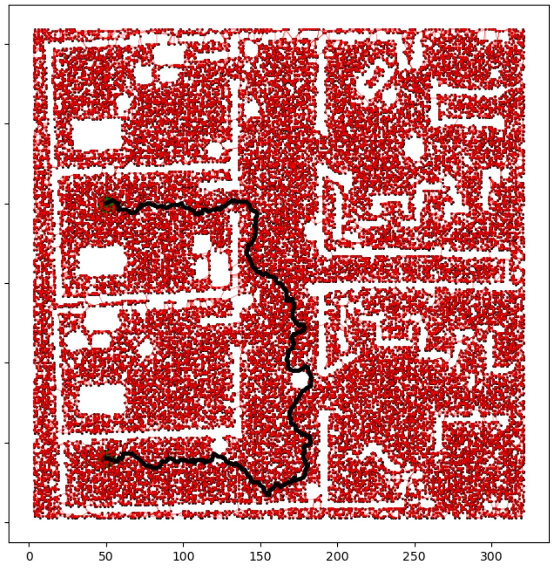
Fig 9. Path generated for the modified map.

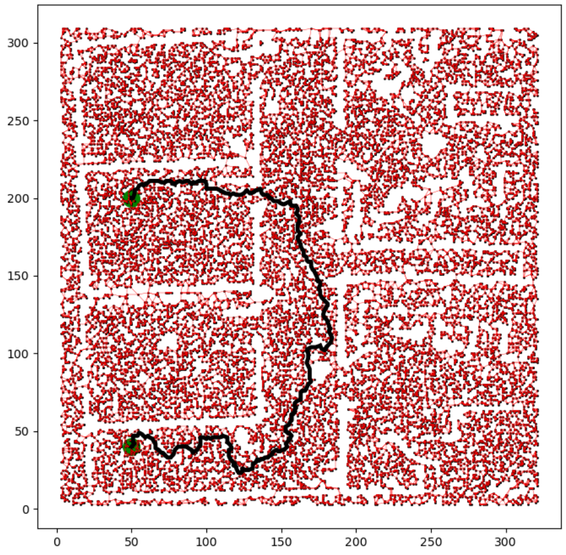
Fig 7. Path generated when circular obstacles were added to the map.

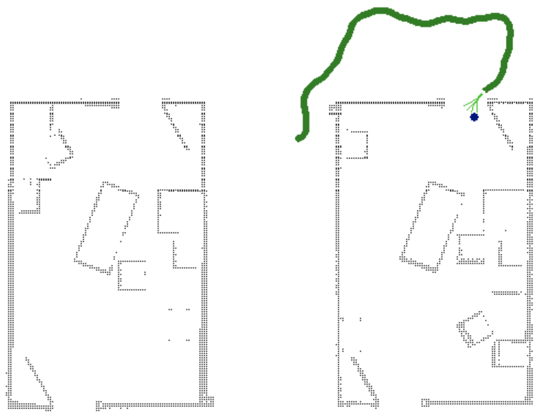
Fig 10. Path generated by DWA as local planner with RRT

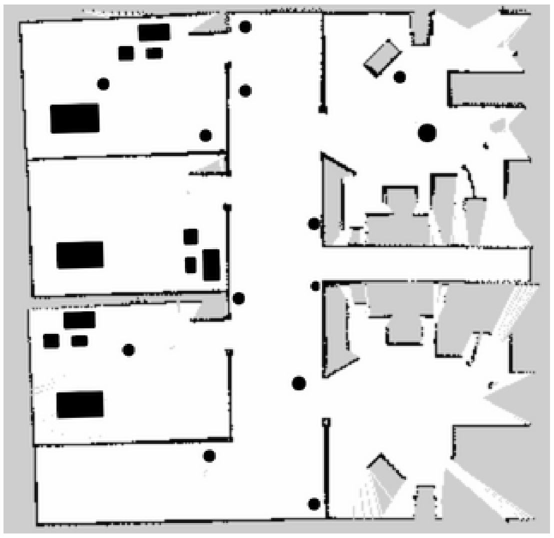
Fig 8. Path generated when rectangular obstacles were added to the map.

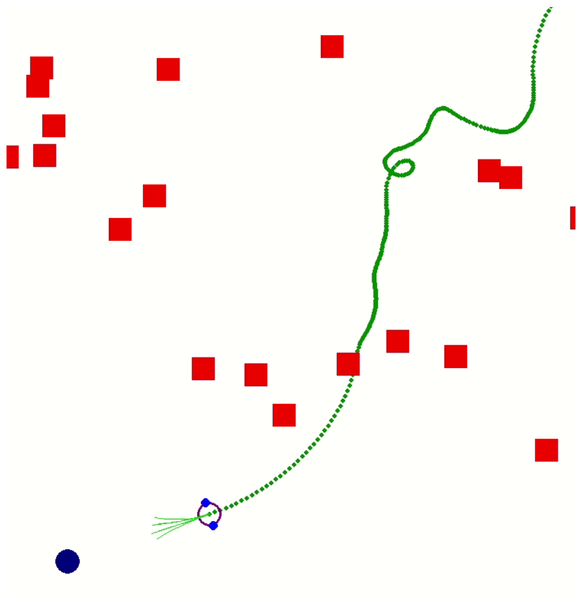
Fig 11. Path generated by spline based DWA in a toy problem with dynamic obstacles.

For the spline based Dynamic window approach, we created and tested three different versions of the DWA algorithm, including the original version. Two different spline trajectory generation methods have been tested, namely B-spline and Cubic spline. Although, cubic and B-spline methods showcase similar results. It is evident that spline optimization of the generated trajectory improved the performance. Based on the optimization parameters we have noticed a trade-off between optimality and safety. For the Cost function of the DWA, called 'obstacle weight' and 'forward weight' determines this trade-off.

Unlike D* and HPPRM, using a occupancy map is redundant for DWA, since it directly deals with the robot trajectory, i.e. pose and velocity of the robot.

Based on the experimental results and evaluation we learned that objective function is often overlooked. We tried different objective functions from literature and found that for a complex environment such as that of a hospital, the real-time planner may fail due various reasons. The objective function could take care of it, only if the hyperparameters are assigned dynamically. From simulation, we also observed that the algorithm performs very well with dynamic obstacles as the environment changes. We Identified that spline based DWA can be beneficial for local reactive planning in dynamic environments. Figure 11 shows the path taken by a mobile robot using DWA for obstacle avoidance. DWA can be incorporated with a global planner for planning in complex environments. Figure 10 shows that the robot planned a path using RRT as global planner and DWA as a local planner.

## VII. References

[1] Koenig, Sven, and Maxim Likhachev. "Fast replanning for navigation in unknown terrain." IEEE Transactions on Robotics 21, no. 3 (2005): 354-363

[2] M. Missura and M. Bennewitz, "Predictive Collision Avoidance for the Dynamic Window Approach," 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 2019, pp. 8620-8626, doi: 10.1109/ICRA.2019.8794386.

[3] Fox, D., et al. "The Dynamic Window Approach to Collision Avoidance." IEEE Robotics & Automation Magazine, vol. 4, no. 1, Mar. 1997, pp. 23–33, https://doi.org/10.1109/100.580977.

[4] B. Demir, F. Hoeller, F. Garcia Rosas, D. Schulz and N. Goerke, "Spline-Based Robot Trajectory Generation Using the Dynamic Window Approach," 2019 Third IEEE International Conference on Robotic Computing (IRC), Naples, Italy, 2019, pp. 9-16, doi: 10.1109/IRC.2019.00011.

[5] Feng, D., Deng, L., Sun, T., Liu, H., Zhang, H., Zhao, Y. (2021). Local Path Planning Based on an Improved Dynamic Window Approach in ROS. In: Liu, Q., Liu, X., Shen, T., Qiu, X. (eds) The 10th International Conference on Computer Engineering and Networks. CENet 2020. Advances in Intelligent Systems and Computing, vol 1274. Springer, Singapore. https://doi.org/10.1007/978-981-15-8462-6_133

[6] Ankit A. Ravankar, Abhijeet A. Ravankar, Takanori Emaru, and Yukinori Kobayashi. "HPPRM: Hybrid Potential Based Probabilistic Roadmap Algorithm for Improved Dynamic Path Planning of Mobile Robots". IEEE Access, vol. 8, 2020, pp. 221743-221766.

[7] Y. Zhang, L. Zhang, L. Lei and F. Xu, "An Improved Potential Field-Based Probabilistic Roadmap Algorithm for Path Planning," 2022 6th International Conference on Automation, Control and Robots (ICACR), Shanghai, China, 2022, pp. 195-199, doi: 10.1109/ICACR55854.2022.9935557

[8] D. Aarno, D. Kragic and H. I. Christensen, "Artificial potential biased probabilistic roadmap method," IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004, New Orleans, LA, USA, 2004, pp. 461-466 Vol.1, doi: 10.1109/ROBOT.2004.1307192

[9]https://gist.github.com/betandr/541a1f6466b6855471de5ca30b74cb31