Obstacle Avoidance in Dynamic Environments using Generalized Velocity Obstacles Method and Non-Linear Model Predictive Control

Shiva Kumar Tekumatla
Robotics Engineering Department
Worcester Polytechnic Institute
Worcester,MA
stekumatla@wpi.edu

Anujay
Robotics Engineering Department
Worcester Polytechnic Institute
Worcester,MA
asharma6@wpi.edu

Gaurav Bhosale
Robotics Engineering Department
Worcester Polytechnic Institute
Worcester,MA
gbhosale@wpi.edu

***Abstract*—In this paper, we present different methods that are used for navigation of a mobile robot in unknown and dynamic environments. We compare various methods for real-time navigation of a mobile robot , and present collision free control using generalized velocity obstacles(VO) and non-linear model predictive control(NMPC). In this paper, we also discuss about the usage of various physical constraints of a general mobile-robot. The disadvantages of a physical constrains for motion planning are tackled using these methods. Generalised Velocity-Obstacle provides a combined solution for both holonomic and non-holonomic robots. NMPC , a variation of MPC, that is used here , does not require any reference trajectory , thus performing computationally well. We compared these two methods for navigation in dynamic environments. We use the performance of these algorithms in complete static or a few dynamic obstacle environment as baseline for testing other cases. We tested the performance of these algorithms using different evaluation parameters. We also presented a clear understanding of usage of various types of adhoc parameters for each of these algorithms. Finally, we simulate our results in RVIZ simulation platform using ROS framework. We see the application of our methods where robot needs to navigate environments such a hospital corridor with humans other robots , social robots such as in airports etc.**

***Index Terms*—Motion Planning, Velocity Obstacles, Model Predictive Control, RVIZ and ROS**

I. Introduction

Over the past decades, the field of motion planning for mobile robots has seen development of numerous algorithms that work well for almost all types of environments. However , most of the methods have some drawbacks in the practical implementations. For example, reaching local minima, sensor noise etc. This can be due to different reasons. Constraints of robot can cause design stabilization problems during implementation.Same way, the limited knowledge of environment or the computation speed can play a huge role in finding a good solution.

We need to come up with generalised solutions to solve the problem in dynamic environments. This means we cannot always compute the model for the environment. The first method of Velocity Obstacle is used for avoiding obstacles as we go. Geometric Methods are used to compute collision profiles. These profiles can be avoided to traverse a smooth trajectory.

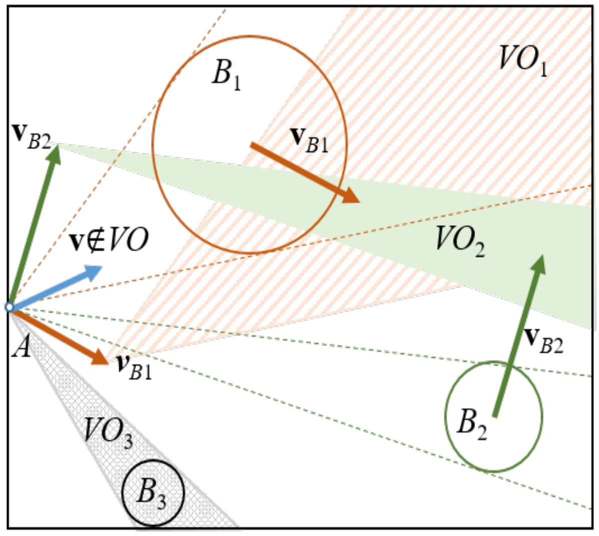


Fig. 1. Avoiding obstacles with collision cones

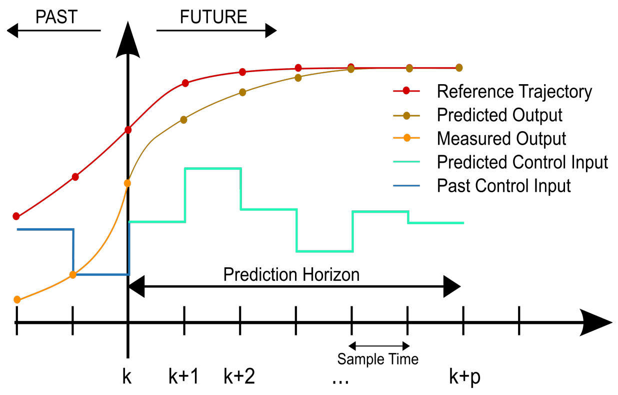


Fig. 2. General working method of MPC. MPC uses a finite-horizon method to compute the optimized inputs foe the system. In the given time frame , MPC checks the system's performance against a reference trajectory.

MPC, a method that was widely used process industries is making it's way into motion planning for autonomous vehicles and also to mobile robots. The consideration of dynamic model of the mobile robot in modeling of MPC makes it one of the best performing motion planning methods in dynamic environments. Given a good computation power and the physical system data , MPC can result optimized system inputs. MPC also takes future events into consideration thus computing effective and optimized paths.

MPC works on computing optimized results in finite-horizon in iterative passion[1]. Figure 1 [1] shows the general working of MPC. In [2][3][4] , the authors presented various use cases and implementations for MPC. Over the years , different version of MPC are well studied. Nonlinear, explicit and Robust MPC algorithms have use cases for different types of systems.

II. Related Work

In this section, we try to explore some related work in the field of obstacle avoidance and navigation.

A. *Velocity-Obstacle*

Motion Planning in dynamic environments was first addressed by adding a dimension of time to robot's configuration space However it was assumed that the obstacles would have bounded velocities and known trajectories. Reif *et.al* [5] solved it by searching a visibility graph in this configuration-time space. Erdmann *et.al* [6]., discretized this configuration-time space to get a sequence of slices at successive intervals.

Another approach of planning is to decompose the problem into two sub-problems of : path planning and velocity planning, first computing the best path assuming static obstacles and then selecting the velocity to avoid moving obstacles. This method however it requires knowledge of the world model.An alternative to completely planning the path beforehand, is to plan for the robot as it moves. This method takes in the sensory inputs as they come for planning locally. This idea to define a set of velocities that if taken would result in collision in the future is called Velocity-Obstacle, given by $Fiorini$ and $Shiller$[7]. In the original paper VO was only applied to objects assuming that obstacles had a constant linear velocity and the mobile robot could make piecework linear velocity steps.

J. Snape[8] *et.al* defined a more realistic scenario for velocity obstacle. They provided a proportional gain to the desired velocity as accelerated velocity obstacle, i.e. the relative velocities of one mobile robot with respect to other robot/obstacle that would result in a collision between them before time t, with parameter δ for the new relative velocity v. They also defined a constraint of maximum acceleration, which meant the difference of velocities (new and old), multiplied with step factor could not exceed the maximum acceleration of system. They also defined a convex subset of half planes for this new formulation of reciprocal velocities which could be used to define an $ORCA$ (Optimal Reciprocal Collision Avoidance).Incase the new convex subset was a null space then the least unsafe velocity was taken into account.

J. van den Berg[9] *et.al* considered non-passive robots and reduced oscillations suffered by vanilla VO. The collision cone keeps adjusting to varied velocities sampled for the two non-passive robots.Reciprocal velocity took care of this oscillation by defining half planes.It averaged out the velocity for a pair of velocity of the objects and the velocity obstacle.This adjustment was piece wise and could be done without any communication between the two mobile robots. Both the objects were guaranteed to pass on the same side.

J. Snape[10] *et.al* defined another extension of Reciprocal velocity obstacles[9], which could avoid the problem of reciprocal dance. They reused the concept of half planes but also defined the side of traversal for multi-agent systems.Hybrid RVO enlarged the RVO on the side that should not be attempted. If a robot attempted to pass the restricted side, it would have to give full priority to the other robot. This reduced the possibility of oscillations while not over constraining the motion of the other robot. This method handled well the uncertainty in position and dynamics. This method also generated a smoother path as compared to VO and RVO however, took a lot more time to compute per time step.

TABLE I
Comparison of different types of Velocity Obstacle

Feature	Vanilla	AVO	RVO	HRVO
Time	Low	Medium	Medium	High
Multiagent	No	Yes	Yes	Yes
Noisy Trajectory	Yes	Yes	Yes	No
Parameter Tuning	Time Horizon	Step Size	Agent Priority	Agent Priority

B. *MPC*

Li *et.al*[11] used MPC coupled with APF to generate trajectories for obstacle avoidance. In this work, authors used APF to generate the local path by introducing a virtual goal. The virtual goal is then used with offline explicit MPC to compute the optimized control inputs to a mobile robot. The planned path is mainly coupled with a PID controller to avoid the collisions in dynamic environments.

Bojadzic *et.al* [12] used RRT and MPC to generate path and trajectories for an autonomous cycle rickshaw. In this work, RRT is used as a global planner and MPC is used local planner. Non-holonomic constrains are used to make the algorithms effective using Kinematic Single-Track Model. In this wrok, authors implemented these algorithms on a physical cycle rickshaw.

Kamle *et.al* [13] used nonlinear model predictive control (NMPC) for multi-micro aerial vehicles for robust collision avoidance. Authors developed a model-based controller to achieve reference trajectories simultaneously. Uncertainty of the state estimation is compensated using higher order robustness. This method does not require any collision free reference trajectory. We intend to replicate this work and compare its results with various other algorithms.

Prior to this Shim *et.al* [14] presented a decentralized NMPC for multiple flying object. In this work , authors combined stabilization of dynamic systems and trajectory generation by including a potential function. They tested this method on multiple autonomous flying helicopters.

There are different versions of NMPC presented in the previous literature using different methods. Mayne *et.al* [15] presented a tube based robust nonlinear algorithm that is robust to disturbances in the system. In this work, the authors created a tube like area that acts as center for a nominal trajectory. They considered cross sectional area of this tube as an invariant set, and is robust. As part of this work, they created two controllers: Nominal controller that generates the path for the robot to traverse ,and ancillary controller that endeavors to steer the trajectories of the uncertain system to the central path.

Shin *et.al* [16] used NMPC for formation of flight for UAVs. In this work , they could make the UAVs to avoid obstacles using three different architectures. They are: Centralized, Sequential decentralized and fully decentralized. Using various experiments, they proved that centralized architecture requires more computation time, decentralized architecture is more faster and uses only the other agents information to complete the task successfully, and fully decentralized architecture is more efficient compared to the other two.

Scholte *et.al* [17] created a robust NMPC algorithm that is efficient even if the states information is not fully known. This algorithm is robust to model uncertainties and bounded noise sources. In this implementation , the authors used kalman filter based estimation technique to estimate bounds that result in lower computation time, and good for satisfying constraints with partial state information. Comparison of different types of VO algorithms is given by Table 1.

TABLE II
COMPARISON OF DIFFERENT TYPES OF NMPC

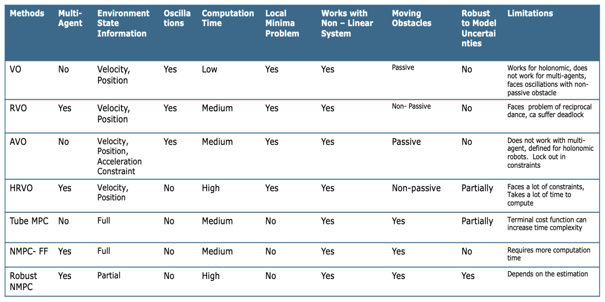
Feature	Tube MPC	NMPC for Flight formation	Robust NMPC with partial state information
Robust to Model uncertainties	Partially	No	Yes
Multiagent	No	Yes	Yes
Involves State Estimator	No	Cost-Estimate	Yes
Works with nonlinear Systems	Yes	Yes	Yes

Figure 3 shows the overall comparison of different types of velocity obstacle methods and nonlinear model predictive control methods.

III. Proposed Method

A. Velocity-Obstacle

In this method we propose to use the Generalized Velocity-Obstacle method for a collision free movement in a dynamic environment. We assume both the object and the mobile robot to have a circular dimension without loss of generality. The approach focuses on iteratively sensing and avoiding the obstacles. The authors[18] coin the term as $sense-plan-act$. The core idea is to define a set of velocities that would lead to collision with an obstacle sometime in the future.



Methods	Multi-Agent	Environment State Information	Oscilla tions	Computation Time	Local Minima Problem	Works with Non – Linear System	Moving Obstacles	Robust to Model Uncertai nties	Limitations
VO	No	Velocity, Position	Yes	Low	Yes	Yes	Passive	No	Works for holonomic, does not work for multi-agents, faces oscillations with non-passive obstacle
RVO	Yes	Velocity, Position	Yes	Medium	Yes	Yes	Non- Passive	No	Faces problem of reciprocal dance, ca suffer deadlock
AVO	No	Velocity, Position, Acceleration Constraint	Yes	Medium	Yes	Yes	Passive	No	Does not work with multi-agent, defined for holonomic robots. Lock out in constraints
HRVO	Yes	Velocity, Position	No	High	Yes	Yes	Non-passive	Partially	Faces a lot of constraints, Takes a lot of time to compute
Tube MPC	No	Full	No	Medium	No	Yes	Yes	Partially	Terminal cost function can increase time complexity
NMPC- FF	Yes	Full	No	Medium	No	Yes	Yes	No	Requires more computation time
Robust NMPC	Yes	Partial	No	High	No	Yes	Yes	Yes	Depends on the estimation

Fig. 3. Overall comparison of different types of Velocity obstacle methods and nonlinear model predictive control methods

For a circular shaped mobile robot, $\mathbf{A}$, and another circular shaped obstacle $\mathbf{B}$, we define their radii as r_a and r_b, respectively. The velocity obstacle for A induced by B is then $VO_{A|B}$, the set of all velocities that would result in a collision at a point in the future. This set can be defined in a geometric manner.

Let us assume that the center of the mobile robot and obstacle are at ρ_A and ρ_B respectively. We then take a disc of size β_{A+B} centered at ρ_B. We can now define a collision cone C, for the velocities for A that would lead to a collision with B. The collsion cone is formed by the set of rays that shoot from A towards the boundary/circumference of B. To account for multiple obstacles, it becomes useful if we can have an equivalent condition considering the absolute velocities of A. This can be done by simply shifting the entire collision cone by the vector $\mathbf{v_b}$, i.e. the velocity of the obstacle B.

$$VO_{A|B} = \{\mathbf{v} \| \exists t > 0 :: \rho_a + t(\mathbf{v} - \mathbf{v_b}) \varepsilon \beta\} \tag{1}$$

The generalized obstacle is defined as follows. Given the obstacle B, we can denote it's position at a time t by $B(t)$. For the mobile robot A, we can define its position at time t and a control input u (i.e. output velocity) by $A(t, u)$. The obstacle in our control space can be the defined by :

$$\{u \| \exists t > 0 :: \|A(t, u) - B(t)\| < r_a + r_b\} \tag{2}$$

Since an imminent collision between the robot and the obstacle would occur at the shortest distance between A and B, we usually consider a time set where only the collisions that happened before a time horizon are addressed. Let us consider a time $t_{min}(u)$, which is the time when the distance between the centers of the mobile robot and the obstacle is minimal for a given control input u for the mobile robot:

$$t_{min}(u) = argmin \|A(t, u) - B(t)\| \tag{3}$$

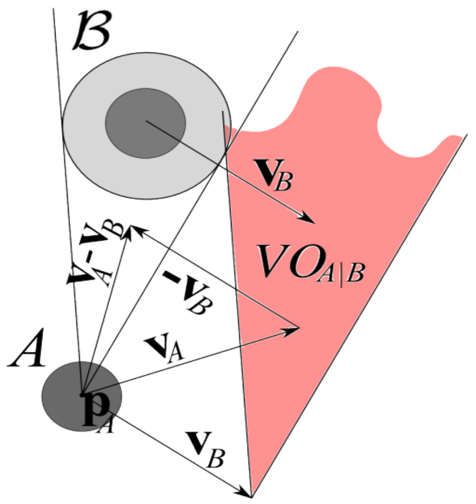


Fig. 4. Velocity obstacle $VO_{A|B}$

This t_{min} can be obtained by solving the closed form expression:

$$\frac{\partial \|A(t,u)-B(t)\|)}{\partial t}=0 \tag{4}$$

If the robot is given a control velocity $\mathbf{v}$ and the obstacle it at a position ρ_b with velocity $\mathbf{v_b}$ then we can write,

$$A(t,\mathbf{v})=tv, \tag{5}$$

$$B(t)=\rho_b+tv_b \tag{6}$$

Solving the closed form equation would then give us,

$$t_{min}(v)=\frac{\rho_b.(v-v_b)}{\|v-v_b\|^2} \tag{7}$$

A preferred control input $u*$ can be computed from the start position to the goal based on the assumption there are no obstacles. The control closest to this preferred $u*$ but outside velocity obstacle is then chosen to navigate the robot. In some cases it may also be possible that an equation for velocity obstacle can be found by using this t_{min} and solving the equation for u,

$$\|A(t_{min}(v),v)-B(t_{min}(v))\|<r_a+r_b \tag{8}$$

In cases where a closed form is solution is not available control can be found using a sampling approach although that is not usually complete.

The actual control given to the robot is given by,

$$u=argmin\|u^*-u'\| \tag{9}$$

Thus the problem of navigating in the dynamics environment can be solved by minimizing the difference between the preferred control, $u*$ and a sampled control u' , which is subjected to a constraint by each moving obstacle. The algorithm is given in Algorithm 1,

Algorithm 1 Velocity Obstacle

```
1: for i = 0 to n do
2:     $u$ ← controls from set U
3:     $t_{lim} \leftarrow sample\ time\ limit \varepsilon(0,max]$
4:     $free \leftarrow true$
5:     $min \leftarrow \infty$
6:     for Moving Obstacles do
7:         let D(t) ← distance between A(t,u) and B(t)
8:         $t_{lim} \leftarrow min(D(t)^2)$
9:         $d \leftarrow D(t_{min})$
10:        if $d < r_b + r_b$ then
11:            $free \leftarrow false$
12:        end if
13:    end for
14:    if $\|u-u*\| < min$ then
15:        $min \leftarrow \|u-u*\|$
16:        $argmin \leftarrow u$
17:    end if
18: end for
19: return $argmin$
```

B. MPC

In this work, We are using NMPC to generate collision free trajectories for mobile robots with dynamic obstacles. For simplicity, We are considering mobile robot as a circle that can move on a 2D plane. This robot has 2 generalized coordinates, the motions in X-axis and Y-axis. The control inputs to the system is the velocity in XY directions. The formal definition of controller for this mobile robot is given by equation 10 and 11.

$$p=\begin{bmatrix}x & y\end{bmatrix} \tag{10}$$

$$u=\begin{bmatrix}\dot{x} & \dot{y}\end{bmatrix} \tag{11}$$

In every time step, we solve optimal control problem(OCP). The OCP is given by equation 12.

$$\int_{t=0}^{T} J_x(x(t),x_r(t))+J_u(u(t),u_r(t))+J_c(x(t))\,\mathrm{d}t+J_T(x(T)) \tag{12}$$

subject to:

$$\dot{x}=f(x,u)$$
$$u(t)\in U$$
$$G(x(t)<=0$$
$$x(0)=x(t_0)$$

Here, f is the state space model of the mobile robot. J_x ,J_y and J_c are the cost function for reference trajectory x_r tracking, control input penalty and collision cost function and J_T is the terminal cost function.

The first term in equation 3 penalizes the deviation of the predicted state x from the desired state x_r using equation 13 shown below.

$$J_x(x(t),x_r(t))=\|x(t)-x_r(t)\|^2_{Q_x} \tag{13}$$

Here, Q_x is a tuning parameter. Same way , the second term in equation 12 is cost function that is related to the penalty of the control input. This cost function is given by equation 14.

$$J_u(u(t), u_r(t)) = \|u(t) - u_r(t)\|^2_{R_u} \tag{14}$$

Here, R_u is a tuning parameter. The collision cost J_c is given by equation 15.

$$J_c(x(t)) = \sum_{j=1}^{N_{obstacle}} \frac{Q_{c,j}}{(1 + expk_j(d_j(t) - r_{th,j}(t))} \tag{15}$$

Here, $Q_{c,j}$ is a tuning parameter. $d_j(t)$ is Euclidean distance. And $r_{th,j}(t)$ is a threshold distance between obstacles and mobile robot. Threshold distance is an important parameter that can guarantee collision avoidance.

It is essential to predict the movement of dynamic objects as well for the implementation of NMPC. A constant velocity model can be used for prediction for simplicity. If more accuracy is required better sophisticated models can be used. Given the current position and the velocity of a dynamic obstacle, we can predict it's future trajectory in limited time window by following equation 16.

$$p_j(t) = p_j(t_0) + v_j(t_0)(t - t_0 + \delta) \tag{16}$$

δ compensates the delays that are caused by computation and communication. For better performance *t* can be set as a constant window.

The whole implementation of the above proposed method can be understood by the following pseudo code.

Algorithm 2 Nonlinear Model Predictive Control

```
 1: procedure NMPC
 2:     $obstacles \leftarrow$ Predict Obstacle Position
 3:     $x_r \leftarrow$ Compute Reference Trajectory
 4:     $Inputs \leftarrow$ Compute the optimized control inputs
 5:     $RobotState \leftarrow$ Update the Robot State
 6:     if $RobotState = Goal\ State$ then return True
 7:         close;
 8:     end if
 9:     goto top.
10: end procedure
```

Obstacle positions are predicted by using equation 7. The procedure for it is given by the Algorithms 3.

Algorithm 3 Predict Obstacle position

```
    procedure PREDICTION
2:      Initilize $p_j(t) Predictions = None$
        for $j \in TotalObstacles$ do
4:          $p_j(t) = p_j(t_0) + v_j(t_0)(t - t_0 + \delta)$
        end for
6:  end procedure
```

For a given robot position , the reference trajectory is computed in the direction of goal. This trajectory is divided in small steps. After reaching each of these steps, the obstacles path is predicted again. Before computing the optimized control inputs, a total cost consisting of tracking cost and collision cost is computed. For now , there is no penalty given for control input. If the robot actuators have control limits , it is important to consider cost for control inputs as well. Since , the mobile robot is used here considered as a circle moving in 2D plane, the penalty for control inputs may not be that effective in finding the optimal solution.

Algorithm 4 Cost computation

```
    procedure COST
        $RobotState \leftarrow Update\ the\ Robot\ State$
3:      $c1 \leftarrow Tracking\ cost$
        $c2 \leftarrow Collision\ Cost$ return $c1 + c2$
    end procedure
```

Once the cost is computed, the optimized control inputs are computed using Sequential Least SQuares Programming Optimizer (SLSQP). The inputs for the system in this case are velocity component in XY directions.

IV. Platform

You can see the Simulation Platform below. The blue connectors are modules which are implemented by us while the red lines are our setup.

We take the data from the Map and sensors as laser scans and the costmap converter converts them into obstacles which are then sent to the local planner. This gives it to Global Planner which provides waypoints to the Local planner. The local Planner has both VO and MPC in our demonstration these outputs are giving to steering angle.

Our Platform uses ROS [19][20] , Navigation Package, Costmap converter and Rviz for vizulization . The simulator is a 3D environment with obstacles modelled as point obstacles in blue.

The robot is in red it modelled as non-holonomic model it is has radius of 0.5 units . The robot is controlled by our steering node which converts ours map planners from velocities to steering velocity and velocity for non-holonomic robot.

The loop runs until the robot reaches the goal state.

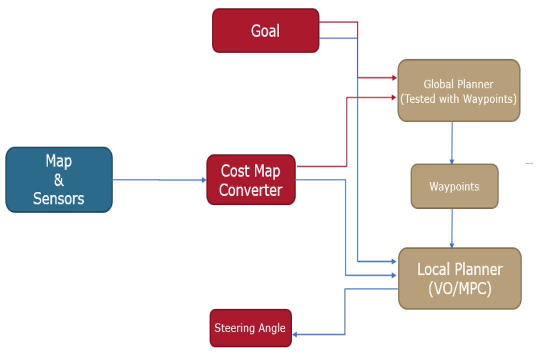


Fig. 5. Simulation Platform

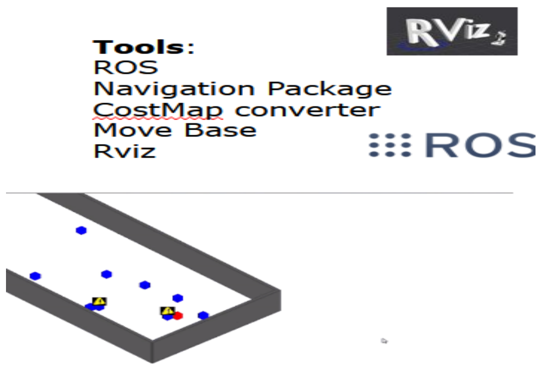


Fig. 6. Simulation Platform

V. Simulation Experiments

We conduct five simulation experiments:

a) One Dynamic Obstacle : In this case there are multiple static robots and a single dynamic robot.

b) All Dynamic Obstacle : In this case there are all dynamic robots moving up and down.

c) More Dynamic Obstacle : In this case there are 13 dynamic robots moving up and down.

d) All Dynamic random Obstacle : In this case there are all dynamic robots moving randomly.

e) No boundary : In this case there are all dynamic robots moving up and down but there is no boundary around the map.

VI. Metrics

We use four different metrics for our comparisons between algorithms.

A. Task completion time

Task completion time helps us compare if our algorithm is aggressive or passive enough . A lower task completion time is generally considered better. Our metric is calculated in seconds.

B. Clearance with respect to obstacles

We calculate the clearance w.r.t both dynamic and static obstacle. We look at the minimum distance during the traversal.

C. Length of traversed path

In this we calculate the length of our traversed path. A shorter path is preferred in this metric.

D. Distance Ratio

Distance travelled by the robot divided by euclidean distance between start and goal. Shorter is better.

VII. Velocity Obstacle Results

The robot navigation using Velocity Obstacle is tested in different types of environments. For each of the environment, different types of scenarios are considered. Results for each of these test cases are discussed in the below subsections.

A. One Dynamic Obstacle

Figure 8 shows the test environment for one dynamic obstacle. All the other obstacles are kept static. The Adhoc parameters used are given by Table 3. Table 4 shows the results we got from two experiments.

TABLE III
Adhoc Parameters for Velocity Obstacle

Adhoc Parameters	Value
Vmax	3.00
Vmin	0.50
Robot Radius	0.5

TABLE IV
Results for environment with one dynamic obstacle

Parameter	**Case 1**	**Case 2**
Distance Travelled	13.02	12.79
Distance Ratio	1.19	1.13
Time Taken	14.06	13.12
Closest Distance to Obstacle	1.24	1.38
Closest Obstacle Number	6	1

B. All Dynamic Obstacles

This experiment uses the same scenario as Figure 8, but instead all the obstacles are dynamic. The Adhoc parameters used and the results are listed in Table 5 and Table 6 respectively. Here to get a collision free path the minimum velocity parameters had to be reduced. It was shifted to zero to avoid an oscillatory behaviour for the mobile robot because of Ackermann Steering system.

TABLE V
Adhoc Parameters for Velocity Obstacle

Adhoc Parameters	Value
Vmax	3.00
Vmin	0.00
Robot Radius	0.3

TABLE VI
Results for environment with one dynamic obstacle

Parameter	**Case 1**	**Case 2**
Distance Travelled	12.69	12.45
Distance Ratio	1.12	1.10
Time Taken	13.09	12.89
Closest Distance to Obstacle	0.50	0.70
Closest Obstacle Number	6	4

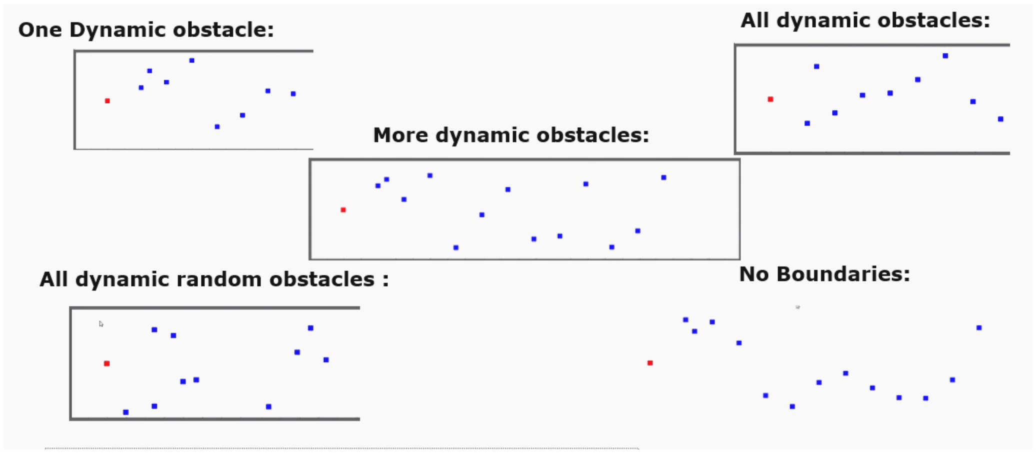


Fig. 7. Simulation Experiments

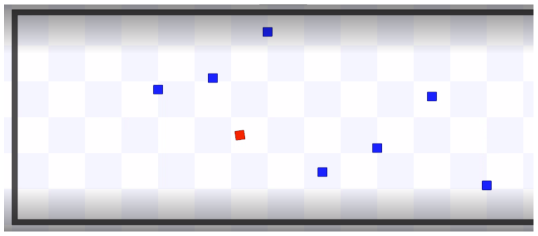


Fig. 8. Environment with static obstacles and one dynamic obstacle.

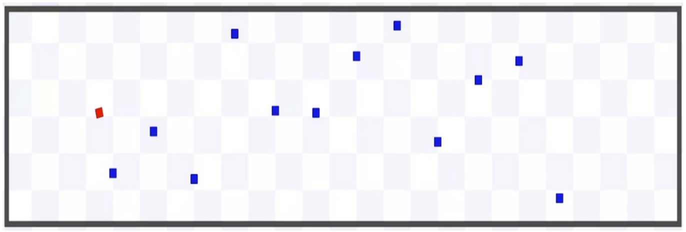


Fig. 9. Environment with all dynamic obstacles

C. All Random Dynamic Obstacles

This experiment uses the same scenario as Figure 8, but instead all the obstacles have random velocities. The Adhoc parameters used and the results are listed in Table 7 and Table 8 respectively. Here to get a collision free path the maximum velocity parameters had to be reduced. This way the mobile robot sampled velocities that did not cause problems with Ackermann Steering system on change.

TABLE VII
ADHOC PARAMETERS FOR VELOCITY OBSTACLE

Adhoc Parameters	Value
Vmax	2.00
Vmin	0.00
Robot Radius	0.3

TABLE VIII
RESULTS FOR ENVIRONMENT WITH ONE DYNAMIC OBSTACLE

Parameter	**Case 1**	**Case 2**
Distance Travelled	13.79	12.49
Distance Ratio	1.34	1.013
Time Taken	13.09	12.92
Closest Distance to Obstacle	0.48	1.70
Closest Obstacle Number	5	3

D. More Dynamic Obstacles

This experiment uses the scenario as Figure 9. All the obstacles have dynamic velocities int he Y direction. The Adhoc parameters used and the results are listed in Table 9 and Table 10 respectively. Here more dynamic obstacles were used for a larger map traversal. The robot was able to successfully reach the goal position.

TABLE IX
ADHOC PARAMETERS FOR VELOCITY OBSTACLE

Adhoc Parameters	Value
Vmax	2.50
Vmin	0.00
Robot Radius	0.3

TABLE X
RESULTS FOR ENVIRONMENT WITH ONE DYNAMIC OBSTACLE

Parameter	**Case 1**
Distance Travelled	25.15
Distance Ratio	1.19
Time Taken	21.07
Closest Distance to Obstacle	0.87
Closest Obstacle Number	11

E. Failure Cases

There were failure cases, which were found with exhaustive experimentation. These experiments helped in understanding the behaviour of different tuning parameters and their effect on the mobile robot control.

1) Hyperplane Separation: Since sampling velocity outside of the collision cone is not in our control, the mobile robot might sample a velocity which is correctly out of the collision cone but does not take into account the future movement of the robots. The obstacles may switch behaviour suddenly or come together in a situation where further sampling is difficult.

2) Low Radius Assumption: It was found that when the size of bounding box is small, the velocities sampled is much closer to desired velocity. The robot travels in an almost straight path however, there is no consideration of the size of the obstacle and leads to collision.

3) High Radius Assumption: It was found that when the size of bounding box is very large, the velocities sampled is much further from the desired velocity. The robot travels along a longer path since the velocity sampled is not inline with the desired velocity. The robot fails to go towards the goal when the number of obstacles is large.

4) High Maximum Velocity: When the maximum velocity is high then the sampled velocities would also be higher. This leads to problems when a high velocity is chosen. In case of an Ackermann Steering system it then becomes hard to suddenly move away in another direction if an obstacle is encountered.

5) Low Maximum Velocity: This does not cause any collision but the mobile robot is slow to move which is not an ideal solution for real world problems. The robot samples very small velocities almost stopping on encountering an obstacle.

6) High Minimum Velocity: This essentially means that we can avoid stopping the robot and there would always be a velocity for the robot to move. This is appropriate in static environments however, with a large number of obstacles the mobile robot suffers from oscillations and longer trajectories to traverse.

VIII. NMPC Results

The robot navigation using NMPC is tested in different types of environments. For each of the environment, different types of scenarios are considered. Results for each of these test cases are discussed in the below subsections.

A. One Dynamic Obstacle

Figure 10 shows the test environment that is used for this case. In this case, all the obstacles are static except the last one. This case is tested with two different initial positions of the last dynamic obstacle. Adhoc parameters used for this test case as shown in Table 11. Table 12 shows the results in this environment for two scenarios. The closest robot came to obstacle is 1.054 to the 6th obstacle in 2nd case. It took a minimum of 13.77 seconds to traverse the given path with a distance ratio of 1.11

B. All Dynamic Obstacles

For this case, same environment as shown in figure 10 is used, but all the obstacles are set to move in vertical direction back and forth. Same adhoc parameters as in Table 3 are used

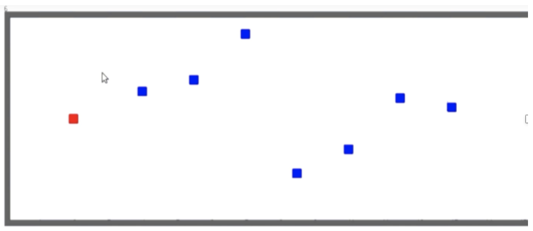


Fig. 10. Environment with one dynamic obstacle. Here the last obstacle is moving in vertical direction, and every other obstacle is static.

TABLE XI
Adhoc Parameters for NMPC

Parameter	Value
Simulation Time	2
Vmax	2
Collision Cost	1
Horizon Length	4

TABLE XII
Results for environment with one dynamic obstacle

Parameter	Case 1	Case 2
Distance Travelled	14.5	14.93
Distance Ratio	1.11	1.14
Time Taken	13.77	13.83
Closest Distance to Obstacle	1.099	1.054
Closest Obstacle Number	1	6

for this case as well. Results for this case are shown Table 13. In this case , NMPC took a minimum of 12.91 seconds to traverse the path and came closest to 7th obstacle at a distance of 1.03. The minimum distance ratio in this case is 1.023

TABLE XIII
Results for environment with all dynamic obstacles

Parameter	Case 1	Case 2
Distance Travelled	17.44	13.328
Distance Ratio	1.34	1.023
Time Taken	13.88	12.91
Closest Distance to Obstacle	1.021	1.03
Closest Obstacle Number	4	7

C. All Dynamic Obstacles Moving in Random Directions

In third test environment, the obstacles are set to move in random directions. The same adhoc parameters as before are used here as well. Robot took a minimum of 13.06 seconds to reach the goal with a distance ratio of 1.08 and came close to 5th obstacle at a distance of 0.95. Results for this are shown Table 14.

D. More Dynamic Obstacles

In the 4th test environment, we set more dynamic obstacles moving in vertical direction. Here we increased the horizon length to 8 to better accommodate the future obstacle positions because the more number of moving uncertainties. Here , robot traversed to the goal in 20.07 seconds with a distance ratio of

TABLE XIV
RESULTS FOR ENVIRONMENT WITH ALL DYNAMIC OBSTACLES MOVING IN RANDOM DIRECTIONS

Parameter	**Case 1**	**Case 2**
Distance Travelled	14.19	14.8
Distance Ratio	1.08	1.13
Time Taken	13.06	13.22
Closest Distance to Obstacle	1.04	0.95
Closest Obstacle Number	1	5

1.14. Closest distance to obstacle was 0.95. Adhoc parameters for this test case are given in Table 15 and the results are given in Table 16. Environment for this test case is given in figure 11.

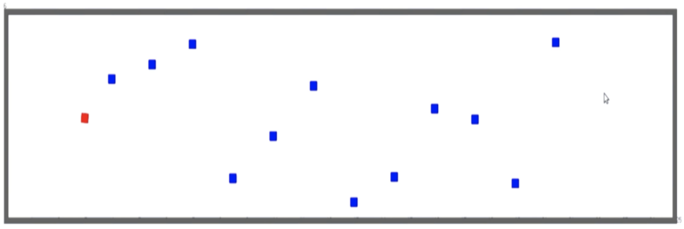


Fig. 11. Environment with more dynamic obstacles. Here all the obstacles are moving in vertical direction.

TABLE XV
ADHOC PARAMETERS FOR NMPC WITH MORE OBSTACLES

Parameter	**Value**
Simulation Time	2
Vmax	2
Collision Cost	1
Horizon Length	8

TABLE XVI
RESULTS FOR ENVIRONMENT WITH MORE DYNAMIC OBSTACLES

Parameter	**Case 1**
Distance Travelled	24.15
Distance Ratio	1.14
Time Taken	20.07
Closest Distance to Obstacle	0.95
Closest Obstacle Number	4

Further , we took the same environment and removed the static boundaries. In this case , the robot, as expected, robot took a longer path trying to avoid all the obstacles by moving away from them.In this case the robot reached goal in 39.43 seconds with a distance ratio of 1.73. Closest the robot came to an obstacle is 1.13. The environment for this test case in given in Figure 12. Results for this case is given by Table 17. Same adhoc parameters given in table 11 are used for this case.

E. NMPC- Failure Cases

Robot reached the goal successfully in all the above test cases. However, robot failed in many cases. For example, if we set the low collision cost, robot collided with the very first obstacle. This failure case is shown in figure 11. If the horizon length is set low, the robot could not compute the obstacle future positions properly and collided with 2nd obstacle. This case is shown figure 13. In all of our test cases , we assumed obstacles to be points. Because of this , though robot is successfully avoiding the collisions, we can still see robot grazing the obstacles. These failure cases are seen in figure 14. In our test environments , we did not consider the collision between obstacles. This can cause improper computations for NMPC. This failure case is given figure 15.

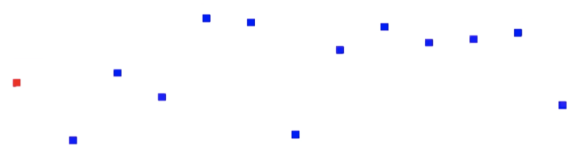


Fig. 12. Environment with more dynamic obstacles and no boundaries.In this case robot tried avoiding all the obstacle by going away from them.

TABLE XVII
RESULTS FOR ENVIRONMENT WITH MORE DYNAMIC OBSTACLES AND NO BOUNDARIES

Parameter	**Case 1**
Distance Travelled	36.45
Distance Ratio	1.73
Time Taken	39.43
Closest Distance to Obstacle	1.13
Closest Obstacle Number	11

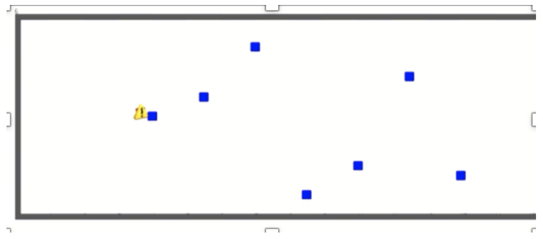


Fig. 13. Failure case because low collision cost. Robot tried going through the first obstacle because very low weightage is given to collision cost.

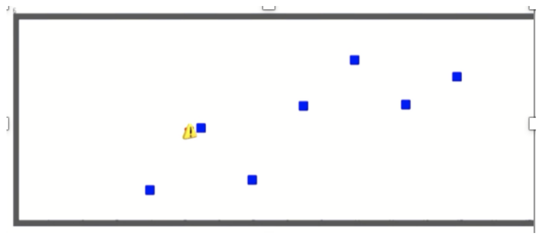


Fig. 14. Failure case because low horizon. Robot could not compute the future position of the second obstacle properly .

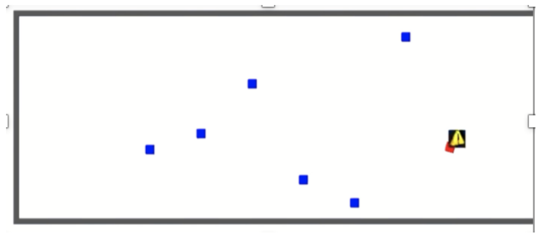


Fig. 15. Failure case because of considering obstacles as points. In this case robot reaches goal but glazes the obstacles in the path

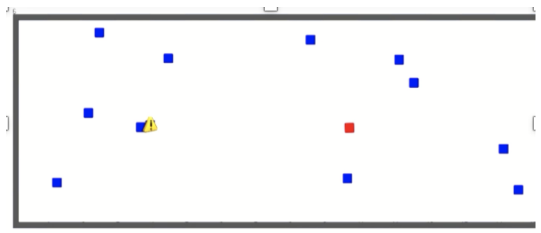


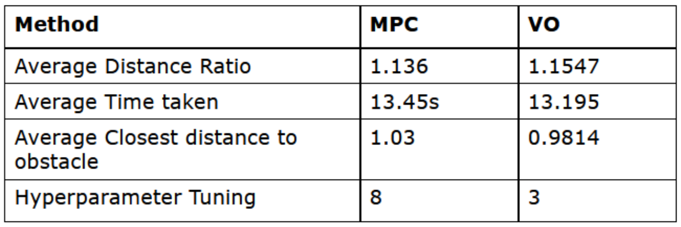
Fig. 16. Failure case because obstacles are hitting each other. Even in this case robot reached the goal, but the obstacles hitting each other can cause the robot to compute the obstacle locations incorrectly.

IX. Limitations

The limitations of our project is as follows:

- Points obstacles. These may cause collisions in real world.
- Parameter tuning for both MPC and VO is time consuming and difficult.
- May not work in a noisy environemnt.
- Path tracked in not optimal.
- Sampling of VO is not in our control as we see in hyperplane failures.
- Lots of failure cases in tuning the parameters this is not possible in real world scenarios

X. Conclusion



Method	MPC	VO
Average Distance Ratio	1.136	1.1547
Average Time taken	13.45s	13.195
Average Closest distance to obstacle	1.03	0.9814
Hyperparameter Tuning	8	3

Fig. 17. Final Comparisons

From the above comparisons we infer the following things:

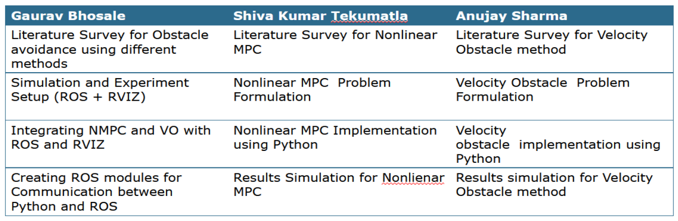
- A Lot of Hyper Parameter tuning is required based on environment conditions
- VO has a smaller compute.
- Online- tuning of parameters required
- MPC observes a safer distance w.r.t Obstacles
- Comparable times and distance ratio for both of them
- Lots of failure cases in tuning the parameters this is not possible in real world scenarios

After testing these algorithm we could conclude that MPC with Its ability to evaluate the future risk involved with a certain control behavior enables it to find a way out of very difficult situations. The simpler VO algorithm on the other hand, performs very well in the less complicated situations, but struggles in scenarios where multiple obstacles behave in a more random manner.

A lot more parameter tuning is required for an environment for both MPC and VO but in general we find MPC is more robust than VO due Its ability to evaluate the future risk involved with a certain control behavior enables it to find a way out of very difficult situations.

We see the application of our methods where robots needs to navigate environments such a hospital corridor with humans other robots , social robots such as in airports etc. This paper is able to show dynamic path planning and a comprehensive comparison between both MPC and VO.

XI. Task Division



Gaurav Bhosale	Shiva Kumar Tekumatla	Anujay Sharma
Literature Survey for Obstacle avoidance using different methods	Literature Survey for Nonlinear MPC	Literature Survey for Velocity Obstacle method
Simulation and Experiment Setup (ROS + RVIZ)	Nonlinear MPC Problem Formulation	Velocity Obstacle Problem Formulation
Integrating NMPC and VO with ROS and RVIZ	Nonlinear MPC Implementation using Python	Velocity obstacle implementation using Python
Creating ROS modules for Communication between Python and ROS	Results Simulation for Nonlienar MPC	Results simulation for Velocity Obstacle method

Fig. 18. Task Division Breakdown

References

[1] Model Predictive Control. (2022, March 8). In Wikipedia. https://en.wikipedia.org/wiki/Model_predictive_control.

[2] Kwon, W. H.; Bruckstein, Kailath (1983). ”Stabilizing state feedback design via the moving horizon method”. International Journal of Control. 37 (3): 631–643. doi:10.1080/00207178308932998.

[3] Garcia, C; Prett, Morari (1989). ”Model predictive control: theory and practice”. Automatica. 25 (3): 335–348. doi:10.1016/0005-1098(89)90002-2.

[4] Findeisen, Rolf; Allgower, Frank (2001). ”An introduction to nonlinear model predictive control”. Summerschool on ”The Impact of Optimization in Control”, Dutch Institute of Systems and Control. C.W. Scherer and J.M. Schumacher, Editors.: 3.1–3.45.

[5] J. Reif and M. Sharir, ”Motion planning in the presence of moving obstacles,” 26th Annual Symposium on Foundations of Computer Science (sfcs 1985), 1985, pp. 144-154, doi: 10.1109/SFCS.1985.36.

[6] Erdmann, Michael A. and Tomas Lozano-Perez. “On multiple moving objects.” Proceedings. 1986 IEEE International Conference on Robotics and Automation 3 (1986): 1419-1424.

[7] 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

[8] J. Snape, J. v. d. Berg, S. J. Guy and D. Manocha.The Hybrid Reciprocal Velocity Obstacle. IEEE Transactions on Robotics.

[9] J. van den Berg, Ming Lin and D. Manocha Reciprocal Velocity Obstacles for real-time multi-agent navigation 2008 IEEE International Conference on Robotics and Automation

[10] J. 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

[11] Li J, Sun J, Liu L, Xu J. Model predictive control for the tracking of autonomous mobile robot combined with a local path planning. Measurement and Control. 2021;54(9-10):1319-1325. doi:10.1177/00202940211043070

[12] D. Bojadzic, J. Kunze, D. Osmankovic, M. Malmir, en A. C. Knoll, “Non-Holonomic RRT & MPC: Path and Trajectory Planning for an Autonomous Cycle Rickshaw”, CoRR, vol abs/2103.06141, 2021.

[13] M. Kamel, J. Alonso-Mora, R. Siegwart, en J. I. Nieto, “Nonlinear Model Predictive Control for Multi-Micro Aerial Vehicle Robust Collision Avoidance”, CoRR, vol abs/1703.01164, 2017.

[14] D. H. Shim, H. J. Kim, and S. Sastry, “Decentralized nonlinear model predictive control of multiple flying robots,” in Decision and control, 2003. Proceedings. 42nd IEEE conference on, vol. 4. IEEE, 2003, pp. 3621–3626.

[15] D. Q. Mayne, E. C. Kerrigan, E. J. vanWyk and P. Falugi, “Tube-based robust nonlinear model predictive control,” Int. J. Robust Nonlinear Control 21(11), 1341–1353 (2011).

[16] J. Shin and H. J. Kim, “Nonlinear model predictive formation flight,” IEEE Trans. Syst. Man Cybern. Part A: Syst. Humans 39(5), 1116–1125 (2009).

[17] E. Scholte and M. E. Campbell, “Robust nonlinear model predictive control with partial state information,”IEEE Trans. Control Syst. Technol. 16(4), 636–651 (2008).

[18] D. Wilkie, J. van den Berg and D. Manocha, ”Generalized velocity obstacles,” 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009, pp. 5573-5578, doi: 10.1109/IROS.2009.5354175.

[19] ROS (Robot Operating System) and Rviz https://www.ros.org/

[20] TeB local planner http://wiki.ros.org/teb_local_planner

[21] D. Connell and H. M. La, ”Dynamic path planning and replanning for mobile robots using RRT,” 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2017, pp. 1429-1434, doi: 10.1109/SMC.2017.8122814.