Obstacle Avoidance of Autonomous Vehicle using MPC

Subhransu Mahapatra
line 2: *Mechanical Engineering and Applied Mechanics*
University of Pennsylvania
Philadelphia, USA
subhu@seas.upenn.edu

Samarth Kalluraya
line 2: *Mechanical Engineering and Applied Mechanics*
University of Pennsylvania
Philadelphia, USA
subhu@seas.upenn.edu

s

***Abstract*—Motion Planning and control in dynamic environment requires real-time localization, planning and control inputs. This paper assumes localization data is available and uses a combination of A* and potential function for path planning in an environment with moving obstacles. The intention of using a potential function is based on its simplicity, safety and low computational cost. Once the path has been generated an MPC controller with finite horizon is deployed to give a suboptimal input to the vehicle.**

Keywords—component, formatting, style, styling, insert (*key words****)***

I. INTRODUCTION

As robots become ubiquitous the interaction with other robots and humans increases which demands for more dynamic planning and control methodologies. When dynamic obstacles are present, a robot must plan around their present and predicted future trajectories, updating its plan in real time at a high enough frequency to remain reactive to its surroundings [1]. In this paper we will deploy a strategy of using a global and local planner. For the global planner we use A* strategy which is computationally intensive but provides the shortest route to the goal in an environment. The A* algorithm is run during the start of the navigation process and doesn't take into account dynamic obstacles. For the local planner we use the potential field approach because of its safety, simplicity and less computationally intensive. This combination helps to attenuate the problem of local minima which plagues the usability of potential field functions and computationally intensive nature of A* (or RRT, RRT*) in case of dynamic obstacle in which case one has to rerun the search algorithm in specific time intervals. Anytime A* algorithm is also an efficient way to address the problem in discussion and can produce suboptimal solutions in an anytime manner [2]. Anytime graph search algorithms are not included in the scope of this paper and are left as future work on comparison of search algorithms in dynamic environments.

Use of MPC has been used to solve real world problems in multiple domains. The motivation of using an MPC has been motivated by its ability of incorporating both state and control constraints into the control variables during the evolution of state [3]. Use of MPC in an autonomous vehicle uses a mathematical optimization taking into account the vehicle dynamic and constraint of the environment into account and provides optimal input solutions. The cost function can be tuned as per usability. In this paper the cost function has been tuned to provide comfort and safety with minimal jerks. The obstacles move in a random fashion, so an online-MPC is run in a closed-loop to ensure optimal control inputs in a dynamic environment.

II. PRELIMINARIES

A. A* Algorithm

```
let the openList equal empty list of nodes
let the closedList equal empty list of nodes
put the startNode on the openList (leave if it's f at zero)

while the openList is not empty
  let the currentNode equal the node with the least f value
  remove the currentNode from the openList
  add the currentNode to the closedList

  if currentNode is the goal
        Congrats! You've found the end! Backtrack to get
        path

  for each child in the children
    if child is in the closedList
      continue to beginning of for loop
    child.g = currentNode.g + distance between child and …
                … current
    child.h = distance from child to end
    child.f = child.g + child.h
     if child.position is in the openList's nodes positions
       if the child.g is higher than the openList node's g
         continue to beginning of for loop
     add the child to the openList
```

For relatively smaller number of nodes, A* provides an efficient way that guarantees a shortest path solution. We define a cost constituting of the cost of reaching a node from the starting position *g(n)* and the estimated cost of the node to the goal *h(n)*. We combine this two to get *f(n)* which contains the information of both the cost. In the algorithm we choose nodes with the least *f(n)*.

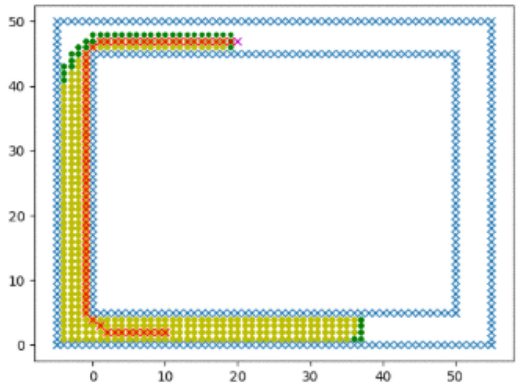

Fig 1 A Path Planning*

B. Potential Function

In its simplest of form a Potential Function constitutes an attractive potential (1) towards the goal and a repulsive potential (2) away from the goal. The vehicle moves in the direction of negative gradient or the least potential. Potential function doesn't provide solutions in an environment with local minima. To address this problem, we use potential function in localized areas in conjunction with a A* (or any global planner) algorithm.

$$V_{att} = \alpha \left\| d_{vehicle}(x,y,t) - d_{goal}(x,y) \right\| \quad (1)$$

$$V_{rep} = \beta e^{-\|d_{vehicle}(x,y,t) - D_{obstacle}(x,y,t)\|} \quad (2)$$

Where V_{att} is the attractive potential field, α is the attractive potential coefficient, $d_{vehicle}(x,y,t)$ is the vehicle location in the Cartesian coordinate system, $d_{goal}(x,y,t)$, V_{rep} is the repulsive potential field, β is the repulsive potential coefficient and $D_{obstacle}(x,y,t)$ are the co-ordinates of current and predicted positions of the moving obstacles.

The attractive potential is a linear function which decreases as the vehicle nears the goal. The repulsive potential is an exponential which ensures the potential rises exponentially as the vehicle approaches the obstacle and has no effect when the vehicle is far away. To account for the velocity of the moving obstacle we use a matrix of predicted position based on its current velocity to create the repulsive field.

C. MPC Formulation

The philosophy of receding horizon is, at time t to solve an optimal control problem over a finite future horizon of steps N and at time t +1 acquire future measurements, repeat the optimization and so on [3]. The control equation for a discrete system can be formulated as (3)

$$x(t+1) = A(t)x(t) + B(t)u(t) \quad (3.1)$$

$$y(t+1) = C(t)x(t) + D\,(t)u(t) \quad (3.2)$$

Where $x(t) \in \mathbb{R}^n$, $y(t) \in \mathbb{R}^m$, $u(t) \in \mathbb{R}^k$ are the state, output and control input of the system. $A(t)$ is the system matrix linearized at a time t, $B(t)$ is the input matrix and , $C(t)$ is the output matrix. $x(t)$ and $u(t)$ are members of a convex set subject to set of linear constraints.

As we are predicting the points into the future based on the model, we state that the system is in an open loop with the goal to drive the system towards the goal. Based on the state, input we define a cost function for this optimization problem to provide optimal inputs based on the constraint (4).

$$C = T(x_N) + \sum_{k=0}^{N-1} G(x_k, u_k) \quad (4)$$

Where $T(x_N)$ is the cost at the horizon and $G(x_k, u_k)$ are the stage cost. The states must satisfy system dynamics (3.1) and the applied constraints.

$$H(x_k, u_k) < 0 \; k \in 0 \text{ to } N-1 \quad (5)$$

In case of a finite horizon there might be constraint on the final state which we formulate as (6).

$$x_k \in X \quad (6)$$

The stage and the final cost can be formulated using positive definite matrices.

$$G(x_k, u_k) = x^T Q x + u^T R u \quad (7.1)$$

$$T(x_k, u_k) = x^T P x \quad (7.2)$$

Where Q, R and P positive definite matrix. P is so chosen so as to make the system drive towards the final goal.

The output obtained by optimizing the cost function provides the state vector and the input vector for the N points in the horizon. Using this we give input *u*(t)* to the system and obtain the feedback of the state at *x(t+1).*

D. Stability

The stability of a linear MPC can be proved from the cost function. Provided constraints on *u* and *x* are met in the optimized solution, the cost at the horizon must contain a Lyapunov function for the system [3].

III. Vehicle Model and Linearization

The vehicle was modeled according to the kinematics of a bicycle model. In this model we assume that the vehicle only has a one front wheel and one rear wheel for simplification of the model. The control inputs for the model are the acceleration a given to the

vehicle and the steering angle of the front wheel δ, with the assumption that only the front wheel is used for steering. The center of the vehicle is assumed to be at the midpoint of the rear axle and in case of the bicycle model, at the rear wheel. We define the state of the system as

$$S = [x, y, v, \Psi]$$

where x and y are the position coordinates of the vehicle, Ψ is the yaw of the vehicle and v is its velocity. The model can then be written as [4]

$$\dot{x} = v * \cos(\Psi)$$
$$\dot{y} = v * \sin(\Psi)$$
$$\dot{v} = a$$
$$\dot{\Psi} = \frac{v * \tan(\delta)}{L}$$

In this equation L is the wheelbase. This model can be then represented as,

$$\dot{S} = A' * S + B' * u$$

or as a a function of state and input,

$$\dot{S} = f(S, u)$$

Where A' is the Jacobian of the state and B' is the Jacobian of the control input [5]

$$A' = \begin{bmatrix} 0 & 0 & \cos(\Psi) & -v * \sin(\Psi) \\ 0 & 0 & \sin(\Psi) & v * \cos(\Psi) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{\tan(\delta)}{L} & 0 \end{bmatrix}$$

$$B' = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & \frac{v}{L * cos^2(\delta)} \end{bmatrix}$$

We can convert this model into discrete time analysis by discretizing using a sampling time dt. Thus, the state at the next time step can be calculated as,

$$S_{k+1} = S_k + \dot{S} * dt$$

Using Taylor series expansion up to first degree around $\hat{S}$ and $\hat{u}$ we get,

$$S_{k+1} = S_k + \left(f(\hat{S}, \hat{u}) + A'S_k + B'u_k - A'\hat{S} - B'\hat{u}\right)dt$$

$$S_{k+1} = (I + dtA')S_k + (dtB')u_k + \left(f(\hat{S}, \hat{u}) - A'\hat{S} - B'\hat{u}\right)dt$$

This can be simplified as

$$S_{k+1} = A * S_k + B * u_k + C$$

Where,

$$A = \begin{bmatrix} 1 & 0 & \cos(\Psi)\,dt & -v * \sin(\Psi)\,dt \\ 0 & 1 & \sin(\Psi)\,dt & v * \cos(\Psi)\,dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{\tan(\delta)}{L}dt & 1 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ dt & 0 \\ 0 & \frac{v}{L * cos^2(\delta)}dt \end{bmatrix}$$

$$C = \begin{bmatrix} v * \sin(\Psi) * \Psi * dt \\ -v * \cos(\Psi) * \Psi * dt \\ 0 \\ \frac{v * \delta}{L * cos^2(\delta)}dt \end{bmatrix}$$

Along with the kinematic model it is necessary to define other vehicle constraints which are listed below,

Maximum speed = 15 m/s
Maximum reverse speed = 5 m/s
Maximum steering angle = 45°
Maximum steering rate = 30°/s
Maximum acceleration = 2 m/s^2
Maximum deceleration = 6 m/s^2

IV. IMPLEMENTATION

To control the autonomous car to travel from a start point to and end point, three different modules are used. Firstly, the map in which the car will operate, the starting location and the goal is defined and provided to the program. Then the A* planner is executed on the known environment to generate a global path from the start to goal. The main purpose of running an A* algorithm is to generate a path that will not get stuck in local minima's that might be present on the map if we use only a potential field. In this step we do not consider any obstacles on the map. We then discretize the path provided by the A* algorithm into multiple waypoints spaced equally apart. These waypoints will act as intermediate potential field goals. That means, initially only the waypoint closest to the starting position of the car will generate an attractive potential field. As the car comes close to this waypoint it will cease to attract and the goal will shift to the next closest point. This way these intermediate waypoints will guide the car across the map. The main purpose of having such potential field goals is that the car will be able to navigate through real-time obstacles which will have a repulsive field on them. The potential field planner is executed at every time-step and it generates a path up to few time-steps in the future. Since this path generated using potential fields is updated continuously, moving obstacles can also be taken into consideration as the path finds the best route around them.

The path planning algorithm gives a short, desired path for the car to follow. The model predictive controller runs in a continuous loop to ensure that the car does follow that path. In order to do this the controller takes as input, the current state of the vehicle

and the desired path for the vehicle and a predicted set of control inputs for the future time steps till the horizon. The desired states consist of states found at points on the actual path that the car should be at in the future time-steps based on its current velocity. The desired velocity is always the velocity which is set by the user at the start of the program. The controller then calculates a predicted set of states for the same number of time-steps using the given set of control input values. It uses these predicted states to approximate the kinematic model of the vehicle at those particular points. It then finds the cost of its actions. The cost is a summation of all the errors in the desired and current states, the inputs and the rate of change of inputs, each multiplied by its own specific weight. The cost at each point *i* is calculated as,

$$C = \sum_{i=0}^{N-1}\Big(Q\begin{bmatrix}(x_i - x_{i_{des}})^2\\(y_i - y_{i_{des}})^2\\(v_i - v_{i_{des}})^2\\(\Psi_i - \Psi_{i_{des}})^2\end{bmatrix} + R_1\begin{bmatrix}a_i\\\delta_i\end{bmatrix} + R_2\begin{bmatrix}a_{i+1} - a_i\\\delta_{i+1} - \delta_i\end{bmatrix}\Big) + P\begin{bmatrix}(x_f - x_{f_{des}})^2\\(y_f - y_{f_{des}})^2\\(v_f - v_{f_{des}})^2\\(\Psi_f - \Psi_{f_{des}})^2\end{bmatrix}$$

Where, Q is a 4x4 diagonal matrix and R_1 and R_2 are 2x2 diagonal matrices. The total cost at each time step is summation of all cost for cost at all points *i* till the horizon. The point on the horizon won't have cost associated with rate of change of inputs.

Once the total cost resulting due to a set of control inputs is calculated, an optimizer tries to optimize the current states and control inputs so as to reduce the value of this cost. While optimizing following considerations are taken into consideration,

- $S_{i+1} = A * S_i + B * u_i + C$ (vehicle model)
- Maximum and minimum speed
- Maximum acceleration and braking
- Maximum steering rate
- Maximum steering angles

The control inputs found using this method is given to the car for a single time step. Then this entire process is repeated for the next time step with its new state values.

V. Results

A. Path Following using MPC

In fig 3 we simulate the path followed by the vehicle where the MPC controller generates inputs within the given constraints for path following. We see that it tries to maintain the desired velocity, with minimum positional error. Here the weights of the cost function play an important role in deciding whether the controller will try to maintain the path closely or the velocity closely (Fig 3 and Fig 4).

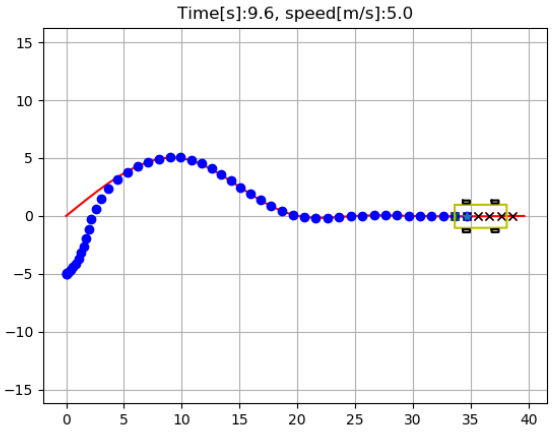

Fig 3 Simulation of Path Following using MPC

If there is a desired path that the vehicle model cannot satisfy the MPC tries to generate a sub-optimal path which can be followed (Fig 4).

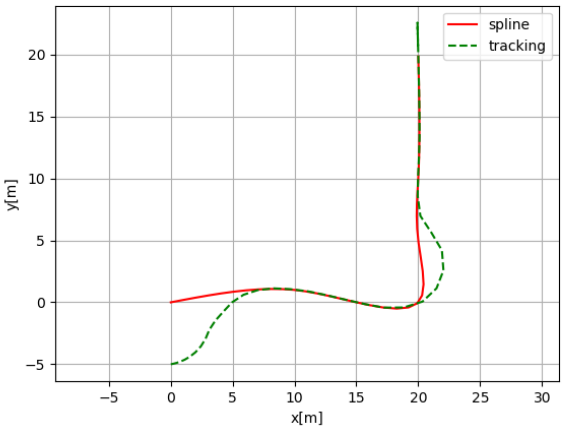

Fig 4 Simulation of path following where MPC follows a sub-optimal path with horizon 5

B. Dynamic Obstacle Avoidance in unconstrained environment

Fig 5 shows obstacle avoidance with multiple obstacles where there are no constraints on the path in the environment. If there is a high density of of obstacle around the desired path the control algorithm fails and Collison are observed.

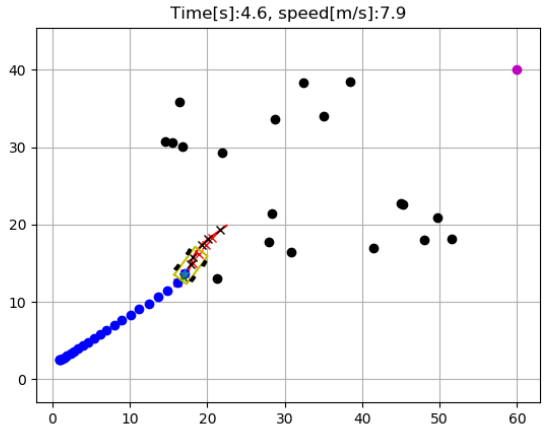

Fig 6. Simulation of Path following in an open environment with dynamic obstacles

C. Dynamic Obstacle Avoidance in a constrained environment.

Here we are trying to navigate towards the goal following a path provided by the A* algorithm. There is a moving obstacle in the path that we are trying to avoid using the potential field generated by the obstacle.

For simulation results in Fig 6,7,8,9 using low value of R_2 is used. In Fig 8 and 9 we can observe oscillations of steering angle and acceleration which will be experiences as jerks on the vehicle due to low value of R_2.

For Fig 10, 11 high values of R_2 where used we can see the effect of increasing the cost value R_2 where we obtain a smooth change in acceleration and steering angle.

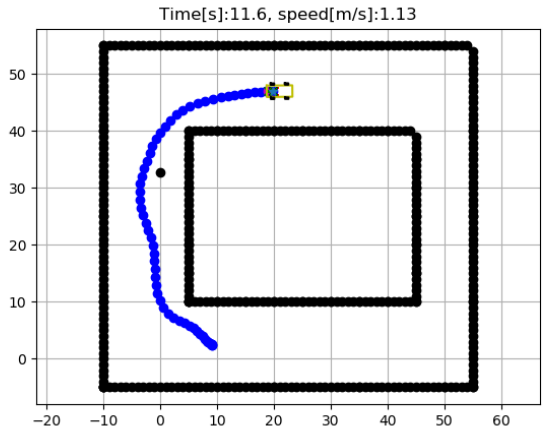

Fig 6. Simulation of Path following in a closed environment with dynamic obstacles

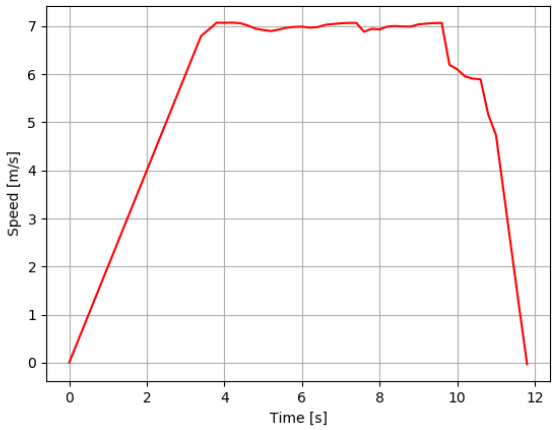

Fig 7. Velocity vs Time Graph

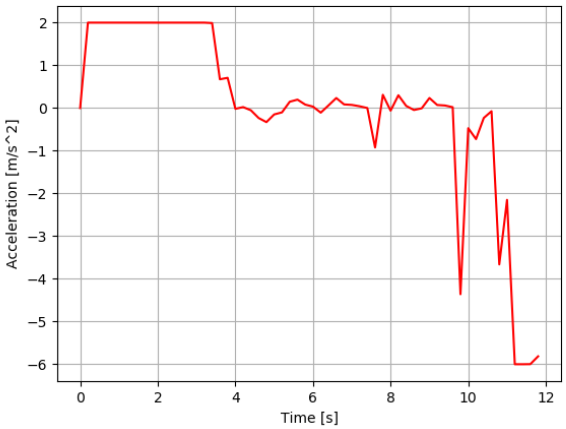

Fig 8. Acceleration vs Time Graph

D. Effect of length of Horizon

The length of horizon chosen an important role in finite horizon MPC. If we choose a value very low (*ex – 2)* we see the system becomes unstable and doesn't follow the desired path (fig 15). For our system we have chosen a value in between 5 to 7 for which the controller ensured the vehicle followed the intended path.

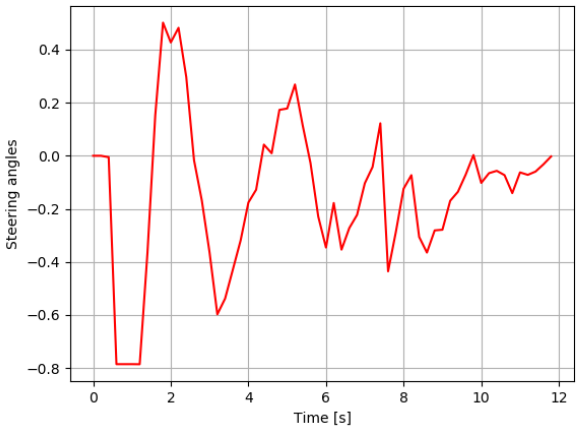

Fig 9. Steering Angle vs Time Graph

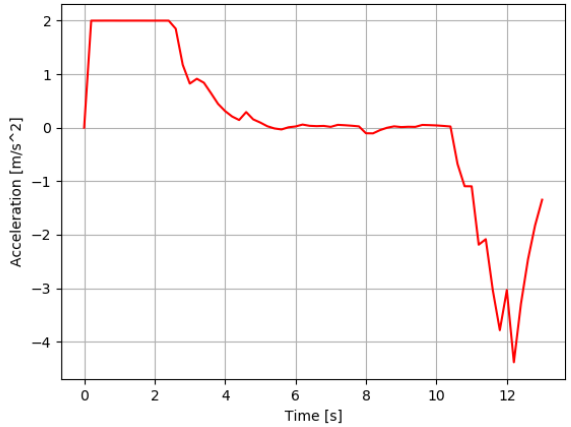

Fig 10. Acceleration vs Time Graph with modified R_2

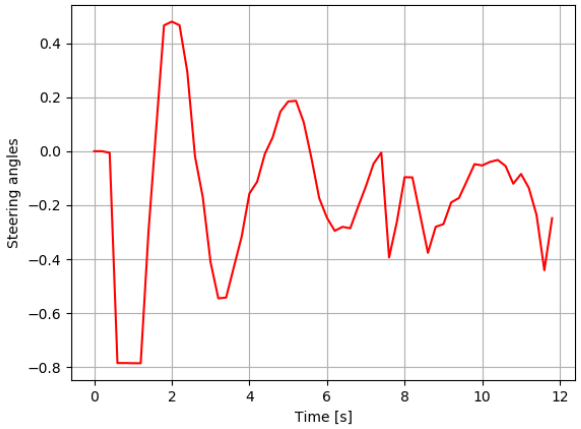

Fig 11. Steering angle vs Time Graph with modified R_2

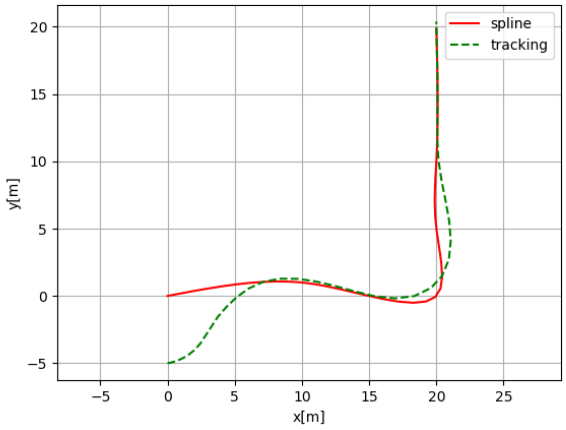

Fig 12 Simulation of path following where MPC follows a sub-optimal path with horizon 2

E. Stability

We constructed the Lyapunov function using eqn (7.2) and where able to find to be +ve and decreasing (fig 14) in most location. Whenever there is a change in the desired state or the controller is not able to achieve the required constraints and horizon the Lyapunov increases suggesting there are regions of instability and the solution is only suboptimal (Fig 16).

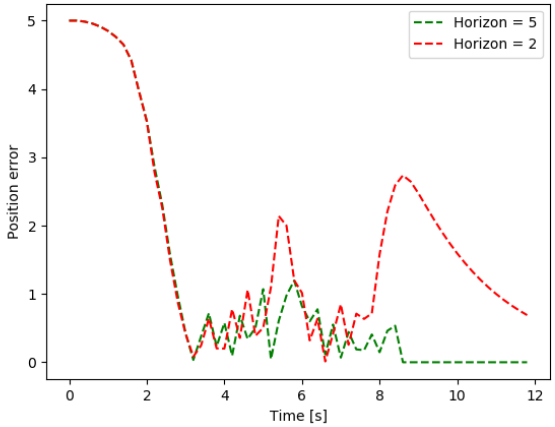

Fig 13. Error in Position with different Horizon Length

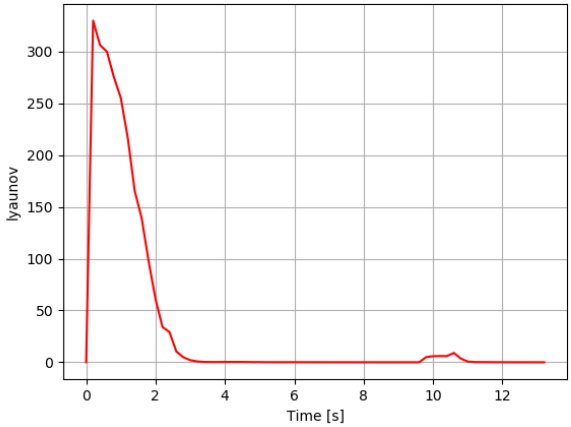

Fig 14 Lyapunov Function plot using the Horizon Cost as the Function.

F. Edge Cases:

1. High density of moving obstacle in a region along the shortest path – Whenever there is a high density of moving obstacles along the path, the controller failed to generate the required solution within the time frame and Collison was observed.
2. For obstacles with random accelerations the controller didn't perform well and was not able to avoid the obstacles.
3. The vehicle was not able to outmaneuver obstacles having high velocity and traveling towards the vehicle at an angle or from behind. The algorithm doesn't work well with obstacles coming from behind.

ACKNOWLEDGMENT

We will like to thank Vinutha Kallem, David Levine and Nanda Kishore Vasudevan and all the students for successful completion of the project.

REFERENCES

[1] J. Canon, K. Rose and W. Ruml, "Real Time Motion Planning with Dynamic Obstacle " Proceedings of the fifth Annual Symposium on Combinatorial Search

[2] M Likhachev, D Ferguson† , G Gordon† , A Stentz† , and S Thrun. "Anytime Dynamic A*: An Anytime, Replanning Algorithm"

[3] Muhammad Farhan Manzoor and Qinghe Wu, Control and Obstacle Avoidance of Wheeled Mobile Robot, 7th International Conference on Computational Intelligence, Communication Systems and Networks

[4] P. Polack; F Altche; B d'Andrea-Novel ; A de La Fortelle, "The Kinematic Bicycle Model: a Consistent Model for Planning Feasible Trajectories for Autonomous Vehicles". IEEE Intelligent Vehicles Symposium (IV).

[5] Pepy, R., Lambert, A., and Mounier, H. (2006). Path planning using a dynamic vehicle model. In 2006 2nd International Conference on Information Communication Technologies, volume 1, pages 781–786.

[6] Jasmin Velagic, Bakir Lacevic and Nedim Osmic, "Efficient Path Planning Algorithm for Mobile Robot Navigation with a Local Minima Problem Solving", IEEE, 2006.

[7] B. Pluymers∗, L. Roobrouck, J. Buijs, J.A.K. Suykens, B. De Moor, "Constrained linear MPC with time-varying terminal cost using convex combinations", Automatica 41 (2005) 831 – 837

[8] Kai Liu ID , Jianwei Gong *, Shuping Chen, Yu Zhang ID and Huiyan Chen, Article Model Predictive Stabilization Control of High-Speed Autonomous Ground Vehicles Considering the Effect of Road Topography, Applied Science Journey.