

Model Predictive Control for Autonomous Driving Based on Time Scaled Collision Cone

Mithun Babu¹, Yash Oza¹, Arun Kumar Singh², K. Madhava Krishna¹, Shanti Medasani³

Abstract—In this paper, we present a Model Predictive Control (MPC) framework based on *path velocity decomposition* paradigm for autonomous driving. The optimization underlying the MPC has a two layer structure wherein first, an appropriate path is computed for the vehicle followed by the computation of optimal forward velocity along it. The very nature of the proposed path velocity decomposition allows for seamless compatibility between the two layers of the optimization.

A key feature of the proposed work is that it offloads most of the responsibility of collision avoidance to velocity optimization layer for which computationally efficient formulations can be derived. In particular, we extend our previously developed concept of *time scaled collision cone* (TSCC) constraints and formulate the forward velocity optimization layer as a convex quadratic programming problem. We perform validation on autonomous driving scenarios wherein proposed MPC repeatedly solves both the optimization layers in receding horizon manner to compute lane change, overtaking and merging maneuvers among multiple dynamic obstacles.

I. INTRODUCTION

Model Predictive Control or MPC has emerged as a powerful tool for navigation of car like vehicles. Its strength stems from the fact that it can incorporate constraints on state and control while minimizing a user defined cost function. For car-like vehicles, the optimization underlying the MPC becomes strongly non-linear and non-convex due to the motion model of the vehicle and collision avoidance constraints. Current state of the art approaches rely on iterative techniques like sequential convex programming [1], [2] to solve the underlying optimization [3], [4]. However, without a good heuristic for initialization and under the strict real time constraints demanded by autonomous driving, these iterative techniques can sometime fail to provide any feasible solution. As a prospective remedy to this, most MPC frameworks rely on the use of so called "warm start" initializations for the underlying optimization. Herein solutions computed at previous instants are used as an initialization for solving the optimization problem at the current instant. The MPC frameworks proposed in [5], [6] follow an alternate methodology. They directly substitute non-linear motion model with approximate linear form, thus significantly reducing the complexity of the optimization. However, it should be noted that trajectories thus obtained are not guaranteed to be kino-dynamically feasible or collision free.

1. RRC, IIIT-Hyderabad, India 2. AUT, Tampere University of Technology, Finland 3. Mathworks Inc. Equal contribution from first two authors. Third author was supported by TUT Postdoc fellowship.

A. Overview of Proposed Approach and Contributions

The proposed approach has a similar flavor as that of the concept of *path velocity decomposition* first proposed in [7]. However, the technical approach is significantly different. Specifically, in contrast to [7], the proposed approach considers the reactive setting and uses the concept of collision cone [8] for modeling dynamic collision avoidance. Similar to [7], we observe that the collision avoidance maneuvers has two components namely the path taken by the robot and the forward velocity along it. In line with this observation, in this paper, we propose an MPC framework wherein the underlying optimization has a two layer structure for separate path and forward velocity optimization. It is essential that both the optimization layers are compatible with each other. That is, the paths obtained at the first layer should allow us to easily obtain collision free forward velocities along it. In the proposed work, this compatibility is naturally achieved given that both the optimizations solves the same collision cone constraints but with different set of variables. The path optimization solves the collision cone constraints with angular velocity as the variable. The forward velocity optimization solves the so called *time scaled collision cone* (TSCC) constraint [9], [10] which is a projection of collision cone constraints along a given path.

A key feature of the proposed work is that it reduces the reliance on using iterative techniques and solving general non-linear optimizations for computing a collision free trajectory. This is achieved by offloading most of the responsibility of exactly satisfying the collision avoidance constraints to the forward velocity optimization layer for which computationally efficient structures can be derived. To be precise, we extend the concept of TSCC beyond [9], [10] and formulate the forward velocity optimization as a convex quadratic programming problem. Both the optimizations layers are solved repeatedly in receding horizon manner to obtain lane change, overtaking and lane merging scenarios.

The rest of the paper is organized as follows. Section II presents a review of the related works. Section III briefly describes the main notations and symbols used in the formulation. Section IV introduces the path optimization. Section V reviews the time scaling concepts and formulates the forward velocity optimization. Section VI presents the reformulation of the forward velocity optimization into a convex quadratic programming problem. Section VII describes the method of encompassing tangents through which the collision avoidance between rectangular objects can be treated in the same way as that between disk shaped objects.

Section VIII presents the simulation results validating our proposed formulation.

II. RELATED WORK

In this section, we first review the existing approaches for motion planning in dynamic environments for non-holonomic robots in the backdrop of autonomous driving. Subsequently, we review the more niche approaches specifically designed for autonomous driving.

A. Motion Planning in Dynamic Environments.

Motion planning in dynamic environments has been extensively studied through the lens of velocity obstacle (VO) [11] which are analogous and independently developed along with collision cone (CC) [8]. The strength of (VO) or (CC) concept is that it models dynamic collision avoidance as a first order constraint in combined position-velocity space. As a result, imminent collisions till some future time instants can be modeled based on just current velocity and positions. This feature has been exploited in works like [12], [13], [14] to formulate a one step MPC framework. Both the concept of VO and CC were originally developed for holonomic robots moving along straight line trajectories. Their extension to non-holonomic robots is non-trivial and still an active area of research. The extension proposed in [13] is built on the idea that a non-holonomic robot would exhibit a finite tracking error while tracking a holonomic (piece-wise straight line) trajectory. Moreover, these tracking errors can be modeled as a function of the current state and the commanded holonomic velocities. In other words, this abstraction essentially signifies that for a given tracking error, it is possible to define a space of allowed holonomic velocities. Thus, [13] proceeds by first enlarging the radius of the robot by the tracking error that would be observed while simultaneously constraining the collision free velocity to lie within a specific set of allowed holonomic velocities. This last step of constraining the collision free velocity is the key as the set of allowed holonomic velocities are usually non-convex. A convex approximation can be obtained at the expense of unearthing a reduced space of collision free velocities.

An alternative extension has been proposed in [14] where the VO constraints for non-holonomic robots are solved by exhaustive control sampling. Various heuristics and off line pre-computations are exploited to make the process computationally efficient. It is clear the effectiveness of this approach depends on the number of control samples which in turn depends on the resolution of discretization. However, it is worth pointing out that the computational efficiency and the resolution of discretization are conflicting criteria in this methodology.

The above discussed formulations were primarily designed for general robotic environments with mobile robots. Consequently, their extension to autonomous driving setting poses some key challenges. For example, the technique of enlarging the size of the robot as proposed in [13] is not suitable for autonomous driving setting where lanes of the road

provide a restricted space for maneuvers. Moreover, such size increment was proposed for only disk shaped robots and their extension to rectangular car-like vehicles is not straightforward. In contrast, the control sampling approach of [14] is good prospective solution provided good heuristics are available for restriction of search space. A recent approach called control obstacles (CO) [15] was used for computing collision free trajectories in autonomous driving setting [16]. Herein, the robot's kinematics as well as constraints are linearized to allow for quick computation. However, as mentioned earlier, the trajectories computed through linearized models are not guaranteed to be collision free. The proposed formulation provides some key advantages over these cited works. one such advantage being that it can work with rectangular geometries of the vehicle and can satisfy the collision avoidance constraints for provided non-holonomic motion model.

B. Motion Planning for Autonomous Driving

Both sampling/graph search based planners and optimization based approaches are currently used for motion planning for autonomous driving. Either of these approaches can be used in a receding horizon manner to develop a MPC framework. Refer [17] for an extensive survey on different existing approaches. In the context of the proposed work, we will primarily focus on reviewing the optimization based approaches. Works like [18], [19], [20] uses optimization to refine a coarse collision free trajectory obtained from the sampling based planners. A similar approach is presented in [21] wherein, the vehicle's motion model is first approximated as a series of integrators and subsequently optimized trajectories are computed for a set of sampled points. These cited approaches do not explicitly include collision avoidance constraints directly within the optimization. A more holistic approach was presented in [3] wherein various kino-dynamic and collision avoidance constraints are directly included in the optimization framework. The proposed formulation shares the same holistic nature but differs from [3] in the technical approach followed. As mentioned earlier, we rely on exploiting structure of *path velocity decomposition* to limit our reliance on general non-linear optimization techniques. Moreover, unlike [3], we model collision avoidance between rectangular objects through the concept of encompassing tangents. This is less conservative than the multiple circle approximation used in [3].

III. SYMBOLS AND NOTATIONS

We use normal faced lower cased letters to represent scalars while bold faced small case letters would be used to represent vectors or vector valued functions. For example, $\mathbf{x}(t) = (x(t), y(t))$ and $\mathbf{x}_j(t) = (x_j(t), y_j(t))$ respectively represents the trajectory of the vehicle and the j^{th} obstacle in some future time horizon. With a slight abuse of notation, we will use $\mathbf{x}(t_i)$ and $\mathbf{x}_j(t_i)$ to represent the respective positions at time instant t_i . Both the vehicle and the obstacles are assumed to be disk shaped with radius d and d_j respectively. The symbols, $v(t_i)$, $\theta(t_i)$ would be used to represent the

forward velocity and heading angle of the vehicle at time instant t_i . Matrices would be denoted by bold faced higher case letters.

IV. PATH OPTIMIZATION

Consider the following optimization with variable as the angular velocity $\dot{\theta}(t_i)$ acting in a constant manner over the interval $[t_i \ t_{i+1}]$ and resulting in a trajectory $\mathbf{x}(t)$. The forward velocity $v(t_i)$ is kept constant over $[t_i \ t_{i+1}]$. Further, $\Delta t = t_{i+1} - t_i$.

$$\min_{\dot{\theta}(t_i)} J = w_1(\theta(t_{i+1}) - \theta_d)^2 + w_2\dot{\theta}(t_i)^2 \quad (1a)$$

$$(\mathbf{x}(t_{i+1}), \dot{\mathbf{x}}(t_{i+1})) = \mathbf{f}(\mathbf{x}(t_i), v(t_i), \theta(t_i)) \quad (1b)$$

$$\mathbf{c}_{bounds}(v(t_i), \theta(t_i)) \leq 0 \quad (1c)$$

$$\theta_d = \text{atan2}(y(t_i) - y_f, x(t_i) - x_f) \quad (2)$$

$$\mathbf{f}(\mathbf{x}(t_i), v(t_i), \theta(t_i)) = \begin{cases} x(t_{i+1}) = x(t_i) + \dot{x}(t_{i+1})\Delta t \\ y(t_{i+1}) = y(t_i) + \dot{y}(t_{i+1})\Delta t \\ \dot{x}(t_{i+1}) = v(t_i) \cos(\theta(t_i) + \dot{\theta}(t_i)\Delta t) \\ \dot{y}(t_{i+1}) = v(t_i) \sin(\theta(t_i) + \dot{\theta}(t_i)\Delta t) \end{cases} \quad (3)$$

$$\mathbf{c}_{bounds}(v(t_i), \theta(t_i)) \leq 0 : \begin{cases} -\kappa_{max}v(t_i) \leq \dot{\theta}(t_i) \leq \kappa_{max}v(t_i) \\ |\dot{\theta}(t_i)| \leq \dot{\theta}_{max} \end{cases} \quad (4)$$

$$\mathbf{c}_{avoid}(\cdot) \leq 0 : \frac{(\mathbf{r}_j^T \mathbf{v}_j)^2}{\|\mathbf{v}_j\|^2} - \|\mathbf{r}_j\|^2 + (d_j + d)^2 \leq 0, \forall j = 1, 2, \dots, n \quad (5)$$

$$\mathbf{r}_j = \begin{bmatrix} x(t_{i+1}) - x_j(t_{i+1}) \\ y(t_{i+1}) - y_j(t_{i+1}) \end{bmatrix}, \mathbf{v}_i = \begin{bmatrix} \dot{x}(t_{i+1}) - \dot{x}_i(t_{i+1}) \\ \dot{y}(t_{i+1}) - \dot{y}_i(t_{i+1}) \end{bmatrix}.$$

The first term in the cost function, (1a) is inspired from path following algorithms [22], [17] and aims to align the heading angle of the vehicle to that of the straight line connecting its current position to the goal (refer 2). It is easy to see that under such condition, a non-holonomic vehicle can easily navigate towards the goal. The second term of the cost function penalizes the use of high magnitude of angular velocities and induces smoothness in the resulting trajectory. The weights w_1, w_2 trades-off smoothness with goal reaching behavior. The equality constraints (1b) ensures that the position and velocity at time t_{i+1} conform to the motion model of the vehicle (refer 3). The inequalities (1c) models the bounds on curvature (κ_{max}) of the resulting trajectory as well as magnitude of the angular velocity. The inequalities (5) model the collision avoidance between the vehicle and the dynamic obstacles through the concept of collision cone. Note that we only include such dynamic obstacle in (5) which are seen as converging from the vehicle's perspective [8].

A. Simplification and Solution Process

The main challenge in solving optimization (1a)-(5) stems from the non-convex motion model and collision avoidance constraints. Our solution process involves first eliminating the equality constraints (1b) by explicitly substituting its right hand side in (5). We next linearize the resulting collision avoidance constraints to obtain the affine approximation $\mathbf{A}_{avoid}\dot{\theta}(t_{i+1}) \leq \mathbf{B}_{avoid}$. Consequently, we solve a simpler

variant of optimization (1a)-(5) wherein there are no equality constraints and the collision avoidance constraints have an affine form.

V. TIME SCALED COLLISION CONE (TSCC)

The optimization in the last section provides us with a trajectory $\mathbf{x}(t), t \in [t_i \ t_{i+1}]$ which may be very close to being collision free¹. It could also be completely collision free but in general there is no such guarantee since the optimization in the previous section only dealt with an affine approximation of the collision avoidance constraints. In this section, we apply the concept of TSCC over the trajectory $\mathbf{x}(t)$ and ensure exact satisfaction of collision avoidance constraints by appropriately modifying the forward velocity profile of the trajectory. To this end, we next present a preview of the time scaling and TSCC concept followed by the construction of the second layer optimization.

A. Time Scaling

Given a trajectory, $\mathbf{x}(t)$, a change in its time scale from t to τ does not alter the geometric path associated with the trajectory. In other words, let $\mathbf{x}(t_i)$ correspond to a position on the trajectory at time t_i . Then, a change in time scale from t to τ would mean that the same point on the trajectory would be reached but now at a different time instant τ_i . It is clear that this change in time of traversal is simply due to the fact the time scaling transformation changes the forward velocity along the path associated with $\mathbf{x}(t)$. The exact change in velocity and acceleration are characterized by the following set of equations.

$$\begin{aligned} \dot{\mathbf{x}}(\tau) &= \dot{\mathbf{x}}(t)\dot{s}(t) \\ \ddot{\mathbf{x}}(\tau) &= \ddot{\mathbf{x}}(t)(\dot{s}(t))^2 + \dot{\mathbf{x}}(t)\ddot{s}(t) \\ \frac{dt}{d\tau} &= \dot{s}(t) \end{aligned} \quad (6)$$

The term $\frac{dt}{d\tau}$ in (6) is called the scaling function and it decides the transformation between the time scales t and τ .

Now, assuming $\frac{dt}{d\tau}$ as a linear function in arbitrary time interval $[t_i \ t_{i+1}]$, we obtain the following relation [23].

$$\tau_{i+1} - \tau_i = \frac{2(t_{i+1} - t_i)}{\dot{s}(t_i) + \dot{s}(t_{i+1})} = \frac{2\Delta t}{\dot{s}(t_i) + \dot{s}(t_{i+1})} \quad (7)$$

Equation (7) describes how an arbitrary time interval $[t_i \ t_{i+1}]$ in the current time scale gets modified to an interval $(\tau_i \ \tau_{i+1})$ in the new time scale τ .

Using (7), we can derive the following expression for $\frac{d^2t}{d\tau^2}$ [23].

$$\frac{d^2t}{d\tau^2} \approx \frac{\dot{s}(t_{i+1})^2 - \dot{s}(t_i)^2}{2\Delta t} \quad (8)$$

Equation (8) suggests that $\frac{d^2t}{d\tau^2}$ is approximated as a constant in time interval $(t_i \ t_{i+1})$ (or alternately $(\tau_i \ \tau_{i+1})$), with its value dependent on the magnitude of the scaling function $\frac{dt}{d\tau}$ at the two end of the interval $[t_i \ t_{i+1}]$.

¹From closeness, we mean that the collision avoidance constraints are very close to the boundary of the feasible region i.e $\mathbf{c}_{avoid}(\cdot) \approx 0$

B. TSCC

Using (6) and the discussions presented in the previous section, the space of velocities that can be achieved at position $\mathbf{x}(t_{i+1})$ for the vehicle can be characterized in the following manner in terms of the scaling function.

$$\dot{\mathbf{x}}(\tau_{i+1}) = \dot{s}(t_{i+1})\dot{\mathbf{x}}(t_{i+1}) \quad (9)$$

Using (9), the following time scaled variant of the collision avoidance constraints (5) can be obtained

$$\frac{((\mathbf{r}_{ij}^s)^T \mathbf{v}_j^s)^2}{\|\mathbf{v}_j^s\|^2} - \|\mathbf{r}_{ij}^s\|^2 + (d + d_j)^2 \leq 0. \quad (10)$$

$$\mathbf{r}_{ij}^s = \begin{bmatrix} x(t_{i+1}) - x_j(t_{i+1}) \\ y(t_{i+1}) - y_j(t_{i+1}) \end{bmatrix}, \mathbf{v}_i = \begin{bmatrix} \dot{s}(t_{i+1})\dot{x}(t_{i+1}) - \dot{x}_j(t_{i+1}) \\ \dot{s}(t_{i+1})\dot{y}(t_{i+1}) - \dot{y}_j(t_{i+1}) \end{bmatrix}. \quad (11)$$

In (10), the entities $x(t_{i+1})$, $\dot{x}(t_{i+1})$, $y(t_{i+1})$, $\dot{y}(t_{i+1})$, $\dot{x}_i(t_{i+1})$ etc. are known since the vehicle trajectory, $\mathbf{x}(t)$ in old time scale is given. Inequalities (10) can be represented in a compact form as the following generic quadratic inequalities.

$$a_j \dot{s}(t_{i+1})^2 + b_j \dot{s}(t_{i+1}) + c_j \leq 0 \quad (12)$$

Where, a_j , b_j and c_j are functions of $x(t_{i+1})$, $\dot{x}(t_{i+1})$, $y(t_{i+1})$, $\dot{y}(t_{i+1})$... etc.

C. Velocity Optimization

We now formulate an optimization problem to compute the scaling function $\dot{s}(t)$, $t \in [t_i, t_{i+1}]$ and consequently a collision free forward velocity along the path associated with $\mathbf{x}(t)$

$$\arg \min_{\dot{s}(t)} J_2 = \sum_i (\dot{s}_i(t_{i+1}) - \dot{s}^{pref}). \quad (13a)$$

$$v_{min} \leq \dot{s}(t_{i+1})\sqrt{\dot{x}(t_{i+1})^2 + \dot{y}(t_{i+1})^2} \leq v_{max}. \quad (13b)$$

$$\frac{a_{min}}{\sqrt{2}} \leq \dot{s}(t_{i+1})^2 \ddot{x}(t_i) + \ddot{s}(t_i) \dot{x}(t_i) \leq \frac{a_{max}}{\sqrt{2}}. \quad (13c)$$

$$\frac{a_{min}}{\sqrt{2}} \leq \dot{s}(t_{i+1})^2 \ddot{y}(t_i) + \ddot{s}(t_i) \dot{y}(t_i) \leq \frac{a_{max}}{\sqrt{2}}. \quad (13d)$$

$$a_j \dot{s}(t_{i+1})^2 + b_j \dot{s}(t_{i+1}) + c_j \leq 0, \forall j = 1, 2, 3...n. \quad (13e)$$

$$s(t_i) = 1, \forall i = 1, 2, 3...n \quad (13f)$$

$$\dot{s}^{pref} = \frac{v^{pref}}{\sqrt{\dot{x}_i(t_{i+1})^2 + \dot{y}_i(t_{i+1})^2}} \quad (14)$$

The cost function in (13a) seeks to compute such scaling functions which minimizes the deviation of the vehicle from its preferred forward velocity, v^{pref} . This preference velocity calculated based on output of Behavioural layer and average lane speed. The chosen design of the cost function assumes a specially important role in the context of autonomous driving. For example, in overtaking scenarios, v^{pref} should necessarily be higher the dynamic obstacles it is overtaking. Inequalities (13b) enforces the constraints that the forward velocity resulting from time scaling transformation should respect the minimum and maximum bounds. In our formulation, v_{min} is strictly greater than zero to account for the fact that in autonomous driving scenarios, bringing the vehicle

to a halt may disrupt the traffic flow. Inequalities (13c)-(13d) enforces the acceleration bounds. For computational reasons, we have split the acceleration bounds into equivalent bounds for the separate x and y components. Inequalities (13e) is same as (12). Finally, the equality (13f) ensures that the velocity profile resulting from the time scaling transformation has continuity with the current velocity profile at initial time instant t_i .

VI. SIMPLIFICATION OF VELOCITY OPTIMIZATION

In this section, we show that velocity optimization (13a)-(13f) can be reformulated to a convex quadratic programming (QP) problem. We start with the convexification of TSCC constraints (10) or (12) and then subsequently show that its intersection space with velocity and acceleration bounds can be described in terms of linear inequalities. For the ease of exposition, we introduce the following change of variables [23].

$$\dot{s}(t_i)^2 = z_i(t_i), \dot{s}(t_{i+1})^2 = z_i(t_{i+1}). \quad (15)$$

With the help of (8), we can also obtain the following relation

$$\ddot{s} \approx \frac{z(t_{i+1}) - z(t_i)}{2\Delta t}. \quad (16)$$

A. Convexification of TSCC Constraints

In our past work [9], [10], we have shown the convexity of (12) depends on the sign of a_j , which in turn depends on the relative positions and velocities between the vehicle and the dynamic obstacles. There, we exploited the fact that for each dynamic obstacle (12) represents a single variable quadratic inequality and thus can be solved in closed form. Subsequently, we employed a sorting algorithm to compute the resultant solution space. However, we have observed that such process become computationally expensive especially if many non-convex instances of (12) are present. Thus, here, we present an alternate methodology wherein two new insights are provided. Firstly, we show that even when $a_j < 0$, inequalities (12) can be converted to a convex form provided we have $c_j \geq 0$. Secondly, we show that for the non-convex instance of (12), we can linearize it to obtain an approximation which is more conservative than the original constraints. In other words, satisfaction of linearized constraint guarantees satisfaction of the original constraints.

The following analysis is for one dynamic obstacle but can be easily extended to the general case of n obstacles

1) $a_j \geq 0$, $c_j \leq 0$: The solution space of (12) for this case is given by

$$\dot{s}(t_{i+1}) \in [\dot{s}_{min}^j, \dot{s}_{max}^j]. \quad (17)$$

$$\dot{s}_{min}^j = \min\left(\frac{-b_j + \sqrt{(b_j^2 - 4a_j c_j)}}{2a_j}, \frac{-b_j - \sqrt{(b_j^2 - 4a_j c_j)}}{2a_j}\right). \quad (18)$$

$$\dot{s}_{max}^j = \max\left(\frac{-b_j + \sqrt{(b_j^2 - 4a_j c_j)}}{2a_j}, \frac{-b_j - \sqrt{(b_j^2 - 4a_j c_j)}}{2a_j}\right). \quad (19)$$

Using (15), we have

$$z(t_{i+1}) \geq (\dot{s}_{min}^j)^2, \quad z(t_{i+1}) \leq (\dot{s}_{max}^j)^2. \quad (20)$$

2) $a_j \leq 0, c_j \geq 0$: In this section, we start with writing (12) in a slightly different form as below

$$a_j + b \frac{1}{\dot{s}(t_{i+1})} + c_j \frac{1}{\dot{s}(t_{i+1})^2} \leq 0 \quad (21)$$

The above inequality is convex with respect to $\frac{1}{\dot{s}(t_{i+1})}$ and whose solution space can be characterized in the following form.

$$\frac{1}{\dot{s}(t_{i+1})} \in [\dot{s}_{min}^j, \dot{s}_{max}^j] \Rightarrow \dot{s}(t_{i+1}) \in [\frac{1}{\dot{s}_{max}^j}, \frac{1}{\dot{s}_{min}^j}]. \quad (22)$$

$$\dot{s}_{min}^j = \min(\frac{-b_j + \sqrt{(b_j^2 - 4a_j c_j)}}{2c_j}, \frac{-b_j - \sqrt{(b_j^2 - 4a_j c_j)}}{2c_j}). \quad (23)$$

$$\dot{s}_{max}^j = \max(\frac{-b_j + \sqrt{(b_j^2 - 4a_j c_j)}}{2c_j}, \frac{-b_j - \sqrt{(b_j^2 - 4a_j c_j)}}{2c_j}). \quad (24)$$

Finally, using (15), we have

$$z(t_{i+1}) \geq \frac{1}{(\dot{s}_{max}^j)^2}, \quad z(t_{i+1}) \leq \frac{1}{(\dot{s}_{min}^j)^2}. \quad (25)$$

3) $a_j \geq 0, c_j \geq 0$: The previous two discussed cases are a subset of this case and any of the approaches can be followed.

4) $a_j \leq 0, c_j \leq 0, b_j \geq 0$: This represents a non-convex instance of (12) and thus is the most difficult to handle. We start with recalling (15) and writing the (12) in the following form

$$a_j z(t_{i+1}) + b_j \sqrt{z(t_{i+1})} + c_j \leq 0 \quad (26)$$

Inequality (26) is in the so called *convex-concave* form wherein the first term is convex while the second term is purely concave. We linearize the second term around some initial guess z^* resulting in the following affine approximation for (26).

$$a_j z(t_{i+1}) + b_j(\sqrt{z^*} + \frac{1}{2\sqrt{z^*}}(z(t_{i+1}) - z^*)) + c_j \leq 0 \quad (27)$$

An interesting point about (27) is that it is a more conservative constraint than (26) [1]. Thus, satisfaction of (27) guarantees satisfaction of (26).

B. Velocity and Acceleration Bounds

Squaring both sides of (13b), we obtain

$$(v^{min})^2 \leq \dot{s}(t_{i+1})^2(x(t_{i+1})^2 + y(t_{i+1})^2) \leq (v^{max})^2 \\ \Rightarrow (v^{min})^2 \leq z(t_{i+1})((x(t_{i+1})^2 + y(t_{i+1})^2)) \leq (v^{max})^2 \quad (28)$$

Using (15), (16), the acceleration bounds can be put in the following form.

$$\frac{a^{min}}{\sqrt{2}} \leq z(t_i)\ddot{x}(t_i) + \dot{x}(t_i)\frac{z(t_{i+1}) - z_i(t_i)}{2\Delta t} \leq \frac{a^{max}}{\sqrt{2}}. \quad (29)$$

$$\frac{a^{min}}{\sqrt{2}} \leq z(t_i)\ddot{y}(t_i) + \dot{y}(t_i)\frac{z(t_{i+1}) - z_i(t_i)}{2\Delta t} \leq \frac{a^{max}}{\sqrt{2}}. \quad (30)$$

Clearly, (29)-(30) are set of linear inequalities in terms of variable $z(t_i), z(t_{i+1})$.

Summary: The discussions presented in this section has shown that the TSCC constraints and the velocity and acceleration bounds can all be represented as linear inequalities in terms of variable $z(t_i), z(t_{i+1})$. Let these inequalities be compactly represented as $\mathbf{A}_{ineq}\mathbf{z} - \mathbf{b}_{ineq} \leq 0$, where $\mathbf{z} = (z(t_i), z(t_{i+1}))$. The optimization (13a)-(13f) can then be presented in the following simple form.

$$\arg \min_{z(t_i), z(t_{i+1})} J_2 = \sum_i (z_i(s_c) - (s_i^{pref})^2)^2. \quad (31a)$$

$$\mathbf{A}_{ineq}\mathbf{z} - \mathbf{b}_{ineq} \leq 0. \quad (31b)$$

$$z_i(t_{i+1}) \geq 0. \quad (31c)$$

$$z_i(t_i) = 1. \quad (31d)$$

As can be seen, (31a)-(31d) is a convex QP problem. The variables in the optimization are $z(t_i), z(t_{i+1})$. The scaling functions can be easily recovered through $\dot{s}(t_i) = \sqrt{z(t_i)}, \dot{s}(t_{i+1}) = \sqrt{z(t_{i+1})}$.

VII. METHOD OF ENCOMPASSING TANGENTS

The formulations presented in the last couple of sections have assumed a disk shape for the vehicle and dynamic obstacles. In this section, we show that using the method of encompassing tangents [8], the same formulation can be easily adapted for rectangular shaped objects as well. The following presentation is a slight adaptation of the original concept proposed in [8].

Consider Fig. 1(a) where P (vehicle) and Q (dynamic obstacle) represent two rectangular objects. A Minkowski sum boundary polygon R , can be mathematically represented as $R = Q \oplus (-P)$, Where $P \oplus Q = [p+q \mid p \in P, q \in Q]$. As shown in Fig. 1(b), the Minkowski sum allows us to reduce the rectangle P to a point and enlarge Q into polygon R . The minkowski sum R can be computed in $O(m+n)$ time for two polygons with m and n edges. Next, we compute the so called encompassing tangents (solid black line in Fig. 1(b)) which are straight lines drawn from P to polygon R with an enclosing angle φ given by

$$\varphi = \max_{i,j} (|\arctan(p_i) - \arctan(p_j)|)$$

where p_i is the slope of the lines drawn from P to i^{th} vertex of R . Following [8] and as shown in Fig. 1(b), the collision avoidance between rectangles P and Q can be equivalently modeled as that between a point and the circle C_{obst} constructed on the basis of encompassing tangents. Their velocities will be same as that of P and Q respectively. Note that the C_{obst} is much less conservative than the circumscribing circle of R . C_{obst} also has less area than that obtained by representing the polygon R with multiple circles [3].

VIII. SIMULATIONS RESULTS

A. MPC and Simulation Framework

The employed MPC framework is simple and consists of repeatedly solving the path and velocity optimization and executing the computed trajectory. Nevertheless, few key points are worth pointing out. Firstly, only a part of the current computed trajectory is executed before a new set of

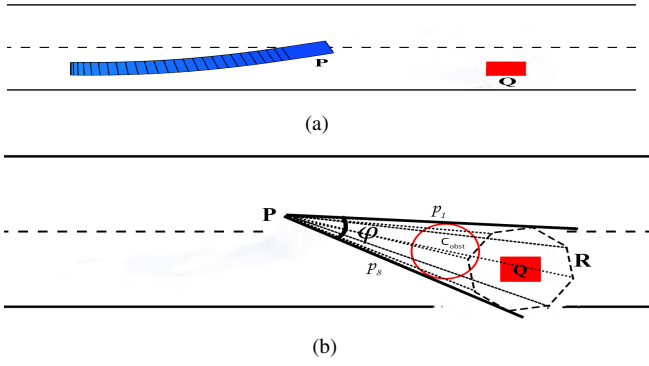


Fig. 1. (a) Collision modeling between rectangular objects. The blue represents the planned vehicle while the red represents the dynamic obstacles. (b) The planned vehicle is reduced to a point and the dynamic obstacle is enlarged through the concept of Minkowski sum. The solid black lines represent the so called encompassing tangents. Collision avoidance between rectangles can be modeled as that between a point and a circle (C_{obst}) constructed based on encompassing tangents.

computations are initiated. In other words, our planning horizon is much smaller (typically 10 times) than the planning horizon which is typical for an MPC framework. Secondly, in autonomous driving scenarios the vehicle is required to follow a path rather than reach a specific goal position. In our MPC implementation, we induce this behavior by giving a pseudo velocity to the goal position. The magnitude of this pseudo-velocity is tuned to get the best results. Finally, to constrain the vehicle to stay within the road boundaries, we created some fictitious static obstacles along the boundaries and included them in the path optimization framework as dynamic obstacles with zero velocities. Simulations were performed in Gazebo [24] while optimizations were solved in Python with CvxOpt [25] as the solver. Gazebo is a 3D simulator developed and supports multiple physics engines. The obstacle locations and their velocities were sensed by LIDAR mounted on top of car. In the Gazebo snapshots shown later, the gray color SUV represents the vehicle while rest of the cars are dynamic obstacles. A simulation can be found at <https://www.youtube.com/watch?v=8re0I9z0LEk&feature=youtu.be>

B. Benchmark Scenarios

1) *Overtaking*: Fig. 2(a)-2(c) shows the simulation of overtaking maneuvers in Gazebo. The dynamic obstacles are all moving at a speed of $5m/s$ along straight line paths. Fig. 2(d) shows the path of the concerned vehicle. As can be seen, the vehicle first departs from the lane and comes back to it after overtaking. The forward velocity plots shown in Fig. 2(e) reveal an interesting trend. The initial forward velocity is around $2m/s$ for the vehicle which is significantly less than the dynamic obstacles ahead. Thus, to initiate an overtaking maneuver, we set $v_{pref} = 15m/s$ in (13a). As can be seen from Fig. 2(e), the vehicle gradually increases its forward velocity and settles down at $15m/s$ after overtaking. The angular velocity plots are shown in Fig. 2(f) and exhibit a smooth profile.

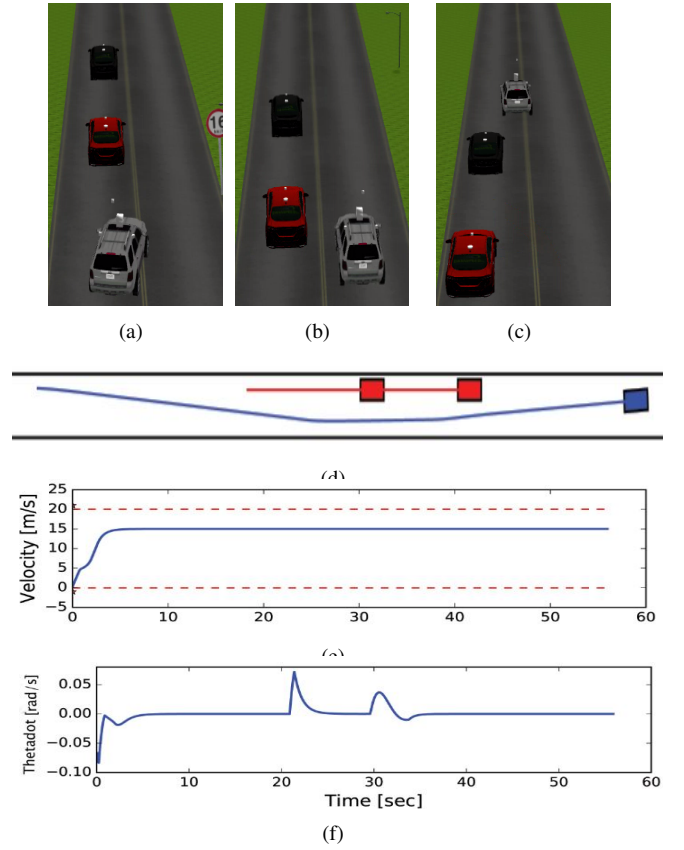


Fig. 2. (a)-(c): Gazebo snapshots of overtaking maneuver. (d): Path of the planned vehicle. (e)-(f): Forward and angular velocity plots.

2) *Overtaking and Following*: Fig. 3(a)-3(f) show the concerned vehicle change lane to overtake a slow moving dynamic obstacle and then slow down to follow the dynamic obstacle ahead. The behavior can also be cross-validated against the forward velocity plots shown in Fig. 3(g). The forward velocity increases to $v^{pref} = 8m/s$ during the initiation of the overtaking and then decreases to $4m/s$ after coming back to the lane. In between, the forward velocity jumps momentarily to $12m/s$ to quickly complete the overtaking maneuver. The angular velocity plots shown in Fig. 3(h) again exhibit a smooth profile.

3) *Merging and overtaking*: Fig. 4(a)-4(f) show the concerned vehicle first moving along a free lane and then converging to a lane with slow moving dynamic obstacles followed by an overtaking maneuver. As shown in Fig. 4(g), the forward velocity increases to the preferred magnitude of $8m/s$ and then slows down to $4m/s$ to converge and overtake. The angular velocity plots are shown in Fig. 2(f) and is similar in nature to that obtained for the previous cases.

4) *Overtaking and Lane Change*: Fig. 5(a)-5(f) shows the vehicle changing its lane by overtaking two dynamic obstacles ahead. Due to lack of space, we do not show the forward and angular velocity plots but they show similar smoothness as in the previous cases.

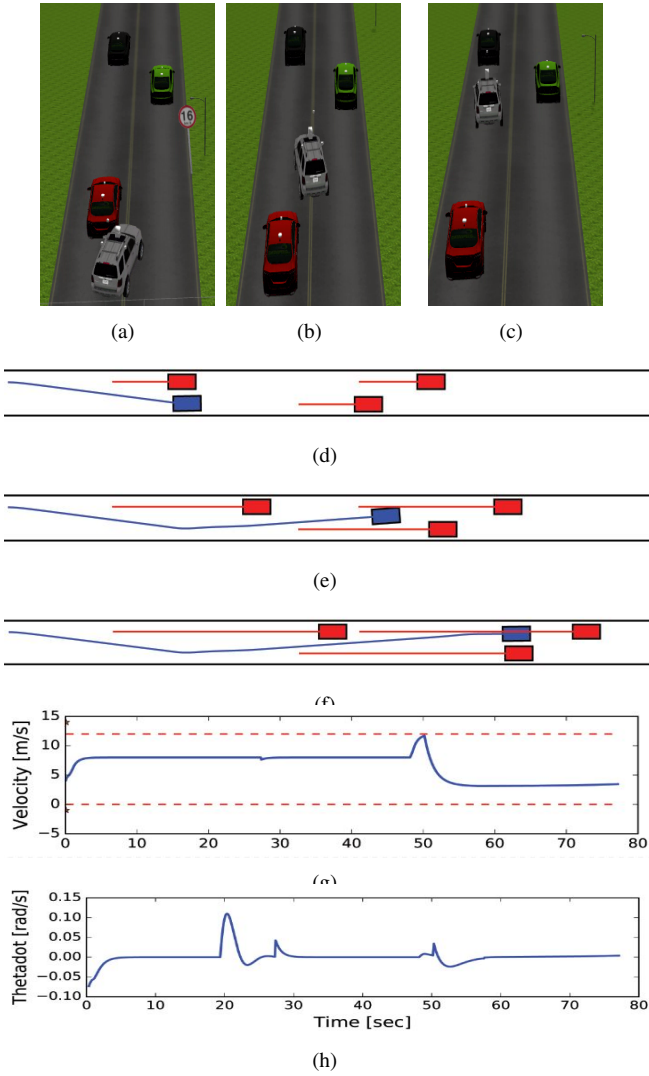


Fig. 3. (a)-(c): Gazebo snapshots of overtaking combined with following maneuver. (d)-(f): Path of the planned vehicle at three different time instants. (g)-(h): Plots of forward and angular velocity.

C. Computation Time

The mean computation time for each iteration of our MPC loop observed across all the examples presented in the previous section is shown in Fig. 6. The implementation was done in Python on a 64 bit laptop with 6GB RAM, i5 processor with 2.60 GHz clock speed. As can be seen, the proposed MPC with hierarchical path and velocity optimization can be solved at around 13ms for a scenario with 10 obstacles, resulting in an update rate of almost 77Hz.

IX. CONCLUSION AND FUTURE WORK

In this paper, we presented a computationally efficient MPC framework for autonomous driving. The underlying optimization is based on *path velocity decomposition* paradigm and can exactly satisfy the complex kino-dynamic and collision avoidance constraints. In contrast to existing works, the proposed MPC limits its reliance on solving general non-linear optimizations. Instead, the forward velocity optimization layer assumes most of the responsibility of guaranteeing

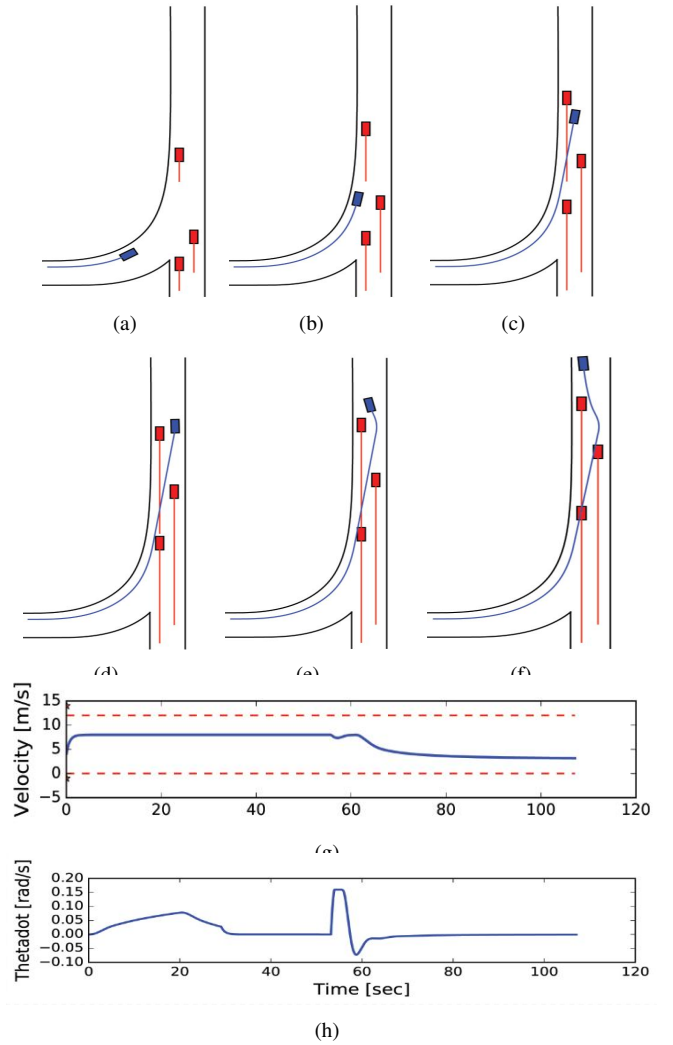


Fig. 4. (a)-(f): Snapshots and path of the planned vehicle for lane merging followed by overtaking maneuver. (g)-(h): Plots of forward and angular velocity.

collision avoidance. We have shown that the computation time of our MPC is very low and it in fact can run at an update rate of about 77 Hz in realistic scenarios.

Our future efforts are focused towards including uncertainty in our MPC framework. A probabilistic variant of TSOC had already been developed in [26] and we are looking to extend that formulation to autonomous driving setting.

REFERENCES

- [1] S. Boyd, "Sequential convex programming," *Lecture Notes, Stanford University*, 2008.
- [2] J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization," in *Robotics: science and systems*, vol. 9, no. 1, 2013, pp. 1–10.
- [3] A. Stentz, H. Herman, A. Kelly, E. Meyhofer, G. C. Haynes, D. Stager, B. Zajac, J. A. Bagnell, J. Brindza, C. Dellin *et al.*, "Chimp, the emu highly intelligent mobile platform," *Journal of Field Robotics*, vol. 32, no. 2, pp. 209–228, 2015.
- [4] A. Carvalho, Y. Gao, A. Gray, H. E. Tseng, and F. Borrelli, "Predictive control of an autonomous ground vehicle using an iterative linearization approach," in *Intelligent Transportation Systems (ITSC)*,

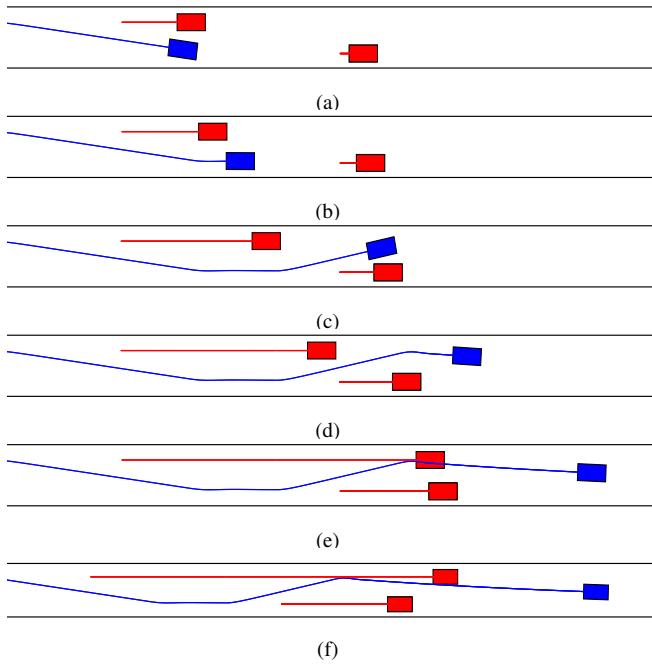


Fig. 5. Lane change by overtaking two dynamic obstacles

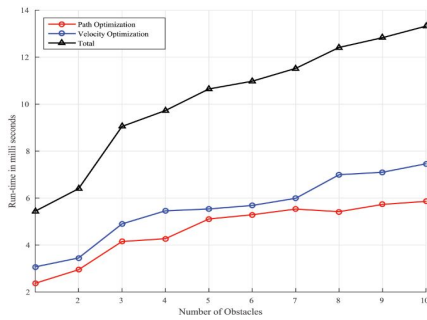


Fig. 6. Plot for individual path, forward velocity optimization and total computation time. The values represents the average across all the simulations presented in the previous section.

2013 16th International IEEE Conference on. IEEE, 2013, pp. 2335–2340.

- [5] J. Nilsson, P. Falcone, M. Ali, and J. Sjöberg, “Receding horizon maneuver generation for automated highway driving,” *Control Engineering Practice*, vol. 41, pp. 124–133, 2015.
- [6] J. Nilsson, M. Brännström, J. Fredriksson, and E. Coelingh, “Longitudinal and lateral control for automated yielding maneuvers,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 5, pp. 1404–1414, 2016.
- [7] K. Kant and S. W. Zucker, “Toward efficient trajectory planning: The path-velocity decomposition,” *The International Journal of Robotics Research*, vol. 5, no. 3, pp. 72–89, 1986.
- [8] A. Chakravarthy and D. Ghose, “Obstacle avoidance in a dynamic environment: A collision cone approach,” *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 28, no. 5, pp. 562–574, 1998.
- [9] A. K. Singh and K. M. Krishna, “Reactive collision avoidance for multiple robots by non linear time scaling,” in *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*. IEEE, 2013, pp. 952–958.
- [10] B. Gopalakrishnan, A. K. Singh, and K. M. Krishna, “Time scaled collision cone based trajectory optimization approach for reactive planning in dynamic environments,” in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 4169–4176.

- [11] P. Fiorini and Z. Shiller, “Motion planning in dynamic environments using velocity obstacles,” *The International Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998.
- [12] J. Van den Berg, M. Lin, and D. Manocha, “Reciprocal velocity obstacles for real-time multi-agent navigation,” in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. IEEE, 2008, pp. 1928–1935.
- [13] J. Alonso-Mora, A. Breitenmoser, M. Ruffli, P. Beardsley, and R. Siegwart, “Optimal reciprocal collision avoidance for multiple non-holonomic robots,” in *Distributed Autonomous Robotic Systems*. Springer, 2013, pp. 203–216.
- [14] D. Wilkie, J. Van Den Berg, and D. Manocha, “Generalized velocity obstacles,” in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. IEEE, 2009, pp. 5573–5578.
- [15] D. Bareiss and J. van den Berg, “Generalized reciprocal collision avoidance,” *The International Journal of Robotics Research*, vol. 34, no. 12, pp. 1501–1514, 2015.
- [16] A. Best, S. Narang, L. Pasqualin, D. Barber, and D. Manocha, “Autonovi: Autonomous vehicle planning with dynamic maneuvers and traffic constraints,” in *Intelligent Robots and Systems (IROS 2017), 2017 IEEE/RSJ International Conference on*. IEEE, 2017, p. to appear.
- [17] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, “A survey of motion planning and control techniques for self-driving urban vehicles,” *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, 2016.
- [18] M. McNaughton, C. Urmson, J. M. Dolan, and J.-W. Lee, “Motion planning for autonomous driving with a conformal spatiotemporal lattice,” in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 4889–4895.
- [19] W. Xu, J. Wei, J. M. Dolan, H. Zhao, and H. Zha, “A real-time motion planner with trajectory optimization for autonomous vehicles,” in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 2061–2067.
- [20] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, “Path planning for autonomous vehicles in unknown semi-structured environments,” *The International Journal of Robotics Research*, vol. 29, no. 5, pp. 485–501, 2010.
- [21] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, “Optimal trajectory generation for dynamic street scenarios in a frenet frame,” in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 987–993.
- [22] R. S. Wallace, A. Stentz, C. E. Thorpe, H. P. Moravec, W. Whittaker, and T. Kanade, “First results in robot road-following,” in *IJCAI*, 1985, pp. 1089–1095.
- [23] K. Hauser, “Fast interpolation and time-optimization with contact,” *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1231–1250, 2014.
- [24] N. Koenig and A. Howard, “Design and use paradigms for gazebo, an open-source multi-robot simulator,” in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3. IEEE, 2004, pp. 2149–2154.
- [25] M. S. Andersen, J. Dahl, and L. Vandenberghe, “Cvxopt: A python package for convex optimization, version 1.1. 6,” *Available at cvxopt.org*, vol. 54, 2013.
- [26] B. Gopalakrishnan, A. K. Singh, and K. M. Krishna, “Closed form characterization of collision free velocities and confidence bounds for non-holonomic robots in uncertain dynamic environments,” in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 4961–4968.