University of California, Los Angeles

MAE 271D Course Project

Optimization-based Collision Avoidance

Team 7

Ayush Agrawal
Mechanical Engineering
ayushagrawal26@ucla.edu

Debjyoti Chakrabarti
Aerospace Engineering
debjyoti@ucla.edu

Radha Lahoti
Mechanical Engineering
radhalahoti@ucla.edu

June 7, 2025

Contents

List of Figures

1 Introduction

Trajectory planning in obstacle-rich environments is a fundamental challenge for autonomous systems, including robots, self-driving cars, and aerial vehicles. A key difficulty lies in modeling collision avoidance constraints, which are typically nonconvex and nondifferentiable, making them hard to handle in real-time optimization frameworks.

Existing optimization-based methods often suffer from major limitations:

1. Use of linear approximations for collision constraints, resulting in uncertainty of error bounds.
2. Modeling of controlled objects as point masses, neglecting their true geometry (this conservative approach can lead to an infeasible solution).
3. Dependence on integer variables to model polyhedral obstacles, which limits its real-time applicability for nonlinear systems.

To address these issues, Zhang et al.[ZLB21] propose a novel method for reformulating collision avoidance constraints into smooth, differentiable forms using strong duality from Convex Optimization. This allows:

1. Exact and smooth reformulations applicable to convex obstacle shapes.
2. Support for both point-mass and full-dimensional controlled objects.
3. Signed distance formulation for minimum-penetration trajectories when collisions are unavoidable.
4. Compatibility with general-purpose nonlinear solvers (e.g., IPOPT).

The proposed method is demonstrated on an autonomous parking task in tight environments. Numerical results show that the approach enables robust, real-time trajectory planning while respecting dynamic constraints.

2 Objective

In this project, we have implemented both a one-shot optimal control (OCP) and Model Predictive Control (MPC) to obtain the optimal trajectories for reverse parking and parallel parking maneuvers while avoiding collisions with static and dynamic obstacles using the methods presented in [ZLB21]. The specific tasks we performed are:

- Implemented the smooth and exact reformulations of collision avoidance constraints using convex optimization duality.
- Implement **one-shot Optimal Control Problem (OCP)** and **Model Predictive Control (MPC)** with these constraints alongside dynamics, terminal constraints, and state and control input bounds.

- Demonstrated the **collision-free** and **minimum penetration** trajectories for reverse parking and parallel parking maneuver.
- Analysed the effect of the penetration penalty and prediction horizon on the optimal trajectory.
- Experimented with an enhanced objective function to minimize jerk.
- Extended the proposed method from [ZLB21] to tackle obstacle avoidance in case of **dynamic (translating and rotating) obstacles** moving along predefined paths.

3 Methodology

3.1 Optimization Problem

In [ZLB21], the constrained finite horizon optimal control problem for obstacle avoidance is given as,

$$\begin{aligned}
\min_{\mathbf{x},\mathbf{u}} \quad & J(x_k, u_k) && (1)\\
\text{s.t.} \quad & x_0 = x(0), \quad x_{N+1} = x_F, && (2)\\
& x_{k+1} = f(x_k, u_k), && (3)\\
& h(x_k, u_k) \leq 0, && (4)\\
& \mathbb{E}(x_k) \cap \mathbb{O}^{(m)} = \emptyset, && (5)\\
& \qquad \text{for } k = 0, \ldots, N, \quad m = 1, \ldots, M. && (6)
\end{aligned}$$

Here, (1) is the cost, (2) are the initial and terminal constraints respectively, (3) represents the dynamics constraints, (4) represents the bounds on the states and control inputs, N corresponds to the length of the time horizon. The collision avoidance constraint is given by (5). $\mathbf{x} := [x_0, x_1, \ldots, x_{N+1}]$ is the collection of all states, and $\mathbf{u} := [u_0, u_1, \ldots, u_N]$ is the collection of all inputs. The set $\mathbb{E}(x_k)$ representing the object is defined as:

$$\begin{aligned}
&\text{(Point-mass model)} \quad \mathbb{E}(x_k) = p_k && (7)\\
&\text{(Full-dimensional model)} \quad \mathbb{E}(x_k) = R(x_k)\mathbb{B} + t(x_k) && (8)
\end{aligned}$$

where p_k is the position of the point mass, $R(x_k)$ is a rotation matrix, $t(x_k)$ is a translation vector, and $\mathbb{B}$ is a fixed base shape (e.g., unit ball or polytope) centered at the origin.

The obstacle set $\mathbb{O}^{(m)}$ is defined as:

$$\mathbb{O}^{(m)} = \left\{ y \in \mathbb{R}^n : A^{(m)} y \preceq_{\mathcal{K}} b^{(m)} \right\} \tag{9}$$

where $A^{(m)} \in \mathbb{R}^{l \times n}$ and $b^{(m)} \in \mathbb{R}^l$ define the constraint representation of the obstacle, and $\preceq_{\mathcal{K}}$ denotes a generalized inequality induced by a convex cone $\mathcal{K}$.
A key challenge in solving problem 6, even for linear systems with a convex objective function

and convex state/input constraints, is the presence of the collision-avoidance constraints 5, which in general are non-convex and non-differentiable . In [ZLB21], two novel approaches are proposed for modeling the collision avoidance constraints that preserve continuity and differentiability. The collision avoidance problem for a full-dimensional controlled object is shown below, following these two approaches.

3.1.1 Collision-free Trajectory Problem

The collision-free trajectory generation problem for a full-dimensional object model is given as,

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{u},\boldsymbol{\lambda},\boldsymbol{\mu}} \quad & J(x_k, u_k) && (10)\\
\text{s.t.} \quad & x_0 = x(0), \quad x_{N+1} = x_F, && (11)\\
& x_{k+1} = f(x_k, u_k), && (12)\\
& h(x_k, u_k) \leq 0, && (13)\\
& -g^\top \mu_k^{(m)} + \left(A^{(m)} t(x_k) - b^{(m)}\right)^\top \lambda_k^{(m)} > d_{\min}, && (14)\\
& G^\top \mu_k^{(m)} + R(x_k)^\top A^{(m)\top} \lambda_k^{(m)} = 0, && (15)\\
& \left\| A^{(m)\top} \lambda_k^{(m)} \right\|_* \leq 1, && (16)\\
& \lambda_k^{(m)} \succeq_{\mathcal{K}^*} 0, \quad \mu_k^{(m)} \succeq_{\bar{\mathcal{K}}^*} 0, && (17)\\
& \text{for } k = 0, \ldots, N, \quad m = 1, \ldots, M.
\end{aligned}
$$

The equations (14), (15), (16), (17) correspond to the obstacle avoidance constraints where $\lambda_k^{(m)}, \mu_k^{(m)}$ are the dual variables associated with the collision avoidance constraints and M is the number of obstacles in the path. Note that in the case of the point mass approximation for modeling the object (car), **all** $\mu_k^{(m)}$ **would be** 0.

3.1.2 Minimum Penetration Trajectory Problem

If the collision-free trajectory generation problem is infeasible, it is reformulated as a "least intrusive" trajectory generation problem by softening the constraints. This is achieved by introducing slack variables in the constraints and also adding a regularization term in the cost function (bounding the magnitude of slack variables). This formulation is termed as the "minimum penetration trajectory" problem. The problem formulation for a full-dimensional controlled object is given by,

$$\min_{\mathbf{x},\mathbf{u},\mathbf{s},\boldsymbol{\lambda},\boldsymbol{\mu}} \quad J(x_k, u_k) + \kappa \sum_{k=0}^{N} \sum_{m=1}^{M} s_k^{(m)} \tag{18}$$

$$\text{s.t.} \quad x_0 = x_S, \quad x_{N+1} = x_F, \tag{19}$$

$$x_{k+1} = f(x_k, u_k), \tag{20}$$

$$h(x_k, u_k) \leq 0, \tag{21}$$

$$-g^\top \mu_k^{(m)} + \left(A^{(m)} t(x_k) - b^{(m)}\right)^\top \lambda_k^{(m)} > -s_k^{(m)}, \tag{22}$$

$$G^\top \mu_k^{(m)} + R(x_k)^\top A^{(m)\top} \lambda_k^{(m)} = 0, \tag{23}$$

$$\left\| A^{(m)} \lambda_k^{(m)} \right\|_* = 1, \tag{24}$$

$$s_k^{(m)} \geq 0, \quad \lambda_k^{(m)} \succeq_{\mathcal{K}} 0, \quad \mu_k^{(m)} \succeq_{\mathcal{K}^*} 0, \tag{25}$$

$$\text{for } k = 0, \ldots, N, \quad m = 1, \ldots, M.$$

where $s_k^{(m)} \in \mathbb{R}_+$ is the slack variable associated with obstacle $\mathbb{O}^{(m)}$ at time step k, and $\kappa \geq 0$ is a weight factor that forces the slack variable to be as small as possible.

3.2 Dynamics

The car is described by the classical kinematic bicycle model (1), which is well-suited for velocities used in typical parking scenarios. The state vector $x = [X, Y, \theta, v]^T$. The states (X, Y) correspond to the center of the rear axis, while θ is the yaw angle with respect to the x-axis and v is the velocity with respect to the rear axis. The control inputs $u = [a, \delta]^T$ consist of the steering angle δ and the acceleration a.

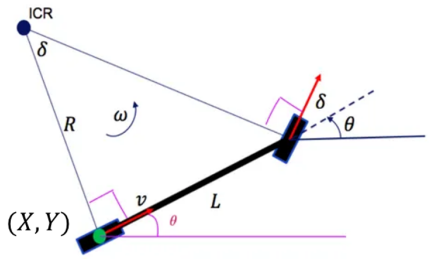


Figure 1: Bicycle Model

The continuous time system dynamics is given by,

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\theta} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v\cos(\theta) \\ v\sin(\theta) \\ \frac{v\tan(\delta)}{L} \\ a \end{bmatrix} \tag{26}$$

where $L = 2\,\text{m}$ is the wheel base of the car. Discretization is done via forward Euler integration as follows,

$$\begin{bmatrix} X_{k+1} \\ Y_{k+1} \\ \theta_{k+1} \\ v_{k+1} \end{bmatrix} = f(x_k, u_k) = \begin{bmatrix} X_k + v_k \cos(\theta_k)\Delta t \\ Y_k + v_k \sin(\theta_k)\Delta t \\ \theta_k + \frac{v_k \tan(\delta_k)}{L}\Delta t \\ v_k + a_k \Delta t \end{bmatrix} \tag{27}$$

Vehicle and Obstacle Representation: The vehicle is represented as a rotated and translated rectangle using a convex set transformation:

$$E(x_k) = R(x_k)B + t(x_k),$$

where B is the base shape of the vehicle. For this project, the set is defined as a rectangle of dimension $L \times \frac{L}{2}$.
Obstacles are modeled as rectangles.

- **Reverse parking:**
  Two closed-packed rectangles of dimension 8 m × 3 m are considered. They are centered at locations $(4, -4)$ m and $(14, -4)$ m respectively.
- **Parallel parking:** Two closed-packed rectangles of dimension 6 m × 3 m, along with another rectangle of dimension 3 m × 1.5m m are considered. They are centered at locations $(3, -4)$ m, $(7.5, -4.75)$ m and $(12, -4)$ m respectively.

3.3 Obstacle Avoidance Constraints

The obstacle avoidance constraints (static obstacle case) for collision-free trajectory and minimum penetration trajectory are explained below.

3.3.1 Collision-free Trajectory Problem

Table 1 summarizes the role of each constraint.

1. **Constraint** (14)**: Separation condition**

$$-g^\top \mu_k^{(m)} + \left(A^{(m)} t(x_k) - b^{(m)}\right)^\top \lambda_k^{(m)} > d_{\min}$$

Purpose: Ensures a minimum distance $d_{\min}$ between the object and the m-th obstacle at time step k.

Here $A^{(m)}x \leq b^{(m)}$: halfspace representation of the obstacle; $t(x_k)$: position of the object at time k; $\lambda_k^{(m)}$, $\mu_k^{(m)}$: dual variables for the obstacle and object, respectively; g: object geometry vector, which in our case is $g = [L/2, L/4, L/2, L/4]^T$.

This constraint ensures that a valid dual certificate of separation exists, meaning the robot is outside the obstacle by at least $d_{\min}$. Note that in the case of the point mass approximation for modeling the car, **all** $\mu_k^{(m)}$ **would be** 0.

2. **Constraint** (15)**: Dual Equilibrium**

$$G^\top \mu_k^{(m)} + R(x_k)^\top A^{(m)\top} \lambda_k^{(m)} = 0$$

Purpose: Enforces alignment between the robot's shape normals and obstacle contact normals via dual variables.

Here $R(x_k)$: rotation matrix of the object body, which in 2D is,

$$R(x_k) = \begin{bmatrix} \cos\theta_k & -\sin\theta_k \\ \sin\theta_k & \cos\theta_k \end{bmatrix};$$

G: face normal matrix of the object geometry, which in our case is,

$$G = \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

This constraint ensures force balance in dual space: the object's contact configuration must align with that of the obstacle.

3. **Constraint (**(16)**: Dual Norm Bound**

$$\left\| A^{(m)\top} \lambda_k^{(m)} \right\|_* \leq 1$$

Purpose: Constrains the dual variable magnitude to remain within the unit ball of the dual norm $\|\cdot\|_*$, which depends on the obstacle's shape. This prevents unbounded dual values and ensures dual feasibility relative to the obstacle geometry.

4. **Constraint** (17)**: Dual Nonnegativity**

$$\lambda_k^{(m)} \succeq_* 0, \quad \mu_k^{(m)} \succeq_{\bar{*}} 0$$

Purpose: Enforces non-negativity of dual variables, as required in the dual formulation of inequality constraints.

In our case, $\lambda_k^{(m)} \geq 0$: for obstacle halfspaces; $\mu_k^{(m)} \geq 0$: for object geometry support functions.

These conditions guarantee that only active constraints contribute to the dual optimization.

Constraint	Role	Interpretation
(14)	Separation	Enforces robot is at least $d_{\min}$ away from obstacle
(15)	Dual equilibrium	Aligns object and obstacle normals
(16)	Norm bound	Bounded and normalized dual variables
(17)	Dual feasibility	Ensures duals are non-negative

Table 1: Role of the obstacle avoidance constraints

3.3.2 Minimum Penetration Trajectory Problem

When collision-free motion is not feasible due to tight environments or conflicting constraints, we allow for controlled violation of the separation condition using slack variables. This results in a minimum penetration trajectory that tries to remain feasible with the least amount of overlap with obstacles.

1. **Constraint (22): Slack-enhanced Separation Condition**

$$-g^\top \mu_k^{(m)} + \left(A^{(m)} t(x_k) - b^{(m)}\right)^\top \lambda_k^{(m)} > -s_k^{(m)}$$

   Purpose: Allows limited penetration between the object and obstacle, controlled by a slack variable $s_k^{(m)} \geq 0$. When $s_k^{(m)} = 0$, the condition reduces to strict separation. A positive slack indicates how far the robot has penetrated into the obstacle.

2. **Cost Reformulation (18): Objective Penalization**

$$J(x_k, u_k) + \kappa \sum_{k=0}^{N} \sum_{m=1}^{M} s_k^{(m)}$$

   Purpose: Penalizes the total amount of constraint violation (penetration) via the scalar weight $\kappa \geq 0$, which controls the trade-off between optimality and safety. A large κ heavily discourages any slack, effectively recovering the collision-free case.

3. **Constraint (25): Slack Constraints**

$$s_k^{(m)} \geq 0$$

   Purpose: Enforces non-negativity of slack variables. Negative slack would represent "extra clearance," which is not meaningful in this formulation.

The slack variable $s_k^{(m)} \in \mathbb{R}_+$ quantifies the amount of allowable penetration into obstacle $\mathbb{O}^{(m)}$ at time step k. The weight $\kappa \geq 0$ in the cost function ensures that any violation of the separation condition is minimized as much as possible.

3.4 Cost

Our control objective is to navigate the vehicle to the goal position in a **reverse parking maneuver** while avoiding excessive control inputs. We combine these competing goals into our cost function as a weighted sum of the form

$$J(x_k, u_k) = (x_N - x_{goal})^T Q_f (x_N - x_{goal}) + \sum_{k=0}^{N-1} \Big[(x_k - x_{goal})^T Q (x_k - x_{goal}) + u_k^T R u_k \Big] \quad (28)$$

where, $Q_f = Q_f^T \succeq 0$, $Q = Q^T \succeq 0$, and $R = R^T \succeq 0$ are weighting factors.
In our work, the choice of the weight matrices are as follows

- **Reverse parking:**
  $Q = \text{diag}\{[22, 22, 10, 20]\}$, $Q_f = \text{diag}\{[40, 40, 40, 20]\}$ and $R = \text{diag}\{[0.1, 0.1]\}$
- **Parallel parking:**
  $Q = \text{diag}\{[800, 800, 800, 400]\}$, $Q_f = \text{diag}\{[1600, 1600, 400, 400]\}$ and $R = \text{diag}\{[1, 0.1]\}$

Later on in our study, we also try to generate trajectories with minimum jerk in the control inputs. To achieve this we panalize the rate of change in control inputs, i.e., Δu_k. Formally, we do by redefining the cost function as follows:

$$J(x_k, u_k) = (x_N - x_{goal})^T Q_f (x_N - x_{goal}) + \sum_{k=0}^{N-1} \Big[(x_k - x_{goal})^T Q (x_k - x_{goal}) + u_k^T R u_k + \Delta u_k^T S \Delta u_k \Big] \quad (29)$$

For both parallel and reverse parking, we choose $S = \text{diag}\{[0.1, 0.1]\}$.

3.5 Initial and Terminal Constraints and Bounds

Reverse Parking
The initial state vector of the object is a zero vector. The vehicle needs to reach the target coordinate of $(9, -4)$ m, with orientation, $\theta = \pi/2$ and velocity, $v = 0$ m/s.

Parallel Parking
The initial state vector of the object is a zero vector. The vehicle needs to reach the target coordinate of $(7.5, -3.25)$ m, with orientation, $\theta = 0$ and velocity, $v = 0$ m/s.

The velocity of the vehicle is bounded by, $v \in [-1, 2]\ m/s$. The control input constraints are given by, $\delta \in [-1, 1]\ rad$ and $a \in [-1, 1]\ m/s$. Later, for minimum jerk trajectory, we also set bounds on rate of change of steering angle, $\dot{\delta} \in [-1, 1] \frac{rad}{s}$. No bounds are set on jerk, $\dot{a}$, due to absence of this information in [ZLB21].
We impose the terminal constraint in the one-shot OCP problem but remove it in the MPC, since imposing it makes it difficult to find a feasible solution for the MPC. The target is achieved in the MPC case by using the terminal cost, which is why we set the Q_f slightly higher than Q.

3.6 Problem Setup

The collision avoidance problem is solved using both a one-shot Optimal Control problem (OCP) and a Model Predictive Control (MPC) setup. For the reverse parking problem, the one-shot OCP was solved with a time horizon of $T = 12$ seconds and a discretization step size of $dt = 0.1$ seconds. In case of MPC formulation, the prediction horizon of 8 sec is chosen, with the total simulation horizon of 16 sec. Note that shorter time horizons worked for the cases with no dynamic obstacle, but these values were chosen for most cases (those where these were not used have been mentioned in the results) for uniformity across simulations. The optimal control problem for the one-step Optimal Control and in each iteration of MPC is solved using IPOPT via the CasaDi framework in Python.

3.7 Choice of Initial Guess

Since the given trajectory design problem is nonlinear, a proper initial guess needs to be provided as *warm starting*. For single-shot OCP, a linear interpolation of the state trajectory (between initial and final states) and a zero control input trajectory are provided to the solver. For the MPC formulation, the initial guess of state and control inputs is chosen as zero vectors at the first iteration. For subsequent iterations of MPC, the initial guesses are provided as the solutions of the state and control inputs from the previous iteration.

3.8 Dynamic Obstacle

We extended the collision avoidance formulation in [ZLB21] to the dynamic obstacle case, where the obstacle is a rectangular polytope that can translate or rotate.

3.8.1 Obstacle Representation

A polytope can be represented as $Ax \leq b$. If the polytope is moving with time, then we will have time varying matrices A and b. If a rectangular obstacle $\{A_0 x \leq b_0\}$, rotates by a rotation matrix R_k and translates by a vector t_k, then the A_k and b_k matrices in its new configuration $\{A_k x \leq b_k\}$ are given by,

$$A_k = A_0 R_k^T, \quad b_k = b_0 + A_0 R_k^T t_k.$$

3.8.2 Obstacle Avoidance Constraints

The dynamic obstacle avoidance constraints are reformulated for both collision-free trajectory formulation as well as minimum penetration trajectory generation.

- **Collision-free Case:**

The collision avoidance constraints, in the dual form, can be rewritten as:

$$
\begin{aligned}
&- g^\top \mu_k^{(m)} + \left(A_k^{(m)} t(x_k) - b_k^{(m)}\right)^\top \lambda_k^{(m)} > d_{\min}, && (30)\\
&G^\top \mu_k^{(m)} + R(x_k)^\top A_k^{(m)\top} \lambda_k^{(m)} = 0, && (31)\\
&\left\| A_k^{(m)\top} \lambda_k^{(m)} \right\|_* \leq 1, && (32)\\
&\text{for } k = 0, \ldots, N, \quad m = 1, \ldots, M.
\end{aligned}
$$

where, $A_k^{(m)\top}$ and $b_k^{(m)}$ represents the time-varying obstacle location at the k-th instant. The other constraints remain the same as the static obstacle case.

- **Minimum Penetration Case:**
  For this case, the collision avoidance constraints are given by:

$$
\begin{aligned}
&- g^\top \mu_k^{(m)} + \left(A_k^{(m)} t(x_k) - b_k^{(m)}\right)^\top \lambda_k^{(m)} > -s_k^{(m)}, && (33)\\
&G^\top \mu_k^{(m)} + R(x_k)^\top A_k^{(m)\top} \lambda_k^{(m)} = 0, && (34)\\
&\left\| A_k^{(m)} \lambda_k^{(m)} \right\|_* = 1, && (35)\\
&\text{for } k = 0, \ldots, N, \quad m = 1, \ldots, M.
\end{aligned}
$$

4 Assumptions

While our implementation is based on the work by Zhang et al. [ZLB21], we make the following simplifying assumptions to tailor the methodology to our project constraints:

- **Euclidean norm and standard inequalities:** We use the Euclidean norm $\| \cdot \|_2$ instead of general cone norms, and replace conic inequalities with element-wise inequalities since we are using polyhedral shapes for object and obstacle.
- **No steering rate constraints:** We do not include constraints on the rate of change of the steering input.
- **Fixed discretization step:** A constant time step is used for discretizing the system dynamics, rather than optimizing the step size as suggested in the paper.
- **Minimum-time objective omitted:** The cost function does not include a term for minimizing the total maneuver time.

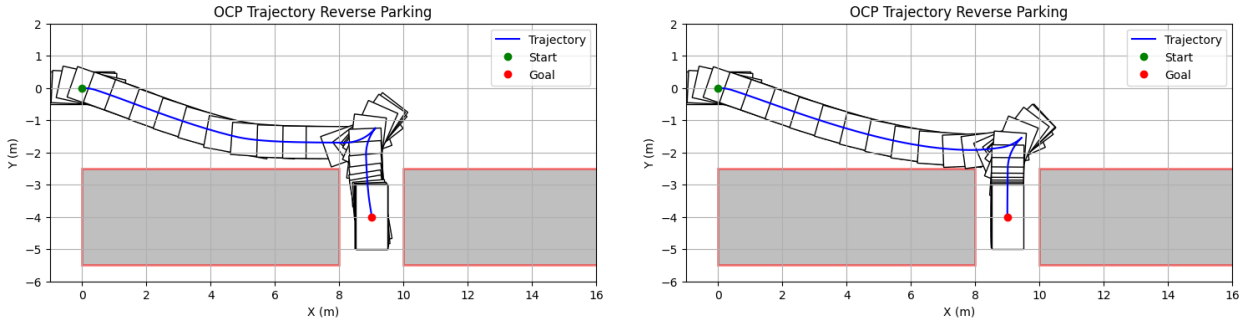
5 Results

5.1 OCP for Collision-free Trajectory Generation

The single-shot optimal control problem (OCP) is solved to generate a collision-free trajectory and minimum-penetration trajectory for both Reverse and Parallel Parking.

5.1.1 Reverse Parking

Figure2 shows the trajectories obtained using the collision-free formulation and minimum penetration formulation for reverse parking. In both cases, the vehicle successfully navigates to the goal. It can be clearly seen that in the case of minimum penetration, the vehicle brushes past the obstacle. For the collision-free case we set the minimum allowable distance, $d_{\min}$, to be 0.3 m. For the minimum penetration case, we set the value of the penalty on penetration, κ, to be $50,000$.



(a) Collision-free trajectory with safety margin $d_{min} = 0.3$ m

(b) Minimum penetration trajectory with $\kappa = 50,000$

Figure 2: Single-shot OCP for reverse parking ($N = 120$)

5.1.2 Parallel Parking

Figure 3 shows the trajectory obtained using the collision-free formulation for the parallel parking maneuver. Similar to reverse parking, the vehicle successfully parks itself in the space provided by reaching the assigned terminal point. However, we had to choose a smaller safety margin of $d_{min} = 0.1m$. This is important because, unlike reverse parking, the vehicle cannot directly move in the lateral direction to reach the terminal point. It needs space to perform short forward and backward motions to reach the terminal point. Hence, to give more space to the vehicle, we reduce the safety margin to $d_{min} = 0.1m$. For higher values of d_{min}, the solver may not converge. For the minimum penetration case, we set $\kappa = 60,000$ and we observe the same behavior as reverse case. The vehicle brushes past the obstacle to enter the parking space, then penetrates the obstacle on the right and finally converges to the terminal point exactly. This behavior can be observed in 7.

5.2 MPC Formulation

Next, we showcase the results of implementing MPC for the reverse and parallel parking scenarios. We implemented both collision-free and minimum penetration formulations for both the cases and investigated how they compare.

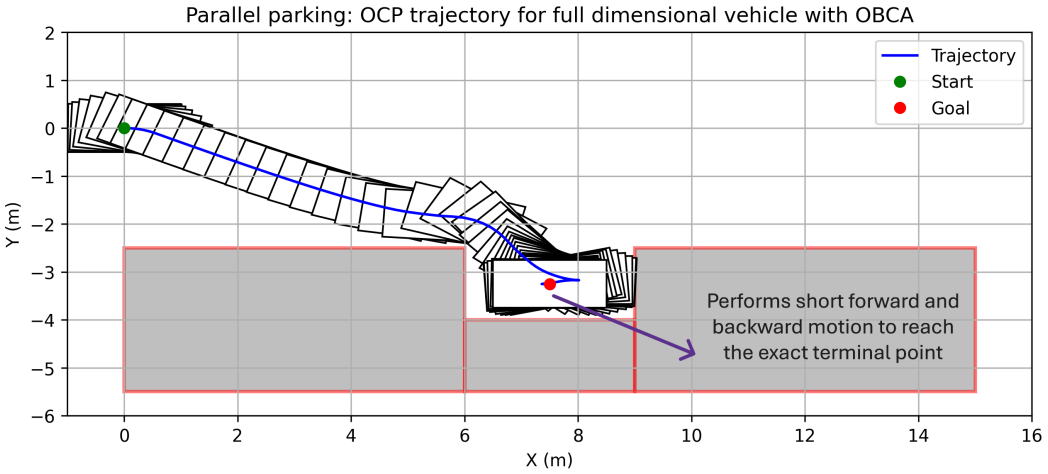


Figure 3: Collision-free trajectory using OCP with $d_{min} = 0.1$m for parallel parking parking

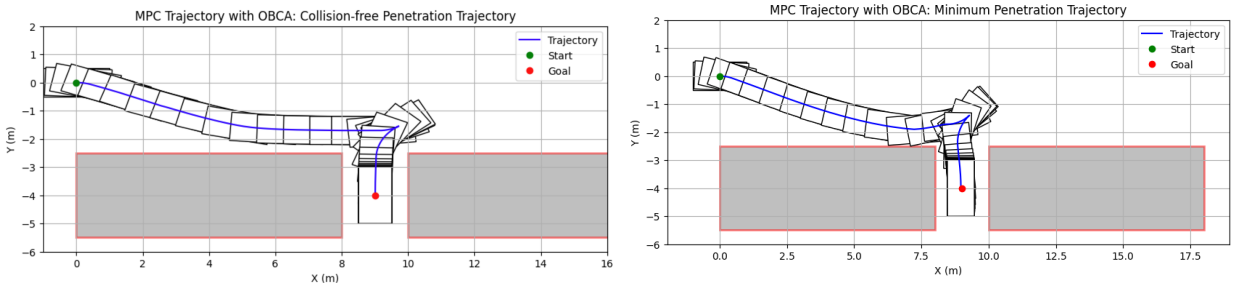
5.2.1 Reverse Parking

For the reverse parking maneuver, we chose a prediction horizon of 60 timesteps, each timestep is 0.1 sec long. The total simulation time is 12 sec.

A. Collision-free case: For the collision-free trajectory we chose a $d_{\min}$ value of 0.3 m. Figure 4a shows the trajectory of the car as it successfully maneuvers to the goal state between the obstacles all the while maintaining a distance more than 0.3 m from the obstacles.

B. Minimum penetration case:

The minimum penetration trajectory is obtained for $\kappa = 50{,}000$, as illustrated in Fig. 4b. The remaining MPC parameters—namely Q, Q_f, R, and the horizon length N—are identical to those used in the collision-free trajectory case. It is evident that the controlled object makes contact with one of the static obstacles while reaching the goal pose.

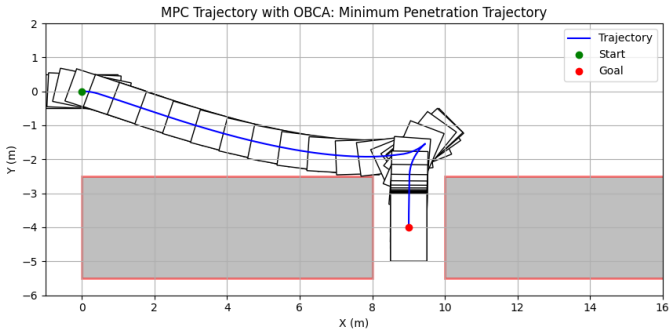


(a) Collision-free trajectory with $d_{min} = 0.3$m (b) Minimum penetration trajectory with $\kappa = 50{,}000$

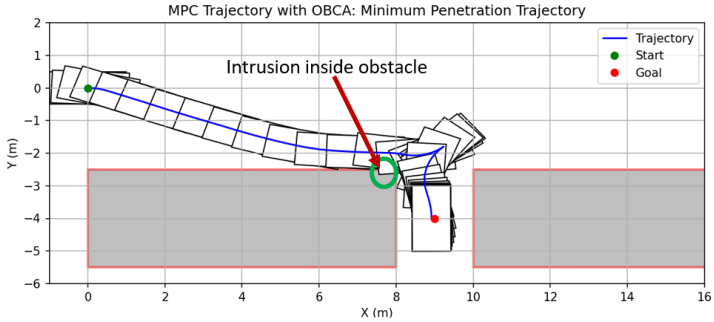
Figure 4: Comparison of MPC-based reverse parking strategies

To further analyze the influence of the penalty parameter κ on the intrusiveness of the trajectory, two additional cases are considered: $\kappa = 100$ and $\kappa = 500{,}000$, as shown in Fig. 5. For a low value of κ (100), the resulting trajectory is more intrusive, with the vehicle passing through the static obstacle. Moreover, the computation time for the entire simulation is

203 seconds for this case. In contrast, for a high value of κ (500,000), the object avoids penetration entirely. However, the computation time for this case 248 seconds. The larger weight (κ) in the regularization term of (22) forces the slack variable to be as close to zero as possible, which explains the larger computation time. In the ideal limit where κ is very large, the slack variables tend to zero, thereby recovering behavior similar to strict collision avoidance. However, it should be noted that increasing κ makes the optimization problem more difficult to solve, especially when tight constraints are present.



(a) $\kappa = 500{,}000$



(b) $\kappa = 100$

Figure 5: Impact of κ variation on minimum penetration trajectory for reverse parking

5.2.2 Parallel Parking

We further test efficacy of constraint by simulating them alongside MPC for a parallel parking maneuver. In this section, we compare two cases, i.e., MPC with collision avoidance constraint and MPC with penetration constraint.

A. Collision-free case:

For this case, we had to increase the weight matrices in the cost by an order of magnitude in comparison to the reverse parking case, in order to be able to successfully complete the maneuver. We also had to decrease the minimum allowable distance ($d$min) to 0.1 as compared to 0.3 in case of reverse parking. This emphasizes that the parallel parking maneuver is harder to complete. The prediction horizon for MPC was set to 60 time steps with each time step, $\Delta t = 0.1$s. Rest of the parameters for this case are mentioned in 3.4.

In figure 6, notice that the vehicle maintains the specified minimum safety margin with the obstacles. Although the safety margin saves the vehicle from collision, it virtually reduces the parking space available to the vehicle, i.e., $3 - 2 * d_{min} = 3 - 0.2 = 2.8m$. This reduction in maneuverable space for the vehicle prevents the vehicle from performing short forward and backward motion to reach the exact terminal point. Hence, the vehicle converges at an offset from the terminal point. The total time of maneuver is 9 seconds.

B. Minimum penetration case:
In this case, we keep all the parameters same except that for the minimum penetration constraint, we define κ instead of d_{min}. Here, we choose $\kappa = 60,000$. The value of κ for a particular simulation is frozen via trial and error. We wait till the solver becomes stable and converges for the entire duration of the simulation. Allowing the vehicle to penetrate the obstacle means that we are virtually extending the maneuverable space available to the vehicle, hence allowing it to converge exactly to the terminal point.

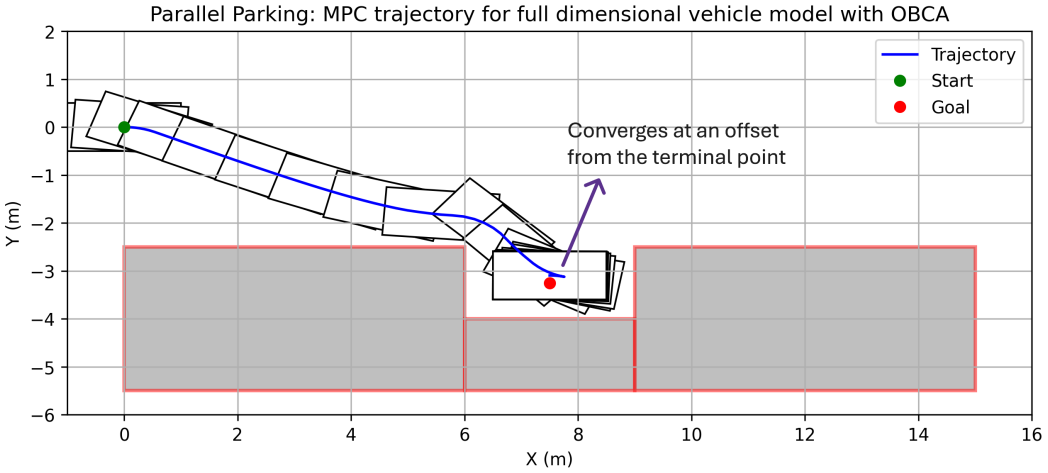


Figure 6: Collision-free MPC trajectory with $d_{min} = 0.1$m for parallel parking

5.3 Dynamic Obstacle Avoidance

The dynamic obstacle avoidance problem is illustrated in the context of the reverse parking scenario.

5.3.1 Translating Obstacle

In this case, the dynamic obstacle moves along the y-axis with a constant velocity of $0.5, \mathrm{m/s}$. The objective is for the controlled vehicle to reach the designated goal position while avoiding both the dynamic and static obstacles.

A. OCP Case:
In this case, the problem is formulated to generate a collision-free trajectory. The optimization is found to be feasible for a minimum separation distance of $\mathbf{d_{min} = 0.1}$ m, applied

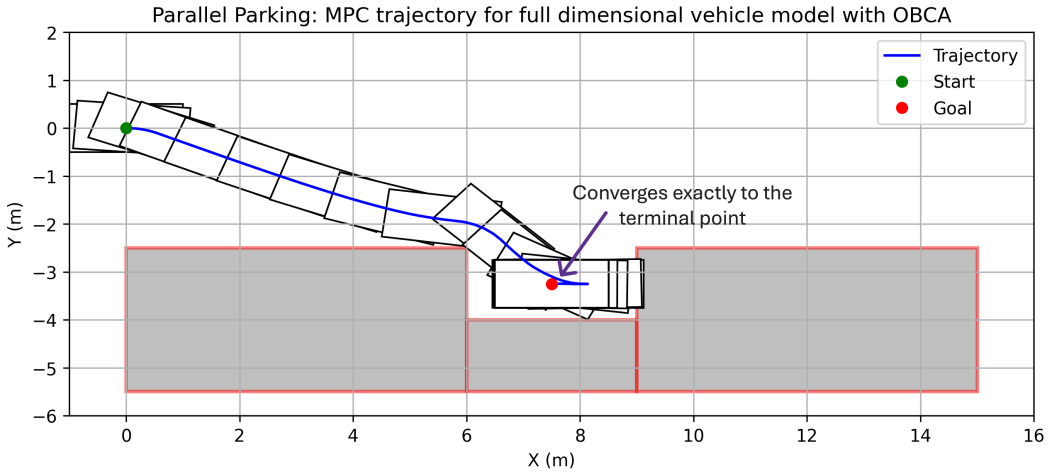
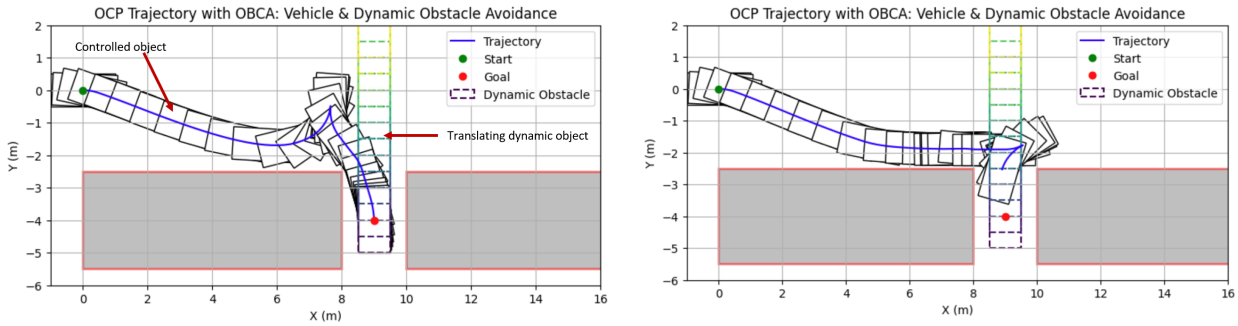


Figure 7: Minimum penetration trajectory with $\kappa = 60,000$ for parallel parking

uniformly to both static and dynamic obstacles.

Two key observations arise from the single-shot OCP formulation. First, when the terminal boundary condition—i.e., reaching the exact goal state—is explicitly imposed, the problem becomes feasible only for relatively long prediction horizons. Specifically, feasibility is achieved for horizon lengths exceeding $N = 120$. Figure 8a illustrates the resulting trajectory for $N = 120$, where the controlled vehicle successfully reaches the target pose.

Conversely, when the terminal constraint is omitted from the OCP formulation, the problem remains feasible even for smaller horizon lengths. However, in such cases, the ego vehicle fails to reach the goal pose, even when $N = 120$, as depicted in Figure 8b.



(a) With terminal constraint

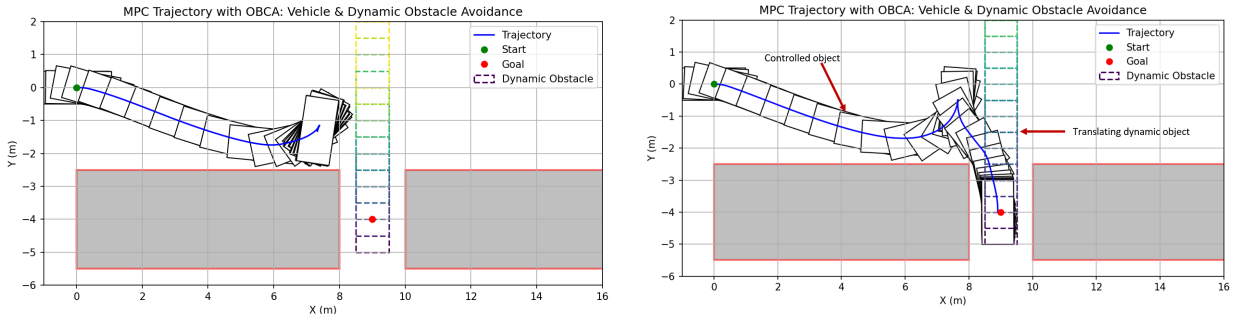
(b) Without terminal constraint

Figure 8: Single-shot OCP for dynamic (translating) obstacle avoidance for $N = 120$

B. MPC Case:

The same dynamic obstacle avoidance problem is now addressed using an iterative manner in the Model Predictive Control (MPC) formulation. It is observed that for shorter prediction horizons, the controlled vehicle is unable to reach the goal. When the horizon length exceeds $N = 80$, the MPC problem may converge in the initial iteration but often fails to converge in subsequent steps. This highlights a key limitation: **recursive feasibility** is not guaranteed in the presence of dynamic obstacles.

Furthermore, the choice of an appropriate MPC horizon does not exhibit a consistent or predictable trend; instead, it is highly sensitive to the specific dynamical conditions of the scenario. Figures 9a and 9b illustrate the **collision-free trajectory** evolution under MPC for horizon lengths $N = 20$ and $N = 60$, respectively. The **minimum penetration**



(a) Prediction horizon $N = 20$ (b) Prediction horizon $N = 60$

Figure 9: MPC formulation for dynamic (translating) obstacle avoidance

trajectory for the case of a translating dynamic obstacle is shown in Fig. 10 for a prediction horizon of $N = 60$. The problem is found to be feasible for a penalty weight of $\kappa = 50{,}000$, and the corresponding trajectory is generated under this setting. In this case, the motion of

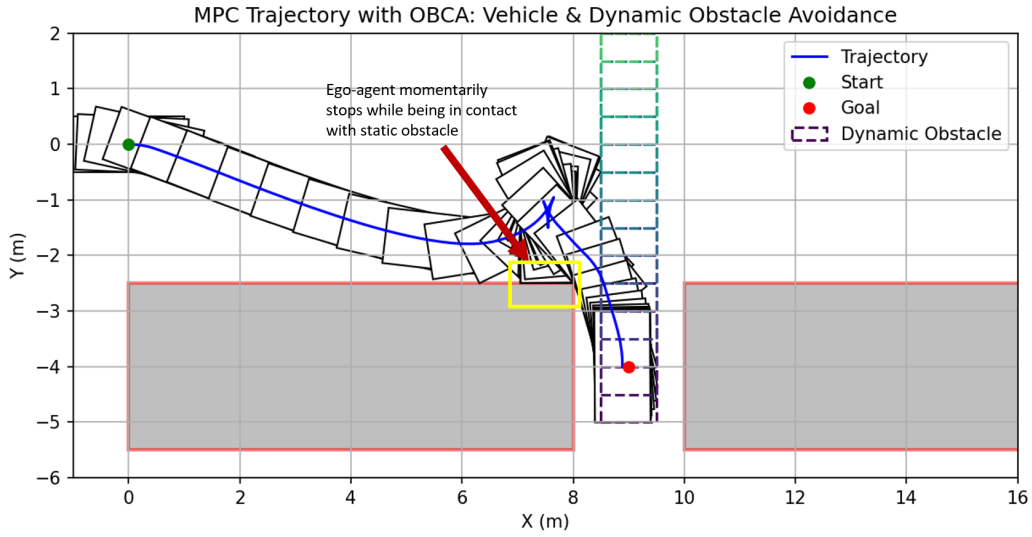


Figure 10: Minimum penetration trajectory with dynamic obstacle (translating) for $\kappa = 50,000$

the ego agent differs slightly from that of the collision-free trajectory. Notably, the vehicle exhibits a momentary stop while in contact with a static obstacle, allowing the dynamic obstacle to pass before completing the remainder of its path. This behavior is consistent with the intended characteristics of the minimum penetration formulation, where controlled intrusion into obstacles is permitted but penalized rather than strictly avoided.

5.3.2 Rotating obstacle

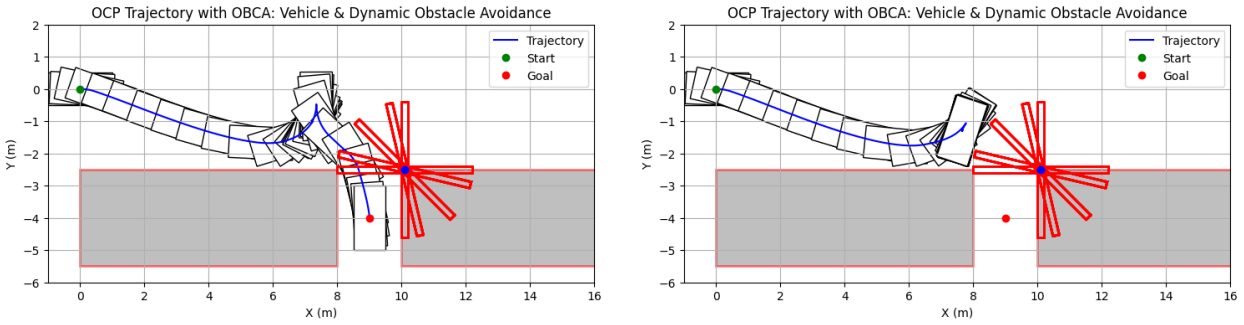
In this case, the dynamic obstacle periodically rotates about its center at the point $(10, -2.5)$, which is the corner of its entry point for its goal position in the reverse parking case. The obstacle rotates a 90° angle with a frequency of 0.25 Hz. The objective is for the controlled vehicle to reach the designated goal position while avoiding both the dynamic and static obstacles. We had to relax the bounds on velocity slightly for this problem in order to be able to achieve the goal position, $v \in [-2\text{m/s}, 2\text{m/s}]$.

A. OCP Case:

In this case, the problem is formulated to generate a collision-free trajectory. The optimization is found to be feasible for a minimum separation distance of $\mathbf{d}_{\min} = \mathbf{0.1}$ m, applied uniformly to both static and dynamic obstacles.

We observe that when the terminal boundary condition—i.e., reaching the exact goal state—is explicitly imposed, the problem becomes feasible only for relatively longer simulation horizons. This is as per expectation since the obstacle keeps going back and forth blocking the path of the ego vehicle fully and the vehicle needs time to wait and reach the goal position when it sees a chance to go through, while maintaining the bounds on the states (in this case its velocity) and the control inputs (acceleration and steering angle). Specifically, feasibility is achieved for horizon lengths exceeding $N = 120$. Figure 11a illustrates the resulting trajectory for $N = 120$, where the controlled vehicle successfully reaches the target pose. Similar to the case of the translating obstacle, when the terminal constraint is omitted from the OCP formulation, the problem remains feasible even for smaller horizon lengths, but the vehicle fails to even enter the parking zone, even when $N = 120$, as depicted in Figure 11b.

B. MPC Case:



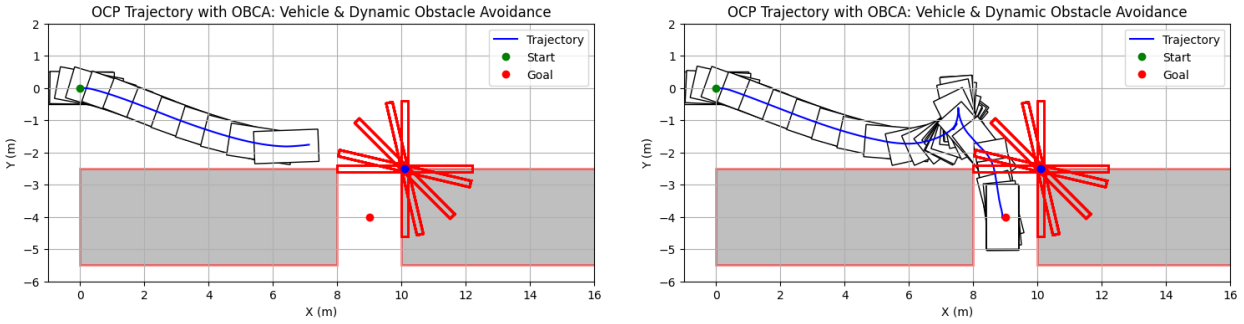
(a) With terminal constraint

(b) Without terminal constraint

Figure 11: Single-shot OCP for dynamic (rotating) obstacle avoidance for $N = 120$

We now show the results of using MPC to address the rotating dynamic obstacle avoidance problem. Similar to the translating obstacle case, we found that for prediction horizons shorter than 80 (for a total simulation horizon of 160 (16 sec)), the controlled vehicle is unable to reach the goal. Further, for the same total time of simulation, when the horizon length exceeds $N = 80$, the MPC problem may converge in the initial iteration but often fails to converge in subsequent steps, which also shows the lack of the guarantee of **recursive**

feasibility in the presence of dynamic obstacles. Due to this non-intuititve trend, finding an appropriate prediction horizon was a challenging task and needed significant amount of tuning. The particular choice is highly problem-specific and also depends on the speed of the dynamic obstacles. Interestingly, for the two very different scenarios of our translating and rotating obstacles, similar values worked. Figures 12a and 12b illustrate the **collision-free trajectory** evolution under MPC for horizon lengths $N = 20$ and $N = 80$, respectively. In figure 13 we show the trajectory for minimum penetration case for the dynamic rotating obstacle avoidance. We used a kappa value of 10,000 which resulted in attaining the goal successfully and no visible penetration, although the vehicle brushes past both the static and dynamic obstacle multiple times.



(a) Prediction horizon $N = 20$ (b) Prediction horizon $N = 80$

Figure 12: MPC for dynamic (rotating) obstacle avoidance (simulation time is 16 sec)

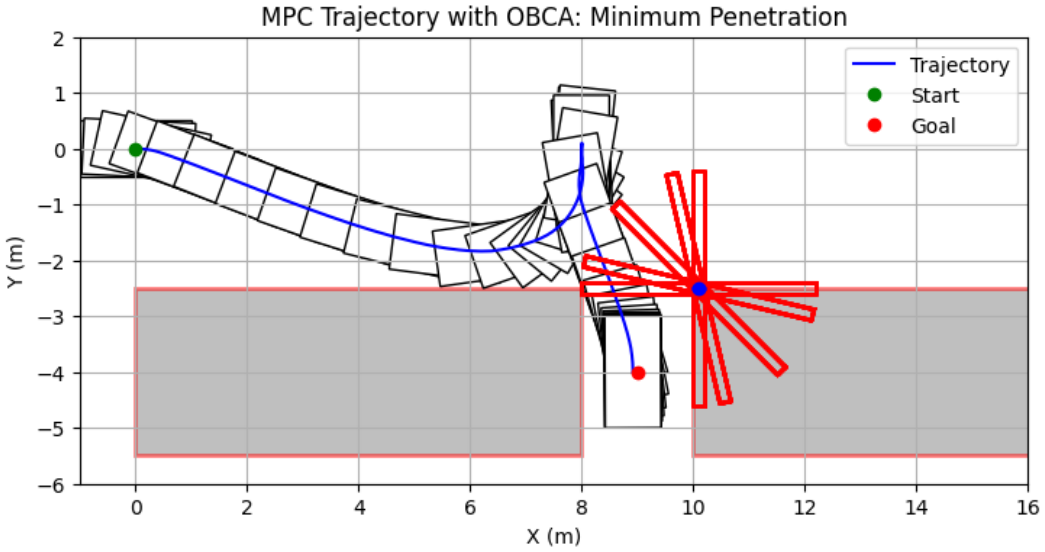


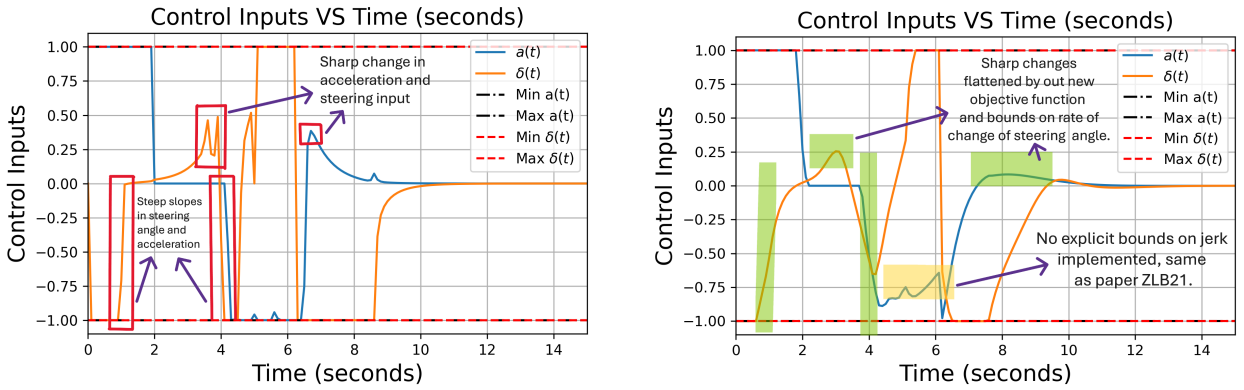
Figure 13: Minimum penetration trajectory for dynamic (rotating) obstacle with $\kappa = 10,000$

5.4 Study on Minimum Jerk Optimal Control Problem

In our previous simulations, we observed that the rate of change of control were high in magnitude. Steep slopes in the acceleration indicate high magnitudes of jerk in longitudinal motion of vehicle, and high rate of change in steering angle may breach the actuation capacity of the motor governing the steering angle, see figure .
To tackle this issue, we updated the previously used objective function with with an additional term to penalize the rate of change of control inputs, Δu_k.

$$\text{Penalty on } \Delta u_k\text{: } \sum_{k=0}^{N-1} \Delta u_k^T S \Delta u_k \tag{36}$$

, where S is a positive-semi definite matrix containing the penalty weights on Δu_k. This particular update as indicated by figure , smoothens out the sharps changes in control inputs ultimately aiding passenger comfort and ensuring safer actuation.



(a) Shape changes in control input without penalizing Δu_k in the objective function.

(b) Smoother profiles of u_k after penalizing Δu_k in the objective function.

Figure 14: Impact on control inputs due to minimum jerk objective function

6 Discussion

The following observations highlight important insights drawn from our simulation results:

- **Minimum penetration formulation offers robustness but at a computational cost.**
  This approach introduces additional variables to explicitly minimize the overlap between the vehicle and obstacles. As a result, it is computationally slower to solve. However, its robustness makes it particularly advantageous in constrained or cluttered environments, where strict avoidance is critical.

- **Minimal retuning is needed when adapting to dynamic obstacles for the minimum penetration trajectory.**
  In the case of minimum penetration formulations, the transition from stationary to

moving obstacles required little to no re-tuning of parameters, which further points to its robustness.

- **One-shot optimal control problem (OCP) with hard terminal constraints reliably achieves the goal.**
  The OCP approach, when enforced with exact terminal conditions (i.e., final state exactly matching the goal), successfully completed both parallel and reverse parking maneuvers. In contrast, Model Predictive Control (MPC) with similar terminal constraints often became infeasible for comparable time horizons, likely due to accumulated numerical or modeling errors across the horizon.

- **MPC without terminal constraints yields better goal tracking.**
  Despite the infeasibility with hard terminal constraints, MPC demonstrated superior tracking performance when such constraints were removed. This is attributed to its receding horizon and real-time feedback structure, which helps it adapt dynamically to intermediate state deviations.

- **Parallel parking presents a greater challenge than reverse parking.**
  Comparative analysis of convergence behavior and parameter sensitivity showed that parallel parking required more precise tuning and had a higher rate of solver failure, confirming its higher complexity.

- **Obstacle uncertainty destabilizes MPC and motivates the use of Stochastic MPC.**
  The introduction of even minor uncertainties—such as Gaussian noise in the perceived position or velocity of obstacles—led to repeated MPC failures. This highlights the need for robust formulations like Stochastic MPC to explicitly account for uncertainty in obstacle dynamics.

- **Prediction horizon length is critical in dynamic settings, but tuning it is non-trivial.**
  In the presence of moving obstacles, the prediction horizon directly influences performance and feasibility. However, the relationship is non-monotonic; neither longer nor shorter horizons consistently improve results. This suggests the presence of a "sweet spot"—a specific horizon length that balances foresight and adaptability. This shows that recursive feasibility is not guaranteed, making careful tuning essential.

7 Conclusion

In this work, we implemented an optimization-based approach to obstacle avoidance for generating both collision-free and minimum penetration trajectories, utilizing both one-shot optimal control problems (OCP) and model predictive control (MPC). The formulation enables precise control over trajectory behavior through tunable parameters. In particular, adjusting the penetration penalty weight (κ) was found to directly influence the extent of obstacle intrusion as well as the overall simulation time, offering a trade-off between computational cost and robustness.

To smoothen the control profiles, we included a penalty on changes in the control input within the cost function. This regularization not only improved the trajectory quality but also enhanced numerical stability during optimization. Furthermore, we extended the method introduced in [ZLB21] to accommodate dynamic obstacles—both translating and rotating—by introducing time-varying constraints. This allowed the generation of safe trajectories in more realistic, time-evolving environments, for both collision-free and minimum-penetration cases.

In summary, the main advantages of the optimization-based collision avoidance methods presented here include the exact inclusion of geometric obstacle constraints within the optimization framework and the ability to generate minimum penetration trajectories in constrained scenarios. However, certain limitations remain, such as the lack of guaranteed recursive feasibility, sensitivity to obstacle uncertainty, and the computational overhead that may hinder real-time implementation on hardware.

8 Table of Contribution

The following table 2 highlights the contribution of each team member in the successful completion of this project.

The team members contributed equally in preparing the presentations and reports for both the midterm and final review. Hence, those are not separately mentioned in the table.

Team Member	Contribution
Ayush Agrawal	• Implemented the one-shot OCP and MPC formulation for collision-free parallel parking problem (static obstacles). • Implemented the minimum penetration constraint with MPC for parallel parking. • Implemented objective function with penalty and bounds on Δu_k to generate more practically feasible control inputs.
Debajyoti Chakrabarti	• Developed minimum penetration controller for reverse parking • Studied the impact of κ variation on the intruded trajectory • Developed single-shot OCP for translating dynamic obstacles • Implemented MPC formulation for translating dynamic obstacles (which includes both collision-free and minimum penetration trajectories).
Radha Lahoti	• Implemented the obstacle avoidance constraints. • Developed the one-shot OCP and MPC formulation for the reverse parking problem (static obstacles) for both collision-free and minimum penetration trajectories. • Developed the one-shot OCP and MPC formulations for rotating dynamic obstacle for the reverse parking problem.

Table 2: Team Members and Their Contributions

References

- X. Zhang, A. Liniger, F. Borrelli, "Optimization-Based Collision Avoidance," IEEE TCST, 2021. `https://github.com/XiaojingGeorgeZhang/OBCA`

Github Repository Link

`https://github.com/Radha-Lahoti/MAE271D-Seminar.git`

The jupyter notebook files have been named according to the content.

Overleaf Link

`https://www.overleaf.com/7536857387vsxzkvtjzcfw#8c2a4b`