Implementation of Collision avoidance algorithms for Gopher Robot

Durga Prakash Karuppannan
Robotics Engineering
Worcester Polytechnic Institue
Worcester, USA
dkarupannan@wpi.edu

Pranav Moorthy
Robotics Engineering
Worcester Polytechnic Institue
Worcester, USA
pmoorthy@wpi.edu

Sailesh Rajagopalan
Robotics Engineering
Worcester Polytechnic Institute
Worcester, MA
srajagopalan@wpi.edu

***Abstract*—Motion Planning in dynamic environment is a common problem faced in Robotics which has been solved using various approaches and algorithms. The main issue with motion planning in dynamic environments is that the position of obstacles changes with respect to time and thereby it is computationally expensive to plan a path for the robot for every position of the obstacle. Some of the approaches are discussed in detail. We prop to implement various algorithms on the gopher robot in Unity simulations and compare them to the A* algorithm based on the comparison metrics that have been discussed below.**

Keywords—Collision Avoidance, Velocity obstacle, Acceleration Velocity Obstacle, Non-Holonomic Robot, A* algorithm

I. INTRODUCTION

Motion Planning in dynamic environments we face major problems in means of planning in the state space, where we will have to solve both the path and velocity planning. Path planning is defined as a kinematic problem wherein we define methods and solve the robot's motion from start to goal, but the velocity planning is dynamic in nature. In reference, first motion planning was addressed by time dimensions to the configuration space of a robot. in nature. In reference, first motion planning was addressed by time dimensions to the configuration space of a robot. Real-time navigation in dynamic environments has multiple methodical approaches which can define both global and local planners, identifying controls that enable collision with a moving obstacle that is present in space. The ability of approaches to defining a collision-free path for a robot in a dynamic environment avoiding obstacles is very important in robotic applications and is currently used in various scenarios such as warehouses, robotic wheelchairs, car-like robots. The main problem with motion planning when dynamic obstacles are involved are that the position of the obstacles change continuously with respect to the environment it is present in. In this paper we have described in detail the related work which has provided us with suggestions for our methodical approaches which have been explained. This paper talks in detail on approaches and comparison metrics with toy problems to provide an overview onto how the implement in Unity the planners that have been discussed in detail in furtther sections.

II. RELATED WORK

A. Motion Planning in Dynamic Environments using Velocity Obstacles

Velocity obstacle single and multiple objects are researched in this paper selecting avoidance maneuvers from the static and dynamic obstacles in the velocity space. In avoidance maneuvers, for reachable avoidance velocities, we can map the reachable velocities over Δt by mapping the actuator constraints to acceleration constraints. The set of reachable velocities is defined as the difference between the reachable and the velocity obstacles,

$$\mathrm{RAV}(t+\Delta t) = \mathrm{RV}(t+\Delta t) \ominus VO(t)$$

Computing the avoidance trajectories, that avoid establishing static and moving obstacles, and satisfying the dynamic constraints by performing a global search for the offline application and a heuristic search for online applications, The al search can be represented by the reachable avoidance velocities of every segment of the trajectory will avoid the obstacles that are defined within reachable parameters in a time set the tree representation for the global search, we can assign a cost to each branch which are the trajectories. The obstacles avoided in the previous can be generated in a shorter time horizon or minimal trajectories. (Fironini,1995).

Heuristics are defined by providing it to satisfying the prioritized series of goals, for example defining the survivability as a primary goal or reaching desired targets and selecting a structure for trajectory approaches and defining secondary goals as it. In this paper, heuristics has been defined as

1. The velocity chosen was the highest avoidance velocity to the goal – to goal
2. The angle from the line to the goal should be constrained when selecting the maximum avoidance velocity - Maximum velocity
3. The velocity chosen is highest among the rear avoidance velocities.

The paper provides a simple approach by geometric representation for avoidance maneuvers, the reachable avoidance maneuvers for the trajectory which are excluded can be recomputed in a later stage, the avoidance works well on dynamic and stationary obstacles, while in accepting the robot dynamics and actuator constraints that are present. The avoidance maneuvers are defined by representing the obstacles in a velocity space velocity chosen in the scenario is the highest among rear avoidance velocity.

B. Generalized Velocity Obstacles

The velocity obstacle concept for its ability to handle robots with kinematic constraints is approached in a multivariate way by formulating the velocity obstacle, In major cases, the generalized velocity obstacle approach is common in navigating car-like robots in dynamic obstacle environments. y iteratively sensing and avoiding obstacles, the methodology is incorporated in a global planner to

compute the collision-free paths in an incremental process. Prior work on its aspects has been approached by developing ac complete trajectory-based planner moving among obstacles by generating a feasible path and planning a velocity profile to safely traverse the path. An alternate method that could be used to devise it is by taking new sensor inputs as they arrive and planning locally, based on velocity obstacles. A similar approach which was detailed more by working extensively on velocity obstacles was the Reciprocal velocity obstacles which identify and provide a solution to the n-body collision avoidance problem.

The generalized velocity obstacle concepts address the difficulty behind using the velocity obstacles with kinematically constrained agents such as car like robots. In this paper, it is defined as an obstacle *B*, the position at time *t,* and the position of the agent *A,* after undertaking *u* for an interval *t* is *A(t,u).* The obstacle in the control space is defined by

$$\{u \mid \exists t > 0 :: \|A(t,u) - B(t)\| < r_A + r_B\}.$$

In scenarios when a closed form solution is not obtained, a sampling scheme is used, Controls in *U* are sampled, in each control *u,* the minimum achievable distance is calculated.

Navigating velocity obstacles for car kinematics, with the coordinate position of the robot, and its orientation the kinematic constrains are derived.

$$\begin{aligned} x'(t) &= u_s \cos\theta(t), \\ y'(t) &= u_s \sin\theta(t), \\ \theta'(t) &= u_s \frac{\tan u_\phi}{L}, \end{aligned}$$

The speed, steering angle and the wheelbase *L* are the elements which yield an expression for the position of the car at time *t.*

$$A(t,u) = \begin{pmatrix} \frac{1}{\tan(u_\phi)} \sin(u_s \tan(u_\phi) t) \\ -\frac{1}{\tan(u_\phi)} \cos(u_s \tan(u_\phi) t) + \frac{1}{\tan(u_\phi)} \end{pmatrix}.$$

An optimization procedure, among multiple objects, *u** is the robot control if no moving obstacles are present in the environment. In this paper the preferred control is the control that enables in successfully bringing the agent closer to its goal, ignoring the local minima. The control *u* given to the robot is

$$u = \underset{u' \notin \bigcup_{B_i} VO_{A|B_i}}{\arg\min} \|u^* - u'\|.$$

At discrete time steps *t* for *n* controls and its parameters, it is determined as if the control id outside all of the velocity obstacles makes it feasible ad will be tested with the current available feasible control in terms of distance from the preferred control **. The algorithm presented enables in faster navigation for carlike robots and allows in testing an entires control space for a feasible control as most others work based on constrained entities.

C. Reciporcal Collison Avoidance with Accelartion-Velocity Obstacle

The paper is considered as a broader effort to provide a more on collision avoidance for mobile robots, taking into acceleration. The navigation of a single robot amongst moving obstacles was present in this scenario reciprocally moving obstacles while navigating in a common workspace. The concept of Acceleration velocity -obstacle (AVO),

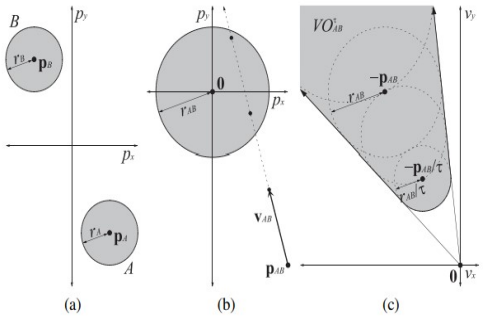


Figure 1: Obstacle position for A and B

In the above image, the configuration A and moving obstacle B, the evolution of their relative position over time when the relative velocity Vab is applied, A and B collide as soon as they are closer to each other than is the sum of their radii. , In the image (c) with velocity ob over a time horizon.

Acceleration velocity obstacle, to ensure the robot smoothly arrives at its new velocity we use proportional control of the acceleration, given by

$$\mathbf{a}_A(t) = \dot{\mathbf{v}}_A(t) = \frac{\mathbf{v}'_A - \mathbf{v}_A(t)}{\delta},$$

We can infer accordingly that from a configuration of the robot amidst obstacles indicting current velocities, over a time horizon by the obstacles, new velocities, v'A for A, There also exists a intersection which is created by the boundaries of the two velocity-obstacle. The robot can imply an acceleration over a time cycle repeating over some time as the difference in velocities over a control parameter which is rated with time. The algorithms tuning parameters primarily include the time horizon τ and the control parameter δ, but too high a value of the control parameter diminishes the new velocities and causing the robot to not navigate the environment, so as defined the good heuristics is as follows which covers for all velocities considered.

$$\delta = \frac{2v^{\max}}{a^{\max}}$$

D. The Hybrid Reciporcal Velocity Obsactle

Hybrid reciprocal velocities obstacle for a Collison free navigation of multiple robots or virtual agents, each robot explicitly considering the factor that since its reciprocal the other robots sense their surrounding and changing the trajectories.

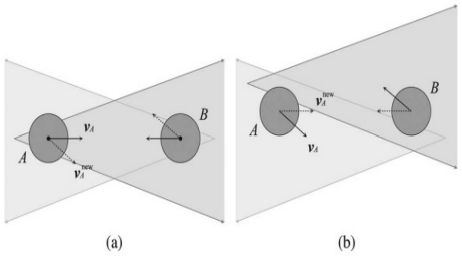


Figure 2: Relative Velocity position for A and B

In the above scenario two robots oscillating velocity is because of using velocity obstacles, the velocity closest to their current velocity closest to that induces by the other robot, in (b) for each time step, the previous velocity outside the velocity obstacle returns the velocity to that following time step

$$VO_{A|B} = \{\boldsymbol{v} \mid \exists t > 0 :: t(\boldsymbol{v} - \boldsymbol{v}_B) \in \mathrm{D}(\boldsymbol{p}_B - \boldsymbol{p}_A, r_A + r_B)\}$$

The overall approach, works y with a set of dynamic or static obstcaleoth a set of dynamic or static obstacles with each robot defined as A and the current position as pA, the velocity as vA and the radius as rA , the preferred, works y with a set of dynamic or static obstcaleoth a set of dynamic or static obstacles with each robot defined as A and the current position as pA, the velocity as vA and the radius as rA , the preferred, works y with a set of dynamic or static obstcaleoth a set of dynamic or static obstacles with each robot defined as A and the current position as pA, the velocity as vA and the radius as rA , the preferred, works y with a set of dynamic or static obstcaleoth a set of dynamic or static obstacles with each robot defined as *A* and the current position as *pA,* the velocity as *vA* and the radius as *rA* , the preferred velocity is defined as:

$$\boldsymbol{v}_{A_i}^{\text{pref}} = v_{A_i}^{\text{pref}} \frac{\boldsymbol{p}_{A_i} - \boldsymbol{p}_{A_i}^{\text{goal}}}{\|\boldsymbol{p}_{A_i} - \boldsymbol{p}_{A_i}^{\text{goal}}\|_2}$$

Using clearpath efficient geometric algorithm we can compute the velocity, which applies to many variations of velocity obstacles. With the uncertainty in radius position and velocity, using Kalman filter provides an estimate of variance and covariance of the measured quantities, As a conclusion, this paper first assumes velocity outside all hybrid reciprocal velocity exists but reduce it only to dense case scenarios incorporation static obstacles does not allow for navigation through narrow passages.

III. PROPOSED METHODOLOGY

*A. A**

A* is a global planner pathfinding algorithm which is widely used, the major aspect of the algorithm is understanding,
f = *g* + *h*
f – total cost of node
g – the distance between the current and start node
h – The heuristic, which is the estimated distance from current to end node.

The concept of breadth first search is very important concept to be able to understand A* algorithm the graph which is a data structure which provide the neighbors for each graph location and the weighted graph t=rough which we can get a cost to move along the path, Locations are label that are present in the graph which includes various other information, the Search section is when the algorithm starts traversing the graph, the queue is a data structure which is used by the search section to decide the order in which the process is done.

B. Velocity Obstacle

Velocity obstacle is an approach that is used in cases of dynamic obstacles [1]. When a robot is moving towards a dynamic obstacle, The robot is reduced to point size and the obstacle is expanded by the sum of the radius of the robot and the obstacle. The obstacle is made stationary, and the robot is given the relative velocity of the robot w.r.t obstacle. The relative velocity cone given below is the set of all relative velocities of the robot A which would collide with the robot A which would collide with obstacle B. It is expressed mathematically as

$$CC_{A,B} = \{V_{A,B} | L_{A\beta} \cap B \neq \phi\}$$

Where:
$L_{A\beta}$ is the line connecting A and B. The collision cone is the set of all relative velocities $V_{A,B}$ of A with respect to B which would result in a collision between A and B. $CC_{A,B}$is the collision cone.

The obstacle is avoided by selecting a relative velocity outside the relative velocity cone
.

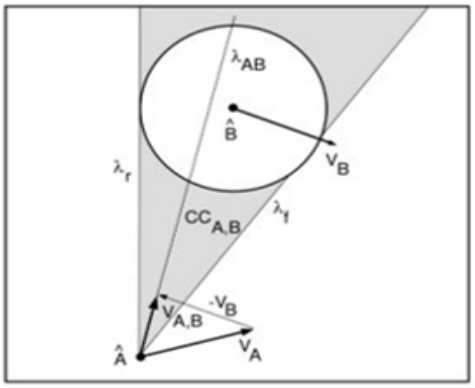


Figure 3: Relative Velocity cone [1]

The absolute velocity of the robot is given by offsetting the relative velocity cone by the velocity vector of the obstacle. The newly calculated velocity cone is calculated shown in the forthcoming image.

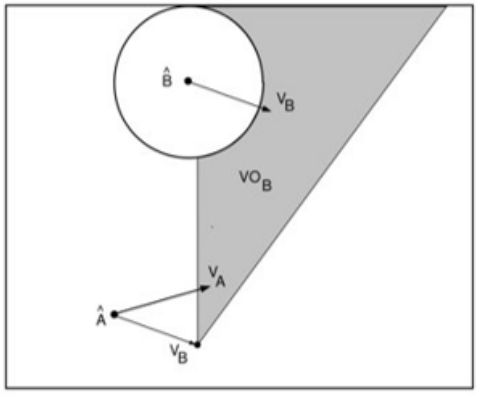


Figure 4: Velocity cone [1]

The problem with the approach is that it is generally unfeasible for a robot to immediately reach a desired velocity. Therefore, an algorithm is used to determine the set of feasible controls for a robot allowing it to avoid obstacles [2].

Algorithm 1 Find best feasible control

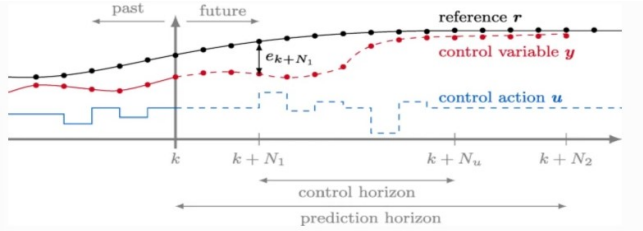
for $i = 0$ to n **do**
  $u \leftarrow$ sample controls from the set of all controls U
  $t_{\lim} \leftarrow$ sample time limit $\in (0, max]$
  $free \leftarrow$ true
  $min \leftarrow \infty$
  for all Moving Obstacles B **do**
    let $D(t)$ = the distance between $A(t,u)$ and $B(t)$
    $t_{\min} \leftarrow$ solve $\min(D(t))$ for $t \in [0, t_{\lim}]$.
    $d \leftarrow D(t_{\min})$.
    if $d < r_A + r_B$ **then**
      $free \leftarrow$ false
    end if
  end for
  if $\|u - u^*\| < min$ **then**
    $min \leftarrow \|u - u^*\|$
    $argmin \leftarrow u$
  end if
end for
return $argmin$

Figure 5: The algorithm used for finding best feasible control [2]

Model Predictive Control

MPC is a collection of sophisticated control methods that explicitly employ a model to forecast the system's future behavior. By solving a restricted optimization problem with this prediction in mind, the MPC generates an optimum output u. It is one of the few control approaches that takes restrictions into account directly. The cost function is frequently written so that the system output y tracks a given reference r for a specified horizon N2, as seen in Fig. 3. The system only receives the first value of the optimal output trajectory. Each time instance, the prediction and optimization are repeated. MPC is also known as "receding horizon" control because of this.

The basic premise is that a short-term (predictive) optimization yields long-term optimality. This is supposed to be true since the inaccuracy of a proximal forecast is minor in comparison to a distant prediction. The fundamental distinction from traditional control systems, which rely on pre-computed control principles, is the coupling of prediction and optimization



The prediction horizon N2 must be long enough to capture the impact of changing the manipulated variable u on the control variable y. The lower prediction horizon N1 can be used to account for delays, or they can be included into the system model. The latter is frequently preferred, and the lowest prediction horizon is set at N1 = 1 to account for computing time (hence the computation is conducted in one time step, the solution u is implemented not before the next time step). Assuming arbitrary system. Model-based predictive control (MPC) is a collection of sophisticated control methods that employ a process model to forecast the managed system's future behavior. MPC finds the control rule implicitly by solving a—potentially constrained—optimization problem. This moves the focus of controller design to the modeling of the process to be regulated. The first barrier to applying control is eliminated with MPC because such models are accessible in many sectors of engineering. Its implicit formulation preserves the physical understanding of system characteristics, making controller tuning easier. Model-based predictive control (MPC) can even govern systems that traditional feedback controllers.

IV. UNITY

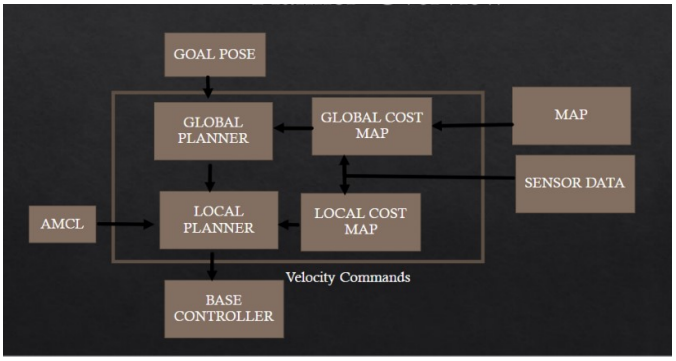


Figure 6: Planner overview

A. TEB

The TEB primarily tries to seek for the time-optimal solution, but you can also configure it for global reference path fidelity. The approach discretizes the trajectory along

the prediction horizon in terms of time and applies a continuous numerical optimization scheme. The planned trajectories are closer to the actual optimal solution, but the constraints are implemented as penalties, they are suitable for all robot types making the planning of multiple trajectory candidates in multiple topologies and the Dynamic obstacle support as larger computational burden.

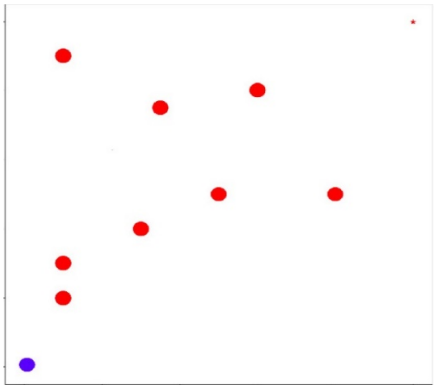
B. DWA

The DWA performs a sample-based optimization. It samples a control action in the feasible velocity space which is usually a translational/angular velocity pair and rolls out the trajectory for these particular sampled actions, In Suboptimal solutions without motion reversals the control actions are kept constant along the prediction horizon as are well suited for diff-drive/omnidirectional robots, but not for car-like robots, advantages include its ability to supports non-smooth cost functions

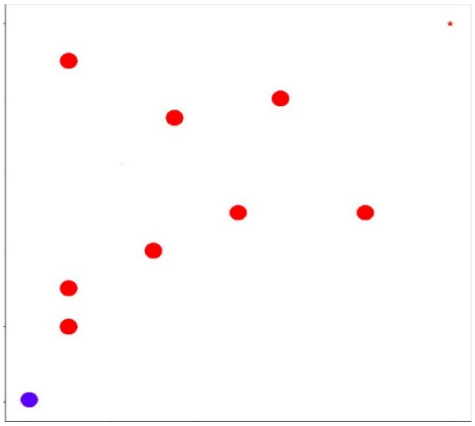
V. EXPERIMENT AND EVALUATION

A. TOY PROBLEM

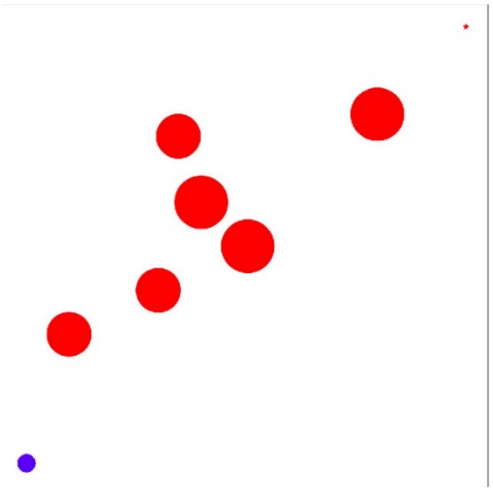
The below images show the set of scenarios that are selected in order to compare the proposed algorithms. The proposed methods are compared to the baseline of Artificial Potential Field. We have selected 5 particular scenarios to demonstrate the superiority of the selected local planners. The blue circle represents our robot and the red circles represent static or dynamic obstacles such as nurses, patients, carts or stations. Scenario 1 is a rather simple one where the robot needs to go past static obstacles to reach its goal. Scenario 2 is when all the obstacles except one are dynamic and truly test the capabilities of our algorithm. Scenario 3 consists of slightly enlarged obstacles with more or less the same setup as the other two scenarios. Scenario 4 is a special scenario. In this scenario, the obstacle lies between the goal and starting position and moves towards the robot. This scenario is a known scenario in which APF method fails due to the repulsive force exceeding the attractive force. The Velocity Obstacles and Model Predictive Control are able to avoid the obstacle and successfully reach the goal. In Scenario 5, we have setup an environment similar to that of the obstacle and tested the three algorithms.



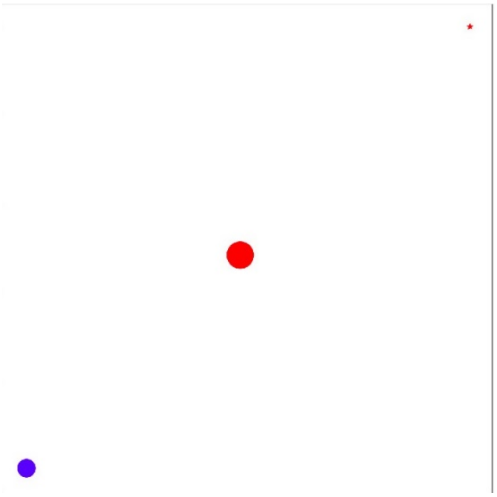
Scenario 1



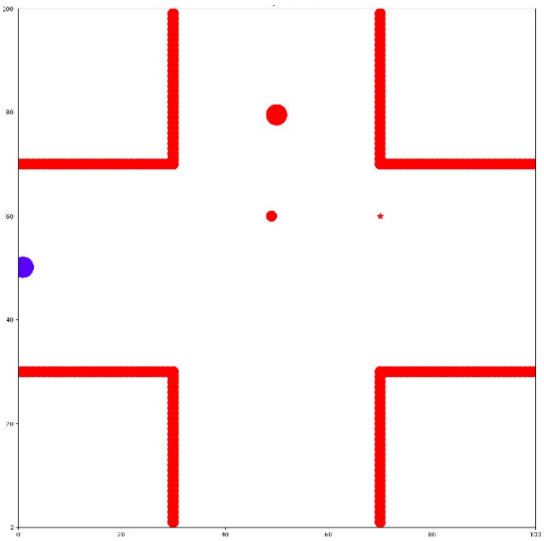
Scenario 2



Sceanrio 3



Scenario 4



Scenario 5

B. UNITY

Gmapping was carried out to get the hospital map of the unity to perform localization of the gopher robot. Navigation in Unity consists of two planners – Local and Global Planner. Global planner takes in global cost map from the generated hospital map to send the global way points for the global path to the local planner. A* was used in our global planner. The local planner takes in the local cost map as input from the sensor data. AMCL pose and global path from the global planner are also used as inputs for the local planner and finally velocity commands of the robot to the base controller.

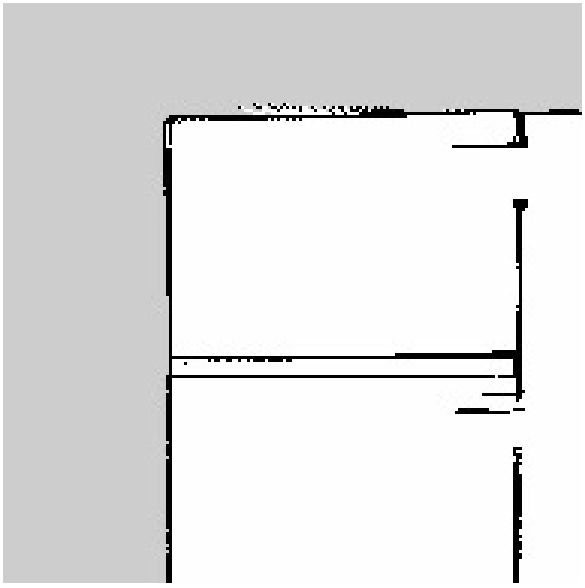


Figure 7: Generated hospital map

Figure 8: Hospital environment gopher

Local planners – Time Elastic Band, MPC and Dynamic window approach local planners are implemented in the Unity environment and the important parameters of the global and local costmaps and planners are tuned for better performance of the gopher robot in Unity environment. The parameters are tuned considering the following factors – Robot configuration, goal tolerance, trajectory configuration, obstacles, optimization and local and global planners.

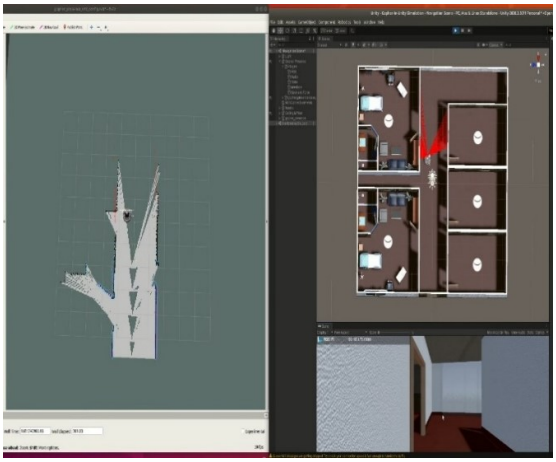


Figure 9: Gmapping of hospital environments

Different scenarios of hospital environment are simulated in unity to navigate robot in dynamic environments. Nurses moving in hospitals are simulated as shown in figure 8. The robot is made to move from the start to the goal by navigating around moving nurses.

The main Metrics considered for evaluation of the performance of the local planners are the time taken and path length taken by the robot to move from the start to goal

position. Local planners performance are compared in different scenarios to find out the advantages and disadvantages of all planners.

VI. RESULTS

A. TOY PROBLEM

Table I: Time (in seconds)

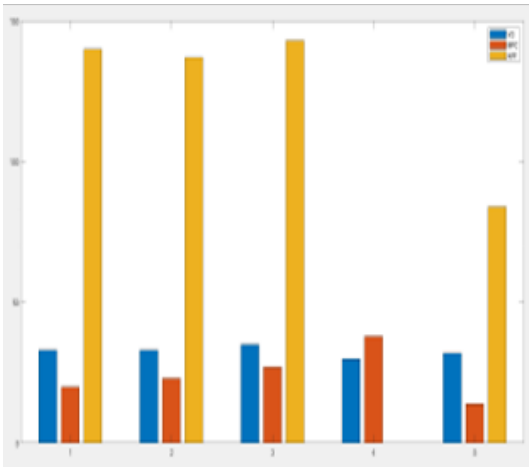
	Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5
VO	33	33	35	30	32
MPC	20	23	27	38	14
APF	140	137	143	NA	84

Table II: Path length (in meters)

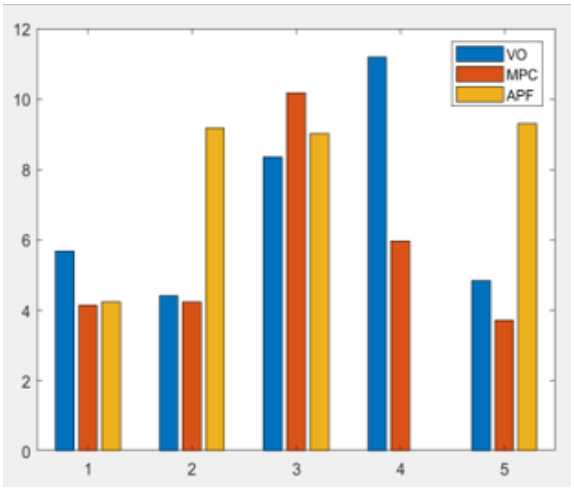
	Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5
VO	150.3	150.20	156.33	144.30	83.09
MPC	139.33	151.10	169	149.04	77.24
APF	140.0	136.99	143.0	NA	84.0

Table III: Minimum Clearance(in meters)

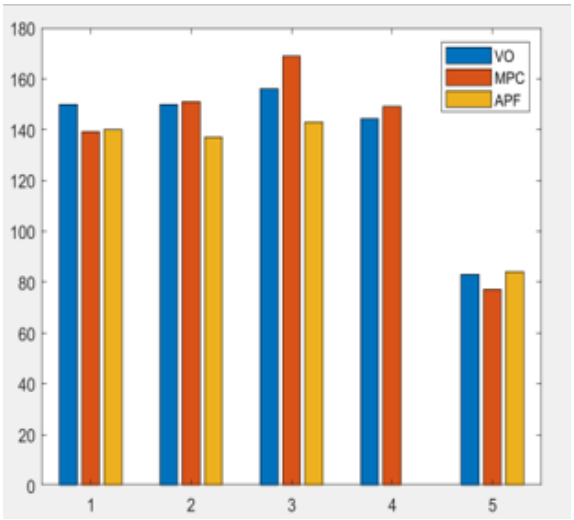
	Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5
VO	5.68	4.406	8.3546	11.19	3.72
MPC	4.144	4.24	10.18	5.96	3.72
APF	8.97	9.17	9.017	NA	9.3



Task Completion Time



Clearance wrt objects



Length traversed by path

The above table and charts provide us with a greater insight, we have compared the implemented algorithms with respect to the above metrics for the discussed scenarios. For task completion time, we can see that MPC takes the shortest time consistently (Except for scenario 4) followed by VO, both of which outperform APF by a large margin. As for clearance w.r.t obstacles, both MPC and VO find paths which may involve going closer to the obstacles to find a path, both are lesser than APF. Considering length of traversed paths, we can see that the path found by APF compared to the other two algorithms is shorter for most scenarios.

B. UNITY

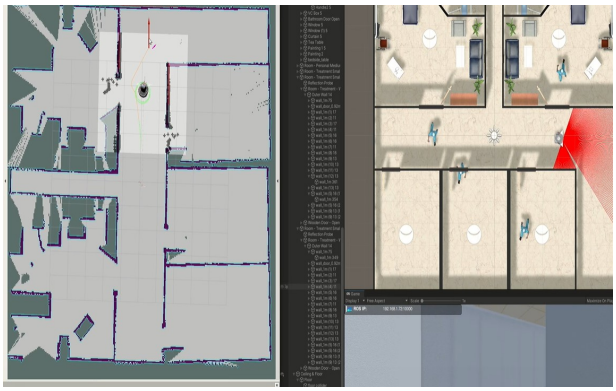


Figure 10: DWA Planner

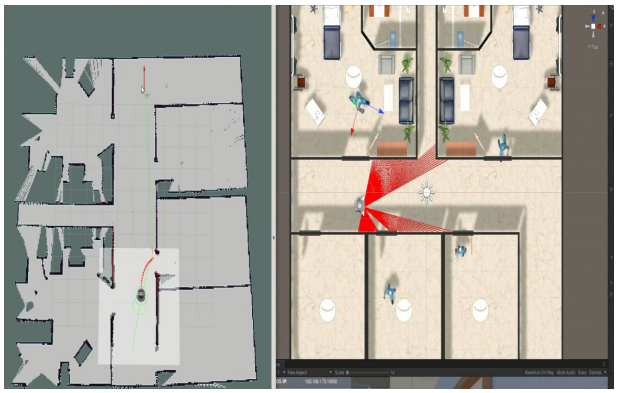


Figure 11: TEB Planner

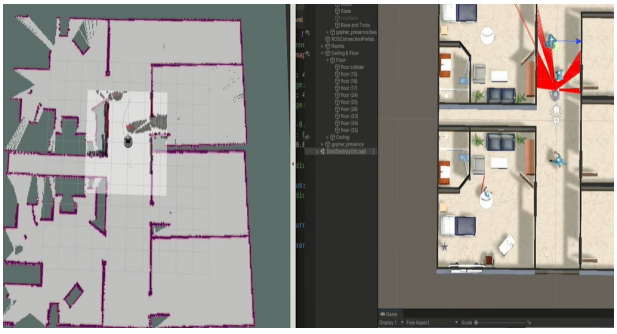


Figure 12: MPC Planner

Task Completion Time	MPC > TEB > DWA
Path Length	MPC > TEB > DWA

For most of the scenarios the task completion time is in the order shown in the results table. The path length and task completion time for MPC is usually higher than the TEB which is in turn more than DWA planner.

VII. CONCLUSION AND FUTURE WORK

Toy problem for three type of planners – MPC, APF and Velocity Obstacles were implemented and compared based on different metrics. Then MPC local planner was implemented in Unity with ROS as middleware and their performance were compared with two other local planners namely DWA and TEB. Parameter tuning was carried out to optimize the performance of all the three local planners. MPC local planner still have limitations as it doesn't perform properly in high dynamic obstacles environment. It takes comparatively more time in most of the scenarios.

VIII. REFERENCE

[1] Fiorini P, Shiller Z. Motion Planning in Dynamic Environments Using Velocity Obstacles. *The International Journal of Robotics Research*. 1998;17(7):760-772. doi:10.1177/027836499801700706

[2] Wilkie, David, Jur Van Den Berg, and Dinesh Manocha. "Generalized velocity obstacles." In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5573-5578. IEEE, 2009.

[3] Lim, Heonyoung, Yeonsik Kang, Changwhan Kim, Jongwon Kim, and Bum-Jae You. "Nonlinear model predictive controller design with obstacle avoidance for a mobile robot." In *2008 IEEE/ASME International Conference on Mechtronic and Embedded Systems and Applications*, pp. 494-499. IEEE, 2008.

[4] Hoy, Michael, Alexey S. Matveev, and Andrey V. Savkin. "Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey." *Robotica* 33, no. 3 (2015): 463-497.

[5] Montiel, Oscar, Ulises Orozco-Rosas, and Roberto Sepúlveda. "Path planning for mobile robots using Bacterial Potential Field for avoiding static and dynamic obstacles." *Expert Systems with Applications* 42, no. 12 (2015): 5177-5191

[6] van den Berg, J. Snape, S. J. Guy and D. Manocha, "Reciprocal collision avoidance with acceleration-velocity obstacles," 2011 IEEE International Conference on Robotics

and Automation, 2011, pp. 3475-3482, doi: 10.1109/ICRA.2011.5980408.

[7] J. Snape, J. v. d. Berg, S. J. Guy and D. Manocha, "The Hybrid Reciprocal Velocity Obstacle," in IEEE Transactions on
Robotics, vol. 27, no. 4, pp. 696-706, Aug. 2011, doi: 10.1109/TRO.2011.2120810.

[9] C. Rösmann, A. Makarow, and T. Bertram: Online Motion Planning based on Nonlinear Model Predictive Control with Non-Euclidean Rotation Groups, arXiv:2006.03534 '[cs.RO]', June 2020

[10] C. Rösmann, M. Krämer, A. Makarow, F. Hoffmann and T. Bertram: Exploiting Sparse Structures in Nonlinear Model Predictive Control with Hypergraphs, IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), New Zealand, July 2018

[11] Hiro-Wpi, “Hiro-WPI/gopher-in-unity-simulation: Gopher nursing robot in unity simulation,” GitHub. [Online]. Available: https://github.com/hiro-wpi/Gopher-In-Unity-Simulation. [Accessed: 03-May-2022].

[12] “Wiki,” ros.org. [Online]. Available: http://wiki.ros.org/navigation. [Accessed: 03-May-2022].