Collision Avoidance of Mobile Robots in Dynamic Environments

Abhiroop Ajith
Robotics Department
Worcester Polytechnic Institute
Worcester, MA
aajith@wpi.edu

Mithulesh Ramkumar
Robotics Department
Worcester Polytechnic Institute
Worcester, MA
mramkumar@wpi.edu

Chinmaya Khamesra
Robotics Department
Worcester Polytechnic Institute
Worcester, MA
ckhamesra@wpi.edu

***Abstract*—Motion Planning in a complex and dynamic environment is an essential component of robot navigation that has improved dramatically over recent times. This paper presents navigation of a single mobile robot to reach its goal destination without interacting with the static and dynamic obstacles or other agents in the environment using two advanced motion planners: velocity obstacles and dynamic window approach. The work uses algorithms like APF and velocity obstacles to avoid obstacles. The algorithms are implemented on Pygame and Gazebo environment. The performance of different algorithms are compared on the basis of the different performance metrics defined like task completion time, clearance, path length, and path smoothness. It was found that VO performed better in three of the four performance metrics when compared with the baseline and APF algorithms**

***Index Terms*—Velocity-Obstacle, Artificial Potential Field, Mobile Robots, Dynamic Obstacles, Pygame, Gazebo**

I. Introduction

The mobile robot industry is at an inflection point where the vast improvements in computer science like AI and cloud have started to filter into mobile robots making them increasingly capable to accomplish normal human task. Applications for this include surveillance, personal service, automation, and many other applications. Mobile robots are useful not only to industries but are equally important to everyday consumers. Due to these rapid changes safe robot navigation has become of paramount importance. Among safe robot navigation, the problem of obtaining a collision free path among a dynamic environment is difficult. These can be useful for a wide array of applications like automated transport systems, automated industries, and for daily consumer usage like robotic wheelchairs [1]. While many of these applications involve robot with non-holonomic kinematic constraints, our work will use a holonomic robot to simplify the problem. The safe navigation problem is also widely studied for crowd simulation in computer graphics, virtual environments, video games, traffic engineering, etc

Of these, motion planning is one of the tenets of safe robot navigation. This paper presents two advanced planners: Velocity Obstacles and Dynamic Window Approach. These proposed methods are used for various applications like differential drive robot, automated transportation systems, automated factories, application involving human-robot interactions such as robotic wheelchair. It is used for real time navigation in dynamic environments for car-like robot and vacuum cleaner robots like Irobot roomba. For our implementation, a turtlebot3 robot on gazebo is going to be tested in a cafeteria environment using these motion planners.

II. Related Work

A. Generalized Velocity Obstacles

The paper "Generalized Velocity Obstacles" [1] deals with velocity obstacles for collision avoidance, it also takes into account the given mobile robots kinematic constraints.It is used in applications like Real-time navigation in dynamic environments for car like roots. One of the drawbacks of this method is,it is not able to take into account the steering angle velocity and hence the paths are just piece-wise smooth it is not able to take into account the steering angle velocity and hence the paths are just piece-wise smooth

B. Reciprocal Collision Avoidance with Acceleration-Velocity Obstacles

The paper introduces Acceleration-Velocity Obstacle (AVO) [2] which lets it avoid obstacles while being able to characterize a new set of velocities the robot can safely reach and adopt using proportional control of acceleration.One of the main drawbacks is that the paper has assumed the robots move in a 2D workspace that is disc-shaped and capable of omni-directional acceleration. It is not sure whether the approach can be used for arbitrarily kinematically constrained systems In the figure [1] we can see that , in that first trajectory Two

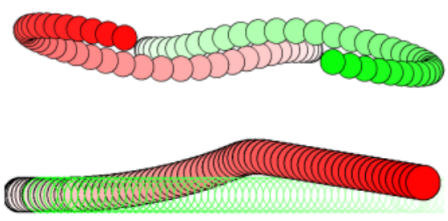

Fig. 1. Robot Collisions

robots with acceleration constraints avoid collisions with each other while exchanging positions, and pass their target location

with high speed. Newer frames are on top of older frames and darker and in the below trajectory robot with acceleration constraints avoid collisions with an oncoming obstacle. Newer frames are on top of older frames and darker.

C. Hybrid Reciprocal Velocity Obstacles

It deals with collision-free and oscillation-free navigation of multiple mobile robots.The paper takes into account obstacles in the environment,uncertainty in radius,position, and velocity. It also takes into consideration dynamics and kinematics of the robots. The main advantage of this paper is that it takes into account reciprocity. Each robot assumes that other robots are co-operating to avoid collisions. Each robot acts completely independently without central coordination and communication with each other. A total of 100 virtual agents are spaced

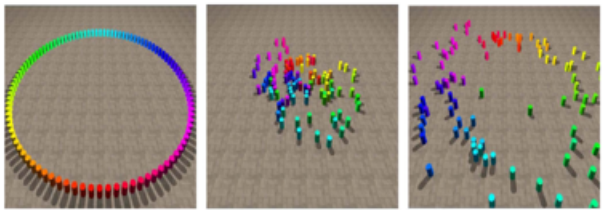

Fig. 2. 100 Virtual Agents and various scenarios

evenly on the perimeter of a circle. Their goals are to navigate to antipodal positions on the circle. The agents will meet and have to negotiate around each other in the center . shows traces of the five robots in scenario 1 for three variations of the velocity obstacle formation. (a), when the robots use velocity obstacles, the traces are not smooth due to oscillations, while in (b), for reciprocal velocity obstacles, the beginnings of the traces are not smooth due to reciprocal dances. The traces in (c), with robots navigating using hybrid reciprocal velocity obstacles, show no oscillations or reciprocal dances over their entire lengths for any robot

D. Generalized reciprocal collision avoidance

It presents a unified method for reciprocal collision avoidance of non-homogeneous systems of robots with non-linear equations of motion. It also presents the control obstacle for homogeneous systems of roots with linear equations of motion Extended control obstacles for use with non-linear equation of motions and/or non-homogenous systems. Implemented on differential drive robot, differential drive with off-axle trailer, car-like robot and hovercraft

E. Improved Artificial Potential Field

The paper improves upon the already existing APF. An improved artificial potential field based regression search method is proposed to obtain a global optimal/suboptimal path without local minima and oscillations in complete known environment information In figure [3] , the Blue line is the path which is planned by the improved APF, while red line is the optimal path utilizing regression search method.

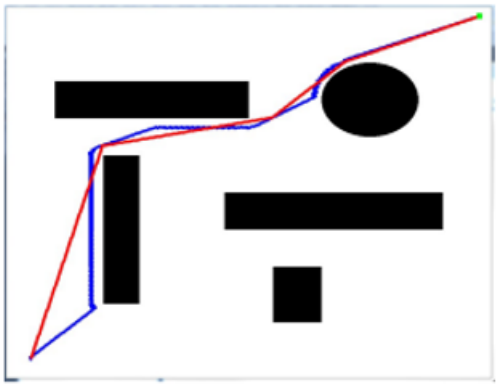

Fig. 3. Improved APF

III. Proposed Method

A. Artificial potential field

APF deals with Potential function, which is scalar function and depends on the robot configuration, goal position and obstacle positions. There are two types of potential function considered for the method, the attractive potential function which attracts the robot towards the goal whereas the repulsive potential function which avoids the robot to get close to the obstacle. For calculating the distance between the goal and the robot position, Euclidean distance is taken into consideration. The Total potential field of the robot is the force field which attracts the robot towards the goal while avoiding the obstacles. There can be cases when the attractive and repulsive forces are equal in magnitude, gradient becomes zero and the robot will not be able to move from that position and thus would not be able to reach the goal position. To overcome this, the project refers to advance techniques like velocity-obstacle and reciprocal collision avoidance with acceleration velocity obstacles.

Attractive Potential field and Force:

$$U_{att}(q) = \frac{1}{2}\,\zeta\, d^2(q, q_{goal})$$

$$F_{att}(q) = -\nabla U_{att}(q) = -\zeta \mathrm{d}(q, q_{goal})$$

Repulsive Potential field and Force:

$$U_{rep}(q) = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{d(q,q_{obs})} - \frac{1}{d_0}\right)^2 d(q,q_{goal})^n & \text{if } \mathrm{d}(q,q_{obs}) < d_0 \\ 0 & \text{if } \mathrm{d}(q,q_{obs}) > d_0 \end{cases}$$

$$F_{rep}(q) = -\nabla U_{rep}(q) = \begin{cases} \eta\left(\frac{1}{d(q,q_{obs})} - \frac{1}{d_0}\right)\frac{(q-q_{obs})}{d^3(q-q_{obs})} & , \text{ if } \mathrm{d}(q,q_{obs}) < d_0 \\ 0 & , \text{ if } \mathrm{d}(q,q_{obs}) > d_0 \end{cases}$$

Total Force:

$$F(q) = F_{att}(q) + F_{rep}(q)$$

Fig. 4. APF Equation

B. Velocity Obstacle

The Velocity-obstacle algorithm is based on the assumption that the robot equation of motion is single integrator kinematic model which deals with only position and velocity.Consider a static disc-shaped agent A and a dynamic disc-shaped obstacle B with radius rA and rB respectively as shown in

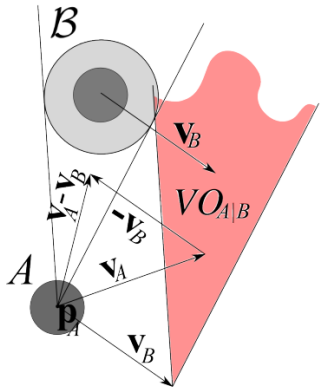

Fig. 5. The velocity obstacle $VO_{A|B}$ for robot A relative to dynamic obstacle B.

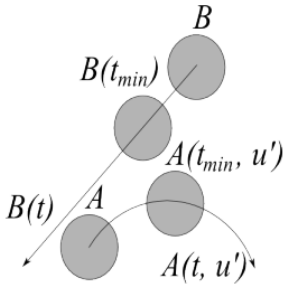

Fig. 6. The velocity obstacle $VO_{A|B}$ for robot A relative to dynamic obstacle B.

Algorithm 1 Find best feasible control

for $i = 0$ to n **do**
  $u \leftarrow$ sample controls from the set of all controls U
  $t_{\text{lim}} \leftarrow$ 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 $B(t)$
    $t_{\text{min}} \leftarrow$ solve $\min(D(t))$ for $t \in [0, t_{\text{lim}}]$.
    $d \leftarrow D(t_{\text{min}})$.
    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 $argmin$

Fig. 7. Pseudocode

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

Generalised Velocity Obstacle:

$$VO_{A|B} = \{\mathbf{v} \mid \exists t > 0 :: \mathbf{p}_A + t(\mathbf{v} - \mathbf{v}_B) \in \mathcal{B}\}.$$

$$t_{\min}(u) = \arg\min_{t>0} \|A(t,u) - B(t)\|.$$

Fig. 8. VO Equation

the figure 5.The pA and Pb are the center points and the system is generalized using minkowski sum. $VO_{A|B}$ is the set of velocities for A that would result into collision with B in some time in the future. The generalized velocity obstacle seeks to addresses the difficulty of using velocity obstacles with kinematic constraints agents. the result in the change in the robot configuration is due to control input u which is a set of input to the Kinematic-dynamic model. The minimum time formula is use to evaluate the distance between A and B and the minimum distance evaluates weather the control would be collision free or not.

As a dynamic constraint robot like car -like robot need to follow an arc to achieve the selected velocity, and the method does not provide any guarantee that the arc will be collision free. Thus, this approach does not guarantee collision-free navigation for dynamic constraint robots.

Numerical methods are used to evaluate the minimum distance which can lead to inaccuracy. The method does not guarantee's a feasible solution as it is a probabilistic algorithm. The algorithm relies on measuring the position and velocities which can be noisy in the real world. The current implementation of algorithm in the project uses greedy control which is not suitable for complex environment with local minima's.

C. RRT(Baseline)

A rapidly exploring random tree (RRT) is an algorithm designed to efficiently search nonconvex, high-dimensional spaces by randomly building a space-filling tree. The tree is constructed incrementally from samples drawn randomly from the search space and is inherently biased to grow towards large unsearched areas of the problem
. This paper uses RRT as the baseline because of it is good at finding paths from the start to the goal, albeit it may not be the optimal path. We further make it even more efficient by pairing it with A* as the global planner and making RRT as the local planner, after the workspace has been discritized into vornoi maps. RRT can be computationally expensive especially in environments like ours, Therefore it being one of the main factors we used various other local planners as we will discuss below.

D. DWA

The DWA is chosen as a local planner as a precursor to implementing Velocity Obstacles. DWA works really well with static obstacle collision avoidance, and gives us a platform

to build from to implement Velocity Obstacles.The algorithm works by sampling discretely from the given Robots workspace. From this sample,the velocity can be obtained and it is used to perform the forward simulations from the the robots current state and it uses this to predict the outcome if the sampled velocity was applied. Each of these trajectories are evaluated and the illegal trajectories are discarded. The highest scoring trajectory is chosen along with the associated velocities with respect to the mobile base. Evaluate each trajectory resulting from the forward simulation.DWA samples are from the set of achievable velocities for just one simulation step given the acceleration limits of the robot. DWA is a more efficient algorithm because it samples only a small space and therefore it has higher efficiency gains than an algorithm like Trajectory Rollout

E. Performance Metric

The proposed algorithm performance metrics to be tested on the following criteria:

1) Task completion time
2) Clearance w.r.t. obstacles
3) Length of traversed path
4) Smoothness of path to obstacles.

IV. Implementation

The implementations were carried out on gazebo, python and pygame. The experimental method table is summarised in table I

A. Pygame Environment

Using pygame an environment was created with randomly moving obstacles. The obstacles were a separate class while the robot cell was a separate class. In the obstacles class, functions existed to give a random directions and speeds to each obstacle created. Eight directions were chosen from : N, W, S, E, NE, NW, SE, SW. The robot class had additional functions where the robot node would move with a given velocity. It also had a global carrot planner to move the robot cell from start to goal. Using these functions and classes, both VO and APF were implemented on Pygame with randomly moving dynamic obstacles.

An example of the Pygame environment is shown in figure 9.

Category	**Description**
Planning Strategy	APF, VO, DWA
Global Planner	Dijkstra, Carrot Planner
Local Planner	Velocity Obstacle Approach
Environment	ROS, Python
Robot	Turtlebot
Simulation	Python, Pygame and Gazebo

TABLE I
METHODOLOGY TABLE

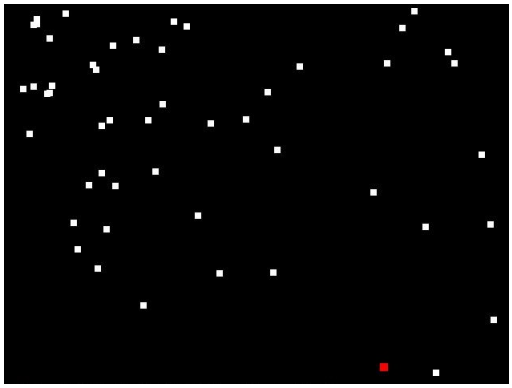

Fig. 9. Pygame Environment

B. Artificial Potential Field (APF)

For APF, the potential function is calculated. Robot start and goal position along with obstacle position is given as variables to the function. Motion captured is done to get the neighbours of the robot position. The potential field is calculated. The potential field is calculated which comprises of two components i.e. the attractive potential and the repulsive potential. The attractive potential is the product of potential gain and the hypotenuses distance between the robot and the goal position whereas the repulsive potential is a function of repulsive potential gain and the hypotenuse distance between the robot position and obstacle positions. The minimum distance in x and y direction is calculated, each potential is added to the map function and the oscillation detection is checked by comparing the previous values with the detection length. The path is plotted and further the animation can be obtained using matplotlib and heat maps.

C. Velocity Obstacles

For velocity obstacles, the velocity cones were constructed. To start off, each obstacle was inflated with the use of Minkowski sum. The radius the robot was added to each obstacle and the robot was then considered as point object. The relative position and relative velocity of each obstacle was calculated with respect to the robot. The velocity cone was constructed using dot product and Pythagoras theorem. In the most extreme case of velocity cone the velocity is barely touching the obstacle and this forms a right angled triangle. If the robot's relative velocity was inside this right angled triangle, it was considered to be in the path of the obstacle otherwise it was deemed as safe. Additional constraints were added where the velocity cones were only considered for a time horizon of 1 second which for our environment was 125 pixel space. Only velocity cones inside this space was considered. If the velocity of the robot was in the direction of the velocity cone, a random direction was chosen in to get out of it. Once the local planner was able to solve the obstacle constraint, the global planner took over and delivered the robot to the goal position

In figure 10, we can see the obstacle cone calculations which verifies if a given velocity is within that cone or not. It returns true or false based on that. In figure 10 the true and false are calculated real time with obstacles randomly moving and the robot moving towards the goal.

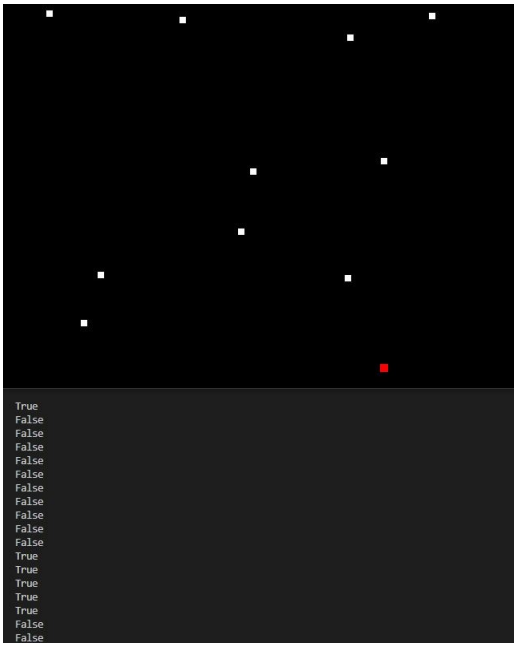

Fig. 10. Velocity Obstacle Cone

D. Gazebo - Global and Local Planner

The Final Part of the implementation of this paper comes in ROS on Gazebo which is a simulation environment which acts as our sandbox to test our algorithms. Gazebo is paired along with Rviz which works as the visualization tool and helps see the information like the paths, the costmap and information from the various topics. The paper uses turtlebot and huskybot for the experiments conducted in the various gazebo worlds. The first part of the project deals with the implementation of the baseline. The Global planner used is A* which finds the shortest path from a point defined to the goal position. This is performed after the given map has been discretized. RRT is used as the local planner which performs at a smaller radius than the global planner. The outputs from the baseline are shown in the figure 11

For the reasons mentioned above, this paper then shifts its main focus into implementing Dynamic Obstacle Avoidance. Dijkstra's Algorithm is used as the Global Planner and DWA Algorithm is used as the Local Planner, we have also Incorporated a Local Costmap. Dijkstra's algorithm is used to find the shortest path from the start to the goal and the local planner works on the collision avoidance part of this implementation, making sure the robot does not collide with any new obstacles that get in its way. The local costmap that has been implemented helps with a factor of safety and makes sure the robot does not go too close to the object even

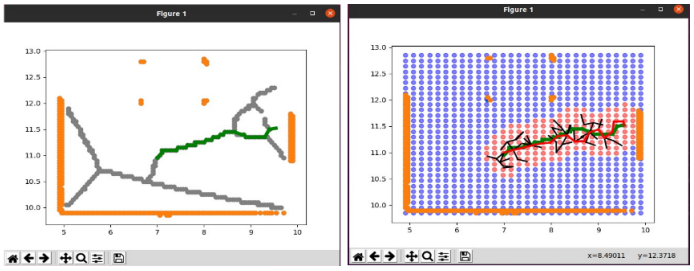

Fig. 11. A* and RRT on Gazebo

though we have a global and local planner. The Parameters are tuned for both the DWA Algorithm and the tolerances are changed and tuned for the Local CostMap to get the desired best results. The Cafeteria map is used as the map of choice to test the experiments as it has various obstacles and some curves in the paths, where the robot can be tested for different test cases which induces varying degrees of trajectories. The environment that shows Rviz along with the implemented Costmap and the Global Planner (Green) and Local Planner (Red) is shown below in the figure 18 figure 18

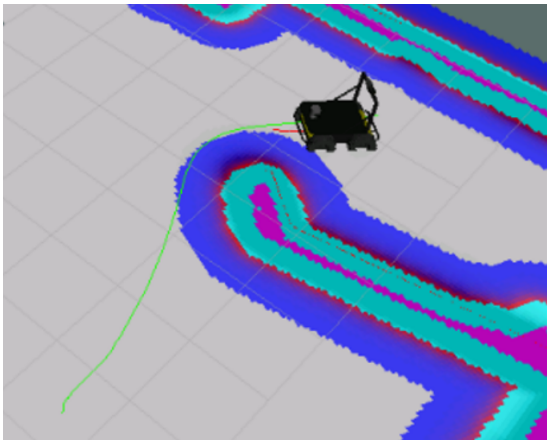

Fig. 12. DWA Planner on Gazebo

Fig. 13. Gazebo environment

V. Results

A. Artificial Potential Field (APF)

A 2D hospital floor environment was setup with 2 operating rooms and lobby passage. The robot was assigned to go from

the start position to the goal position (operating room2) using APF algorithm. The robot was successful in planning the path from the start position to the goal position. In Fig 14, the path generated by the robot has been captured.

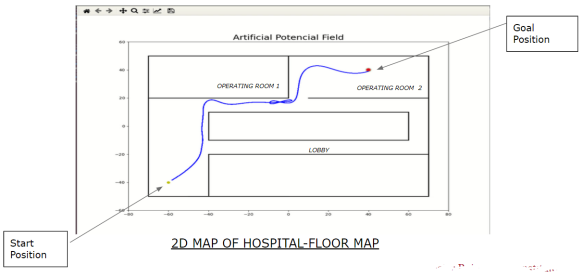

Fig. 14. Robot stuck at local minima (APF algorithm)

There are some limitations of using APF algorithms:
1. If the potential function of the goal and the obstacle becomes equal, gradient tends to zero and the robot is stuck at the local minima in Fig 16 shows a scenario of the robot trajectory when the robot is stuck at the local minima. Fig 15 shows the oscillated velocity profile when the robot is stuck at the local minima.
2. APF algorithm needs to compute the potential function after every step of the robot. The computation cost increases when the dynamic obstacles are added in the environment. These leads to slow convergence of the robot to the goal position.

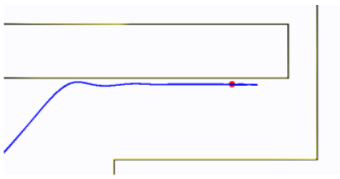

Fig. 15. Robot stuck at local minima (APF algorithm)

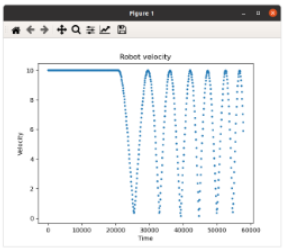

Fig. 16. Velocity profile of robot stuck at local minima

B. Velocity Obstacles

The velocity obstacles was implemented in Pygame with randomly moving obstacles. On most cases the trajectory was found where the robot was able to navigate the environment safely. In some cases, where the number of obstacles were too large or if the time horizon was too large or small, the VO program hit the obstacles. If the number of obstacles increases, this just increases the chance of hitting an obstacle. If the time horizon was too small, VO considers a smaller VO cone and therefore is not able to properly predict a future. However, if the time horizon was too large, robot tended to oscillate between each obstacle and not reach the goal. Due to the larger size of the constraint cone, any changes in velocity was inside it therefore reducing the amount of velocities available to the robot. Most times this crashed the program due to the number of inputs given to the Pygame environment.

Fig. 17. APF implementation on Pygame with dynamic obstacles

C. DWA Planner

The DWA planner was implemented on Gazebo and performed well in most cases. It was tested for cases with straight paths, paths where sharp turns had to be taken and paths that would have collisions in them. The algorithm performed well when tuned along with the CostMap that was tuned as well. It was tuned depending on the environment. If it had too high tolerance there was a chance that the path would not be found even though the robot would have any space, and if the tolerance was too low, there was a chance for a collision. Both the DWA and Local Costmap were tuned to make sure it worked well for the given application.

		APF	VO	Baseline
1	Task Completion Time	✗	✓	✗
2	Clearance	✓	✓	✓
3	Path Length	✗	✓	✓
4	Path Smoothness	✗	✗	✗

Fig. 18. Performance metric

D. Performance Metric

The performance of all the algorithms were compared on the basis of performance metric which is shown in the Fig18.

For APF algorithm, when the robot is stuck at the local minima, task completion time cannot be obtained. Velocity obstacles works good with the dynamic obstacles so the algorithm was always able to complete the task and thus was always able to give the task completion time. For clearance with respect to the obstacles, minkowski sum was used for all the algorithms which inflated the area around the robot properly. Hence all the algorithms were able to provide good clearance. Due to local minima problem faced in APF, the path length can become quite questionable sometime. The other two methods were able to give the path length properly. In terms of path smoothness, all the algorithms failed to provide path smoothness. The VO which provided jagged path can be changed by providing a direction which is very close to the global path.

VI. Conclusion and Future Work

The collision avoidance of the robot with with static and dynamic obstacles were done. First, Algorithms like APF and Velocity obstacle were tried on the 2D static and dynamic environment. The work was then extended by testing the algorithms as local planner on Gazebo environment with local planner as RRT.

There are some limitation faced during the project. The Velocity Obstacles algorithm needs real time velocity data which was difficult to obtain in Gazebo with active perception. The APF algorithm became slow as the computational complexity increased when it was introduced with dynamic obstacles. The integration of VO can be done with better global planner.

The work can be extended by implementing the Velocity Obstacles with active perception on Gazebo which we are attempting currently. All these implementations of the algorithms can be extended to the Unity environment. The Paper will also highly benefit by testing in various other worlds with different test cases and parameters of velocity obstacles.

References

[1] 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.

[2] Van Den Berg, Jur, et al. "Reciprocal collision avoidance with acceleration-velocity obstacles." 2011 IEEE International Conference on Robotics and Automation. IEEE, 2011.

[3] Snape, Jamie, Jur Van Den Berg, Stephen J. Guy, and Dinesh Manocha. "The hybrid reciprocal velocity obstacle." IEEE Transactions on Robotics 27, no. 4 (2011): 696-706.

[4] Bareiss, Daman, and Jur van den Berg. "Generalized reciprocal collision avoidance." The International Journal of Robotics Research 34, no. 12 (2015): 1501-151

[5] Li, Guanghui, Atsushi Yamashita, Hajime Asama, and Yusuke Tamura. "An efficient improved artificial potential field based regression search method for robot path planning." In 2012 IEEE International Conference on Mechatronics and Automation, pp. 1227-1232. IEEE, 2012.