LOCAL PATH PLANNING VIA MPC FOR SAFE NAVIGATION IN UNKNOWN MULTI-VEHICLE ENVIRONMENTS

LOCAL PATH PLANNING VIA MODEL PREDICTIVE CONTROL FOR SAFE NAVIGATION IN UNKNOWN MULTI-VEHICLE ENVIRONMENTS

By CHRISTIAN SCHAIBLE, B.Eng.

A Thesis

Submitted to the Department of Electrical & Computer Engineering

and the School of Graduate Studies

of McMaster University

in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science

Master of Applied Science (2025) McMaster University
(Electrical & Computer Engineering) Hamilton, Ontario, Canada

TITLE:	Local Path Planning via Model Predictive Control for Safe Navigation in Unknown Multi-Vehicle Environments
AUTHOR:	Christian Schaible B.Eng. (Electrical Engineering), McMaster University, Hamilton, Canada
SUPERVISOR:	Prof. Shahin Sirouspour
NUMBER OF PAGES:	xviii, 193

Lay Abstract

In the field of autonomous vehicles and mobile robots, a future path is determined given certain sensor information about the surroundings. Some contexts like search and rescue, racing, mapping and exploration require a robot to make intelligent path planning decisions with no prior information on what the environment looks like, what obstacles exist or where to go. Here, a standalone local path planner is needed to provide a reasonable trajectory based on limited information. This thesis presents algorithms that allow vehicles to traverse safe local paths in real-time when the environment is unknown. Safety is obtained by maintaining large distances to nearby obstacles where possible while also promoting a smooth, controlled path. Extensions to this framework dynamically promote higher speeds, account for the future paths of other detected vehicles and maintain path safety while simultaneously pursuing a leader vehicle. Simulation and experimental results show the comparison of algorithms detailed in the thesis and how safer, superior paths are made compared to existing local planners.

Abstract

This thesis presents the development of safe and versatile local path planning algorithms through Model Predictive Control (MPC) in the absence of a global path planner for applications such as search and rescue, mobile robotics and autonomous vehicle racing. Rather than formulating a local planner that tracks and controls towards a known global path, the foundational novel method presented in this thesis relies solely on continually updating an optimal local reference trajectory and solving a non-convex optimization MPC problem for control commands to track this path.

Successive tracking lines are generated through a quadratic optimization to maximize the distance to nearby obstacles (detected through LiDAR) while having heading directions aimed towards large open spaces. Quadratic costs are derived to follow these lines closely, subject to nonholonomic system constraints as described by the kinematic bicycle model. Via a Sequential Least Squares Quadratic Programming (SLSQP) online solver utilizing analytical gradients, feasible and locally optimal solutions are reliably found in real-time operating circumstances.

This approach is extended to incorporate dynamic obstacles into vehicle avoidance. Vehicle detection exploits a custom-trained Convolutional Neural Network (CNN) using the You Only Look Once (YOLO) architecture. Red, Green, Blue (RGB) detections are projected into depth space, and an Extended Kalman Filter (EKF) obtains predicted vehicle paths. These tracked vehicle paths are then added to the path planning algorithm to handle adversarial vehicle avoidance in multi-vehicle scenarios.

An alternative formulation introduces a fourth-order Bezier curve model in place of the successive tracking lines, which combines the generation of an optimal path and actuation to the path into one non-convex optimization. Constraints on vehicle dynamics are incorporated directly into the construction of the curve, and potential fields from nearby obstacles ensure a safe path is maintained. Computation complexity is reduced, and smooth paths are reliably found in real-time.

The aforementioned local planners are incorporated in a leader-follower hierarchy, which balances the need for pursuit tracking with the safety of the generated path as before. Arbitrary following configurations are enabled through the pursuit formulation, and adaptive pursuit vs. safety weightings are dictated by continuously updating obstacle proximity. This framework sets the basis for modular and extendable flexible formation fleets in unknown areas using only local path planning.

Results are exhibited in both a simulation environment and on a 1/10th scale autonomous vehicle. Real-time navigation is achievable, and trajectories are shown to be safer and achieve superior performance compared to existing local planners when the assumption of a known global planned path is removed. The algorithms presented in this thesis are compared, and their ability to obtain their desired objectives in varying environments is shown.

Acknowledgements

My sincerest thanks are extended to all those who made this research possible. First, I would like to thank my supervisor, Dr. Shahin Sirouspour, for his guidance and direction during the undertaking of my MASc studies. The invaluable experience I have gained in his lab has enriched not only my academic knowledge but also my personal and professional growth. Additionally, I would like to thank Dr. Tim Davidson for providing me with my first exposure to research as an undergraduate student and for continuing to share his wealth of knowledge and experience with me during my graduate endeavors.

Moreover, I appreciate my fellow graduate students, Amin Eljirby & Milad Farjadnasab, for sharing their wisdom during my studies. Their support as teaching assistants during my undergraduate education and later as graduate school lab colleagues has guided my research interests and aspirations. Finally, I am grateful for the steadfast support and love from my parents and sister during my academic journey. I am very fortunate to have such excellent role models in my family and appreciate all the ways each has contributed to making both my academic and personal achievements possible.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AMCL	Adaptive Monte Carlo Localization
APF	Artificial Potential Field
CNN	Convolutional Neural Network
DWA	Dynamic Window Approach
EKF	Extended Kalman Filter
FGM	Follow the Gap Method
FOV	Field of View
GMM	Gaussian Mixture Model
GPU	Graphics Processing Unit
IMU	Inertial Measurement Unit
KF	Kalman Filter
LiDAR	Light Detection and Ranging
MPC	Model Predictive Control

MPCC	Model Predictive Contouring Control
MPPI	Model Predictive Path Integral
P-MPC	Pursuit Model Predictive Control
P-QBMPC	Pursuit Quartic Bezier Model Predictive Control
P-STLMPC	Pursuit Sequential Tracking Line Model Predictive Control
PD	Proportional-Derivative
PID	Proportional-Integral-Derivative
QBMPC	Quartic Bezier Model Predictive Control
RGB	Red, Green, Blue
RGB-D	Red, Green, Blue - Depth
ROS	Robot Operating System
RRT	Rapidly-exploring Random Tree
SLSQP	Sequential Least Squares Quadratic Programming
SQP	Sequential Quadratic Programming
STLMPC	Sequential Tracking Line Model Predictive Control
TEB	Timed Elastic Band
SVM	Support Vector Machine
YOLO	You Only Look Once

Chapter 1

Introduction

1.1 Motivation

The development of faster real-time hardware has proved to be a valuable synergy for modern robotics and autonomous vehicle systems. To achieve safe and efficient navigation, autonomous systems need to make intelligent decisions quickly and responsively [1]. Increased sensor and computing capabilities have expanded the set of methods through which these decisions can be effectively made. Extensive prior research explores the navigation of autonomous vehicles & mobile robots in environments that have defining, structured characteristics (such as roads) [2–5] or are known in advance of the robot's travel [6–9]. These assumptions are impractical in general situations, particularly in applications such as search and rescue, certain racing scenarios, exploration & mapping tasks, as well as military operations. In such events, the environment is unstructured, unknown, and a desired goal position is elusive.

Local path planning assumes no knowledge of the robot's global surroundings and thus, many existing navigation methods are non-transferable to these situations. The focus in these circumstances turns from generating an ideal path to a goal towards instead continuously updating the most sensible short-term trajectory as the robot progresses through the unknown space. Guarantees on successful passage are limited given this imperfect global knowledge, so prioritizing safe and locally optimal paths becomes the target.

A unifying, parameterized framework is needed to generalize successful navigation from a single, assumed environment to the diverse and unknown spaces encountered in real-world conditions. Some simple local path planning approaches have limited forward-looking capabilities, focusing predominantly on the next control action while relying on a predefined global reference path to track [10, 11]. More thoughtful existing exploratory techniques chart local trajectories but again rely on a global path for tracking [12–15]. When these local planners are paired with frontier exploration in the absence of a known fixed global path, performance degrades and distance to obstacles decreases, heightening the risk of collisions. Thus, a thorough construction of a receding optimal, feasible path is introduced that can be transferable to an arbitrary environment with no prior knowledge of a desired global path.

The deficiencies in existing standalone local path planners motivate the development of a new framework for local navigation. This framework should prioritize safety and establish a general, environment-independent formulation to simultaneously generate and actuate to an optimal, feasible local trajectory. Leveraging modern hardware capabilities, this framework generates safer paths in real-time than existing standalone local planners, with adaptations meeting further control desires. This thesis presents the formulation of such a framework and extensions to further variants. These include one that tracks and avoids dynamic obstacles, one that prioritizes faster speeds, one that generates smooth paths even computationally faster and one that maintains a safe path while synchronously pursuing a leader vehicle.

1.2 Problem Statement

A major challenge of local path planning is the lack of broader situational knowledge that comes with a global plan. Sensible path planning decisions can now only be made with local context, and so our ability to reach certain unknown global regions may be impaired. The robot's objective is therefore reformulated from reaching a known location to instead progressing along the best choice of local path for the duration of its travel. The choice of the best local path, formulated and continuously updated online, is typically made with respect to a known global path but must now be determined separately. Therefore, the dual problem of generating an optimal path and controlling the vehicle to follow the path arises. Model Predictive Control (MPC) is a commonly used method for generating and controlling towards desired paths [1]. In MPC, an optimization is formulated to obtain the system control inputs over a future receding time horizon; the first input is applied, and the process is repeated for the next sample time. A general approach is needed for this optimal local path design that applies in any arbitrary environment where the vehicle could be deployed.

Another prevalent concern for path planning is reliably computationally tractable solvers for real-time applications. Since responsive autonomous systems must continually update and adapt on the scale of fractions of a second, control decisions need to be determined quickly. Path planning problems typically yield non-convex & non-linear optimizations [16, 17] which are difficult to quickly solve globally. Some methods simplify the problem through linearizations [18]; however, model accuracy and applicability are lost. Important non-linearities can arise from steering angle dependencies and vehicle dynamics based on the kinematic bicycle model. These relationships must be fully expressed in order to achieve accurate trajectories over longer horizons. Using Sequential Quadratic Programming (SQP) solvers is a way to efficiently obtain good solutions to non-linear, non-convex optimizations [19, 20] in real-time if valuable insights into the problem are exploited. For the sacrifice of global optimality, a sufficient locally optimal solution is instead used, which, when paired with a sensible starting guess, yields promising results.

In global planners, the existence of dynamic obstacles raises the need for responsive local planning to update the trajectory in real-time. This issue applies to the case of local planning in unknown environments with greater flexibility, as the ideal reference path can be greatly affected by the behavior of moving obstacles. Synchronized sensor fusion is required for object detection and tracking in order to both identify specific moving obstacles and quantify state properties. This matter becomes more complex in unstructured environments when compared to roads, as moving obstacles can come in many forms and arrangements, increasing the dimensionality of the problem space. Instead of representing estimated future trajectories assuming zero curvature [21], a constant curvature model is used to increase the accuracy of the predicted model for arbitrary obstacle motion. By fully parameterizing all nearby moving obstacles and monitoring changes in motion, the local planner can be designed to predict the future behavior of hazards. Thus, the vehicle can correspondingly act early to avoid collisions & formulate safer paths in unknown territories and contexts like multi-vehicle racing.

With developments in single robot path planning, increased interest has turned towards collaborative multi-robot exploration teams. Fixed formations are maintained in the absence of obstacles [22], whereas individual vehicle safety is preserved through collision avoidance [23], thus temporarily breaking formation. In new, unknown environments, local planners must achieve path safety through a flexible fleet formation. Fleets may have communication channels between agents [24]; however, this may not always be the case, which complicates the problem. A general structure is desired for local planning in unknown areas, which enables arbitrary formation patterns, sizes and shapes through modular characteristics.

Current outstanding challenges in the literature motivate the design of a universal local path planner for situations where the surroundings are unknown and inter-vehicle collaborative or adversarial dependencies may exist. A framework must be introduced to enable real-time, responsive path planning based only on current knowledge and predictions while incorporating system non-convexities and non-linearities to preserve accuracy. Safety considerations should be prioritized while also meeting requirements on vehicle dynamics and inter-vehicle interaction. This thesis seeks to explore the development of such a local path planning formulation.

1.3 Thesis Contributions

In this thesis, the challenge of safe local path planning in unknown, multi-vehicle circumstances is addressed through the construction of novel MPC algorithms with accessible, open-source implementation[1] for general use in real-time operation. These proposed planners rely only on local sensor data (which can be obtained via LiDAR) without a known global path or desired goal location. Safer paths are yielded than existing exploration methods, making these planners attractive for search and rescue

[1]`https://github.com/schaiblc/McMaster_AEV_MPC_Algorithms`

applications. Subsequent developments extend the standalone local planner to accommodate more complex situations involving other vehicles, both through collision avoidance and pursuit navigation. Thus, these local planning approaches can also be applied to autonomous multi-vehicle racing contexts for safe navigation on unknown courses. Non-convex MPC optimization problems are formulated, and smooth, continuous analytical gradients are exploited to enable fast, real-time solving through an SQP-based solver.

A novel MPC local planner and control algorithm is proposed, entitled Sequential Tracking Line Model Predictive Control (STLMPC). This approach solves successive quadratic programming optimizations to obtain discontinuous line segments representing the ideal local reference path. This cumulative path maximizes distance to local obstacles while proceeding in the direction of larger, open local spaces. The kinematic bicycle model is used to represent nonholonomic vehicle constraints and limits on control inputs. Costs are attributed to tracking accuracy of the local reference path and control effort expended. The MPC's receding prediction horizon accounts for future behavior in the present and updates at the controller rate (equivalent in practice to that of the LiDAR sensor's publishing rate). Detected & tracked vehicle trajectories are estimated based on a constant curvature model and are considered temporally to mitigate collision risk over the future time horizon. Forward velocities are maximized subject to safety limits under a fast exploration alternate scheme. Velocity restrictions arise based on turning angle conditions and obstacle proximity. To the author's knowledge, this is the first formulation of a safe, standalone MPC local planner with the aforementioned design targets, applicable to multi-vehicle contexts.

An alternative algorithm that combines planning and control into a single path is presented through Quartic Bezier Model Predictive Control (QBMPC). A fourth-order Bezier curve smoothly incorporates constraints on vehicle dynamics while also meeting limitations on the path to obstacle vicinity. Potential field-like forces are developed to obtain a curve that maintains safety from hazards. Smooth paths are generated, and a reduction in optimization variables ensues as the Bezier control points are found to describe the curve's motion. To the author's knowledge, this formulation is the first Bezier curve method to achieve simultaneous local planning and control in unknown environments with no tracking reference and incorporate physical limits on maximum changes in curvature & forward velocity. Also, this method enables the predicted path duration time to be arbitrarily scaled while maintaining the fixed parameterization of the Bezier curve.

The novel STLMPC and QBMPC approaches are modified to fit dynamic pursuit scenarios of a leader vehicle through the P-STLMPC and P-QBMPC algorithms, respectively. Adaptive weighting is provided to balance the focus on maintaining a pursuit formation as opposed to traversing a safe path based on instantaneous proximity to obstacles. These methods provide a flexible framework for arbitrary formation configurations and fleet sizes in unknown environments where local planning is exclusively used. Vehicle-to-vehicle communication is not required to coordinate formation, thus simplifying hardware requirements and removing the risk of communication failure. To the author's knowledge, this is the first approach to develop communication-less, flexible fleet formations that can contend with obstacle hazards in unknown environments using only local planning and no known tracking reference.

Finally, the novel algorithms are implemented in practice, and results are shown. These experiments show the applicability and performance of these methods across different testing setups and environments. Validation occurs both in simulation and experimentally using a 1/10th scale autonomous vehicle.

1.4 Organization of the Thesis

The remainder of the thesis is structured as follows. Chapter 2 explores existing literature focused on the areas of local planning, model predictive control, and multi-vehicle navigation contexts. Chapter 3 introduces the Sequential Tracking Line Model Predictive Control method, which serves as a basis for further navigation algorithms developed. The simulation and experiment testing architectures used in the thesis are presented, and results are illustrated for the performance of this base algorithm. In Chapter 4, vehicle detection and tracking via extended Kalman filtering are covered to establish the foundation of a multi-vehicle path planning approach. Vehicle avoidance is derived from the tracked vehicle paths and is implemented in STLMPC, where implementation results are provided.

Chapter 5 expands the complexity of the STLMPC method by introducing a non-constant velocity. Here, the velocity is influenced by vehicle and environmental conditions, which leads to more flexible, controllable path planning. In Chapter 6, an alternative formulation to STLMPC, the Quartic Bezier Model Predictive Control scheme is developed. This method aims to reduce complexity while providing smoother paths, and its performance is compared to STLMPC. Chapter 7 augments both STLMPC and QBMPC to consider the pursuit of a leader vehicle in an unknown space. Now, objectives on both pursuit tracking and maintaining safety in the local region are weighed for effective planning. Lastly, Chapter 8 completes the thesis by presenting conclusions and highlighting some possible directions for future research.

Chapter 2

Literature Review

This chapter presents an overview of existing literature pertinent to the topics discussed subsequently in this thesis. An overview of methods for local path planning is presented with a specific focus on the use of MPC in existing approaches. Previous work on detection & tracking of dynamic obstacles is explored, as well as planning methods that consider these factors. Existing research on the use of Bezier curves for path generation and smoothing is explored, and finally, studies on multi-vehicle fleet navigation problems are highlighted.

2.1 Local Path Planning Techniques

Navigation techniques for mobile robots are primarily categorized as either global planners or local planners [25]. Often, the role of the local planner is to accommodate real-time, unexpected changes in a known environment to track the global planner while performing online obstacle avoidance [26]. One simple method for doing this is through PID (Proportional-Integral-Derivative) control for tracking a desired trajectory. This approach is not optimization-based and thus has minimal computation time and can yield acceptable performance in simple simulated environments [11]. However, possible system constraints are not considered, and optimality is not defined, while performance is also reliant on a previously defined reference trajectory. Even in simple simulation cases, tracking a reference is done with smaller errors when using a more sophisticated MPC framework [27, 28]. PID controller performance is also highly subject to tuning, which can greatly affect tracking error and accuracy [29].

Another local planning technique that tracks a reference path is through the pure pursuit algorithm [30]. This method uses a chosen look-ahead distance to determine the necessary curvature that will move the vehicle to the identified point on the reference path. The vehicle effectively chases a point ahead of it on the reference trajectory, continually ensuring it tracks the path. This method's performance is highly dependent on the choice of look-ahead distance, and adaptive selection methods have yielded decreased tracking errors [10, 31, 32]. Variable linear velocities have been incorporated into pure pursuit, lending greater flexibility for safe operation [33]. This approach has limited computation and time costs but lacks the complexity of MPC methods, which can accommodate more difficult, dynamic local navigation scenarios.

The Artificial Potential Field (APF) method [34] seeks to determine the best local path for obstacle avoidance through an expression of attractive and repulsive forces. The presence of a desired goal location attracts the vehicle while nearby obstacles & hazards repel the path to maintain safety. Challenges with this approach have been identified, namely oscillations in the presence of obstacles & narrow passages, susceptibility to falling into local minima solutions and failure of passage when obstacles are closely spaced [35]. Modifications have subsequently been made to encourage the escape of local minima [36–38]; however, in more complicated environments, the computation costs rise and the risk of poor local minima selection increases. Therefore, many modern path planning approaches turn to using global planners that are grid-based, like Dijkstra's Algorithm [39] & A* [40], or sampling-based, like Rapidly-exploring Random Trees (RRTs) [41], for navigation to a known goal (the attraction force). Local planning is often done by one of the following methods, which ensure collision avoidance (the repulsive force) while tracking the global trajectory.

Through the Dynamic Window Approach (DWA) [12], the search space for the desired local path is reduced to two dimensions: the translational and rotational velocities. Only feasible velocity pairs are considered based on vehicle dynamics and unsafe trajectories that result in collisions are ignored. The optimization prioritizes a heading towards the goal location, clearance of nearby obstacles, and a larger forward velocity. Velocity pairs are sampled, and the pair producing the highest objective result is chosen. This approach places explicit considerations on vehicle dynamics constraints in contrast to previous methods and engages in collision avoidance. However, the velocity and curvature of the future trajectory are assumed to be constant, which is restrictive and impractical. Furthermore, cost weights are fixed, which can lead to poorer performance in different environments, and the path update rate may be lower than the real-time control rate. Recent work has aimed to extend DWA to cases with dynamic obstacles [21, 42] and make DWA adaptive to varying environments through reinforcement learning [43, 44].

Finally, using an elastic band algorithm, the nature of the global path is preserved while deforming to accommodate obstacle avoidance. Regional bubbles are introduced to express free space along the global trajectory for the collision-free path and are adjusted to satisfy real-time collision avoidance if the original path becomes unsafe [13]. The Timed Elastic Band (TEB) planner [14] extends on this by accounting for constraints on vehicle dynamics. Local solutions are found through large-scale optimizers for sparse systems, though the path update rate can become slower in complex environments. Shorter paths are encouraged, even more so in further work [45], which may lead to close proximity to obstacles & therefore collision risk under uncertainty or with dynamic obstacles. In these cases and in general, parameter tuning can highly impact performance [46]. MPC extensions of TEB have been examined [47]; however, MPC has come to be preferred in the place of TEB for local planning of robots with complex dynamics like nonholonomic constraints [15].

2.2 Model Predictive Control for Actuation & Planning

The application of Model Predictive Control (MPC) to real-time path planning and control purposes has been a more recent development, due to improvements in computational resources and algorithm efficiency [48, 49]. MPC was first widely used in oil & chemical industry control processes [50] where its ability to handle hard constraints and its predictive nature yielded better performance than classical controllers like bang-bang & PID [51]. MPC [52, 53] explicitly uses a model to solve an optimization problem for the optimal inputs over a future time horizon. Often, the cost function is formatted to track a reference (Figure 2.1) while considering system

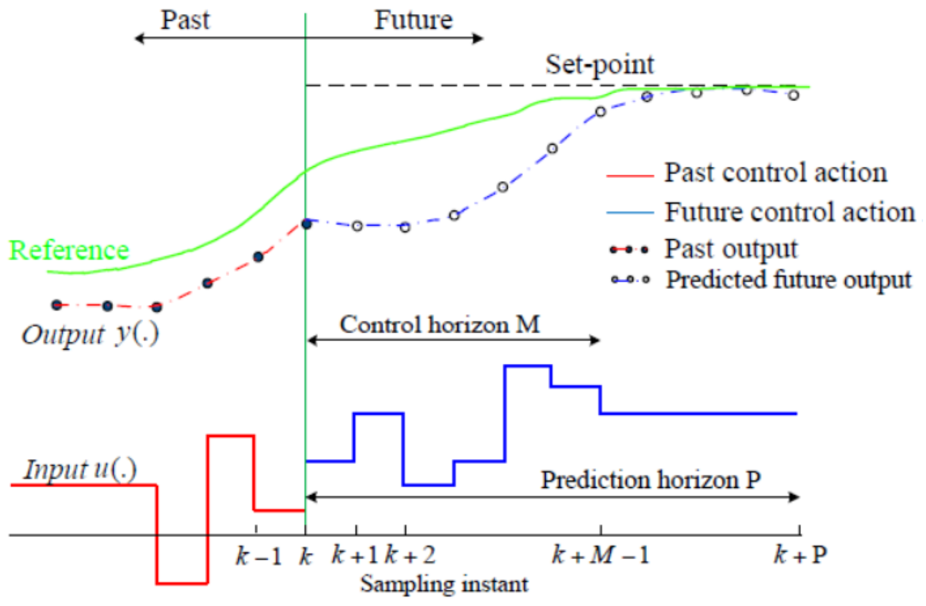


Figure 2.1: Graphical representation of Model Predictive Control, illustrating future control inputs, outputs and reference trajectory.

constraints. For each time step, only the first control action of the optimized future trajectory is applied before conducting the optimization again at the next sample [54]. This receding prediction horizon balances current and future performance, providing anticipatory behavior based on the configurable definition of optimality.

The use of MPC in autonomous vehicle and mobile robot applications is split into two categories: motion control and path planning [1]. For motion control purposes, the reference trajectory is already defined, and the goal of the MPC problem is to determine the optimal control inputs to track the known path accurately. This is often the role of MPC in local planners when performing obstacle avoidance and tracking a predefined global path. Extensive research on collision avoidance in road traffic conditions for autonomous vehicles has used MPC as a motion planner. One popular method uses a 3D potential field approach to avoid collisions, forming a reference trajectory which multiconstrained MPC then tracks, generating front steering angle commands [2]. MPC has also been used to control complex maneuvers like lane changes [55] while ensuring yaw stability [56] as well as maintaining lateral stability in icy road conditions [16]. In mobile robot contexts, potential field trajectory planning and MPC tracking have been applied with an adaptive prediction horizon length [57]. A prominent open-source MPC local planner [15] can both track a reference trajectory as well as perform collision avoidance while reaching intermediate goal positions along a globally known path.

Some local planners attempt to increase the role of MPC in path planning while also providing motion control. In a warehouse application, only a single goal location is known, and MPC is used to obtain the optimal local path and control inputs [58]. While this method removes the need for a global planner, significant assumptions in the formulation are made on the warehouse environment to ensure success, and the obstacle setup used in simulation is simplistic. Motion planning in autonomous vehicles has been explored by incorporating actuator dynamics, nonholonomic constraints and collision avoidance to generate local trajectories [59]. This approach only formulates its trajectory to terminate at a desired local goal position instead of tracking a full reference path. Another paper uses a potential field method in a human-like approach to autonomous vehicle decision making, which avoids following a predefined path [60]. Instead, conditions like obstacle avoidance and maintaining safe lane driving are prioritized for MPC path planning while making lane change decisions based on a game theory framework.

Many studies have sought to extend the use of MPC with adaptations and additional steps. A model-free predictive controller is developed, removing the need for model learning, which can sometimes be complex & laborious [61]. This proposed method only holds for a short preview horizon until accuracy fades, meaning it is inapplicable to path planning tasks such as collision avoidance. Tubes are introduced in MPC to handle uncertainty and provide a robust, collision-free region around the nominal path [62]. Further research has focused on reducing computational costs through geometric reformulations [63]. Although real-time applicability has improved for nonlinear solvers, attempts have been made at linearizing certain nonlinear MPC problems. Depending on the situation, these approximations may not significantly deteriorate performance while reducing computational costs [18]. Reinforcement learning has been paired with MPC both in a hierarchical setup [64] and using MPC in training. When properly trained, the reinforcement learning method is shown to decrease computational time while maintaining similar paths to MPC in simple, simulated environments [65].

Applications of varying schemes of model predictive control to autonomous vehicle racing are also common in existing literature. One approach extracts a local map from sensor data before estimating the track center line based on curved walls constructed by consecutive obstacles (assuming constant track width) [66]. This method achieves comparable performance to global methods but with higher computation times. Model Predictive Contouring Control (MPCC) is used here and in other racing studies [67], implementing linear approximations and a contraction constraint to ensure stability while attempting to both minimize the center line contour following error and maximize vehicle speed [68]. Alternatively, a sampling-based optimization approach, Model Predictive Path Integral (MPPI) control, makes parallel samples in the control space, removing the need for separate planning and execution steps [69].

By guiding the convergence target of the solution, an extension enhances performance for racing when multimodalities exist in the optimal action distribution [70].

2.3 Dynamic Obstacle Detection & Tracking

Assuming static environments for path planning may be insufficient for many real-world applications. Thus, the need for real-time dynamic obstacle recognition and tracking must be incorporated for versatile path planners. Vehicle detection and object classification have been prominent subjects of focus due to modern developments in machine learning and, more recently, deep learning [71]. An early approach used a Gaussian mixture model (GMM) to determine if pixels were part of the background or not, attempting to handle lighting and seasonal changes [72]. Thereafter, machine learning methods have been designed to use known properties for vehicle detection and classification from RGB images. Shape- or edge-based methods identify vehicles from their outlines in contrast to the background [73], while appearance-based techniques perform classification after training on corresponding datasets [74].

Most vehicle detection used presently takes the form of deep learning, where neural networks make class predictions based on learned visual features [75]. Two types are Faster Region-based Convolutional Neural Networks (Faster R-CNNs) [76] and You Only Look Once (YOLO) networks [77]. YOLO splits an image into grid cells and determines bounding boxes for predicted detection classes all in one pass of the neural network, making it popular in real-time detection/classification applications [78–80]. Deep learning methods are also applied to enhance the use of Doppler radar in detection and tracking applications [81, 82].

Once moving obstacles are detected, positional and orientational states must be tracked to direct path planners and avoid collisions. A classic approach that recursively estimates states with updated observation knowledge is known as Kalman filtering [83]. Noisy measurements are filtered, and a time-varying state estimate allows for tracking given uncertainties [84, 85]. This method is broadened to the Extended Kalman Filter (EKF) by performing linearizations on non-linear systems, which are common in autonomous vehicle tracking problems [86]. EKFs can then estimate target positions and velocities to predict future motion for either tracking [87] or collision avoidance purposes [88]. Although some detection methods, like those using CNNs, can be susceptible to occasional missing frames, state estimation remains effective using Kalman filtering [89].

Path planning with tracked dynamic targets becomes a more challenging problem than in the static case. One extension of DWA considers time such that trajectories of dynamic obstacles aren't crossed. Going further, it introduces a tree formulation where future curvature arcs are considered over a horizon to choose the best velocity pair to apply at the current time [42]. Using a constant velocity model for dynamic obstacles detected from a local costmap, TEB is extended to penalize close distances to time-sampled moving objects [90]. Similarly, an MPC planner parameterizes moving obstacles by fixed, blocked regions at sampled times to maintain distance and prevent collisions [15]. In a racing application, vehicle overtaking is achieved through nonlinear MPC, where overtaking is enacted only if the maneuver has a low risk of collision [91]. In application, sensor fusion of RGB-D (RGB-Depth) cameras with LiDAR has enabled path tracking of dynamic obstacles in 3D space [92, 93].

2.4 Bezier Curve Trajectory Formulations

The Bezier curve is an important tool for designing smooth, efficient trajectories not only in path planning but also in computer graphics and animation [94]. Bezier curves are based on Bernstein polynomials [95], where a few control points can fully characterize a parametric curve with fixed start and end coordinates. Figure 2.2 shows the increased flexibility of Bezier curves with added control points, where each point effectively pulls & shapes the curve towards it. Bezier curves are effective in smoothing predefined, piecewise linear paths and can also be used for collision avoidance and path planning.

The most prevalent use of Bezier curves in the field of autonomous vehicles and mobile robotics is for curve interpolation and smoothing [96]. In one study, Dijkstra's algorithm selects the shortest path based on a Voronoi graph where Bezier curves are then applied to smooth the piecewise linear segments into a less rigid trajectory [97]. Quintic Bezier curves are used on piecewise linear paths to achieve C^2 (curvature) continuity, ensuring feasibility for autonomous vehicle travel before MPC is performed to select the optimal path velocity [98]. A further extension ensures C^3 continuity by using quintic trigonometric Bezier curves with two shape parameters to smooth a predefined skeleton path [99]. Similarly, other methods rely on waypoints to achieve smooth turns while also maintaining continuity between segments [100, 101]. A quartic Bezier curve formulation seeks to generate control points that provide adaptive speeds under different path curvatures without additionally computing the speed profile [102]. This method maintains continuity while following waypoints corresponding to the predefined path.

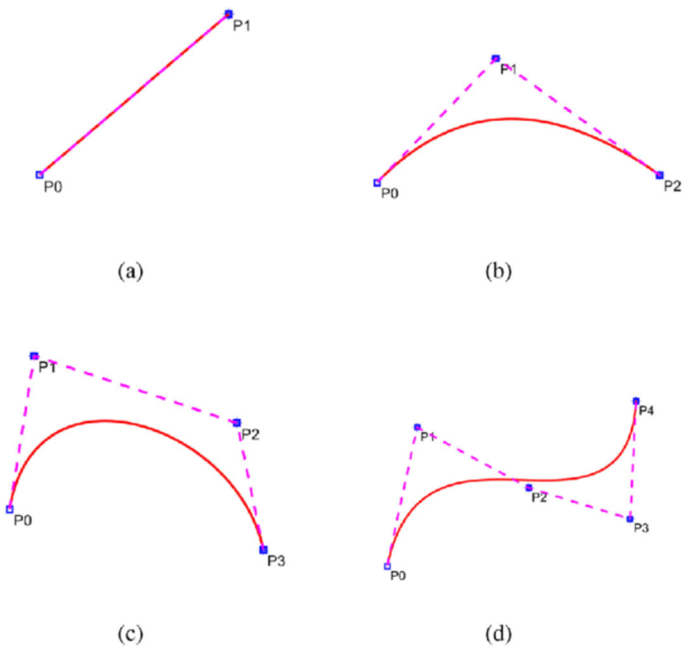


Figure 2.2: Bezier curves, control points and control polygons for each order curve. (a) Linear Bezier Curve (b) Quadratic Bezier Curve (c) Cubic Bezier Curve (d) Quartic Bezier Curve

Some approaches use Bezier curves more thoroughly for path generation as opposed to smoothing and interpolation. In a simulated autonomous vehicle study, obstacle avoidance is achieved while maintaining travel along waypoints of an initial path using quadratic programming and Hildreth's algorithm [103]. The Bezier method is shown to generate better paths than a separate potential field approach while also having faster computation times. Another paper evaluates a dynamic trajectory planning method for varying simulated road conditions based on potential fields to guide the Bezier curve's path [104]. Unknown parameters are reduced through the use of the Bezier curve, and vehicle dynamics direct the trajectory choice. Separate optimizations are done; first for the Bezier curve curvature and second for the speed and acceleration profiles.

Through a Bezier path generation algorithm, a study regarding mobile robots achieves local collision avoidance while pursuing a global path [105]. Control points are added to avert local obstacles while the global trajectory is formed through a Bezier curve interpolation of an RRT path. Lastly, genetic algorithms are augmented with Bezier curves to achieve security, minimal length and smoothness of the resulting path [106, 107]. Here, the genetic algorithm explores the space between a start and local goal point to select the control points that maximize the fitness function and thus produce the best trajectory.

2.5 Navigation in Pursuit & Fleet Formation Contexts

Some applications require cooperative navigation among agents, either via leader-follower schemes or extendable fleet configurations. Often, some form of communication exists between vehicles [1], whether it be directional (from leader to follower) [24] or bidirectional [108]. A common architecture is distributed MPC, where communication occurs between neighboring agents regarding positions and planned paths [22, 109]. This allows each agent to make their own decentralized decisions while providing this information to other agents, maintaining swarm intelligence. Sometimes, however, communication isn't possible, and each agent must operate using only its own sensor-based knowledge [110].

In leader-follower formations, information typically flows downstream such that the followers track the leader [111]. Research has explored the leader-follower hierarchy for teams of agents where the leader follows global planners like RRT [112], D*Lite [23] and Dijkstra's Algorithm [113]. The pursuing agents track the leader in the predetermined formation while potential field forces maintain shape and prevent collisions both with obstacles and between agents. In another application, a lead vehicle is tracked in an outdoor, off-road environment using LiDAR, RGB cameras and radar sensors [110]. The controlled vehicle pursues the tracked leader by solving an MPC problem through an interior point solver, maintaining a safe separating distance that increases when traveling at higher speeds.

By considering both maximum allowed distances between vehicles due to communication range requirements and minimum distances to prevent collisions, another study stacks the leader-follower approach to a successive string of robots [114]. The leader-follower framework can also be used in varying formations where N robots are controlled through $N-1$ decentralized leader-follower connections. This approach is used in a three-level control architecture where relative formation positions are subject to obstacle avoidance [115]. It is assumed that a high-level coordinator can reassign leader-follower connections and parameters, yielding varying formation patterns.

For more general, distributed MPC problems, agents consider interactions with all nearby agents instead of a single leader. To ensure agents don't collide or significantly interfere with each other when planning towards a goal, a centralized nonlinear MPC approach is investigated [116]. Deadlocks can occur when vehicles outside of the local communication range become close, presenting the risk of inter-vehicle collisions. The centralized optimization uses knowledge of all vehicles to ensure that vehicle paths maintain separating distances and presents collision-free navigation in simulation.

Conversely, a different method distributes the computational load of the optimal control problem between the agents and only performs one solution iteration per control update [117]. This saves computation & communication costs, and the solution converges as the vehicles head towards their destination. To enable smooth formation splitting and merging in the presence of obstacles and with communication loss, homotopy classes can be used to split agents into appropriate, adaptable groups [118].

Further studies on formation frameworks explore different governing rules and principles for collective navigation. Per one approach, optimality is redefined to determine the minimal energy paths for agents, locally or globally, in a decentralized manner [119]. Another paper explores a centralized planning layer that generates dynamic targets for each agent to track while using potential field forces between agents, keeping flexibility and safety [120]. A comprehensive framework achieves static path planning and dynamic tracking while being able to alter robot formation through inter-vehicle and vehicle-to-cloud communication channels [121]. Reinforcement learning has also been integrated into multi-vehicle navigation problems where training occurs over different formations, goal points and obstacle configurations [122, 123].

Chapter 3

Sequential Tracking Line Model Predictive Control

This chapter introduces the method entitled Sequential Tracking Line Model Predictive Control (STLMPC), which is expanded upon in additional chapters. This approach provides a safe, standalone local path planning technique that maximizes local distances to obstacles while conforming to constraints on vehicle dynamics. At each time step, quadratic programming is used to obtain successive ideal reference tracking lines, and a nonlinear MPC problem is solved through an SQP solver for the optimal steering angle commands to navigate.

The outline of the chapter is as follows. First, the formulation of the STLMPC algorithm is shown, including both the creation of tracking lines and the MPC optimization that ensues. The case of localization through Adaptive Monte Carlo Localization (AMCL) is shown, which allows for increases in the look-ahead horizon of STLMPC. The simulation and experimental architectures for this and all subsequent chapters are discussed, and results are provided for the tested STLMPC algorithm.

3.1 STLMPC Formulation

3.1.1 Generation of Tracking Lines

The unknown environment is constructed based on a set of detected obstacles (in practice, through a sensor such as LiDAR). Here, the N_{obs} obstacles, $\{(x_{obs,i}, y_{obs,i})\}_{i=0}^{N_{obs}-1}$ are denoted with respect to the moving vehicle frame, $T_{base} = [x_{base}, y_{base}, \theta_{base}]^T$. For each obstacle, the polar coordinates, $\{(r_{obs,i}, \theta_{obs,i})\}_{i=0}^{N_{obs}-1}$ can be obtained accordingly through $r_{obs,i} = \sqrt{x_{obs,i}^2 + y_{obs,i}^2}$ and $\theta_{obs,i} = \tan^{-1}(\frac{y_{obs,i}}{x_{obs,i}})$ while the reversion back to the Cartesian frame follows from the inverse equations. The proposed convention assigns the $+x$ direction to point ahead of the vehicle, $+y$ to point left and $+\theta$ to rotate counterclockwise (starting from 0 along the positive x axis) as seen in Figure 3.1.

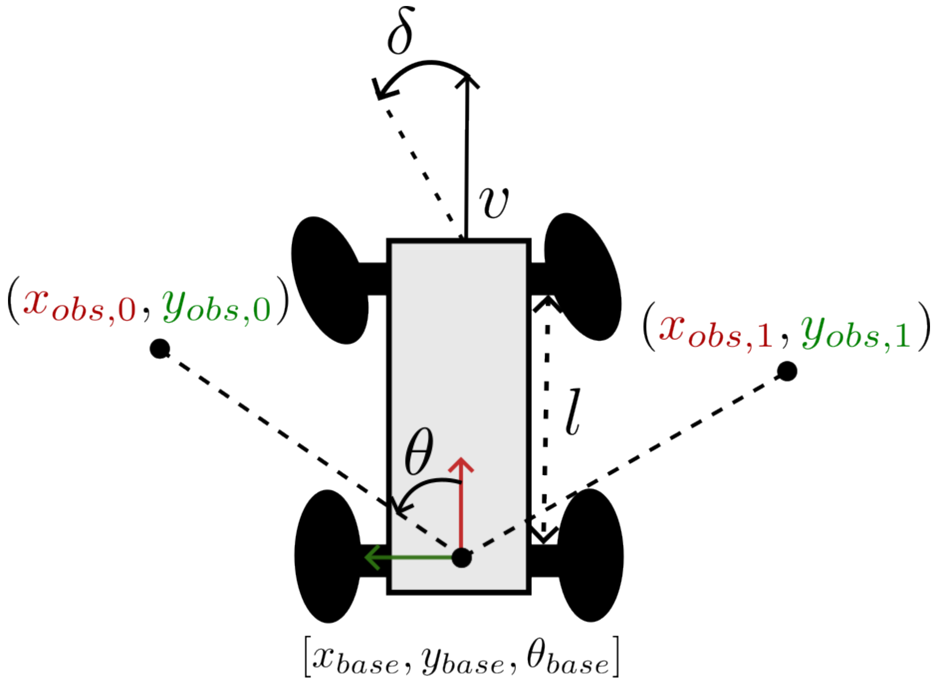


Figure 3.1: Vehicle states and coordinate vehicle frame with positive x (red) and y (green) directions shown. The coordinates of two obstacles are shown with respect to the base frame.

The vehicle state is described by five variables, $[x, y, \theta, \delta, v]$ where x and y are the vehicle position coordinates, θ is the vehicle orientation, δ is the current steering/turning angle and v is the forward velocity. The fixed wheelbase parameter, l, describes the distance between the front and rear axles of a given vehicle.

To obtain the reference tracking line, a heading direction is first required. The approach taken, inspired by the Follow the Gap Method (FGM) [124], initially orders the N_{obs} obstacles in increasing order of $\theta_{obs,i}$ where for all $i \in \{0, ..., N_{obs} - 1\}$, $-\pi \leq \theta_{obs,i} < \pi$. A threshold distance is defined by d_{safe}, where all obstacles with $r_{obs,i} \leq d_{safe}$ are assumed to be immediate potential hazards. For all obstacles in the front π radian window ($-\frac{\pi}{2} \leq \theta_{obs,i} \leq \frac{\pi}{2}$), the largest range-weighted angular gap with only obstacles further than d_{safe} is found, providing the gap's start and end angles, θ_{start} & θ_{end} (Algorithm 1). The safest gap balances the angular size with larger obstacle ranges in the gap & exceeding d_{safe} to ensure the safest local direction of travel. Once the safest angular gap is obtained, the heading direction is calculated as:

$$\theta_{head} = \frac{\theta_{start} + \theta_{end}}{2} \tag{3.1.1}$$

Now, the obstacles are split into left, right and excluded clusters with respect to the heading angle. Obstacles with $\theta_{obs,i} \in [\theta_{head} + a_l, \theta_{head} + b_l]$ belong to the left cluster, $\mathcal{O}_l$, while obstacles satisfying $\theta_{obs,i} \in [\theta_{head} - b_r, \theta_{head} - a_r]$ fall into the right cluster, $\mathcal{O}_r$. The Cartesian forms, $\mathcal{O}_l = \{(x_{obs,i}, y_{obs,i})\}_{i=0}^{N_{left}-1}$ and $\mathcal{O}_r = \{(x_{obs,i}, y_{obs,i})\}_{i=0}^{N_{right}-1}$ are then used in a quadratic optimization to obtain the reference tracking line. Here, N_{left} and N_{right} denote the number of left and right obstacles, respectively. Obstacles can be subsampled to limit the number of constraints and, thereby, optimization complexity.

Algorithm 1: Safest Local Angular Gap Formulation

```
Input: Obstacle Ranges $\{r_{obs,i}\}$, Obstacle Angles $\{\theta_{obs,i}\}$
Output: $\theta_{start},\ \theta_{end}$
1  Sort obstacles by ascending angle;
2  LargestGap = 0, CurrentGap = 0, GapLength = 0;
3  for $\theta_{obs,i} \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ do
4      if $r_{obs,i} > d_{safe}$ then
5          CurrentGap += $r_{obs,i}\frac{\theta_{obs,i+1}-\theta_{obs,i-1}}{2}$;
6          if GapLength = 0 then
7              $\theta_{start\_curr} = \theta_{obs,i}$;
8          GapLength += 1;
9      if $r_{obs,i} \leq d_{safe}$ or $\theta_{obs,i+1} \notin [-\frac{\pi}{2}, \frac{\pi}{2}]$ then
10         if CurrentGap > LargestGap then
11             LargestGap = CurrentGap;
12             $\theta_{start} = \theta_{start\_curr}$;
13             $\theta_{end} = \theta_{obs,i-1}$;
14         CurrentGap = 0, GapLength = 0;
15 end
```

The basis for the tracking line optimization comes from the concept of Support Vector Machines (SVMs) [125]. SVM identifies a decision boundary (linear or non-linear) that best separates classes of points by maximizing the margin between them. This approach can be used for path planning that maximizes distance to obstacles and has been explored in global planning approaches to find paths between a start point and goal [126, 127].

The parallel right and left bounding lines on the obstacle clusters are described respectively (Figure 3.2) by:

$$w^T q + b = 1 \tag{3.1.2}$$

and

$$w^T q + b = -1 \tag{3.1.3}$$

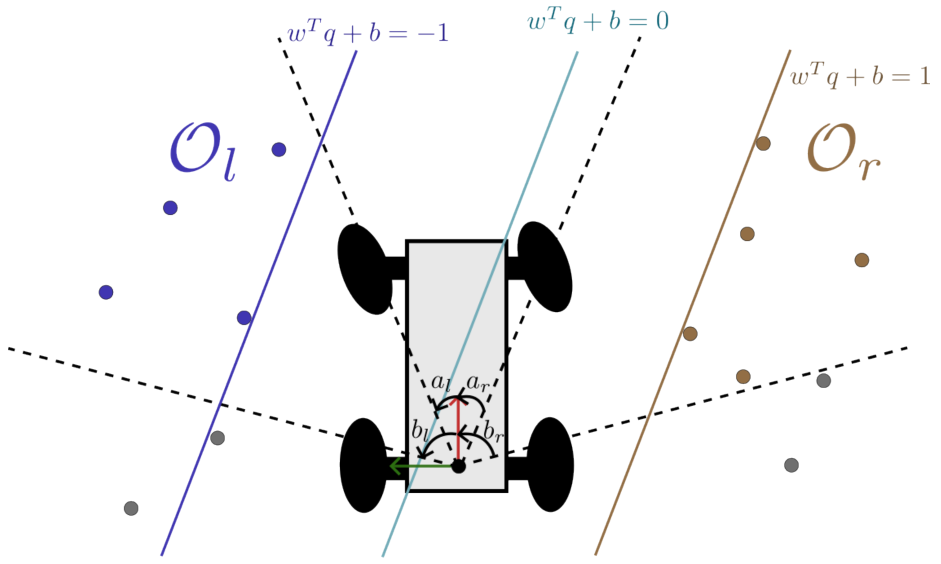


Figure 3.2: Left & right bounding lines for the defined obstacle clusters as well as the optimal center tracking line generated.

where $w^T = [w_x, w_y]$ and $q^T = [x, y]$. These parallel bounding lines define a consistent maximum margin and preserve symmetry, forming the basis for the convex optimization problem. The optimization variables are the line parameters, namely $x^T = [w_x, w_y, b]$. To maximize the distance between the lines, $d = \frac{2}{\sqrt{w^Tw}}$, the denominator is minimized via a standard quadratic objective. Linear inequality constraints are developed to ensure obstacles remain on the separated side of each bounding line. Finally, b is bounded to ensure that the (0,0) reference exists between the two bounding lines and thus the vehicle is in the middle of the bounded obstacle clusters. The quadratic optimization with linear inequality constraints is proposed:

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}w^Tw \\ \text{subject to} \quad & w^Tp_i + b - 1 \geq 0 \quad \forall p_i \in \mathcal{O}_r \\ & w^Tp_j + b + 1 \leq 0 \quad \forall p_j \in \mathcal{O}_l \\ & -1 + \epsilon \leq b \leq 1 - \epsilon \end{aligned} \tag{3.1.4}$$

which can be solved through Goldfarb & Idnani's active-set dual method [128]. Here, ϵ is assumed to be a small number, and in practice, a small quadratic objective term is placed on b to ensure the problem is strictly convex.

The tracking line becomes the center line between the bounding left and right lines, $w^T q + b = 0$, and through normalization, the tracking line parameters are reduced from three to two:

$$w^T q + 1 = 0, \text{ where we redefine } w^T \leftarrow \frac{1}{b}[w_x, w_y] \tag{3.1.5}$$

This process is now extended to the general case of n_{MPC} sequential tracking lines, each of k_{MPC} future samples (with sample time Δt). Using the current reference frame point, $p_{base,i} = [x_{base,i}, y_{base,i}]^T$, the closest point on the tracking line, $p_{start,i}$, is:

$$[x_{start,i}, y_{start,i}]^T = p_{base,i} - \frac{w_i^T p_{base,i} + 1}{\|w_i\|^2} \cdot w_i \tag{3.1.6}$$

Taking the orientation of the tracking line using the two-argument inverse tangent function, $\theta_{track,i} = -\text{atan2}(w_{x,i}, w_{y,i})$ and the vehicle's current velocity, v, the end point of the tracking line, $p_{end,i}$ is found through:

$$[x_{end,i}, y_{end,i}]^T = p_{start,i} + v\Delta t\, k_{MPC} \cdot [\cos(\theta_{track,i}), \sin(\theta_{track,i})]^T \tag{3.1.7}$$

This becomes the reference point for the next tracking line, $p_{base,i+1} = p_{end,i}$. All obstacles are transformed from the original reference, $p_{base,0}$ (with obstacles denoted $\{(x^{(0)}_{obs,j}, y^{(0)}_{obs,j})\}_{j=0}^{N_{obs}-1}$) to the new reference through:

$$\begin{bmatrix} x_{obs,j}^{(i+1)} \\ y_{obs,j}^{(i+1)} \end{bmatrix} = \begin{bmatrix} \cos(\theta_{track,i}) & \sin(\theta_{track,i}) \\ -\sin(\theta_{track,i}) & \cos(\theta_{track,i}) \end{bmatrix} \cdot \begin{bmatrix} x_{obs,j}^{(0)} - x_{end,i} \\ y_{obs,j}^{(0)} - y_{end,i} \end{bmatrix}, \quad j \in \{0, ..., N_{obs} - 1\} \tag{3.1.8}$$

where the process then repeats, starting from finding the best heading direction. After each sequential tracking line, w_i (for $i \in \{0, ..., n_{MPC} - 1\}$) is solved for (Equations 3.1.4 & 3.1.5) in its corresponding frame of reference, a normalized transformation is done back to the initial base frame for ensuing Equations 3.1.6 & 3.1.7:

$$\begin{bmatrix} w_{x,i} \\ w_{y,i} \end{bmatrix} \leftarrow A \begin{bmatrix} \cos(\theta_{track,i-1}) & -\sin(\theta_{track,i-1}) \\ \sin(\theta_{track,i-1}) & \cos(\theta_{track,i-1}) \end{bmatrix} \cdot \begin{bmatrix} w_{x,i} \\ w_{y,i} \end{bmatrix} \tag{3.1.9}$$

with the normalization term,

$$A = \frac{1}{1 - \begin{bmatrix} w_{x,i} & w_{y,i} \end{bmatrix} \cdot \begin{bmatrix} \cos(\theta_{track,i-1}) & \sin(\theta_{track,i-1}) \\ -\sin(\theta_{track,i-1}) & \cos(\theta_{track,i-1}) \end{bmatrix} \cdot \begin{bmatrix} x_{end,i-1} \\ y_{end,i-1} \end{bmatrix}} \tag{3.1.10}$$

The parameters, w_i, for these n_{MPC} tracking lines (Figure 3.3) are next used in the MPC optimization formulation.

3.1.2 MPC Optimization - Objective

The non-linear MPC optimization follows from tracking the ideal reference path subject to constraints on vehicle dynamics. In this STLMPC formulation, the vehicle states are reduced to $[x, y, \theta, \delta]$ where v is assumed constant over the future trajectory. The control inputs, δ_i, determine the turning behavior of the vehicle and thus the predicted path, which is represented by the x_i, y_i & θ_i states.

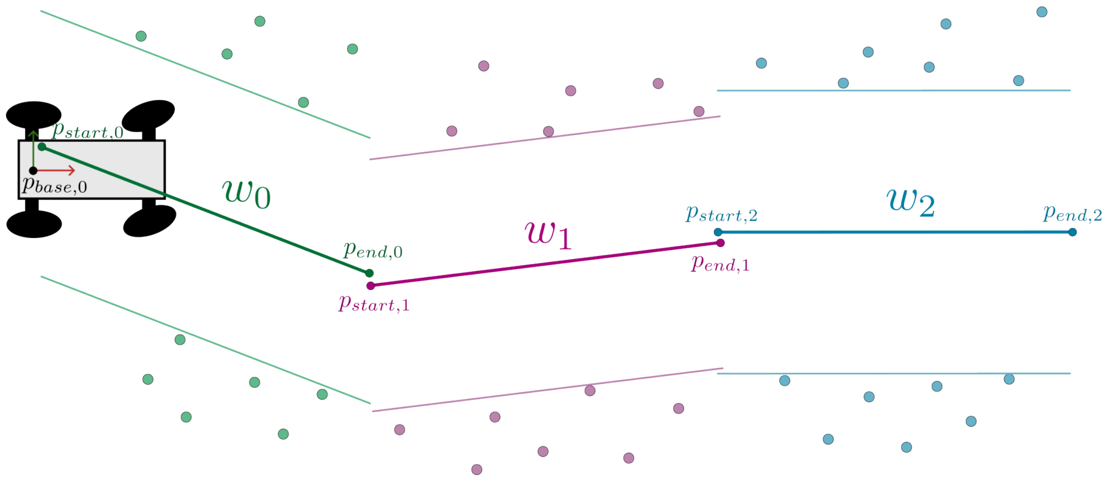


Figure 3.3: Sequential tracking lines with bounding lines and respective obstacles for $n_{MPC} = 3$. Tracking line start and end points are denoted accordingly with respect to the initial base frame.

The MPC horizon length is based on the number of sequential tracking lines generated, where n_{MPC} lines, each tracked for k_{MPC} samples, yield a horizon length of $n_{MPC}k_{MPC}$ samples. For the given state variables, discretized over the future horizon samples, $4n_{MPC}k_{MPC}$ total optimization variables are used. The multi-term objective is constructed to balance proximity to the ideal piecewise linear path with a heading direction aligned along the path as well as the control effort expended over the horizon. The form of the objective function is therefore:

$$F_{obj}(\delta_i) = \lambda_{d^2}F_{d^2}(\delta_i) + \lambda_{\dot{d}^2}F_{\dot{d}^2}(\delta_i) + \lambda_{\delta^2}F_{\delta^2}(\delta_i) \tag{3.1.11}$$

where the $\lambda = [\lambda_{d^2}, \lambda_{\dot{d}^2}, \lambda_{\delta^2}]$ weights determine the relative significance of each objective term, and normalization by one weight reduces the number of weights by one.

The measure of proximity used between the vehicle and the piecewise path is a summation of squared Euclidean distances along the prediction horizon. This ensures a path that maintains closeness to the reference over the full trajectory while also having a convenient quadratic cost. Here, each vehicle position is measured against the corresponding tracking line, $\tilde{w}_i$, at that sample time:

$$\tilde{w}_i = w_j \quad \text{for} \quad jk_{MPC} \leq i < (j+1)k_{MPC} \tag{3.1.12}$$

which produces the objective term:

$$F_{d^2}(\delta_i) = \sum_{i=0}^{n_{MPC}k_{MPC}-1} \frac{(\tilde{w}_i^T P_i + 1)^2}{\|\tilde{w}_i\|^2} \tag{3.1.13}$$

where $P_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}$ represents the vehicle positional state at the i^{th} time sample.

The $F_{\dot{d}^2}(\delta_i)$ term denotes the cost placed on the vehicle's orientation with respect to the tracking lines, which is minimized when the predicted trajectory and reference path are parallel. In practice, this term minimizes oscillations and discourages paths that turn aggressively towards the reference path. The time derivative for each sample's distance between vehicle and tracking line is found, squared and summed to get a cost of similar form to Equation 3.1.13. This follows from:

$$\sum_{i=0}^{n_{MPC}k_{MPC}-1} d_i = \sum_{i=0}^{n_{MPC}k_{MPC}-1} \frac{(\tilde{w}_i^T P_i + 1)}{\|\tilde{w}_i\|} \tag{3.1.14}$$

where:

$$\sum_{i=0}^{n_{MPC}k_{MPC}-1} \dot{d}_i = \frac{d}{dt} \sum_{i=0}^{n_{MPC}k_{MPC}-1} \frac{(\tilde{w}_i^T P_i + 1)}{\|\tilde{w}_i\|} = \sum_{i=0}^{n_{MPC}k_{MPC}-1} \frac{(\tilde{w}_i^T \dot{P}_i)}{\|\tilde{w}_i\|} \tag{3.1.15}$$

assuming only P_i changes over time and disregarding the discontinuous jumps in the otherwise constant $\tilde{w}_i$ when the next tracking line is followed. Then:

$$F_{\dot{d}^2}(\delta_i) = \sum_{i=0}^{n_{MPC}k_{MPC}-2} \dot{d}_i^2 = \sum_{i=0}^{n_{MPC}k_{MPC}-2} \frac{(\tilde{w}_i^T \dot{P}_i)^2}{\|\tilde{w}_i\|^2} \tag{3.1.16}$$

and since $\dot{P}_i = \frac{1}{\Delta t}\begin{bmatrix} x_{i+1} - x_i \\ y_{i+1} - y_i \end{bmatrix}$ is obtained via forward differencing, the last term of the summation has no forward difference and is discarded, leading to a sum over $i \in \{0, ..., n_{MPC}k_{MPC} - 2\}$.

The final term captures the cost of control effort over the future horizon. Control inputs consist of the steering angles, δ_i, assuming the constant velocity trajectory. The magnitudes are minimized to reduce control effort and unnecessary oscillations in the predicted path, promoting stability. Again, the objective term takes on the form of a summation of squares through:

$$F_{\delta^2}(\delta_i) = \sum_{i=0}^{n_{MPC}k_{MPC}-1} \delta_i^2 \tag{3.1.17}$$

The objective function is thus a sum of three quadratic terms; the first two are functions of position (as well as orientation and δ_i indirectly), and the third is a function of steering angle. The quadratic construction is well-suited for standard non-convex solvers (non-convexities arise in the constraints), such as SQP-based methods like SLSQP, which is implemented. The objective's analytical gradients with respect to each state variable are provided for reference in Appendix A.1 and are used in the gradient-based SLSQP solver [129] for proper numerical conditioning & stability.

3.1.3 MPC Optimization - Constraints

In order to minimize the prior objective function, several equality (denoted by h) & inequality (denoted by g) constraints must be satisfied relating to the vehicle kinematics and control input limits. The initial vehicle position and orientation for the future path are fixed to start at the initial base frame. Therefore, since all parameters of the future path are taken with respect to T_{base}, the initial state values correspond to $x_0 = 0$, $y_0 = 0$ & $\theta_0 = 0$.

From here, the vehicle dynamics follow from the kinematic bicycle model [130] under the assumption of no slip. This commonly used approach simplifies the vehicle's motion to that of a single-track/bicycle model with front-wheel steering, which holds for reasonably low tire slip angles and lateral accelerations [131]. Under this model, the positional arguments vary over the future trajectory by the equations:

$$x_{i+1} = x_i + \Delta t\, v_i \cos(\theta_i + \beta_i) \quad \forall i \in \{0, ..., n_{MPC}k_{MPC} - 2\} \tag{3.1.18}$$

and

$$y_{i+1} = y_i + \Delta t\, v_i \sin(\theta_i + \beta_i) \quad \forall i \in \{0, ..., n_{MPC}k_{MPC} - 2\} \tag{3.1.19}$$

where β_i represents the vehicle slip angle at sample i. Simplifying by assuming v is constant and $\beta_i = 0$ for all i yields $h_{x,i}$ and $h_{y,i}$ through:

$$\begin{aligned} x_{i+1} &= x_i + \Delta t\, v \cos(\theta_i) \quad \forall i \in \{0, ..., n_{MPC}k_{MPC} - 2\} \\ h_{x,i} &= x_{i+1} - x_i - \Delta t\, v \cos(\theta_i), \quad h_{x,i} = 0 \end{aligned} \tag{3.1.20}$$

and

$$\begin{aligned} y_{i+1} &= y_i + \Delta t\, v \sin(\theta_i) \quad \forall i \in \{0, ..., n_{MPC}k_{MPC} - 2\} \\ h_{y,i} &= y_{i+1} - y_i - \Delta t\, v \sin(\theta_i), \quad h_{y,i} = 0 \end{aligned} \tag{3.1.21}$$

which represent $2(n_{MPC}k_{MPC} - 1)$ equality constraints. The vehicle orientation evolves over the trajectory according to the model according to:

$$\theta_{i+1} = \theta_i + \Delta t \frac{v_i \cos(\beta_i)}{l}(\tan(\delta_{f,i}) - \tan(\delta_{r,i})) \quad \forall i \in \{0, ..., n_{MPC}k_{MPC} - 2\} \tag{3.1.22}$$

with fixed wheelbase, l as well as front ($\delta_{f,i}$) & rear ($\delta_{r,i}$) steering angles. Again, the slip angle is set to 0 and v is constant, while front-wheel steering means $\delta_{r,i} = 0$ for all i. Taking $\delta_i = \delta_{f,i}$ as the control inputs generates $h_{\theta,i}$:

$$\begin{aligned} \theta_{i+1} &= \theta_i + \Delta t \frac{v}{l} \tan(\delta_i) \quad \forall i \in \{0, ..., n_{MPC}k_{MPC} - 2\} \\ h_{\theta,i} &= \theta_{i+1} - \theta_i - \Delta t \frac{v}{l} \tan(\delta_i), \quad h_{\theta,i} = 0 \end{aligned} \tag{3.1.23}$$

for an additional $n_{MPC}k_{MPC} - 1$ equality constraints. The non-convex equality constraints, $h_{x,i}, h_{y,i} \,\&\, h_{\theta,i}$, described in Equations 3.1.20, 3.1.21 & 3.1.23 importantly reflect the non-linear vehicle dynamics and therefore cause the optimization to be non-convex.

Now, the control input bounds are expressed based on the physical limitations of the vehicle. First, the steering angles are bounded in magnitude by the servo motor or steering system's maximum limits:

$$-\delta_{max} \leq \delta_i \leq \delta_{max} \quad \forall i \in \{0, ..., n_{MPC}k_{MPC} - 1\} \tag{3.1.24}$$

where δ_{max} denotes the largest allowable steering angle magnitude in radians, applicable to both left (+) & right (-) turning angles. Another restriction is on the steering angle's maximum rate of change, $\Delta\delta_{max}$, in both the left and right directions. Again, influenced by the physical limits on the steering method, the inequality constraints, $g_{\delta^+,i}$ & $g_{\delta^-,i}$, follow from:

$$-\Delta t\,\Delta\delta_{max} \leq \delta_{i+1} - \delta_i \leq \Delta t\,\Delta\delta_{max} \quad \forall i \in \{0, ..., n_{MPC}k_{MPC} - 2\}$$

$$g_{\delta^+,i} = \delta_{i+1} - \delta_i - \Delta t\,\Delta\delta_{max}, \quad g_{\delta^+,i} \leq 0 \tag{3.1.25}$$

$$g_{\delta^-,i} = -\delta_{i+1} + \delta_i - \Delta t\,\Delta\delta_{max}, \quad g_{\delta^-,i} \leq 0 \tag{3.1.26}$$

where forward differencing means only $2(n_{MPC}k_{MPC} - 1)$ inequality constraints ensue.

Lastly, the initial steering angle is fixed based on the previous time sample's control input. Since the servo motor or steering mechanism used requires time to transition from the last to a new steering angle, the control input is applied a sample in advance. This ensures that by the next time sample, the steering angle will be equal to the desired value according to the optimized path. In practice, this approach minimizes oscillations and improves stability as well as the ability of the vehicle to follow predicted paths. This corresponds to:

$$\delta_0 = \delta_{last} \tag{3.1.27}$$

where δ_{last} was the control input applied based on the previous step's MPC optimization. The steering angle applied every control step is $\delta_{cmd} = \delta_1$, the first control input over the predicted trajectory that is not fixed. Analytical gradients for the constraints with respect to optimization variables are provided for reference in Appendix A.1.

3.1.4 STLMPC Algorithm Pseudocode

Based on the constructed objective and constraint functions, the STLMPC optimization problem takes the form:

$$\begin{aligned} \min_{x_j, y_j, \theta_j, \delta_j} \quad & \lambda_{d^2} F_{d^2} + \lambda_{\dot{d}^2} F_{\dot{d}^2} + \lambda_{\delta^2} F_{\delta^2} \\ \text{subject to} \quad & g_{\delta^+,i} \leq 0, \quad g_{\delta^-,i} \leq 0 \\ & h_{x,i} = 0, \quad h_{y,i} = 0, \quad h_{\theta,i} = 0 \\ & -\delta_{max} \leq \delta_j \leq \delta_{max} \\ & x_0 = 0, \quad y_0 = 0, \quad \theta_0 = 0, \quad \delta_0 = \delta_{last} \\ & \forall i \in \{0, ..., n_{MPC} k_{MPC} - 2\} \\ & \forall j \in \{0, ..., n_{MPC} k_{MPC} - 1\} \end{aligned} \tag{3.1.28}$$

which contains quadratic objective terms, non-linear equality and linear inequality constraints as well as fixed initial state variables and control input. This non-convex problem is solvable through non-linear solvers, where high-quality local optimum solutions are found through a feasible starting guess in the region of the global optimum. The choice of an effective starting guess is significant to ensure both convergence and that the obtained local optimum achieves a low objective value, similar to the global optimum. Algorithm 2 indicates the process for obtaining effective starting guesses.

Here, $\tilde{\theta}_{track,i}$ denotes the orientations of the tracking lines, $\tilde{w}_i$, in the initial base frame. The first steering angle, δ_0, is fixed based on the control input applied in the previous timestep, where all successive δ_i values come from aligning the vehicle orientation, θ_i, to that of the tracking line, $\tilde{\theta}_{track,i}$. The limits on change in steering angle and bounds on magnitude are considered when finding successive control inputs, and after each future sample, the remaining states, $x_i, y_i \,\&\, \theta_i$ are updated based on the kinematic bicycle model. As a result, the starting guess for the optimization variables is found, which seeks to align the predicted path's orientation with that of the sequential tracking lines.

Algorithm 2: Formulation of Starting Guess for STLMPC Optimization

```
   Input: Tracking Line Orientations ($\tilde{\theta}_{track,i}$), Last Steering Angle ($\delta_{last}$)
   Output: $x_i, y_i, \theta_i, \delta_i$
1  for $j = 0$ to $n_{MPC} - 1$ do
2      for $i = 0$ to $k_{MPC} - 1$ do
3          if $j = 0$ and $i = 0$ then
4              $\delta_{i+jk_{MPC}} = \delta_{last}$, $x_{i+jk_{MPC}} = 0$, $y_{i+jk_{MPC}} = 0$, $\theta_{i+jk_{MPC}} = 0$;
5          else
6              $\delta_{track} = \tan^{-1}(\frac{l}{\Delta t\, v}(\tilde{\theta}_{track,i+jk_{MPC}} - \theta_{i+jk_{MPC}}))$;
7              if $\theta_{i+jk_{MPC}} < \tilde{\theta}_{track,i+jk_{MPC}}$ then
8                  $\delta_{i+jk_{MPC}} = \min(\delta_{track}, \delta_{i+jk_{MPC}-1} + \Delta t\, \Delta\delta_{max}, \delta_{max})$;
9              else if $\theta_{i+jk_{MPC}} > \tilde{\theta}_{track,i+jk_{MPC}}$ then
10                 $\delta_{i+jk_{MPC}} = \max(\delta_{track}, \delta_{i+jk_{MPC}-1} - \Delta t\, \Delta\delta_{max}, -\delta_{max})$;
11             $x_{i+jk_{MPC}} = x_{i+jk_{MPC}-1} + \Delta t\, v \cos\theta_{i+jk_{MPC}-1}$;
12             $y_{i+jk_{MPC}} = y_{i+jk_{MPC}-1} + \Delta t\, v \sin\theta_{i+jk_{MPC}-1}$;
13         if $i < k_{MPC} - 1$ or $j < n_{MPC} - 1$ then
14             $\theta_{i+jk_{MPC}+1} = \theta_{i+jk_{MPC}} + \Delta t \frac{v}{l} \tan\delta_{i+jk_{MPC}}$;
15     end
16 end
```

The SLSQP solver has two termination conditions set in practice to ensure the real-time control rate is maintained. One condition ensures that when the relative change in the optimization variables (using an L_1 norm) decreases below a certain threshold, the optimization concludes. This is because small changes in the solver's optimization steps indicate it has arrived at a local optimum, and further progress is likely to be limited. Alternatively, a maximum optimization time is set such that if the solver times out, the best feasible solution up to that point is returned and used. In the unlikely case of optimization failure, the previous control inputs are maintained until the next control step, where the STLMPC algorithm conducts a new optimization with the updated data.

The complete structure of the STLMPC algorithm is shown in Algorithm 3. This process follows the outlined steps in this chapter, generating n_{MPC} tracking lines through quadratic optimizations before performing the MPC optimization for the next steering angle command to apply. By updating the reference frame to the end of each tracking line, future navigation decisions are made with respect to forward positions while the path is mapped back to the base frame. Sampling each of the n_{MPC} tracking lines for k_{MPC} samples provides the path to track for each timestep, and an effective starting guess ensures a high-quality solution is obtained. The vehicle commands, δ_{cmd} & $v_{cmd} = v$, are applied to maintain autonomous driving, and the process repeats at the next control step.

Algorithm 3: STLMPC Algorithm Framework

```
   Input: Obstacle Ranges $\{r^{(0)}_{obs,i_{obs}}\}$, Obstacle Angles $\{\theta^{(0)}_{obs,i_{obs}}\}$, Last Steering
   Angle ($\delta_{last}$), Constant Velocity ($v$)
   Output: Steering Angle Command ($\delta_{cmd}$), Constant Velocity ($v$)
1  $p_{base,0} = [0,0]^T$, $\theta_{track,-1} = 0$, $i_{obs} \in \{0, ..., N_{obs}-1\}$, $i \in \{0, ..., n_{MPC}k_{MPC}-1\}$;
2  $\{(x^{(0)}_{obs,i_{obs}}, y^{(0)}_{obs,i_{obs}})\} \leftarrow \{(r^{(0)}_{obs,i_{obs}}, \theta^{(0)}_{obs,i_{obs}})\}$;
3  for $j = 0$ to $n_{MPC} - 1$ do
4      $\theta_{head} \leftarrow$ SafestAngularGap($\{(r^{(j)}_{obs,i_{obs}}, \theta^{(j)}_{obs,i_{obs}})\}$);
5      $\mathcal{O}^{(j)}_l, \mathcal{O}^{(j)}_r \leftarrow$ FindObstacleClusters($\theta_{head}, \{(x^{(j)}_{obs,i_{obs}}, y^{(j)}_{obs,i_{obs}})\}$);
6      $w_j \leftarrow$ SolveTrackingLineQP($\mathcal{O}^{(j)}_l, \mathcal{O}^{(j)}_r$);
7      $w_j \leftarrow$ TransformtoBaseFrame($w_j, p_{base,j}, \theta_{track,j-1}$);
8      $p_{base,j+1}, \theta_{track,j} \leftarrow$ getNextReferenceFrame($p_{base,j}, w_j, v$);
9      $\{(x^{(j+1)}_{obs,i_{obs}}, y^{(j+1)}_{obs,i_{obs}})\} \leftarrow$
         NextFrameObstacles($p_{base,j+1}, \theta_{track,j}, \{(x^{(0)}_{obs,i_{obs}}, y^{(0)}_{obs,i_{obs}})\}$);
10 end
11 $\tilde{w}_i \leftarrow$ SampledTrackingLines($w_j$);
12 $x_i, y_i, \theta_i, \delta_i \leftarrow$ MakeStartingGuess($\tilde{w}_i, \delta_{last}$);
13 $x_i, y_i, \theta_i, \delta_i, \delta_{cmd} \leftarrow$ SLSQP_Opt($x_i, y_i, \theta_i, \delta_i, v, \tilde{w}_i$);
14 Apply $\delta_{cmd}, v_{cmd} = v$ and repeat process for next control step;
```

3.2 STLMPC using Localization

The standard STLMPC algorithm presented can be extended in the case of localization to an existing occupancy grid. The context of local planning remains the same in that a goal location is not known, and global planning is not done. However, now the observation range is extended such that obstacles are seen even further in advance, enabling more accurate future tracking lines and path planning decisions. Furthermore, imperfect sensor data can miss certain detections and obstacles, but through localization, a complete grid map can augment detections for more complete coverage.

The Adaptive Monte Carlo Localization (AMCL) framework uses a particle filter to estimate the robot's moving position in a known map [132]. The likelihoods of sampled states being the true robot position are updated when new sensor data is received, while the sample with the highest weight is estimated as the state. Adaptive sampling ensures fewer samples are used when there is high confidence in the estimate and more samples are used when globally localizing with low confidence [133]. Localization performs best and an accurate estimate is obtained quickest when the approximate initial vehicle position is known; however, localization is also possible with an unknown initial state.

Through AMCL, an updated effective transform (two successive transforms in practice since odometry is used, T_{odom}) is obtained between the fixed known map and the moving vehicle base frame. The AMCL transform that represents the localized vehicle pose in the fixed map frame is given by $T_{AMCL} = [x_{AMCL}, y_{AMCL}, \theta_{AMCL}]^T$ where obstacles in the fixed map are converted to the vehicle's moving frame through:

$$\begin{bmatrix} x^{(0)}_{obs_{map},i} \\ y^{(0)}_{obs_{map},i} \end{bmatrix} = \begin{bmatrix} \cos(\theta_{AMCL}) & \sin(\theta_{AMCL}) \\ -\sin(\theta_{AMCL}) & \cos(\theta_{AMCL}) \end{bmatrix} \cdot \begin{bmatrix} x_{map,i} - x_{AMCL} \\ y_{map,i} - y_{AMCL} \end{bmatrix},\ i \in \{0, ..., N_{map} - 1\} \tag{3.2.1}$$

The map obstacles can be subsampled to obtain N_{map} points, maintaining computational tractability. Moreover, only map obstacles with $r^{(0)}_{obs_{map},i} < d_{max}$ are used in optimization, ensuring that non-immediate obstacles (far from the vehicle) are ignored. For future tracking line frames of reference, a similar transform to Equation 3.1.8 is performed where now all obstacles, $\{(x^{(j)}_{obs_{tot}}, y^{(j)}_{obs_{tot}})\}$ are considered:

$$\{(x^{(j)}_{obs_{tot}}, y^{(j)}_{obs_{tot}})\} = \{\{(x^{(j)}_{obs,i}, y^{(j)}_{obs,i})\}_{i=0}^{N_{obs}-1}, \{(x^{(j)}_{obs_{map},i}, y^{(j)}_{obs_{map},i})\}_{i=0}^{N_{map}-1}\} \tag{3.2.2}$$

From here, the same steps as before proceed for the tracking line and non-linear MPC optimizations, except with added and more complete obstacle coverage. The case of navigation using localization to a known map is shown in Figure 3.4, where the selection of future tracking lines is aided by the increased map obstacle coverage ahead of the vehicle. As shown, the maximum map obstacle distance, d_{max}, exceeds the LiDAR sensor range, d_{LIDAR}.

Thus, while the sensor cannot detect past the obstacle directly ahead, using the map data ensures effective, extended path planning decisions beyond this look-ahead distance. Also, coverage is improved even within the sensor range, as obscured areas that the sensor cannot detect are now known from the map. In the results to come, the performance of STLMPC when using AMCL is compared to the case without, and the look-ahead range for effective path planning is contrasted.

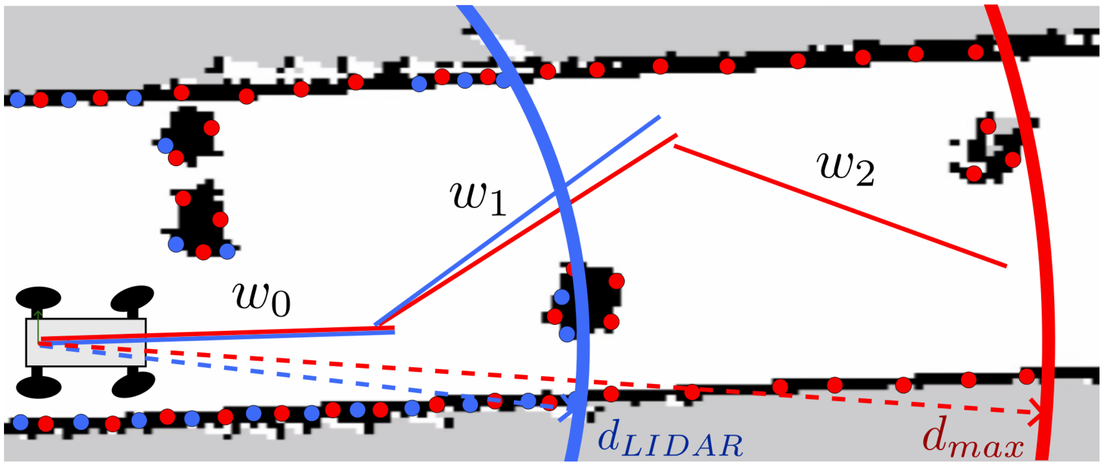


Figure 3.4: Detected obstacles, range and reference tracking lines for navigation with only sensor (blue) as well as with AMCL (red). Longer-term path planning is possible through AMCL, and obstacle detection coverage is more complete.

3.3 System Architecture

For testing and implementation of the STLMPC algorithm as well as work in future chapters, both simulation and experimental environments are used. For simulation purposes, testing is done in the *f1tenth_simulator* environment [134] with varying maps while visualization is done using RViz. The vehicle's sensors are replaced by passing map information to the vehicle, such as in the form of laser scans. Detections of other vehicles are represented by known simulated vehicle positions and orientations. Testing in the *f1tenth_simulator* allows for performance evaluation in an effectively ideal environment before the algorithm is ported to the physical vehicle.

The software stack is implemented in C++ & Python using ROS Noetic, the final ROS 1 distribution, on Ubuntu 20.04.6 LTS with Jetpack 5.1.4. Navigation and path planning are done in a C++ node, whereas inference for vehicle detection is performed by a Python node using NVIDIA TensorRT. The tracking line quadratic optimizations are done using the C++ library, *QuadProg++*, while the MPC non-linear optimizations use the NLOPT library, specifically NLOPT's SLSQP solver [129]. For wireless interfacing between the target machine and a laptop (Intel i7-8565U, 8GB RAM) for development and testing, NoMachine was used. Remote access to the target machine was established over a 2.4GHz wireless connection using the laptop's mobile hotspot.

For experimental purposes, McMaster University's MacAEV (Figure 3.5) is used, which is built on a 1/10th scale RC vehicle platform [135]. The NVIDIA Jetson AGX Orin serves as the onboard processing unit featuring a 12-core ARM Cortex-A78AE CPU, a 2048-core NVIDIA Ampere GPU and 64GB of LPDDR5 RAM. For LiDAR, the RPLIDAR A2M8 laser scanner has a range of 12m with a rotational speed of $\sim$10 Hz (the control rate for navigation) and an angular resolution $< 0.5°$.

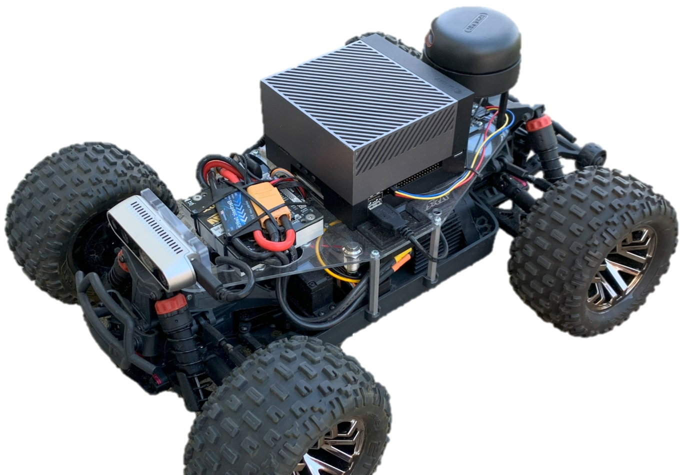


Figure 3.5: McMaster University Autonomous Electrified Vehicle (MacAEV)

Using the RealSense D435i RGB-D camera at 30 fps, 640 x 480 color frames are obtained in a 69° x 42° FOV while 848 x 480 depth frames are found in a 87° x 58° FOV. The BNO055 IMU is used for dead reckoning and fusing information from a gyroscope, accelerometer and magnetometer; vehicle state data is provided at 100 Hz through I^{2}C. The system is powered by a 3-cell, 11.1 V, 5,000 mAh, 35C LiPo battery, and the MacAEV wheelbase, l, is measured as 0.287 m.

The vehicle velocity is provided through a 3,200 kV (RPM per volt), four-pole, brushless DC motor, while a servo motor provides the vehicle steering angle. An electronic speed controller, VESC, is used to control velocity and steering angle through published commands derived from path planning. In addition to remote wireless control via a laptop, a Logitech F710 joystick can enable autonomous driving as well as manual driving if necessary. The integrated system supports real-time navigation through STLMPC as well as the additional algorithms proposed in this thesis.

The system architecture is illustrated through a block diagram in Figure 3.6. The navigation framework is shown integrated with the sensor data received and remote interfacing commands, while the output steering angle and velocity commands are applied to the servo and brushless DC motors, respectively, enabling autonomous driving. Again, T_{odom} represents the vehicle position in the fixed, global frame using dead reckoning, where AMCL provides the position in the known map frame through T_{AMCL}. The use of the RGB-D camera and YOLO model for vehicle detection, tracking and avoidance via MPC will be discussed more extensively in Chapter 4. The 3D tensor $\tilde{C}$ denotes the color frame pixels obtained from the RGB-D camera, while $\tilde{D}$ represents the depth frame pixels. The bounding boxes for all vehicles in frame, as detected by YOLO, are denoted by B.

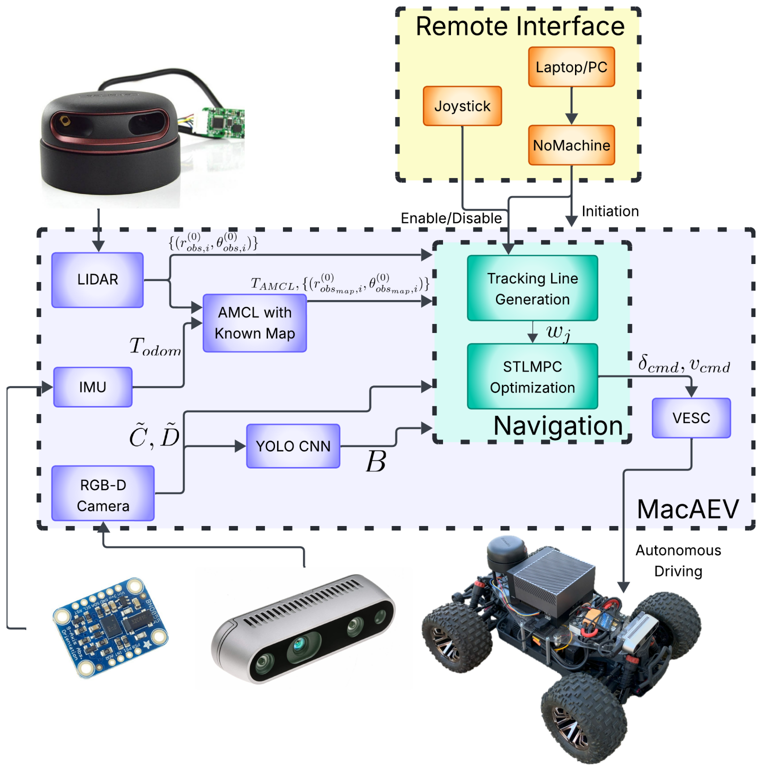


Figure 3.6: The integrated system's block diagram, which illustrates the sensors used, navigation framework, actuation and remote interfacing with the target machine. In simulation, sensors and vehicle actuation are replicated.

3.4 Simulation Results

The simulation results in this section are carried out in the *f1tenth_simulator* environment where the kinematic bicycle model is used to model the MacAEV, setting l =0.287 m, δ_{max} =0.4189 rad, $\Delta\delta_{max}$ =3.2 rad/s, constant v=1.5 m/s and Δt=0.1 s. For heading angle selection, d_{safe} =2 m, a_l=a_r=$\frac{\pi}{9}$ rad & b_l=b_r=$\frac{\pi}{2}$ rad. Two successive tracking lines are used, where n_{MPC} = 2 and k_{MPC} = 8 create a receding prediction time horizon of 1.6 s. The objective base weight factors are $\lambda_{d^2} = 1, \lambda_{\dot{d}^2} = 30$ & $\lambda_{\delta^2} = 1$ while the solver terminates if the relative change in optimization variables decreases below 0.1% or the optimization time exceeds 50 ms.

The successive reference tracking lines are shown in a simulated test case with the resulting optimized trajectory at two separate time samples (Figure 3.7). In each case, the predicted trajectory is smooth (minimizing control effort) and balances close tracking of the reference path with maintaining a heading direction parallel to each successive tracking line.

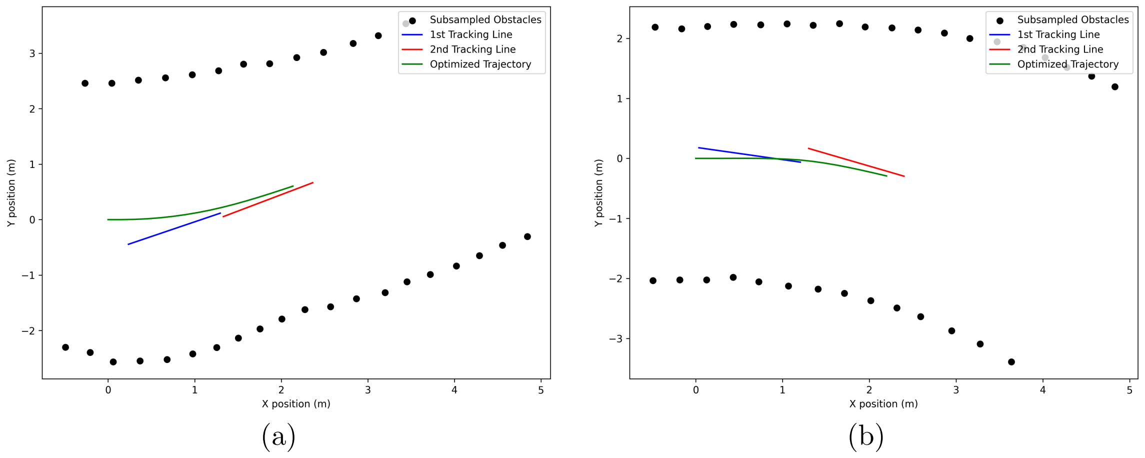


Figure 3.7: Resulting trajectory after optimization for the given successive tracking lines at two separate time samples. The subsampled obstacles are shown for visualization, although subsampling is not done when finding the tracking lines.

Now, STLMPC is evaluated against other local path planning methods, and performance is contrasted. These methods follow the standard ROS navigation stack where the central *move_base* node links a local and global planner, integrated via *nav_core*, to achieve navigation & obstacle avoidance using 2D occupancy grid costmaps. As this thesis explores navigation in unknown environments, no global goal location is provided, and instead, these planners use a common exploration package, *explore_lite*. This method updates unexplored frontiers in real-time, where in these tests, the largest one ahead of the vehicle is used as the current goal.

The global planner then plots a path to the time-varying goal while the local planner performs collision avoidance, yielding the optimized path for immediate motion.

Three common local planners in ROS—*dwa_local_planner* [12], *teb_local_planner* [14] & *mpc_local_planner* [15]—are used here with default parameters except for MacAEV-specific values for kinematics & physical properties. Additionally, a limited, non-predictive PD (Proportional-Derivative) approach is compared. This scheme generates a single tracking line (through the QP approach used in STLMPC) at each control step and maintains tracking by performing feedback linearization to obtain each step's desired steering angle control input (assuming constant velocity).

Table 3.1: Performance of STLMPC & other local planners in simulated Map #1

Local Planner	$\min_{\tilde{k}} d_{min,\tilde{k}}$ (m)	$\bar{d}_{min}$ (m)	$\overline{\lvert\delta_{cmd,\tilde{k}}\rvert}$ (rad)	$\mathrm{Var}(\delta_{cmd,\tilde{k}})$ (rad^2)	$\bar{v}_{cmd,\tilde{k}}$ (m/s)	$\mathrm{Var}(v_{cmd,\tilde{k}})$ (m^2/s^2)
DWA_explore	0.207	1.147	0.060	0.008	0.492	0.011
TEB_explore	0.521	1.289	0.075	0.011	1.474	0.027
MPC_explore	0.278	1.268	0.124	0.032	1.246	0.201
PD	0.508	1.677	0.080	0.012	1.482	0.009
STLMPC	0.611	1.718	0.082	0.014	1.480	0.009

The performance of these four alternative local planning approaches is contrasted with that of STLMPC on a simulated map course. Table 3.1 provides key metrics where $\tilde{k} \in \{0, ..., \tilde{k}_{end}\}$ denotes the sample time, ranging over the full test and $d_{min,\tilde{k}}$ indicates the minimum obstacle proximity to the current vehicle position at sample $\tilde{k}$. Moreover, $\min_{\tilde{k}} d_{min,\tilde{k}}$ represents the minimum obstacle proximity over the full test and $\bar{d}_{min}$ denotes the average minimum proximity while the averages $(\overline{|\delta_{cmd,\tilde{k}}|}, \bar{v}_{cmd,\tilde{k}})$ and variances $(\mathrm{Var}(\delta_{cmd,\tilde{k}}), \mathrm{Var}(v_{cmd,\tilde{k}}))$ of the control inputs are also included.

Notably, STLMPC achieves superior safety through $\min_{\tilde{k}} d_{min,\tilde{k}}$ & $\bar{d}_{min}$ compared to other planners while attaining low control effort and a velocity near the desired constant value. Figure 3.8 presents each planner's full trajectory, where STLMPC effectively maintains a path near the center of the course throughout. The value of the tracking line approach is shown as PD achieves a central path similar to STLMPC, whereas the exploration methods do not, thereby risking closer obstacle proximities.

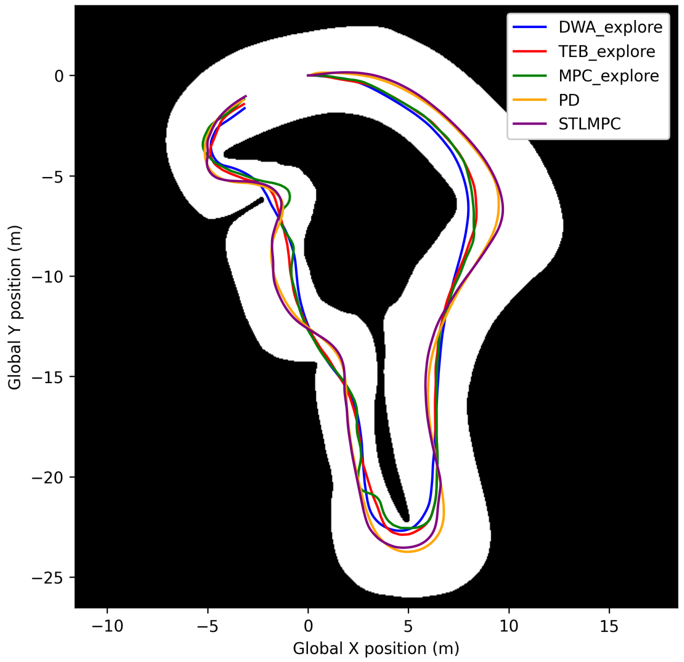
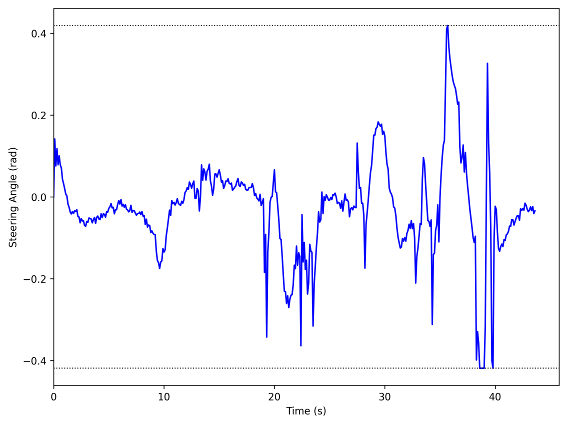


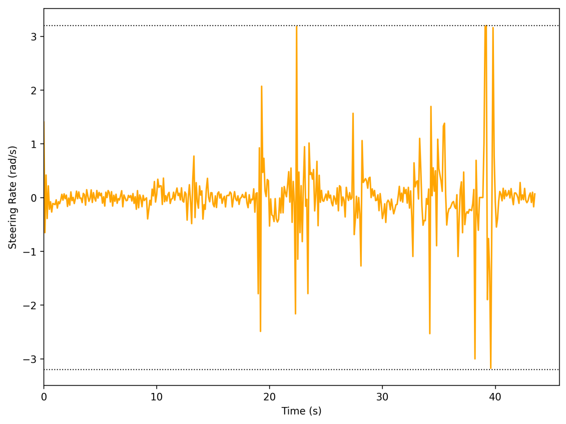
Figure 3.8: Trajectories obtained by each local planner in simulated Map #1. Each trajectory starts at the (0,0) global position & proceeds clockwise around the course.

Of the exploration planners, local planning via TEB is most effective where DWA progresses at a lower speed than desired and MPC experiences unexpected oscillations at times due to occasional poor optimization solutions.

The steering angle control inputs and steering rates are also provided for STLMPC (Figure 3.9). The resulting trajectory over the full course evidently satisfies the optimization's bounds on steering angle magnitude ($-\delta_{max} \leq \delta_i \leq \delta_{max}$) and constraints on steering angle rate ($g_{\delta^+,i}, g_{\delta^-,i} \leq 0$). Control inputs are smooth for most of the simulation, while aggressive steering occurs near the end at the sharp hairpin turn.



(a)



(b)

Figure 3.9: In the simulation using Map #1, STLMPC achieves a path where (a) steering angle control inputs remain within the dotted bounds ($\pm\delta_{max}$) & (b) steering rates satisfy the dotted extents ($\pm\Delta\delta_{max}$).

3.5 Experimental Results

Experimental testing is conducted using the MacAEV in varying environmental configurations, beginning in this section with Experiment #1 (the course layout is provided in Appendix B). These tests validate local planner performance in real-world conditions with imperfect sensor capabilities and vehicle dynamics that are not fully captured by the kinematic bicycle model. Navigation using this chapter's STLMPC in Experiment #1 is recorded by video for visualization[1]. The same parameter values used in simulation are retained in the experiment, and performance is also evaluated for several local exploration planners as well as the non-predictive PD approach.

Each local planner trajectory over the map is given by Figure 3.10, where the STLMPC, AMCL-localized STLMPC & PD trajectories maintain smooth travel along the center of the course compared to the exploration-based planners. Each planner's performance is further evaluated in Table 3.2 with the same metrics as in simulation.

[1]`https://www.youtube.com/watch?v=hOUxxvMQrGM`

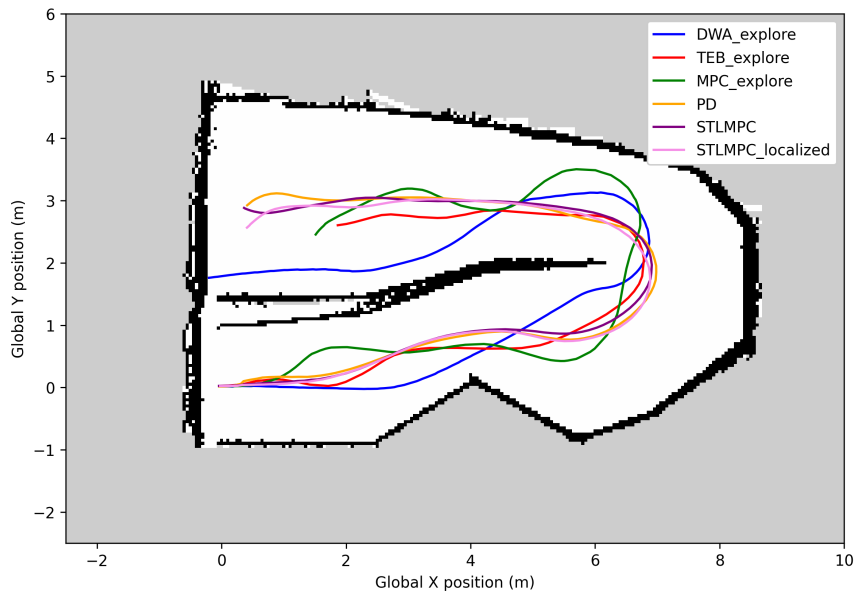


Figure 3.10: Trajectories obtained by each local planner in Experiment #1. Each path starts at the (0,0) global position & proceeds counterclockwise around the map.

The STLMPC methods perform best in terms of the obstacle proximity metrics while also exhibiting low control effort and attaining an average velocity (including initial acceleration and deceleration upon course completion) near the desired speed of 1.5 m/s. Here, PD performs comparably to STLMPC, although marginally worse across the metrics tested, while the TEB local planner achieves the best performance among the exploration-based approaches.

Table 3.2: Performance of STLMPC & other local planners in Experiment #1

Local Planner	$\min_{\tilde{k}} d_{min,\tilde{k}}$ (m)	$\bar{d}_{min}$ (m)	$\overline{\lvert\delta_{cmd,\tilde{k}}\rvert}$ (rad)	$\text{Var}(\delta_{cmd,\tilde{k}})$ (rad^2)	$\bar{v}_{cmd,\tilde{k}}$ (m/s)	$\text{Var}(v_{cmd,\tilde{k}})$ (m^2/s^2)
DWA_explore	0.385	0.654	0.118	0.028	0.491	0.012
TEB_explore	0.570	0.843	0.157	0.032	1.356	0.139
MPC_explore	0.313	0.831	0.253	0.073	0.858	0.149
PD	0.587	0.978	0.147	0.032	1.281	0.107
STLMPC	0.688	0.972	0.113	0.025	1.304	0.071
STLMPC_localized	0.640	0.982	0.116	0.019	1.296	0.065

Furthermore, each planner's computation time ($t_{comp,\tilde{k}}$) is compared in both the average ($\bar{t}_{comp}$) and worst ($\max_{\tilde{k}} t_{comp,\tilde{k}}$ denoted as $\max t_{comp}$ for brevity) case via Table 3.3. Here, the exploration-based planners have three planning stages: updating unexplored frontiers & setting a new local goal, global planning to this goal and finally, local planning to track this path while performing collision avoidance. Each exploration method differs in the third stage where the computation times for each step are denoted by $t_{frontier,\tilde{k}}$, $t_{global,\tilde{k}}$ & $t_{local,\tilde{k}}$ respectively and summed for the total planning computation time, $t_{comp,\tilde{k}}$.

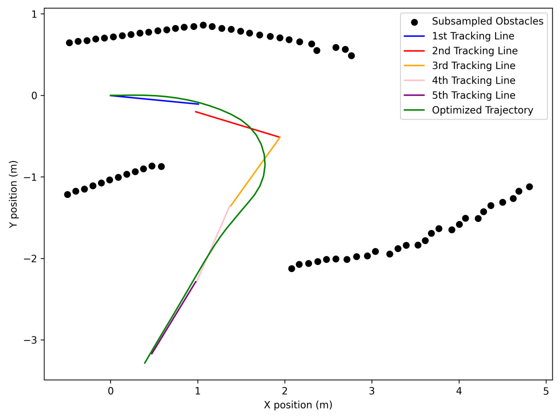
The simplistic PD approach achieves the lowest average computation time, while STLMPC with $n_{MPC} = 2$ & $k_{MPC} = 8$ also attains reliably low planning times. The computational performance of STLMPC in this experiment is similar to that of TEB, while the introduction of localization via AMCL has minimal impact on computation time. Notably, MPC-based exploration has significantly longer planning times, far exceeding the 100-ms control period in the worst case.

Table 3.3: Planning computation times for STLMPC & other local planners in Experiment #1

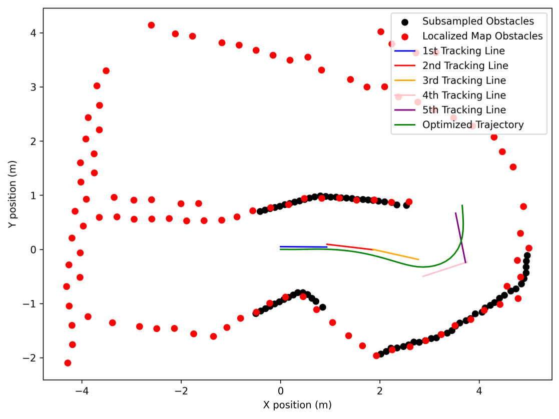
Local Planner	$\bar{t}_{frontier}$ (ms)	$\bar{t}_{global}$ (ms)	$\bar{t}_{local}$ (ms)	$\boldsymbol{\bar{t}_{comp}}$ **(ms)**	$\max t_{frontier}$ (ms)	$\max t_{global}$ (ms)	$\max t_{local}$ (ms)	$\mathbf{max}\, \boldsymbol{t_{comp}}$ **(ms)**
DWA_explore	3.6	0.9	11.8	**16.3**	7.4	1.9	27.9	**37.1**
TEB_explore	1.9	0.8	3.2	**5.9**	5.2	1.5	8.3	**15.0**
MPC_explore	2.0	1.1	43.7	**46.7**	3.8	10.2	484.8	**498.8**
PD				**0.5**				**1.7**
STLMPC				**6.5**				**15.8**
STLMPC_localized				**6.9**				**12.3**

While the overall performance of STLMPC with and without AMCL-based localization is similar, the effect on path planning at individual iterations is significant. Figure 3.11a shows the STLMPC tracking lines and optimized path at a single iteration in Experiment #1 without localization, while Figure 3.11b shows the same planning iteration, now using localized map data. The planner without AMCL does not see around the right wall and therefore plans to turn right.

However, using localized map data, it is clear that this unseen area is bounded by a wall out of sight, and thus, the AMCL-localized planner correctly plans ahead to turn left. In practice, STLMPC without AMCL proceeds along the correct path once it detects the obscured wall, and this deficiency in effective look-ahead horizon leads to negligible differences in performance, as seen in earlier results. The tests conducted in Experiment #1 illustrate the performance improvements of STLMPC over existing methods and the applicability of STLMPC to real-life contexts for safe, standalone local path planning.



(a)



(b)

Figure 3.11: Tracking lines and optimized trajectory at a shared timestep for STLMPC with/without AMCL in Experiment #1. The predicted paths differ as (a) STLMPC without AMCL turns right, not seeing an obscured wall, while (b) STLMPC with AMCL turns left, detecting the obscured wall via the known map.

Chapter 4

Vehicle Detection, Tracking & Avoidance

In this chapter, the process for detecting, estimating, tracking and avoiding moving obstacles, specifically other vehicles, is explored. The motivation is to extend the STLMPC algorithm from Chapter 3 to better predict the behavior of dynamic objects and plan a safe path accordingly. This multi-stage process works in real-time conditions, ensuring quick information flow to the path planner for operation in rapidly evolving environments.

This chapter proceeds chronologically as the dynamic obstacle accommodation framework does. First, the use of a convolutional neural network is discussed, which outputs detected vehicle bounding boxes in the RGB frame. Then, a method is proposed for associating boxes with known detections and fusing RGB & depth information for pose estimation. Using an extended Kalman filter, detected vehicle states are updated and tracked before being incorporated into Chapter 3's STLMPC. Lastly, simulation and experimental results are provided for tracking & path planning.

4.1 Vehicle Detection with a Convolutional Neural Network

In order to achieve perception of dynamic obstacles like other vehicles, computer vision is required beyond the laser scanning used for static obstacle detection. Now, a module is used with both an RGB camera as well as depth capabilities through stereo vision using two infrared cameras. The image frame of RGB pixels is denoted by $\tilde{C} \in \mathbb{R}^{H_c \times W_c \times 3}$ where H_c is the RGB image height and W_c is the width. Similarly, the depth image is represented by $\tilde{D} \in \mathbb{R}^{H_d \times W_d}$ with height, H_d, and width, W_d, not generally being equal to that of the RGB image.

A Convolutional Neural Network (CNN) is used following the You Only Look Once (YOLO) framework for real-time vehicle detection[1] using RGB images, $\tilde{C}$. YOLO achieves fast performance through a single pass of the neural network and outputs bounding boxes for classified objects as well as class probabilities. The YOLOv5s model is implemented, which scales down the base YOLOv5 model architecture to achieve real-time inference while maintaining detection accuracy.

The YOLOv5s model uses 7.2 million parameters, 213 layers and a standard backbone–neck–head architecture. The backbone, CSPDarknet in this case, is the main body that extracts features from the image, while feature aggregation occurs in the neck, using PANet. The detection head, YOLO Head, yields final predictions as outputs for bounding boxes, classes and class probabilities. Letterboxing is done to convert the 640 x 480 x 3 RGB image to a 416 x 416 x 3 input for the CNN while maintaining the original aspect ratio, since YOLO performs better on square

[1]It should be noted that the YOLO model used in this thesis for vehicle detection was trained and implemented on the MacAEV by Leo Calogero during his time as an undergraduate summer researcher in the Department of Electrical and Computer Engineering at McMaster University.

images. Upon detection, the resulting bounding boxes are rescaled to the original image dimensions.

Custom training data is gathered and compiled on Roboflow before training is done on the YOLOv5s model through Ultralytics. Custom training allows for classification of the MacAEV, which isn't an otherwise standard recognizable class by YOLO. This is the only detectable class, and so all detections represent other MacAEV agents in the environment. Training is done on augmented data, such as through image rotation, blurring, increased noise, and image shearing, which reduces model overfitting. This is also valuable as the camera is attached to the MacAEV, which, during motion and especially rotation, can experience increased blurring. Leveraging the 2048-core NVIDIA Ampere GPU and inference through the TensorRT engine, MacAEV detection maintains a 30 fps throughput on the RGB image pipeline.

As a result of vehicle detection through YOLO, classified objects with probabilities and bounding boxes are provided. To reduce the chance of false positives, only detections with confidence exceeding the threshold of 40%, s_{thresh}=0.4 are considered:

$$s_i > s_{thresh}, \quad i \in \{0, ..., N_{det} - 1\} \tag{4.1.1}$$

where s_i denotes the confidence of the i^{th} detection and N_{det} is the total number of detections in the current image frame. Each detection's class, ℓ_i, corresponds to the MacAEV since this was the only object class used in testing. Each detection's bounding box, b_i, in the 640 x 480 RGB image frame is represented by:

$$b_i = [x_i^{min}, y_i^{min}, x_i^{max}, y_i^{max}]^T \quad i \in \{0, ..., N_{det} - 1\} \tag{4.1.2}$$

where the rectangle that encapsulates the classified object is fully parameterized by its two diagonal corners, (x_i^{min}, y_i^{min}) and (x_i^{max}, y_i^{max}). In this image coordinate system, x represents the width direction and increases in the rightward direction, while y represents the height direction and increases in the downward direction. Here, the RGB image, $\tilde{C}$ has RGB pixels denoted by $\tilde{c}_{y,x} \in \mathbb{R}^3$. This coordinate frame also applies to the depth image, $\tilde{D}$, where pixel depths are denoted by $\tilde{d}_{y,x}$. The bounding boxes for detected vehicles in the image coordinate system are depicted in Figure 4.1. The set of all bounding boxes, B, for the current image frame is denoted by $B = \{b_0, ..., b_{N_{det}-1}\}$, which is empty in the case of no detections. The bounding boxes obtained for each RGB image frame are then used in further steps of vehicle estimation, tracking and avoidance.

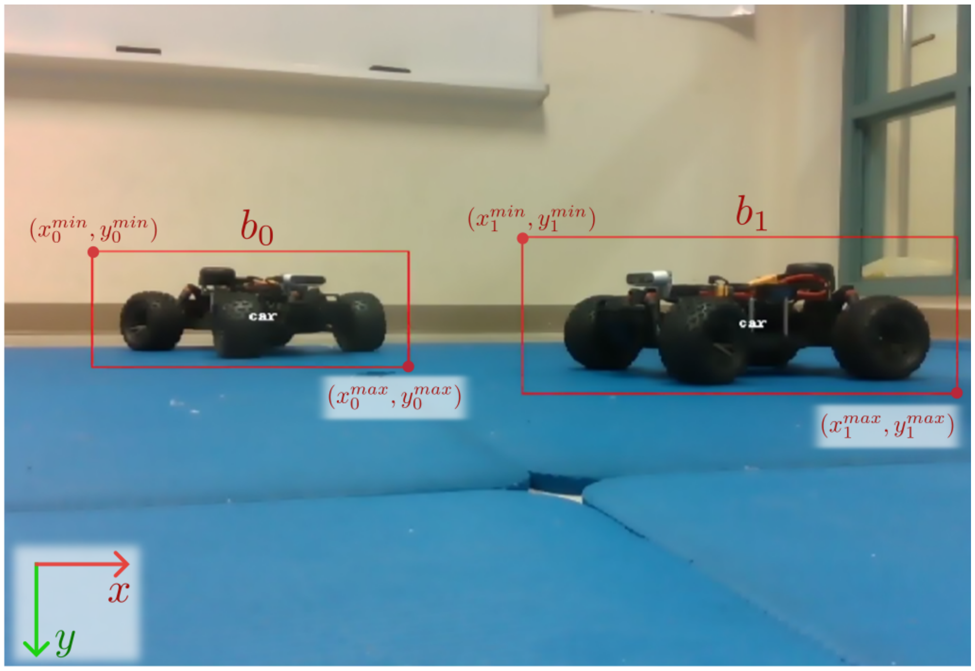


Figure 4.1: Two detected vehicles are identified via YOLO with corresponding bounding boxes, b_0 & b_1. Each bounding box is fully described by its two diagonal corner points, while the image coordinate frame is indicated for the $W_c = 640$ pixel, $H_c = 480$ pixel RGB image.

4.2 Pose Estimation of Observed Vehicles

Pose estimation proceeds from the detected bounding boxes acquired from the YOLO model previously. To accommodate multiple vehicle detections, association of bounding boxes to known previous detections must be done. In this sense, a framework is proposed for handling new detections, associating detections with their respective tracked vehicles in the multi-detection case, and removing tracked vehicles if no new corresponding detections are received. Now, each grouped detection is converted from a bounding box in the RGB frame to a pose in 3D space through the relevant depth camera data. These become the positional measurements for the detected vehicles, which are used in extended Kalman filtering for vehicle state tracking.

4.2.1 Vehicle Association

In multi-object tracking, detections must be paired with their corresponding tracked objects across a stream of RGB images. However, issues can arise in the presence of noise, blur & occlusions and in the case of overlapping classified objects in frame. Deep learning can perform both detection and tracking of center points based on successive frames; however, identity switches occur frequently for detections, especially during occlusion [136]. Meanwhile, methods that track bounding box position and size uncertainties to match objects based on prior detections in the multi-object tracking case can handle temporary occlusions or movements out of frame [137].

For assigning new detections from the YOLO model to existing tracked vehicles, $\mathcal{V}_{track}$:

$$\mathcal{V}_{track} = \{\nu_{track,0}, ..., \nu_{track,N_{track}-1}\} \tag{4.2.1}$$

the bounding boxes, B, of the latest RGB image are used. The number of detections in the image frame, N_{det}, and the number of currently tracked vehicles, N_{track}, are generally not equal, so detection association considers new as well as missing detections. The currently detected bounding boxes are denoted as b_i, $i \in \{0, ..., N_{det} - 1\}$, while each tracked vehicle has its own corresponding known bounding box from its most recent previous detection, $b_{track,j}$, $j \in \{0, ..., N_{track} - 1\}$. The bounding box midpoints are gathered as:

$$b_i^{mid} = (\frac{x_i^{min} + x_i^{max}}{2}, \frac{y_i^{min} + y_i^{max}}{2}) \quad \forall i \in \{0, ..., N_{det} - 1\} \tag{4.2.2}$$

and

$$b_{track,j}^{mid} = (\frac{x_{track,j}^{min} + x_{track,j}^{max}}{2}, \frac{y_{track,j}^{min} + y_{track,j}^{max}}{2}) \quad \forall j \in \{0, ..., N_{track} - 1\} \tag{4.2.3}$$

for the detected and known tracked vehicle bounding boxes, respectively.

For each detection's midpoint, the unassigned, tracked vehicle with the closest previous bounding box midpoint is assigned the detection's bounding box. Once assigned, only the remaining tracked vehicles are considered for the other detected bounding boxes. If the distance of the closest midpoint match exceeds a threshold, d_{assoc}, the detection is not associated with an existing tracked vehicle and instead represents a new tracked vehicle. For detection association, the closest tracked vehicle (indexed by j_i^*) for each detection (indexed by i) is found through:

$$j_i^* = \arg \min_{\substack{j \in \mathcal{J} \\ \|b_{track,j}^{mid} - b_i^{mid}\|_2 \leq d_{assoc}}} \|b_{track,j}^{mid} - b_i^{mid}\|_2 \quad \forall i \in \{0, ..., N_{det} - 1\} \tag{4.2.4}$$

where $\mathcal{J}$ denotes the set of unassigned tracked vehicle indices, which contracts for every assigned detection in this iterative process. All associated bounding boxes are assigned to their appropriate tracked vehicle for the current frame:

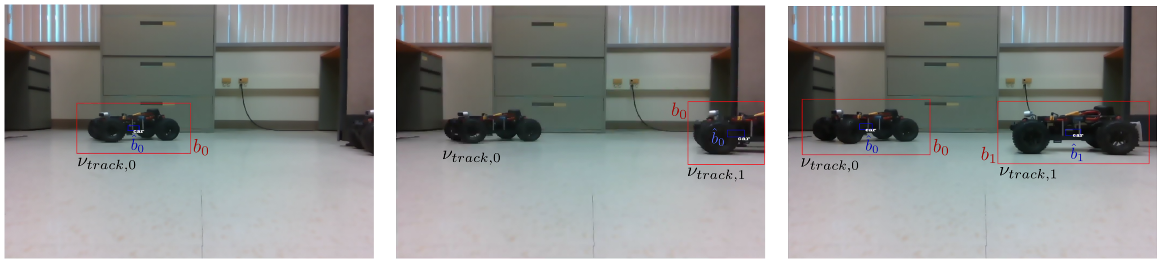
$$b_{track,j_i^*} = b_i \quad \forall i \in \{0, ..., N_{det} - 1\} \text{ if } j_i^* \text{ exists} \tag{4.2.5}$$

which both resets the respective missed frame count, $N_{miss,j}$, to 0 and allows for the current pose measurement to be made using image depth data. For any tracked vehicles remaining in $\mathcal{J}$ after all detections are processed, a new measurement is not made and the number of missed frames increases.

If there exists no corresponding j_i^* for the i^{th} detection since either $\forall j \in \mathcal{J}$, $\|b_{track,j}^{mid} - b_i^{mid}\|_2 > d_{assoc}$ or $N_{det} > N_{track}$, the new tracked detection, ν_{new} is incorporated in the tracked vehicle ($\mathcal{V}_{track} \leftarrow \mathcal{V}_{track} \cup \{\nu_{new}\}$) and known bounding box ($B_{track} \leftarrow B_{track} \cup \{b_i\}$) sets. This multi-object approach therefore associates detections based on closest bounding boxes in successive frames while also handling the cases of new and missed detections (Figure 4.2). Vehicles continue to be tracked for several consecutive missed frames, and accuracy can be maintained even with intermittent detections.

4.2.2 Sensor Fusion

Now, using the bounding boxes, b_i, which are appropriately associated with the correct tracked vehicles, position measurements are derived, which will be used to update vehicle state tracking. Given the relevant bounding box for the color pixels, the relevant depth pixels are retrieved through integration of the two images, and an averaged 3D position is obtained from the image. Unlike going from a depth to



(a) (b) (c)

Figure 4.2: Successive RGB frames with YOLO detected inner ($\hat{b}_i$) and outer (b_i) bounding boxes in the case of: (a) initial detection of $\nu_{track,0}$, (b) missed ($\nu_{track,0}$) and new ($\nu_{track,1}$) detections & (c) recovered detection for $\nu_{track,0}$.

color pixel, however, the transformation from a color to depth pixel is not direct or guaranteed due to the lack of 3D knowledge of the color pixel (corresponding to a ray of possible depths). Methods that align the depth and camera frames in preprocessing resolve this issue but introduce meaningful lag in the RGB-D image pipeline. Thus, timestamped color and depth images are paired, and a process is developed to extract the 3D positions of only relevant color pixels at 30 fps for vehicle state estimation without aligning all pixels in a time-consuming, lag-inducing manner.

To convert a depth pixel to a 3D point in the depth frame, the depth camera intrinsics are considered, namely the focal lengths, $f_{x,d}$ & $f_{y,d}$ and principal point offsets, $c_{x,d}$ & $c_{y,d}$. Assuming a distortion-free image (where distortions are undistorted iteratively otherwise), the 3D point, $p^{depth}_{y_d,x_d}$ for a depth pixel, $\tilde{d}_{y_d,x_d}$ is obtained in the depth frame via:

$$p^{depth}_{y_d,x_d} = [\frac{\tilde{d}_{y_d,x_d}(x_d - c_{x,d})}{f_{x,d}}, \frac{\tilde{d}_{y_d,x_d}(y_d - c_{y,d})}{f_{y,d}}, \tilde{d}_{y_d,x_d}]^T \tag{4.2.6}$$

and can be converted to the base frame for path planning through the transformation, T_{depth}. To convert the point, $p^{depth}_{y_d,x_d}$ to the color camera frame, the transform between the depth (described by T_{depth}) and color (described by T_{color}) frames is applied.

For the 3D point in the color frame, $p^{color}_{y_c,x_c} = [x, y, z]^T$, the color pixel location, (x_c, y_c) in its frame coordinates is found through:

$$(x_c, y_c) = (f_{x,c}\frac{x}{z} + c_{x,c}, f_{y,c}\frac{y}{z} + c_{y,c}) \tag{4.2.7}$$

using the color camera intrinsics, $f_{x,c}, f_{y,c}, c_{x,c}$ & $c_{y,c}$ where again no distortion is assumed. The steps to convert $(x_d, y_d) \rightarrow (x_c, y_c)$ are used to iteratively acquire the corresponding depth pixels for the color pixels used in the bounding box. A smaller, centered, self-contained bounding box is defined for each detected box in order to average depth measurements over a central area of the MacAEV. This limits the risk of background depth pixels being included and provides an estimate of the vehicle's central position. The self-contained boxes, $\hat{b}_i$ are defined by:

$$\hat{b}_i = \begin{bmatrix} 1-\zeta_{x_{min}} & 0 & \zeta_{x_{min}} & 0 \\ 0 & 1-\zeta_{y_{min}} & 0 & \zeta_{y_{min}} \\ 1-\zeta_{x_{max}} & 0 & \zeta_{x_{max}} & 0 \\ 0 & 1-\zeta_{y_{max}} & 0 & \zeta_{y_{max}} \end{bmatrix} \cdot b_i \tag{4.2.8}$$

considering the scaling factors, $\zeta_{x_{min}}$, $\zeta_{y_{min}}$, $\zeta_{x_{max}}$ & $\zeta_{y_{max}} \in [0, 1]$ which control the edges of the inner bounding box. These factors can be shifted from the center towards a particular edge of the outer box to better track vehicles with bounding boxes cut off by the frame extents.

A starting point of half the inner box's corner, $(\hat{x}^{min}_i, \hat{y}^{min}_i)$, is selected as the initial estimate of depth pixel corresponding to this corner's color pixel (due to the depth camera's larger FOV). Then, applying Equation 4.2.6, the depth-to-color frame transform and Equation 4.2.7 to obtain the corresponding color pixel coordinates, the difference between these and the desired starting color pixel coordinates, $(\hat{x}_i^{min}, \hat{y}_i^{min})$, is halved (for quicker computation time than a standard line search) and applied to the depth pixel estimate. This iterative process continues until the matching color pixel estimate is within the inner bounding box.

Iterating over the grid of consecutive depth pixels until the color pixels in the inner bounding box are fully covered, the resulting depth values are converted to 3D positions (Equation 4.2.6) and averaged over all valid depths. The resulting measured position transformed to the vehicle base frame, $p_{track,i}$, is therefore the average over the inner bounding box, which provides a less noisy, central estimate of vehicle position.

4.3 Extended Kalman Filter Vehicle Tracking

4.3.1 Overview & Initialization

A common method for target tracking is the Kalman Filter (KF), while in the case of nonlinearities in the process or measurement models, the Extended Kalman Filter (EKF) is used. This approach estimates the states of detected vehicles, which update as new position measurements are processed from the depth images received. Here, the measurement $\mathbf{z}_{j,k} = [x_{track,j,k}, y_{track,j,k}]^T$ corresponding to the j^{th} tracked vehicle is used in the EKF to estimate the vehicle states, $\hat{\mathbf{x}}_{j,k} = [x_{j,k}, y_{j,k}, \theta_{j,k}, \delta_{j,k}, v_{j,k}]^T$ at time k according to a constant velocity & curvature model.

The update rate of the EKF is equal to the control rate of the path planner (in practice, determined by the publication rate of the laser sensor), which is typically slower than the camera rate. Thus, multiple image frames are received and converted to position measurements per EKF update. The most recent measurement since the last EKF update is used, where this sample aggregation process reduces the chance of missed frames between filter updates. Furthermore, as position changes at 30 fps may be indistinguishable in successive frames, reducing the EKF update rate helps eliminate redundant measurements. Meanwhile, averaging aggregated measurements can reduce noise, assuming it is zero-mean and uncorrelated. After a number of EKF updates without new measurements (where $N_{miss,j} > N_{miss}^{max}$), a vehicle is no longer tracked; however, it can be recovered as a separate tracked vehicle instance if it reappears later.

To initialize newly tracked vehicles, two-point initialization is used. Once measurements are obtained for consecutive updates, the state is initialized as:

$$\hat{\mathbf{x}}_{j,k} = \begin{bmatrix} x_{track,j,k} \\ y_{track,j,k} \\ \text{atan2}(y_{track,j,k} - y_{track,j,k-1}, x_{track,j,k} - x_{track,j,k-1}) \\ 0 \\ \min(\frac{\|\mathbf{z}_{j,k} - \mathbf{z}_{j,k-1}\|_2}{\Delta t}, v_0^{max}) \end{bmatrix} \tag{4.3.1}$$

where zero curvature is initially assumed and v_0^{max} limits the initial velocity to a reasonable value in the case of noisy measurements. The covariance is initialized as:

$$\mathbf{P}_{j,k} = \begin{bmatrix} \sigma_{x_0}^2 & 0 & 0 & 0 & 0 \\ 0 & \sigma_{y_0}^2 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{\theta_0}^2 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{\delta_0}^2 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{v_0}^2 \end{bmatrix} \tag{4.3.2}$$

where $\sigma_{x_0}^2, \sigma_{y_0}^2, \sigma_{\theta_0}^2, \sigma_{\delta_0}^2$ & $\sigma_{v_0}^2$ describe the initial variances for each state estimate. The tracking process follows three general steps: transformation, prediction & correction.

First, the latest measurement and previous vehicle state are transformed from the ego vehicle's previous frame to the current one (since it too is moving in the global frame). Then, the EKF prediction and correction steps are performed to obtain the updated state and covariance estimates.

4.3.2 State & Measurement Transformation

To transform past state and measurement data to the current ego vehicle's base frame, the past and current transforms between the moving base frame and fixed global frame are stored. For state estimates, after each EKF update, the transform $T_{odom,k}$ (via odometry) is stored. In the next EKF update, the old and current transforms (respectively denoted now as $T_{odom,k-1} = [x_{odom,k-1}, y_{odom,k-1}, \theta_{odom,k-1}]^T$ and $T_{odom,k} = [x_{odom,k}, y_{odom,k}, \theta_{odom,k}]^T$) are used to transform the position & orientation state estimates to the current frame:

$$\begin{bmatrix} x^{new}_{j,k-1} \\ y^{new}_{j,k-1} \\ 1 \end{bmatrix} = T^{new}_{odom} \cdot \begin{bmatrix} \cos(\theta_{odom,k-1}) & -\sin(\theta_{odom,k-1}) & x_{odom,k-1} \\ \sin(\theta_{odom,k-1}) & \cos(\theta_{odom,k-1}) & y_{odom,k-1} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x^{old}_{j,k-1} \\ y^{old}_{j,k-1} \\ 1 \end{bmatrix} \tag{4.3.3}$$

$$\theta^{new}_{j,k-1} = \theta^{old}_{j,k-1} - (\theta_{odom,k} - \theta_{odom,k-1}) \tag{4.3.4}$$

where the transformation between the fixed frame and the new vehicle base frame, T^{new}_{odom}, is described by:

$$T^{new}_{odom} = \begin{bmatrix} \cos(\theta_{odom,k}) & \sin(\theta_{odom,k}) & -x_{odom,k}\cos(\theta_{odom,k}) - y_{odom,k}\sin(\theta_{odom,k}) \\ -\sin(\theta_{odom,k}) & \cos(\theta_{odom,k}) & x_{odom,k}\sin(\theta_{odom,k}) - y_{odom,k}\cos(\theta_{odom,k}) \\ 0 & 0 & 1 \end{bmatrix} \tag{4.3.5}$$

For measurements, the same process holds, but now the fixed to old vehicle frame transform used has the closest timestamp to the time when the measurement's depth image was captured. Here, the position measurement uses this distinct old transform in the process described in Equation 4.3.3. This provides the most accurate, time-sensitive conversion of the measured, tracked vehicle position to the fixed frame, then to the new ego vehicle frame.

The base frame in consecutive EKF updates is depicted in Figure 4.3 for transforming the prior tracked vehicle estimates to the new vehicle base frame. Two vehicles are tracked, where their final EKF updated states at time k and corresponding future trajectories are shown. Note that the tracked vehicle position states correspond to the vehicle's center due to the average position measurement being obtained over the central vehicle section from before. While the position and orientation of the ego vehicle are quantified with respect to the fixed frame here, the tracked vehicles are always assessed with respect to the ego vehicle base frame.

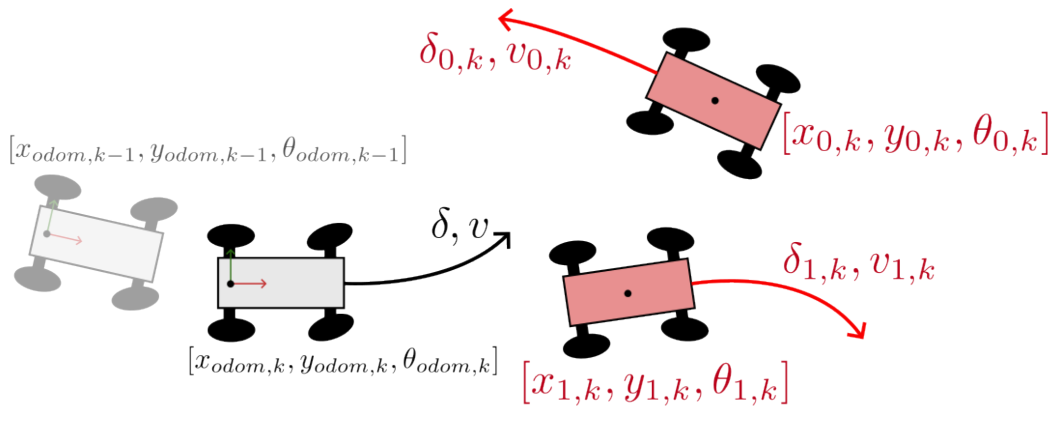

$\delta_{0,k}, v_{0,k}$
$[x_{0,k}, y_{0,k}, \theta_{0,k}]$
$[x_{odom,k-1}, y_{odom,k-1}, \theta_{odom,k-1}]$
δ, v
$\delta_{1,k}, v_{1,k}$
$[x_{odom,k}, y_{odom,k}, \theta_{odom,k}]$
$[x_{1,k}, y_{1,k}, \theta_{1,k}]$

Figure 4.3: Consecutive ego vehicle base frame transformations to the fixed frame are shown, used to convert tracked vehicle states and measurements to the current frame. Two tracked (red) vehicles are illustrated with their predicted paths.

4.3.3 Prediction

The next step is to predict the tracked vehicle state, $\hat{\mathbf{x}}_{j,k|k-1}$ and covariance matrix, $\mathbf{P}_{j,k|k-1}$ using the last sample's state & covariance matrix as well as the model dynamics and adaptive process noise covariances. State prediction uses the nonlinear kinematic bicycle model based on the current estimated steering angle and velocity according to:

$$\hat{\mathbf{x}}_{j,k|k-1} = \begin{bmatrix} x_{j,k-1} + \Delta t\, v_{j,k-1}\cos(\theta_{j,k-1}) \\ y_{j,k-1} + \Delta t\, v_{j,k-1}\sin(\theta_{j,k-1}) \\ \theta_{j,k-1} + \Delta t\, \frac{v_{j,k-1}}{l}\tan(\delta_{j,k-1}) \\ \delta_{j,k-1} \\ v_{j,k-1} \end{bmatrix} \tag{4.3.6}$$

For the nonlinear process model (Equation 4.3.6), $\mathbf{f}(\cdot)$, the linearized state transition matrix, $\mathbf{F}_{j,k-1}$ is obtained through the Jacobian:

$$\mathbf{F}_{j,k-1} = \begin{bmatrix} 1 & 0 & -\Delta t\, v_{j,k-1}\sin(\theta_{j,k-1}) & 0 & \Delta t\, \cos(\theta_{j,k-1}) \\ 0 & 1 & \Delta t\, v_{j,k-1}\cos(\theta_{j,k-1}) & 0 & \Delta t\, \sin(\theta_{j,k-1}) \\ 0 & 0 & 1 & \Delta t\, \frac{v_{j,k-1}}{l\cos^2(\delta_{j,k-1})} & \frac{\Delta t}{l}\tan(\delta_{j,k-1}) \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.3.7}$$

which enables the predicted covariance to be found through matrix multiplication.

Another necessary component is the process noise covariance, $\mathbf{Q}_{j,k-1}$, which represents the uncertainty in the system's evolution due to unmodeled dynamics or disturbances. The process noise covariance is initialized to the initial covariance matrix, $\mathbf{Q}_{j,k-1} = \mathbf{P}_{j,k-1}$ (Equation 4.3.2), where it is assumed that each state variable's process noise is uncorrelated. If no measurement is obtained, the estimated state vector and covariance matrix take on the values in Equations 4.3.6 and 4.3.10, respectively, with the forthcoming correction step skipped (leading to an estimate with lower confidence). Here, the process noise covariance is considered exclusively, which increases uncertainty over time. If measurements are once again obtained, the correction step is applied to the state estimate and covariance matrix, increasing confidence.

The process noise is adaptively scaled by a term reflecting both the ego vehicle (v) and the tracked vehicle ($v_{j,k-1}$) current velocities, as higher speeds introduce greater uncertainty in the estimate. This term is given by q_{scale} where:

$$q_{scale} = \begin{cases} 1 & \text{if } \frac{v\, v_{j,k-1}}{\alpha_Q} \leq 1 \\ \frac{v\, v_{j,k-1}}{\alpha_Q} & \text{if } 1 < \frac{v\, v_{j,k-1}}{\alpha_Q} < q_{scale_{\max}} \\ q_{scale_{\max}} & \text{if } \frac{v\, v_{j,k-1}}{\alpha_Q} \geq q_{scale_{\max}} \end{cases} \tag{4.3.8}$$

The process noise scaling factor is bounded by 1 and $q_{scale_{\max}}$ to ensure reasonable values, while the velocity-dependent term is scaled by α_Q to control the rate at which higher velocities increase the process noise. The resulting adaptive process noise after being initialized at each step via $\sigma^2_{x_0}, \sigma^2_{y_0}, \sigma^2_{\theta_0}, \sigma^2_{\delta_0}$ & $\sigma^2_{v_0}$ is scaled according to:

$$\mathbf{Q}_{j,k-1} \leftarrow q_{scale}\mathbf{Q}_{j,k-1} \tag{4.3.9}$$

ensuring that process noise is considered adaptively under varying tracking conditions.

The linearized state transition and process noise covariance matrices are incorporated in the covariance prediction described by:

$$\mathbf{P}_{j,k|k-1} = \mathbf{F}_{j,k-1}\mathbf{P}_{j,k-1}\mathbf{F}^{T}_{j,k-1} + \mathbf{Q}_{j,k-1} \tag{4.3.10}$$

where the predicted state vector and covariance matrix are now augmented with the acquired measurement data to update the estimate for each in the final EKF step.

4.3.4 Correction

To complete the EKF update, the position measurement of each tracked vehicle is compared to the prior estimated state based on the confidence in each to obtain a new, updated estimation. The constant measurement model, **H**, is linear as the position measurement directly represents the position states, so the innovation is found by:

$$\mathbf{y}_{j,k} = \mathbf{z}_{j,k} - \mathbf{H}\hat{\mathbf{x}}_{j,k|k-1}, \quad \mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{4.3.11}$$

The innovation, $\mathbf{y}_{j,k}$, describes the difference between the measurement observed and what is expected based on the predicted states, while the innovation covariance, $\mathbf{S}_{j,k}$, quantifies the uncertainty of this updated difference.

Another necessary covariance matrix is that of the measurement noise, $\mathbf{R}_{j,k}$, with individual measurement noises $\sigma^2_{x_{meas,j,k}}$ & $\sigma^2_{y_{meas,j,k}}$:

$$\mathbf{R}_{j,k} = \begin{bmatrix} \sigma^2_{x_{meas,j,k}} & 0 \\ 0 & \sigma^2_{y_{meas,j,k}} \end{bmatrix} \tag{4.3.12}$$

which, similarly to the process noise covariance, is updated adaptively. The measurement noise is scaled by the range between the ego and tracked vehicles since larger distances increase the absolute position uncertainty of the depth camera measurements. Additionally, the noise is scaled by a term reflecting the ego and tracked vehicle velocities (just as with the process noise) because higher velocities decrease confidence in the time-sensitive position measurements.

From here, the innovation covariance matrix is found from the predicted and measurement noise covariance matrices:

$$\mathbf{S}_{j,k} = \mathbf{H}\mathbf{P}_{j,k|k-1}\mathbf{H}^T + \mathbf{R}_{j,k} \tag{4.3.13}$$

The innovation covariance matrix is then used to find the Kalman gain, $\mathbf{K}_{j,k}$:

$$\mathbf{K}_{j,k} = \mathbf{P}_{j,k|k-1}\mathbf{H}^T\mathbf{S}_{j,k}^{-1} \tag{4.3.14}$$

which directly provides the degree to which the predicted state estimate and covariance matrix are adjusted in light of the new measurement. The updated state estimate for the j^{th} tracked vehicle at time k becomes:

$$\hat{\mathbf{x}}_{j,k|k} = \hat{\mathbf{x}}_{j,k|k-1} + \mathbf{K}_{j,k}\mathbf{y}_{j,k} \tag{4.3.15}$$

while for the covariance matrix, the updated value equals:

$$\mathbf{P}_{j,k|k} = (\mathbf{I} - \mathbf{K}_{j,k}\mathbf{H})\mathbf{P}_{j,k|k-1} \tag{4.3.16}$$

where $\mathbf{I}$ denotes the 5 x 5 identity matrix.

The updated state vector, $\hat{\mathbf{x}}_{j,k} = \hat{\mathbf{x}}_{j,k|k}$ and covariance matrix, $\mathbf{P}_{j,k} = \mathbf{P}_{j,k|k}$ are used again in the next EKF update starting from Equation 4.3.3 and following the transformation, prediction, correction framework. The updated state estimate can now be used to model the detected vehicle's trajectory over a future time horizon. This information is incorporated into the MPC path planning algorithm, enabling the ego vehicle to avoid dynamic, tracked obstacles in real-time.

4.4 Avoidance of Dynamic Obstacles in STLMPC

The base STLMPC algorithm is now extended to consider the future trajectories of other tracked vehicles according to a constant velocity & curvature model. Each control step, vehicle tracking and estimation occur following the EKF process before the updated state vectors are used to ensure collision avoidance and safe path planning in these more complex, multi-vehicle environments. Now, the tracked future trajectories are segmented into successive timeframes (just as with the tracking lines), where obstacles that reflect the tracked vehicle paths are used in the tracking line quadratic optimizations.

Using the kinematic bicycle model, future tracked vehicle state estimates are predicted based on fixed steering angle and velocity according to the current estimate (as in state prediction via Equation 4.3.6). The future predicted paths are therefore described by:

$$\hat{\mathbf{x}}_{j,k+i|k} \quad \forall j \in \{0, ..., N_{track} - 1\}, \, i \in \{0, ..., n_{MPC}k_{MPC} - 1\} \tag{4.4.1}$$

For simplification as before, $\hat{\mathbf{x}}_{j,k+i} = \hat{\mathbf{x}}_{j,k+i|k}$ where the future states of the N_{track}

vehicles are considered in successive timeframes to generate the q^{th} tracking line by:

$$\mathcal{O}^{(q)}_{obs_{tot}} \leftarrow \mathcal{O}^{(q)}_{obs_{tot}} \cup \{(x^{(q)}_{j,k+i}, y^{(q)}_{j,k+i})\} \text{ for } qk_{MPC} \leq i < (q+1)k_{MPC},\ q \in \{0, ..., n_{MPC}-1\} \tag{4.4.2}$$

This samples the tracked trajectories just as the tracking lines into n_{MPC} successive segments, where each is k_{MPC} samples long. Here, $\mathcal{O}^{(q)}_{obs_{tot}} = \{(x^{(q)}_{obs_{tot}}, y^{(q)}_{obs_{tot}})\}$ denotes the set of obstacles for the q^{th} tracking line which is later split into left ($\mathcal{O}^{(q)}_{l}$) and right ($\mathcal{O}^{(q)}_{r}$) clusters for tracking line generation. The set of obstacles for each tracking line now becomes the combination of sensor detections ($\mathcal{O}^{(q)}_{obs} = \{(x^{(q)}_{obs,i}, y^{(q)}_{obs,i})\}_{i=0}^{N_{obs}-1}$), known map obstacles via AMCL (if used) ($\mathcal{O}^{(q)}_{obs_{map}} = \{(x^{(q)}_{obs_{map},i}, y^{(q)}_{obs_{map},i})\}_{i=0}^{N_{map}-1}$) and newly, the detected vehicle paths ($\mathcal{O}^{(q)}_{obs_{det}} = \{(x^{(q)}_{j,k+i}, y^{(q)}_{j,k+i})\}_{i=qk_{MPC},\, j=0}^{(q+1)k_{MPC}-1,\, N_{track}-1}$):

$$\mathcal{O}^{(q)}_{obs_{tot}} = \mathcal{O}^{(q)}_{obs} \cup \mathcal{O}^{(q)}_{obs_{map}} \cup \mathcal{O}^{(q)}_{obs_{det}} \tag{4.4.3}$$

where $x^{(q)}_{j,k+i}$ & $y^{(q)}_{j,k+i}$ represent the transformed tracked vehicle position in the frame of the q^{th} tracking line optimization.

As before (Equation 3.1.8), all of the obstacles must be transformed from the base frame to the future tracking line reference frame for each quadratic optimization. This applies to the future states of the tracked vehicle, where the orientation is also transformed via:

$$\theta^{(q)}_{j,k+i} = \theta_{j,k+i} - \theta_{track,q-1} \tag{4.4.4}$$

expressing $\theta_{j,k+i}$ in the base frame as $\theta^{(q)}_{j,k+i}$ in the q^{th} tracking line optimization frame.

To more accurately model the moving vehicle and its shape, the outline is used to generate obstacles for avoidance as opposed to the moving center described in Equation 4.4.1. The four corners of the vehicle body, $\hat{C}^{(q)}_{j,k+i}$ are extracted based on orientation in the future frame (Equation 4.4.4), position, as well as vehicle length, $\mathrm{l_v}$ and width, $\mathrm{w_v}$ dimensions:

$$\hat{C}^{(q)}_{j,k+i} = \begin{bmatrix} \cos(\theta^{(q)}_{j,k+i}) & -\sin(\theta^{(q)}_{j,k+i}) \\ \sin(\theta^{(q)}_{j,k+i}) & \cos(\theta^{(q)}_{j,k+i}) \end{bmatrix} \cdot \begin{bmatrix} \frac{\mathrm{l_v}}{2} & \frac{\mathrm{l_v}}{2} & -\frac{\mathrm{l_v}}{2} & -\frac{\mathrm{l_v}}{2} \\ -\frac{\mathrm{w_v}}{2} & \frac{\mathrm{w_v}}{2} & \frac{\mathrm{w_v}}{2} & -\frac{\mathrm{w_v}}{2} \end{bmatrix} + \begin{bmatrix} x^{(q)}_{j,k+i} \\ y^{(q)}_{j,k+i} \end{bmatrix} \cdot \mathbf{1}^T \quad (4.4.5)$$

Here, $\mathbf{1}$ denotes the 4 x 1 column vector of ones and the resulting 2 x 4 vehicle corner matrix contains the four corner position pairs for the j^{th} tracked vehicle at time $k + i$ in the q^{th} tracking line optimization frame. For each corner position, $\hat{c}^{(q)}_{j,k+i,n}$, $n \in \{0, ..., 3\}$, the vehicle edge is discretized and the points are added to $\mathcal{O}^{(q)}_{obs_{tot}}$ in place of just the vehicle center point. For the q^{th} tracking line optimization, this is denoted by:

$$\mathcal{O}^{(q)}_{obs_{tot}} \leftarrow \mathcal{O}^{(q)}_{obs_{tot}} \cup \{(1 - \frac{m}{n_{edge}})\hat{c}^{(q)}_{j,k+i,n} + \frac{m}{n_{edge}}\hat{c}^{(q)}_{j,k+i,n+1 \bmod 4}\}, m = \{0, \frac{1}{n_{edge}}, ..., \frac{n_{edge}-1}{n_{edge}}\} \quad (4.4.6)$$

where n_{edge} denotes the number of interpolated edge points along each side of the tracked vehicle, $qk_{MPC} \leq i < (q+1)k_{MPC}$, $j \in \{0, ..., N_{track} - 1\}$ and $n \in \{0, ..., 3\}$.

Therefore, for each tracked vehicle at each sampled trajectory point corresponding to the relevant tracking line optimization, the estimated vehicle outline is transformed and interpolated. The interpolated points for each respective tracking line & all points in $\mathcal{O}^{(q)}_{obs_{tot}}$ are used starting from finding the safest angular gap, through to the quadratic optimization (Equation 3.1.4). With the n_{MPC} tracking lines, STLMPC fulfills the same non-linear optimization as before, but now with tracking lines that perform dynamic obstacle avoidance. Figure 4.4 shows the generation of tracking lines using the tracked vehicle outline over successive segments of the future trajectory.

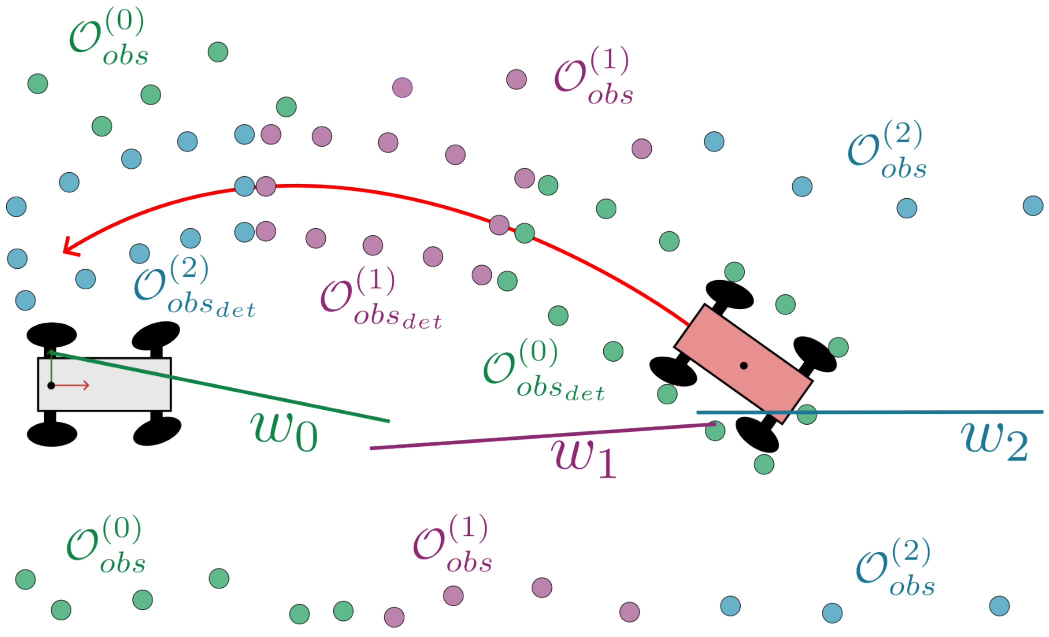
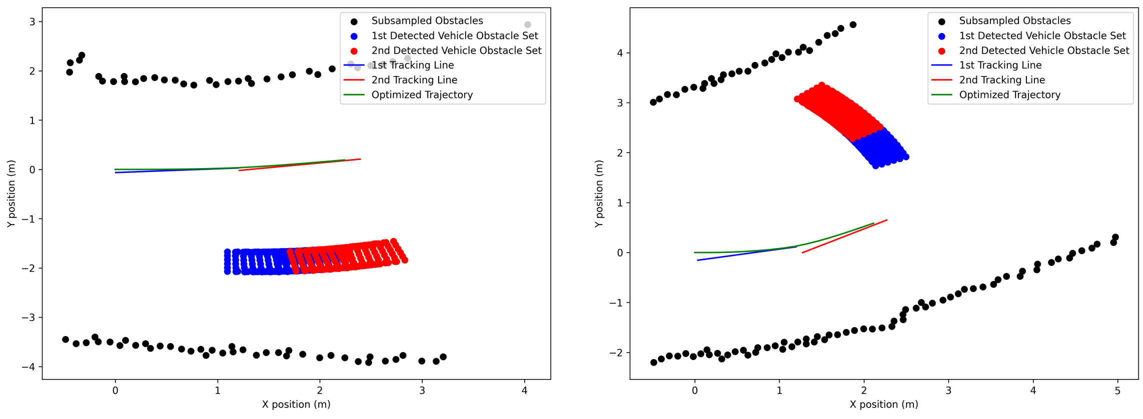

$\mathcal{O}_{obs}^{(0)}$
$\mathcal{O}_{obs}^{(1)}$
$\mathcal{O}_{obs}^{(2)}$
$\mathcal{O}_{obs_{det}}^{(2)}$
$\mathcal{O}_{obs_{det}}^{(1)}$
$\mathcal{O}_{obs_{det}}^{(0)}$
w_0
w_1
w_2
$\mathcal{O}_{obs}^{(0)}$
$\mathcal{O}_{obs}^{(1)}$
$\mathcal{O}_{obs}^{(2)}$

Figure 4.4: The detected (red) vehicle's trajectory is predicted and segmented into three intervals. The outline of the vehicle along the trajectory is converted into obstacles, $\mathcal{O}_{obs_{det}}^{(q)}$ which, when added to the static obstacles, $\mathcal{O}_{obs}^{(q)}$ generate the three successive ideal tracking lines.

4.5 Simulation Results

The *f1tenth_simulator* environment is again used, following the same setup as presented in Chapter 3 but now for a new map and accounting for a simulated, detected vehicle. Camera-based vehicle detection is not used here, so instead, the adversarial vehicle pose is directly used as the measurement at each control step in the EKF, enabling vehicle tracking. Some relevant parameters in addition to those in Chapter 3 include $\mathrm{l_v}$=0.5 m, $\mathrm{w_v}$=0.4 m, $\sigma_{x_0}^2 = \sigma_{y_0}^2 = 0.01\ \mathrm{m}^2, \sigma_{\theta_0}^2 = \sigma_{\delta_0}^2 = (\frac{\pi}{36})^2\ \mathrm{rad}^2,\ \sigma_{v_0}^2 = 0.05\ \frac{\mathrm{m}^2}{\mathrm{s}^2}$, $N_{miss}^{max} = 5$ & $n_{edge} = 5$. The successive tracking lines and optimized path are shown in the simulated environment at two sample times (Figure 4.5). Here, dynamic obstacle avoidance is clear, as the detected vehicle trajectory influences tracking line generation, constructing reference and ensuing optimized paths that avoid collisions.



(a) (b)

Figure 4.5: Local path planning in simulation at two samples, considering dynamic obstacle avoidance. The detected vehicle shape is projected over its future trajectory and used as obstacles for generating the successive tracking lines.

The detected vehicle trajectory is used sequentially, where the first half of the estimated path is used to construct the first tracking line, with the second half being used for the second line. Figure 4.5a shows the case of overtaking the detected vehicle while maintaining a central, safe path. On the other hand, Figure 4.5b illustrates a detected vehicle that crosses in front of the ego vehicle, forcing a maneuver behind the detected vehicle's predicted trajectory to avoid a collision.

Evaluation of the local planners in a full simulated test case is shown on a new map (Map #2) in Figure 4.6 for Chapter 3's STLMPC and TEB which only consider obstacles statically here as well as Chapter 4's STLMPC which considers detected vehicle obstacles dynamically through prediction. TEB is chosen for evaluation as it is the most effective exploration planner based on earlier results. The paths are mostly the same until the detected vehicle is encountered, at which point static obstacle STLMPC—and even more so, dynamic obstacle STLMPC—swerves opposite to the detected vehicle's motion to prevent collision. The positions of the vehicle using each planner are shown at an instance before the detected vehicle is avoided, where dynamic obstacle STLMPC maintains the safest path in the presence of the adversarial vehicle.

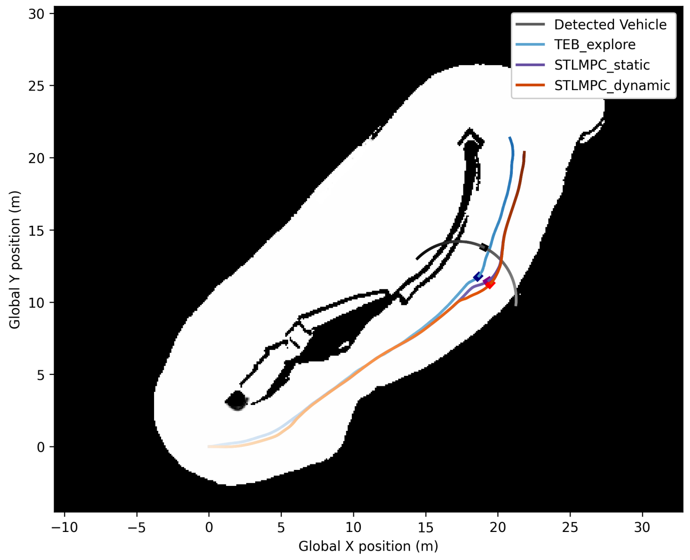


Figure 4.6: Local planner trajectories for dynamic obstacle avoidance in simulated Map #2. Darker color gradients for each path show time progression, while one shared time sample before vehicle avoidance is indicated by a point on each path.

The performance of each approach in this simulation is quantified in Table 4.1. Static & dynamic obstacle STLMPC perform comparably with marginal improvements in the dynamic approach. In this wide track, TEB performs similarly, although slightly inferior to the STLMPC approaches in terms of safety and control effort.

Table 4.1: Performance of local planners including STLMPC for dynamic obstacle avoidance in simulated Map #2

Local Planner	$\min_{\tilde{k}} d_{min,\tilde{k}}$ (m)	$\bar{d}_{min}$ (m)	$\overline{\lvert\delta_{cmd,\tilde{k}}\rvert}$ (rad)	$\mathrm{Var}(\delta_{cmd,\tilde{k}})$ (rad^2)	$\bar{v}_{cmd,\tilde{k}}$ (m/s)	$\mathrm{Var}(v_{cmd,\tilde{k}})$ (m^2/s^2)
TEB_explore	1.491	2.081	0.064	0.008	1.472	0.034
STLMPC_static	1.561	2.284	0.049	0.007	1.482	0.015
STLMPC_dynamic	1.596	2.258	0.048	0.006	1.482	0.015

4.6 Experimental Results

Vehicle detection, tracking and avoidance are now evaluated in real-world conditions using Experiment #2's layout (Appendix B). The use of RGB & depth camera data and a YOLO-based model for detection presents multiple challenges to effective dynamic obstacle avoidance through both noisy sensor data and imperfect vehicle detection. Additionally, the fact that only the position is observable leads to difficulties in accurately tracking the orientation, steering angle and velocity states. Thus, the covariance values used for initialization, as well as process and measurement noises, are increased. Some values modified from use in simulation include $\sigma_{x_0}^2 = \sigma_{y_0}^2 = 0.05\ \mathrm{m}^2, \sigma_{\theta_0}^2 = \sigma_{\delta_0}^2 = (\frac{\pi}{18})^2\ \mathrm{rad}^2,\ \&\ \sigma_{v_0}^2 = 0.15\ \frac{\mathrm{m}^2}{\mathrm{s}^2}$.

Vehicle detection and tracking are assessed while the ego vehicle is stationary, where the tracked trajectory is obtained & compared to the true path of the detected vehicle (Figure 4.7).

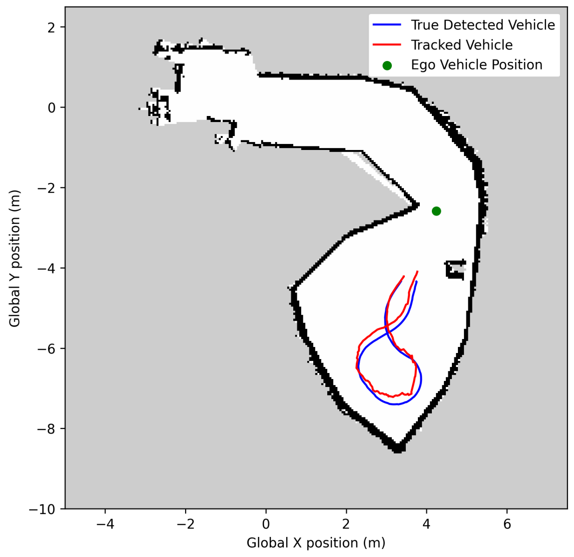
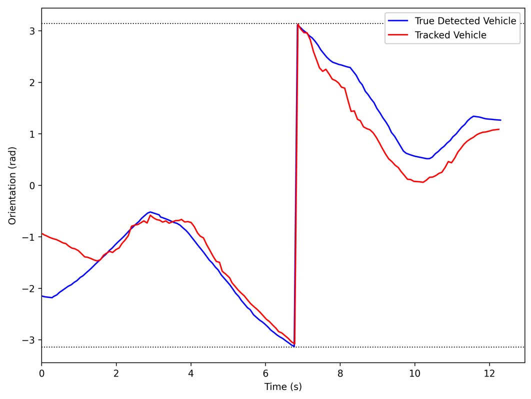
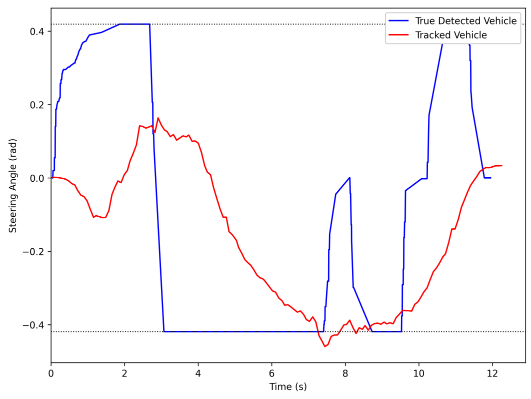


Figure 4.7: True detected vehicle and EKF estimated trajectories in Experiment #2. The ego vehicle is stationary at the given position while the detected vehicle drives in a clockwise loop starting from the point (3.36, -4.34).

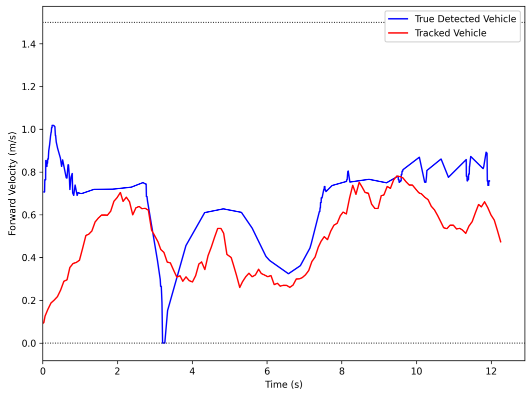
The estimated trajectory is noisy, although closely accurate both in time and space to the true path, owing to the direct observability from depth camera measurements. Additionally, the unobservable states are evaluated over time (Figures 4.8a, 4.8b & 4.8c) as well as the determinant of the covariance matrix, illustrating confidence (Figure 4.8d). The orientation and velocity states are closely tracked, although the steering angle is estimated more poorly, where sharp changes in turning go undetected. The confidence decreases for times of higher steering angles while sudden spikes occur when detections are missed and the EKF correction step is forgone.



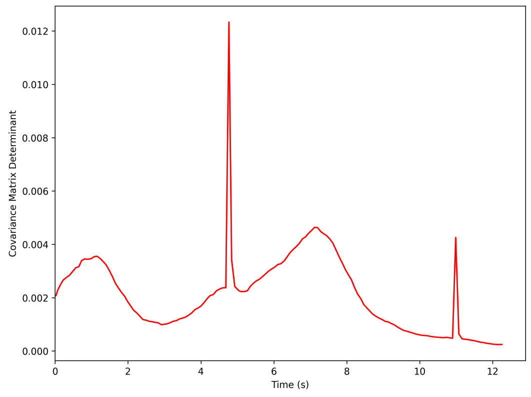
(a)



(b)



(c)



(d)

Figure 4.8: The true detected vehicle and tracked unobservable states are recorded over time for (a) orientation, (b) steering angle & (c) velocity. The (d) covariance matrix determinant is obtained from EKF tracking to quantify confidence over time.

Dynamic obstacle/vehicle avoidance is tested in Experiment #2, where the performance of STLMPC is recorded and provided by video[2]. STLMPC is evaluated both with and without dynamic obstacle avoidance, while TEB-based exploration is additionally assessed where obstacles are only considered statically. Each planner's trajectory is provided in Figure 4.9 alongside that of the detected vehicle, where the TEB-based planner takes minimal action to avoid the adversarial vehicle. Here, Chapter 3's STLMPC swerves around the moving vehicle while this chapter's STLMPC makes the most significant maneuver to avoid the detected & tracked vehicle, ensuring safety. At a shared time sample before the detected vehicle is avoided, it is evident how dynamic obstacle STLMPC maintains the greatest clearance from the oncoming detected vehicle.

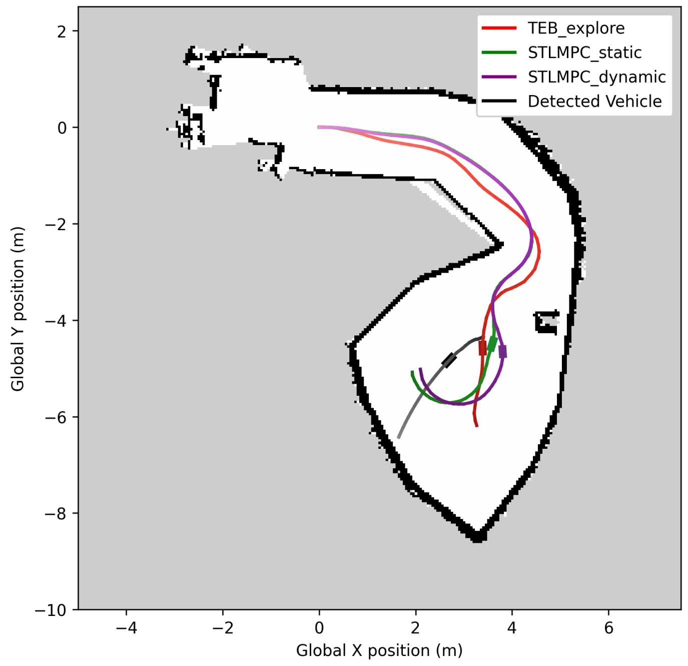


Figure 4.9: Local planner trajectories for dynamic obstacle avoidance in Experiment #2. Darker color gradients for each path show time progression, while one shared time sample before vehicle avoidance is indicated by a point on each path.

[2]`https://www.youtube.com/watch?v=uKgcKcMBytk`

The performance metrics for each tested planning algorithm are shown in Table 4.2. Dynamic obstacle STLMPC performs best in terms of obstacle proximity, with static obstacle STLMPC performing similarly, differing only at the point of dynamic vehicle avoidance. All planners exhibit similar control efforts, where the additional vehicle avoidance maneuver leads to slightly larger average steering angle magnitudes for the STLMPC methods.

Table 4.2: Performance of local planners including STLMPC for dynamic obstacle avoidance in Experiment #2

Local Planner	$\min_{\tilde{k}} d_{min,\tilde{k}}$ (m)	$\bar{d}_{min}$ (m)	$\overline{\lvert\delta_{cmd,\tilde{k}}\rvert}$ (rad)	$\mathrm{Var}(\delta_{cmd,\tilde{k}})$ (rad^2)	$\bar{v}_{cmd,\tilde{k}}$ (m/s)	$\mathrm{Var}(v_{cmd,\tilde{k}})$ (m^2/s^2)
TEB_explore	0.487	0.846	0.181	0.050	1.249	0.142
STLMPC_static	0.531	0.908	0.195	0.048	1.211	0.069
STLMPC_dynamic	0.562	0.910	0.196	0.045	1.234	0.053

Computation times for each planner are given in Table 4.3, where each planner achieves similar performance. Average computation times are low for all methods, while static obstacle STLMPC achieves the lowest worst-case time. Dynamic obstacle STLMPC has the poorest worst-case time, which occurs around the time of detected vehicle avoidance, as the local path optimization becomes more difficult. This time is still within the 100-ms control period, maintaining on-time vehicle actuation.

Table 4.3: Planning computation times for local planners including dynamic obstacle STLMPC in Experiment #2

Local Planner	$\bar{t}_{frontier}$ (ms)	$\bar{t}_{global}$ (ms)	$\bar{t}_{local}$ (ms)	$\boldsymbol{\bar{t}_{comp}}$ **(ms)**	$\max t_{frontier}$ (ms)	$\max t_{global}$ (ms)	$\max t_{local}$ (ms)	$\mathbf{max}\, \boldsymbol{t_{comp}}$ **(ms)**
TEB_explore	1.7	0.9	3.7	**6.3**	2.5	3.4	13.6	**19.6**
STLMPC_static				**6.4**				**12.7**
STLMPC_dynamic				**6.8**				**42.3**

Chapter 5

Non-Uniform Velocity STLMPC

The aforementioned STLMPC algorithm assumes a fixed velocity over the future predicted trajectory and, moreover, over the course of the vehicle's operation. This can be restrictive and potentially problematic in certain navigation circumstances. By introducing an additional degree of freedom in the STLMPC optimization, more flexible navigation can be achieved while meeting limitations to ensure safe, environment-aware velocities. Higher, safe velocities are prioritized, which makes racing a natural application of this new path planning algorithm.

This chapter presents the extension to STLMPC by first introducing the differences in the approach. The modified objective function and additional velocity-dependent constraints are then presented. Finally, analysis is conducted, showing performance compared to the original STLMPC method in both simulation and experimental tests.

5.1 Modifications to STLMPC

The non-uniform velocity STLMPC approach draws on the same framework as the approach outlined in Chapter 3 (with extensions in Chapter 4), except now, adding velocity considerations. The key differences compared to the original STLMPC process are outlined as follows:

- The ego vehicle states & therefore optimization variables become $[x_i, y_i, \theta_i, \delta_i, v_i]$ $\forall i \in \{0, ..., n_{MPC}k_{MPC} - 1\}$, thereby denoting a total of $5n_{MPC}k_{MPC}$ optimization variables.
- The objective function has an added velocity-dependent term that prioritizes higher velocities over the future horizon.
- Existing constraints now use v_i in place of v; the same follows for the ensuing gradients.
- Limits on velocity magnitude as well as acceleration are implemented.
- A velocity limiting function is constructed that varies with the steering angle. This ensures that during tighter turns, the vehicle speed is reduced, which maintains safety, reducing the chances of skidding and, in hypothetical extreme cases, rollover.
- Velocity is limited by proximity to nearby obstacles in the forward direction. A function that performs bandpass filtering is applied to obstacles, where the soft minimum distance to objects directly ahead is obtained. This minimum distance is used for vehicle slowdown & braking to prevent collisions and preserve safe speeds in tighter environments.

- The current vehicle speed is obtained through the VESC and is used as v_0, which forms the initial state of the predicted trajectory through MPC.
- The optimization's initial guess now requires a selection of initialized velocity variables, which here are assumed constant for simplicity.
- The first free velocity variable from the optimization solution, v_1, is applied to the vehicle for navigation (v_{cmd}) in tangent with the steering angle.

5.2 Augmented Objective Function

For racing applications specifically, fast navigation at higher velocities is promoted. This can transfer to more general scenarios where the fastest speed is encouraged, while still maintaining safety as defined by subsequent metrics. To encourage quicker paths, an additional velocity-dependent term, $F_{v^2}(v_i)$ is introduced into the STLMPC objective function:

$$F_{v^2}(v_i) = \sum_{i=0}^{n_{MPC}k_{MPC}-1} \frac{1}{v_i^2} \tag{5.2.1}$$

This term takes a similar form to the squared sum of steering angles term (Equation 3.1.17) except now, higher velocities are encouraged, so the reciprocal of each squared velocity is summed and minimized. This additional term, $F_{v^2}(v_i)$ with weight λ_{v^2} is added to the multi-term objective with Chapter 3's quadratic form terms (Equation 3.1.11), where each is weighted according to its relative significance:

$$F_{obj_v}(\delta_i, v_i) = \lambda_{d^2} F_{d^2}(\delta_i, v_i) + \lambda_{\dot{d}^2} F_{\dot{d}^2}(\delta_i, v_i) + \lambda_{\delta^2} F_{\delta^2}(\delta_i) + \lambda_{v^2} F_{v^2}(v_i) \tag{5.2.2}$$

Here, only three weights are required if normalized to remove a parameter.

The final objective function is denoted by $F_{obj_v}(\delta_i, v_i)$, which is a function of the control inputs, both δ_i and v_i. The weighting emphasis placed on the new velocity term compared to the original STLMPC terms determines how much higher speeds are prioritized at the cost of poorer tracking line following and higher steering angle control effort. The pertinent gradients for the augmented objective function, which differ from those of the original STLMPC approach, are detailed in Appendix A.2.

5.3 Additional Velocity-Based Constraints

While pursuing quicker velocity paths, the increased flexibility in the optimization means behavior can be further tuned/constrained to meet specific criteria. The existing STLMPC constraints and corresponding gradients now use the varying v_i (Appendix A.2) instead of the constant v. Limits on the velocity variables come from both physical limits on vehicle dynamics as well as safety considerations. In terms of limits on vehicle dynamics, the maximum (v_{max}) and minimum (v_{min}) bounds on allowed velocities for the particular vehicle are used:

$$v_{min} \leq v_i \leq v_{max} \quad \forall i \in \{0, ..., n_{MPC}k_{MPC} - 1\} \tag{5.3.1}$$

To prevent backwards motion, v_{min} is set to 0 but can be suitably increased to force forward motion or decreased to allow reverse driving in a potential future extension of STLMPC (although the singularity in the velocity-dependent objective term would need to be considered).

Linear inequality constraints are developed from the maximum change in consecutive applied velocities over the future MPC horizon. Here, the maximum acceleration denoted by Δv_{max} and similarly, deceleration denoted by $-\Delta v_{max}$ are used to form the constraints, $g_{v^+,i}$ & $g_{v^-,i}$ according to:

$$-\Delta t\,\Delta v_{max} \leq v_{i+1} - v_i \leq \Delta t\,\Delta v_{max} \quad \forall i \in \{0, ..., n_{MPC}k_{MPC} - 2\}$$

$$g_{v^+,i} = v_{i+1} - v_i - \Delta t\,\Delta v_{max}, \quad g_{v^+,i} \leq 0 \tag{5.3.2}$$

$$g_{v^-,i} = -v_{i+1} + v_i - \Delta t\,\Delta v_{max}, \quad g_{v^-,i} \leq 0 \tag{5.3.3}$$

The velocity applied to the vehicle in the last sample's MPC optimization is given by v_{last}. Just as with the steering angle, the DC motor for the velocity control input takes time to transition to the next commanded velocity. The velocity command, v_{cmd}, is therefore applied a sample in advance to ensure that by the next control step, it has reached this state per the predicted MPC path. So, the current initial velocity is fixed by the last command (confirmed in practice through the VESC's updated reading), $v_0 = v_{last}$, while the command applied at the current time is the first optimized control input not fixed, $v_{cmd} = v_1$.

5.3.1 Turn-Rate-Dependent Velocity Limits

To ensure vehicle safety is preserved, especially when path planning at higher velocities, constraints are introduced on velocity during turns. If turning with a high instantaneous steering angle, $|\delta_i|$, a high instantaneous velocity, v_i, may cause the vehicle to lose control, skid, or, in the worst case, rollover. Thus, a velocity-limiting function, $f_{v_{steer}}$, is introduced, which sets the maximum allowable velocity based on the current steering angle magnitude. Instead of expressing magnitude as an absolute value, a reciprocal quadratic steering angle expression is used to ensure smooth gradients:

$$f_{v_{steer,i}} = \frac{v_{max}}{1 + (\frac{\delta_i}{\delta_{max}})^2} \tag{5.3.4}$$

This maximum allowable velocity function is shown in Figure 5.1 where the limit decreases from v_{max} when the turning angle is zero to $\frac{v_{max}}{2}$ when the steering angle magnitude equals $|\delta_{max}|$. Using $f_{v_{steer,i}}$, inequality constraints ($g_{v_{steer,i}}$) can be formulated for each sampled velocity to ensure the safe maximum velocity function is not exceeded for the current steering angle:

$$\begin{gathered} v_i \leq f_{v_{steer,i}} \quad \forall i \in \{0, ..., n_{MPC}k_{MPC} - 1\} \\ g_{v_{steer,i}} = v_i - \frac{v_{max}}{1 + (\frac{\delta_i}{\delta_{max}})^2}, \quad g_{v_{steer,i}} \leq 0 \end{gathered} \tag{5.3.5}$$

These additional $n_{MPC}k_{MPC}$ inequality constraints are incorporated in the optimization with analytical gradients as presented in Appendix A.2.

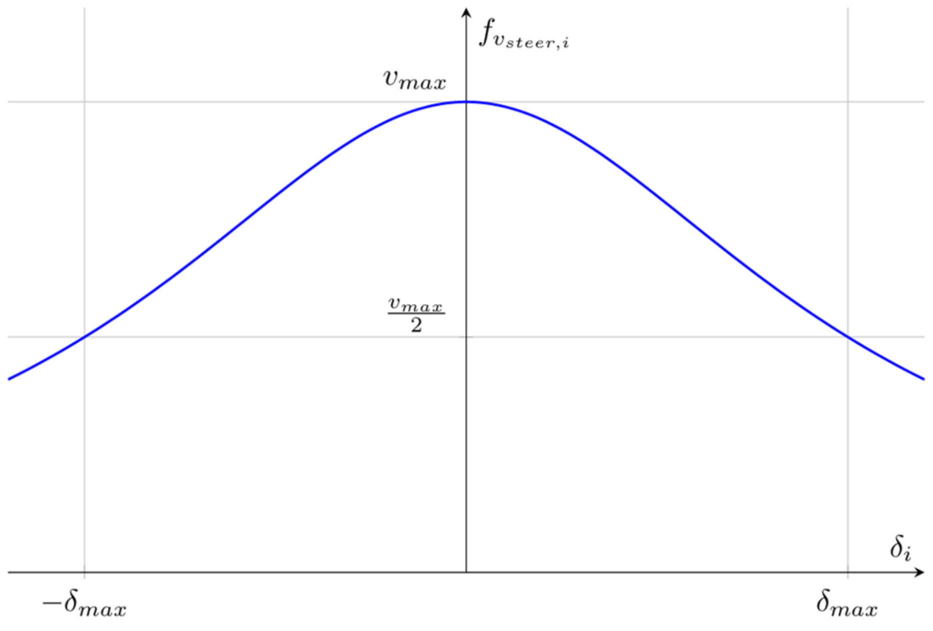


Figure 5.1: The velocity-limiting function, dependent on the current steering angle. At zero turning angle, the maximum velocity is allowed; meanwhile, at the maximum turning angle, only half the maximum velocity is permitted.

5.3.2 Vehicle Slowdown via Obstacle Proximity

To handle tight spaces with collision risks in close proximity to the vehicle, velocity is reduced to maintain control. Particularly, hazards in front of the agent present immediate dangers where the vehicle must either swerve around or brake before collision. A safe stopping distance is identified to slow the vehicle down and prevent head-on collisions while considering the vehicle's maximum deceleration. This allows the vehicle to maneuver safely in crowded spaces while coming to a stop and avoiding collisions in the worst case, if no safe path exists.

Using all obstacles in the vehicle base frame, the set is subsampled to maintain computational tractability by only using obstacles with a separation distance exceeding d_{sep} to all other subsampled obstacles:

$$\mathcal{O}_{obs_{sub}} = \{(x_{obs_{tot},i}, y_{obs_{tot},i}) \in \mathbb{R}^2 \mid \|(x_{obs_{tot},i}, y_{obs_{tot},i}) - (x_{obs_{sub},j}, y_{obs_{sub},j})\|_2 \geq d_{sep}\} \tag{5.3.6}$$

where the subsampled set is expanded iteratively by considering all remaining obstacle distances in the original set to the subsampled set ($\forall i \in \{0, ..., N_{obs_{tot}} - 1\}$ and for expanding $N_{obs_{sub}}$, $j \in \{0, ..., N_{obs_{sub}} - 1\}$). Here, d_{sep} is set low but can be increased if the number of subsampled obstacles remains too high. For each obstacle in the subset $\{(x_{obs_{sub},j}, y_{obs_{sub},j})\}_{j=0}^{N_{obs_{sub}}-1}$, the distance and heading angle between each predicted vehicle state ($x_i, y_i \,\&\, \theta_i \;\forall i \in \{0, ..., n_{MPC}k_{MPC} - 1\}$) and obstacle is considered. The distance between the vehicle at sample i and the j^{th} obstacle is given by:

$$d_{i,j} = \sqrt{(x_i - x_{obs_{sub},j})^2 + (y_i - y_{obs_{sub},j})^2} \tag{5.3.7}$$

where the heading angle in the base frame is:

$$\theta_{i,j} = \text{atan2}(y_{obs_{sub},j} - y_i, x_{obs_{sub},j} - x_i) \tag{5.3.8}$$

However, the predicted vehicle state has orientation in the base frame of θ_i, so the obstacle heading angle is found with respect to this orientation instead:

$$\theta_{\text{diff},i,j} = \text{atan2}(\sin(\theta_{i,j} - \theta_i), \cos(\theta_{i,j} - \theta_i)) \tag{5.3.9}$$

where this equation wraps the angular difference correctly to the normalized range, $(-\pi, \pi]$. To consider only hazardous obstacles ahead of the vehicle during each future sample (Figure 5.2), a passband is constructed by $-\theta_{\text{band}_{\text{max}}} \leq \theta_{\text{diff},i,j} \leq \theta_{\text{band}_{\text{max}}}$ where $\theta_{\text{band}_{\text{max}}}$ is the maximum angular difference at which an obstacle is considered. The distance from the vehicle to the nearest obstacle in the passband ($d_{\text{band}_{\text{min}}}$) is used in a velocity limiting function to slow the vehicle down when objects are in close proximity, directly ahead.

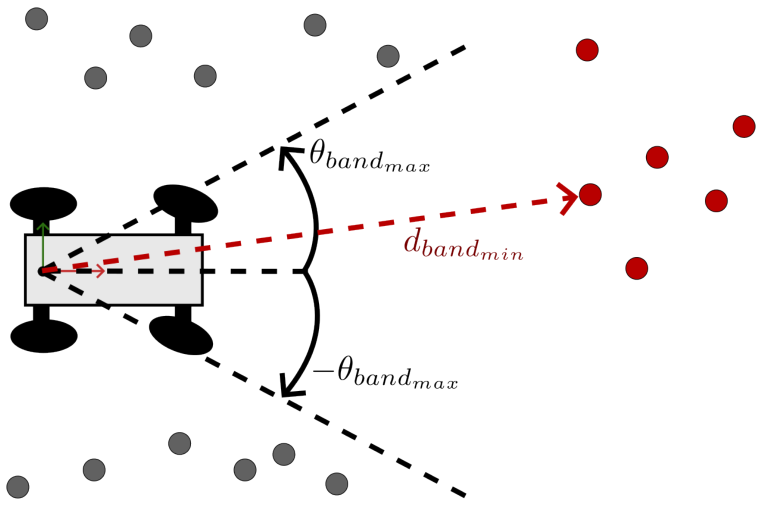


Figure 5.2: Passband obstacles (red) with the minimum obstacle proximity, $d_{\text{band}_{\text{min}}}$.

To consider only obstacles in the vehicle's forward FOV, a smooth bandpass function is created (Figure 5.3a) which ensures analytical gradients can be found. This function takes the form:

$$\theta_{\text{band},i,j} = \frac{1}{1 + e^{-s_\theta(\theta_{\text{diff},i,j} + \theta_{\text{band}_{\text{max}}})}} - \frac{1}{1 + e^{-s_\theta(\theta_{\text{diff},i,j} - \theta_{\text{band}_{\text{max}}})}} \tag{5.3.10}$$

with transition steepness between the passband and stopband denoted by the parameter s_θ. Obstacles with relative heading angles in the passband have $\theta_{\text{band},i,j} \approx 1$ while those in the stopband have $\theta_{\text{band},i,j} \approx 0$. Now, a weighted softmin function (Figure 5.3c) is used to get the approximate minimum obstacle distance at sample i, $d_{\text{band}_{\text{min}},i}$, considering all obstacles in the passband:

$$d_{\text{band}_{\text{min}},i} = -\frac{1}{\beta_v} \log\Big(\sum_{j=0}^{N_{obs_{sub}}-1} \theta_{\text{band},i,j}\, e^{-\beta_v d_{i,j}}\Big) \tag{5.3.11}$$

The sharpness of the approximation is controlled by β_v, where higher values more accurately model the true minimum but with poorer numerical stability. Using the softmin function as a smooth approximation ensures continuous gradients and numerical stability for the optimization solver.

The final velocity-limiting function (Figure 5.3b) based on obstacle proximity, $f_{v_{obs,i}}$ is represented by an exponential term, dependent on $d_{\text{band}_{\text{min}},i}$:

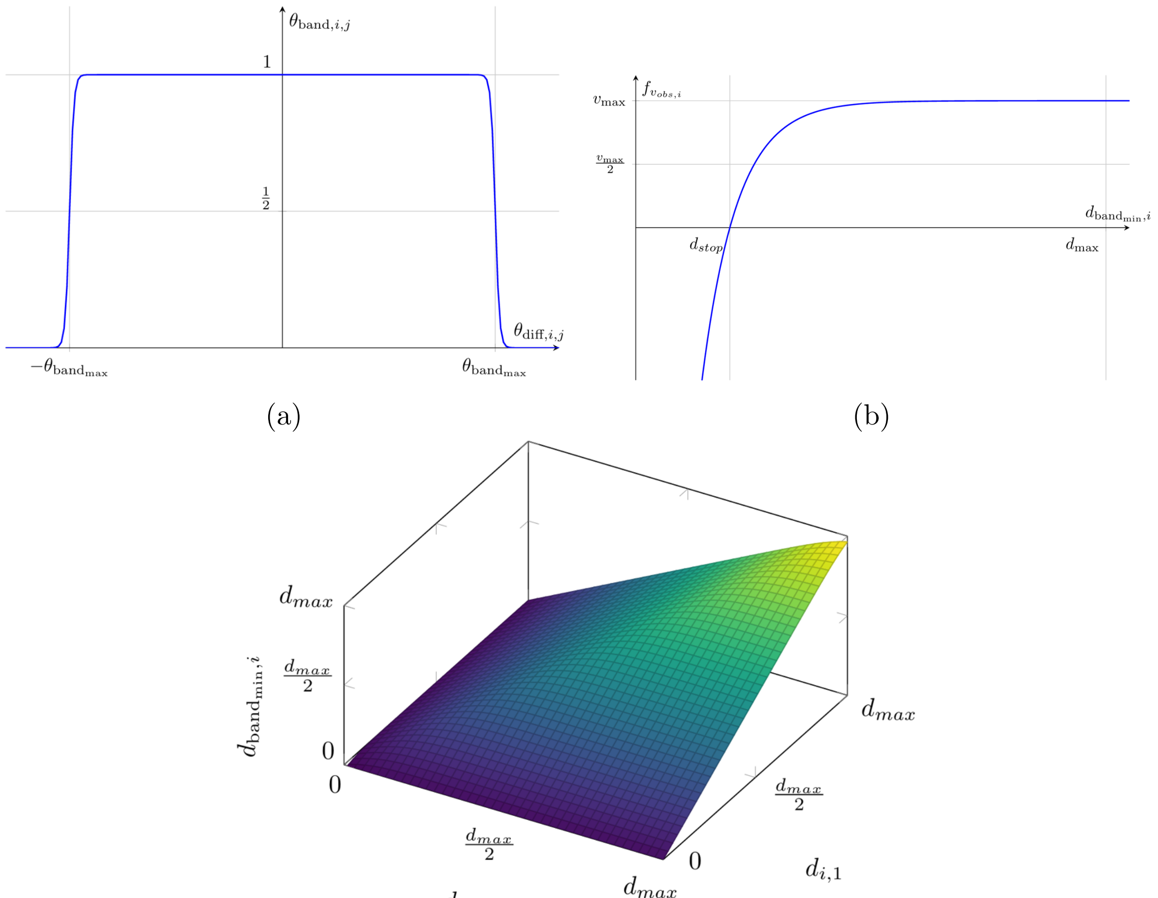
$$f_{v_{obs,i}} = v_{max}\Big(1 - e^{-\frac{d_{\text{band}_{\text{min}},i} - d_{stop}}{\alpha_v}}\Big) \tag{5.3.12}$$

Here, d_{stop} denotes the obstacle distance where the vehicle is forced to stop, and the decay rate, α_v, controls how sharp the velocity limit decrease is as $d_{\text{band}_{\text{min}},i}$ is reduced.

Finally, inequality constraints are constructed ($g_{v_{obs,i}}$) for each sample to limit the velocity variables according to $f_{v_{obs,i}}$:

$$v_i \leq f_{v_{obs,i}} \quad \forall i \in \{0, ..., n_{MPC}k_{MPC} - 1\}$$
$$g_{v_{obs},i} = v_i - v_{max}(1 - e^{-\frac{d_{\text{band}_{\text{min}},i} - d_{stop}}{\alpha_v}}), \quad g_{v_{obs},i} \leq 0 \tag{5.3.13}$$

where corresponding gradients are provided in Appendix A.2.



(c)

Figure 5.3: Function shapes for: (a) the heading angle passband, (b) the velocity-limiting function based on obstacle proximity & (c) the softmin function assuming only two distances where both $\theta_{\text{band},i,0}$ and $\theta_{\text{band},i,1} = 1$.

5.4 Final Optimization Problem Formulation

The ensuing optimization problem makes the necessary alterations to Chapter 3's STLMPC optimization (Equation 3.1.28), thus formulating:

$$\begin{aligned}
\min_{x_j,y_j,\theta_j,\delta_j,v_j} \quad & \lambda_{d^2}F_{d^2} + \lambda_{\dot{d}^2}F_{\dot{d}^2} + \lambda_{\delta^2}F_{\delta^2} + \lambda_{v^2}F_{v^2} \\
\text{subject to} \quad & g_{\delta^+,i} \leq 0, \quad g_{\delta^-,i} \leq 0, \quad g_{v^+,i} \leq 0, \quad g_{v^-,i} \leq 0 \\
& g_{v_{steer},j} \leq 0, \quad g_{v_{obs},j} \leq 0 \\
& h_{x,i} = 0, \quad h_{y,i} = 0, \quad h_{\theta,i} = 0 \\
& -\delta_{max} \leq \delta_j \leq \delta_{max}, \quad v_{min} \leq v_j \leq v_{max} \\
& x_0 = 0, \quad y_0 = 0, \quad \theta_0 = 0, \quad \delta_0 = \delta_{last}, \quad v_0 = v_{last} \\
& \forall i \in \{0, ..., n_{MPC}k_{MPC} - 2\}, \quad \forall j \in \{0, ..., n_{MPC}k_{MPC} - 1\}
\end{aligned} \tag{5.4.1}$$

where all existing instances of v in STLMPC are replaced with the respective optimization variable, v_j. The initial guess for the optimization follows the framework detailed in Algorithm 2 where v_j replaces v and each velocity variable is initialized to a constant value for simplicity, such as v_{last} or $\frac{v_{max}}{2}$.

With higher velocities promoted if safe, the solver will provide a solution with a varying, non-constant velocity over the predicted path's time horizon. As in the original STMPC approach, SLSQP is used as the optimization solver and the optimized path is provided by $[x_j, y_j, \theta_j, \delta_j, v_j]$, $j \in \{0, ..., n_{MPC}k_{MPC} - 1\}$. The first free steering angle and velocity variables, δ_1 & v_1 are applied as δ_{cmd} & v_{cmd} before the process is repeated at the next control step (similar to the process in Algorithm 3).

5.5 Simulation Results

This new formulation is now evaluated in simulation on a third map (Map #3) with existing parameters repeated from prior simulations. Some parameter values specific to this chapter include $\lambda_{v^2} = 1$, v_{min} =0 m/s, v_{max} =3 m/s, Δv_{max} =2.5 m/s^2, $\theta_{\text{band}_{\max}} = \frac{\pi}{8}$ rad, s_θ =200 rad^{-1}, β_v=10 m^{-1}, d_{stop} =0.8 m & α_v =0.5 m. The nominal velocity remains 1.5 m/s from previous simulations for the other tested methods (Chapter 3's STLMPC & exploration via TEB) as well as for this chapter's initial guess to maintain feasibility. However, the maximum allowed speed of 3 m/s allows this chapter's approach to achieve faster speeds when safe. For lower steering angle magnitudes, higher velocities at the corresponding samples are allowed beyond $\frac{v_{max}}{2}$.

The three local planner trajectories in Map #3 are shown in Figure 5.4, where navigation occurs around the track clockwise. The variable-velocity STLMPC approach achieves a faster course time than the other methods while maintaining a velocity that satisfies all safety constraints, thereby ensuring lower velocities during sharp turns and vehicle slowdown before the course's dead-end wall.

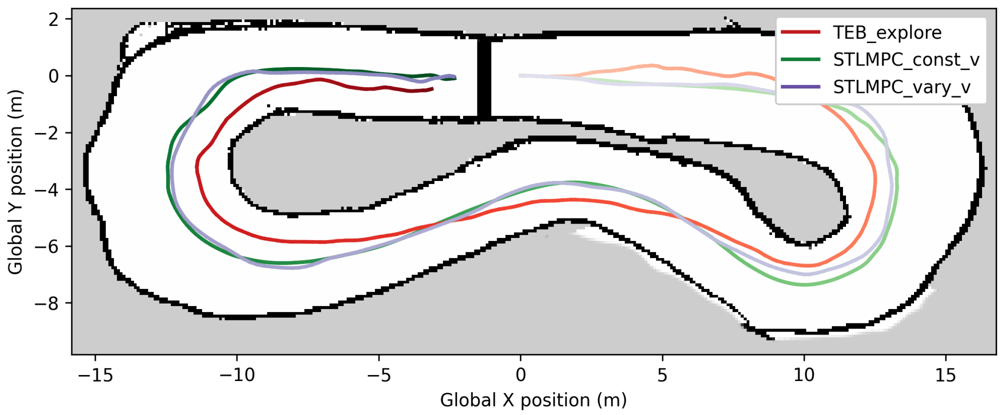


Figure 5.4: Local planner trajectories including for non-uniform velocity STLMPC in simulated Map #3. Darker color gradients for each path show time progression, where variable-velocity STLMPC completes the course significantly faster and is therefore lighter throughout.

Table 5.1: Performance of variable-velocity STLMPC & other local planners in simulated Map #3

Local Planner	$\min_{\tilde{k}} d_{min,\tilde{k}}$ (m)	$\bar{d}_{min}$ (m)	$\overline{\lvert\delta_{cmd,\tilde{k}}\rvert}$ (rad)	$\mathrm{Var}(\delta_{cmd,\tilde{k}})$ (rad^2)	$\bar{v}_{cmd,\tilde{k}}$ (m/s)	$\mathrm{Var}(v_{cmd,\tilde{k}})$ (m^2/s^2)	t_{tot} (s)
TEB_explore	0.521	1.076	0.107	0.020	1.360	0.097	41.3
STLMPC_const_v	1.070	1.591	0.079	0.014	1.452	0.030	40.7
STLMPC_vary_v	0.957	1.589	0.101	0.017	2.288	0.357	26.2

The speed improvement is shown for variable-velocity STLMPC in Table 5.1 where t_{tot} indicates the total time to complete the track. Here, each STLMPC method achieves similar safety improvements over TEB-based exploration, while this chapter's STLMPC has a higher velocity variance since it is now optimized and required to quickly change to satisfy constraints. These rapid fluctuations are shown in Figure 5.5 where velocities above 1.5 m/s must satisfy the steering angle-based constraint, which is only guaranteed for the other planners below 1.5 m/s. A gradual reduction in velocity upon course completion is evident in this chapter's approach.

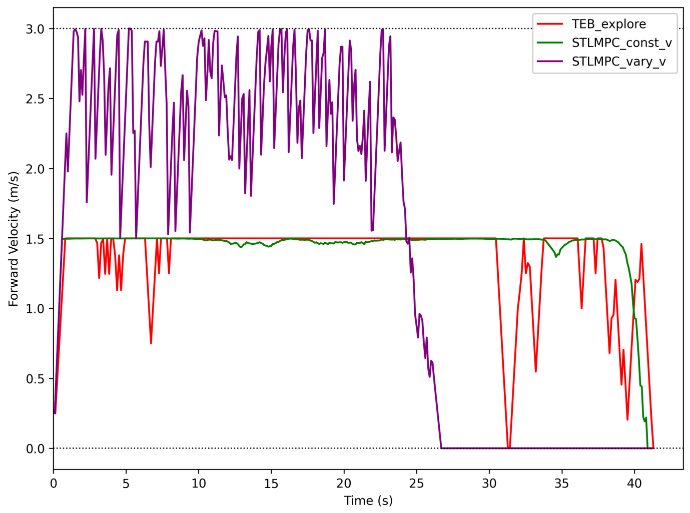


Figure 5.5: Velocity control inputs for each local planner in simulation using Map #3. Dotted bounds indicate the vehicle's minimum and maximum allowed velocities.

5.6 Experimental Results

Vehicle racing using velocity-varying STLMPC is tested in real-world circumstances via the layout constructed in Experiment #3 (Appendix B). The track contains tight corners, meaning a large, constant velocity will lead to crashes and thus, a varying velocity is required, which reduces for turns and increases for straightaways. STLMPC parameters follow from simulation except now, $d_{safe} = 1.25$ m & $d_{stop} = 0.5$ m to accommodate the tighter track. This track poses challenges to a number of the tested local planners, where only a select few complete the course.

DWA- & MPC-based exploration proceed very slowly and repeatedly run into obstacles in this setup; therefore, they are not included in the following results. Furthermore, the PD approach fails to successfully navigate the first turn, even at low speeds, crashing early on in the experiment. The TEB-based planner fails at high speeds, but when the maximum permissible speed is reduced to 1 m/s, the vehicle is able to complete the course, although it encounters collisions at two specific track sections. Constant-velocity STLMPC at 3 m/s fails but succeeds at 1.5 m/s while variable-velocity STLMPC uses $v_{min} = 0$ m/s & $v_{max} = 3$ m/s as in simulation.

The performance of variable-velocity STLMPC is shown by video[1] as well as among all planners that complete the course in Figure 5.6. Variable-velocity STLMPC takes several corners tighter than constant-velocity STLMPC and sacrifices safety in some spots to achieve higher speeds and a faster track time. Operating at faster speeds means less reaction time and poorer maneuverability; however, adequate safety is maintained by the variable-velocity formulation's constraints, and a visibly safe path is produced. The TEB-based planner completes the course but collides along the left

[1]`https://www.youtube.com/watch?v=yKPFWdbwx-4`

track wall as well as at the top of the map due to the sharp turns within the course.

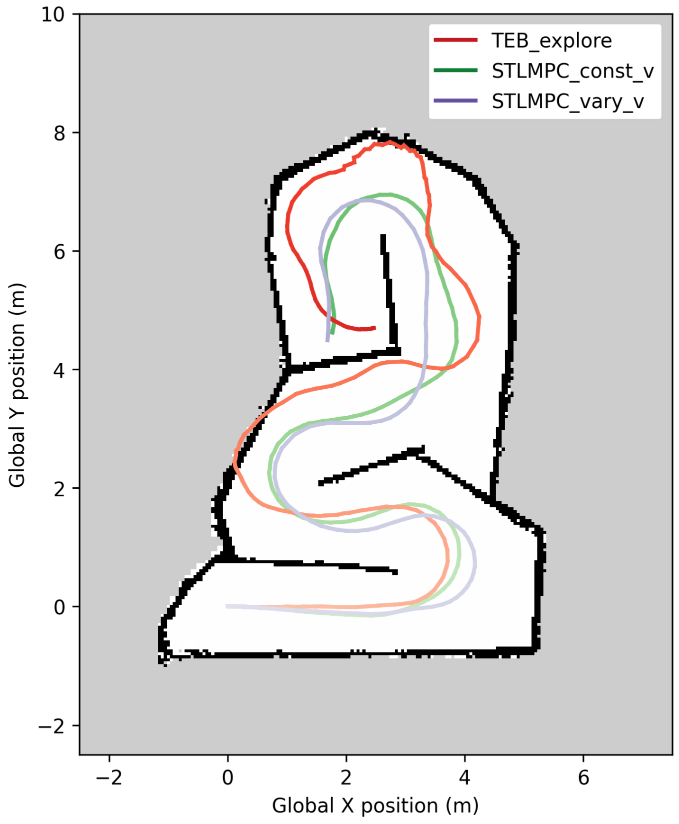


Figure 5.6: Local planner trajectories including for variable-velocity STLMPC in Experiment #3. Darker color gradients for each path show time progression, where variable-velocity STLMPC completes the course fastest and is therefore lightest.

The planners are evaluated further in Table 5.2, which presents proximity, control input and total track time metrics. The minimum obstacle proximity for the TEB planner is not applicable, as collision occurs, while variable-velocity STLMPC attains poorer proximity metrics compared to constant-velocity STLMPC due to more aggressive motion at higher speeds. However, velocity-varying STLMPC achieves the lowest steering angle average magnitude and variance due to its direct, fast motion on straight track sections. The variable-velocity method achieves the highest average velocity & fastest track time by increasing speed when safe during navigation.

Table 5.2: Performance of variable-velocity STLMPC & other local planners in Experiment #3

Local Planner	$\min_{\tilde{k}} d_{min,\tilde{k}}$ (m)	$\bar{d}_{min}$ (m)	$\overline{\lvert\delta_{cmd,\tilde{k}}\rvert}$ (rad)	Var($\delta_{cmd,\tilde{k}}$) (rad^2)	$\bar{v}_{cmd,\tilde{k}}$ (m/s)	Var($v_{cmd,\tilde{k}}$) (m^2/s^2)	t_{tot} (s)
TEB_explore	N/A	0.532	0.261	0.086	0.863	0.092	25.4
STLMPC_const_v	0.528	0.747	0.224	0.068	1.176	0.082	15.8
STLMPC_vary_v	0.403	0.720	0.196	0.057	1.456	0.331	12.8

Planning times are provided for each method (Table 5.3), where velocity-varying STLMPC has significantly longer times compared to STLMPC with a constant velocity. The additional velocity variables and constraints introduce added dimensionality to the optimization, increasing the complexity. Nonetheless, solutions are found reliably within the control period, while in the worst case, the solver terminates gracefully after timing out at the 50-ms limit. TEB struggles with the course's sharp turns, where the worst-case computational time is obtained, exceeding the control period.

Table 5.3: Planning computation times for variable-velocity STLMPC & other local planners in Experiment #3

Local Planner	$\bar{t}_{frontier}$ (ms)	$\bar{t}_{global}$ (ms)	$\bar{t}_{local}$ (ms)	$\boldsymbol{\bar{t}_{comp}}$ **(ms)**	$\max t_{frontier}$ (ms)	$\max t_{global}$ (ms)	$\max t_{local}$ (ms)	$\boldsymbol{\max t_{comp}}$ **(ms)**
TEB_explore	4.5	1.0	8.6	**14.1**	14.6	2.5	102.4	**119.6**
STLMPC_const_v				**6.7**				**18.9**
STLMPC_vary_v				**32.2**				**61.4**

Each planner's speed profile is detailed in Figure 5.7 over the course of Experiment #3. Variable-velocity STLMPC attains speeds over 2 m/s on straight track sections, while speeds approaching 3 m/s are not used, maintaining safety on the tight track. Constant-velocity STLMPC fluctuates in speed in practice, additionally incorporating a slowdown mechanism (external to the optimization) when nearing obstacles ahead. Both methods maintain high respective speeds compared to TEB, which slows greatly when obstacle collision arises, taking significantly longer to complete the track.

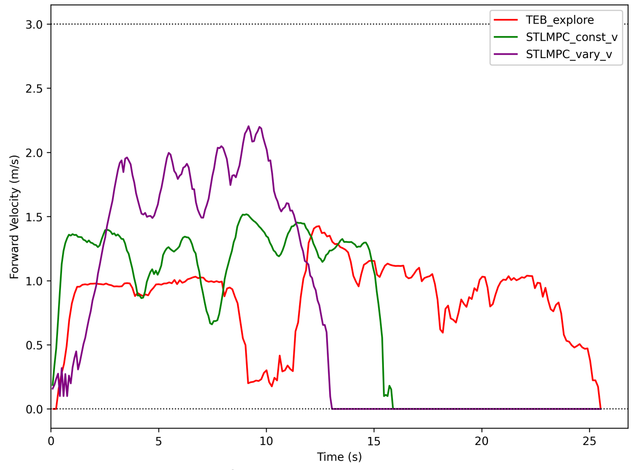


Figure 5.7: Velocity control inputs for each local planner in Experiment #3. Dotted bounds indicate the vehicle's minimum and maximum allowed velocities.

Finally, the variable-velocity STLMPC algorithm's speed profile is presented over time against the dynamic bounds used in the optimization's constraints (Figure 5.8). Velocity is bounded here by both a function that limits speeds when making sharp turns (Equation 5.3.5) as well as a slowdown function when approaching obstacles directly ahead (Equation 5.3.13). The speed profile successfully satisfies these bounds within the control limits while attaining the highest possible speeds for fast navigation.

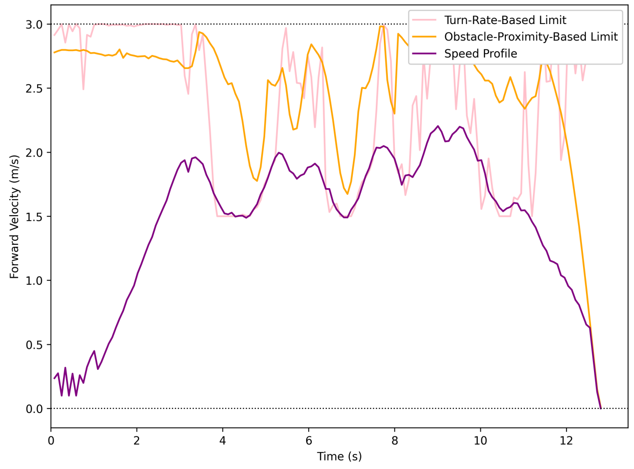


Figure 5.8: Speed profile for velocity-varying STLMPC in Experiment #3. The vehicle's forward velocity is limited at each step within control limits by functions of steering angle and forward obstacle proximity, used as constraints in optimization.

Chapter 6

Quartic Bezier Model Predictive Control

An alternative path planning method is proposed to STLMPC, entitled Quartic Bezier Model Predictive Control (QBMPC). This approach removes the need for tracking line generation & following, instead using potential fields from obstacles to achieve the optimal fourth-order Bezier curve path. Constraints on vehicle dynamics are incorporated directly into the curve's shape while maintaining a minimum distance to obstacles. The non-linear optimization proceeds for every control sample, attaining smoother predicted paths in less computation time compared to STLMPC.

The chapter follows by highlighting the inspiration for the QBMPC algorithm's development and the construction of the new path planner using Bezier curves. The algorithm is detailed, from steps that precede the optimization to the objective and constraint function configurations, concluding with vehicle control using the predicted path. Tests are then performed in both simulation and physical environments, comparing QBMPC performance to that of prior methods.

6.1 Motivation & Bezier Curve Composition

Previous chapters have explored the use of the STLPMC algorithm and some extensions therein. By approaching the problem of local path planning in unknown environments from a new perspective, the QBMPC method is created, which allows for comparisons between the algorithms and subsequent discussion. One facet of the STLMPC approach is that n_{MPC} quadratic optimizations are performed before the MPC optimization to acquire the successive reference tracking lines. These convex optimizations are computationally fast & efficient; however, this new alternative algorithm aims to achieve simultaneous planning and control using only the non-linear MPC optimization.

Here, real-time local tracking lines aren't generated, and instead, a fourth-order Bezier curve provides the path, influenced by potential fields from obstacles. While an initial MPC optimization guess is formulated using successive safest local angular gap headings, no predefined path is followed, which lends additional flexibility to the resulting solution. Local optima are sufficient using the potential fields approach as the non-convex solver provides safe, high-performing local solutions with reasonable initial guesses near the global optimum.

Using a Bezier curve provides a steep decrease in optimization variables, as now only the curve's control points determine the shape. For a Bezier curve of order n, only $n + 1$ control points exist, compared to the $5n_{MPC}k_{MPC}$ variables used for the discretized path in STLMPC, assuming a non-constant velocity. As a result, the average optimization time is lower for QBMPC, which enables faster control rates and thus, even more responsive vehicle behavior.

Finally, the use of a Bezier curve encourages smoother paths with more gradual changes in turning angle. This is in contrast to the STLMPC approach, where the objective was to closely track disjoint, piecewise linear reference paths. These factors, therefore, motivate the discussion & formulation of the QBMPC algorithm as a new technique for local path planning in unknown environments.

For an n-degree Bezier curve, the general parametric formula follows as:

$$\mathbf{B}(t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i \mathbf{P}_i, \quad t \in [0,1] \tag{6.1.1}$$

where $\mathbf{P}_i = [x_i, y_i]^T$ is the curve's i^{th} control point and t denotes the parameter which moves a point along the Bezier curve from $\mathbf{P}_0$ at $t = 0$ to $\mathbf{P}_n$ at $t = 1$. The curve is deformed towards the intermediate control points according to the Bernstein basis and is always contained entirely inside the convex hull of its control points. Here, t does not denote arc length but instead time on a fixed unit scale, which for the purposes of QBMPC is extended to describe motion over a general time, t_ξ.

In the ensuing algorithm, a fourth-order ($n = 4$) Bezier curve is used:

$$x(t) = (1-t)^4 x_0 + 4(1-t)^3 t x_1 + 6(1-t)^2 t^2 x_2 + 4(1-t) t^3 x_3 + t^4 x_4, \quad t \in [0,1] \tag{6.1.2}$$

$$y(t) = (1-t)^4 y_0 + 4(1-t)^3 t y_1 + 6(1-t)^2 t^2 y_2 + 4(1-t) t^3 y_3 + t^4 y_4, \quad t \in [0,1] \tag{6.1.3}$$

which provides path flexibility while not overfitting and introducing unnecessary path fluctuations. However, the degrees of freedom in this curve are reduced due to initial starting conditions, and thus the optimization variables are reduced from $2(n+1)$. First off, the original position $(x_0, y_0) = (0, 0)$ is fixed in the moving vehicle base frame for each optimization. Additionally, using the kinematic bicycle model, the initial velocity is fixed in the $+x$ direction ($\theta_0 = 0$) and so $x'(0) = v_{last}$ and $y'(0) = 0$. Already considering $(x_0, y_0) = (0, 0)$, this means:

$$x'(t) = (-16t^3 + 36t^2 - 24t + 4)x_1 + (24t^3 - 36t^2 + 12t)x_2 + (-16t^3 + 12t^2)x_3 + 4t^3 x_4 \tag{6.1.4}$$

$$y'(t) = (-16t^3 + 36t^2 - 24t + 4)y_1 + (24t^3 - 36t^2 + 12t)y_2 + (-16t^3 + 12t^2)y_3 + 4t^3 y_4 \tag{6.1.5}$$

So, $x'(0) = 4x_1 = v_{last}$ & $y'(0) = 4y_1 = 0$ meaning that $\mathbf{P}_1$ is fixed in the unit time scale case as $(x_1, y_1) = (\frac{v_{last}}{4}, 0)$. However, when parameterizing the curve for $t \in [0, 1]$ but over a true, scaled timeframe $\tau \in [0, t_\xi]$, now $t = \frac{\tau}{t_\xi}$. Thus, the actual physical velocity vector is represented by $[\frac{dx}{d\tau}, \frac{dy}{d\tau}]^T = [\frac{dx}{dt} \cdot \frac{dt}{d\tau}, \frac{dy}{dt} \cdot \frac{dt}{d\tau}]^T$ where at $t = 0$, $\begin{bmatrix} \frac{dx}{dt} \cdot \frac{dt}{d\tau} \\ \frac{dy}{dt} \cdot \frac{dt}{d\tau} \end{bmatrix}\Bigg|_{\tau=0} = \begin{bmatrix} v_{last} \\ 0 \end{bmatrix} = \begin{bmatrix} 4x_1 \cdot \frac{1}{t_\xi} \\ 4y_1 \cdot \frac{1}{t_\xi} \end{bmatrix}$ which means in the general case, the second fixed control point is $(x_1, y_1) = (\frac{v_{last} t_\xi}{4}, 0)$.

Furthermore, the initial steering angle, δ_{last}, is fixed, removing another degree of freedom. The curvature of a Bezier curve over time, $\kappa(t)$, describes the instantaneous sharpness of the curve's turn with respect to arc length (independent of choice of t_ξ):

$$\kappa(t) = \frac{x'(t)y''(t) - y'(t)x''(t)}{((x'(t))^2 + (y'(t))^2)^{3/2}} \tag{6.1.6}$$

where positive curvatures denote counterclockwise turns and negative values correspond to clockwise rotations. As required for calculating the curvature, the second derivatives of the fourth-order Bezier curve (where $y_1 = 0$) are given as:

$$x''(t) = (-48t^2 + 72t - 24)x_1 + (72t^2 - 72t + 12)x_2 + (-48t^2 + 24t)x_3 + 12t^2 x_4 \tag{6.1.7}$$

$$y''(t) = (72t^2 - 72t + 12)y_2 + (-48t^2 + 24t)y_3 + 12t^2 y_4 \tag{6.1.8}$$

Now, the curvature is expressed with respect to time by considering the magnitude of the variable velocity, $v(t) = \|\mathbf{B}'(t)\|_2$ to obtain the orientation derivative, $\dot{\theta}(t) = \kappa(t) \cdot \|\mathbf{B}'(t)\|_2$. From the kinematic bicycle model (Equation 3.1.23), $\dot{\theta}(t) = \frac{v(t)}{l}\tan(\delta(t))$. Comparing the equations yields:

$$\kappa(t) = \frac{\tan(\delta(t))}{l} \tag{6.1.9}$$

where for $t = 0$ & $\delta(0) = \delta_{last}$, the initial curvature becomes $\kappa(0) = \frac{48x_1y_2}{64x_1^3} = \frac{3y_2}{4x_1^2}$. Using the fixed value of $x_1 = \frac{v_{last}t_\xi}{4}$, now $y_2 = \frac{4x_1^2\tan(\delta_{last})}{3l}$ based on initial conditions. Therefore, a general n^{th} order curve is reduced to $2(n+1) - 5$ variables, meaning the quartic Bezier curve used in the MPC optimization has only five variables: x_2, x_3, y_3, x_4 & y_4 (Figure 6.1).

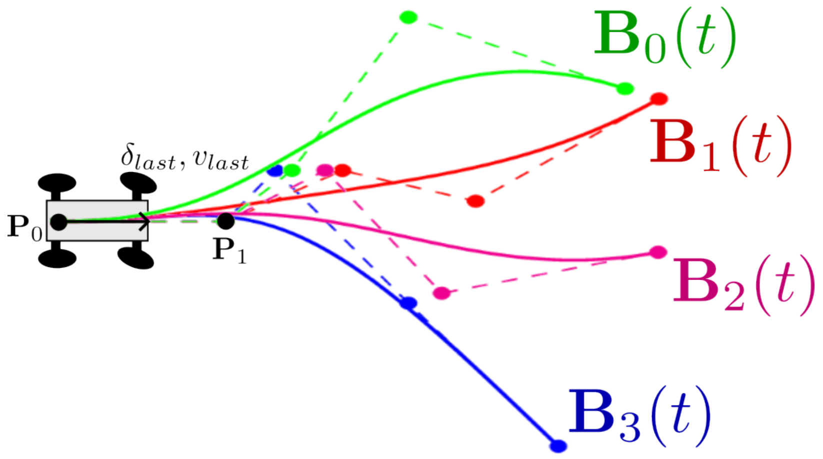


Figure 6.1: Four potential Bezier curve paths (solid) and their respective control polygons (dotted) with fixed $\mathbf{P}_0, \mathbf{P}_1$ & y_2 according to δ_{last} and v_{last}.

6.2 Initialization

The initial guess for the quartic Bezier curve control points, used in the MPC optimization, is derived from two successive safest angular gap headings. Consequently, the approach outlined in Algorithm 1 and Equation 3.1.1 is used as was done in Chapter 3 to get the left and right obstacle clusters for the reference tracking line optimization. Now, though, θ_{head} is directly used in the place of the tracking line for the initial path guess, and no quadratic optimizations are conducted. Initially, $[x_{start,0}, y_{start,0}]^T = [0, 0]^T$ and instead of Equation 3.1.7, now:

$$[x_{end,i}, y_{end,i}]^T = \begin{cases} p_{start,i} + \frac{3}{n} v_{last} t_\xi \cdot [\cos(\theta_{head,i}), \sin(\theta_{head,i})]^T & \text{if} \quad i = 0 \\ p_{start,i} + \frac{1}{n} v_{last} t_\xi \cdot [\cos(\theta_{head,i}), \sin(\theta_{head,i})]^T & \text{if} \quad i > 0 \end{cases} \tag{6.2.1}$$

where $i \in \{0, ..., n-3\}$ since the first three control points are at least partially fixed.

Thus, the first free control point, $p_{end,0}$ is determined by proceeding along the heading direction for $\frac{3}{n}$ of the total horizon time, t_ξ, compared to $\frac{1}{n}$ for all subsequent control point guesses. For the next heading angle, $p_{start,i+1} = p_{end,i}$ and the process repeats with the transformation of all obstacles to the new frame (similar to Equation 3.1.8 but now using $\theta_{head,i}$ instead of $\theta_{track,i}$). Each safest gap heading is converted back to the base frame through $\theta_{head,i} \leftarrow \theta_{head,i} + \theta_{head,i-1}$ before Equation 6.2.1 to acquire the control points in the current vehicle frame. For the fourth-order Bezier curve, the initial control point guesses are:

$$x_2 = \frac{2}{3} x_{end,0}, \quad (x_3, y_3) = (x_{end,0}, y_{end,0}), \quad (x_4, y_4) = (x_{end,1}, y_{end,1}) \tag{6.2.2}$$

This provides an initial path in the direction of the safest successive angular gaps (Figure 6.2) as in STLMPC and is extendable to curves of order $n > 4$ as well.

Using AMCL-based localization and dynamic obstacles operates exactly as in STLMPC, where the obstacle set is extended by the map and tracked vehicle points. The only difference is that the base frame obstacle set contains the entire trajectories of tracked vehicles until time t_ξ, as opposed to successive timeframe segments that were used for tracking line generation (Equation 4.4.2). Therefore, the trajectory over the full time horizon is used to find each successive safest angular gap, as well as in the subsampled obstacle set used in the QBMPC objective and constraint functions.

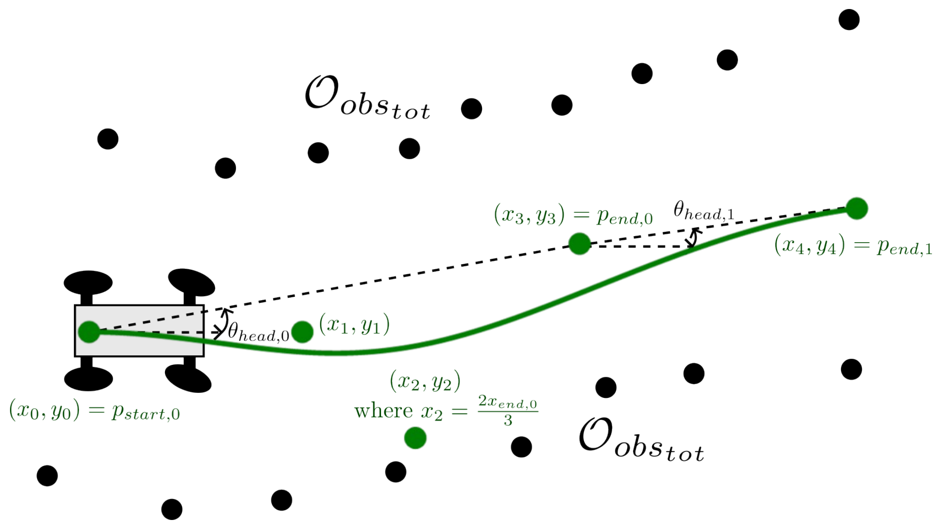

$\mathcal{O}_{obs_{tot}}$
$(x_3, y_3) = p_{end,0}$
$\theta_{head,1}$
$(x_4, y_4) = p_{end,1}$
$\theta_{head,0}$
(x_1, y_1)
(x_2, y_2)
where $x_2 = \frac{2x_{end,0}}{3}$
$(x_0, y_0) = p_{start,0}$
$\mathcal{O}_{obs_{tot}}$

Figure 6.2: The initial QBMPC guess of control points and corresponding curve (green). Safest heading angles with respect to the base frame $+x$ axis dictate the initial guess for x_2, x_3, y_3, x_4 & y_4 while x_0, y_0, x_1, y_1 & y_2 are fixed.

6.3 Potential Field-Based Objective

The optimal quartic Bezier curve is attained by minimizing a potential field function that discourages close proximities to obstacles over the full trajectory, thus maintaining a safe path. This method requires no reference path to follow and responds to unknown environments with strong repulsive forces pushing the path from obstacles towards safe regions through the objective's control point gradients. Through a strategic initial guess generated as described, the Bezier curve path explicitly promotes high local obstacle clearance like in STLMPC but without a tracking reference.

The velocity over the predicted path is allowed to vary, providing flexibility, but higher velocities aren't prioritized like in Chapter 5. Nonetheless, incorporating a velocity-based objective term could be a future extension of QBMPC for racing applications. There is also no objective term on control effort as in STLMPC; however, the Bezier curve path inherently generates smooth, gradually turning paths by design.

To gauge obstacle proximity over the full duration of the curve, the path is discretized evenly in time, t (and effectively τ). Here, the points on the Bezier curve, $(x_{\xi,i}, y_{\xi,i})$ are:

$$(x_{\xi,i}, y_{\xi,i}) = (x(t_i), y(t_i)), \quad t_i = \frac{i}{n_\xi - 1}, \quad i \in \{0, ..., n_\xi - 1\} \tag{6.3.1}$$

for a total of n_ξ discretized points, which depend on the optimization variables: the control points. Now, the potential field function is developed, which considers subsampled obstacles and all n_ξ discretized points. The obstacle subsampling follows the same process as described in Equation 5.3.6 for all detected, map and tracked vehicle points in the total obstacle set. This obtains the obstacle subset $\mathcal{O}_{obs_{sub}} = \{(x_{obs_{sub},j}, y_{obs_{sub},j})\}_{j=0}^{N_{obs_{sub}}-1}$ where each obstacle enacts a potential field upon the Bezier curve path.

The potential field-based objective function is thus formulated as a sum of the repulsive forces from each subsampled obstacle to each discretized Bezier curve point:

$$F_{obj_\xi}(x_2, x_3, y_3, x_4, y_4) = \sum_{i=0}^{n_\xi - 1} \sum_{j=0}^{N_{obs_{sub}}-1} \frac{1}{d_{\xi,i,j}^2} e^{-\alpha_\xi d_{\xi,i,j}^2} \tag{6.3.2}$$

with gradients in Appendix A.3 where each squared curve-to-obstacle proximity is given as:

$$d_{\xi,i,j}^2 = (x_{\xi,i} - x_{obs_{sub},j})^2 + (y_{\xi,i} - y_{obs_{sub},j})^2 \tag{6.3.3}$$

Instead of a standard inverse-square relationship, an exponentially decaying weight for higher squared distances is incorporated in each summed term, further reducing the effect of numerous far obstacles in favor of the effect of one obstacle in close proximity. This prioritizes the nearest obstacles the most in the cost, such that the optimal path avoids close proximities over the predicted horizon's full duration. Here, α_ξ affects how sharply the weight decays for higher squared distances, controlling the degree to which closer proximities dominate in the objective. The selection of three α_ξ values and their effect on the objective cost for a single curve-to-obstacle distance is shown in Figure 6.3.

6.4 Constraints on the Bezier Curve Profile

6.4.1 Velocity-Embedded Curve Constraints

In order to restrict the Bezier curve path to be feasible, vehicle dynamics are incorporated into the curve shape. The first limits on vehicle behavior come from the minimum ($v_{\xi,\min}$) & maximum ($v_{\xi,\max}$) allowed velocities, which are incorporated for a general path time horizon, t_ξ. A positive minimum velocity is necessary in this formulation to prevent unnecessary stopping under normal operation, as the objective does not inherently encourage higher velocities like in Chapter 5. In the event that there is no safe route and path planning fails, the vehicle can slow to a stop based on the predefined d_{stop} to prevent collisions.

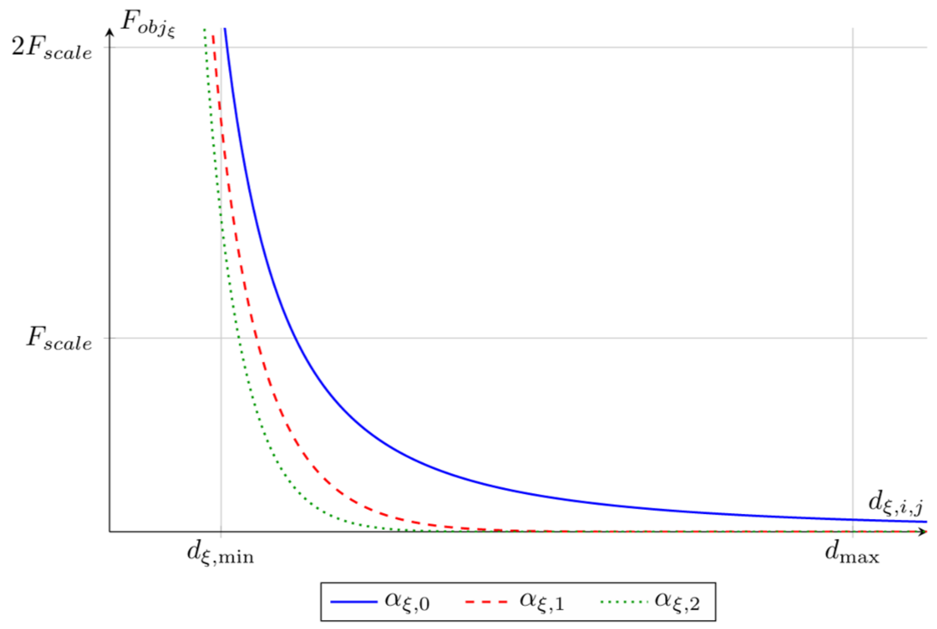


Figure 6.3: The potential field function for a single curve-to-obstacle proximity with three weight decay factors, $\alpha_{\xi,0} = 0 < \alpha_{\xi,1} < \alpha_{\xi,2}$. F_{scale} denotes a symbolic scale for the objective function, $d_{\max}$ is the maximum range from earlier chapters and $d_{\xi,\min}$ is the minimum allowed proximity, used as a constraint.

The velocity of the Bezier curve, $\tau \in [0, t_\xi]$ is denoted by the derivative's magnitude:

$$v(\tau) = \|\frac{d\mathbf{B}}{dt} \cdot \frac{dt}{d\tau}\|_2 = \sqrt{(\frac{x'(t)}{t_\xi})^2 + (\frac{y'(t)}{t_\xi})^2} \tag{6.4.1}$$

with the first derivatives of the Bezier curve given by Equations 6.1.4 & 6.1.5. Again, the curve is discretized by $v_{\xi,i} = v(t_i) = \|\mathbf{B}'(t_i)\|_2$ for $t_i = \frac{i}{n_\xi - 1}$, $i \in \{0, ..., n_\xi - 1\}$. Taking the squared velocity for simplification, the velocity at each discretized point is constrained between the minimum and maximum:

$$v_{\xi,\min}^2 \leq (v(\tau))^2 \leq v_{\xi,\max}^2 \quad \text{where for } i \in \{0, ..., n_\xi - 1\},$$

$$g_{v_\xi^+,i} = (x'(t_i))^2 + (y'(t_i))^2 - t_\xi^2 v_{\xi,\max}^2, \quad g_{v_\xi^+,i} \leq 0 \tag{6.4.2}$$

$$g_{v_\xi^-,i} = -(x'(t_i))^2 - (y'(t_i))^2 + t_\xi^2 v_{\xi,\min}^2, \quad g_{v_\xi^-,i} \leq 0 \tag{6.4.3}$$

By choosing a sufficiently high n_ξ for the Bezier curve order n, satisfying the constraints for the discretized points means the constraints are effectively satisfied for the entire curve. Sampling evenly in time ensures that all regions of the curve over the future horizon are evaluated and made to satisfy the limits on vehicle dynamics. This discretization provides tractable closed-form constraints for maintaining a viable path, which depend on the placement of the quartic Bezier curve's control points.

Similarly, the acceleration at each discretized point is evaluated where the acceleration in the direction of motion of the Bezier curve is given by:

$$a(t) = \frac{\mathbf{B}'(t) \cdot \mathbf{B}''(t)}{\|\mathbf{B}'(t)\|_2} = \frac{x'(t)x''(t) + y'(t)y''(t)}{\sqrt{(x'(t))^2 + (y'(t))^2}} \tag{6.4.4}$$

This expression projects the acceleration vector, $\mathbf{B}''(t)$ onto the normalized velocity vector, $\frac{\mathbf{B}'(t)}{\|\mathbf{B}'(t)\|_2}$ to determine the acceleration or deceleration in the forward direction, which is constrained to the feasible range. By taking $x'(\tau) = x'(t) \cdot \frac{dt}{d\tau} = \frac{x'(t)}{t_\xi}$ and $x''(\tau) = \frac{x''(t)}{t_\xi} \cdot \frac{dt}{d\tau} = \frac{x''(t)}{t_\xi^2}$ (similarly for y), acceleration for general time, $\tau \in [0, t_\xi]$ is:

$$a(\tau) = \frac{\frac{1}{t_\xi^3}(x'(t)x''(t) + y'(t)y''(t))}{\frac{1}{t_\xi}\sqrt{(x'(t))^2 + (y'(t))^2}} = \frac{1}{t_\xi^2} a(t) \tag{6.4.5}$$

Now, for the maximum acceleration, $a_{\xi,\max}$ and deceleration, $-a_{\xi,\max}$ limits, the squared forward acceleration is constrained for each discretized point via $a_{\xi,i} = a(t_i)$. This ensures that both positive and negative forward accelerations are bounded in magnitude by $|a_{\xi,\max}|$ in one constraint for each of the n_ξ points:

$$-a_{\xi,\max} \leq a(\tau) \leq a_{\xi,\max} \rightarrow (a(\tau))^2 \leq a_{\xi,\max}^2 \quad \text{where for } i \in \{0, ..., n_\xi - 1\},$$
$$g_{a_\xi,i} = \frac{(x'(t_i)x''(t_i) + y'(t_i)y''(t_i))^2}{(x'(t_i))^2 + (y'(t_i))^2} - t_\xi^4 a_{\xi,\max}^2, \quad g_{a_\xi,i} \leq 0 \tag{6.4.6}$$

Therefore, the quartic Bezier curve is constrained to the vehicle's permissible velocity dynamics in a general formulation for any time horizon length, t_ξ (corresponding analytical gradients are in Appendix A.3).

6.4.2 Curvature-Embedded Curve Constraints

Additional path restrictions are influenced by limitations on steering angle, which translate to the curvature of the Bezier curve. As in STLMPC, the steering angle is constrained by inequalities on both magnitude and rate of change. The path curvature, which depends on the optimization variables (control points), is given by Equation 6.1.6 (and is independent of the choice of t_ξ) while the relation to steering angle is provided in Equation 6.1.9.

Invoking $\delta_{\xi,\max}$ and $-\delta_{\xi,\max}$ in the place of $\delta(t)$ yields the maximum and minimum allowed curvature along the path, respectively. By ensuring the curvature at each discretized point, $\kappa_{\xi,i} = \kappa(t_i)$ is within the tolerated range, the curvature constraints are created:

$$\frac{\tan(-\delta_{\xi,\max})}{l} \leq \kappa(\tau) = \kappa(t) \leq \frac{\tan(\delta_{\xi,\max})}{l} \quad \text{where for } i \in \{0, ..., n_\xi - 1\},$$
$$g_{\kappa_\xi^+,i} = \frac{x'(t_i)y''(t_i) - y'(t_i)x''(t_i)}{((x'(t_i))^2 + (y'(t_i))^2)^{3/2}} - \frac{\tan(\delta_{\xi,\max})}{l}, \quad g_{\kappa_\xi^+,i} \leq 0 \tag{6.4.7}$$
$$g_{\kappa_\xi^-,i} = -\frac{x'(t_i)y''(t_i) - y'(t_i)x''(t_i)}{((x'(t_i))^2 + (y'(t_i))^2)^{3/2}} - \frac{\tan(\delta_{\xi,\max})}{l}, \quad g_{\kappa_\xi^-,i} \leq 0 \tag{6.4.8}$$

The limit on the steering angle's rate of change can be expressed by ensuring

$\delta'(t)$ is within the limits of $-\delta'_{\xi,\max}$ and $\delta'_{\xi,\max}$. Expressing the curvature through the steering angle, $\delta(t) = \tan^{-1}(l\kappa(t))$, the derivative becomes:

$$\delta'(t) = \frac{l\kappa'(t)}{1 + (l\kappa(t))^2} \tag{6.4.9}$$

where, applying the quotient rule to Equation 6.1.6, the curvature derivative $\kappa'(t)$ is:

$$\kappa'(t) = \frac{(x'y''' - y'x''')((x')^2 + (y')^2) - 3(x'x'' + y'y'')(x'y'' - y'x'')}{((x')^2 + (y')^2)^{5/2}} \tag{6.4.10}$$

omitting functional notation for the sake of brevity. The third derivatives are now required, obtained by differentiating Equations 6.1.7 and 6.1.8:

$$x'''(t) = (-96t + 72)x_1 + (144t - 72)x_2 + (-96t + 24)x_3 + 24tx_4 \tag{6.4.11}$$

$$y'''(t) = (144t - 72)y_2 + (-96t + 24)y_3 + 24ty_4 \tag{6.4.12}$$

The steering angle derivative is converted to general time, τ where $\kappa(\tau) = \kappa(t)$ but using $x'''(\tau) = \frac{x'''(t)}{t_\xi^2} \cdot \frac{dt}{d\tau} = \frac{x'''(t)}{t_\xi^3}$ (similarly for y) and the appropriate conversions for the first and second derivatives, Equation 6.4.10 becomes $\kappa'(\tau) = \frac{1}{t_\xi}\kappa'(t)$. Therefore, $\delta'(\tau) = \frac{1}{t_\xi}\delta'(t)$ and the curvature derivative constraints can now be formulated at each of the n_ξ discretized curve points (where $\kappa'_{\xi,i} = \kappa'(t_i)$):

$$-\delta'_{\xi,\max} \leq \delta'(\tau) \leq \delta'_{\xi,\max} \quad \text{where for } i \in \{0, ..., n_\xi - 1\},$$

$$g_{\kappa'^+_\xi,i} = \frac{l\kappa'(t_i)}{1 + (l\kappa(t_i))^2} - t_\xi\delta'_{\xi,\max}, \quad g_{\kappa'^+_\xi,i} \leq 0 \tag{6.4.13}$$

$$g_{\kappa'^-_\xi,i} = -\frac{l\kappa'(t_i)}{1 + (l\kappa(t_i))^2} - t_\xi\delta'_{\xi,\max}, \quad g_{\kappa'^-_\xi,i} \leq 0 \tag{6.4.14}$$

These curvature constraints (gradients included in Appendix A.3) are generalized to an arbitrary timescale, t_ξ, and complete the vehicle dynamics-based quartic Bezier curve restrictions.

6.4.3 Minimum Curve-to-Obstacle Proximity Limit

The final set of constraints ensures a minimum safe obstacle clearance, $d_{\xi,\min}$, is maintained through the duration of the predicted path. While the objective encourages higher distances to nearby obstacles through potential field forces, it doesn't explicitly prohibit a single low curve-to-obstacle distance if the rest of the curve achieves a low cost function. In order to prevent collisions and ensure that only safe paths are selected, the minimum obstacle proximity constraint and its gradients shape the control points such that the curve maintains at least the buffer distance, $d_{\xi,\min}$, to the nearest obstacle.

To obtain each curve-to-obstacle distance, the curve is discretized and the obstacles are subsampled as in the objective (Equation 6.3.2). The minimum obstacle distance for each discretized curve point is obtained through a softmin function, similar to that used in Equation 5.3.11 but with no weighting factors on each term:

$$d_{\xi,\min,i} = -\frac{1}{\beta_\xi}\log\Big(\sum_{j=0}^{N_{obs_{sub}}-1} e^{-\beta_\xi d_{\xi,i,j}}\Big), \quad i \in \{0,...,n_\xi - 1\} \tag{6.4.15}$$

where β_ξ serves as the approximation sharpness in place of Equation 5.3.11's β_v and the individual curve-to-obstacle distances $d_{\xi,i,j}$ are from the square root of Equation 6.3.3. This discretization and subsampling approximates the full curve-to-obstacle minimum distance expression (not closed-form) in a computationally tractable form. The smooth minimum function ensures gradients (Appendix A.3) can be found so the optimization solver can move the control points (and thus the curve) away from obstacles in close proximity.

The softmin distance is finally compared to the minimum permissible distance in the curve-to-obstacle proximity inequality constraints:

$$d_{\xi,\min} \leq d_{\xi,\min,i}, \quad i \in \{0, ..., n_\xi - 1\}$$
$$g_{d_\xi,i} = \frac{1}{\beta_\xi}\log(\sum_{j=0}^{N_{obs_{sub}}-1} e^{-\beta_\xi d_{\xi,i,j}}) + d_{\xi,\min}, \quad g_{d_\xi,i} \leq 0 \tag{6.4.16}$$

This set of constraints is sampled evenly in t & τ and is the same, irrespective of the choice of t_ξ. The planned quartic Bezier curve path is shown in Figure 6.4, sampled in both t & τ, which maintains the minimum proximity, $d_{\xi,\min}$ to any nearby obstacles. When n_ξ is sufficiently high, satisfying the minimum proximity requirement for each discretized point effectively satisfies the condition for the full curve.

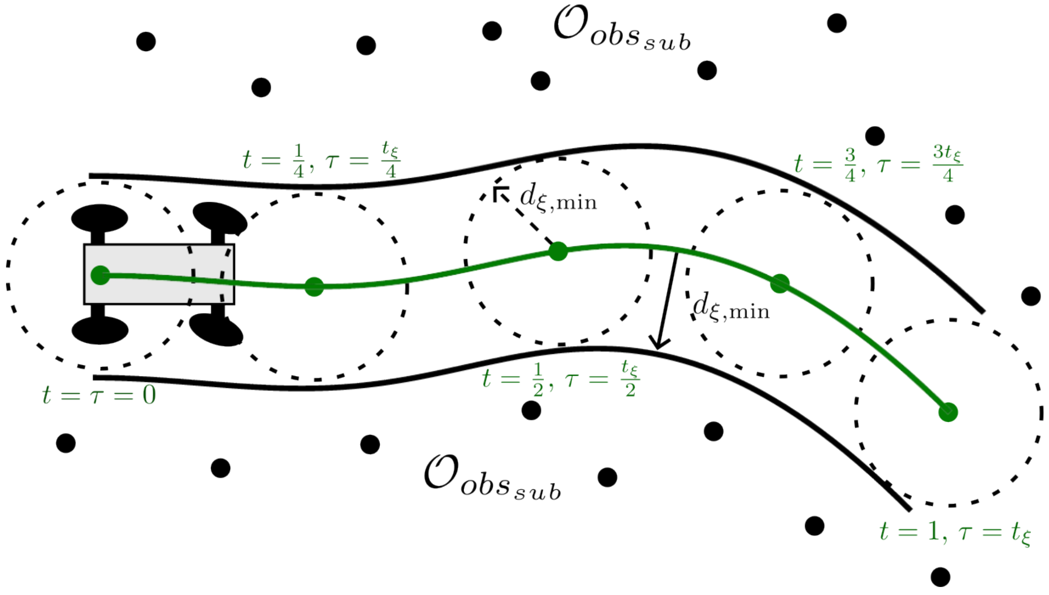

$\mathcal{O}_{obs_{sub}}$
$t = \frac{1}{4}, \tau = \frac{t_\xi}{4}$
$d_{\xi,\min}$
$t = \frac{3}{4}, \tau = \frac{3t_\xi}{4}$
$d_{\xi,\min}$
$t = \tau = 0$
$t = \frac{1}{2}, \tau = \frac{t_\xi}{2}$
$\mathcal{O}_{obs_{sub}}$
$t = 1, \tau = t_\xi$

Figure 6.4: Quartic Bezier curve path (green) and discretized curve points in both the Bezier curve parameter, t, and physical time, τ, for a low curve point count for illustration purposes, $n_\xi = 5$. The subsampled obstacles all exceed the minimum distance, $d_{\xi,\min}$, to both the curve (black, solid bounds) and each discretized curve point (dashed, circle bounds).

6.5 Culminating Optimization & Actuation

After completing the QBMPC optimization, the predicted path is obtained through the fixed control point coordinates, $(x_0, y_0), (x_1, y_1)$ & y_2 as well as the optimized control point coordinates, $x_2, (x_3, y_3)$ & (x_4, y_4). In practice, the control points are bounded in the optimization by $-d_{\max} \leq x_2, x_3, y_3, x_4, y_4 \leq d_{\max}$ to confirm the predicted trajectory stays within the maximum look-ahead range used for all local obstacles in $\mathcal{O}_{obs_{sub}}$. The final QBMPC optimization problem is thereby described as:

$$\begin{aligned}
\min_{x_2,x_3,y_3,x_4,y_4} \quad & F_{obj_\xi} \\
\text{subject to} \quad & g_{v_\xi^+,i} \leq 0, \quad g_{v_\xi^-,i} \leq 0, \quad g_{a_\xi,i} \leq 0, \quad g_{d_\xi,i} \leq 0 \\
& g_{\kappa_\xi^+,i} \leq 0, \quad g_{\kappa_\xi^-,i} \leq 0, \quad g_{\kappa'^+_\xi,i} \leq 0, \quad g_{\kappa'^-_\xi,i} \leq 0 \\
& -d_{\max} \leq x_2, x_3, y_3, x_4, y_4 \leq d_{\max} \\
& x_0 = 0, \quad y_0 = 0, \quad x_1 = \frac{v_{last} t_\xi}{4}, \quad y_1 = 0, \quad y_2 = \frac{4x_1^2 \tan(\delta_{last})}{3l} \\
& \forall i \in \{0, ..., n_\xi - 1\}
\end{aligned} \tag{6.5.1}$$

Now, the final control points reflect the quartic Bezier curve optimization solution for local path planning. In order to actuate the vehicle along the path, the steering angle and velocity commands must be extracted from the curve upon successful termination of the optimization.

Since the Bezier curve's initial conditions (as in STLMPC) correspond to δ_{last} and v_{last}, the control inputs according to the curve at the next control step time, Δt, are found and applied at the current time. This ensures the steering angle and velocity transition over the nonzero required time to the values at the next timestep, given by the predicted path (this control input scheme is carried over from earlier chapters).

Converting the physical time $\tau_{step} = \Delta t$ to the normalized curve parameter $t_{step} = \frac{\Delta t}{t_\xi}$ ensures the Bezier curve is evaluated at the correct scaled point.

Now, ascertaining the curvature, $\kappa(t)$ at t_{step} (where $\kappa(\tau) = \kappa(t)$) yields the desired steering angle command through the relationship described in Equation 6.1.9:

$$\delta_{cmd} = \delta(\tau_{step}) = \tan^{-1}(l\kappa(t_{step})) \tag{6.5.2}$$

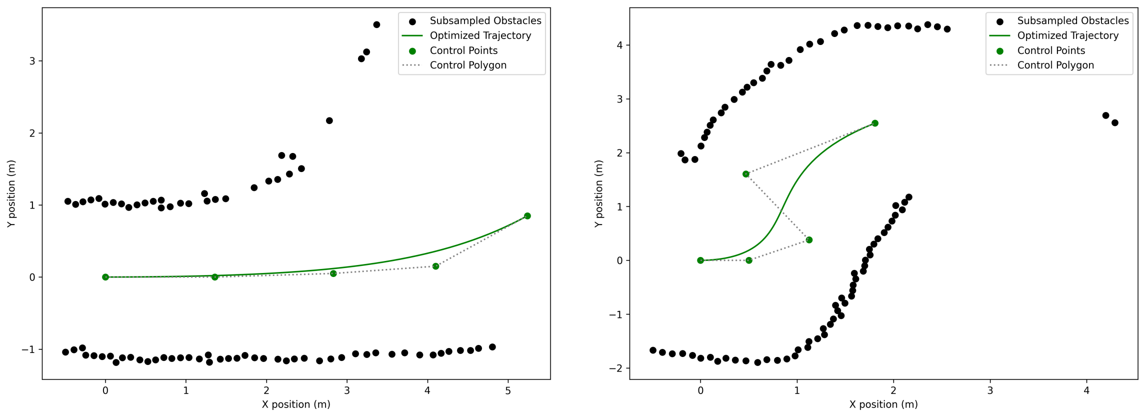
For the velocity command, Equation 6.4.1 is used to obtain the physical velocity at τ_{step} through the Bezier curve velocity and scaled time horizon parameter, t_ξ:

$$v_{cmd} = v(\tau_{step}) = \frac{1}{t_\xi}\sqrt{(x'(t_{step}))^2 + (y'(t_{step}))^2} \tag{6.5.3}$$

The resulting control input pair (δ_{cmd}, v_{cmd}) provides vehicle actuation to achieve the predicted Bezier curve path during navigation. The physical actuation transition from $(\delta_{last}, v_{last})$ to (δ_{cmd}, v_{cmd}) over Δt matches the Bezier curve's smooth transition over time in both curvature and velocity. The QBMPC approach continues for each control step to attain real-time local path planning in unknown environments.

6.6 Simulation Results

The QBMPC method is now contrasted to Chapter 5's non-uniform velocity STLMPC as well as TEB-based exploration in a fourth simulation environment (Map #4). Here, QBMPC-related parameters include t_ξ =2 s, n_ξ =10, α_ξ =5.5 m^{-2}, β_ξ =10 m^{-1}, $v_{\xi,\min}$ =1.5 m/s, $v_{\xi,\max}$ =3 m/s & $d_{\xi,\min}$ =0.3 m. The QBMPC optimized path is provided at two sample times with the governing control points subject to potential fields from nearby obstacles (Figure 6.5).



(a) (b)

Figure 6.5: Resulting trajectory and associated control points after optimization for QBMPC at two separate time samples. Each optimized curve is smooth and far from obstacles due to acting potential field forces.

A high minimum velocity is incorporated as QBMPC does not prioritize higher velocities, and a minimum velocity of 0 m/s causes the planner to slow to a stop. The choice of minimum and maximum velocity allows for comparison against Chapter 5's approach and TEB-based exploration with a relatively constant 1.5 m/s speed. Steering angle-based velocity constraints are guaranteed to be satisfied via TEB at this speed, while instead of directly considering these constraints, QBMPC achieves smooth, safe curves for trajectories inherently. The performance of each of these planners on Map #4 is illustrated in Table 6.1.

Table 6.1: Performance of QBMPC & other local planners in simulated Map #4

Local Planner	$\min_{\tilde{k}} d_{min,\tilde{k}}$ (m)	$\bar{d}_{min}$ (m)	$\overline{\lvert\delta_{cmd,\tilde{k}}\rvert}$ (rad)	Var($\delta_{cmd,\tilde{k}}$) (rad^2)	$\bar{v}_{cmd,\tilde{k}}$ (m/s)	Var($v_{cmd,\tilde{k}}$) (m^2/s^2)	t_{tot} (s)
TEB_explore	0.469	0.943	0.109	0.021	1.476	0.014	44.6
STLMPC	0.618	1.115	0.111	0.022	2.397	0.208	27.9
QBMPC	0.819	1.254	0.078	0.013	1.742	0.184	43.1

Once again, STLMPC outperforms TEB-based exploration while also achieving the fastest course time. QBMPC achieves a slower time as it does not prioritize speed but shows value through improved safety metrics compared to STLMPC. Additionally, control effort is lower, owing to the Bezier curve's smoothness, where variances in both steering angle and velocity control inputs are lower than in STLMPC. This is further shown in Figure 6.6 where QBMPC attains a safer & smoother albeit slower path than STLMPC. Thus, each method presents its own advantages and use cases, while minor adjustments in each formulation can raise new possibilities, such as QBMPC in racing contexts.

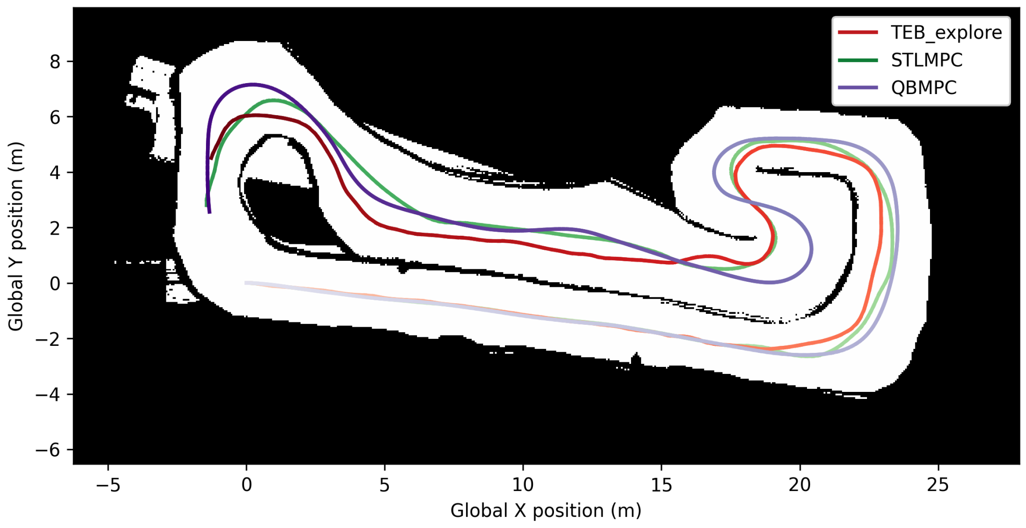


Figure 6.6: STLMPC, QBMPC & TEB-based exploration local planners in simulated Map #4. Darker color gradients for each path show time progression (proceeding counterclockwise around the track) where STLMPC completes the course fastest and is thus lighter throughout.

The safety advantages of the QBMPC approach are further visualized in Figure 6.7. QBMPC reliably maintains the highest minimum obstacle proximity of the approaches, preserving a clearance consistently exceeding 1 m even in narrower sections and around corners. All three approaches maintain obstacle clearances above the minimum allowed amount, which is used as a constraint in the QBMPC formulation.

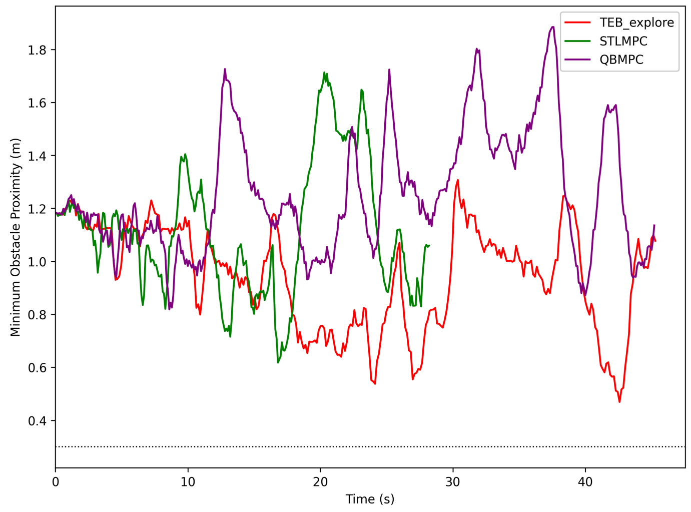


Figure 6.7: Minimum obstacle proximity at each sample time ($d_{\min,\tilde{k}}$) for the three tested local planners in simulation using Map #4. The dotted lower bound indicates the minimum allowed obstacle proximity in QBMPC ($d_{\xi,\min}$).

6.7 Experimental Results

Using the MacAEV in two new experiment setups (Appendix B), QBMPC is evaluated both in a static environment (Experiment #4) and in a dynamic environment with an adversarial, detected vehicle (Experiment #5). Carrying over parameters used in Chapter 5's experiment for variable-velocity STLMPC, QBMPC maintains the values used in simulation except with a shorter prediction time horizon, $t_\xi = 1$ s. All aforementioned exploration-based planners are contrasted in Experiment #4 as well as the non-predictive, single tracking line PD approach.

Operation of the QBMPC algorithm in Experiment #4 is recorded[1] and the trajectories obtained by each approach are documented in Figure 6.8. In this experiment,

[1]https://www.youtube.com/watch?v=3j0edNW95D0

Chapter 5's STLMPC achieves a significantly faster course time than all other methods, while QBMPC attains a smooth path that best follows the center of the track. Here, exploration-based planners have more path fluctuations, where corners are cut the most by DWA, risking collision. The PD method struggles at the widest section of the track, failing to predict the turn and overshooting, narrowly avoiding collision.

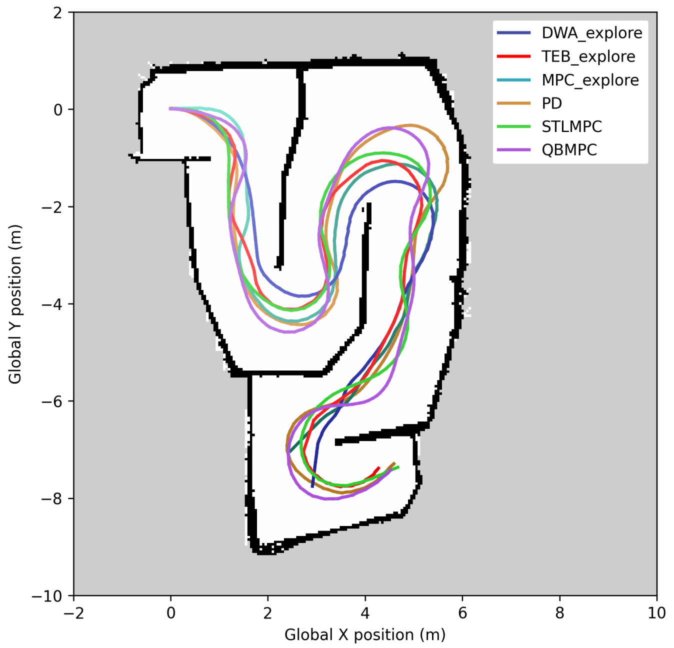


Figure 6.8: Local planner trajectories including for QBMPC in Experiment #4. Darker color gradients for each path show time progression, where variable-velocity STLMPC completes the course fastest and is therefore lightest.

Additionally, these results are displayed in Table 6.2, proving the value of variable-velocity STLMPC in achieving faster navigation and QBMPC in attaining safer, smoother travel. The DWA and MPC planners complete the track with very slow times while not excelling in other metrics compared to the alternative planners. PD & TEB perform comparably, where PD has a poorer minimum obstacle proximity but a better average proximity. Across the board, STLMPC & QBMPC perform best with high obstacle clearance throughout, while STLMPC attains the lowest course time and QBMPC is the second fastest while expending low control effort.

Table 6.2: Performance of QBMPC & other local planners in Experiment #4

Local Planner	$\min_{\tilde{k}} d_{min,\tilde{k}}$ (m)	$\bar{d}_{min}$ (m)	$\overline{\lvert\delta_{cmd,\tilde{k}}\rvert}$ (rad)	$\mathrm{Var}(\delta_{cmd,\tilde{k}})$ (rad^2)	$\bar{v}_{cmd,\tilde{k}}$ (m/s)	$\mathrm{Var}(v_{cmd,\tilde{k}})$ (m^2/s^2)	t_{tot} (s)
DWA_explore	0.219	0.604	0.228	0.072	0.499	0.009	30.8
TEB_explore	0.487	0.715	0.298	0.104	0.948	0.214	19.2
MPC_explore	0.512	0.745	0.236	0.074	0.492	0.001	35.5
PD	0.388	0.758	0.242	0.079	1.123	0.142	20.5
STLMPC	0.469	0.781	0.235	0.070	1.693	0.296	12.0
QBMPC	0.596	0.861	0.222	0.067	1.284	0.077	17.9

Computation times for each algorithm (Table 6.3) indicate the significant improvement yielded by QBMPC over STLMPC. QBMPC substantially reduces the number of optimization variables, opting for a smooth path instead of a more complex, discretized trajectory. This produces a far lower planning time than STLMPC, both on average and in the worst case when considering a variable velocity (Figure 6.9). QBMPC performs on the scale of the simple PD method, while STLMPC is comparable to the exploration planners, using graceful optimization timeout to maintain real-time control. Reducing the prediction time horizon (and thus optimization variables) via n_{MPC} & k_{MPC} improves planning time at the cost of shorter look-ahead.

Table 6.3: Planning times for QBMPC & other local planners in Experiment #4

Local Planner	$\bar{t}_{frontier}$ (ms)	$\bar{t}_{global}$ (ms)	$\bar{t}_{local}$ (ms)	$\boldsymbol{\bar{t}_{comp}}$ **(ms)**	$\max t_{frontier}$ (ms)	$\max t_{global}$ (ms)	$\max t_{local}$ (ms)	$\mathbf{max}\, \boldsymbol{t_{comp}}$ **(ms)**
DWA_explore	3.1	0.8	25.1	**29.0**	7.1	1.8	42.5	**51.4**
TEB_explore	2.9	1.0	4.8	**8.7**	5.8	4.5	16.3	**26.6**
MPC_explore	2.8	0.8	20.4	**24.1**	8.5	1.8	147.0	**157.3**
PD				**0.5**				**8.0**
STLMPC				**33.7**				**55.5**
QBMPC				**1.2**				**2.3**

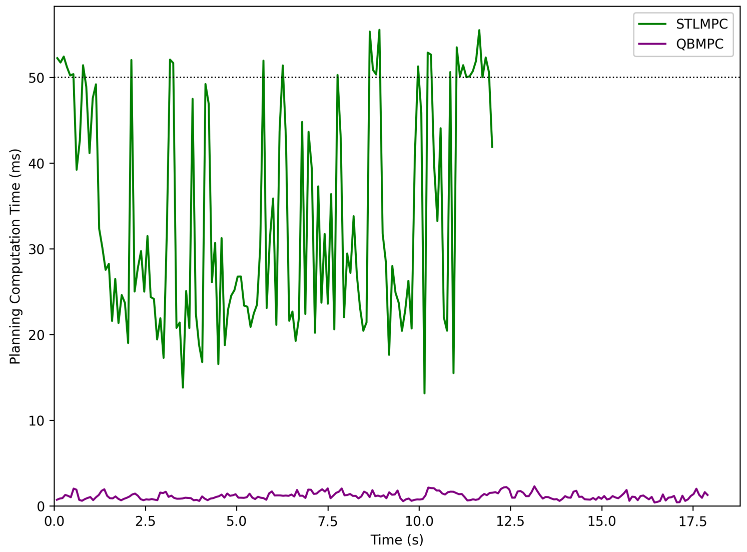


Figure 6.9: Planning computation times for STLMPC & QBMPC at each timestep over the full navigation duration in Experiment #4. The dotted upper bound indicates the optimization timeout limit, set to 50 ms.

Next, the QBMPC algorithm is evaluated in Experiment #5 alongside Chapter 5's STLMPC & TEB, the highest performing exploration planner. Dynamic vehicle avoidance is achieved by QBMPC, illustrated by video[2], while each local planner's performance is quantified via Table 6.4. Again, QBMPC maintains the largest obstacle clearance while also expending low control effort despite the maneuvers enacted to maintain safety, such as avoiding the detected vehicle. STLMPC is considerably faster than the other methods, attaining higher speeds and larger velocity fluctuations.

Table 6.4: Performance of local planners including QBMPC for dynamic obstacle avoidance in Experiment #5

Local Planner	$\min_{\tilde{k}} d_{min,\tilde{k}}$ (m)	$\bar{d}_{min}$ (m)	$\overline{\lvert\delta_{cmd,\tilde{k}}\rvert}$ (rad)	$\mathrm{Var}(\delta_{cmd,\tilde{k}})$ (rad^2)	$\bar{v}_{cmd,\tilde{k}}$ (m/s)	$\mathrm{Var}(v_{cmd,\tilde{k}})$ (m^2/s^2)	t_{tot} (s)
TEB_explore	0.242	0.778	0.205	0.046	1.201	0.158	15.4
STLMPC	0.371	0.807	0.167	0.036	1.743	0.602	9.9
QBMPC	0.467	0.895	0.184	0.044	1.328	0.155	13.8

[2]`https://www.youtube.com/watch?v=m4K5vlIFxEA`

Each planner's trajectory is given in Figure 6.10 along with the detected vehicle path, which poses an immediate danger, forcing collision avoidance. Both STLMPC & QBMPC react to the detected vehicle moving rightward directly ahead by swerving left, behind the adversarial vehicle. STLMPC, operating here with a higher speed than QBMPC, has a longer prediction horizon and therefore reacts earlier to avoid the vehicle, although both methods maintain safe paths. Alternatively, TEB-based exploration considers the vehicle statically, failing to correctly swerve around the oncoming vehicle. It crosses directly in front of the adversarial vehicle and risks collision if the other vehicle is not manually slowed, as in this case. Computational times are shown for this experiment in Table 6.5 where QBMPC again vastly outperforms STLMPC.

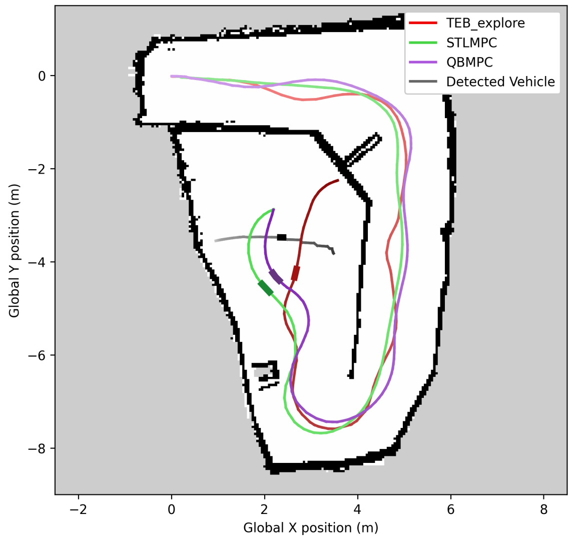


Figure 6.10: Local planner trajectories for dynamic obstacle avoidance in Experiment #5. Darker color gradients for each path show time progression, while one shared time sample before vehicle avoidance is shown by a point on each path.

Table 6.5: Planning times for QBMPC & other local planners in Experiment #5

Local Planner	$\bar{t}_{frontier}$ (ms)	$\bar{t}_{global}$ (ms)	$\bar{t}_{local}$ (ms)	$\boldsymbol{\bar{t}_{comp}}$ **(ms)**	$\max t_{frontier}$ (ms)	$\max t_{global}$ (ms)	$\max t_{local}$ (ms)	$\mathbf{max}\, \boldsymbol{t_{comp}}$ **(ms)**
TEB_explore	3.0	2.1	5.7	**10.8**	6.1	10.8	53.6	**70.4**
STLMPC				**29.4**				**57.8**
QBMPC				**1.2**				**3.2**

Chapter 7

Pursuit Model Predictive Control

The local path planning methods detailed throughout the thesis are now adapted to cooperative multi-vehicle contexts through a flexible pursuit formulation. Using vehicle detection & tracking from Chapter 4, detected vehicles with pursuable, feasible trajectories are followed instead of avoided as before. Here, the pursuit reformulation of Chapter 5's STLMPC (with a varying velocity) is designated as P-STLMPC, while Chapter 6's QBMPC is entitled P-QBMPC (P- denotes pursuit). These frameworks allow for decentralized multi-vehicle fleets where each agent adaptively balances pursuit formation with traversing a safe local path in the unknown environment. Each vehicle can dynamically join or leave the pursuit formation, performing its original path planning algorithm otherwise, and no fixed, designated leader is required.

The development of these adaptive pursuit algorithms is presented in this chapter. A summary of the approach & its features is documented before the framework for flexible, adaptive pursuit is outlined. The P-STLMPC & P-QBMPC algorithms are then formulated individually, starting from their original constructions. The performance of these methods is given in simulated & experimental multi-vehicle setups.

7.1 Overview

The STLMPC and QBMPC algorithms previously formulated provide local path planning in unknown environments with detected vehicles considered adversarially as obstacles for avoidance. However, some navigation applications use teams of mobile robots or autonomous vehicles that interact cooperatively and maintain a defined formation. Now, vehicle detection and tracking yield a desired target to pursue, as long as the path is achievable and the detected vehicle is heading away from the ego vehicle. Here, the original path planning approach is enhanced with a pursuit adaptation for integration in multi-vehicle, collaborative contexts.

Under normal circumstances, the original, chosen path planning algorithm is conducted; however, if a pursuable, tracked vehicle is identified, the ego vehicle seeks a path that maintains an arbitrary formation to the leader vehicle. When exclusively considering pursuit, the desired path lags the leader's position in the moving leader base frame by (x_ρ, y_ρ) generally. This enables any formation or arrangement to be chosen between the two vehicles when navigating. However, each vehicle is still responsive to the unknown surroundings during path planning and adaptively weighs the mode of operation between full pursuit based on the arbitrary formation, (x_ρ, y_ρ), and the original MPC algorithm for safe local path planning & obstacle avoidance.

An adaptive pursuit weight parameter ρ is introduced to dynamically update the emphasis placed on the pursuit and safe local planning objectives to achieve flexible pursuit in unknown environments. Using the minimum distance between the predicted path and the subsampled obstacles after each optimization, the pursuit weight ρ is updated to reflect the adaptive operation mode. If the optimized path has a close proximity to obstacles over multiple control steps, the pursuit weight is continually decreased in favor of prioritizing a safe generated path via either STLMPC or QBMPC. However, when obstacle proximity becomes less of a concern, pursuit is prioritized, which can slightly sacrifice the choice of a safe path in order to follow the leader in the designated formation.

This approach thereby extends the path planning framework to incorporate collaborative interactions with other vehicles in unstructured, unknown surroundings. Here, the leader detection indicator $\eta \in \{0, 1\}$ is used to enact adaptive pursuit when a leader is detected ($\eta = 1$) while disabling pursuit when no leader is found ($\eta = 0$). Detected vehicles that present collision risks are still dynamically avoided, and in the case of no detections, the algorithm reverts to the base path planner. No inter-vehicle communication is required as each vehicle dynamically responds individually to modularly construct or disassemble formation where required. When pursuing a leader vehicle, a minimum following proximity is satisfied to prevent vehicle collisions and provide space for the follower to respond in case of unpredictable leader motion.

No vehicle is explicitly designated the leader; instead, the vehicle with a leading position performs normal path planning while all trailing vehicles latch on to formation individually using pursuit-augmented path planning. Each vehicle can break off to perform standard local path planning if needed to preserve a safe path and can later rejoin the multi-vehicle formation if a leader is re-detected. Therefore, this setup provides a flexible framework for pursuit in changing environmental conditions to navigate safely as before while maintaining a scalable vehicle formation when possible. The case of two vehicles in a leader-follower scheme is primarily covered in this chapter, although this modular, decentralized framework forms a basis for accommodating more complex multi-vehicle fleet configurations.

7.2 Flexible Formation Framework

Using an adaptive pursuit weight based on obstacle proximity, ρ, and an arbitrary pursuit formation defined by (x_ρ, y_ρ), flexible multi-vehicle navigation is developed. The leader vehicle is assumed to path plan normally with no knowledge of a pursuing vehicle. The follower vehicle is taken as the ego vehicle in pursuit-augmented path planning, where the leader is detected and tracked as $\nu_{lead} = \nu_{track,0}$ according to Chapter 4. There is assumed to only be one tracked vehicle at a time for pursuit purposes in this leader-follower scheme, where, in the case of multiple simultaneous detections ($N_{track} > 1$), each is treated adversarially and dynamically avoided.

The present EKF estimated states for the leader vehicle are given in the follower vehicle base frame by $[x_{lead}, y_{lead}, \theta_{lead}, \delta_{lead}, v_{lead}]^T$. Depending on these states, the leader is either pursued and the leader detection indicator $\eta = 1$, or the trajectory is determined to be a collision risk or unable to be pursued. In this case, no leader is pursued, $\eta = 0$ and the vehicle is avoided. The necessary conditions where $\eta = 1$, since the detected leader is both pursuable and not a collision risk, are:

- $v_{lead} \leq v_{lead_{max}}$: The leader vehicle velocity is below a maximum threshold.
- $-\frac{\pi}{2} \leq \theta_{lead} \leq \frac{\pi}{2}$: The orientation of the leader vehicle in the follower's base frame is in a direction away from the follower vehicle.
- $-\delta_{max} \leq \delta_{lead} \leq \delta_{max}$: The leader vehicle's estimated steering angle is feasible.
- $\sqrt{x_{lead}^2 + y_{lead}^2} \leq d_{lead_{max}}$: The leader is within a reasonable distance of the pursuing vehicle.
- $x_{lead} > 0$: The detected vehicle is, in fact, leading in front of the ego vehicle.

- $|\hat{s}_{lead}(t_\xi)\kappa_{lead}| = \left|t_\xi v_{lead}\frac{\tan(\delta_{lead})}{l}\right| < 2\pi$ where $\hat{s}$ denotes arc length: Specifically for QBMPC, the leader's predicted trajectory doesn't make a full rotation over t_ξ, the future time horizon. This is to ensure the quartic Bezier curve model of the constant curvature trajectory remains accurate.

With a single tracked leader now, the set of obstacles used via subsampling in the optimization is similar to before. Now, though, the detected vehicle points according to the leader are used for pursuit, not within $\mathcal{O}_{obs_{det}}$ for dynamic obstacle avoidance. These obstacles, therefore, aren't included in the set for tracking line and optimization purposes. Furthermore, any sensor-based obstacle detections in $\mathcal{O}_{obs}$ corresponding to the leader vehicle's vicinity are removed to encourage pursuit and not avoidance:

$$\mathcal{O}_{obs} \leftarrow \mathcal{O}_{obs} \setminus \{(x_i, y_i) \in \mathcal{O}_{obs} \mid \|(x_i, y_i) - (x_{lead}, y_{lead})\|_2 \leq \mathrm{l_v}\} \tag{7.2.1}$$

All detections within the vehicle length, $\mathrm{l_v}$, of the leader's position are excluded from the obstacle set, which is then used for path planning as before.

Now, the P-STLMPC & P-QBMPC algorithms proceed with a shared framework, using the adaptive pursuit weight, ρ, to update the significance of each method meeting its respective pursuit objective as opposed to its respective original navigation objective. This weight is updated after each optimization using the minimum subsampled obstacle proximity along the predicted path. For P-STLMPC, the minimum proximity is:

$$d_{obs_{min}} = \min_{\substack{i\in\{0,\ldots,n_{MPC}k_{MPC}-1\} \\ (x_j,y_j)\in\mathcal{O}_{obs_{sub}}}} \|(x_i, y_i) - (x_j, y_j)\|_2 \tag{7.2.2}$$

while for P-QBMPC, it is:

$$d_{obs_{min}} = \min_{\substack{i \in \{0,\dots,n_\xi - 1\} \\ (x_j, y_j) \in \mathcal{O}_{obs_{sub}}}} \|(x_{\xi,i}, y_{\xi,i}) - (x_j, y_j)\|_2 \tag{7.2.3}$$

The weight factor is constrained to the range $\rho \in [0, 1]$ where $\rho = 0$ indicates original STLMPC or QBMPC while $\rho = 1$ means the new pursuit objective is exclusively considered. This temporal weight updates based on obstacle proximity, whether or not a leader vehicle is currently detected, such that if a new leader is found, adaptive pursuit can proceed based on existing safety conditions. When the minimum obstacle proximity, $d_{obs_{min}} < d_{nav}$, the original safe local navigation algorithm weighting $(1 - \rho)$ is increased by the maximum allowed change. On the other hand, if $d_{obs_{min}} > d_\rho$, the pursuit weighting, ρ is increased by the maximum allowed change. Under steady-state conditions (nearly constant $d_{obs_{min}}$), the weight stabilizes, balancing both objectives.

The adaptive pursuit weight update term, γ_ρ after each optimization is therefore:

$$\gamma_\rho = \begin{cases} -s_\rho & \text{if} \quad d_{obs_{min}} \leq d_{nav} \\ \frac{2s_\rho}{d_\rho - d_{nav}} d_{obs_{min}} - s_\rho - \frac{2s_\rho d_{nav}}{d_\rho - d_{nav}} & \text{if} \quad d_{nav} < d_{obs_{min}} < d_\rho \\ s_\rho & \text{if} \quad d_{obs_{min}} \geq d_\rho \end{cases} \tag{7.2.4}$$

where s_ρ denotes the maximum transition magnitude allowed in a single update. For $d_{nav} < d_{obs_{min}} < d_\rho$, the update term increases linearly as $d_{obs_{min}}$ increases according to Figure 7.1. The resulting adaptive pursuit weight is thus updated and bounded via:

$$\rho = \begin{cases} 0 & \text{if} \quad \rho + \gamma_\rho \leq 0 \\ \rho + \gamma_\rho & \text{if} \quad 0 < \rho + \gamma_\rho < 1 \\ 1 & \text{if} \quad \rho + \gamma_\rho \geq 1 \end{cases} \tag{7.2.5}$$

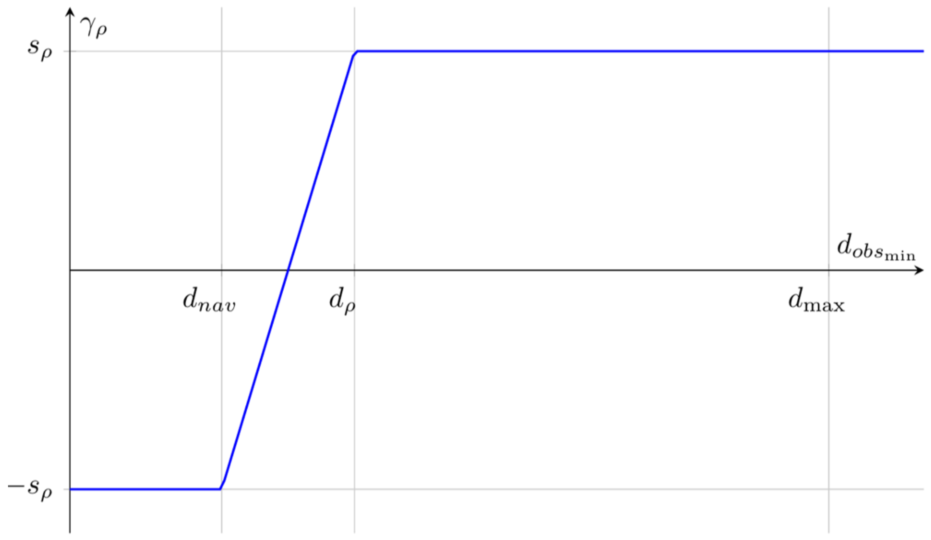


Figure 7.1: Update term, γ_ρ (added to the adaptive pursuit weight) as a linear function of minimum obstacle proximity for $d_{nav} < d_{obs_{min}} < d_\rho$. Here, the update magnitude is clipped at $-s_\rho$ when $d_{obs_{min}} \leq d_{nav}$ and at s_ρ when $d_{obs_{min}} \geq d_\rho$.

7.3 P-STLMPC Formulation

7.3.1 Initialization Procedure

The P-STLMPC approach extends beyond the design of STLMPC from Chapter 3 by dynamically weighing this original method against the new target of pursuit in real-time. Just as in STLMPC, this new approach uses the full set of obstacles (excluding those associated with the leader vehicle) to generate n_{MPC} successive tracking lines. Using these tracking line headings and the last commanded steering angle & velocity via Algorithm 2, the initial feasible guess for the original STLMPC approach is denoted by the variables $x_{nav,i}, y_{nav,i}, \theta_{nav,i}, \delta_{nav,i}, v_{nav,i}$ where $i \in \{0, ..., n_{MPC}k_{MPC}-1\}$.

Now, however, in the case that a leader is detected ($\eta = 1$), the starting guess is modified to also achieve pursuit according to the current adaptive weighting. The balance of these objectives in formulating an effective initial guess is significant for attaining an effective local optimum from the non-convex solver. The initial guess formulation incorporates $x_{pur,i}, y_{pur,i}, \theta_{pur,i}, \delta_{pur,i}, v_{pur,i}$ which provides a starting point for the pursuit optimization & is weighed with the original algorithm's guess according to ρ while maintaining feasibility.

The constant curvature model for the leader's trajectory corresponds to a circular arc with the signed radius (+ for counterclockwise rotation, − if clockwise):

$$r_{lead} = \frac{1}{\kappa_{lead}} = \frac{l}{\tan(\delta_{lead})} \tag{7.3.1}$$

where $\delta_{lead} \leftarrow \operatorname{sgn}(\delta_{lead}) \cdot \max(|\delta_{lead}|, \delta_{lead_{min}})$ is bounded by a minimum magnitude, $\delta_{lead_{min}}$ in order to avoid singularities & prevent numerical instability when dividing by small values. The leader position along the future estimated trajectory in the leader's base frame (discretizing using the control step time, Δt) is then:

$$(x_{lead,i}, y_{lead,i}) = (r_{lead} \sin(\frac{i\, v_{lead}\, \Delta t}{r_{lead}}), r_{lead}(1 - \cos(\frac{i\, v_{lead}\, \Delta t}{r_{lead}}))) \;\; \forall i \in \{0, ..., n_{MPC}k_{MPC} - 1\} \tag{7.3.2}$$

while the future velocity vector in the leader vehicle's current base frame becomes:

$$[v_{lead_x,i}, v_{lead_y,i}]^T = v_{lead} \cdot [\cos(\frac{i\, v_{lead}\, \Delta t}{r_{lead}}),\ \sin(\frac{i\, v_{lead}\, \Delta t}{r_{lead}})]^T \;\; \forall i \in \{0, ..., n_{MPC}k_{MPC} - 1\} \tag{7.3.3}$$

The velocity vector is then transformed to the pursuing vehicle's base frame, and the heading angle is acquired through:

$$\begin{bmatrix} v_{lead_x,i} \\ v_{lead_y,i} \end{bmatrix} \leftarrow R(\theta_{lead}) \cdot \begin{bmatrix} v_{lead_x,i} \\ v_{lead_y,i} \end{bmatrix} = \begin{bmatrix} \cos(\theta_{lead}) & -\sin(\theta_{lead}) \\ \sin(\theta_{lead}) & \cos(\theta_{lead}) \end{bmatrix} \cdot \begin{bmatrix} v_{lead_x,i} \\ v_{lead_y,i} \end{bmatrix} \tag{7.3.4}$$

$$\theta_{lead,i} = \operatorname{atan2}(v_{lead_y,i}, v_{lead_x,i}) \tag{7.3.5}$$

To pursue the leader with a lagging position, (x_ρ, y_ρ) in the leader's moving frame, the desired pursuit path positions, $(x_{\rho,i}, y_{\rho,i})$ and velocities, $(v_{\rho_x,i}, v_{\rho_y,i})$ are found. Transforming from the evolving leader base frame to the current follower frame over the path planning time horizon, the discretized reference pursuit trajectory that maintains the desired formation is generated in the ego vehicle base frame (Figure 7.2):

$$\begin{bmatrix} x_{\rho,i} \\ y_{\rho,i} \end{bmatrix} = \begin{bmatrix} x_{lead} \\ y_{lead} \end{bmatrix} + R(\theta_{lead}) \cdot \begin{bmatrix} x_{lead,i} \\ y_{lead,i} \end{bmatrix} + R(\theta_{lead,i}) \cdot \begin{bmatrix} -x_\rho \\ -y_\rho \end{bmatrix} \tag{7.3.6}$$

where the rotation matrices $R(\theta_{lead})$ & $R(\theta_{lead,i})$ follow the same form as in Equation 7.3.4. The transformed velocity vector in the follower's frame for each point on the discretized reference pursuit trajectory ($i \in \{0, ..., n_{MPC}k_{MPC} - 1\}$) incorporates the standard rigid-body velocity relation [138] through:

$$\begin{bmatrix} v_{\rho_x,i} \\ v_{\rho_y,i} \end{bmatrix} = R(\theta_{lead})(\begin{bmatrix} v_{lead_x,i} \\ v_{lead_y,i} \end{bmatrix} + \frac{v_{lead}}{r_{lead}} M(\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times \begin{bmatrix} -x_\rho \\ -y_\rho \\ 0 \end{bmatrix})) = R(\theta_{lead})(\begin{bmatrix} v_{lead_x,i} \\ v_{lead_y,i} \end{bmatrix} + \frac{v_{lead}}{r_{lead}} M \begin{bmatrix} y_\rho \\ -x_\rho \\ 0 \end{bmatrix}) \tag{7.3.7}$$

expressing both the translational leader velocity and the lagging pursuit position's rotational velocity. The 2D vector is restored from the cross product by $M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$.

Now that the ideal pursuit path is formulated, the initial guess seeks to follow it while maintaining feasibility. The procedure shown in Algorithm 2 is again used, now comparing $\theta_{pur,i}$ to atan2$(y_{\rho,i} - y_{pur,i}, x_{\rho,i} - x_{pur,i})$ and choosing the $\delta_{pur,i}$ that aligns the orientation with the heading direction needed to reach the ideal pursuit path's current point.

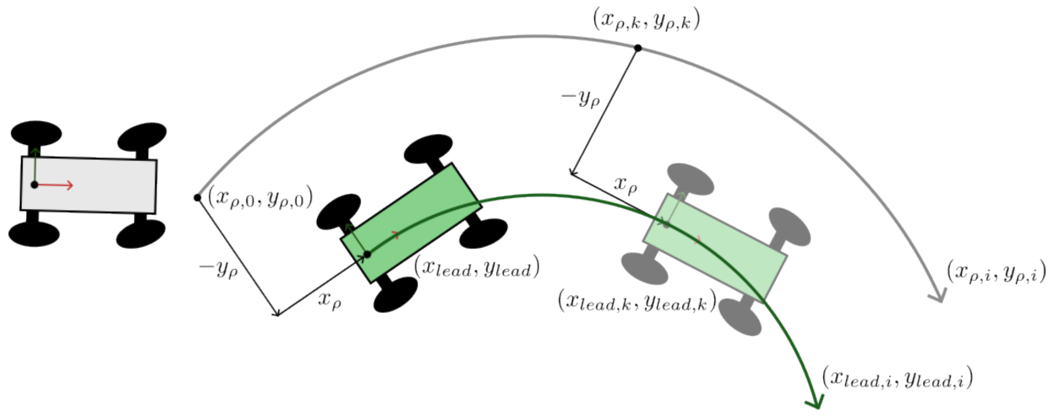

$(x_{\rho,k}, y_{\rho,k})$
$-y_\rho$
x_ρ
$(x_{\rho,0}, y_{\rho,0})$
$-y_\rho$
x_ρ
(x_{lead}, y_{lead})
$(x_{lead,k}, y_{lead,k})$
$(x_{\rho,i}, y_{\rho,i})$
$(x_{lead,i}, y_{lead,i})$

Figure 7.2: The leader vehicle, a future predicted position (at sample k) and its constant curvature path (green) for $(x_{lead,i}, y_{lead,i})$. The lagging position with respect to the leader's frame, (x_ρ, y_ρ), is negative in y by convention since the leader is right of the lagging position. The ideal pursuit trajectory (gray) is given by $(x_{\rho,i}, y_{\rho,i})$.

In turn, $x_{pur,i}, y_{pur,i}$ & $\theta_{pur,i}$ are updated after each steering angle is found and all velocities are set as $v_{pur,i} = v_{lead}$. Additionally, each $\delta_{pur,i}$ can be set to δ_{lead} if initially near the desired pursuit path ($\sqrt{(x_{\rho,0} - x_{pur,0})^2 + (y_{\rho,0} - y_{pur,0})^2} \leq d_{\rho,thresh}$) to provide a smooth, constant curvature initial guess for the pursuit path without unnecessary fluctuations.

The initial guesses based on tracking lines and pursuit are now combined by weighing the positions of the trajectories according to ρ, where the weighted path is bounded by the extent of each prior guess path:

$$(x_i, y_i) = ((1-\rho)x_{nav,i} + \rho x_{pur,i}, (1-\rho)y_{nav,i} + \rho y_{pur,i}) \tag{7.3.8}$$

with the orientations, $\theta_j = \text{atan2}(y_{j+1} - y_j, x_{j+1} - x_j)\ \forall j \in \{0, ..., n_{MPC}k_{MPC} - 2\}$ which are wrapped to the correct interval. Now, the feasible initial guess for the sampled velocity ($j \in \{1, ..., n_{MPC}k_{MPC} - 2\}$) is found through:

$$v_j = \frac{\sqrt{(x_{j+1} - x_j)^2 + (y_{j+1} - y_j)^2}}{\Delta t},\ v_{n_{MPC}k_{MPC}-1} = v_{n_{MPC}k_{MPC}-2} \tag{7.3.9}$$

Algorithm 4: P-STLMPC Initial Guess Procedure

```
Input: Tracking Lines ($\tilde{w}_i$), Last Control Inputs ($\delta_{last}, v_{last}$), Lagging Pursuit
       Position ($x_\rho, y_\rho$), Leader Detection Indicator ($\eta$), Leader States
       ($\nu_{lead}$), Adaptive Pursuit Weighting ($\rho$)
Output: Initial P-STLMPC Guess ($x_i, y_i, \theta_i, \delta_i, v_i$)
1  if $\eta = 1$ then
2      $x_{nav,i}, y_{nav,i}, \theta_{nav,i}, \delta_{nav,i}, v_{nav,i} \leftarrow$ STLMPC_Guess($\tilde{w}_i, \delta_{last}, v_{last}$);
3      $x_{\rho,i}, y_{\rho,i} \leftarrow$ RefPursuitPath($\nu_{lead}, x_\rho, y_\rho$);
4      for $i = 0$ to $n_{MPC}k_{MPC} - 1$ do
5          $v_{pur,i} = v_{lead}$;
6          if $i > 0$ and $\sqrt{x_{\rho,0}^2 + y_{\rho,0}^2} > d_{\rho,thresh}$ then
7              $\delta_{pur,i} \leftarrow$
                 TurnToRefPath($x_{\rho,i}, y_{\rho,i}, x_{pur,i}, y_{pur,i}, \theta_{pur,i}, \delta_{pur,i-1}, v_{pur,i}, \delta_{max}, \Delta\delta_{max}$);
8          else if $i > 0$ and $\sqrt{x_{\rho,0}^2 + y_{\rho,0}^2} \leq d_{\rho,thresh}$ then
9              $\delta_{pur,i} \leftarrow$ MatchLeaderCurvature($\delta_{pur,i-1}, \delta_{lead}, \delta_{max}, \Delta\delta_{max}$);
10         else
11             $\delta_{pur,i} = \delta_{last},\ x_{pur,i} = 0,\ y_{pur,i} = 0,\ \theta_{pur,i} = 0$;
12         if $i < n_{MPC}k_{MPC} - 1$ then
13             $x_{pur,i+1}, y_{pur,i+1}, \theta_{pur,i+1} \leftarrow$
                 BicycleKinematics($x_{pur,i}, y_{pur,i}, \theta_{pur,i}, \delta_{pur,i}, v_{pur,i}$);
14     end
15     $x_0, y_0, \theta_0 = 0$;
16     for $i = 1$ to $n_{MPC}k_{MPC} - 1$ do
17         $x_i, y_i \leftarrow$ WeighPositions($x_{nav,i}, y_{nav,i}, x_{pur,i}, y_{pur,i}, \rho$);
18         $v_{i-1} \leftarrow$ FeasibleVelocityForwardDiff($x_{i-1}, y_{i-1}, x_i, y_i, v_{max}, v_{min}$);
19         if $i > 1$ then
20             $\theta_{i-1} \leftarrow$ FindOrientations($x_{i-1}, y_{i-1}, x_i, y_i$);
21             $\delta_{i-2} \leftarrow$ FeasibleSteeringAngles($\theta_{i-2}, \theta_{i-1}, \delta_{i-3}, v_{i-2}, \delta_{max}, \Delta\delta_{max}$);
22     end
23     $v_{n_{MPC}k_{MPC}-1} = v_{n_{MPC}k_{MPC}-2},\ \delta_{n_{MPC}k_{MPC}-1} = \delta_{n_{MPC}k_{MPC}-2} = \delta_{n_{MPC}k_{MPC}-3}$;
24     $\delta_0 = \delta_{last}, \quad v_0 = v_{last}$;
25     for $i = 0$ to $n_{MPC}k_{MPC} - 2$ do
26         $x_{i+1}, y_{i+1}, \theta_{i+1} \leftarrow$ BicycleKinematics($x_i, y_i, \theta_i, \delta_i, v_i$);
27     end
28 else if $\eta = 0$ then
29     $x_i, y_i, \theta_i, \delta_i, v_i \leftarrow$ STLMPC_Guess($\tilde{w}_i, \delta_{last}, v_{last}$);
30 Begin P-STLMPC Optimization using $x_i, y_i, \theta_i, \delta_i, v_i\ \forall i \in \{0, ..., n_{MPC}k_{MPC} - 1\}$;
```

and bounded to the feasible range, $v_{min} \leq v_j, v_{n_{MPC}k_{MPC}-1} \leq v_{max}$. The steering angle is then found based on using successive orientations in the kinematic bicycle model ($j \in \{1, ..., n_{MPC}k_{MPC} - 3\}$):

$$\delta_j = \tan^{-1}\left(\frac{l(\theta_{j+1} - \theta_j)}{v_j\,\Delta t}\right),\ \delta_{n_{MPC}k_{MPC}-1} = \delta_{n_{MPC}k_{MPC}-2} = \delta_{n_{MPC}k_{MPC}-3} \tag{7.3.10}$$

The steering angles are then bounded by the maximum allowed magnitude & rate of change (per Algorithm 2's structure). Initial conditions fix $\delta_0 = \delta_{last}$ and $v_0 = v_{last}$ while the final x_i, y_i & θ_i come from applying the kinematic bicycle model once more with the final δ_i & v_i formulated.

Therefore, this initial guess approach appropriately weighs the tracking line & pursuit guesses before obtaining the weighted trajectory's corresponding control inputs. For samples when the control input is infeasible, the closest feasible value is used before proceeding as usual for subsequent samples. A feasible trajectory is thus generated in this method (Algorithm 4), beginning the optimization with a suitable guess based on varying ρ & aiming to initialize near the region of the global optimum.

7.3.2 Adaptive Tracking Line vs. Pursuit Objective

The objective for P-STLMPC adaptively weighs terms that promote following the generated reference tracking lines with terms that achieve pursuit in formation. This approach uses both the varying pursuit weight, ρ, and the current leader detection indicator, η. The complete adaptive pursuit objective is therefore given by:

$$\begin{aligned} F_{obj_{\rho,w}} = {} & (1-\eta\rho)\lambda_{d^2}F_{d^2} + (1-\eta\rho)\lambda_{\dot{d}^2}F_{\dot{d}^2} + \eta(1-\rho)\lambda_{v^2}F_{v_\rho^2} + \lambda_{\delta^2}F_{\delta^2} \\ & + (1-\eta)\lambda_{v^2}F_{\Delta v^2} + \eta\rho\lambda_{\rho,d^2}F_{\rho,d^2} + \eta\rho\lambda_{\rho,\dot{d}^2}F_{\rho,\dot{d}^2} \end{aligned} \tag{7.3.11}$$

with base weighting factors, $\lambda_{\rho,w} = [\lambda_{d^2}, \lambda_{\dot{d}^2}, \lambda_{v^2}, \lambda_{\delta^2}, \lambda_{\rho,d^2}, \lambda_{\rho,\dot{d}^2}]$ where normalizing all weights by one weight factor removes the redundancy in parameters.

Here, the objective in the case of no leader detection ($\eta = 0$) is equivalent to $\lambda_{d^2}F_{d^2}+\lambda_{\dot{d}^2}F_{\dot{d}^2}+\lambda_{\delta^2}F_{\delta^2}+\lambda_{v^2}F_{\Delta v^2}$ which corresponds to non-uniform velocity STLMPC (Equation 5.2.2) but with a modified velocity-based objective term, $F_{\Delta v^2}$ instead of F_{v^2}. Meanwhile, if a leader is detected ($\eta = 1$), the adaptive pursuit weight balances the tracking line formulation terms, $F_{d^2}, F_{\dot{d}^2}$ & $F_{v_\rho^2}$ (where $F_{v_\rho^2}$ is a newly developed velocity term), with the pursuit terms, F_{ρ,d^2} & $F_{\rho,\dot{d}^2}$ while using a constant steering angle (control effort) term, F_{δ^2}. For higher ρ, the pursuit terms have higher relative weights, while for lower ρ, the STLMPC-based terms are favored in optimization.

Instead of prioritizing higher speeds, this approach seeks to navigate at a safe speed when performing standard STLMPC while matching the leader's speed in the case of pursuit. Thus, the new velocity-based term in the case of no leader is given:

$$F_{\Delta v^2} = \sum_{i=0}^{n_{MPC}k_{MPC}-2} (v_{i+1} - v_i)^2 \tag{7.3.12}$$

This term ensures that unnecessary changes in velocity (acceleration or deceleration) are avoided and a relatively constant velocity is maintained for smoothness, where possible. Initializing the optimization with a moderate, constant speed such as $\frac{v_{max}}{2}$ ensures safe velocities are sustained throughout. Furthermore, for temporary missed leader detections, the vehicle will progress at a moderate speed, attempting to remain behind the leader and prevent accidental acceleration into the back of the leader.

When a leader is detected and tracked, a term is incorporated to align the follower's velocity with that of the estimated leader:

$$F_{v_\rho^2} = \sum_{i=0}^{n_{MPC}k_{MPC}-1} (v_i - v_{lead})^2 \tag{7.3.13}$$

This term is given higher weight when ρ is less, since it corresponds to the STLMPC-based terms. When a leader is known but obstacles are in close proximity, STLPMC is prioritized, using $F_{v_\rho^2}$ to ensure the leader's velocity is matched. This means space is maintained between the vehicles, and effective pursuit occurs even when it is not explicit. When obstacle clearance increases over time once more, pursuit is weighed more heavily and the follower's velocity is allowed to vary & adjust to follow the reference pursuit path, thus maintaining formation.

Now, the pursuit terms are constructed which promote a trajectory that maintains the arbitrary formation, (x_ρ, y_ρ), with the leader. The sampled pursuit reference path that maintains this lagging position in the leader's frame was previously derived as $(x_{\rho,i}, y_{\rho,i})$ $\forall i \in \{0, ..., n_{MPC}k_{MPC} - 1\}$ (Equation 7.3.6). Thus, a term similar to the tracking line term F_{d^2} is created, which seeks to minimize the sum of squared distances between the sampled predicted trajectory and the pursuit reference path:

$$F_{\rho,d^2} = \sum_{i=0}^{n_{MPC}k_{MPC}-1} ((x_i - x_{\rho,i})^2 + (y_i - y_{\rho,i})^2) \tag{7.3.14}$$

Similarly to STLMPC's $F_{\dot{d}^2}$, a squared derivative term is also utilized, which reduces path fluctuations and aligns the predicted trajectory's orientation with that of the pursuit reference path at each sample. Based on Equation 7.3.14:

$$\sum_{i=0}^{n_{MPC}k_{MPC}-1} d_{\rho,i} = \sum_{i=0}^{n_{MPC}k_{MPC}-1} \sqrt{(x_i - x_{\rho,i})^2 + (y_i - y_{\rho,i})^2} \tag{7.3.15}$$

$$\sum_{i=0}^{n_{MPC}k_{MPC}-1} \dot{d}_{\rho,i} = \sum_{i=0}^{n_{MPC}k_{MPC}-1} \frac{(x_i - x_{\rho,i})(\dot{x}_i - \dot{x}_{\rho,i}) + (y_i - y_{\rho,i})(\dot{y}_i - \dot{y}_{\rho,i})}{d_{\rho,i}} \tag{7.3.16}$$

$$\sum_{i=0}^{n_{MPC}k_{MPC}-1} \dot{d}^2_{\rho,i} = \sum_{i=0}^{n_{MPC}k_{MPC}-1} \frac{((x_i - x_{\rho,i})(\dot{x}_i - \dot{x}_{\rho,i}) + (y_i - y_{\rho,i})(\dot{y}_i - \dot{y}_{\rho,i}))^2}{(x_i - x_{\rho,i})^2 + (y_i - y_{\rho,i})^2} \tag{7.3.17}$$

Now, the derivatives of the predicted trajectory's position are found by forward differencing: $\dot{x}_i = x_{i+1} - x_i$ & $\dot{y}_i = y_{i+1} - y_i$. This restricts the sum to the first $n_{MPC}k_{MPC} - 1$ elements where the forward difference is defined. The derivatives of the reference pursuit path's positions, $x_{\rho,i}$ & $y_{\rho,i}$ are given by the respective velocity components, $v_{\rho_x,i}$ & $v_{\rho_y,i}$ as expressed in Equation 7.3.7. Therefore, the final pursuit objective term is described by:

$$F_{\rho,\dot{d}^2} = \sum_{i=0}^{n_{MPC}k_{MPC}-2} \frac{((x_i - x_{\rho,i})(x_{i+1} - x_i - v_{\rho_x,i}) + (y_i - y_{\rho,i})(y_{i+1} - y_i - v_{\rho_y,i}))^2}{(x_i - x_{\rho,i})^2 + (y_i - y_{\rho,i})^2} \tag{7.3.18}$$

Using these newly formulated pursuit terms, the leader vehicle is able to be followed in a flexible framework (Equation 7.3.11) that also balances safe local navigation for varying obstacle-based safety considerations via ρ. The analytical gradients for all new objective terms presented in this section are provided in Appendix A.4.

7.3.3 Minimum Leader-Follower Proximity Constraint

While achieving adaptive pursuit via the aforementioned approach, additional safety requirements must be realized. All prior constraints on vehicle dynamics, permissible control inputs and obstacle proximity are carried over from Chapter 5. Now, a pursuit constraint is introduced specifically for when a leader is detected. In order to prevent collisions with the leader, a minimum following distance is satisfied over the full predicted trajectory. This limit ensures that, in the event the leader stops or moves unexpectedly, space and time are preserved for the follower to react and avoid a collision.

Akin to prior minimum distance constraints (Equations 5.3.13 & 6.4.16), a smooth softmin function is used, ensuring analytical gradients can be found (Appendix A.4). The future leader position (Equation 7.3.2) is transformed to the follower's frame via:

$$\begin{bmatrix} x_{lead,i} \\ y_{lead,i} \end{bmatrix} \leftarrow \begin{bmatrix} x_{lead} \\ y_{lead} \end{bmatrix} + R(\theta_{lead}) \cdot \begin{bmatrix} x_{lead,i} \\ y_{lead,i} \end{bmatrix} \tag{7.3.19}$$

since the leader position is directly tracked and not the lagging pursuit position given in Equation 7.3.6. Instead of directly using the leader's transformed position though, the four corners of the vehicle's shape are derived using the leader's future heading angle, $\theta_{lead,i}$ (Equation 7.3.5) and physical length & width dimensions ($\mathrm{l_v}$ & $\mathrm{w_v}$):

$$\hat{C}_{lead,i} = \begin{bmatrix} \cos(\theta_{lead,i}) & -\sin(\theta_{lead,i}) \\ \sin(\theta_{lead,i}) & \cos(\theta_{lead,i}) \end{bmatrix} \cdot \begin{bmatrix} \frac{\mathrm{l_v}}{2} & \frac{\mathrm{l_v}}{2} & -\frac{\mathrm{l_v}}{2} & -\frac{\mathrm{l_v}}{2} \\ -\frac{\mathrm{w_v}}{2} & \frac{\mathrm{w_v}}{2} & \frac{\mathrm{w_v}}{2} & -\frac{\mathrm{w_v}}{2} \end{bmatrix} + \begin{bmatrix} x_{lead,i} \\ y_{lead,i} \end{bmatrix} \cdot \mathbf{1}^T \tag{7.3.20}$$

with $\mathbf{1}$, the 4 x 1 column vector of ones. This expression is similar to Equation 4.4.5 (used in dynamic vehicle avoidance), where now $\hat{C}_{lead,i}$ denotes the leader vehicle's four corners at the i^{th} sample while $\hat{c}_{lead,i,n},\ n \in \{0, ..., 3\}$ represents each individual corner (Figure 7.3). The distance between the predicted trajectory and the leader's n^{th} corner at sample i is:

$$d_{lead,i,n} = \sqrt{(x_i - \hat{c}_{lead_x,i,n})^2 + (y_i - \hat{c}_{lead_y,i,n})^2} \quad \forall i \in \{0, ..., n_{MPC}k_{MPC} - 1\},\ n \in \{0, ..., 3\} \tag{7.3.21}$$

Therefore, the resulting soft minimum distance between follower and leader over the trajectory is formulated as:

$$d_{lead_{min}} = -\frac{1}{\beta_\rho} \log\Big(\sum_{i=0}^{n_{MPC}k_{MPC}-1} \sum_{n=0}^{3} e^{-\beta_\rho d_{lead,i,n}} \Big) \tag{7.3.22}$$

The use of β_ρ controls the accuracy of the softmin function in this context, similar to β_v & β_ξ used previously. As before, higher values correspond to closer approximations to the true minimum at the risk of poorer numerical stability.

Finally, the single constraint is formed to ensure the closest inter-vehicle proximity over the trajectory exceeds the minimum safe following distance, $d_{lead_{safe}}$, through:

$$\begin{aligned} & d_{lead_{safe}} \leq d_{lead_{min}} \quad \text{when} \quad \eta = 1 \\ \text{if} \quad \eta = 1, \quad & g_{d_{lead}} = \frac{1}{\beta_\rho} \log\Big(\sum_{i=0}^{n_{MPC} k_{MPC}-1} \sum_{n=0}^{3} e^{-\beta_\rho d_{lead,i,n}} \Big) + d_{lead_{safe}}, \quad g_{d_{lead}} \leq 0 \end{aligned} \tag{7.3.23}$$

where the inequality is only considered if a leader vehicle is detected. This ensures that, in practice, for no leader detection, the constraint is not included in the active set during solving via SLSQP.

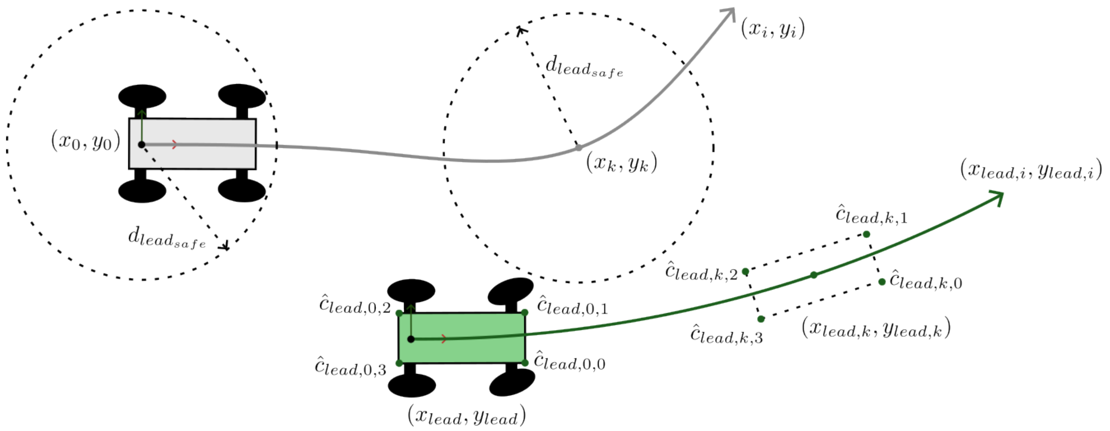


Figure 7.3: The predicted ego vehicle trajectory (gray), which maintains a separation of $d_{lead_{safe}}$ from the leader (green) throughout. Here, the sampled paths are shown for visualization at $i = 0, k$ where k is an arbitrary future sample. The leader vehicle is characterized by its box-bounded corner points, $\hat{c}_{lead,i,n}$, for ensuring a safe following distance is maintained.

7.4 P-QBMPC Formulation

7.4.1 Bezier Curve Initialization

As an alternative to P-STLMPC, the P-QBMPC algorithm is presented, which is inspired by Chapter 6's QBMPC method. This approach has a similar framework to P-STLMPC, where the adaptive pursuit weight is now used to balance safe planning via potential fields with pursuit. Initializing a path guess for optimization is done as in P-STLMPC, where here, the original QBMPC guess is weighed with the new pursuit guess based on ρ. The QBMPC guess obtained using successive safest angular gap headings is given by $x_{nav_\xi,2}, x_{nav_\xi,3}, y_{nav_\xi,3}, x_{nav_\xi,4}, y_{nav_\xi,4}$, which is used exclusively if $\eta = 0$. However, when $\eta = 1$, the pursuit initial guess, $x_{pur_\xi,2}, x_{pur_\xi,3}, y_{pur_\xi,3}, x_{pur_\xi,4}, y_{pur_\xi,4}$, is incorporated into the final guess as well.

For path initialization to achieve effective pursuit, a quartic Bezier curve model of the leader's constant curvature & velocity trajectory is constructed. An approach [139] is used, which achieves an approximation order of eight & a lower Hausdorff distance between arc and approximation than prior methods when the central angle, $0 < \tilde{\alpha}_{lead_\xi} < \frac{\pi}{2}$. The presented model is designed to approximate circular arcs of unit radius with $0 < \tilde{\alpha}_{lead_\xi} < \pi$ by the quartic Bezier curve defined by the control points:

$$\begin{aligned}
&(\tilde{x}_{lead_\xi,0}, \tilde{y}_{lead_\xi,0}) = (1,0), \quad (\tilde{x}_{lead_\xi,1}, \tilde{y}_{lead_\xi,1}) = (1, \tilde{u}_{lead_\xi}) \\
&(\tilde{x}_{lead_\xi,2}, \tilde{y}_{lead_\xi,2}) = \tilde{r}_{lead_\xi}(\cos(\frac{\tilde{\alpha}_{lead_\xi}}{2}), \sin(\frac{\tilde{\alpha}_{lead_\xi}}{2})) \\
&(\tilde{x}_{lead_\xi,3}, \tilde{y}_{lead_\xi,3}) = (\cos(\tilde{\alpha}_{lead_\xi}), \sin(\tilde{\alpha}_{lead_\xi})) + \tilde{u}_{lead_\xi}(\sin(\tilde{\alpha}_{lead_\xi}), -\cos(\tilde{\alpha}_{lead_\xi})) \\
&(\tilde{x}_{lead_\xi,4}, \tilde{y}_{lead_\xi,4}) = (\cos(\tilde{\alpha}_{lead_\xi}), \sin(\tilde{\alpha}_{lead_\xi}))
\end{aligned} \tag{7.4.1}$$

where:

$$\tilde{r}_{lead_\xi} = \frac{8}{3} - \frac{5}{3}\cos(\frac{\tilde{\alpha}_{lead_\xi}}{2}) - \frac{4}{3}\tilde{u}_{lead_\xi}\sin(\frac{\tilde{\alpha}_{lead_\xi}}{2}) \tag{7.4.2}$$

$$\tilde{u}_{lead_\xi} = \frac{3\cos(\frac{\tilde{\alpha}_{lead_\xi}}{2})\sin(\frac{\tilde{\alpha}_{lead_\xi}}{2}) - 2\sin(\frac{\tilde{\alpha}_{lead_\xi}}{2}) + 4\sqrt{2}\sin^3(\frac{\tilde{\alpha}_{lead_\xi}}{4})}{2\cos^2(\frac{\tilde{\alpha}_{lead_\xi}}{2})} \tag{7.4.3}$$

For the leader's full trajectory over the predicted horizon, the central angle, $\tilde{\alpha}_{lead_\xi} = \frac{\hat{s}_{lead}(t_\xi)}{r_{lead}}$ is derived from the arc radius (Equation 7.3.1) and total arc length, $\hat{s}_{lead}(t_\xi) = t_\xi v_{lead}$. The signed radius is again used here, indicating the rotational direction of the circular arc's progression depending on δ_{lead}. This approximation is applied to represent trajectories with central angles, $-2\pi < \tilde{\alpha}_{lead_\xi} < 2\pi$, over the predicted horizon, where reduced—although acceptable—accuracy for increasing angles is shown in Figure 7.4. While full rotations are unlikely over a short prediction time horizon, for an arbitrary and sufficiently long t_ξ, these motions can be adequately modeled by this approximation.

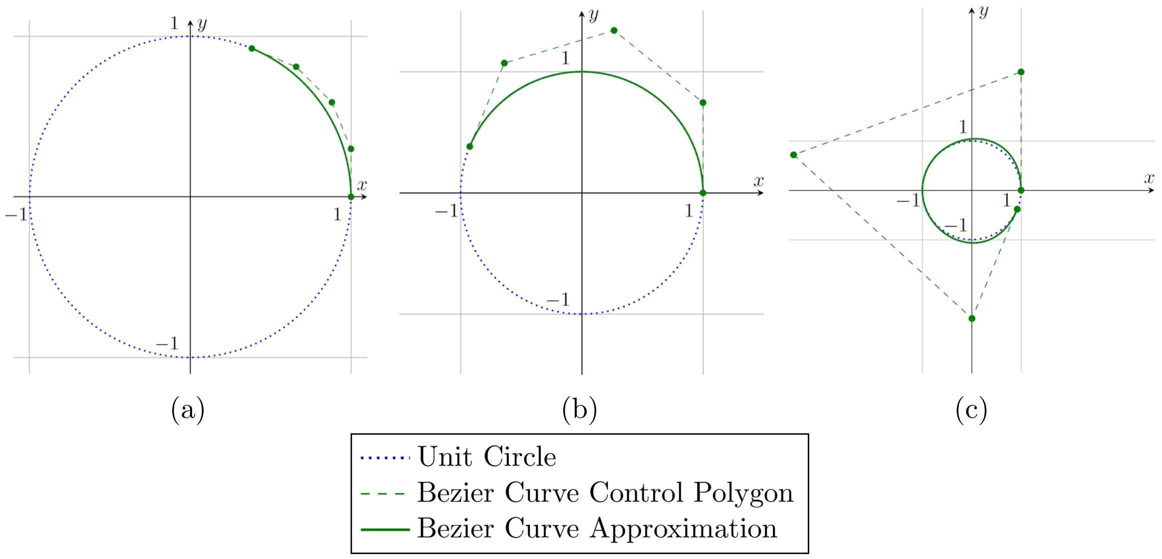


Figure 7.4: Quartic Bezier curve approximations to unit circular arcs with corresponding control points & polygons for increasing central angles. The model is shown for: (a) $\tilde{\alpha}_{lead_\xi} = \frac{3\pi}{8}$, (b) $\tilde{\alpha}_{lead_\xi} = \frac{7\pi}{8}$ and (c) $\tilde{\alpha}_{lead_\xi} = \frac{15\pi}{8}$.

The control points and their corresponding Bezier curve for the unit circular arc are now transformed in the general case of arbitrary radius, orientation and position for application in pursuit. The curve is first scaled by the radius magnitude, $|r_{lead}|$ and translated to begin at the origin via:

$$\tilde{x}_{lead_\xi,k} \leftarrow |r_{lead}|\,(\tilde{x}_{lead_\xi,k} - 1), \quad \tilde{y}_{lead_\xi,k} \leftarrow |r_{lead}|\,(\tilde{y}_{lead_\xi,k}), \quad k \in \{0, ..., 4\} \tag{7.4.4}$$

Now, the curve with positive (counterclockwise) rotation begins in the $+y$ direction while the curve with negative (clockwise) rotation proceeds initially in the $-y$ direction. Thus, the $+y$ direction is taken as the base frame $+x$ direction (since this is assumed to be the direction of forward velocity) while $+x$ is represented as $-y$ in the vehicle base frame. The scaled approximated Bezier curve control points are now transformed to the follower's base frame, obtaining $(x_{lead_\xi,k}, y_{lead_\xi,k})$ using this coordinate transform and the leader's position & orientation:

$$\begin{bmatrix} x_{lead_\xi,k} \\ y_{lead_\xi,k} \end{bmatrix} = \begin{bmatrix} x_{lead} \\ y_{lead} \end{bmatrix} + R(\theta_{lead}) \cdot \begin{bmatrix} \tilde{y}_{lead_\xi,k} \\ -\tilde{x}_{lead_\xi,k} \end{bmatrix}, \quad k \in \{0, ..., 4\} \tag{7.4.5}$$

Note that for this calculation, $\theta_{lead} + \pi$ is used in place of θ_{lead} if δ_{lead} is negative in order to orient the approximation's initial $-y$ direction motion correctly to the $+x$ base frame direction.

To complete the initial pursuit path guess, the leader's Bezier curve control points are used to find the lagging pursuit path's control points. The leader's curve is first sampled by $t_k = \frac{k}{4}, \quad k \in \{0, ..., 4\}$, reflecting a sample for each control point. Using the control points $(x_{lead_\xi,k}, y_{lead_\xi,k})$ and sampled t_k values, the curve is represented by $(x_{lead_\xi,\xi,k}, y_{lead_\xi,\xi,k})$ through the form of Equations 6.1.2, 6.1.3 & 6.3.1. The first derivative is $(x'_{lead_\xi,\xi,k}, y'_{lead_\xi,\xi,k})$ which follows Equations 6.1.4 & 6.1.5, now incorporating the nonzero terms $(4t_k^3 - 12t_k^2 + 12t_k - 4)x_{lead_\xi,0}$ & $(4t_k^3 - 12t_k^2 + 12t_k - 4)y_{lead_\xi,0}$ respectively in each coordinate.

From here, the leader's future orientation along the predicted Bezier curve in the follower's frame is:

$$\theta_{lead_\xi,k} = \text{atan2}(y'_{lead_\xi,\xi,k}, x'_{lead_\xi,\xi,k}) \tag{7.4.6}$$

Assuming the evenly spaced samples correspond approximately to each control point for the constant curvature & velocity trajectory, each point is transformed (similarly to Equation 7.3.6) to the path lagging by (x_ρ, y_ρ) using its sample's orientation:

$$\begin{bmatrix} x_{pur_\xi,k} \\ y_{pur_\xi,k} \end{bmatrix} = \begin{bmatrix} x_{lead_\xi,k} \\ y_{lead_\xi,k} \end{bmatrix} + R(\theta_{lead_\xi,k}) \cdot \begin{bmatrix} -x_\rho \\ -y_\rho \end{bmatrix}, \quad k \in \{0, ..., 4\} \tag{7.4.7}$$

These control points represent the reference pursuit path that attains the arbitrary leader-follower formation as a quartic Bezier curve and form the initial guess, which is likely infeasible. However, this is the unconstrained optimal path in the case of exclusive pursuit, and initializing in this manner yields solutions in the region near the feasible global optimum. Finally, the QBMPC and pursuit initial guesses are weighed by ρ when $\eta = 1$ (each coordinate is then bounded by $\pm\, d_{max}$ as in QBMPC):

$$x_{k_x} = (1-\rho)x_{nav_\xi,k_x} + \rho x_{pur_\xi,k_x}, \ \ y_{k_y} = (1-\rho)y_{nav_\xi,k_y} + \rho y_{pur_\xi,k_y}, \ k_x \in \{2,3,4\}, k_y \in \{3,4\} \tag{7.4.8}$$

where only the control points that aren't fixed by initial conditions are used in the resulting initial guess, and thereafter, optimization. Therefore, depending on the adaptive pursuit weight, the initial guess balances prioritizing safe obstacle avoidance through potential fields with pursuit in formation via obstacle proximity over time.

7.4.2 Adaptive Potential Field vs. Pursuit Objective

The new objective introduces a pursuit term, F_{ρ_ξ,d^2} weighed against the original QBMPC objective's single term, F_{obj_ξ} from Equation 6.3.2:

$$F_{obj_{\rho,\xi}} = (1-\eta\rho)\lambda_\xi F_{obj_\xi} + \eta\rho\lambda_{\rho_\xi,d^2} F_{\rho_\xi,d^2} \tag{7.4.9}$$

with base weighting factors, $\lambda_{\rho,\xi} = [\lambda_\xi, \lambda_{\rho_\xi,d^2}]$ where normalization means only one weight factor is a free parameter. In the scenario where $\eta = 0$, the problem reduces to the original QBMPC algorithm, where the objective becomes F_{obj_ξ} when normalizing by the base weighting factor. However, when a leader exists & $\eta = 1$, pursuit is enacted where the adaptive pursuit weight balances the objective terms dynamically.

From the initialization step, the reference pursuit path that maintains the desired formation (x_ρ, y_ρ) with the leader is given by the control points $(x_{\rho_\xi,k}, y_{\rho_\xi,k})$, equivalent to the initial pursuit guess $(x_{pur_\xi,k}, y_{pur_\xi,k})$ from Equation 7.4.7. Now, the pursuit objective term is constructed by the sum of squared distances between the predicted trajectory and reference pursuit path for each non-fixed control point coordinate:

$$F_{\rho_\xi,d^2} = \sum_{k_x=2}^{4}(x_{k_x} - x_{\rho_\xi,k_x})^2 + \sum_{k_y=3}^{4}(y_{k_y} - y_{\rho_\xi,k_y})^2 \tag{7.4.10}$$

This objective term prioritizes paths that more closely resemble the trajectory that maintains formation with the leader by shaping the governing control points accordingly. Instead of discretizing the paths and evaluating distances between trajectories, the control points are used directly here to promote path similarity. The analytical gradients for this new objective term are presented in Appendix A.5. A visualization of the quartic Bezier curves and their corresponding control points for both the optimized path and reference pursuit trajectory is shown in Figure 7.5.

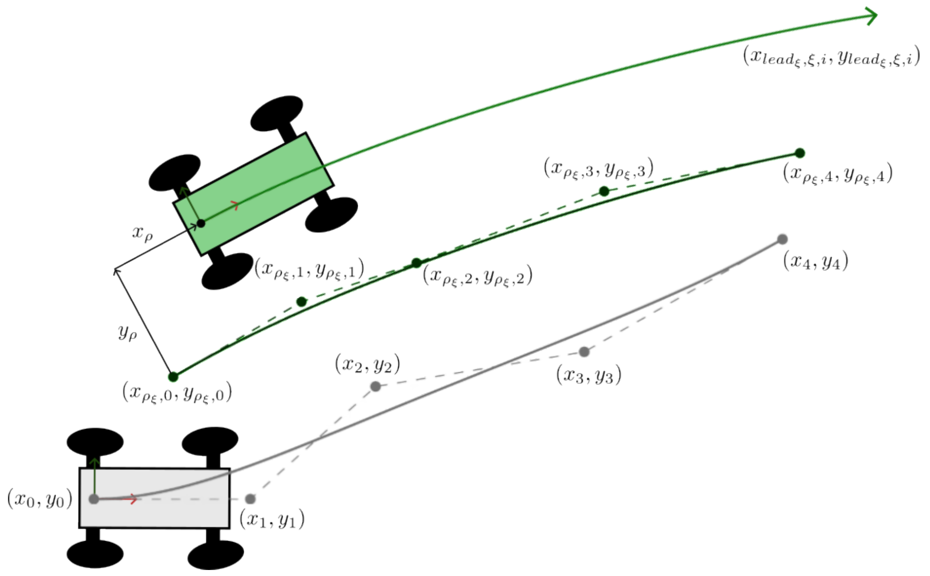

$(x_{lead_\xi,\xi,i}, y_{lead_\xi,\xi,i})$
$(x_{\rho_\xi,3}, y_{\rho_\xi,3})$
$(x_{\rho_\xi,4}, y_{\rho_\xi,4})$
x_ρ
$(x_{\rho_\xi,1}, y_{\rho_\xi,1})$
$(x_{\rho_\xi,2}, y_{\rho_\xi,2})$
(x_4, y_4)
y_ρ
(x_2, y_2)
(x_3, y_3)
$(x_{\rho_\xi,0}, y_{\rho_\xi,0})$
(x_0, y_0)
(x_1, y_1)

Figure 7.5: Leader (light green), reference pursuit (dark green) & optimized ego vehicle (gray) quartic Bezier curve paths. The predicted trajectory attains a feasible path which aligns non-fixed control points (x_{k_x}, y_{k_y}) with those governing the reference pursuit path $(x_{\rho_\xi,k_x}, y_{\rho_\xi,k_y})$ in the case of exclusive pursuit.

7.4.3 Bezier Curve Pursuit Proximity Constraint

When engaging in pursuit, the leader-follower proximity is monitored through the predicted trajectory, and a minimum is enforced as in P-STLMPC. This constraint ensures the follower remains behind the leader with adequate space to maintain safety in multi-vehicle environments. A soft minimum function is used once more, where n_ξ samples are taken of each curve ($i \in \{0, ..., n_\xi - 1\}$) over t_ξ, the prediction time horizon. For the ego vehicle Bezier curve path being optimized, the discretized points, $(x_{\xi,i}, y_{\xi,i})$ follow the form of Equation 6.3.1.

Meanwhile, the constant curvature & velocity model is used for the leader's trajectory $(x_{lead,i}, y_{lead,i})$, eliminating any minor inaccuracies that may arise from using the quartic Bezier curve model. The leader's path is sampled evenly in $\hat{s}_{lead,i}$, where now $\hat{s}_{lead,i} = \frac{i}{n_\xi - 1} t_\xi v_{lead}$ with redefined i (instead of $i\, v_{lead}\, \Delta t$ from P-STLMPC) is used in

Equations 7.3.2 & 7.3.3. Proceeding with the new sampling definition for i, Equations 7.3.4 & 7.3.5 yield the leader's orientation at each sample, $\theta_{lead,i}$ and Equation 7.3.19 expresses the leader's discretized position, $(x_{lead,i}, y_{lead,i})$ in the follower's frame.

Just as in P-STLMPC, Equation 7.3.20 uses the leader's position & orientation to obtain the four corners of the vehicle's shape, $\hat{C}_{lead,i}$ for more practical inter-vehicle distance measurements. Now, the distance between the discretized ego vehicle trajectory and the leader's n^{th} corner at sample i becomes:

$$d_{lead_{\xi},i,n} = \sqrt{(x_{\xi,i} - \hat{c}_{lead_x,i,n})^2 + (y_{\xi,i} - \hat{c}_{lead_y,i,n})^2} \quad \forall i \in \{0, ..., n_{\xi} - 1\},\ n \in \{0, ..., 3\} \tag{7.4.11}$$

The soft minimum distance parallels Equation 7.3.22 where it is now represented by:

$$d_{lead_{min,\xi}} = -\frac{1}{\beta_{\rho}} \log(\sum_{i=0}^{n_{\xi}-1} \sum_{n=0}^{3} e^{-\beta_{\rho} d_{lead_{\xi},i,n}}) \tag{7.4.12}$$

The ensuing single constraint ensures inter-vehicle distances exceed the minimum allowed value, $d_{lead_{safe}}$, over the full prediction duration and is given through:

$$\begin{gathered} d_{lead_{safe}} \le d_{lead_{min,\xi}} \quad \text{when} \quad \eta = 1 \\ \text{if} \quad \eta = 1, \quad g_{d_{lead,\xi}} = \frac{1}{\beta_{\rho}} \log(\sum_{i=0}^{n_{\xi}-1} \sum_{n=0}^{3} e^{-\beta_{\rho} d_{lead_{\xi},i,n}}) + d_{lead_{safe}}, \quad g_{d_{lead,\xi}} \le 0 \end{gathered} \tag{7.4.13}$$

This inequality is only enforced when a leader is detected and has analytical gradients as provided in Appendix A.5. By evenly sampling over the leader & follower trajectories in time, the proximity between the vehicles is equally considered throughout all stages of the future time horizon. This constraint is enforced regardless of how much priority pursuit is given via ρ such that multi-vehicle navigation remains safe and collision-free both in and out of formation.

7.5 P-STLMPC & P-QBMPC Algorithm Pseudocodes

The P-STLMPC & P-QBMPC frameworks, therefore, both provide flexible pursuit in formation with a minimum permissible following distance while also satisfying safe local path planning as before. The complete formulation of the P-STLMPC optimization problem is denoted by:

$$\begin{aligned} \min_{x_j,y_j,\theta_j,\delta_j,v_j} \quad & (1-\eta\rho)\lambda_{d^2}F_{d^2} + (1-\eta\rho)\lambda_{\dot{d}^2}F_{\dot{d}^2} + \eta(1-\rho)\lambda_{v^2}F_{v_\rho^2} + \lambda_{\delta^2}F_{\delta^2} \\ & + (1-\eta)\lambda_{v^2}F_{\Delta v^2} + \eta\rho\lambda_{\rho,d^2}F_{\rho,d^2} + \eta\rho\lambda_{\rho,\dot{d}^2}F_{\rho,\dot{d}^2} \\ \text{subject to} \quad & g_{\delta^+,i} \le 0, \quad g_{\delta^-,i} \le 0, \quad g_{v^+,i} \le 0, \quad g_{v^-,i} \le 0 \\ & g_{v_{steer},j} \le 0, \quad g_{v_{obs},j} \le 0, \quad \text{if} \ \ \eta = 1, \ \ g_{d_{lead}} \le 0 \\ & h_{x,i} = 0, \quad h_{y,i} = 0, \quad h_{\theta,i} = 0 \\ & -\delta_{max} \le \delta_j \le \delta_{max}, \quad v_{min} \le v_j \le v_{max} \\ & x_0 = 0, \quad y_0 = 0, \quad \theta_0 = 0, \quad \delta_0 = \delta_{last}, \quad v_0 = v_{last} \\ & \forall i \in \{0, ..., n_{MPC}k_{MPC} - 2\}, \quad \forall j \in \{0, ..., n_{MPC}k_{MPC} - 1\} \end{aligned} \tag{7.5.1}$$

Alternatively, the P-QBMPC optimization problem is expressed as:

$$\begin{aligned} \min_{x_2,x_3,y_3,x_4,y_4} \quad & (1-\eta\rho)\lambda_\xi F_{obj_\xi} + \eta\rho\lambda_{\rho_\xi,d^2}F_{\rho_\xi,d^2} \\ \text{subject to} \quad & g_{v_\xi^+,i} \le 0, \quad g_{v_\xi^-,i} \le 0, \quad g_{a_\xi,i} \le 0, \quad g_{d_\xi,i} \le 0 \\ & g_{\kappa_\xi^+,i} \le 0, \quad g_{\kappa_\xi^-,i} \le 0, \quad g_{\kappa'^+_\xi,i} \le 0, \quad g_{\kappa'^-_\xi,i} \le 0 \\ & \text{if} \ \ \eta = 1, \ \ g_{d_{lead,\xi}} \le 0 \\ & -d_{\max} \le x_2, x_3, y_3, x_4, y_4 \le d_{\max} \\ & x_0 = 0, \quad y_0 = 0, \quad x_1 = \frac{v_{last}t_\xi}{4}, \quad y_1 = 0, \quad y_2 = \frac{4x_1^2\tan(\delta_{last})}{3l} \\ & \forall i \in \{0, ..., n_\xi - 1\} \end{aligned} \tag{7.5.2}$$

After successful optimization, P-STLMPC & P-QBMPC derive the control inputs applied at each step, δ_{cmd} & v_{cmd} from the solution in the same manner as in STLMPC & QBMPC, respectively. The shared procedure for these pursuit algorithms at each control step is provided in Algorithm 5, which enables safe local path planning in multi-vehicle cooperative contexts.

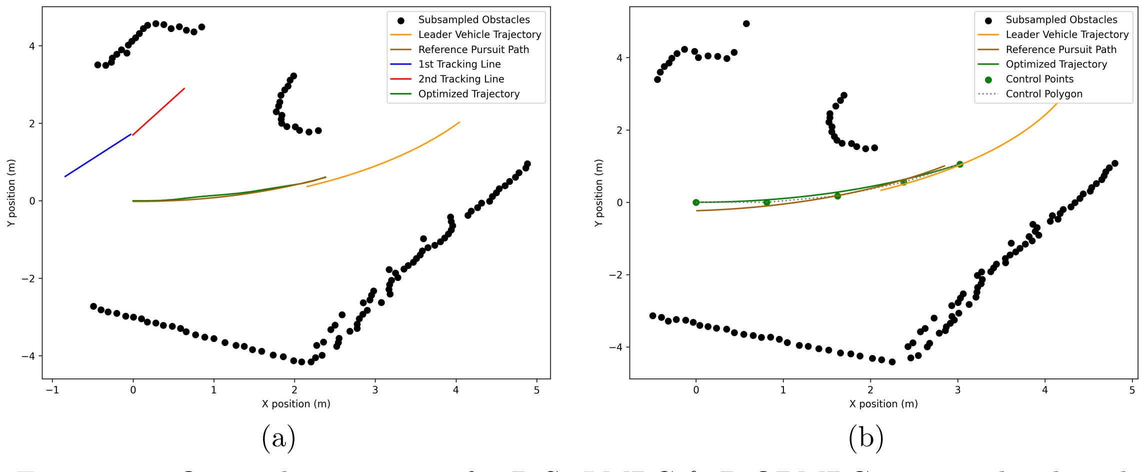
Algorithm 5: Pursuit Model Predictive Control Framework

```
   Input: Detected & Localized Obstacles ($\mathcal{O}_{obs}, \mathcal{O}_{obs_{map}}$), Last Control Inputs
          ($\delta_{last}, v_{last}$), Lagging Pursuit Position ($x_\rho, y_\rho$), Tracked Vehicles
          ($\mathcal{V}_{track}$), Adaptive Pursuit Weighting ($\rho$), Bezier Curve Order ($n = 4$)
   Output: New Control Inputs ($\delta_{cmd}, v_{cmd}$), Updated Pursuit Weighting ($\rho$)
1  if $N_{track} = 1$ and IsPursuableAndSafe($\nu_{track,0}$) then
2      $\eta = 1$;
3      $\nu_{lead} = \nu_{track,0}$;
4      $\mathcal{O}_{obs} \leftarrow$ RemoveObsNearLeader($\mathcal{O}_{obs}, \nu_{lead}$);
5      $\mathcal{O}_{obs_{tot}} = \mathcal{O}_{obs} \cup \mathcal{O}_{obs_{map}}$;
6  else
7      $\eta = 0$;
8      $\mathcal{O}_{obs_{det}} \leftarrow$ VehicleAvoidanceObs($\mathcal{V}_{track}$);
9      $\mathcal{O}_{obs_{tot}} = \mathcal{O}_{obs} \cup \mathcal{O}_{obs_{map}} \cup \mathcal{O}_{obs_{det}}$;
10 $\mathcal{O}_{obs_{sub}} \leftarrow$ ObsSubsampling($\mathcal{O}_{obs_{tot}}, d_{sep}$);
11 if Using_P-STLMPC then
12     $\tilde{w}_i \ \forall i \in \{0, ..., n_{MPC} k_{MPC} - 1\} \leftarrow$ GenerateTrackingLines($v_{last}, \mathcal{O}_{obs_{tot}}$);
13     $x_i, y_i, \theta_i, \delta_i, v_i \leftarrow$ InitialGuess_P-STLMPC($\tilde{w}_i, \delta_{last}, v_{last}, x_\rho, y_\rho, \eta, \nu_{lead}, \rho$);
14     $x_i, y_i, \theta_i, \delta_i, v_i, \delta_{cmd}, v_{cmd} \leftarrow$
         P-STLMPC_Opt($x_i, y_i, \theta_i, \delta_i, v_i, \tilde{w}_i, \delta_{last}, v_{last}, x_\rho, y_\rho, \eta, \nu_{lead}, \rho, \mathcal{O}_{obs_{sub}}$);
15     $d_{obs_{min}} \leftarrow$ MinimumObsProximity($x_i, y_i, \mathcal{O}_{obs_{sub}}$);
16 else if Using_P-QBMPC then
17     $\theta_{head,j} \ \forall j \in \{0, n-3\} \leftarrow$ GenerateHeadingAngles($v_{last}, t_\xi, \mathcal{O}_{obs_{tot}}$);
18     $(x_k, y_k) \ \forall k \in \{0, ..., n\} \leftarrow$
         InitialGuess_P-QBMPC($\theta_{head,j}, \delta_{last}, v_{last}, x_\rho, y_\rho, \eta, \nu_{lead}, \rho, t_\xi$);
19     $x_2, x_3, y_3, x_4, y_4, \delta_{cmd}, v_{cmd} \leftarrow$
         P-QBMPC_Opt($x_k, y_k, x_\rho, y_\rho, \eta, \nu_{lead}, \rho, t_\xi, \mathcal{O}_{obs_{sub}}$);
20     $d_{obs_{min}} \leftarrow$ MinimumObsProximity($x_k, y_k, \mathcal{O}_{obs_{sub}}$);
21 $\rho \leftarrow$ UpdatePursuitWeight($d_{obs_{min}}, \rho$);
22 Apply control inputs, $\delta_{cmd}$ & $v_{cmd}$ then proceed to next control step;
```

7.6 Simulation Results

The P-STLMPC & P-QBMPC algorithms are now evaluated and contrasted in a simple leader-follower scheme in simulation using a new map, Map #5. In this test, a leader vehicle with a constant velocity of 1.7 m/s traverses the environment and is pursued by the ego vehicle using each of the pursuit algorithms. Leader vehicle detection is again simulated without the use of a camera, while each algorithm utilizes parameter values from the tests in prior chapters. The shared flexible pursuit parameters used include d_{nav}=0.8 m, d_ρ=1.5 m, s_ρ=0.05, $d_{lead_{safe}}$=0.7 m & β_ρ=80 m^{-1}. Specifically for P-STLMPC, the base weighting factors are $\lambda_{d^2} = 1, \lambda_{\dot{d}^2} = 30, \lambda_{v^2} = 0.2,$ $\lambda_{\delta^2} = 0.2, \lambda_{\rho,d^2} = 20$ & $\lambda_{\rho,\dot{d}^2} = 2$, while for P-QBMPC, $\lambda_\xi = 1$ & $\lambda_{\rho_\xi,d^2} = 0.1$.

Selecting the following formation as $(x_\rho, y_\rho) = (2.1, -0.6)$, the optimized trajectory is shown at a single time sample for P-STLMPC (Figure 7.6a) & P-QBMPC (Figure 7.6b). Here, the map has two possible routes, split by an obstacle in the middle. P-STLMPC's tracking line method chooses the first open route for navigation; however, the leader vehicle progresses along the second, further route.



(a) (b)

Figure 7.6: Optimal trajectories for P-STLMPC & P-QBMPC at a single, shared time sample. Each algorithm balances its original objective with following the pursuit reference path, lagging the leader in formation.

In the P-STLMPC test, the pursuit weight at the given sample time is 1.00, while for P-QBMPC, it is 0.88 due to high local obstacle clearances, leading each method to strongly prioritize pursuit and follow the leader through the second route.

For the same pursuit formation, the two pursuit trajectories over the full simulation are provided in Figure 7.7. Each approach maintains a central path, prioritizing safety initially before tracking the pursuit reference in formation when the map opens up at the first turn. For the rest of the test, the pursuit weight remains high, and so each algorithm preserves formation behind the leader accurately until the leader reaches a dead end, where navigation terminates. Visibly, P-STLMPC has more path fluctuations to maintain formation, while P-QBMPC is smoother and has somewhat poorer pursuit tracking but with lower control effort expended.

These results are additionally shown in Table 7.1 for both pursuit algorithms, where each achieves successful adaptive pursuit with certain advantages.

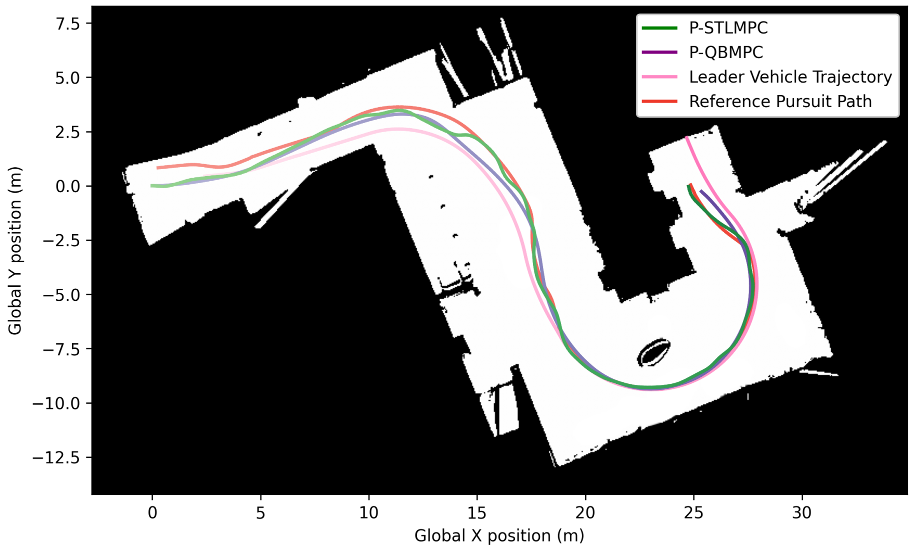


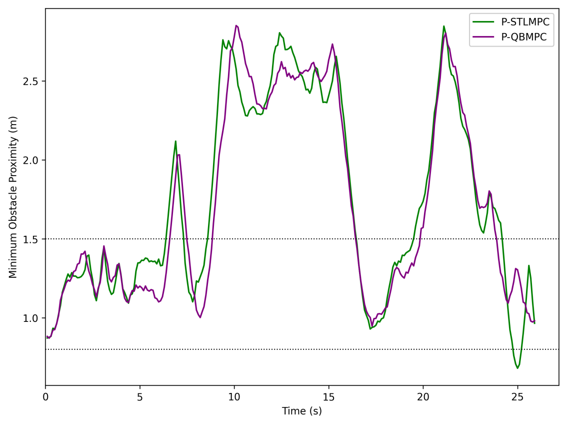
Figure 7.7: P-STLMPC & P-QBMPC for a given leader trajectory and pursuit formation in simulation using Map #5. Darker color gradients for each path show time progression where the P-STLMPC, P-QBMPC and pursuit reference paths all lag behind the leader throughout the simulation.

Table 7.1: Performance of P-STLMPC & P-QBMPC in simulated Map #5

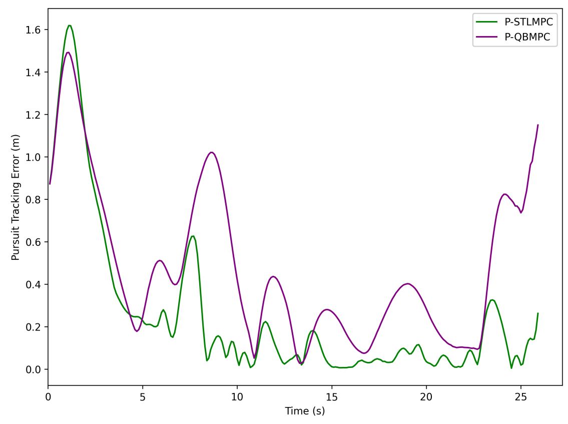
Local Planner	$\min_{\tilde{k}} d_{min,\tilde{k}}$ (m)	$\bar{d}_{min}$ (m)	$\bar{d}_{\rho,0,\tilde{k}}$ (m)	$\overline{\lvert\delta_{cmd,\tilde{k}}\rvert}$ (rad)	$\mathrm{Var}(\delta_{cmd,\tilde{k}})$ (rad^2)	$\bar{v}_{cmd,\tilde{k}}$ (m/s)	$\mathrm{Var}(v_{cmd,\tilde{k}})$ (m^2/s^2)
P-STLMPC	0.680	1.741	0.256	0.124	0.025	1.902	0.167
P-QBMPC	0.871	1.718	0.476	0.049	0.003	1.793	0.149

In addition to metrics used in prior simulations, $\bar{d}_{\rho,0,\tilde{k}}$ represents the average distance error between the ego vehicle and the initial point of the corresponding sample's pursuit reference path, measuring pursuit accuracy. P-STLMPC achieves better pursuit at the cost of inferior minimum obstacle proximity compared to P-QBMPC. Lower control effort is used by P-QBMPC, shown by the average steering angle and variance of both control inputs, while P-STLMPC achieves a higher average speed. P-STLMPC makes more abrupt motions to satisfy control objectives and thus covers more distance in the same simulation timeframe compared to smoother P-QBMPC.

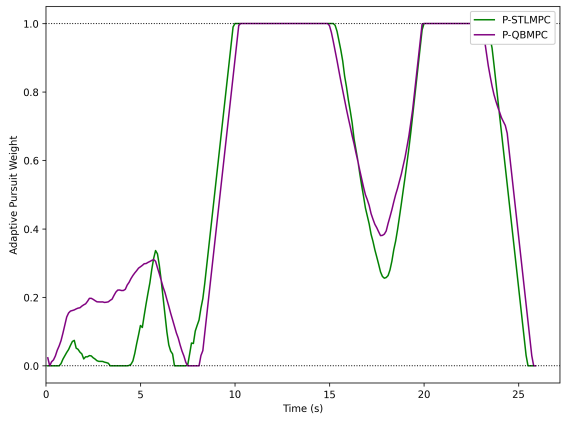
Over this simulation, the minimum obstacle proximity is illustrated over time (Figure 7.8a) as well as conversely, the pursuit tracking error, $\bar{d}_{\rho,0,\tilde{k}}$ (Figure 7.8b). After the initial narrow corridor, the obstacle clearances rise beyond d_ρ, leading the pursuit weight (Figure 7.8c) to increase, prioritizing pursuit. Sudden, momentary drops in obstacle clearance decrease the pursuit weight, such as when the vehicle nears corners or the centrally placed obstacle in Map #5. Also noticeable is how pursuit accuracy is highest when obstacle clearances increase, as safe maneuvers are possible with more open space. At the map's dead end, the ego vehicle slows to a stop, and the pursuit weight becomes zero as obstacles prevent any further navigation. Thus, the adaptive pursuit algorithms are both able to conserve high obstacle clearances while accurately following the leader vehicle through the simulated environment.



(a)



(b)



(c)

Figure 7.8: Over the full simulation using Map #5, P-STLMPC & P-QBMPC balance prioritization of safe navigation with pursuit. This dynamic weighting is controlled by (a) the minimum obstacle proximity over time, where the lower (d_{nav}) and upper (d_{ρ}) bounds affect the pursuit weight update. Furthermore, (b) the error between the vehicle and reference pursuit paths over time indicates pursuit accuracy, while (c) the time-varying pursuit weight governs behavior in the flexible formation.

7.7 Experimental Results

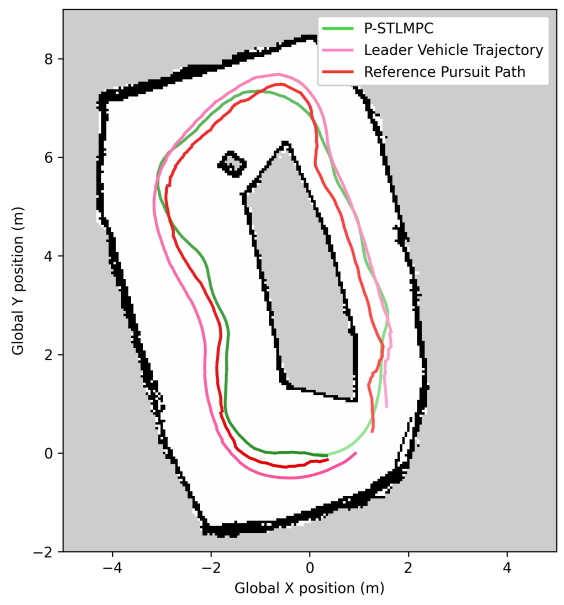
Cooperative multi-vehicle navigation is assessed in real-world conditions for both P-STLMPC & P-QBMPC using the track layout in Experiment #6 (Appendix B).

As in simulation, a simple leader-follower scheme is tested—now with two formation configurations. For each configuration, performance is evaluated in the case when both vehicles operate using P-STLMPC as well as P-QBMPC (as the leader detects no other vehicles, its planning reduces to standard STLMPC & QBMPC, respectively). Parameters from prior experiments are retained while now, $d_{safe} = 2$ m, $d_{nav} = 0.4$ m, $d_\rho = 0.9$ m & $d_{lead_{safe}} = 0.4$ m to accommodate the track layout.

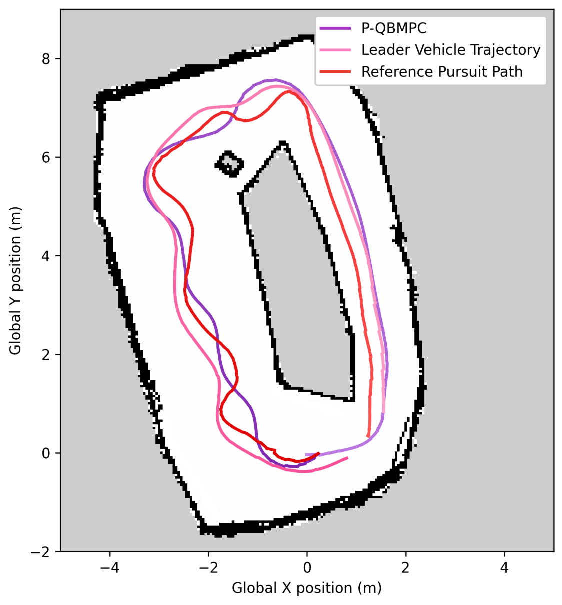
The formations tested are $(x_\rho, y_\rho) = (0.5, -0.3)$ where the pursuing vehicle is behind and to the left of the leader (Experiment #6-1) and $(x_\rho, y_\rho) = (0.5, 0.3)$ where the follower trails on the right (Experiment #6-2). Tighter formations are used due to decreased detection & tracking accuracy at higher ranges. For P-STLMPC, $v_{min} = 0$ m/s while $v_{max} = 1.5$ m/s for the follower and 1 m/s for the leader to ensure the follower can gain on the leader if it falls behind. For P-QBMPC, t_ξ =1.5 s & $v_{min} = 0.5$ m/s while again, $v_{max} = 1.5$ m/s for the follower and 1 m/s for the leader.

The trajectories obtained by P-STLMPC & P-QBMPC in Experiment #6-1 are visualized in Figures 7.9a & 7.9b respectively, while P-STLMPC specifically is recorded and documented by video[1]. This track is designed to encourage safe navigation in the narrow, right portion while promoting pursuit in formation as the track opens up on the left. Both planners attain paths that closely align directly behind the leader in the narrow section before branching to the leader's left in the wider area. Fluctuations in the leader's P-QBMPC path are amplified in the pursuit reference path, although the following vehicle successfully maintains travel to the leader's left. Robust detection and tracking with a limited camera FOV becomes challenging when both vehicles are in motion (especially at higher speeds), although these results illustrate that this flexible formation framework can function in real-world conditions.

[1]`https://www.youtube.com/watch?v=49ws64lPL-c`



(a)



(b)

Figure 7.9: Flexible formation navigation using the trailing leftward formation of Experiment #6-1. Here, leader & follower navigate using (a) P-STLMPC & (b) P-QBMPC. Darker color gradients show time progression where the P-STLMPC, P-QBMPC and pursuit reference paths all lag behind the leader throughout.

Performance of both adaptive pursuit methods is shown numerically in Tables 7.2 & 7.3. P-QBMPC achieves superior performance over P-STLMPC based on obstacle proximity metrics, while planning times are consistently faster. However, P-STLMPC exhibits lower control effort in this experiment (partially as the leader vehicle's STLMPC path is straighter, with less emphasis than QBMPC on maintaining high obstacle clearance). P-STLMPC also has a higher average speed and achieves superior pursuit tracking. Both P-STLMPC & P-QBMPC achieve better pursuit tracking when ρ is larger, as it is further prioritized, while larger pursuit weights also correspond to a higher average obstacle clearance. The minimum following distance over the full experiment, using the four corners of the leader's shape ($\min_{n,\tilde{k}} d_{lead,0,n,\tilde{k}}$) satisfies the smallest allowed distance, $d_{lead_{safe}} = 0.4$ m for both algorithms.

Table 7.2: Control input and planning time performance for P-STLMPC & P-QBMPC in Experiment #6-1

Local Planner	$\min_{\tilde{k}} d_{min,\tilde{k}}$ (m)	$\overline{\lvert\delta_{cmd,\tilde{k}}\rvert}$ (rad)	$\mathrm{Var}(\delta_{cmd,\tilde{k}})$ (rad^2)	$\bar{v}_{cmd,\tilde{k}}$ (m/s)	$\mathrm{Var}(v_{cmd,\tilde{k}})$ (m^2/s^2)	$\bar{t}_{comp}$ (ms)	$\max t_{comp}$ (ms)
P-STLMPC	0.449	0.192	0.043	0.973	0.047	34.9	64.6
P-QBMPC	0.552	0.227	0.065	0.808	0.098	1.4	5.9

Table 7.3: Obstacle proximity and pursuit accuracy performance for P-STLMPC & P-QBMPC in Experiment #6-1

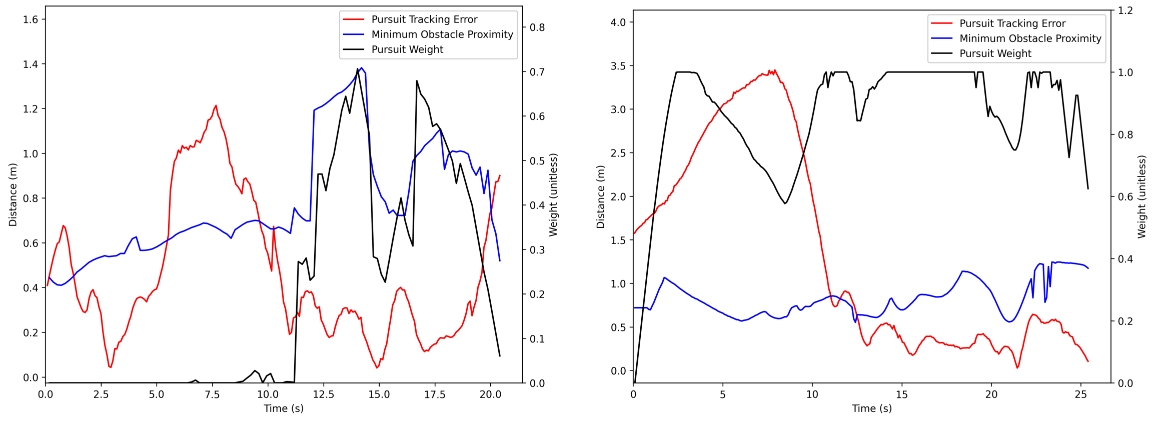
			$\boldsymbol{\rho < 0.5}$		$\boldsymbol{\rho \geq 0.5}$		
Local Planner	$\bar{d}_{min}$ (m)	$\bar{d}_{\rho,0,\tilde{k}}$ (m)	$\bar{d}_{min}$ (m)	$\bar{d}_{\rho,0,\tilde{k}}$ (m)	$\bar{d}_{min}$ (m)	$\bar{d}_{\rho,0,\tilde{k}}$ (m)	$\min_{n,\tilde{k}} d_{lead,0,n,\tilde{k}}$ (m)
P-STLMPC	0.824	0.461	0.752	0.510	1.184	0.201	0.420
P-QBMPC	0.828	1.338	0.716	1.680	0.832	1.322	0.497

Moreover, the minimum obstacle proximity, pursuit tracking error and pursuit weight are evaluated over time for both P-STLMPC (Figure 7.10a) & P-QBMPC (Figure 7.10b) in Experiment #6-1. In both cases, larger obstacle clearance in the second half of the course increases pursuit weight which in turn decreases pursuit tracking error. The interplay between these metrics is evident where each is balanced during navigation dynamically. For P-QBMPC, the leader vehicle initially travels faster than the follower, exiting pursuit range. However, as the follower catches up, it achieves a low pursuit error just as in P-STLMPC (this occurs in both experiments, explaining the deceptively high pursuit errors for P-QBMPC in Tables 7.3 & 7.5).

In the second adaptive pursuit experiment (Experiment #6-2), the follower now trails the leader on the right, where trajectories are shown in Figure 7.11 and QBMPC in particular is recorded by video[2]. Again, each trailing vehicle stays in the center

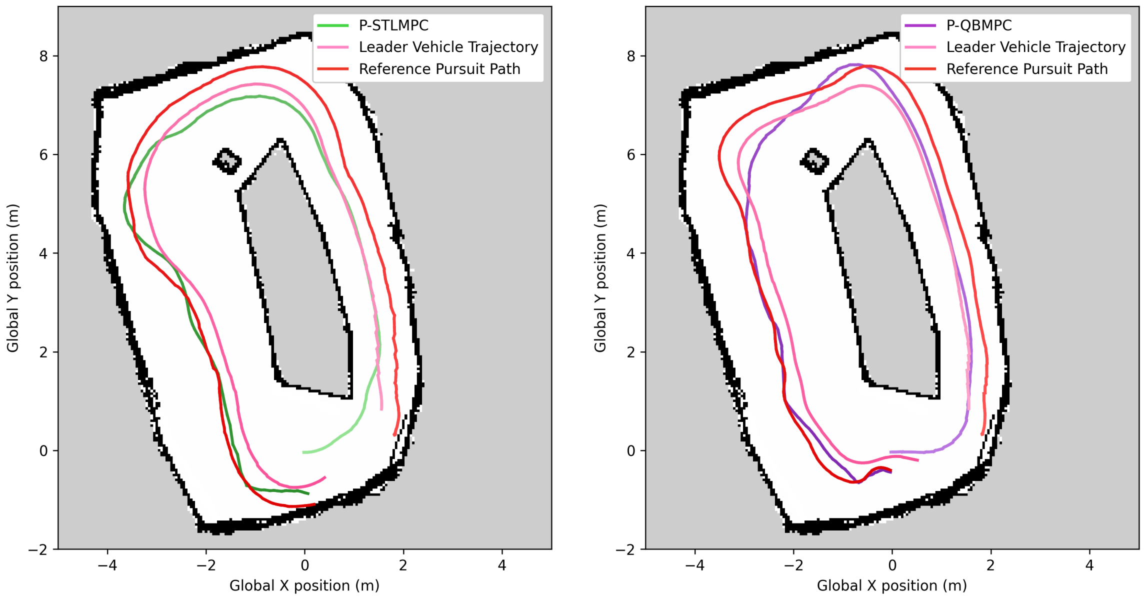
[2]`https://www.youtube.com/watch?v=zwTHNDGbHSE`

of the track during the narrow section before pursuing the leader on the right in the wider region. Thus, flexible formation is achieved as the environment varies.



(a) (b)

Figure 7.10: Obstacle proximity, pursuit tracking error and pursuit weight over time in Experiment #6-1 for (a) P-STLMPC and (b) P-QBMPC.



(a) (b)

Figure 7.11: Flexible formation navigation using the trailing rightward formation of Experiment #6-2. Here, leader & follower navigate using (a) P-STLMPC & (b) P-QBMPC. Darker color gradients show time progression where the P-STLMPC, P-QBMPC and pursuit reference paths all lag behind the leader throughout.

Results from Experiment #6-2 are further shown in Tables 7.4 & 7.5 which confirm trends observed in Experiment #6-1. Again, P-QBMPC maintains higher obstacle clearance while P-STLMPC maintains pursuit formation more accurately. As in Experiment #6-1, P-QBMPC attains low pursuit error once it catches up to the leader after quickly falling behind initially, which is not immediately clear from the numerical results. Nonetheless, the pursuit error is shown to decrease for both planners when the pursuit weight exceeds 0.5 as obstacle clearance increases and more open space exists for maneuvers. Once more, P-QBMPC demonstrates a significant reduction in planning time over P-STLMPC, and both planners exceed the safe following distance.

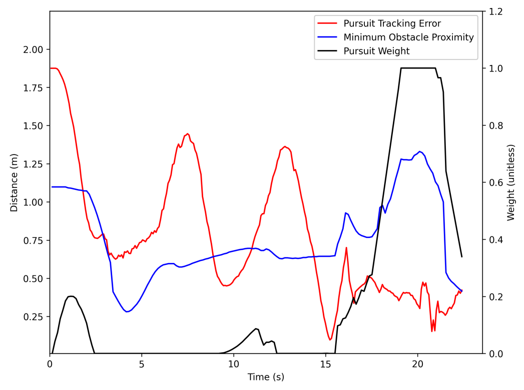
Table 7.4: Control input and planning time performance for P-STLMPC & P-QBMPC in Experiment #6-2

Local Planner	$\min_{\tilde{k}} d_{min,\tilde{k}}$ (m)	$\overline{\lvert\delta_{cmd,\tilde{k}}\rvert}$ (rad)	$\mathrm{Var}(\delta_{cmd,\tilde{k}})$ (rad^2)	$\bar{v}_{cmd,\tilde{k}}$ (m/s)	$\mathrm{Var}(v_{cmd,\tilde{k}})$ (m^2/s^2)	$\bar{t}_{comp}$ (ms)	$\max t_{comp}$ (ms)
P-STLMPC	0.346	0.162	0.035	0.934	0.091	35.8	60.4
P-QBMPC	0.534	0.188	0.057	0.816	0.117	1.4	4.6

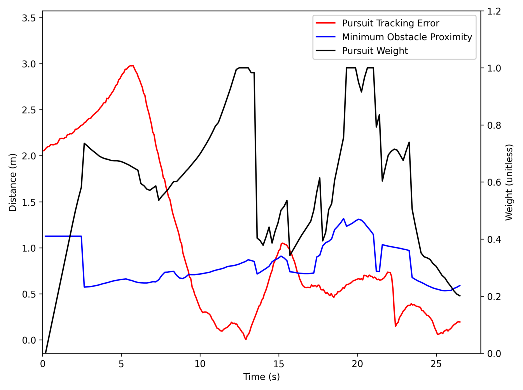
Table 7.5: Obstacle proximity and pursuit accuracy performance for P-STLMPC & P-QBMPC in Experiment #6-2

			$\boldsymbol{\rho < 0.5}$		$\boldsymbol{\rho \geq 0.5}$		
Local Planner	$\bar{d}_{min}$ (m)	$\bar{d}_{\rho,0,\tilde{k}}$ (m)	$\bar{d}_{min}$ (m)	$\bar{d}_{\rho,0,\tilde{k}}$ (m)	$\bar{d}_{min}$ (m)	$\bar{d}_{\rho,0,\tilde{k}}$ (m)	$\min_{n,\tilde{k}} d_{lead,0,n,\tilde{k}}$ (m)
P-STLMPC	0.757	0.766	0.685	0.849	1.103	0.357	0.492
P-QBMPC	0.904	1.122	0.812	1.185	0.925	1.096	0.559

The proximity, pursuit error and pursuit weight time-varying values in Experiment #6-2 are displayed for P-STLMPC & P-QBMPC (Figures 7.12a & 7.12b respectively). Both approaches attain higher pursuit weights based on higher obstacle clearance towards the end of the course. Pursuit errors for each method decrease consistently below 0.5 m during the wide region of the course while each algorithm still maintains a safe path. This performance aligns with that of Experiment #6-1, indicating that that this formulation of adaptive pursuit works in practice for arbitrary formations.



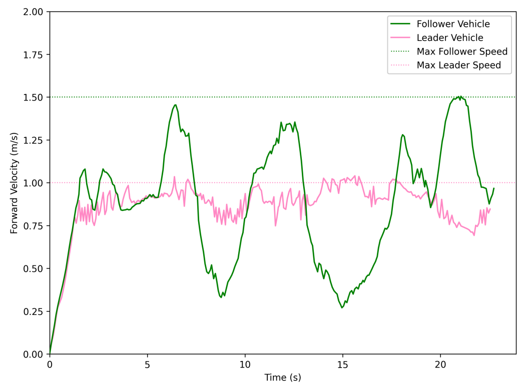
(a)



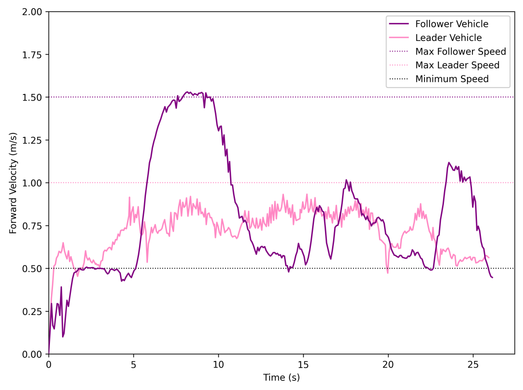
(b)

Figure 7.12: Obstacle proximity, pursuit tracking error and pursuit weight over time in Experiment #6-2 for (a) P-STLMPC and (b) P-QBMPC.

Lastly, the velocities for each leader-follower pair are stipulated in Figure 7.13. The leader maintains a more constant speed while the follower's speed profile fluctuates to maintain formation. Here, it is evident that in P-QBMPC, the follower falls far behind initially and needs to speed up rapidly to achieve accurate pursuit once more.



(a)



(b)

Figure 7.13: Leader & follower speed profiles in Experiment #6-2 for (a) P-STLMPC and (b) P-QBMPC. The velocity bounds are indicated for this experiment, where P-QBMPC requires a nonzero lower bound to ensure constant forward motion.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

This thesis presented the development of novel local path planners that establish safe real-time navigation in unknown environments where a known global goal position is not given. These navigation approaches are applicable to mobile robotics & autonomous vehicles in contexts including search and rescue, racing & multi-vehicle path planning within fleet formation. An MPC framework is implemented where optimizing over a receding prediction horizon provides the optimal local path, which is updated dynamically, considering environmental conditions and vehicle dynamics. These proposed methods extend to any arbitrary environment, characterized by local obstacles, that is traversed by an agent subject to nonholonomic system constraints.

After categorizing & reviewing existing literature pertinent to the topics discussed in this thesis (Chapter 2), the initial formulation of the first novel path planning algorithm was presented in Chapter 3. Here, successive tracking lines are constructed based on local obstacles, where heading angles correspond to the direction of the largest open space, and quadratic optimizations ensure distances to local obstacles are maximized. A non-convex optimization is conducted via SLSQP to determine the optimal path, which follows the tracking line reference path while exhibiting low control effort, all while abiding by the kinematic bicycle model (with constant velocity) and limits on control inputs. Moreover, localization is used against a known map to enhance the set of local obstacles and increase the effective look-ahead range.

Dynamic obstacles such as adversarial detected vehicles were then incorporated into local path planning through Chapter 4. This approach uses a CNN, specifically YOLO, to identify vehicles via bounding boxes, given an RGB image data stream. Multiple detections are handled by properly associating each with its own tracked vehicle, where sensor fusion via a depth camera then provides estimated poses. An EKF is used to track vehicles using transformation, prediction & correction steps, where the adversarial vehicle outlines are mapped over the predicted horizon and added to the obstacle set for local path planning.

An extended & more complete formulation of the original navigation algorithm was presented in Chapter 5, which removes the assumption of constant velocity. Specifically for racing applications, this framework prioritizes higher speeds while maintaining safe paths and satisfying control input limits. Additional optimization constraints focus on reducing speed in tight turns and when obstacles lie immediately ahead, posing significant collision risks.

Alternative to previous tracking line-based approaches, a new formulation uses a fourth-order Bezier curve for local path planning & control via a single non-convex optimization (Chapter 6). Successive heading angles in the direction of the safest local gap are used in the initial guess for the curve's control points. Using initial conditions and a potential field function that penalizes closer distances to local obstacles, a smooth trajectory is generated in low computation time. Vehicle dynamics are incorporated directly into the curve shape, a minimum allowed obstacle proximity over the predicted curve is set & the curve is generalized to an arbitrary prediction time horizon.

The tracking line & Bezier curve methods are each used in a novel cooperative multi-vehicle scheme based on adaptive pursuit in Chapter 7. A time-evolving weight is used, which encourages pursuit when the minimum obstacle distance to the vehicle trajectory is high and prioritizes safe navigation via the respective initial path planning formulation when the minimum obstacle clearance is low. Standard local navigation occurs when no leader exists; however, the presence of a pursuable leader ensures adaptive pursuit is conducted. An arbitrary leader-follower formation can be flexibly maintained while preserving safety by considering nearby obstacles and a minimum following distance. This decentralized, modular framework is extendable to arbitrary fleet formations with more vehicles, where each agent can join or abandon formation, and no fixed leader is required.

8.2 Future Work

The work conducted in this thesis has various opportunities for future extensions and adaptations. Further work can expand the framework presented to fulfill local navigation while satisfying additional criteria or adapt the framework to accommodate alternative, unique path planning applications. Specifically, future work can:

- Formulate the presented algorithms, which assume the kinematic bicycle model for different robot/vehicle models, to extend the framework to a wider set of agents. These potential models include the differential drive model (nonholonomic) and the omnidirectional drive model (holonomic).

- Adaptively modify certain parameters like look-ahead range, prediction time horizon, and base weighting factors for certain objective terms used in optimization. This would allow further adaptive navigation to best suit varying environmental conditions such as local obstacle proximities, amount of local collision-free space & number of detected vehicles. Reinforcement learning could be applied to learn from past navigation experience and, based on observed features, dynamically tune parameters to achieve improved performance.
- Progressively build the global map as more of the unknown environment is explored (in the case where no prior map is known). Global path planning using this map can then gradually be incorporated with existing local planning. In an exploration application, remaining unexplored regions can be identified and targeted for navigation in a hierarchical approach.
- Further exploit camera abilities for more holistic object detection and environment classification beyond the capabilities of LiDAR. Additional application-dependent criteria can be set, such as navigating towards certain static/dynamic obstacles, modifying path planning depending on the surroundings and incorporating parallel yellow lines as lanes in on-road autonomous driving contexts.
- Extend vehicle detection to accommodate the full angular range around the ego vehicle, not just the front FOV. This can ensure more perceptive vehicle avoidance as well as navigation in modular flexible fleet formations, where not just the immediate leader vehicle is considered.

- Incorporate state estimation error covariances from the EKF into vehicle avoidance such that higher covariances correspond to a wider band of occupied space over the detected vehicle's future trajectory. The approach in this thesis assumes the estimate is accurate and is alone considered for the future trajectory; however, this extension yields a more conservative estimate of collision-risk space.

- Accommodate negative velocities for vehicle reversal functionality into the path planning algorithms. This incorporates more flexible driving behavior for handling scenarios like dynamic obstacle avoidance and ensures vehicle navigation does not terminate at a dead end.

- Augment the Bezier curve path planning formulation to prioritize higher velocities for use in racing applications (similar to how the tracking line method was modified in Chapter 5). Additionally, navigation using Bezier curves of different orders can be tested and contrasted with the quartic Bezier curve used in this thesis.

- Extend the modular pursuit path planning framework provided in Chapter 7 to more complicated fleet formations with higher numbers of vehicles. By using this algorithm as a fundamental basis, arbitrary & decentralized multi-vehicle formations in unknown environments can be explored and tested. The case of assembling formation with multiple detected leaders would have to be addressed.

- Accommodate cooperative local navigation for a team of heterogeneous agents. Here, each robot/vehicle would perform specific tasks in the collaborative team and would be detected, identified & modeled uniquely while also following its own distinctive, independent navigation framework.

Appendix A

Optimization Gradients

This appendix provides the analytically derived gradients for each MPC optimization problem. These results are useful in obtaining accurate gradients during optimization using a non-linear, SQP-based solver like SLSQP. For the purposes of this appendix and to encourage brevity, only nonzero optimization gradients are explicitly shown.

A.1 STLMPC

For F_{d^2} (Equation 3.1.13),

$$\frac{\partial F_{d^2}}{\partial x_j} = \sum_{i=0}^{n_{MPC}k_{MPC}-1} \frac{\frac{\partial}{\partial x_j}(\tilde{w}_i^T P_i + 1)^2}{\|\tilde{w}_i\|^2} = \frac{2\tilde{w}_{x,j}(\tilde{w}_j^T P_j + 1)}{\|\tilde{w}_j\|^2} \quad \forall j \in \{0, ..., n_{MPC}k_{MPC} - 1\} \tag{A.1.1}$$

$$\frac{\partial F_{d^2}}{\partial y_j} = \sum_{i=0}^{n_{MPC}k_{MPC}-1} \frac{\frac{\partial}{\partial y_j}(\tilde{w}_i^T P_i + 1)^2}{\|\tilde{w}_i\|^2} = \frac{2\tilde{w}_{y,j}(\tilde{w}_j^T P_j + 1)}{\|\tilde{w}_j\|^2} \quad \forall j \in \{0, ..., n_{MPC}k_{MPC} - 1\} \tag{A.1.2}$$

For $F_{\dot{d}^2}$ (Equation 3.1.16),

$$\frac{\partial F_{\dot{d}^2}}{\partial x_j} = \sum_{i=0}^{n_{MPC}k_{MPC}-2} \frac{\frac{\partial}{\partial x_j}(\tilde{w}_i^T \dot{P}_i)^2}{\|\tilde{w}_i\|^2} = \frac{\frac{\partial}{\partial x_j}(\tilde{w}_{j-1}^T \dot{P}_{j-1})^2}{\|\tilde{w}_{j-1}\|^2} + \frac{\frac{\partial}{\partial x_j}(\tilde{w}_j^T \dot{P}_j)^2}{\|\tilde{w}_j\|^2}$$

$$\frac{\partial F_{\dot{d}^2}}{\partial x_j} = \begin{cases} -\frac{2\tilde{w}_{x,j}(\tilde{w}_j^T \dot{P}_j)}{\|\tilde{w}_j\|^2} & \text{if} \quad j = 0 \\ \frac{2\tilde{w}_{x,j-1}(\tilde{w}_{j-1}^T \dot{P}_{j-1})}{\|\tilde{w}_{j-1}\|^2} - \frac{2\tilde{w}_{x,j}(\tilde{w}_j^T \dot{P}_j)}{\|\tilde{w}_j\|^2} & \text{if} \quad \forall j \in \{1, ..., n_{MPC}k_{MPC} - 2\} \\ \frac{2\tilde{w}_{x,j-1}(\tilde{w}_{j-1}^T \dot{P}_{j-1})}{\|\tilde{w}_{j-1}\|^2} & \text{if} \quad j = n_{MPC}k_{MPC} - 1 \end{cases} \tag{A.1.3}$$

Similarly,

$$\frac{\partial F_{\dot{d}^2}}{\partial y_j} = \begin{cases} -\frac{2\tilde{w}_{y,j}(\tilde{w}_j^T \dot{P}_j)}{\|\tilde{w}_j\|^2} & \text{if} \quad j = 0 \\ \frac{2\tilde{w}_{y,j-1}(\tilde{w}_{j-1}^T \dot{P}_{j-1})}{\|\tilde{w}_{j-1}\|^2} - \frac{2\tilde{w}_{y,j}(\tilde{w}_j^T \dot{P}_j)}{\|\tilde{w}_j\|^2} & \text{if} \quad \forall j \in \{1, ..., n_{MPC}k_{MPC} - 2\} \\ \frac{2\tilde{w}_{y,j-1}(\tilde{w}_{j-1}^T \dot{P}_{j-1})}{\|\tilde{w}_{j-1}\|^2} & \text{if} \quad j = n_{MPC}k_{MPC} - 1 \end{cases} \tag{A.1.4}$$

For F_{δ^2} (Equation 3.1.17),

$$\frac{\partial F_{\delta^2}}{\partial \delta_j} = \sum_{i=0}^{n_{MPC}k_{MPC}-1} \frac{\partial}{\partial \delta_j}\delta_i^2 = 2\delta_j \quad \forall j \in \{0, ..., n_{MPC}k_{MPC} - 1\} \tag{A.1.5}$$

For $h_{x,j}$ (Equation 3.1.20) and $h_{y,j}$ (Equation 3.1.21),

$$\frac{\partial h_{x,j}}{\partial x_j} = -1, \quad \frac{\partial h_{x,j}}{\partial x_{j+1}} = 1, \quad \frac{\partial h_{x,j}}{\partial \theta_j} = \Delta t\, v \sin(\theta_j) \quad \forall j \in \{0, ..., n_{MPC}k_{MPC} - 2\} \tag{A.1.6}$$

$$\frac{\partial h_{y,j}}{\partial y_j} = -1, \quad \frac{\partial h_{y,j}}{\partial y_{j+1}} = 1, \quad \frac{\partial h_{y,j}}{\partial \theta_j} = -\Delta t\, v \cos(\theta_j) \quad \forall j \in \{0, ..., n_{MPC}k_{MPC} - 2\} \tag{A.1.7}$$

For $h_{\theta,j}$ (Equation 3.1.23),

$$\frac{\partial h_{\theta,j}}{\partial \theta_j} = -1, \quad \frac{\partial h_{\theta,j}}{\partial \theta_{j+1}} = 1, \quad \frac{\partial h_{\theta,j}}{\partial \delta_j} = -\Delta t \, \frac{v}{l\cos^2(\delta_j)} \quad \forall j \in \{0, ..., n_{MPC}k_{MPC} - 2\} \tag{A.1.8}$$

For $g_{\delta^+,j}$ (Equation 3.1.25) and $g_{\delta^-,j}$ (Equation 3.1.26),

$$\frac{\partial g_{\delta^+,j}}{\partial \delta_j} = -1, \quad \frac{\partial g_{\delta^+,j}}{\partial \delta_{j+1}} = 1, \quad \frac{\partial g_{\delta^-,j}}{\partial \delta_j} = 1, \quad \frac{\partial g_{\delta^-,j}}{\partial \delta_{j+1}} = -1 \quad \forall j \in \{0, ..., n_{MPC}k_{MPC} - 2\} \tag{A.1.9}$$

A.2 Non-Uniform Velocity STLMPC

All existing gradients (Equations A.1.1-A.1.9) are unchanged, except for replacing v with v_i where used. Now for F_{v^2}, $h_{x,j}$, $h_{y,j}$ and $h_{\theta,j}$ (Equations 5.2.1, 3.1.20, 3.1.21 and 3.1.23 respectively),

$$\frac{\partial F_{v^2}}{\partial v_j} = \sum_{i=0}^{n_{MPC}k_{MPC}-1} \frac{\partial}{\partial v_j}\frac{1}{v_i^2} = -\frac{2}{v_j^3} \quad \forall j \in \{0, ..., n_{MPC}k_{MPC} - 1\} \tag{A.2.1}$$

$$\begin{aligned} \frac{\partial h_{x,j}}{\partial v_j} &= -\Delta t \, \cos(\theta_j), \quad \frac{\partial h_{y,j}}{\partial v_j} = -\Delta t \, \sin(\theta_j) \\ \frac{\partial h_{\theta,j}}{\partial v_j} &= -\frac{\Delta t}{l}\tan(\delta_j) \quad \forall j \in \{0, ..., n_{MPC}k_{MPC} - 2\} \end{aligned} \tag{A.2.2}$$

For $g_{v^+,j}$ (Equation 5.3.2) and $g_{v^-,j}$ (Equation 5.3.3),

$$\frac{\partial g_{v^+,j}}{\partial v_j} = -1, \quad \frac{\partial g_{v^+,j}}{\partial v_{j+1}} = 1, \quad \frac{\partial g_{v^-,j}}{\partial v_j} = 1, \quad \frac{\partial g_{v^-,j}}{\partial v_{j+1}} = -1 \quad \forall j \in \{0, ..., n_{MPC}k_{MPC} - 2\} \tag{A.2.3}$$

For $g_{v_{steer},j}$ (Equation 5.3.5),

$$\frac{\partial g_{v_{steer},j}}{\partial \delta_j} = \frac{2v_{max}\delta_{max}^2\delta_j}{(\delta_j^2+\delta_{max}^2)^2}, \quad \frac{\partial g_{v_{steer},j}}{\partial v_j} = 1 \quad \forall j \in \{0, ..., n_{MPC}k_{MPC}-1\} \tag{A.2.4}$$

For $g_{v_{obs},j}$ (Equation 5.3.13),

$$\frac{\partial g_{v_{obs},j}}{\partial x_j} = \frac{\partial g_{v_{obs},j}}{\partial d_{\text{band}_{\min},j}} \cdot \frac{\partial d_{\text{band}_{\min},j}}{\partial \tilde{s}_j} \cdot \sum_{i=0}^{N_{obs_{sub}}-1} \left(\frac{\partial \tilde{s}_{j,i}}{\partial \theta_{\text{band},j,i}} \cdot \frac{\partial \theta_{\text{band},j,i}}{\partial \theta_{\text{diff},j,i}} \cdot \frac{\partial \theta_{\text{diff},j,i}}{\partial \theta_{j,i}} \cdot \frac{\partial \theta_{j,i}}{\partial x_j} + \frac{\partial \tilde{s}_{j,i}}{\partial d_{j,i}} \cdot \frac{\partial d_{j,i}}{\partial x_j}\right) \tag{A.2.5}$$

$$\frac{\partial g_{v_{obs},j}}{\partial y_j} = \frac{\partial g_{v_{obs},j}}{\partial d_{\text{band}_{\min},j}} \cdot \frac{\partial d_{\text{band}_{\min},j}}{\partial \tilde{s}_j} \cdot \sum_{i=0}^{N_{obs_{sub}}-1} \left(\frac{\partial \tilde{s}_{j,i}}{\partial \theta_{\text{band},j,i}} \cdot \frac{\partial \theta_{\text{band},j,i}}{\partial \theta_{\text{diff},j,i}} \cdot \frac{\partial \theta_{\text{diff},j,i}}{\partial \theta_{j,i}} \cdot \frac{\partial \theta_{j,i}}{\partial y_j} + \frac{\partial \tilde{s}_{j,i}}{\partial d_{j,i}} \cdot \frac{\partial d_{j,i}}{\partial y_j}\right) \tag{A.2.6}$$

$$\frac{\partial g_{v_{obs},j}}{\partial \theta_j} = \frac{\partial g_{v_{obs},j}}{\partial d_{\text{band}_{\min},j}} \cdot \frac{\partial d_{\text{band}_{\min},j}}{\partial \tilde{s}_j} \cdot \sum_{i=0}^{N_{obs_{sub}}-1} \left(\frac{\partial \tilde{s}_{j,i}}{\partial \theta_{\text{band},j,i}} \cdot \frac{\partial \theta_{\text{band},j,i}}{\partial \theta_{\text{diff},j,i}} \cdot \frac{\partial \theta_{\text{diff},j,i}}{\partial \theta_j}\right) \tag{A.2.7}$$

$$\frac{\partial g_{v_{obs},j}}{\partial v_j} = 1 \quad \forall j \in \{0, ..., n_{MPC}k_{MPC}-1\} \tag{A.2.8}$$

Here, $\tilde{s}_j = \sum_{i=0}^{N_{obs_{sub}}-1} \theta_{\text{band},j,i}\, e^{-\beta_v d_{j,i}}$, $\tilde{s}_{j,i} = \theta_{\text{band},j,i}\, e^{-\beta_v d_{j,i}}$ (Equation 5.3.11) and the intermediate function gradients are (Equations 5.3.7 - 5.3.13):

$$\frac{\partial g_{v_{obs},j}}{\partial d_{\text{band}_{\min},j}} = -\frac{v_{max}}{\alpha_v} e^{-\frac{d_{\text{band}_{\min},j}-d_{stop}}{\alpha_v}}, \quad \frac{\partial d_{\text{band}_{\min},j}}{\partial \tilde{s}_j} = -\frac{1}{\beta_v \tilde{s}_j}, \quad \frac{\partial \tilde{s}_{j,i}}{\partial \theta_{\text{band},j,i}} = e^{-\beta_v d_{j,i}} \tag{A.2.9}$$

$$\frac{\partial \theta_{\text{band},j,i}}{\partial \theta_{\text{diff},j,i}} = s_\theta\left(\frac{e^{-s_\theta(\theta_{\text{diff},j,i}+\theta_{\text{band}_{\max}})}}{(1+e^{-s_\theta(\theta_{\text{diff},j,i}+\theta_{\text{band}_{\max}})})^2} - \frac{e^{-s_\theta(\theta_{\text{diff},j,i}-\theta_{\text{band}_{\max}})}}{(1+e^{-s_\theta(\theta_{\text{diff},j,i}-\theta_{\text{band}_{\max}})})^2}\right) \tag{A.2.10}$$

$$\frac{\partial \theta_{\text{diff},j,i}}{\partial \theta_{j,i}} = 1, \quad \frac{\partial \theta_{\text{diff},j,i}}{\partial \theta_j} = -1 \tag{A.2.11}$$

$$\frac{\partial \theta_{j,i}}{\partial x_j} = \frac{y_{obs_{sub},i} - y_j}{(x_{obs_{sub},i} - x_j)^2 + (y_{obs_{sub},i} - y_j)^2}, \quad \frac{\partial \theta_{j,i}}{\partial y_j} = -\frac{x_{obs_{sub},i} - x_j}{(x_{obs_{sub},i} - x_j)^2 + (y_{obs_{sub},i} - y_j)^2} \tag{A.2.12}$$

$$\frac{\partial \tilde{s}_{j,i}}{\partial d_{j,i}} = -\beta_v \tilde{s}_{j,i}, \quad \frac{\partial d_{j,i}}{\partial x_j} = \frac{x_j - x_{obs_{sub},i}}{d_{j,i}}, \quad \frac{\partial d_{j,i}}{\partial y_j} = \frac{y_j - y_{obs_{sub},i}}{d_{j,i}} \tag{A.2.13}$$

A.3 QBMPC

For F_{obj_ξ} (Equation 6.3.2) where $i \in \{0, ..., n_\xi - 1\}$, $j \in \{0, ..., N_{obs_{sub}} - 1\}$ and each control point is respectively indexed by $k_x \in \{2, 3, 4\}$ and $k_y \in \{3, 4\}$ throughout this section,

$$\frac{\partial F_{obj_\xi}}{\partial x_{k_x}} = \sum_{i=0}^{n_\xi - 1} \frac{\partial F_{obj_{\xi,i}}}{\partial x_{\xi,i}} \cdot \frac{\partial x_{\xi,i}}{\partial x_{k_x}}, \quad \frac{\partial F_{obj_\xi}}{\partial y_{k_y}} = \sum_{i=0}^{n_\xi - 1} \frac{\partial F_{obj_{\xi,i}}}{\partial y_{\xi,i}} \cdot \frac{\partial y_{\xi,i}}{\partial y_{k_y}} \tag{A.3.1}$$

With the intermediate function gradients (Equations 6.1.2, 6.1.3, 6.3.1 - 6.3.3):

$$\frac{\partial F_{obj_{\xi,i}}}{\partial x_{\xi,i}} = \sum_{j=0}^{N_{obs_{sub}}-1} \frac{\partial F_{obj_{\xi,i,j}}}{\partial d^2_{\xi,i,j}} \cdot \frac{\partial d^2_{\xi,i,j}}{\partial x_{\xi,i}}, \quad \frac{\partial F_{obj_{\xi,i}}}{\partial y_{\xi,i}} = \sum_{j=0}^{N_{obs_{sub}}-1} \frac{\partial F_{obj_{\xi,i,j}}}{\partial d^2_{\xi,i,j}} \cdot \frac{\partial d^2_{\xi,i,j}}{\partial y_{\xi,i}} \tag{A.3.2}$$

$$\frac{\partial F_{obj_{\xi,i,j}}}{\partial d^2_{\xi,i,j}} = -\left(\frac{1}{d^4_{\xi,i,j}} + \frac{\alpha_\xi}{d^2_{\xi,i,j}}\right) e^{-\alpha_\xi d^2_{\xi,i,j}}, \quad \frac{\partial d^2_{\xi,i,j}}{\partial x_{\xi,i}} = 2(x_{\xi,i} - x_{obs_{sub},j}), \quad \frac{\partial d^2_{\xi,i,j}}{\partial y_{\xi,i}} = 2(y_{\xi,i} - y_{obs_{sub},j}) \tag{A.3.3}$$

$$\frac{\partial x_{\xi,i}}{\partial x_2} = 6(1 - t_i)^2 t_i^2, \quad \frac{\partial x_{\xi,i}}{\partial x_3} = \frac{\partial y_{\xi,i}}{\partial y_3} = 4(1 - t_i) t_i^3, \quad \frac{\partial x_{\xi,i}}{\partial x_4} = \frac{\partial y_{\xi,i}}{\partial y_4} = t_i^4 \tag{A.3.4}$$

For $g_{v_\xi^+,i}$ (Equation 6.4.2) and $g_{v_\xi^-,i}$ (Equation 6.4.3),

$$\frac{\partial g_{v_\xi^+,i}}{\partial x_{k_x}} = \frac{-\partial g_{v_\xi^-,i}}{\partial x_{k_x}} = 2x'_{\xi,i} \frac{\partial x'_{\xi,i}}{\partial x_{k_x}}, \quad \frac{\partial g_{v_\xi^+,i}}{\partial y_{k_y}} = \frac{-\partial g_{v_\xi^-,i}}{\partial y_{k_y}} = 2y'_{\xi,i} \frac{\partial y'_{\xi,i}}{\partial y_{k_y}} \tag{A.3.5}$$

Using intermediate gradients (Equations 6.1.4 and 6.1.5),

$$\frac{\partial x'_{\xi,i}}{\partial x_2} = 24t_i^3 - 36t_i^2 + 12t_i, \quad \frac{\partial x'_{\xi,i}}{\partial x_3} = \frac{\partial y'_{\xi,i}}{\partial y_3} = -16t_i^3 + 12t_i^2, \quad \frac{\partial x'_{\xi,i}}{\partial x_4} = \frac{\partial y'_{\xi,i}}{\partial y_4} = 4t_i^3 \tag{A.3.6}$$

For $g_{a_\xi,i}$ (Equation 6.4.6),

$$\frac{\partial g_{a_\xi,i}}{\partial x_{k_x}} = \frac{2(x'_{\xi,i}x''_{\xi,i}+y'_{\xi,i}y''_{\xi,i})((x''_{\xi,i}\frac{\partial x'_{\xi,i}}{\partial x_{k_x}} + x'_{\xi,i}\frac{\partial x''_{\xi,i}}{\partial x_{k_x}})(x'^2_{\xi,i} + y'^2_{\xi,i}) - x'_{\xi,i}\frac{\partial x'_{\xi,i}}{\partial x_{k_x}}(x'_{\xi,i}x''_{\xi,i}+y'_{\xi,i}y''_{\xi,i}))}{(x'^2_{\xi,i} + y'^2_{\xi,i})^2} \tag{A.3.7}$$

$$\frac{\partial g_{a_\xi,i}}{\partial y_{k_y}} = \frac{2(x'_{\xi,i}x''_{\xi,i}+y'_{\xi,i}y''_{\xi,i})((y''_{\xi,i}\frac{\partial y'_{\xi,i}}{\partial y_{k_y}} + y'_{\xi,i}\frac{\partial y''_{\xi,i}}{\partial y_{k_y}})(x'^2_{\xi,i} + y'^2_{\xi,i}) - y'_{\xi,i}\frac{\partial y'_{\xi,i}}{\partial y_{k_y}}(x'_{\xi,i}x''_{\xi,i}+y'_{\xi,i}y''_{\xi,i}))}{(x'^2_{\xi,i} + y'^2_{\xi,i})^2} \tag{A.3.8}$$

With the additional intermediate gradients (Equations 6.1.7 and 6.1.8),

$$\frac{\partial x''_{\xi,i}}{\partial x_2} = 72t_i^2 - 72t_i + 12, \quad \frac{\partial x''_{\xi,i}}{\partial x_3} = \frac{\partial y''_{\xi,i}}{\partial y_3} = -48t_i^2 + 24t_i, \quad \frac{\partial x''_{\xi,i}}{\partial x_4} = \frac{\partial y''_{\xi,i}}{\partial y_4} = 12t_i^2 \tag{A.3.9}$$

For $g_{\kappa_\xi^+,i}$ (Equation 6.4.7) and $g_{\kappa_\xi^-,i}$ (Equation 6.4.8),

$$\frac{\partial g_{\kappa_\xi^+,i}}{\partial x_{k_x}} = \frac{-\partial g_{\kappa_\xi^-,i}}{\partial x_{k_x}} = \frac{(y''_{\xi,i}\frac{\partial x'_{\xi,i}}{\partial x_{k_x}} - y'_{\xi,i}\frac{\partial x''_{\xi,i}}{\partial x_{k_x}})(x'^2_{\xi,i} + y'^2_{\xi,i}) - 3x'_{\xi,i}\frac{\partial x'_{\xi,i}}{\partial x_{k_x}}(x'_{\xi,i}y''_{\xi,i} - y'_{\xi,i}x''_{\xi,i})}{(x'^2_{\xi,i} + y'^2_{\xi,i})^{5/2}} \tag{A.3.10}$$

$$\frac{\partial g_{\kappa_\xi^+,i}}{\partial y_{k_y}} = \frac{-\partial g_{\kappa_\xi^-,i}}{\partial y_{k_y}} = \frac{(x'_{\xi,i}\frac{\partial y''_{\xi,i}}{\partial y_{k_y}} - x''_{\xi,i}\frac{\partial y'_{\xi,i}}{\partial y_{k_y}})(x'^2_{\xi,i} + y'^2_{\xi,i}) - 3y'_{\xi,i}\frac{\partial y'_{\xi,i}}{\partial y_{k_y}}(x'_{\xi,i}y''_{\xi,i} - y'_{\xi,i}x''_{\xi,i})}{(x'^2_{\xi,i} + y'^2_{\xi,i})^{5/2}} \tag{A.3.11}$$

For $g_{\kappa'^{+}_{\xi},i}$ (Equation 6.4.13) and $g_{\kappa'^{-}_{\xi},i}$ (Equation 6.4.14),

$$\frac{\partial g_{\kappa'^{+}_{\xi},i}}{\partial x_{k_x}} = \frac{-\partial g_{\kappa'^{-}_{\xi},i}}{\partial x_{k_x}} = \frac{l}{1+(l\kappa_{\xi,i})^2}\frac{\partial \kappa'_{\xi,i}}{\partial x_{k_x}} - \frac{2l^3\kappa_{\xi,i}\kappa'_{\xi,i}}{(1+(l\kappa_{\xi,i})^2)^2}\frac{\partial \kappa_{\xi,i}}{\partial x_{k_x}} \tag{A.3.12}$$

$$\frac{\partial g_{\kappa'^{+}_{\xi},i}}{\partial y_{k_y}} = \frac{-\partial g_{\kappa'^{-}_{\xi},i}}{\partial y_{k_y}} = \frac{l}{1+(l\kappa_{\xi,i})^2}\frac{\partial \kappa'_{\xi,i}}{\partial y_{k_y}} - \frac{2l^3\kappa_{\xi,i}\kappa'_{\xi,i}}{(1+(l\kappa_{\xi,i})^2)^2}\frac{\partial \kappa_{\xi,i}}{\partial y_{k_y}} \tag{A.3.13}$$

Here, $\frac{\partial \kappa_{\xi,i}}{\partial x_{k_x}}$ and $\frac{\partial \kappa_{\xi,i}}{\partial y_{k_y}}$ are equivalent to Equations A.3.10 and A.3.11, respectively, and further intermediate gradients (Equations 6.4.10 - 6.4.12) are provided. Using the numerator, $\kappa'_{num,\xi,i} = (x'_{\xi,i}y'''_{\xi,i} - y'_{\xi,i}x'''_{\xi,i})(x'^2_{\xi,i}+y'^2_{\xi,i}) - 3(x'_{\xi,i}x''_{\xi,i}+y'_{\xi,i}y''_{\xi,i})(x'_{\xi,i}y''_{\xi,i}-y'_{\xi,i}x''_{\xi,i})$ of Equation 6.4.10 with the gradients $\frac{\partial \kappa'_{num,\xi,i}}{\partial x_{k_x}}$ and $\frac{\partial \kappa'_{num,\xi,i}}{\partial y_{k_y}}$ (for the quotient rule),

$$\frac{\partial \kappa'_{\xi,i}}{\partial x_{k_x}} = \frac{\frac{\partial \kappa'_{num,\xi,i}}{\partial x_{k_x}}}{(x'^2_{\xi,i}+y'^2_{\xi,i})^{5/2}} - \frac{5x'_{\xi,i}\frac{\partial x'_{\xi,i}}{\partial x_{k_x}}\kappa'_{num,\xi,i}}{(x'^2_{\xi,i}+y'^2_{\xi,i})^{7/2}}, \quad \frac{\partial \kappa'_{\xi,i}}{\partial y_{k_y}} = \frac{\frac{\partial \kappa'_{num,\xi,i}}{\partial y_{k_y}}}{(x'^2_{\xi,i}+y'^2_{\xi,i})^{5/2}} - \frac{5y'_{\xi,i}\frac{\partial y'_{\xi,i}}{\partial y_{k_y}}\kappa'_{num,\xi,i}}{(x'^2_{\xi,i}+y'^2_{\xi,i})^{7/2}} \tag{A.3.14}$$

$$\begin{aligned}\frac{\partial \kappa'_{num,\xi,i}}{\partial x_{k_x}} =& (y'''_{\xi,i}\frac{\partial x'_{\xi,i}}{\partial x_{k_x}} - y'_{\xi,i}\frac{\partial x'''_{\xi,i}}{\partial x_{k_x}})(x'^2_{\xi,i}+y'^2_{\xi,i}) + 2x'_{\xi,i}\frac{\partial x'_{\xi,i}}{\partial x_{k_x}}(x'_{\xi,i}y'''_{\xi,i} - y'_{\xi,i}x'''_{\xi,i})\\ &-3((x'_{\xi,i}\frac{\partial x''_{\xi,i}}{\partial x_{k_x}}+x''_{\xi,i}\frac{\partial x'_{\xi,i}}{\partial x_{k_x}})(x'_{\xi,i}y''_{\xi,i}-y'_{\xi,i}x''_{\xi,i})+(y''_{\xi,i}\frac{\partial x'_{\xi,i}}{\partial x_{k_x}}-y'_{\xi,i}\frac{\partial x''_{\xi,i}}{\partial x_{k_x}})(x'_{\xi,i}x''_{\xi,i}+y'_{\xi,i}y''_{\xi,i}))\end{aligned} \tag{A.3.15}$$

$$\begin{aligned}\frac{\partial \kappa'_{num,\xi,i}}{\partial y_{k_y}} =& (x'_{\xi,i}\frac{\partial y'''_{\xi,i}}{\partial y_{k_y}} - x'''_{\xi,i}\frac{\partial y'_{\xi,i}}{\partial y_{k_y}})(x'^2_{\xi,i}+y'^2_{\xi,i}) + 2y'_{\xi,i}\frac{\partial y'_{\xi,i}}{\partial y_{k_y}}(x'_{\xi,i}y'''_{\xi,i} - y'_{\xi,i}x'''_{\xi,i})\\ &-3((y'_{\xi,i}\frac{\partial y''_{\xi,i}}{\partial y_{k_y}}+y''_{\xi,i}\frac{\partial y'_{\xi,i}}{\partial y_{k_y}})(x'_{\xi,i}y''_{\xi,i}-y'_{\xi,i}x''_{\xi,i})+(x'_{\xi,i}\frac{\partial y''_{\xi,i}}{\partial y_{k_y}}-x''_{\xi,i}\frac{\partial y'_{\xi,i}}{\partial y_{k_y}})(x'_{\xi,i}x''_{\xi,i}+y'_{\xi,i}y''_{\xi,i}))\end{aligned} \tag{A.3.16}$$

$$\frac{\partial x'''_{\xi,i}}{\partial x_2} = 144t_i - 72, \quad \frac{\partial x'''_{\xi,i}}{\partial x_3} = \frac{\partial y'''_{\xi,i}}{\partial y_3} = -96t_i + 24, \quad \frac{\partial x'''_{\xi,i}}{\partial x_4} = \frac{\partial y'''_{\xi,i}}{\partial y_4} = 24t_i \tag{A.3.17}$$

For $g_{d_{\xi},i}$ (Equation 6.4.16),

$$\frac{\partial g_{d_{\xi},i}}{\partial x_{k_x}} = \frac{\sum_{j=0}^{N_{obs_{sub}}-1}(\frac{\partial}{\partial d_{\xi,i,j}}(e^{-\beta_{\xi} d_{\xi,i,j}}) \cdot \frac{\partial d_{\xi,i,j}}{\partial x_{k_x}})}{\beta_{\xi}\sum_{j=0}^{N_{obs_{sub}}-1} e^{-\beta_{\xi} d_{\xi,i,j}}} = -\frac{\sum_{j=0}^{N_{obs_{sub}}-1} \frac{(x_{\xi,i}-x_{obs_{sub},j})}{d_{\xi,i,j}} \frac{\partial x_{\xi,i}}{\partial x_{k_x}} e^{-\beta_{\xi} d_{\xi,i,j}}}{\sum_{j=0}^{N_{obs_{sub}}-1} e^{-\beta_{\xi} d_{\xi,i,j}}} \tag{A.3.18}$$

$$\frac{\partial g_{d_{\xi},i}}{\partial y_{k_y}} = \frac{\sum_{j=0}^{N_{obs_{sub}}-1}(\frac{\partial}{\partial d_{\xi,i,j}}(e^{-\beta_{\xi} d_{\xi,i,j}}) \cdot \frac{\partial d_{\xi,i,j}}{\partial y_{k_y}})}{\beta_{\xi}\sum_{j=0}^{N_{obs_{sub}}-1} e^{-\beta_{\xi} d_{\xi,i,j}}} = -\frac{\sum_{j=0}^{N_{obs_{sub}}-1} \frac{(y_{\xi,i}-y_{obs_{sub},j})}{d_{\xi,i,j}} \frac{\partial y_{\xi,i}}{\partial y_{k_y}} e^{-\beta_{\xi} d_{\xi,i,j}}}{\sum_{j=0}^{N_{obs_{sub}}-1} e^{-\beta_{\xi} d_{\xi,i,j}}} \tag{A.3.19}$$

A.4 P-STLMPC

In this section, $j \in \{0, ..., n_{MPC}k_{MPC}-1\}$ unless otherwise specified. Carrying over all applicable existing gradients from Appendix A.2, additional objective term gradients proceed with velocity terms $F_{\Delta v^2}$ and $F_{v_\rho^2}$ (Equations 7.3.12 and 7.3.13 respectively),

$$\frac{\partial F_{\Delta v^2}}{\partial v_j} = \begin{cases} -2(v_{j+1} - v_j) & \text{if} \quad j = 0 \\ -2(v_{j+1} - 2v_j + v_{j-1}) & \text{if} \quad \forall j \in \{1, ..., n_{MPC}k_{MPC} - 2\} \\ 2(v_j - v_{j-1}) & \text{if} \quad j = n_{MPC}k_{MPC} - 1 \end{cases} \tag{A.4.1}$$

$$\frac{\partial F_{v_\rho^2}}{\partial v_j} = 2(v_j - v_{lead}) \tag{A.4.2}$$

For F_{ρ,d^2} (Equation 7.3.14),

$$\frac{\partial F_{\rho,d^2}}{\partial x_j} = 2(x_j - x_{\rho,j}), \quad \frac{\partial F_{\rho,d^2}}{\partial y_j} = 2(y_j - y_{\rho,j}) \tag{A.4.3}$$

For $F_{\rho,\dot{d}^2}$ (Equation 7.3.18) with intermediate sample set $J_{int} = \{1, ..., n_{MPC}k_{MPC}-2\}$ and each sample's objective term, $F_{\rho,\dot{d}^2,j}$ where $F_{\rho,\dot{d}^2} = \sum_{j=0}^{n_{MPC}k_{MPC}-2} F_{\rho,\dot{d}^2,j}$,

$$\frac{\partial F_{\rho,\dot{d}^2}}{\partial x_j}=\begin{cases} \frac{\frac{\partial F^{num}_{\rho,\dot{d}^2,j}}{\partial x_j}F^{denom}_{\rho,\dot{d}^2,j}-\frac{\partial F^{denom}_{\rho,\dot{d}^2,j}}{\partial x_j}F^{num}_{\rho,\dot{d}^2,j}}{(F^{denom}_{\rho,\dot{d}^2,j})^2} & \text{if } j=0 \\ \frac{\frac{\partial F^{num}_{\rho,\dot{d}^2,j-1}}{\partial x_j}F^{denom}_{\rho,\dot{d}^2,j-1}-\frac{\partial F^{denom}_{\rho,\dot{d}^2,j-1}}{\partial x_j}F^{num}_{\rho,\dot{d}^2,j-1}}{(F^{denom}_{\rho,\dot{d}^2,j-1})^2}+\frac{\frac{\partial F^{num}_{\rho,\dot{d}^2,j}}{\partial x_j}F^{denom}_{\rho,\dot{d}^2,j}-\frac{\partial F^{denom}_{\rho,\dot{d}^2,j}}{\partial x_j}F^{num}_{\rho,\dot{d}^2,j}}{(F^{denom}_{\rho,\dot{d}^2,j})^2} & \text{if } j\in J_{int} \\ \frac{\frac{\partial F^{num}_{\rho,\dot{d}^2,j-1}}{\partial x_j}F^{denom}_{\rho,\dot{d}^2,j-1}-\frac{\partial F^{denom}_{\rho,\dot{d}^2,j-1}}{\partial x_j}F^{num}_{\rho,\dot{d}^2,j-1}}{(F^{denom}_{\rho,\dot{d}^2,j-1})^2} & \text{if } j=n_{MPC}k_{MPC}-1 \end{cases} \tag{A.4.4}$$

$$\frac{\partial F_{\rho,\dot{d}^2}}{\partial y_j}=\begin{cases} \frac{\frac{\partial F^{num}_{\rho,\dot{d}^2,j}}{\partial y_j}F^{denom}_{\rho,\dot{d}^2,j}-\frac{\partial F^{denom}_{\rho,\dot{d}^2,j}}{\partial y_j}F^{num}_{\rho,\dot{d}^2,j}}{(F^{denom}_{\rho,\dot{d}^2,j})^2} & \text{if } j=0 \\ \frac{\frac{\partial F^{num}_{\rho,\dot{d}^2,j-1}}{\partial y_j}F^{denom}_{\rho,\dot{d}^2,j-1}-\frac{\partial F^{denom}_{\rho,\dot{d}^2,j-1}}{\partial y_j}F^{num}_{\rho,\dot{d}^2,j-1}}{(F^{denom}_{\rho,\dot{d}^2,j-1})^2}+\frac{\frac{\partial F^{num}_{\rho,\dot{d}^2,j}}{\partial y_j}F^{denom}_{\rho,\dot{d}^2,j}-\frac{\partial F^{denom}_{\rho,\dot{d}^2,j}}{\partial y_j}F^{num}_{\rho,\dot{d}^2,j}}{(F^{denom}_{\rho,\dot{d}^2,j})^2} & \text{if } j\in J_{int} \\ \frac{\frac{\partial F^{num}_{\rho,\dot{d}^2,j-1}}{\partial y_j}F^{denom}_{\rho,\dot{d}^2,j-1}-\frac{\partial F^{denom}_{\rho,\dot{d}^2,j-1}}{\partial y_j}F^{num}_{\rho,\dot{d}^2,j-1}}{(F^{denom}_{\rho,\dot{d}^2,j-1})^2} & \text{if } j=n_{MPC}k_{MPC}-1 \end{cases} \tag{A.4.5}$$

The numerator, $F^{num}_{\rho,\dot{d}^2,j}=((x_j-x_{\rho,j})(x_{j+1}-x_j-v_{\rho_x,j})+(y_j-y_{\rho,j})(y_{j+1}-y_j-v_{\rho_y,j}))^2$ and denominator, $F^{denom}_{\rho,\dot{d}^2,j}=(x_j-x_{\rho,j})^2+(y_j-y_{\rho,j})^2$ of Equation 7.3.18 are used in the quotient rule with $\frac{\partial F^{denom}_{\rho,\dot{d}^2,j-1}}{\partial x_j}=\frac{\partial F^{denom}_{\rho,\dot{d}^2,j-1}}{\partial y_j}=0$ and the remaining nonzero gradients,

$$\frac{\partial F^{num}_{\rho,\dot{d}^2,j}}{\partial x_j}=2((x_j-x_{\rho,j})(x_{j+1}-x_j-v_{\rho_x,j})+(y_j-y_{\rho,j})(y_{j+1}-y_j-v_{\rho_y,j}))(x_{j+1}-2x_j+x_{\rho,j}-v_{p_x,j}) \tag{A.4.6}$$

$$\frac{\partial F^{num}_{\rho,\dot{d}^2,j}}{\partial y_j}=2((x_j-x_{\rho,j})(x_{j+1}-x_j-v_{\rho_x,j})+(y_j-y_{\rho,j})(y_{j+1}-y_j-v_{\rho_y,j}))(y_{j+1}-2y_j+y_{\rho,j}-v_{p_y,j}) \tag{A.4.7}$$

$$\frac{\partial F^{denom}_{\rho,\dot{d}^2,j}}{\partial x_j}=2(x_j-x_{\rho,j}),\quad \frac{\partial F^{denom}_{\rho,\dot{d}^2,j}}{\partial y_j}=2(y_j-y_{\rho,j}) \tag{A.4.8}$$

$$\frac{\partial F^{num}_{\rho,\dot{d}^2,j-1}}{\partial x_j}=2\big((x_{j-1}-x_{\rho,j-1})(x_j-x_{j-1}-v_{\rho_x,j-1})+(y_{j-1}-y_{\rho,j-1})(y_j-y_{j-1}-v_{\rho_y,j-1})\big)(x_{j-1}-x_{\rho,j-1}) \tag{A.4.9}$$

$$\frac{\partial F^{num}_{\rho,\dot{d}^2,j-1}}{\partial y_j}=2\big((x_{j-1}-x_{\rho,j-1})(x_j-x_{j-1}-v_{\rho_x,j-1})+(y_{j-1}-y_{\rho,j-1})(y_j-y_{j-1}-v_{\rho_y,j-1})\big)(y_{j-1}-y_{\rho,j-1}) \tag{A.4.10}$$

For the constraint $g_{d_{lead}}$ (Equation 7.3.23),

$$\frac{\partial g_{d_{lead}}}{\partial x_j} = -\frac{\sum_{n=0}^{3} \frac{x_j-\hat{c}_{lead_x,j,n}}{d_{lead,j,n}} e^{-\beta_\rho d_{lead,j,n}}}{\sum_{i=0}^{n_{MPC}k_{MPC}-1} \sum_{n=0}^{3} e^{-\beta_\rho d_{lead,i,n}}} \tag{A.4.11}$$

$$\frac{\partial g_{d_{lead}}}{\partial y_j} = -\frac{\sum_{n=0}^{3} \frac{y_j-\hat{c}_{lead_y,j,n}}{d_{lead,j,n}} e^{-\beta_\rho d_{lead,j,n}}}{\sum_{i=0}^{n_{MPC}k_{MPC}-1} \sum_{n=0}^{3} e^{-\beta_\rho d_{lead,i,n}}} \tag{A.4.12}$$

A.5 P-QBMPC

Maintaining all existing gradients provided in Appendix A.3 with notation following $i \in \{0, ..., n_\xi - 1\}, k_x \in \{2,3,4\}, k_y \in \{3,4\}$, the additional pursuit objective term, F_{ρ_ξ,d^2} (Equation 7.4.10) has gradients,

$$\frac{\partial F_{\rho_\xi,d^2}}{\partial x_{k_x}} = 2(x_{k_x} - x_{\rho_\xi,k_x}), \quad \frac{\partial F_{\rho_\xi,d^2}}{\partial y_{k_y}} = 2(y_{k_y} - y_{\rho_\xi,k_y}) \tag{A.5.1}$$

For $g_{d_{lead,\xi}}$ (Equation 7.4.13) with $\frac{\partial x_{\xi,i}}{\partial x_{k_x}}$ and $\frac{\partial y_{\xi,i}}{\partial y_{k_y}}$ provided in Appendix A.3,

$$\frac{\partial g_{d_{lead,\xi}}}{\partial x_{k_x}} = -\frac{\sum_{i=0}^{n_\xi-1} \sum_{n=0}^{3} \frac{x_{\xi,i}-\hat{c}_{lead_x,i,n}}{d_{lead_\xi,i,n}} \frac{\partial x_{\xi,i}}{\partial x_{k_x}} e^{-\beta_\rho d_{lead_\xi,i,n}}}{\sum_{i=0}^{n_\xi-1} \sum_{n=0}^{3} e^{-\beta_\rho d_{lead_\xi,i,n}}} \tag{A.5.2}$$

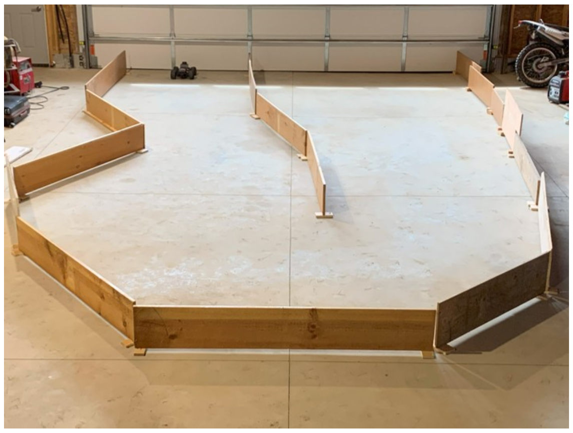
$$\frac{\partial g_{d_{lead,\xi}}}{\partial y_{k_y}} = -\frac{\sum_{i=0}^{n_\xi-1} \sum_{n=0}^{3} \frac{y_{\xi,i}-\hat{c}_{lead_y,i,n}}{d_{lead_\xi,i,n}} \frac{\partial y_{\xi,i}}{\partial y_{k_y}} e^{-\beta_\rho d_{lead_\xi,i,n}}}{\sum_{i=0}^{n_\xi-1} \sum_{n=0}^{3} e^{-\beta_\rho d_{lead_\xi,i,n}}} \tag{A.5.3}$$

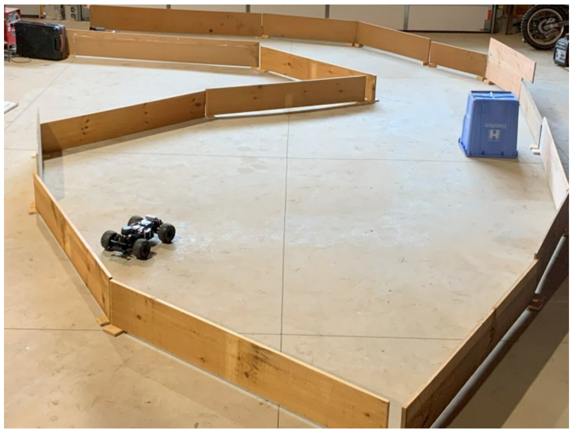
Appendix B

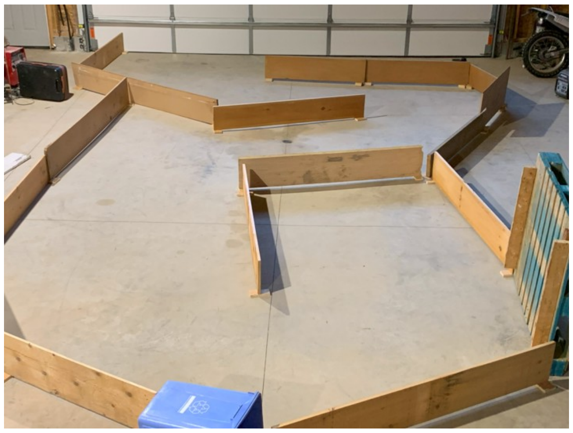
Experiment Course Layouts

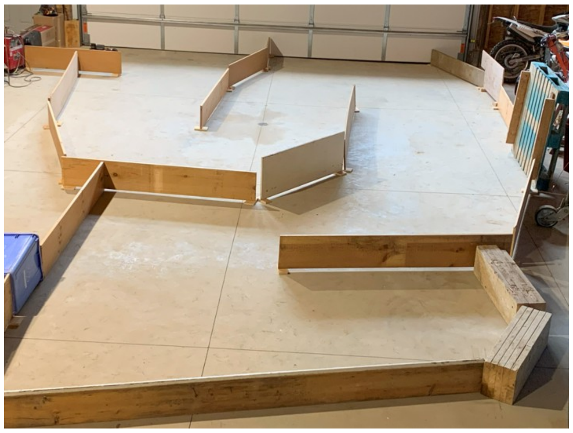
This appendix contains the layouts for all tested experiment setups (Experiments #1-6). The experiments make use of varying environment configurations to illustrate the versatility of this thesis' local planning algorithms. Experiment #1 tests STLMPC with a constant velocity (Figure B.1a) while Experiment #2 incorporates dynamic obstacle/vehicle avoidance into STLMPC (Figure B.1b). Experiment #3 evaluates non-uniform velocity STLMPC for racing (Figure B.1c) and Experiment #4 assesses QBMPC (Figure B.1d). Dynamic obstacle/vehicle avoidance is considered for QBMPC in Experiment #5 (Figure B.1e), and finally, adaptive pursuit is conducted using P-STLMPC & P-QBMPC in Experiment #6 (Figure B.1f). Navigation in each experiment setup is recorded by video and correspondingly provided for viewing each local path planner in action[1]. These experiments illustrate the functionality of the algorithms presented in this thesis when faced with real-world conditions. In these tests, standalone local path planning is achieved while maintaining safety and meeting specific objective criteria in dynamic, unknown, multi-vehicle environments.

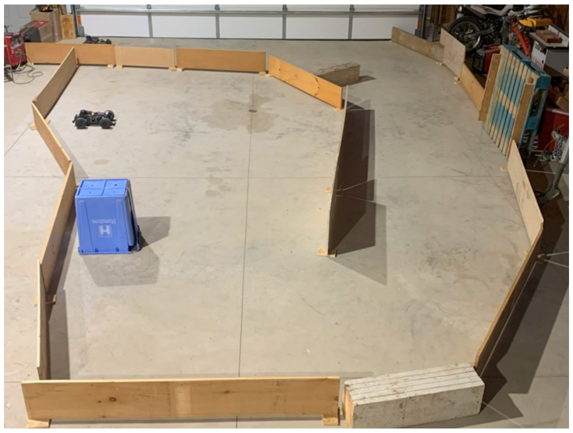
[1]`https://www.youtube.com/playlist?list=PLXepHSd7xhvUFHHfbtSod06hIosCzU3bD`


(a)


(b)


(c)


(d)


(e)

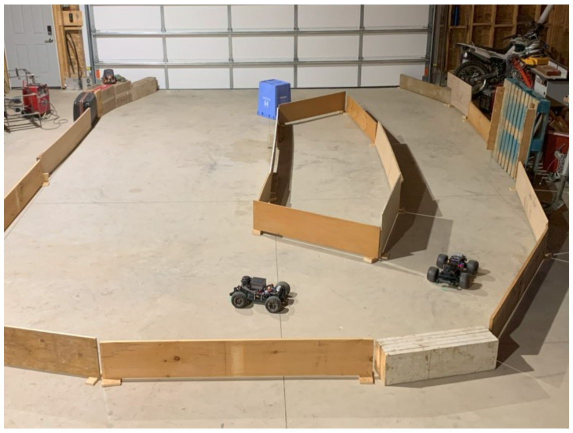

(f)

Figure B.1: Experiment course layouts for (a) Experiment #1, (b) Experiment #2, (c) Experiment #3, (d) Experiment #4, (e) Experiment #5 & (f) Experiment #6.

Bibliography

[1] S. Yu, M. Hirche, Y. Huang, H. Chen, and F. Allgöwer, “Model predictive control for autonomous ground vehicles: a review,” *Autonomous Intelligent Systems*, vol. 1, Aug. 2021.

[2] J. Ji, A. Khajepour, W. W. Melek, and Y. Huang, “Path planning and tracking for vehicle collision avoidance based on model predictive control with multi-constraints,” *IEEE Transactions on Vehicular Technology*, vol. 66, p. 952–964, Feb. 2017.

[3] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat, “Predictive active steering control for autonomous vehicle systems,” *IEEE Transactions on Control Systems Technology*, vol. 15, p. 566–580, May 2007.

[4] S. Bae, D. Isele, A. Nakhaei, P. Xu, A. M. Añon, C. Choi, K. Fujimura, and S. Moura, “Lane-change in dense traffic with model predictive control and neural networks,” *IEEE Transactions on Control Systems Technology*, vol. 31, p. 646–659, Mar. 2023.

[5] A. Khosravian, M. Masih-Tehrani, A. Amirkhani, and S. Ebrahimi-Nejad, “Robust autonomous vehicle control by leveraging multi-stage MPC and quantized

CNN in HIL framework," *Applied Soft Computing*, vol. 162, p. 111802, Sept. 2024.

[6] Z. Dong, H. Sun, G. Chen, N. Ma, Q. Wang, and Y. Yan, "Improved A * algorithm for autonomous vehicle path planning under post-disaster rescue scene," in *2024 8th International Conference on Robotics, Control and Automation (ICRCA)*, p. 206–210, IEEE, Jan. 2024.

[7] Z. Abidin, M. Muis, and W. Djuriatno, "Omni-wheeled robot with Rapidly-exploring Random Tree (RRT) algorithm for path planning," in *2019 International Conference on Advanced Mechatronics, Intelligent Manufacture and Industrial Automation (ICAMIMIA)*, p. 288–292, IEEE, Oct. 2019.

[8] Z. Sun, B. Xia, P. Xie, X. Li, and J. Wang, "NAMR-RRT: Neural adaptive motion planning for mobile robots in dynamic environments," 2024.

[9] S. Alshammrei, S. Boubaker, and L. Kolsi, "Improved Dijkstra algorithm for mobile robot path planning and obstacle avoidance," *Computers, Materials; Continua*, vol. 72, no. 3, p. 5939–5954, 2022.

[10] Y. Huang, Z. Tian, Q. Jiang, and J. Xu, "Path tracking based on improved pure pursuit model and PID," in *2020 IEEE 2nd International Conference on Civil Aviation Safety and Information Technology (ICCASIT)*, p. 359–364, IEEE, Oct. 2020.

[11] M. T. Emirler, I. M. C. Uygan, B. Aksun Güvenç, and L. Güvenç, "Robust PID steering control in parameter space for highly automated driving," *International Journal of Vehicular Technology*, vol. 2014, p. 1–8, Feb. 2014.

[12] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics; Automation Magazine*, vol. 4, p. 23–33, Mar. 1997.

[13] S. Quinlan and O. Khatib, "Elastic bands: connecting path planning and control," in *[1993] Proceedings IEEE International Conference on Robotics and Automation*, IEEE Comput. Soc. Press, 1993.

[14] C. Roesmann, W. Feiten, T. Woesch, F. Hoffmann, and T. Bertram, "Trajectory modification considering dynamic constraints of autonomous robots," in *ROBOTIK 2012; 7th German Conference on Robotics*, pp. 1–6, 2012.

[15] C. Rosmann, A. Makarow, and T. Bertram, "Online motion planning based on nonlinear model predictive control with non-euclidean rotation groups," in *2021 European Control Conference (ECC)*, p. 1583–1590, IEEE, June 2021.

[16] Z. Ercan, M. Gokasan, and F. Borrelli, "An adaptive and predictive controller design for lateral control of an autonomous vehicle," in *2017 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, p. 13–18, IEEE, June 2017.

[17] T. Weiskircher, Q. Wang, and B. Ayalew, "Predictive guidance and control framework for (semi-)autonomous vehicles in public traffic," *IEEE Transactions on Control Systems Technology*, vol. 25, p. 2034–2046, Nov. 2017.

[18] E. Alcalá, V. Puig, and J. Quevedo, "LPV-MPC control for autonomous vehicles," *IFAC-PapersOnLine*, vol. 52, no. 28, p. 106–113, 2019.

[19] W. Lim, S. Lee, M. Sunwoo, and K. Jo, "Hierarchical trajectory planning of an autonomous car based on the integration of a sampling and an optimization

method," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, p. 613–626, Feb. 2018.

[20] R. Oliveira, P. F. Lima, G. Collares Pereira, J. Martensson, and B. Wahlberg, "Path planning for autonomous bus driving in highly constrained environments," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, p. 2743–2749, IEEE, Oct. 2019.

[21] M. Kobayashi and N. Motoi, "Local path planning: Dynamic window approach with virtual manipulators considering dynamic obstacles," *IEEE Access*, vol. 10, pp. 17018–17029, 2022.

[22] J. Matouš, D. Varagnolo, K. Y. Pettersen, and C. Paliotta, "Distributed MPC for formation path-following of multi-vehicle systems," *IFAC-PapersOnLine*, vol. 55, no. 13, p. 85–90, 2022.

[23] P. Xue, D. Pi, C. Wan, C. Yang, B. Xie, H. Wang, X. Wang, and G. Yin, "Collaborative planning and control of heterogeneous multi-ground unmanned platforms," *Engineering Applications of Artificial Intelligence*, vol. 136, p. 108968, Oct. 2024.

[24] A. Ghasemi and S. Rouhi, "A safe stable directional vehicular platoon," *Proc. Inst. Mech. Eng. Pt. D: J. Automobile Eng.*, vol. 229, pp. 1083–1093, July 2015.

[25] L. Liu, X. Wang, X. Yang, H. Liu, J. Li, and P. Wang, "Path planning techniques for mobile robots: Review and prospect," *Expert Systems with Applications*, vol. 227, p. 120254, Oct. 2023.

[26] D. Chikurtev, “Mobile robot simulation and navigation in ROS and Gazebo,” in *2020 International Conference Automatics and Informatics (ICAI)*, p. 1–6, IEEE, Oct. 2020.

[27] A. Anil and V. Jisha, “Trajectory tracking control of an autonomous vehicle using model predictive control and PID controller,” in *2023 International Conference on Control, Communication and Computing (ICCC)*, p. 1–6, IEEE, May 2023.

[28] H. Efheij, A. Albagul, and N. Ammar Albraiki, “Comparison of model predictive control and PID controller in real time process control system,” in *2019 19th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*, p. 64–69, IEEE, Mar. 2019.

[29] A. Mourad and Z. Youcef, “Wheeled mobile robot path planning and path tracking in a static environment using TLBO and PID- TLBO control,” in *2022 IEEE 21st International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*, p. 116–121, IEEE, Dec. 2022.

[30] R. Wallace, A. T. Stentz, C. Thorpe, H. Moravec, W. R. L. Whittaker, and T. Kanade, “First results in robot road-following,” in *Proceedings of 9th International Joint Conference on Artificial Intelligence (IJCAI ’85)*, vol. 2, pp. 1089 – 1095, August 1985.

[31] E. Horvath, C. Hajdu, and P. Koros, “Novel pure-pursuit trajectory following approaches and their practical applications,” in *2019 10th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*, p. 597–602, IEEE, Oct. 2019.

[32] F. Cao, D. Wu, and Y. Wu, "Optimization of the pure pursuit algorithm based on real-time error," *Alexandria Engineering Journal*, vol. 124, p. 603–612, 2025.

[33] S. Macenski, S. Singh, F. Martin, and J. Gines, "Regulated pure pursuit for robot path tracking," 2023.

[34] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2, p. 500–505, Institute of Electrical and Electronics Engineers, 1985.

[35] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, IEEE Comput. Soc. Press, 1991.

[36] R. Szczepanski, "Safe artificial potential field - novel local path planning algorithm maintaining safe distance from obstacles," *IEEE Robotics and Automation Letters*, vol. 8, p. 4823–4830, Aug. 2023.

[37] R. Szczepanski and F. Gul, "Local path planning algorithm based on artificial potential field with adaptive scaling factor," in *2024 28th International Conference on Methods and Models in Automation and Robotics (MMAR)*, p. 229–234, IEEE, Aug. 2024.

[38] L. Zhou and W. Li, "Adaptive artificial potential field approach for obstacle avoidance path planning," in *2014 Seventh International Symposium on Computational Intelligence and Design*, p. 429–432, IEEE, Dec. 2014.

[39] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, p. 269–271, Dec. 1959.

[40] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, p. 100–107, 1968.

[41] S. LaValle and J. Kuffner, "Randomized kinodynamic planning," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, vol. 1 of *ROBOT-99*, p. 473–479, IEEE, 1999.

[42] E. J. Molinos, A. Llamazares, and M. Ocaña, "Dynamic window based approaches for avoiding obstacles in moving," *Robotics and Autonomous Systems*, vol. 118, p. 112–130, Aug. 2019.

[43] J. Wang, L. Yang, H. Cen, Y. He, and Y. Liu, "Dynamic obstacle avoidance control based on a novel dynamic window approach for agricultural robots," *Computers in Industry*, vol. 167, p. 104272, May 2025.

[44] M. Dobrevski and D. Skočaj, "Dynamic adaptive dynamic window approach," *IEEE Transactions on Robotics*, vol. 40, p. 3068–3081, 2024.

[45] J. Wu, X. Ma, T. Peng, and H. Wang, "An improved Timed Elastic Band (TEB) algorithm of Autonomous Ground Vehicle (AGV) in complex environment," *Sensors*, vol. 21, p. 8312, Dec. 2021.

[46] H. Yuan, H. Li, Y. Zhang, S. Du, L. Yu, and X. Wang, "Comparison and improvement of local planners on ROS for narrow passages," in *2022 International Conference on High Performance Big Data and Intelligent Systems (HDIS)*, p. 125–130, IEEE, Dec. 2022.

[47] C. Rosmann, F. Hoffmann, and T. Bertram, "Timed-Elastic-Bands for time-optimal point-to-point nonlinear model predictive control," in *2015 European Control Conference (ECC)*, p. 3352–3357, IEEE, July 2015.

[48] M. N. Zeilinger, D. M. Raimondo, A. Domahidi, M. Morari, and C. N. Jones, "On real-time robust model predictive control," *Automatica*, vol. 50, p. 683–694, Mar. 2014.

[49] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Transactions on Control Systems Technology*, vol. 18, p. 267–278, Mar. 2010.

[50] J. H. Lee, "Model predictive control: Review of the three decades of development," *International Journal of Control, Automation and Systems*, vol. 9, p. 415–424, June 2011.

[51] M. Iancu, M. V. Cristea, and P. S. Agachi, "MPC vs. PID. the advanced control solution for an industrial heat integrated fluid catalytic cracking plant," in *21st European Symposium on Computer Aided Process Engineering*, p. 517–521, Elsevier, 2011.

[52] J. Richalet, A. Rault, J. Testud, and J. Papon, "Model predictive heuristic control," *Automatica*, vol. 14, p. 413–428, Sept. 1978.

[53] C. Cutler and B. Ramaker, "Dynamic matrix control - a computer control algorithm," *The Joint Automatic Control Conference, San Francisco, USA*, 1980.

[54] M. Schwenzer, M. Ay, T. Bergs, and D. Abel, "Review on model predictive control: an engineering perspective," *The International Journal of Advanced Manufacturing Technology*, vol. 117, p. 1327–1349, Aug. 2021.

[55] J. Suh, H. Chae, and K. Yi, "Stochastic model-predictive control for lane change decision of automated driving vehicles," *IEEE Transactions on Vehicular Technology*, vol. 67, p. 4771–4782, June 2018.

[56] W. Zhang, Z. Wang, L. Drugge, and M. Nybacka, "Evaluating model predictive path following and yaw stability controllers for over-actuated autonomous electric vehicles," *IEEE Transactions on Vehicular Technology*, vol. 69, p. 12807–12821, Nov. 2020.

[57] L. Jiang, Y. Xie, Z. Jiang, J. Meng, and W. Li, "Adaptive model predictive control of mobile robot with local path refitting," in *2022 IEEE 17th Conference on Industrial Electronics and Applications (ICIEA)*, p. 596–601, IEEE, Dec. 2022.

[58] S. Ishihara, M. Kanai, R. Narikawa, and T. Ohtsuka, "A proposal of path planning for robots in warehouses by model predictive control without using global paths," *IFAC-PapersOnLine*, vol. 55, no. 37, p. 573–578, 2022.

[59] M. Babu, R. R. Theerthala, A. K. Singh, B. Baladhurgesh, B. Gopalakrishnan, K. M. Krishna, and S. Medasani, "Model predictive control for autonomous driving considering actuator dynamics," in *2019 American Control Conference (ACC)*, p. 1983–1989, IEEE, July 2019.

[60] P. Hang, C. Lv, Y. Xing, C. Huang, and Z. Hu, "Human-like decision making

for autonomous driving: A noncooperative game theoretic approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, p. 2076–2087, Apr. 2021.

[61] Z. Wang and J. Wang, "Ultra-local model predictive control: A model-free approach and its application on automated vehicle trajectory tracking," *Control Engineering Practice*, vol. 101, p. 104482, Aug. 2020.

[62] D. Limon, I. Alvarado, T. Alamo, and E. Camacho, "Robust tube-based MPC for tracking of constrained linear systems with additive disturbances," *Journal of Process Control*, vol. 20, p. 248–260, Mar. 2010.

[63] X. Wu and B. Xiao, "Safe path planning and adjustable zonotope-tube model predictive tracking control for autonomous vehicle," *Journal of the Franklin Institute*, vol. 361, p. 1332–1346, Feb. 2024.

[64] A. Fehér, A. Domina, A. Bárdos, S. Aradi, and T. Bécsi, "Path planning via reinforcement learning with closed-loop motion control and field tests," *Engineering Applications of Artificial Intelligence*, vol. 142, p. 109870, Feb. 2025.

[65] Y. Kim, D.-S. Pae, S.-H. Jang, S.-W. Kang, and M.-T. Lim, "Reinforcement learning for autonomous vehicle using MPC in highway situation," in *2022 International Conference on Electronics, Information, and Communication (ICEIC)*, p. 1–4, IEEE, Feb. 2022.

[66] B. D. Evans, H. W. Jordaan, and H. A. Engelbrecht, "High-performance racing on unmapped tracks using local maps," 2024.

[67] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1:43 scale RC cars," *Optimal Control Applications and Methods*, 2017.

[68] D. Lam, C. Manzie, and M. Good, "Model predictive contouring control," in *49th IEEE Conference on Decision and Control (CDC)*, p. 6137–6142, IEEE, Dec. 2010.

[69] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, p. 1433–1440, IEEE, May 2016.

[70] K. Honda, N. Akai, K. Suzuki, M. Aoki, H. Hosogaya, H. Okuda, and T. Suzuki, "Stein variational guided model predictive path integral control: Proposal and experiments with fast maneuvering vehicles," 2023.

[71] S. Maity, A. Bhattacharyya, P. K. Singh, M. Kumar, and R. Sarkar, "Last decade in vehicle detection and classification: A comprehensive survey," *Archives of Computational Methods in Engineering*, vol. 29, p. 5259–5296, June 2022.

[72] C. Stauffer and W. Grimson, "Adaptive background mixture models for real-time tracking," in *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, CVPR-99, p. 246–252, IEEE Comput. Soc, 1999.

[73] G. K. Yadav, T. Kancharla, and S. Nair, "Real time vehicle detection for rear

and forward collision warning systems," in *Advances in Computing and Communications*, p. 368–377, Springer Berlin Heidelberg, 2011.

[74] D. Dooley, B. McGinley, C. Hughes, L. Kilmartin, E. Jones, and M. Glavin, "A blind-zone detection method using a rear-mounted fisheye camera with combination of vehicle detection methods," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, p. 264–278, Jan. 2016.

[75] M. Maity, S. Banerjee, and S. Sinha Chaudhuri, "Faster R-CNN and YOLO based vehicle detection: A survey," in *2021 5th International Conference on Computing Methodologies and Communication (ICCMC)*, IEEE, Apr. 2021.

[76] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, p. 1137–1149, June 2017.

[77] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, real-time object detection," 2016.

[78] A. Senthil Selvi, P. Sibi Aadesh, B. Manoharan, and S. Hari Narayanan, "Real-time multiple object tracking and object detection using YOLO v7 and Fair-MOT algorithm," in *2023 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES)*, p. 1–5, IEEE, Dec. 2023.

[79] M. Gluhakovic, M. Herceg, M. Popovic, and J. Kovacevic, "Vehicle detection in the autonomous vehicle environment for potential collision warning," in *2020*

Zooming Innovation in Consumer Technologies Conference (ZINC), p. 178–183, IEEE, May 2020.

[80] X. C. Vuong, R.-F. Mou, and T. T. Vu, "Vehicle tracking and speed estimation under mixed traffic conditions using YOLOV4 and SORT: A case study of Hanoi," *Transport Problems*, vol. 17, p. 17–26, Dec. 2022.

[81] W. Ng, G. Wang, Siddhartha, Z. Lin, and B. J. Dutta, "Range-Doppler detection in automotive radar with deep learning," in *2020 International Joint Conference on Neural Networks (IJCNN)*, p. 1–8, IEEE, July 2020.

[82] B. Major, D. Fontijne, A. Ansari, R. T. Sukhavasi, R. Gowaikar, M. Hamilton, S. Lee, S. Grzechnik, and S. Subramanian, "Vehicle detection with automotive radar using deep learning on range-azimuth-Doppler tensors," in *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, IEEE, Oct. 2019.

[83] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[84] K. Rana and P. Kaur, "Simulation based vehicle movement tracking using Kalman filter algorithm for autonomous vehicles," in *2021 2nd International Conference on Intelligent Engineering and Management (ICIEM)*, p. 205–210, IEEE, Apr. 2021.

[85] M. Ghandour, H. Liu, N. Stoll, and K. Thurow, "Improving the navigation

of indoor mobile robots using Kalman filter," in *2015 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings*, p. 1434–1439, IEEE, May 2015.

[86] L. Zheng, H. Gou, K. Xiao, and M. Qiu, "Research on vehicle tracking method based on YOLOv8 and adaptive Kalman filtering: Integrating SVM dynamic selection and error feedback mechanism," *Open Journal of Applied Sciences*, vol. 14, no. 12, p. 3569–3588, 2024.

[87] Q. Ren, Q. Zhao, H. Qi, and L. Li, "Real-time target tracking system for person-following robot," in *2016 35th Chinese Control Conference (CCC)*, p. 6160–6165, IEEE, July 2016.

[88] B. Guo, N. Guo, and Z. Cen, "Obstacle avoidance with dynamic avoidance risk region for mobile robots in dynamic environments," *IEEE Robotics and Automation Letters*, vol. 7, p. 5850–5857, July 2022.

[89] S. Chen and C. Shao, "Efficient online tracking-by-detection with Kalman filter," *IEEE Access*, vol. 9, p. 147570–147578, 2021.

[90] F. Albers, C. Rösmann, F. Hoffmann, and T. Bertram, "Online trajectory optimization and navigation in dynamic environments in ROS," in *Robot Operating System (ROS)*, p. 241–274, Springer International Publishing, July 2018.

[91] A. Buyval, A. Gabdulin, R. Mustafin, and I. Shimchik, "Deriving overtaking strategy from nonlinear model predictive control for a race car," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, p. 2623–2628, IEEE, Sept. 2017.

[92] Z. Xu, X. Zhan, Y. Xiu, C. Suzuki, and K. Shimada, "Onboard dynamic-object detection and tracking for autonomous robot navigation with RGB-D camera," *IEEE Robotics and Automation Letters*, vol. 9, p. 651–658, Jan. 2024.

[93] Z. Xu, H. Shen, X. Han, H. Jin, K. Ye, and K. Shimada, "LV-DOT: LiDAR-visual dynamic obstacle detection and tracking for autonomous robot navigation," 2025.

[94] L. Izdebski, R. Kopiecki, and D. Sawicki, "Bézier curve as a generalization of the easing function in computer animation," in *Advances in Computer Graphics*, p. 382–393, Springer-Verlag, 2020.

[95] R. T. Farouki, "The Bernstein polynomial basis: A centennial retrospective," *Computer Aided Geometric Design*, vol. 29, p. 379–419, Aug. 2012.

[96] M. Reda, A. Onsy, A. Y. Haikal, and A. Ghanbari, "Path planning algorithms in the autonomous driving system: A comprehensive review," *Robotics and Autonomous Systems*, vol. 174, p. 104630, Apr. 2024.

[97] A. Sahraei, M. T. Manzuri, M. R. Razvan, M. Tajfard, and S. Khoshbakht, "Real-time trajectory generation for mobile robots," in *AI*IA 2007: Artificial Intelligence and Human-Oriented Computing*, p. 459–470, Springer Berlin Heidelberg, 2007.

[98] X. Qian, I. Navarro, A. de La Fortelle, and F. Moutarde, "Motion planning for urban autonomous driving using Bézier curves and MPC," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, p. 826–833, IEEE, Nov. 2016.

[99] V. Bulut, "Path planning for autonomous ground vehicles based on quintic trigonometric Bézier curve: Path planning based on quintic trigonometric Bézier curve," *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, vol. 43, Feb. 2021.

[100] J.-w. Choi, R. Curry, and G. Elkaim, "Path planning based on Bezier curve for autonomous ground vehicles," in *Advances in Electrical and Electronics Engineering - IAENG Special Edition of the World Congress on Engineering and Computer Science 2008*, p. 158–166, IEEE, Oct. 2008.

[101] B. Lau, C. Sprunk, and W. Burgard, "Kinodynamic motion planning for mobile robots using splines," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, p. 2427–2433, IEEE, Oct. 2009.

[102] L. An, X. Huang, P. Yang, and Z. Liu, "Adaptive Bézier curve-based path following control for autonomous driving robots," *Robotics and Autonomous Systems*, vol. 189, p. 104969, July 2025.

[103] J. Moreau, P. Melchior, S. Victor, M. Moze, F. Aioun, and F. Guillemard, "Reactive path planning for autonomous vehicle using Bézier curve optimization," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, p. 1048–1053, IEEE, June 2019.

[104] L. Zheng, P. Zeng, W. Yang, Y. Li, and Z. Zhan, "Bézier curve-based trajectory planning for autonomous vehicles with collision avoidance," *IET Intelligent Transport Systems*, vol. 14, p. 1882–1891, Dec. 2020.

[105] C. Zheng, J. Wu, Y. Bao, and C. Wei, "Bezier curve-based motion planning

for efficient robot exploration in unknown environments," in *2024 IEEE International Conference on Unmanned Systems (ICUS)*, p. 56–62, IEEE, Oct. 2024.

[106] Y. Linquan, L. Zhongwen, T. Zhonghua, and L. Weixian, "Path planning algorithm for mobile robot obstacle avoidance adopting Bezier curve based on genetic algorithm," in *2008 Chinese Control and Decision Conference*, p. 3286–3289, IEEE, July 2008.

[107] M. Elhoseny, A. Tharwat, and A. E. Hassanien, "Bezier curve based path planning in a dynamic field using modified genetic algorithm," *Journal of Computational Science*, vol. 25, p. 339–350, Mar. 2018.

[108] S. Knorn, A. Donaire, J. C. Agüero, and R. H. Middleton, "Passivity-based control for multi-vehicle systems subject to string constraints," *Automatica*, vol. 50, p. 3224–3230, Dec. 2014.

[109] G. Wen, J. Lam, J. Fu, and S. Wang, "Distributed MPC-based robust collision avoidance formation navigation of constrained multiple USVs," *IEEE Transactions on Intelligent Vehicles*, vol. 9, p. 1804–1816, Jan. 2024.

[110] A. Bienemann and H.-J. Wuensche, "Model predictive control for autonomous vehicle following," in *2023 IEEE Intelligent Vehicles Symposium (IV)*, p. 1–6, IEEE, June 2023.

[111] A. Soni and H. Hu, "Formation control for a fleet of autonomous ground vehicles: A survey," *Robotics*, vol. 7, p. 67, Nov. 2018.

[112] Y. Li, X. Lv, C. Ni, W. He, C. Zheng, and Z. Mao, “Collaborative pursuit method for unmanned ground vehicle formations based on leader-follower,” in *Advances in Guidance, Navigation and Control*, p. 388–397, Springer Nature Singapore, 2025.

[113] Y. Song, J. Lyu, S. Wei, H. Fang, and Q. Yang, “Modular design and implementation of formation motion planning system for multi-unmanned vehicles,” in *Proceedings of 2023 7th Chinese Conference on Swarm Intelligence and Cooperative Control*, p. 580–590, Springer Nature Singapore, 2024.

[114] S.-L. Dai, S. He, X. Chen, and X. Jin, “Adaptive leader–follower formation control of nonholonomic mobile robots with prescribed transient and steady-state performance,” *IEEE Transactions on Industrial Informatics*, vol. 16, p. 3662–3671, June 2020.

[115] J. Shao, G. Xie, J. Yu, and L. Wang, “Leader-following formation control of multiple mobile robots,” in *Proceedings of the 2005 IEEE International Symposium on, Mediterranean Conference on Control and Automation Intelligent Control, 2005.*, p. 808–813, IEEE, 2005.

[116] A. Salimi Lafmejani and S. Berman, “Nonlinear MPC for collision-free and deadlock-free navigation of multiple nonholonomic mobile robots,” *Robotics and Autonomous Systems*, vol. 141, p. 103774, July 2021.

[117] R. Van Parys and G. Pipeleers, “Distributed MPC for multi-vehicle systems moving in formation,” *Robotics and Autonomous Systems*, vol. 97, p. 144–152, Nov. 2017.

[118] S. Novoth, Q. Zhang, K. Ji, and D. Yu, "Distributed formation control for multi-vehicle systems with splitting and merging capability," *IEEE Control Systems Letters*, p. 355–360, 2020.

[119] R. M. Bhatt, C. P. Tang, and V. N. Krovi, "Formation optimization for a fleet of wheeled mobile robots — a geometric approach," *Robotics and Autonomous Systems*, vol. 57, p. 102–120, Jan. 2009.

[120] W. Wu, X. Qin, J. Qin, X. Yu, and Q. Liu, "Flexible formation control of multiple unmanned vehicles based on artificial potential field method," in *Bio-Inspired Computing: Theories and Applications*, p. 567–577, Springer Nature Singapore, 2023.

[121] J. Gao, S. Li, Y. Lyu, Y. Jiang, G. Zhan, Y. Yang, Y. Yin, M. Wang, and S. E. Li, "CIDC: A flexible formation control framework for intelligent connected vehicles," *IEEE Transactions on Intelligent Vehicles*, p. 1–10, 2024.

[122] C. Bai, P. Yan, W. Pan, and J. Guo, "Learning-based multi-robot formation control with obstacle avoidance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, p. 11811–11822, Aug. 2022.

[123] T. Gao, Z. Li, Z. Xiong, L. Wen, K. Tian, and K. Cai, "Queue formation and obstacle avoidance navigation strategy for multi-robot systems based on deep reinforcement learning," *IEEE Access*, vol. 13, p. 14083–14100, 2025.

[124] V. Sezer and M. Gokasan, "A novel obstacle avoidance algorithm: "Follow the Gap Method"," *Robotics and Autonomous Systems*, vol. 60, p. 1123–1134, Sept. 2012.

[125] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine Learning*, vol. 20, p. 273–297, Sept. 1995.

[126] J. Miura, “Support vector path planning,” in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, p. 2894–2899, IEEE, Oct. 2006.

[127] N. Morales, J. Toledo, and L. Acosta, “Path planning using a multiclass support vector machine,” *Applied Soft Computing*, vol. 43, p. 498–509, June 2016.

[128] D. Goldfarb and A. Idnani, “A numerically stable dual method for solving strictly convex quadratic programs,” *Mathematical Programming*, vol. 27, p. 1–33, Sept. 1983.

[129] D. Kraft, “Algorithm 733: TOMP–Fortran modules for optimal control calculations,” *ACM Transactions on Mathematical Software*, vol. 20, p. 262–281, Sept. 1994.

[130] R. Rajamani, *Vehicle Dynamics and Control*. Springer US, 2012.

[131] P. Polack, F. Altche, B. d’Andrea Novel, and A. de La Fortelle, “The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?,” in *2017 IEEE Intelligent Vehicles Symposium (IV)*, p. 812–818, IEEE, June 2017.

[132] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, “Monte Carlo localization for mobile robots,” in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, vol. 2 of *ROBOT-99*, p. 1322–1328, IEEE, 1999.

[133] D. Fox, “KLD-sampling: Adaptive particle filters.,” in *In Advances in Neural Information Processing Systems 14*, pp. 713–720, Jan. 2001.

[134] V. S. Babu and M. Behl, “f1tenth.dev - an open-source ROS based F1/10 autonomous racing simulator,” in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, p. 1614–1620, IEEE, Aug. 2020.

[135] M. O’Kelly, V. Sukhil, H. Abbas, J. Harkins, C. Kao, Y. V. Pant, R. Mangharam, D. Agarwal, M. Behl, P. Burgio, and M. Bertogna, “F1/10: An open-source autonomous cyber-physical platform,” 2019.

[136] N. Yang, Y. Wang, and L.-P. Chau, “Multi-object tracking with tracked object bounding box association,” 2021.

[137] H. Wu and W. Li, “Robust online multi-object tracking based on KCF trackers and reassignment,” in *2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, p. 124–128, IEEE, Dec. 2016.

[138] K. M. Lynch and F. C. Park, *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, May 2017.

[139] S.-H. Kim and Y. J. Ahn, “An approximation of circular arcs by quartic Bézier curves,” *Computer-Aided Design*, vol. 39, p. 490–493, June 2007.