Real-Time Collision-Free Navigation of Multiple Autonomous Vehicles

Fenil Desai
Robotics Department
Worcester Polytechnic Institute
Worcester, USA
fdesai@wpi.edu

Samarth Shah
Robotics Department
Worcester Polytechnic Institute
Worcester, USA
sshah5@wpi.edu

***Abstract*—This project proposes a decentralized collision avoidance strategy for multiple autonomous vehicles moving in an unstructured environment having static and fast-moving dynamic obstacles. The proposed method relies on "Optimal Reciprocal Collision Avoidance (ORCA)" and "Collision Avoidance with Acceleration-Velocity Obstacles (AVO)", and utilizes a model predictive control (MPC) strategy to generate local collision-free trajectories for each autonomous vehicle. It focuses on the performance comparison between ORCA and AVO, and also shows how OCRA and AVO performs better than Generalized Velocity Obstacles (GVO) approach [1].**

***Index Terms*—MPC, ORCA, AVO, GVO, Kinematic Bicycle Model**

I. Introduction

Decentralized collision avoidance is a fundamental problem in autonomous driving. The problem can generally be defined in the context of an autonomous vehicle navigating in an environment with obstacles and/or other moving entities, where the vehicle employs a continuous sensing-control cycle. In each cycle, the vehicle must compute an action based on its local observations of the environment, such that it stays free of collisions with the moving obstacles and the other vehicles, and progresses towards a goal. The solution to this problem will enable the navigation of autonomous vehicles in a dynamic environment. Another major issue is that autonomous vehicles have limited Field-of-View (FOV) and have to plan collision-free trajectory only by sensing obstacles in its FOV.

To solve this problem, this project considers the autonomous vehicles to be car-like, having kinematic bicycle model. This project considers a car model which takes velocity and steering rate as the control inputs for the kinematic model. Furthermore, to generate collision-free trajectories a global-planner along with a local-planner is utilized. It is observed that a variety of global-planner; like - Dijkstra's, A*; can be implement in such scenarios. Since, major focus of this project is on implementing a local-planner utilizing MPC it is assumed that the local planner receives a global path-plan from an A* planner.

One major consideration for designing a local-planner is that an autonomous vehicle may encounter another autonomous vehicle that actively makes decision based on its environment. A simple Generalized Velocity Obstacle (GVO) avoidance method will not be suffice in such conditions. Therefore, this project focuses on implementing ORCA [3] which specifically account for the reactive nature of the other vehicles without relying on coordination or communication among the vehicles. One major problem with GVO [1] and ORCA [3] is, these methods do not consider the fact that the dynamic obstacle (or another vehicle) may not be moving with a constant velocity and might have some acceleration, hence the project also focuses on implementing Acceleration Velocity Obstacle (AVO) [4] methods which mitigates particularly these issues. This project also focuses on the comparison between GVO, ORCA and AVO abased on performance matrices like:

- Time to complete task
- Clearance to obstacles
- Smoothness of the trajectory and control inputs
- Length of the traversed path

Contribution Breakdown	
Fenil Desai	**Samarth Shah**
MPC: Objective Function	MPC: Optimization
VO: Mathematical Formulation	VO: MPC implementation
AVO: Formulation and MPC implementation	ORCA: Formulation and MPC implementation

II. Related Work and Background

The local planner and controller used in this project is a Model Predictive Control. A detailed survey on methods for dynamic obstacle avoidance is presented in [2]. This survey compares numerous variants of Model Predictive Control for dynamic obstacle avoidance. Detailed Implementation of Model Predictive Control is presented in [5] and [6].

A. Model Predictive Control

Model Predictive Control is an optimal control strategy which consists of motion model, objective function, and an optimizer. With $\mathbf{x}_0$ as the initial state of the system, at

each τ sampling interval, the state model is used to predict behaviour of system over $N\tau$ planning horizon. The optimizer optimizes the future control inputs and minimizes the objective function over the entire planning horizon. Therefore, providing a sequence of optimal control inputs. However, only the first optimal control input is applied to the system. The same procedure is repeated with updated $\mathbf{x}_0$ and shifting the sequence of control inputs.

1) Simple Feedback Principle:

- At decision instant k, measure the state $\mathbf{x}_k$
- Based on $\mathbf{x}_k$, compute the best sequence of actions:

$$\mathbf{u}(\mathbf{x}_k) := (\mathbf{u}(k;\mathbf{x}_k)\ \mathbf{u}(k+1;\mathbf{x}_k)\ ...\ \mathbf{u}(k+2;\mathbf{x}_k)\ ...) \tag{1}$$

- Apply the control $\mathbf{u}(k;\mathbf{x}_k)$ on the sampling period $[k, k+1]$
- At decision instant $k+1$, measure the state $\mathbf{x}_{k+1}$
- Based on $\mathbf{x}_{k+1}$, compute the best sequence of actions:

$$\mathbf{u}(\mathbf{x}_{k+1}) := (\mathbf{u}(k+1;\mathbf{x}_{k+1})\ \mathbf{u}(k+2;\mathbf{x}_{k+1})\ ...) \tag{2}$$

- Apply the control $\mathbf{u}(k+1;\mathbf{x}_{k+1})$ on the sampling period $[k+1, k+2]$
- Repeat

To tackle dynamic obstacles while locally planning the path of the autonomous vehicles, this project heavily relies on the General Velocity Obstacle approach and its variants like ORCA, and AVO. Elaborate understanding on velocity obstacle methods are presented in : [7], [8], [9], [10] and [11].

B. Velocity Obstacle

The idea is to define a set of velocities that would, if used as a control for the agent, lead to a collision with an obstacle at some time in the future.

For a disc-shaped agent A and a disc-shaped moving obstacle B with radii r_A and r_B, respectively, the velocity obstacle for A induced by B, denoted by $VO_{A|B}$, is the set of velocities for A that would, at some point in the future, result in a collision with B. This set is defined geometrically. Let ρ_A and ρ_B be the center points of A and B, respectively, and let $\mathcal{B}$ be a disc centered at ρ_B with a radius equal to the sum of A's and B's. This is generalized as the Minkowski sum. If B is static (i.e. not moving), we could define a cone, $\mathcal{C}$, of velocities for A that would lead to a collision with B as the set of rays shot from pA that intersect the boundary of B. To derive a velocity obstacle from this, we simply translate the cone $\mathcal{C}$ by the velocity v_B of B, as shown in Figure 1. More Formally it can be written as Equation 3.

$$VO_{A|B} = \{\mathbf{v}|\exists t > 0 :: \rho_A + t(\mathbf{v} - \mathbf{v}_B) \in \mathcal{B}\} \tag{3}$$

For many kinematically constrained robots, such as car like robots, the set of feasible velocities at any instant is a single velocity – the specific velocity in the direction of the rear wheels. One way to workaround this constraint and use velocity obstacles is to use the set of velocities that can be achieved over some time interval τ. However, this approach does not guarantee collision-free navigation: consider the set of velocities, V , that a car-like robot can reach after τ seconds. VOs can be constructed for the robot, and a velocity from V outside the VOs can be selected to navigate the robot. In this case, a car-like robot will actually need to follow an arc to achieve the selected velocity, and the VOs provide no guarantee that this arc will be collision-free. Additionally, the robot will be at a different position when it achieves the selected velocity, but the VOs that were computed using the robot's initial position, and thus the velocity is no longer guaranteed to be collision-free. To alleviate these issues, we can select a small value for τ , but this has some implications for navigation: as τ decreases, the set of velocities that are being considered by the robot becomes smaller, and the robot can miss feasible controls.

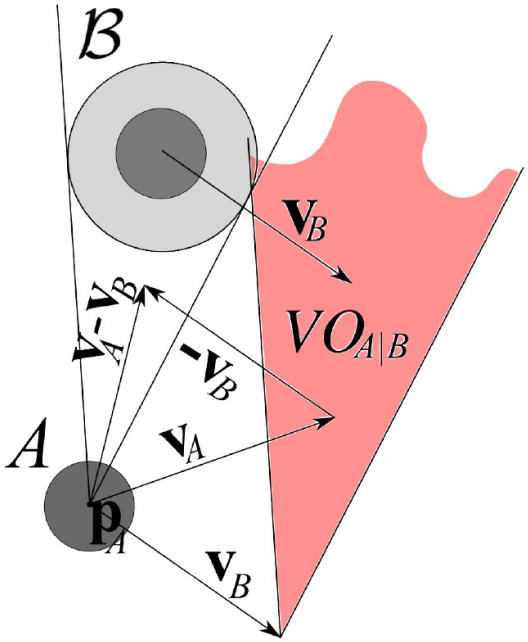

Fig. 1. The velocity obstacle $VO_{A|B}$ for robot A relative to dynamic obstacle B. The cone of velocities that would lead to a collision in the static case is translated by v_B. Note that as the velocity v_A is inside the velocity obstacle, the relative velocity, v_A - v_B, leads to a collision.

Generalized velocity obstacle concept and seeks to address the difficulty of using velocity obstacles with kinematically constrained agents (e.g. car-like robots).

The generalized obstacle can be defined as follows. Given an obstacle B, let us denote its position at time t by $B(t)$. Similarly, given the position of agent A at time t = 0 and a control u, let us denote the position agent A will have after undertaking u for time interval t by $A(t;u)$. Here, a control u is a set of inputs to the Bicycle Kinematic [12] [13] [14] [15] model that results in a change in the robot's configuration. If we continue to restrict our attention to agents and obstacles that are circularly shaped, we can define the obstacle in the

control space as

$$\{u|\exists t > 0 :: ||A(t,u) - B(t)|| < r_A + r_B\} \tag{4}$$

Given a specific set of kinematic constraints for some system, we can apply the $VO_{A|B}$ formulation as follows. Let $t_{min}(u)$ be the time at which the distance between the centers of A and B is minimal for a given control u for A:

$$t_{min}(u) = arg_{t>0} min ||A(t,u) - B(t)|| \tag{5}$$

If a closed expression for $t_{min}(u)$ is obtained, for example by solving $\frac{\partial ||A(t,u)-B(t)||}{dt} = 0$ for t, then it may also be possible that an explicit equation for the velocity obstacle can be found by solving $||A(t,u) - B(t)|| < r_A + r_B$ for u.

For the cases when a closed form solution is not obtainable, this approach can be used in a sampling scheme. In this case, controls in U are sampled. For each control u, the minimum distance that agent A and obstacle B would achieve were A to use control u is numerically calculated. This distance determines whether the control would be collision free.

III. Approach

Here, in this project a simple **Global Planner** like **A*** is used to generate a high-level path-plan. This global plan does not consider the static and dynamic obstacles in the map and simply generates a shortest path to reach the goal. This is utilized so as to save the local planner from getting stuck in any of the local minima. For the sake of implementation it is assumed that the global planner returns a list of intermediate goal nodes which is fed to the local planner. The goal for the local planner is set to an intermediate goal in the beginning. Once the vehicle reaches in the vicinity of that intermediate goal, the goal of the local planner is updated to the next intermediate goal-node and this process is repeated until the vehicle reaches the ultimate GOAL pose.

Here it is assumed that the vehicle is capable of completely and accurately measuring its own state as well as the states of the surrounding obstacles and vehicles. Moving forward with this assumption, the **Local Planner** used here is a **Model Predictive Control**. For MPC different problems are formulated in terms of objective function, path constraints and control bounds. In other words, an optimization problem is formulated which can be solved by an optimizer. For objective functions having non-linearity, as in case of a car-like vehicle, closed loop solutions for the optimization function does not exists and utilizing online optimizations become necessary to solve the optimization problem. The formulation of MPC for the problem under consideration is as follows:

A. Model Predictive Control

*1) **Motion Model**:* This project utilizes a car-like **Kinematic Bicycle Model** to generate results that are more realistic and implementable in real-word scenarios for navigation of multiple autonomous vehicles. The discrete 2-D bicycle kinematic model of a car-like vehicle is given by the following equations.

$$\mathbf{x}_{t+1} = \mathbf{x}_0 + \sum_{t=0}^{t=n-1} \mathbf{f}(\mathbf{u}_t), \tag{6}$$

where states $\mathbf{x}_t = (x_t, y_t, \theta_t, \delta_t)$ consists of the x and y position, θ orientation and δ steering angle of the vehicle. The control inputs $\mathbf{u}_t = (v_t, \phi_t)$ are linear velocity and steering rate. The motion model $\mathbf{f}$ is defined as the following, where τ is the sampling time.

$$\mathbf{f}(\mathbf{u}_t) = \begin{pmatrix} v_t cos\theta_t \tau \\ v_t sin\theta_t \tau \\ v_t tan\delta_t \tau \\ \dot{\phi_t} \tau \end{pmatrix} \tag{7}$$

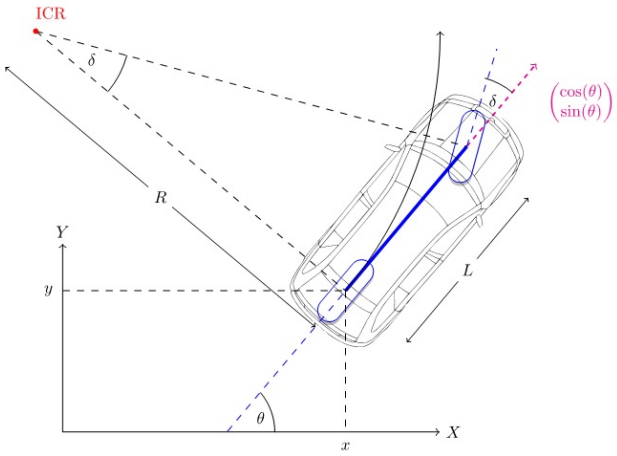

Fig. 2. Kinematic Bicycle Model

*2) **Objective Function**:* The optimization problem for MPC is generally modelled as a cost function which depends on the states of the vehicle and the control inputs. The following are the various types of costs that are considered for the implementations in this project.

- **Goal Reach Cost**
  In a case where the vehicle is to be navigated to a desired goal pose is (x_g, y_g, θ_g). A simple quadratic problem is

formulated as follows

$$\min \sum_{k=0}^{k=N-2} c(x_k, y_k) + c(x_{N-1}, y_{N-1}) + (\theta_{N-1} - \theta_g)^2 \tag{8}$$

$$-v_{max} \leq v_k \leq v_{max}, \forall k \tag{9}$$

$$-\phi_{max} \leq \phi_k \leq \phi_{max}, \forall k \tag{10}$$

$$c(x_k, y_k) = (x_k - x_g)^2 + (y_k - y_g)^2 \tag{11}$$

where, N is the prediction horizon, (9), (10) are bounds on control inputs $\mathbf{u}_k$. Intuitively, the first term of the objective (8) reduces the squared distance-to-goal over $N-1$ predictions. The last two terms are terminal costs or cost at $(N-1)^{th}$ prediction. Assigning a higher weight to the terminal cost enforces the optimizer to generate a trajectory with its last prediction as (x_g, y_g, θ_g). Consequently, at each optimization step, better final position costs are obtained.

- **Smoothness Cost**
  To ensure the trajectory of the robot is smooth, $\dot{v}_k$ and $\dot{\phi}_k$ is added to the cost (8). The derivative of control $\mathbf{u}_k$ is calculated by the finite difference over the sampling time τ.

$$\dot{v}_k = \frac{v_{k+1} - v_k}{\tau} \tag{12}$$

$$\dot{\phi}_k = \frac{\phi_{k+1} - \phi_k}{\tau} \tag{13}$$

- **Static Obstacle Cost**
  To avoid static obstacles in the path, a conventional quadratic collision model is added to the cost (8). The collision cost is defined as

$$c_{obs} = \begin{cases} \sum_{k=0}^{N-1} \sum_{i=0}^{M-1} (dist_i(x_k, y_k))^{-1} & dist_i \leq d_{offset} \\ 0 & else \end{cases} \tag{14}$$

  where $dist_i$ is the distance-to-obstacle, and d_{offset} is the distance above which the cost is zero. Here, each obstacle is modeled as a circle.
- **Constraint Reformulation**
  The constraints from (9) and (10) can be reformulated as costs as follows:

$$\lambda_v = max(0, v_k - v_{max}) + max(0, -v_k - v_{max}) \tag{15}$$

$$\lambda_\phi = max(0, \phi_k - \phi_{max}) + max(0, -\phi_k - \phi_{max}) \tag{16}$$

  Another major constraint that can affect the planning is the maximum steering angle that a car can have, let δ_{max} be the maximum allowable steering angle to the car. This constraint can be reformulated to a cost as:

$$\lambda_\delta = max(0, \delta_k - \delta_{max}) + max(0, -\delta_k - \delta_{max}) \tag{17}$$

- **Total Cost**
  The total optimization cost function can be written as the weighted sum of the Goal Cost, Smoothness Cost, Static Obstacle Cost and Constraint Costs.

*3) **Optimizer**:* Optimizer is the cardinal component of Model Predictive Control. The unconstrained optimization problem formulated above is non-smooth and non-convex, and has a similar form to the common loss function encountered while training deep neural networks. Therefore, this project utilizes the state-of-the-art gradient descent variant RMSProp. RMSProp approximates the curvature of the loss function without computing the Hessian. The steps of RMSProp is given by the following equations,

$$E[\nabla^2]_k = 0.9E[\nabla^2]_{k-1} + 0.1[\nabla]_k^2 \tag{18}$$

$$\mathbf{u}_{k+1} = \mathbf{u}_k - \frac{\eta}{\sqrt{E[\nabla^2] + \epsilon}}[\nabla]_k \tag{19}$$

where $[\nabla]_k$ is gradient, $E[\nabla^2]$ is mean squared gradient, η is learning rate, and ϵ is a constant to avoid zero denominator. To compute the gradient of the optimization function, Autograd [16] package is utilized which efficiently computes numerical gradient of a given function.

Model Predictive control is a very robust approach for avoiding static obstacles. It also performs well in avoiding dynamic obstacles to an extent. However, due Kinematic Bicycle Model with steering rate as the control input is a highly non-linear model and hence, MPC fails at many instances to avoid dynamic obstacles. This led to the inclusion of the Velocity Obstacle methods for dynamic obstacle avoidance. The following subsection discusses various variants of the Velocity Obstacle method and how it has been modeled as cost functions and added to Total Objective Cost function of MPC, which forces the optimizer to find a velocity that avoids a dynamic obstacle.

B. Velocity Obstacle Methods

*1) **Generalized Velocity Obstacle**:* To check the collision between a moving vehicle and a dynamic obstacle, this method assumes that the obstacle will keep moving in the same direction with same velocity for the next $\mathcal{T}$ time steps. The GVO method states that t_{min} should be found using (5). However, a closed loop solution for t_{min} does not exist for the non-linear cases and since car-like kinematic model is highly non-linear this approach for GVO formulation might not work. So to mitigate this issue another approach is to propagate the vehicle with its control input u for $\mathcal{T}$ time steps and propagating the obstacle in its direction of motion with a constant velocity, which is assumed to be correctly measured by the vehicle. Furthermore, to add it as a cost to the MPC objective function, the velocity can be penalized based on how close the vehicle and the obstacle will be after $\mathcal{T}$ time steps.

$$dist_{GVO} = ||(Vehicle(\mathcal{T}, u) - Obstacle(\mathcal{T}, v))|| \tag{20}$$

where, $Vehicle(\mathcal{T}, u)$ is the vehicles position after $\mathcal{T}$ time steps given control input u and $Obstacle(\mathcal{T}, v)$ is the position

of the obstacle after moving with v velocity for $\mathcal{T}$ time steps. The cost to be added to the MPC objective function can be modeled as follows:

$$C_{GVO} = \begin{cases} (dist_{GVO})^{-1} & dist_{GVO} \leq d_{offset} \\ 0 & else \end{cases} \tag{21}$$

However, GVO approach assumes that dynamic obstacles move passively through the environment without active perception of the environment. However, this is not always the case, in the problem under consideration, all the vehicles are smart autonomous vehicles which can act as a dynamic obstacle for another vehicle. It is necessary to consider the fact that those vehicles are not just any passively moving obstacles and it is equally important to take into account the fact that those dynamic obstacles also make efforts to avoid the collision. If the mentioned considerations are not made a phenomenon called **Reciprocal Dance** occurs between the two vehicles. To avoid this issue Reciprocal Velocity Obstacle (RVO) or Optimum Reciprocal Collision Avoidance (ORCA) method has been introduced.

*2) **Optimum Reciprocal Collision Avoidance**:* The general formulation of the ORCA method states that, a velocity that lies outside of the combined Velocity Obstacle region formed by the velocity obstacles of all the autonomous vehicles present in the environment. This essentially means that the two vehicles that might possibly undergo collision share the responsibility of the collision in half and half. This avoids them from doing the reciprocal dance and hence results in a better and smoother control inputs for both the vehicles.

The MPC modelling for this method is slightly different from that of GVO. Simply adding one more cost term to the objective function does not solves the purpose. However, the control input that is fed to the vehicle after every optimization loop can be altered in a way that it accounts for the fact that all the other vehicles share half responsibility for the collision between the vehicle and itself. Therefore the new control input that should be fed to the vehicle under consideration can be modeled as follows:

$$u_{k+1} = \frac{u_k + u_{opt}}{2} \tag{22}$$

where, u_{k+1} is the new optimal control input at time step $k+1$, u_k is the optimal control input fed to the vehicle at time step k and u_{opt} is the optimal control input that is returned by the optimizer after optimizing the cost function.

One limitation of the ORCA method is that it does not consider the fact that with advancement in the vehicle sensing technology the vehicles might be able to accurately measure the current acceleration of other vehicles in the environment and can better predict their pose after $\mathcal{T}$ time steps in the future. To overcome this limitation, a new method called Acceleration Velocity Obstacle method has been introduced in this project.

*3) **Acceleration Velocity Obstacle**:* In this approach, the acceleration of the dynamic obstacle is measured after observing the obstacle over multiple time steps and performing an iterative differentiation of the velocity of the obstacle. This acceleration and the current velocity is used as to estimate the position of the obstacle after $\mathcal{T}$ time steps. On the other hand, the optimal velocity is selected from the Acceleration Velocity Obstacle formed by propagating the obstacle in the same way as it was done in the case of GVO, but here the velocity obstacle is formed after considering the acceleration. Since the kinematic bicycle model is considered in this project the control input is still the velocity and not the acceleration. To implement the AVO method, it cannot be simply added to the objective function of the MPC therefore, the control input that is to fed to the kinematic model after every time step is modeified as follows:

$$u_{k+1} = u_k + (\frac{u_{opt} - u_k}{\mathcal{T}})t \tag{23}$$

where, u_{k+1} is the optimal control input fed to the vehicle at the time instance $k+1$, u_k is the optimal control input that was fed to the vehicle at the time instance k and u_{opt} is the optimal control input obtained after optimizing the objective function at the time instance $k+1$

IV. Experiment

Numerous experiments were performed to validate the performance of the above mentioned methods and to study their limitations. The dashed (- -) line with small squares represents the path returned by a global planner, the star represents the Start, the diamond represents the Goal and the solid line (–) represents the path taken by the vehicle for the simulation results in all the experiments. The color of the articles corresponds to different vehicles in the environment. All the experiments with results and theoretical inferences are as follows:

A. ***Experiment 1:*** *Simple Goal Reach Problem using MPC*

This experiment involves testing how well the MPC performs in a simple case where the global planner returns a global path which does not consider the fact that there might some local obstacles like (potholes, and stop-signs). The MPC is utilized to locally plan the trajectory and navigate thorough the environment following the global path. The results for this experiment can be observed in Figure. 3, 4, 5, 6, 7 and 8. It is clearly visible that the control inputs are within the desired bounds and the vehicle successfully avoids the static obstacle while it reaches the goal position.

B. ***Experiment 2:*** *MPC successfully avoiding Dynamic Obstacles*

This experiment involves testing how well the MPC performs in a simple case where the global planner returns a global path which does not consider the fact that there might some static local obstacles like (potholes, and stop-signs) and some dynamic local obstacles like pedestrian, animals or other vehicles. The MPC is utilized to locally plan the trajectory and

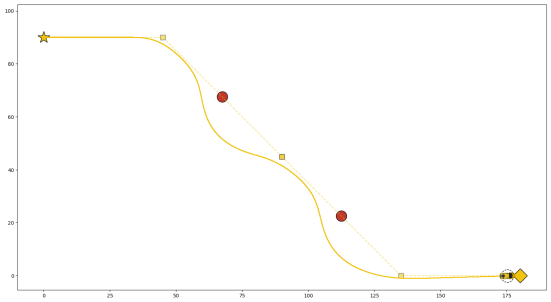

Fig. 3. Simulation result for Experiment 1

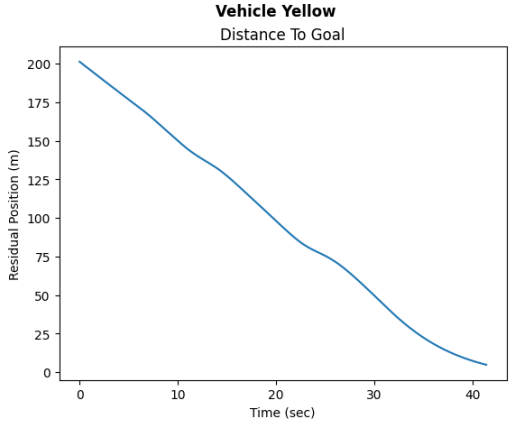

Fig. 4. Distance to Goal evolution for Experiment 1

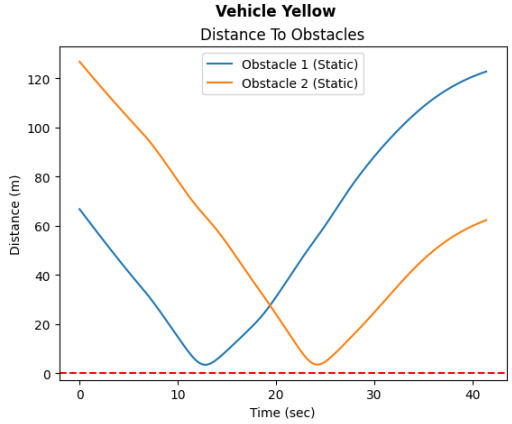

Fig. 5. Distance from all the obstacles for Experiment 1

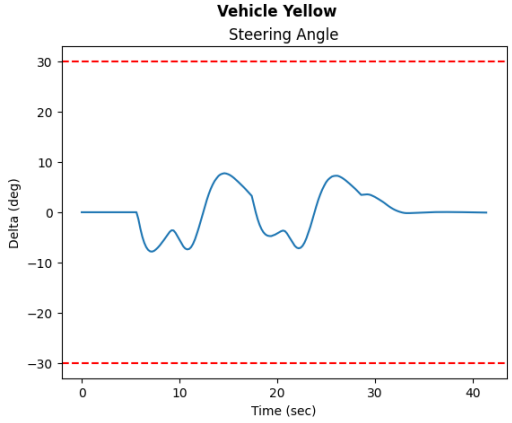

Fig. 6. Steering Angle evolution for Experiment 1

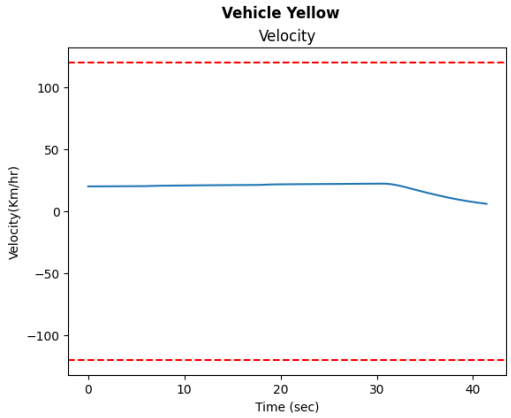

Fig. 7. Control Input 1: Velocity for Experiment 1

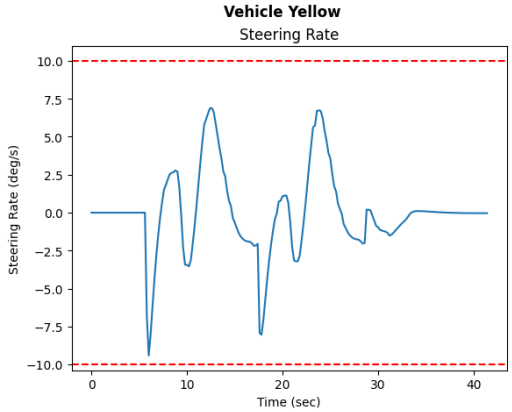

Fig. 8. Control Input 2: Steering Rate for Experiment 1

navigate thorough the environment following the global path. The results for this experiment can be observed in Figure. 9, 10, 11, 12, 13 and 14. It is clearly visible that the control inputs are within the desired bounds and the vehicle successfully avoids the static obstacle while it reaches the goal position.

C. *Experiment 3: MPC fails avoiding Dynamic Obstacles*

In this experiment it is observed that sometimes when the environment consists of fast moving dynamic obstacles utilizing simple MPC formulation is not sufficient for dynamic obstacle avoidance. The results for this experiment can be observed in Figure. 15, 16, 17, 18, 19 and 20. It is clearly visible that the vehicle is not able to avoid the dynamic obstacle and collides with the obstacle, shown by (X) in the simulation, despite the steering rate (control input) reaching its maximum limits as it is clearly seen in Figure. 17 and 20.

D. *Experiment 4: Generalized Velocity Obstacle with MPC*

In this experiment it is observed that including the generalized velocity obstacle costs in the objective function solves the issues faced by the MPC as witnessed in experiment 3. The results for this experiment can be observed in Figure. 21, 22, 23, 24, 25 and 26. It is clearly visible that the vehicle is able to avoid the dynamic obstacle that it was not able to do

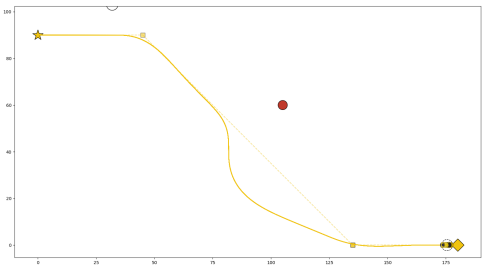

Fig. 9. Simulation result for Experiment 2

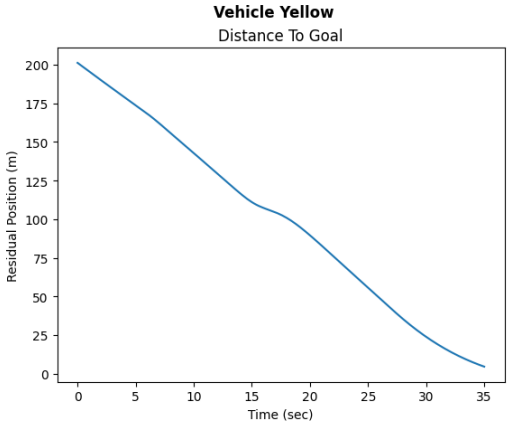

Fig. 10. Distance to Goal evolution for Experiment 2

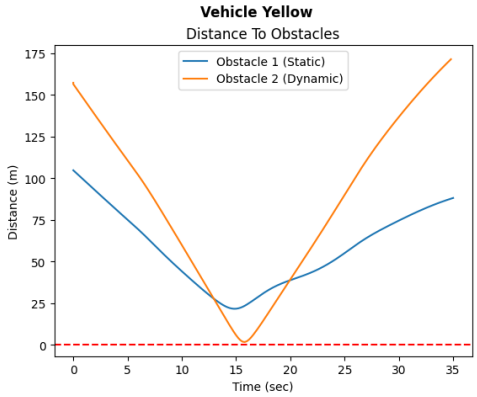

Fig. 11. Distance from all the obstacles for Experiment 2

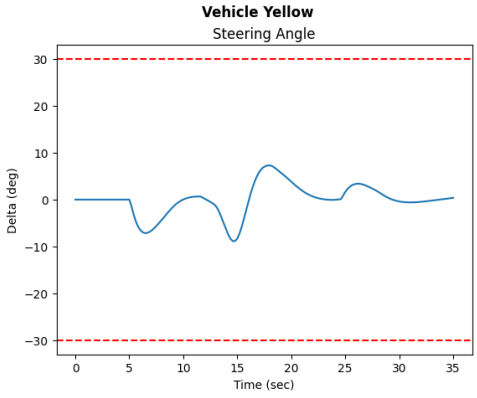

Fig. 12. Steering Angle evolution for Experiment 2

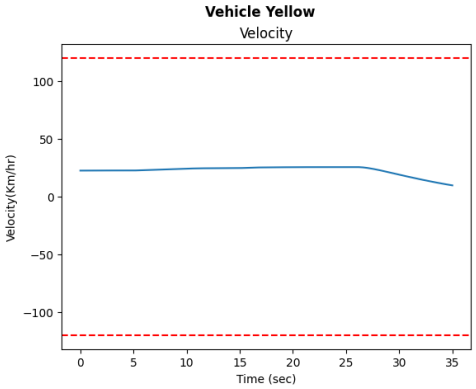

Fig. 13. Control Input 1: Velocity for Experiment 2

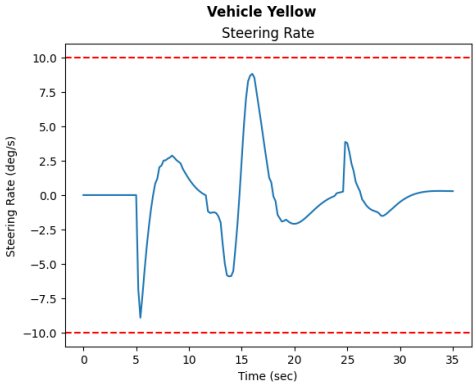

Fig. 14. Control Input 2: Steering Rate for Experiment 2

in experiment 3 and the steering rate (control input) also does not cross its maximum bounds as seen in seen in Figure. 23 and 26.

*E. **Experiment 5:** Multiple Vehicles using GVO and MPC*

In this experiment all the three vehicles Red, Blue and Yellow uses the GVO method. All the vehicles model their counter parts as a passive dynamic obstacle that does not make any accommodation for the potential collision between them and hence tries to solely compensate for the collision. This leads to the Reciprocal Dance between these vehicles as mentioned in the previous sections. The results for this experiment can be seen in Figure. 27, 28, 29, 30 and 31. All the other parameters seen in the previous experiments evolves similarly for experiment 5 as well. Here, the jitter in the steering rates (control inputs) for all 3 vehicles is very evident and points out the drawback of the GVO method in case of decentralized navigation of multiple vehicles.

*F. **Experiment 6:** Multiple Vehicles using ORCA and MPC*

In this experiment all the three vehicles Red, Blue and Yellow uses the ORCA method. Here, all the vehicles does not model their counter parts as a passive dynamic obstacle that do not make any accommodation for the potential collision between them, but considers the fact that the other dynamic entity is a smart vehicle and will make efforts to avoid the

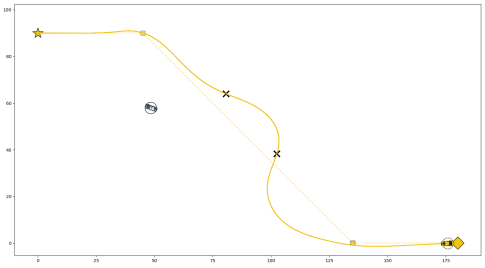

Fig. 15. Simulation result for Experiment 3

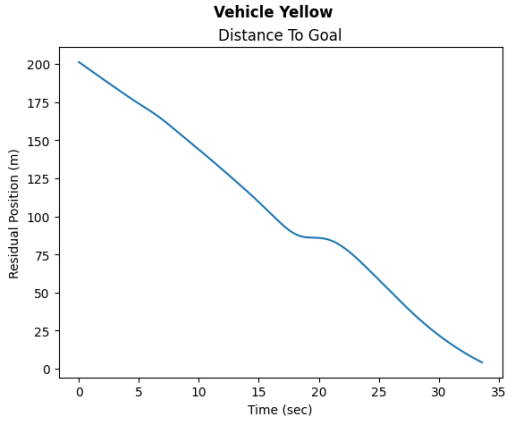

Fig. 16. Distance to Goal evolution for Experiment 3

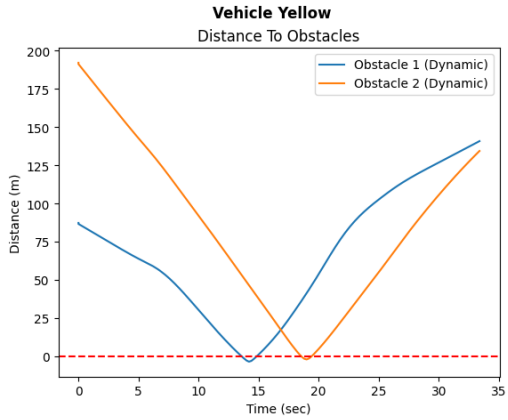

Fig. 17. Distance from all the obstacles for Experiment 3

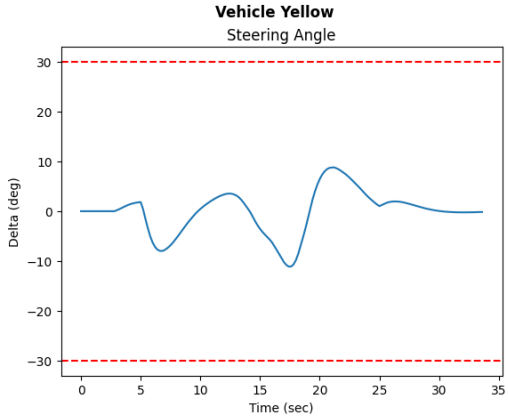

Fig. 18. Steering Angle evolution for Experiment 3

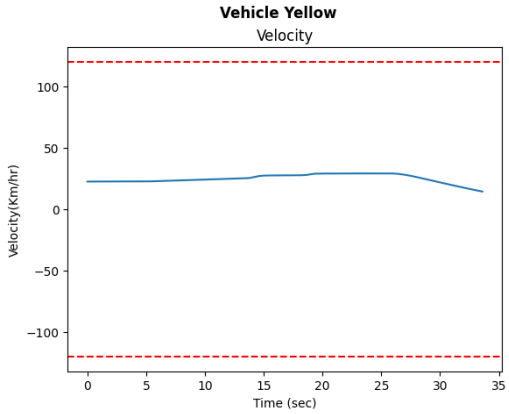

Fig. 19. Control Input 1: Velocity for Experiment 3

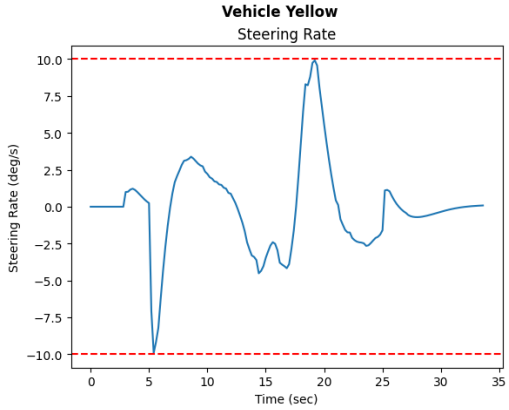

Fig. 20. Control Input 2: Steering Rate for Experiment 3

collision at some future time step. This leads to the avoiding the Reciprocal Dance between these vehicles as mentioned in the previous sections. The results for this experiment can be seen in Figure. 32, 33, 34, 35 and 36. All the other parameters seen in the previous experiments evolves similarly for experiment 6 as well. Here, as compared to experiment 5 the jitter in the steering rates (control inputs) for all 3 vehicles is not visible and points out a positive point of the ORCA method in case of decentralized navigation of multiple vehicles. Here, it is also evident form the output statistics of both experiment 5 and 6, seen in Figure. 31 and 36, that time taken to complete the task is less in case of ORCA method but the path length ins higher in ORCA as compared to the GVO method.

*G. **Experiment 7:** Comparing GVO with AVO*

As mentioned in the previous sections, another limitation of the GVO method is that it does not consider the fact that a dynamic obstacle might not be moving with a constant velocity in the environment but might also have some acceleration at each time step. Another method introduced to mitigate this issue was AVO method. AVO method assumes that the acceleration of the dynamic obstacle is accurately measurable after observing for some time steps. The comparison between the AVO and GVO methods are observed in the Figure. 37, 38,

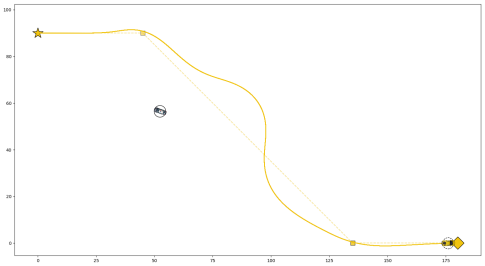

Fig. 21. Simulation result for Experiment 4

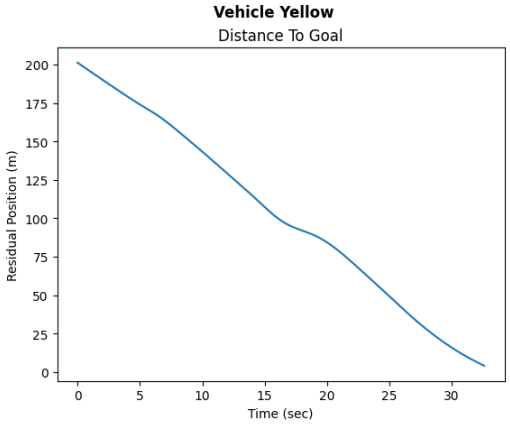

Fig. 22. Distance to Goal evolution for Experiment 4

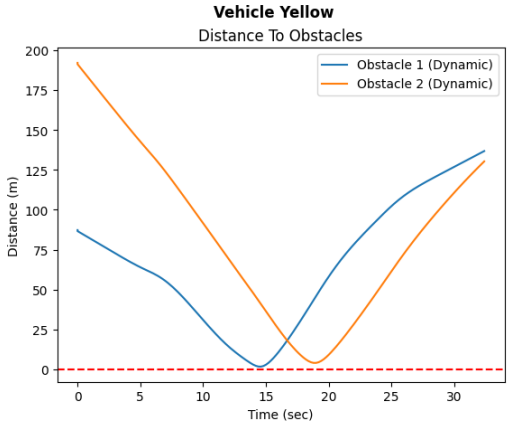

Fig. 23. Distance from all the obstacles for Experiment 4

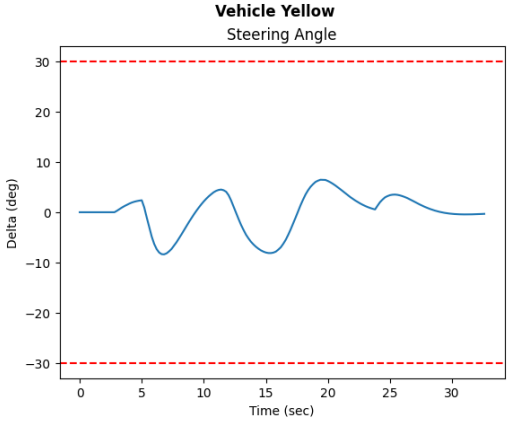

Fig. 24. Steering Angle evolution for Experiment 4

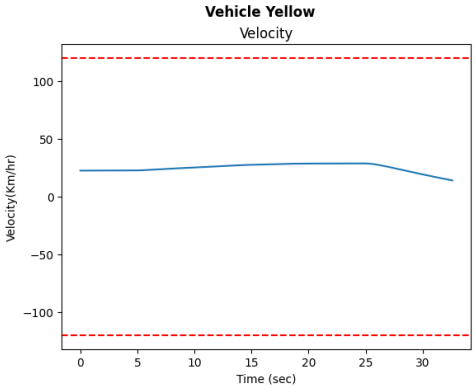

Fig. 25. Control Input 1: Velocity for Experiment 4

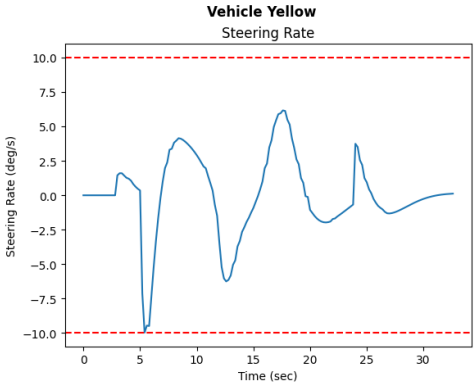

Fig. 26. Control Input 2: Steering Rate for Experiment 4

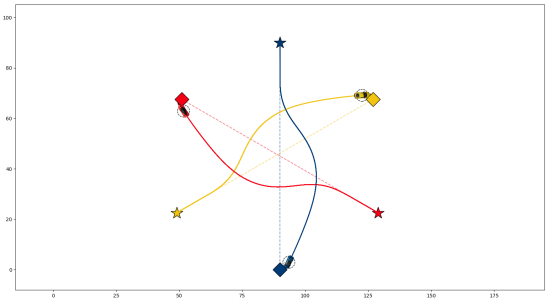

Fig. 27. Simulation result for Experiment 5

39, 40, 41 ans 42. Here, the Red vehicle is using GVO method along with MPC, whereas the yellow vehicle uses AVO along with MPC. The cross mark ”X” on the red trajectory implies that the red vehicle collided with the dynamic obstacle at that particular point. It can be observed that the red vehicle was still not able to avoid the dynamic obstacle despite it going way beyond the steering rate limit which is evident from the Figure. 38 and 39. On the other hand, the yellow vehicle using the AVO method managed to avoid the obstacle while staying within the steering rate limit, which can be seen in Figure. 40 and 41. It is also evident form the output statistics available in figure. 42 that AVO method not only avoided the obstacle but observed a shorter path length as well as shorter completion

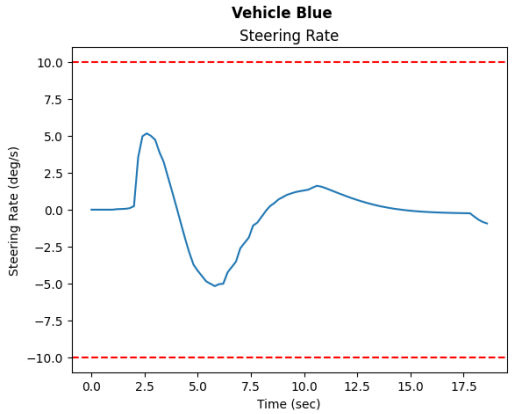

Fig. 28. Steering Rate for Blue Vehicle in Experiment 5

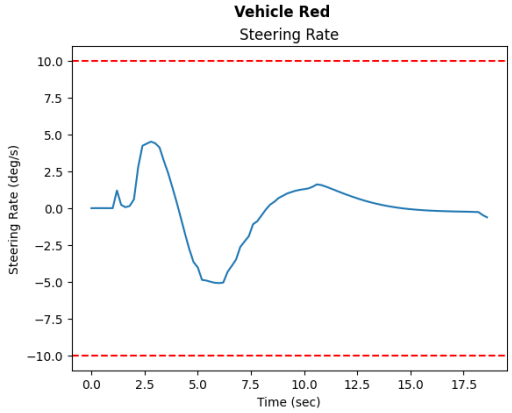

Fig. 29. Steering Rate for Red Vehicle in Experiment 5

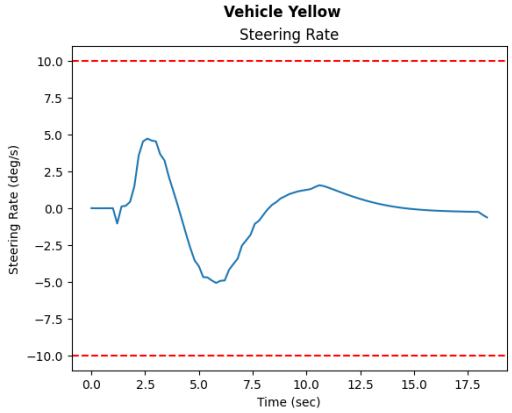

Fig. 30. Steering Rate for Yellow Vehicle in Experiment 5

```
Vehicle Yellow:
 Global Path Length = 90.0499861188218 m
 Local Path Length = 96.97995011924604 m
 Time Taken = 18.399999999999967 secs
Vehicle Red:
 Global Path Length = 90.0499861188218 m
 Local Path Length = 92.70590102532017 m
 Time Taken = 18.599999999999966 secs
Vehicle Blue:
 Global Path Length = 90.0 m
 Local Path Length = 111.69205762544257 m
 Time Taken = 18.599999999999966 secs
```

Fig. 31. Output Statistics for Experiment 5

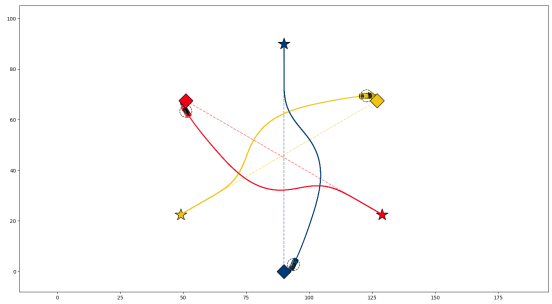

Fig. 32. Simulation result for Experiment 6

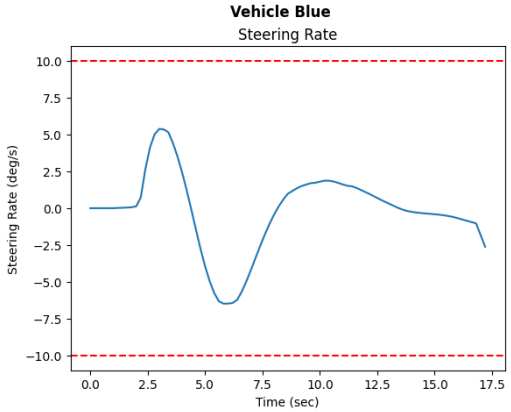

Fig. 33. Steering Rate for Blue Vehicle in Experiment 6

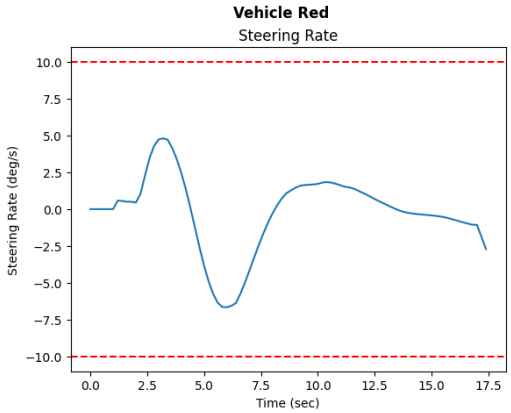

Fig. 34. Steering Rate for Red Vehicle in Experiment 6

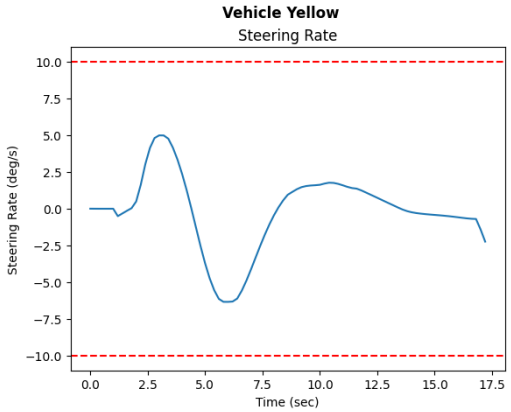

Fig. 35. Steering Rate for Yellow Vehicle in Experiment 6

```
Vehicle Yellow:
 Global Path Length = 90.0499861188218 m
 Local Path Length = 97.58977162847228 m
 Time Taken = 17.19999999999997 secs
Vehicle Red:
 Global Path Length = 90.0499861188218 m
 Local Path Length = 93.87091938994634 m
 Time Taken = 17.39999999999997 secs
Vehicle Blue:
 Global Path Length = 90.0 m
 Local Path Length = 121.6377482928465 m
 Time Taken = 17.19999999999997 secs
```

Fig. 36. Output Statistics for Experiment 6

time.

H. ***Experiment 8:*** *Drawback of using AVO with MPC*

Here, in first part of this experiment all the four vehicles are using the AVO method. It can be very evidently observed from Figure. 43, that the trajectories formed are oscillating in nature as PID controller having very low damping values. This oscillating nature is not desirable for navigation of autonomous vehicles. In another case, two vehicles Yellow and Red are using AVO method whereas the other two vehicles use the GVO method. It is observed form the Figure. 44, that the vehicles using GVO do not tend to oscillate as much as the ones with AVO does. This is one of the major benefit of using GVO over AVO.

V. Limitations

MPC is computationally very expensive as it involves open-loop optimization of the objective function at every time step. This leads to higher requirements for computational resource. One major assumption involved in this project is that the global path planner provides approximate desired angles at the intermediate nodes which is not always possible, additional mathematics is needed to calculate the approximate desired orientation for all the intermediate points based on the current orientation of the vehicle. Another major issue is that this project does not cover better alternative to tackle the limitations of the most advance method AVO. This project covers toy problem implementation of all the discussed methods, therefore, here always the control horizon is assumed to be 1, i.e, the MPC cost is optimized before the vehicle completes 1 time step and hence the new control input is applied to the vehicle at every time step. However, in real world scenarios this might not be possible due to limitations of the computational resources and hence the output of the experiments may be degraded.

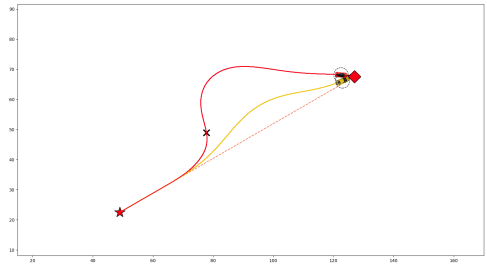

Fig. 37. Simulation result for Experiment 7

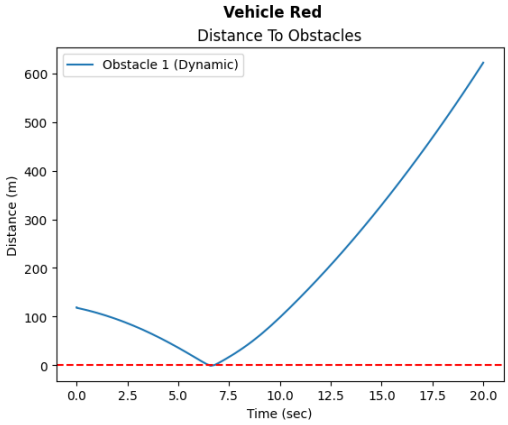

Fig. 38. Distance from all the obstacles for Red Vehicle in Experiment 7

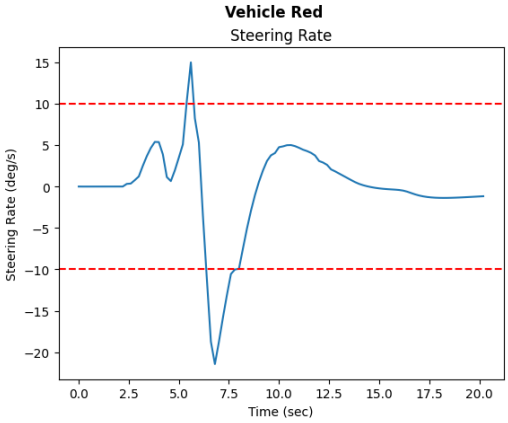

Fig. 39. Steering Rate for Red Vehicle in Experiment 7

References

[1] 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.

[2] Hoy, M., Matveev, A., Savkin, A. (2015). Algorithms for collision-free navigation of mobile robots in complex cluttered environments: A survey. Robotica, 33(3), 463-497. doi:10.1017/S0263574714000289

[3] Alonso-Mora J, Breitenmoser A, Beardsley P, et al. (2012) Reciprocal collision avoidance for multiple car-like robots. In: IEEE international conference on robotics and automation.

[4] 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, 2011, pp. 3475-3482, doi: 10.1109/ICRA.2011.5980408.

[5] D. Q. Mayne and S. Rakovic, "Model predictive control of constrained piecewise affine discrete-time systems," Int. J. Robust Nonlinear Control 13(3–4), 261–279 (2003).

[6] A. Richards and J. P. How, "Robust distributed model predictive control," Int. J. Control 80(9), 1517–1531 (2007).

[7] Fiorini P and Shiller Z (1998) Motion planning in dynamic environments using velocity obstacles. The International Journal of Robotics Research 17: 760–772.

[8] Van den Berg J, Ling M and Manocha D (2008) Reciprocal velocity

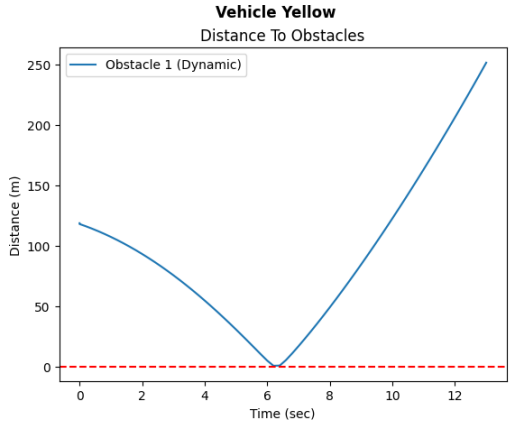

Fig. 40. Distance from all the obstacles for Yellow Vehicle in Experiment 7

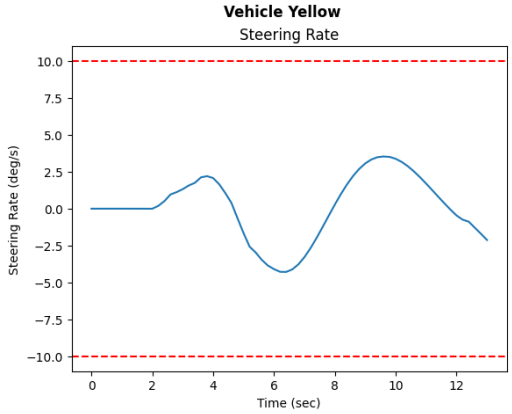

Fig. 41. Steering Rate for Yellow Vehicle in Experiment 7

```
Vehicle Yellow:
 Global Path Length = 90.0499861188218 m
 Local Path Length = 259.90123104293497 m
 Time Taken = 12.999999999999986 secs
Vehicle Red:
 Global Path Length = 90.0499861188218 m
 Local Path Length = 286.69829845696216 m
 Time Taken = 20.19999999999996 secs
```

Fig. 42. Output Statistics for Experiment 7

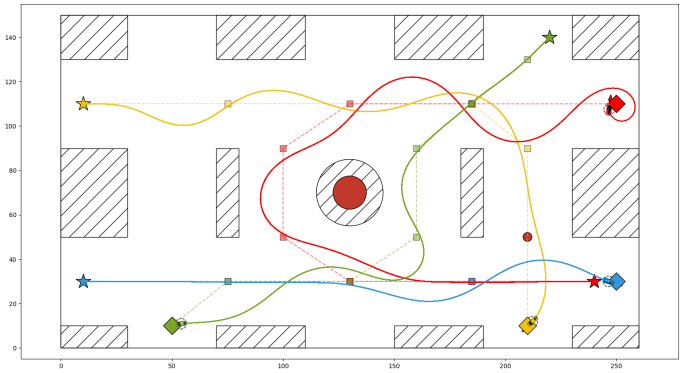

Fig. 43. Simulation result for all the vehicles using AVO in Experiment 8

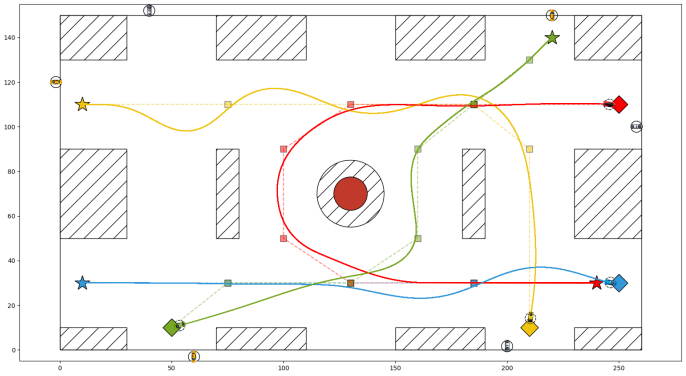

Fig. 44. Simulation result for two vehicles GVO and two using AVO in Experiment 8

obstacles for real-time multi-agent navigation. In: IEEE international conference on robotics and automation

[9] Van den Berg J, Guy S, Lin M, et al. (2009) Reciprocal n-body collision avoidance. In: Proceedings of the international symposium of robotics research.

[10] Bareiss, Daman, and Jur van den Berg. "Generalized reciprocal collision avoidance." The International Journal of Robotics Research 34, no. 12 (2015): 1501-1514

[11] Van den Berg J, Snape J, Guy S, et al. (2012) Reciprocal collision avoidance with acceleration-velocity obstacles. In: IEEE international conference on robotics and automation.

[12] L. Gracia and J. Tornero, "Kinematic modeling and singularity of wheeled mobile robots," Adv. Robot. 21(7), 793–816 (2007).

[13] L. Gracia and J. Tornero, "Kinematic modeling of wheeled mobile robots with slip," Adv. Robot. 21(11), 1253–1279 (2007).

[14] K. Kozlowski, Robot Motion and Control (Springer, London, 2009).

[15] A. Micaelli and C. Samson, "Trajectory tracking for unicycle-type and two-steering wheels mobile robots," Technical Report INRIA: Technical Report No: 2097, Institut national de recherche en informatique et en automatique (1993).

[16] Dougal Maclaurin, David Duvenaud, and Ryan P Adams. "Autograd: Effortless Gradients in Numpy". In: ICML 2015 AutoML Workshop. 2015.