PATHFINDER: LOCAL OBSTACLE AVOIDANCE FOR LONG-TERM AUTONOMOUS PATH FOLLOWING

by

Jordy Sehn

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science

Department of the Institute for Aerospace Studies
University of Toronto

Pathfinder: Local Obstacle Avoidance for Long-Term Autonomous Path Following

Jordy Sehn
Master of Applied Science
Department of the Institute for Aerospace Studies
University of Toronto
2023

Abstract

Visual Teach and Repeat 3 (VT&R3), a generalization of stereo VT&R, achieves long-term autonomous path-following using topometric mapping and localization from a single rich sensor stream. In this thesis, we improve the capabilities of a LiDAR implementation of VT&R3 to reliably detect and avoid obstacles in changing environments. Our architecture simplifies the obstacle-perception problem to that of place-dependant change detection. We then extend the behaviour of generic sample-based motion planners to better suit the teach-and-repeat problem structure by introducing a new edge-cost metric paired with a curvilinear planning space. The resulting planner generates naturally smooth paths that avoid local obstacles while minimizing lateral path deviation to best exploit prior terrain knowledge. While we use the method with VT&R, it can be generalized to suit arbitrary path-following applications. Experimental results from online run-time analysis, unit testing, and qualitative experiments on a differential drive robot show the promise of the technique for reliable long-term autonomous operation in complex unstructured environments.

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Professor Tim Barfoot, for his invaluable guidance, unwavering support, and continuous encouragement throughout the duration of my master's thesis. Professor Barfoot's expertise, profound knowledge, and insightful feedback have played an instrumental role in shaping the direction and success of this research project. I am truly grateful for his mentorship and the opportunity to work under his supervision.

I am also indebted to my colleagues in the Autonomous Space Robotics Laboratory for their collaboration, constructive discussions, and shared enthusiasm for exploring the frontiers of robotics. Their diverse perspectives and expertise have enriched my understanding and aided in overcoming challenges encountered during this research journey. The camaraderie and exchange of ideas within the laboratory have been a constant source of inspiration and motivation. A special thanks to Yuchen Wu who served as a valuable resource to the integration of this work with VT&R3 and to all of the pioneering VT&R2 developers whose work over the last decade was critical in establishing the foundations for this research.

Furthermore, I would like to extend my heartfelt appreciation to my family, friends and loved ones for their unwavering support, patience, and understanding throughout this academic pursuit. Their encouragement and belief in my abilities have been crucial in keeping me motivated and focused. Their unconditional love and belief in my potential have given me the strength to overcome obstacles and persevere through the demanding phases of this thesis.

Lastly, I would like to express my sincere gratitude to all the individuals who have contributed to my growth and development as a researcher and as an individual. Your guidance, encouragement, and friendship have been invaluable, and I am truly fortunate to have you in my life. In particular, I would like to thank Jack Collier from Defence Research and Development Canada for giving me my first opportunity in robotics and for continuing to be a constant source of encouragement, resources, and motivation in my career development. I certainly would not be in the position I am today without your support.

This thesis would not have been possible without the support and assistance of these exceptional individuals. Their contributions have played a significant role in shaping the outcome of this research, and I am deeply grateful for their involvement. Thank you all for being an integral part of my academic and personal journey. Your support and encouragement have made this achievement possible.

Contents

List of Tables

List of Figures

Acronyms

BIT* Batch Informed Trees

DOF Degrees of Freedom

DRDC Defence Research and Development Canada

DWA Dynamic Window Approach

GPS Global Positioning System

GMM Gaussian Mixture Model

GUI Graphical User Interface

LiDAR Light Detection and Ranging

LPA* Lifelong Planning A*

MPC Model Predictive Control

OGM Occupancy Grid Maps

PID Proportional-Integral-Derivative

RHC Receding Horizon Control

RMS Root Mean Squared

ROC Radius of Curvature

ROS2 Robot Operating System 2

RRT Rapidly Exploring Random Trees

STEAM Simultaneous Trajectory Estimation and Mapping

TEB Timed Elastic Band

UGV Unmanned Ground Vehicle

VT&R Visual Teach & Repeat

VT&R3 Visual Teach & Repeat 3

Notation

a	This font is used for quantities that are real scalars
$\mathbf{a}$	This font is used for quantities that are real column vectors
$\mathbf{A}$	This font is used for quantities that are real matrices
$\mathbf{1}$	The identity matrix
$\underrightarrow{\boldsymbol{\mathcal{F}}}_a$	A vectrix representing a reference frame in three dimensions X
$SE(3)$	The special Euclidean group, a matrix Lie group used to represent poses
$\mathfrak{se}(3)$	The Lie algebra associated with $SE(3)$
$\mathbf{T}_{ba}$	A matrix in $SE(3)$ that transforms vectors from frame $\underrightarrow{\boldsymbol{\mathcal{F}}}_a$ to $\underrightarrow{\boldsymbol{\mathcal{F}}}_b$
$\exp(\cdot^\wedge)$	A Lie algebra operator mapping from $\mathfrak{se}(3)$ to $SE(3)$
$\ln(\cdot)^\vee$	A Lie algebra operator mapping from $SE(3)$ to $\mathfrak{se}(3)$

Chapter 1

Introduction

1.1 Motivation

Robot navigation in unstructured outdoor environments is a challenging-yet-critical task for many mobile robotics applications including transportation, mining, and forestry. In particular, robust localization in the presence of both short- and long-term scene variations without reliance on a Global Positioning System (GPS) becomes particularly difficult. Furthermore, the off-road terrain-assessment problem is non-trivial to generalize as the variety of potential obstacles increases, all of which require careful identification, planning, and control to prevent collisions.

VT&R [1] tackles these problems by suggesting that often it is sufficient for a mobile robot to operate on a network of paths previously taught by a human operator. During a learning phase (the *teach pass*) the robot is manually piloted along a route whilst building a visual map of the environment using a rich sensor such as a stereo camera. In the autonomous traversal phase (the *repeat pass*), live stereo images are used to localize to the map with high precision and resiliency to lighting and seasonal changes [2, 3].

In practice, this architecture works well to address both the localization and terrain-assessment problems. By having a human operator drive the teach pass, we exploit the strong prior that the original path was traversable. It follows that, at least in the short term, it is likely the taught path remains collision free and by following the path closely in the repeat, minimal terrain assessment is required.

With the advent of Light Detection and Ranging (LiDAR) implementations of teach and repeat [4–6] whose localization system is less sensitive to viewpoint changes when repeating a path than stereo, we explore the possibility of temporarily deviating from the teach path to avoid new obstacles and increase the practicality of VT&R. In this application, we highlight the reliability of change detection for obstacle perception in difficult environments. Additionally, we emphasize the importance of path planning that minimizes deviations from the taught path to take advantage of the human terrain-assessment prior whenever possible. This allows us to navigate in the safest possible manner in the event of terrain-assessment errors.

While a local obstacle-avoidance scheme with these properties is desirable for teach-and-repeat

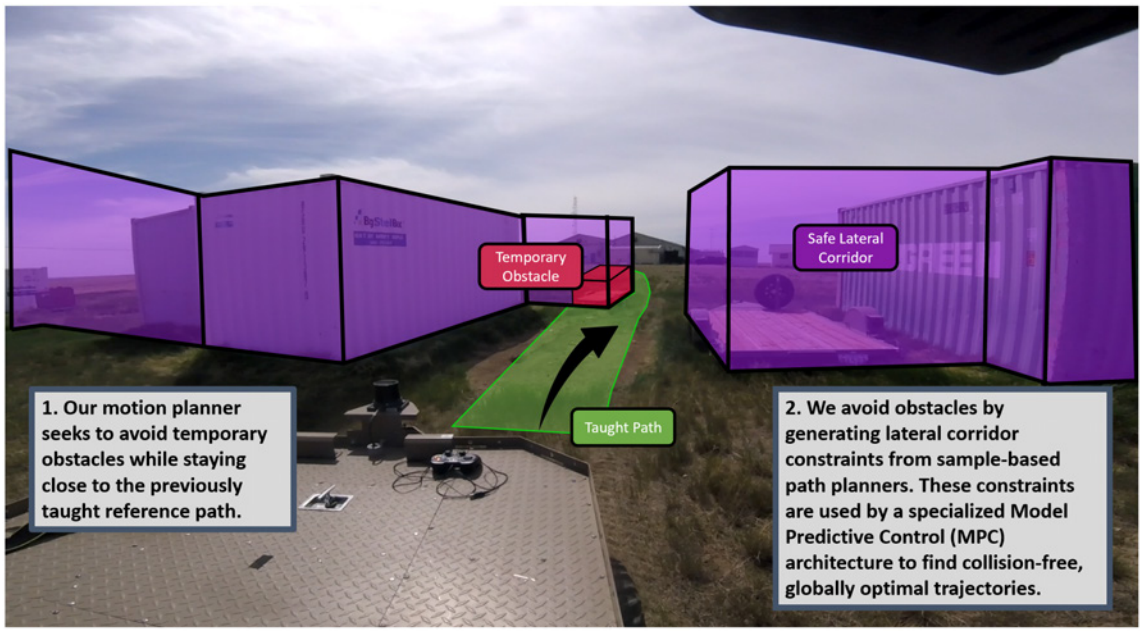


Figure 1.1: We present an architecture for locally avoiding unmapped obstacles along a reference path by extending sample-based motion planners to encourage trajectories with characteristics that best exploit prior path knowledge. Our system is validated on an ARGO Atlas J8 robot in real-world scenarios.

architectures, its practicality extends beyond this specific application. The ability to deviate from a pre-taught path to avoid obstacles and minimize deviations can be beneficial in various path-following scenarios, such as from autonomous driving of map routes or predetermined GPS bread-crumb following. By incorporating such adaptable and robust obstacle perception and path planning strategies, we can enhance the safety and efficiency of autonomous navigation in a wide range of environments. Therefore, the findings of this study not only contribute to the advancement of teach-and-repeat systems but also offer valuable insights for broader path-following applications.

1.2 Contributions

The primary contribution of our system is in the planning domain where we present a novel edge-cost metric that, when combined with a curvilinear planning space, extends the capabilities of a generic sample-based optimal motion planner to generate paths that naturally avoid local obstacles along the teach pass. Notably, in obstacle-free environments, our planner ensures that the solution remains on the original teach path without cutting corners. The key feature of our edge-cost metric is its ability to encourage path solutions that strike a delicate balance between minimizing path length and lateral path deviation, all achieved without relying on waypoints. Additionally, we demonstrate that our new metric maintains the fundamental properties of the underlying planning algorithm by utilizing an admissible sampling heuristic.

After generating suitable obstacle-avoidance plans, we propose two distinct control topologies for trajectory generation using Model Predictive Control (MPC). In the first case, we directly track

the planner output paths while utilizing MPC to enforce basic kinematic and acceleration constraints. This approach generates smooth trajectories that effectively avoid local obstacles. In the second implementation, we decouple the MPC from the planner by explicitly following the taught reference path instead of the planner output. To avoid obstacles, the current path solution is used to enforce a set of corridor state constraints, ensuring a collision-free and robust MPC trajectory.

The motion planning system is then integrated with VT&R3 and supplemented with a custom, multifaceted speed-scheduling algorithm to regulate the repeat velocity, allowing the robot to navigate the environment smoothly and safely while maintaining a consistent pace.

To illustrate the effectiveness of our path planner, we have conducted extensive simulations on a diverse set of realistic path-following scenarios. Through long-term autonomous navigation experiments spanning several kilometres in various unstructured environments, we showcase the entire system's robust obstacle-avoidance capabilities and its potential for real-world deployment.

An overview of the teach-and-repeat framework is presented in Chapter 2 and the specific implementation details of each contribution are presented in Chapters 3 and 4, respectively. We then evaluate the system in online obstacle avoidance experiments on a differential drive robot in Chapter 5. Finally a discussion of the main conclusions for this research and the suggestion of future directions is provided in Chapter 6.

Chapter 2

Visual Teach & Repeat

This chapter delves into the problem structure and software development stack that have shaped the design choices of the proposed local obstacle-avoidance system. It is important to note that while we frame the planning and control problem within the context of VT&R3, the contributions of this work extend beyond teach and repeat and can serve as a local obstacle-avoidance module for more traditional robot navigation schemes.

We begin by providing a review of the teach-and-repeat development history, offering valuable background information. Subsequently, we discuss the limitations observed in previous teach-and-repeat implementations that create the niche that our research aims to address. We then present an overview of the relevant systems we interface with in VT&R3 and a discussion of the related literature.

2.1 History

2.1.1 Background

VT&R is an autonomous navigation framework that enables a robot to learn and repeat paths based on visual information and without GPS. It utilizes a combination of online data collection, map generation, and offline playback to achieve long-range path following. During the teaching phase, a human operator manually guides the robot along the desired path while onboard sensors, such as cameras and range finders, collect data. The robot records a sequence of camera images and captures relational information that suggests how to move from one image to the next. This information could include visual features, keypoint correspondences, or other relevant data. The teaching phase aims to create a reference dataset that represents the desired path.

Once the teaching phase is complete, the system proceeds to the map generation step. Online processing is performed on the recorded data to create a representation of the environment. This consists of local maps or occupancy grids that cover different sections of the path. The maps are interconnected to form a graph structure that represents the path and its surrounding environment and provides a spatial reference for the robot during path repetition. During a repeat, the robot uses

the recorded map and visual information to autonomously navigate along the taught path. It matches the current camera images against the recorded images and employs the relational information to determine its position and orientation relative to the reference path. Based on this information, the robot can follow the path accurately, adjusting its trajectory to minimize tracking errors.

The key advantage of the VT&R approach is its ability to repeat paths in varying environments without relying on external infrastructure like GPS or beacons. Instead, it leverages visual information and the map generated during teaching to localize and navigate autonomously. This makes VT&R particularly suitable for scenarios where accurate and reliable positioning systems may not be available or feasible.

2.1.2 Visual Teach & Repeat 1

The development of VT&R has a rich history. Early pioneers in autonomous navigation systems, such as Matsumoto et al. [7] and Howard [8], explored the concept of the teach-and-repeat framework. Matsumoto et al. [7] referred to the teach pass as a recording run, where a robot captured camera images and relational information to navigate along a recorded path. By matching against these recorded images and leveraging the relational information, the robot could replicate the recorded trajectory. Howard [8] introduced the manifold map, a topometric map similar to the one later used in VT&R, discussing and demonstrating its advantages in incremental mapping and retro-traversing.

The foundation of the VT&R system can be traced back to the work of Marshall et al. [9, 10], who developed an automated system for the "load-haul-dump" (LHD) cycle of under tramming vehicles. The system consisted of three steps: teaching, route profiling, and playback. During teaching, the vehicle was manually driven along the desired route while collecting data from onboard sensors. Offline route profiling used the logged data to generate path profiles, speed profiles, pause profiles, and sequences of local maps represented by overlapping occupancy grids. Subsequently, the system could autonomously play back the route profile. To achieve precise path following at the desired speed, an Unscented Kalman Filter (UKF)-based localizer and a feedback linearization controller were utilized, resulting in tracking errors of less than 50 cm.

Expanding on the success of [10], Furgale and Barfoot introduced VT&R1 in 2010 [1]. This comprehensive navigation system enabled long-range path following in non-planar terrain using only a stereo camera as the sensor. Similar to [10], VT&R1 involved online data collection along the desired path and offline map generation. However, it employed a variant of the Speeded Up Robust Features (SURF) algorithm for stereo visual odometry to handle non-planar terrain with the stereo camera. The local maps generated by VT&R1 consisted of triangulated SURF features as 3D landmarks, creating a hybrid topological/metric representation known as a topometric map.

2.1.3 Visual Teach & Repeat 2

While VT&R1 demonstrated the potential of using overlapping locally consistent maps for long-range autonomy, it had limitations in long-term operation and practicality. Visual features extracted

from stereo images were susceptible to environmental changes, resulting in increased localization failures over time. Interleaving frame-to-frame odometry and frame-to-map localization partially addressed the issue but limited VT&R1's operational window to a few hours.

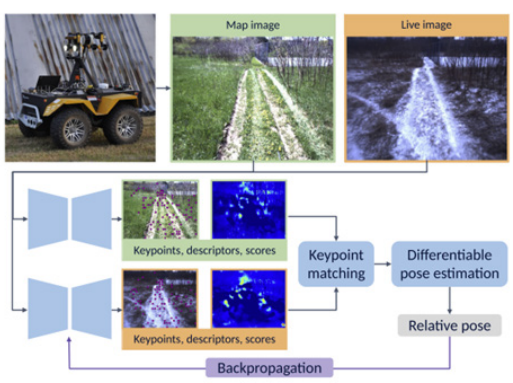
To overcome these limitations and improve usability, VT&R2 was introduced in 2017, accompanied by significant enhancements in long-term operation capability and ease of use. VT&R2 employed a hybrid topological/metric mapping approach, representing the map as a relative pose graph with metric information associated with vertices [11]. Unlike VT&R2, map building occurred online, allowing immediate path repetition after the initial demonstration. Taught paths formed chains of poses in the graph that could connect to other paths on both ends. This approach facilitated teaching paths in multiple sessions, branching off and optionally merging into existing paths. The smooth path-following across branching and merging points ensured seamless repetition of paths constructed from multiple sessions.

VT&R2 addresses the challenge of localization failure caused by changes in the environment through two algorithmic enhancements to the visual localization pipeline employed in VT&R1: Multi-Channel Localization (MCL) [12] and Multi-Experience Localization (MEL) [2]. VT&R2 integrates localization information from multiple sources into a unified estimation problem through Multi-Channel Localization (MCL). Each source, called a channel, consists of stereo images captured by a single camera and undergoes various transformations. VT&R2 modifies the visual localization pipeline to independently extract and track features for each channel, merging the data correspondences for state estimation. It employs two variations: a light-resistant variant using colour-constant images [13] and a duo-stereo variant with two forward and backward facing cameras [14]. The light-resistant variant adds an extra channel for each camera, operating on colour-constant transformed images that are less affected by changes in illumination. The duo-stereo variant includes additional channels to capture more visual features for improved localization, utilizing the expanded field of view. MCL extends the operational window from a few hours to multiple days compared to VT&R1.

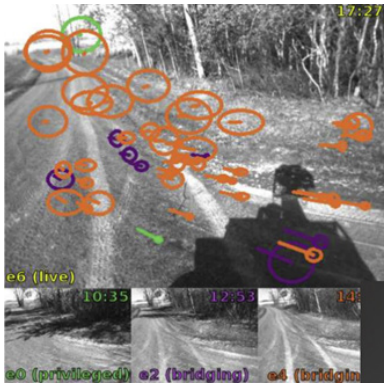
Multi-Experience Localization (MEL) addresses gradual changes in the environment's appearance by utilizing additional views from repeat passes. Experiences consist of visual features and their 3D locations obtained during teach or repeat passes. VT&R2 also introduces the Spatial-Temporal Pose Graph (STPG) to store extra experiences and establish metric relationships with the reference experience from the teach pass. By simultaneously considering all past experiences within the STPG, the live experience can be matched to solve a single estimation problem, significantly enhancing localization despite environmental changes. To manage computational costs, VT&R2 implements two algorithms for selecting relevant experiences. The first employs Bag of Words (BoW) descriptors, enabling fast similarity comparison [15]. The second algorithm utilizes collaborative filtering to select experiences with similar feature matching history [14]. MEL allows VT&R2 to operate effectively across seasonal changes, provided there are sufficient experiences to bridge the appearance gap.

2.1.4 Modern Variants

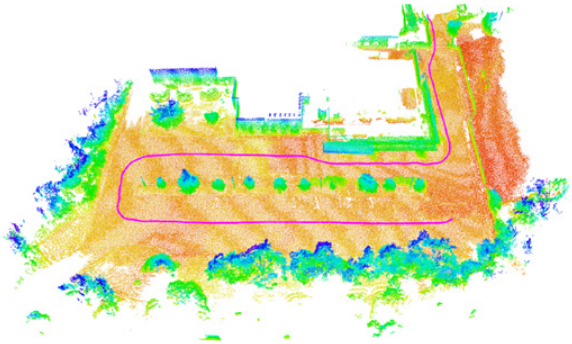
In recent years, there have been various other advancements to the VT&R2 architecture. These include a learning-based path tracker [16, 17], UAV implementations that focus on localizing from satellite imagery [18, 19], and a monocular camera version [20]. A significant development in the VT&R framework occurred in 2022 with the release of VT&R3 [21]. This new version was designed from scratch to be compatible with the latest software and hardware. It inherits the main features of VT&R2 while allowing more flexibility in terms of algorithms and sensor-robot configurations. The estimation, planning, and control problems in VT&R were re-formulated in a generic manner, with interfaces that separate them from the teach-and-repeat navigation logic. Notably, VT&R3 supports both LiDAR [4, 21] and RADAR [22] implementations of teach and repeat, with the former serving as the foundation for the research conducted in this study. A summary of a small selection of important VT&R advances is shown in Fig. 2.1. The code for VT&R3 is open-source and can be found at **https://github.com/utiasASRL/vtr3**.



(a) Deep Learned Visual Features [3]



(b) Multi-Experience Localization [2]



(c) LiDAR Teach and Repeat [21]

(d) Colour-Constant Imagery [13]

Figure 2.1: A summary of recent advances to the VT&R architecture

While VT&R3 represents a significant advancement, it is not the only recent implementation of the teach-and-repeat framework. Other vision-based implementations have been explored in the literature [23–26], primarily focusing on evaluating various visual localization techniques to address

appearance and viewpoint changes. However, most of these implementations lack the ability to construct a network of paths, operate in a 3D environment, or achieve precise metric localization. In contrast, VT&R3 demonstrates a higher level of maturity. Another notable system similar to VT&R is presented in [27], utilizing LiDAR technology for odometry, mapping, and localization based on ICP-based point cloud registration. The use of LiDAR enhances the system's robustness to lighting variations, while the expanded field of view enables deviation from the reference path for obstacle avoidance. Additionally, Baril et al. [28] have recently developed a LiDAR-based teach-and-repeat system, similar to the one in [27], specifically focusing on the robustness of LiDAR localization in challenging environments with dense vegetation and snowfall.

2.2 Limitations

In the previous section we note that past versions of VT&R had limitations in obstacle avoidance due to sensor constraints. Camera-based VT&R systems often faced localization failures when the robot deviated from the taught reference path to avoid obstacles. However, despite this limitation, VT&R remained a highly reliable navigation system. This is because the taught path provides a strong prior on the traversability of the environment. Since the path was manually taught by a human, it is assumed that, at least at one point, the path was completely traversable. Hence, during a repeat, a naive yet practical strategy is to closely follow the taught path blindly, disregarding the terrain assessment problem completely. Although this assumption may break down over time due to changes in the static environment or the presence of dynamic obstacles, over the vast majority of path repeats this assumption holds true. Therefore it is important for any navigation strategy to leverage this prior knowledge in the presence of perception uncertainties.

The emergence of 3D LiDAR-based VT&R architectures [4, 5, 21, 28] with improved localization robustness up to several meters away from the path has made local obstacle avoidance more feasible. This advancement inspired the main objective of this thesis: to develop a reliable local obstacle-avoidance system for teach-and-repeat architectures. This system would allow the robot to temporarily deviate from the taught reference path if necessary to avoid obstacles and successfully accomplish the repeat objective. If successful, this local obstacle-avoidance module can expand the reliability of long-term autonomous navigation using teach-and-repeat methods.

We are not the first to consider the importance of obstacle-avoidance in teach-and-repeat frameworks. Despite the localization issues Cherubini [6] cleverly extends a camera-based teach-and-repeat system to manoeuvre around obstacles in outdoor urban environments. By adding an additional laser scanner, they detect new obstacles during repeats and use an approach similar to the classical Elastic Band [29] to plan a deformable path, free of collisions. Critically, they maintain localization by actuating the camera pan angle to regulate scene visibility, however there is still strict viewpoint limitations that inhibit practicality.

Krusi et al. [5] utilized the additional mobility freedom to navigate reliably in dynamic environments. They actively detected potential obstacles by analyzing irregularities in the distance between

concentric LiDAR rings, assuming local planarity of the ground. Once detected, an online motion planner generated a tree of 21 potential trajectories around the obstacles, selecting the one that best satisfied system constraints. The chosen trajectory was then tracked using an internal control law. This approach improved the long-term autonomy of the teach-and-repeat architecture but was limited to mostly structured environments with simple obstacle geometries.

In 2022, Mattamala et al. [30] explore the use of a locally reactive controller for completing visual teach-and-repeat missions in the presence of obstacles. Their research employs local elevation maps to compute vector representations of the environment and directly generate control twist commands using a Riemannian Motion Policies controller. They deploy this solution on slow-moving (0.3 m/s) quadruped robots within indoor environments.

The approach presented by Mattamala et al. demonstrates effectiveness in handling numerous small obstacles, benefiting from its efficient computational processing. However, it has shown a tendency to encounter challenges when dealing with more complex obstacle formations, as it relies on maneuvering through Signed Distance Fields and can fall into local minima. Additionally, the proposed perception system is primarily evaluated in planar environments, limiting its ability to differentiate between traversable obstacles, such as tall grass, and non-traversable obstacles like barriers. It is also important to note that while the control scheme does exhibit reasonable path tracking errors by employing periodic lookahead goals, the controller lacks explicit considerations for preserving consistent viewpoints and limiting lateral path deviations during obstacle avoidance. These factors are crucial for mitigating localization failures during repeats.

In the remainder of this chapter, we will discuss some of the key modules of VT&R3 that facilitate an obstacle-avoidance system, and then we present the basic structure of our proposed motion planner designed to replace the internal VT&R3 path tracker.

2.3 VT&R3 Architecture Overview

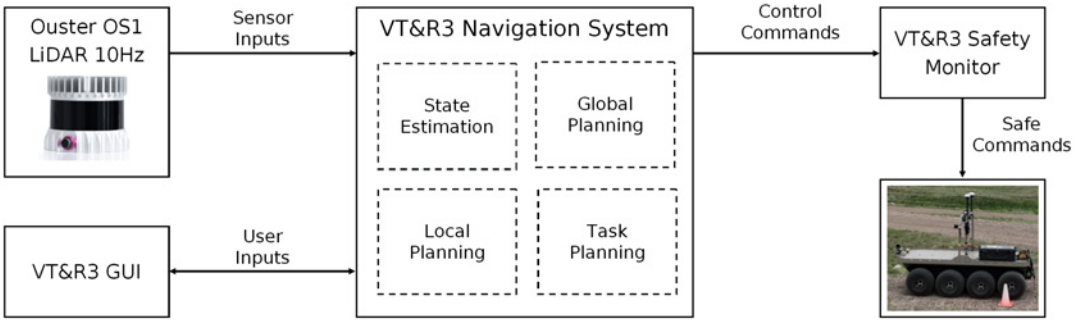


Figure 2.2: This figure provides a high-level overview of the VT&R3 system. It can be summarized into three components: VT&R3 Navigation System, GUI, and Safety Monitor.

A high-level overview of the VT&R3 architecture is shown in Fig. 2.2. Broadly, we can separate the system into three main components: The Navigation System, a Graphical User Interface (GUI)

and a Safety Monitor layer.

The Navigation System, written in C++, utilizes state estimation, planning, and control algorithms to execute tasks within the teach-and-repeat framework. It communicates with the GUI for visualization and user commands, as well as the Safety Monitor for control commands and diagnostics. The Robot Operating System 2 (ROS2) middleware facilitates secure and low-latency communication between these components. The Navigation System is modularized into separate modules for state estimation, perception, task planning, and control, with each module running in its own thread. Notably, the work established in this thesis introduces a significant improvement by replacing the previous blind path-following module with a full motion planner that incorporates local obstacle-avoidance capabilities.

The Safety Monitor acts as an independent watchdog process to ensure safe operation of the robot in critical failure situations. It performs safety checks based on mechanical actuation limits and localization status, overriding underlying modules if necessary to stop the robot.

The GUI is a user-friendly, single-page web application. Its server-side backend is developed in Python, while the client-side frontend is implemented in JavaScript. The backend acts as a message converter, bridging the communication between the navigation system and the frontend. User inputs from the frontend are transmitted to the backend using WebSocket, a bidirectional real-time communication protocol based on TCP. The backend converts these user inputs into ROS2 messages and publishes them to the navigation system. Conversely, incoming ROS2 messages from the navigation system are transmitted back to the frontend.

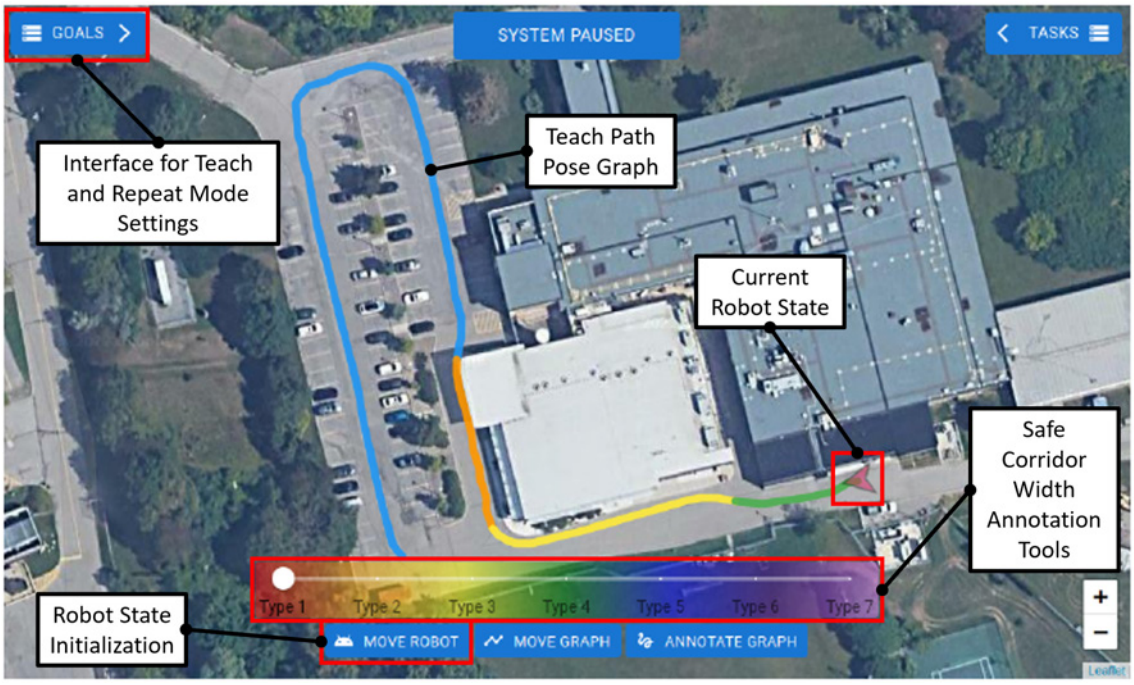


Figure 2.3: This figure shows a screenshot of the VT&R3 Graphical User Interface, with the main components highlighted.

In the planning and control process, the GUI, shown in Fig. 2.3, plays a crucial role by providing essential information to the user. It allows the user to initiate teach-and-repeat operations, expanding the existing network of paths or autonomously traversing them. When a previously taught pose graph is loaded from the GUI, the user can specify objective poses for the robot, determining the start and end point for the repeat operation that defines the reference path for that mission. This information is then passed to the navigation system that retrieves the corresponding pose graph for use in the motion planner.

The GUI additionally provides the option for the user to manually annotate segments of the pose graph with a terrain classification. These terrain classes associated a maximum corridor width around the pose graph network that is meant to establish how much freedom the path planner and controller has while avoiding obstacles. For example, if the path runs alongside a cliff, the local planner should be constrained to a narrow corridor. Conversely, in open field segments, the local planner can have more flexibility to deviate from the path and explore wider trajectories, if it is deemed beneficial.

2.4 Local Obstacle Avoidance Literature

Obstacle avoidance in unstructured environments is a broad and well-studied field. Generally, the literature can be categorized into the solution of two sub-problems: (i) obstacle detection and (ii) local trajectory planning given a set of obstacles.

2.4.1 Obstacle Detection

As stereo-based teach-and-repeat obstacle avoidance is more challenging owing to changes in viewpoint, we focus our discussion to LiDAR perception architectures. Obstacle detection in this work can be related to the problem of unsupervised object discovery from LiDAR change detection, which is often handled jointly with long-term mapping [31–33]. We are concerned with detecting changes in the environment from two sets of LiDAR measurements or their derived representations (i.e., point clouds, voxel grids), with one considered a query and the other as a reference. The most direct approach is accumulating measurements from both sets into point clouds, computing the nearest-neighbour distances between points from the two clouds, and classifying changes by thresholding the distance [34]. Although more sophisticated classification techniques have been explored in [35–37], decent results can be achieved if the point clouds are dense and there is no need to reason about occlusions.

The perception system in VT&R3 [21] is inspired by previous terrain-assessment works for VT&R that also rely on change detection [38, 39]. In [38], the terrain ahead of the robot is organized into robot-sized patches, each of which is a 2D grid storing the estimated terrain height. These patches are compared with those from previous experiences, and any significant difference causes the respective terrain to be marked non-traversable. The approach is improved in [39], with each cell storing a 1D Gaussian Mixture Model (GMM) with two Gaussians in the vertical direc-

tion. One Gaussian models the terrain height and roughness, and the other addresses noises due to overhanging vegetation. Our method learns from [39] to explicitly account for surface roughness in change detection using a Gaussian model, improving the detection of small obstacles in structured environments and applies the notion to LiDAR point clouds.

As the obstacle-detection system is not the focus of evaluation for this work, we direct the reader to Wu [21] for implementation details. At a high level, we treat obstacle detection as a change-detection problem between the environment's long-term static structures and the current LiDAR observation. If a point from the live scan fails to coincide with a mapped structure, it is likely from a new obstacle, or is representative of a significant change to the environment that should be avoided. We then adopt classical methods for classifying points in the scan based on a thresholding method similar to [34], while modelling surface roughness with a Gaussian in our classification metric as in [38, 39]. After generating point-wise 3D obstacle detections, we project the obstacles to local 2D Occupancy Grid Maps (OGM) and apply a temporal filter similar to [40, 41] to help reject false positive detections. Although the approach is not infallible, generally obstacle detection rates are sufficient for navigation in difficult unstructured environments. Going forward we take for granted that all obstacles are reliably detected and projected onto local 2D occupancy grids for collision checking.

2.4.2 Local Path Planning

In the teach-and-repeat problem structure, we have a reference path to follow and wish to temporarily deviate from the path to avoid obstacles. Ensuring the planner returns the exact nominal reference path, including when paths intersect or loop, is imperative under obstacle-free conditions, particularly for applications requiring complete path coverage. Consequently, when the robot deviates from the reference path to avoid obstacles, the safest path is the one that minimizes the lateral deviation from the reference so that we maximize the use of prior terrain knowledge. In this study, we assess the relevant literature on obstacle avoidance within the framework of these crucial properties.

An effective way to locally plan around obstacles was demonstrated by Selek et al. [42]. They avoid local obstacles by generating a tree of trajectories at each time step and select the best one with respect to a cost function. While simple, the output trajectories are not optimized for path-following, even in obstacle-free environments, an undesirable property for high-precision teach-and-repeat applications. A similar trajectory-sampling approach is used by Said et al. [43]. However, in this case the trajectories are sampled from a curvilinear coordinate space with axes defined by the longitudinal and lateral distances along the reference path, respectively. Using this method ensures that at least one of the sampled trajectories lies exactly on the reference path, resulting in an optimal solution in the event there are no obstacles. Our method uses a similar curvilinear planning space to exploit the same property within a sample-based planner, but differs considerably in the application and is more robust to singularities that form over sharp turns in the reference path.

With the prevalence of many high-speed optimization solvers, direct path-tracking MPC has

become a popular choice [44, 45] for the ability to easily enforce system constraints and tune the behaviour of the controller. In these implementations, a quadratic cost function typically models the error between a robot's predicted state and a path given a sequence of control inputs to be optimized.

To avoid obstacles, Dong et al. [46] adds additional state constraints to MPC. Normally obstacle constraints make the problem non-convex due to the spawning of additional path homotopy classes resulting in local minima. They are able to avoid this problem by approximating obstacles as low-dimensional linear convex hulls to find globally optimal solutions for simple vehicle passing manoeuvres. Lindqvist et al. [47] shows how the use of a nonlinear, non-convex solver such as Proximal Averaged Newton for Optimal Control (PANOC) [48] can successfully solve obstacle-avoidance problems by describing sets of obstacles as nonlinear inequality constraints and then relaxing them with penalty functions. These types of approaches work well in structured environments with few obstacles, but fail to scale well when the number and variety of obstacles increases.

Instead of trying to approximately solve non-convex MPC problems directly, an alternative strategy is to first run a separate path-planning algorithm to find a route that avoids obstacles and then reconnects with the global path. Liniger et al. [49] applies this idea to autonomous racing, using dynamic programming to find rough shortest-distance paths avoiding other vehicles. This path solution is representative of the most promising path homotopy class. By defining convex lateral state constraints on the MPC based on this initial homotopy solution, they are able to generate globally optimal trajectories at high control rates. Patterson et al. [50] applies a more straightforward approach that defines the lateral corridor of MPC constraints using geometric rules based on the location of obstacles relative to the road centre-line.

Of these techniques, our method is most comparable to [49]. The distinction is that we use a sample-based planner, combined with our novel edge-cost metric, to find paths that exploit our prior terrain knowledge by reducing lateral error with the reference and are suitable for large networks of paths. To the best of our knowledge, our approach is the first to directly incorporate lateral penalties to the edge cost of sample-based motion planners. This feature offers a number of planning properties that uniquely complement the teach-and-repeat architecture and improves long-term navigation reliability.

2.5 Local Obstacle Avoidance Architecture

This section describes our approach for autonomous relative path following given a previously taught route. An overview of our architecture is provided in Fig. 2.4, with the main contributions of this thesis being the motion planning module. VT&R3 provides knowledge of the current state of the robot, the pose graph for the path we intend to repeat, and real-time updated LiDAR obstacle detection maps. The internal change detection module utilizes LiDAR scans to generate local 2D occupancy grid maps, labelling newly appeared obstacles during the repeat as occupied cells. Our path planner utilizes this information to find paths that lead the robot to the end of the path, aiming to avoid local obstacles while staying close to the original route. We accomplish this by combining

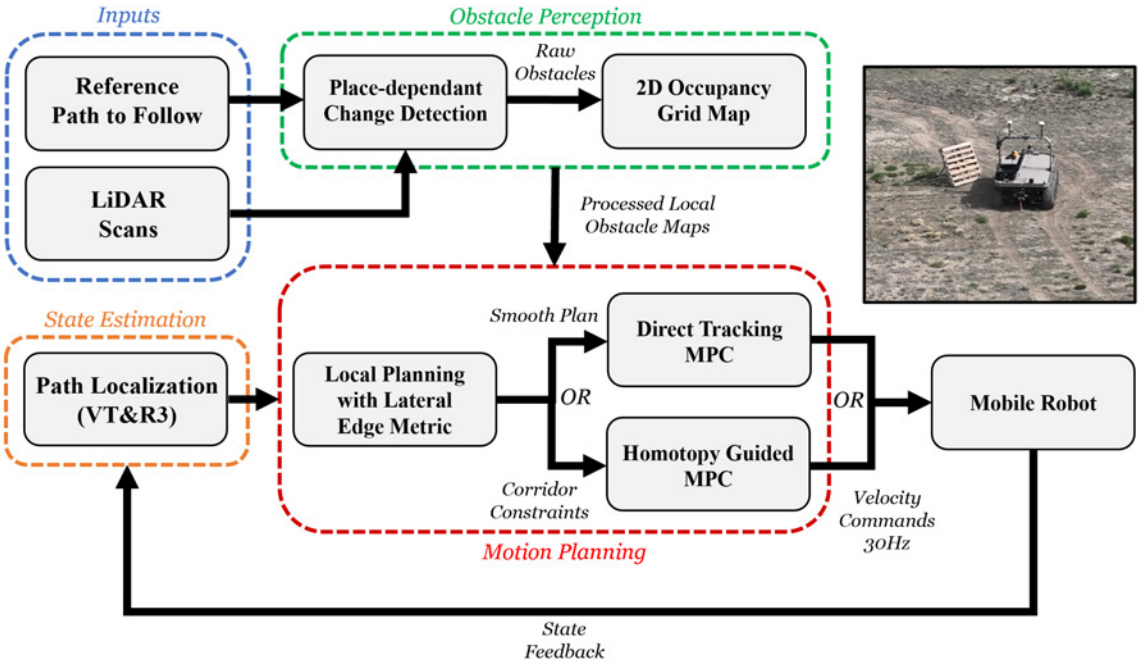


Figure 2.4: An overview of the proposed obstacle-avoidance system. A change-detection LiDAR perception module identifies previously unmapped structures and updates a locally planar 2D occupancy grid map. The planner finds paths that avoid obstacles using our laterally weighted edge-cost metric and we evaluate two different MPC schemes to enforce kinematic constraints on the final trajectory.

a generic sample-based planner with an orthogonal curvilinear coordinate configuration space and a novel lateral edge cost metric that is described in greater detail in Chapter 3.

Next, we present two variants of a secondary MPC module that plays a critical role in finding a kinematically feasible optimal trajectory to safely navigate around obstacles and complete the repeat. In the first implementation, we use a direct tracking MPC that explicitly aims to follow the planned path, making the assumption that it is collision-free. In the second implementation, we decouple the planner from the MPC and adopt a homotopy guided MPC. This approach focuses on directly following the previously taught path while dynamically generating sets of state constraints based on the planner's output. The output of the MPC is a velocity command, conveyed to the robot as a ROS2 Twist message in each iteration of the control loop. A full description of the MPC implementation is provided in Chapter 4.

2.6 Summary

In summary, this chapter provides a comprehensive overview of the teach-and-repeat development history and the limitations observed in previous implementations. We introduce the VT&R framework and its ability to learn and repeat paths based on visual information, emphasizing its advantages in environments where traditional positioning systems may not be available. The chapter further explores the evolution of VT&R from its early versions to the recent VT&R3 release, highlighting the enhancements in long-term operation, robustness, and ease of use. We underscore the

significance of incorporating a local obstacle-avoidance system within a teach-and-repeat architecture to bolster the long-term autonomous reliability of these systems in practical applications. After outlining how a local obstacle-avoidance system seamlessly integrates with VT&R3, we introduce a straightforward yet effective motion planning framework to achieve our goal of safe and efficient path following. We then present a comprehensive evaluation of the system's performance in real-world scenarios, providing valuable insights and practical guidance for implementing teach-and-repeat architectures with improved obstacle-avoidance capabilities.

Chapter 3

Path Planning

In this chapter, we introduce the fundamental contribution of this thesis: a novel adaptation to sample-based path planners, specifically designed for achieving local obstacle avoidance and natural path following. Through our modifications, we effectively leverage the valuable information derived from the pre-existing reference path, allowing us to generate safe and optimized candidate paths that seamlessly navigate around obstacles while adhering to desireable trajectory characteristics. Our initial work, published and presented at the 20th Canadian Conference on Robots and Vision (CRV) [51], serves as the foundation for the research presented and expanded upon in this chapter.

3.1 Introduction

In pursuit of the desired VT&R3 application, our goal is to solve a planning problem that can be succinctly described as follows. Given a reference path, our objective is to plan a path from the robot to the end of the reference path while closely adhering to it, deviating only when necessary to avoid obstacles. Our primary focus is to minimize path tracking error from the reference path in order to effectively utilize prior terrain knowledge, rather than prioritizing speed when repeating the path. Given the wide range of highly diverse unstructured operational environments that can be expected, we also emphasize the critical importance of ensuring the generalizability of the solution. Although our objectives are tailored for VT&R3, it is evident that a similar problem structure can be applied to traditional two-layer local and global planning architectures [52] and that the behaviour we describe is generally desirable for a range of path-following applications.

A naive approach to solving this problem would involve using an established local trajectory planner, such as modern variants of the Dynamic Window Approach (DWA) [53, 54] or the Timed Elastic Band (TEB) [55, 56], configured to follow a series of consecutive waypoints along the global path. While this approach may be effective, working with waypoints can be cumbersome, particularly for complex paths. One crucial question arises: how far ahead should we place the waypoints? Placing them too far in advance risks skipping path segments that loop or intersect, which is undesirable in the context of VT&R3. For example, consider a lawn mowing application where complete path coverage is a necessity of completing the objective. On the other hand, placing

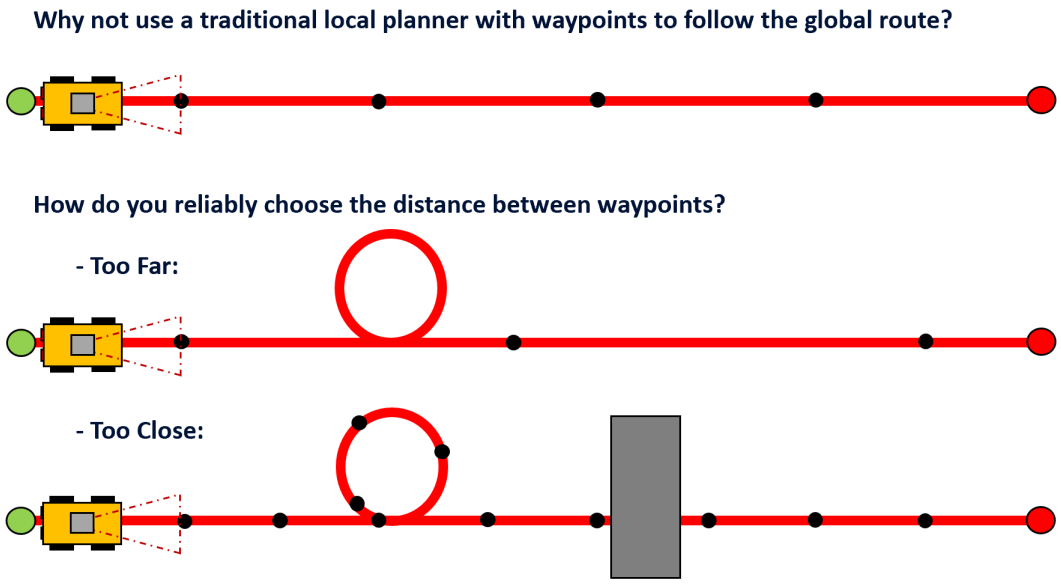


Figure 3.1: In this figure, we outline some potential disadvantages of using sequences of waypoints for path following, particularly when it comes to spacing selection. Sparse waypoints risks bypassing crucial segments of the reference path, while dense waypoints pose challenges in accommodating unforeseen obstacles during planning.

waypoints too close can lead to jittery behaviour and corners being cut at times. Additionally, when obstacles are introduced near waypoints, this spawns another difficult choice. Should we plan right next to the obstacle and then navigate around it? Or do we take a more conservative approach and skip the waypoint? While solutions exist for addressing each of these problems individually, our ultimate objective is to identify a path planner that embodies more natural path following properties.

Inspired by the waypoint example, we sought to establish a set of path planner principles that describe an idealized planner capable of addressing the requirements of global path following with local obstacle avoidance.

Desired Path Planner Principles:

1. *Immediate Obstacle-free Solutions:* When operating in an obstacle-free environment, the planner should naturally return the nominal reference path. Since we already possess knowledge of the optimal obstacle-free solution from the reference plan, the local planner should default to this path-following strategy.

2. *Preservation of the Reference Path:* The planned path should never take shortcuts or deviate from the reference path unnecessarily. This principle is particularly significant in applications such as lawn mowing, where it is crucial to faithfully repeat the reference route without cutting corners.

3. *Balancing Path Length and Lateral Deviation:* In the presence of obstacles, the planner should strike a balance between minimizing both path length and lateral deviation from the

reference path. This principle is of utmost importance in the context of VT&R3, where we possess reliable prior information about the traversability of the reference path. Consequently, actions taken to avoid obstacles should prioritize minimizing lateral deviation while still making forward progress. This approach offers the best chance of avoiding obstacles that may have been missed due to perception errors.

4. *Any-time Planning:* The planner should be capable of providing an "any-time" plan, meaning that it can be queried at any moment to obtain the current best solution. This flexibility allows us to obtain the most up-to-date solution whenever we require it, without unnecessary delays in waiting for a new solution.

5. *Incremental Update for Dynamic Obstacles:* As obstacles are discovered or move within the environment, the planned path should be incrementally updatable to accommodate these changes. This capability eliminates the need to replan from scratch and allows for efficient modification of the plan in response to dynamic obstacles.

6. *Probabilistic Completeness:* The planner should exhibit probabilistic completeness, meaning that given sufficient time to generate a solution, it should asymptotically converge to an optimal solution based on the provided optimization metric.

With these guiding principles in mind, the subsequent section will delve into the existing literature to examine how current path planning solutions either address, or fail to address these objectives. The remainder of the chapter provides a detailed exploration of the primary novel contribution of this work: a specially designed sample-based path planner, accompanied by a tracking MPC trajectory planner, that aims to fulfill the defined planning objectives.

3.2 Related Work

Path planning for local obstacle avoidance is a rich and well studied field with a wide range of approaches. In this work, we are particularly concerned with local obstacle avoidance for unmanned ground vehicles with fixed underlying reference plans, and as such limit our review of the literature to this subsection of obstacle planning. Given the current state of the robot and known positions of obstacles, we explore the methods for generating trajectories that avoid the obstacle and return to the reference plan.

One simple approach to this problem is to optimize for a solution directly at the trajectory generation level. Krusi et al. [5] propose planning robot trajectories using a random spanning tree of potential rolled-out trajectories, considering a desired reference velocity. A cost function is then used to select the best trajectory based on various characteristics. This approach is computationally inexpensive and works well for simple paths with single isolated obstacles. However, it becomes less effective for more complex trajectories, and the quality of the solution is limited by the number of samples explored. In 2016, the same team attempted to improve upon this approach by incorporating

a traditional local planner based on Rapidly Exploring Random Trees (RRT) [27] with dynamic waypoint generation. However, as we make note of previously, the use of waypoints in this form has limitations.

More recently, there has been considerable effort in exploring alternative navigation strategies beyond classical approaches. For example, Meng et al. [57] demonstrate the use of deep learning to learn high-level navigation behaviours from a dataset of example demonstrations. Their network takes in a sequence of images from a "teach" dataset and outputs a short-distance waypoint ahead of the robot based on the current observation. While not explicitly designed for obstacle avoidance, this approach naturally exhibits avoidance behaviour in addition to path following. Although promising, the path-following errors in these techniques [58–60] are currently larger than what is acceptable for precise teach-and-repeat applications.

Another approach to path following with local obstacle avoidance is presented by Liniger et al. [49], as previously introduced in Chapter 2. Drawing inspiration from their techniques, we now delve into a more comprehensive examination of their methodology. They propose a coupled path planner and MPC architecture, applied in the domain of autonomous racing. While the racing context may seem unrelated to VT&R3 objectives, there are significant similarities. In racing, there is often a pre-calculated optimal trajectory that results in the fastest lap times. Therefore, it is advantageous to develop a planner that closely follows these lines and quickly returns to the optimal line after passing competitor vehicles. Their method discretizes the track into a grid of cells and employs the dynamic programming algorithm, [61], to identify the shortest path around competing vehicles, leading back to the optimal line. From this initial solution, they define lateral corridor constraints that guide an MPC controller to find an optimal trajectory within the corridor.

As they can define this corridor as convex, this ensures that the MPC does not get stuck in local minima and they can maintain high control rates. While this approach is powerful and serves as a significant source of motivation for our method, it also has its limitations, particularly in the scalability of the planning approach. The brute force dynamic programming planner is restricted to structured scenes with a few obstacles of known fixed size, as acknowledged by the authors themselves.

This method of motion planning is based on the work of Park et al. [62] and has roots in the concept of path homotopies. A path homotopy class is a mathematical concept used in topology to classify paths based on their topological equivalence [63]. In simple terms, it groups together paths that can be continuously deformed into one another without tearing or intersecting. Two paths belong to the same homotopy class if they can be transformed into each other by smoothly deforming them while keeping their endpoints fixed. The notion of path homotopy is based on the idea that the shape or topology of a path is more important than its precise geometric details. It allows us to study the properties of paths that are preserved under continuous transformations, providing a way to compare and classify different paths based on their underlying structure. By first finding an initial solution using a planning algorithm, you are able to identify one such solution that belongs to a potentially promising homotopy class. MPC is then able to optimize within the class

to find a globally optimal solution.

The popular TEB approach [55] similarly leverages path homotopy classes to find robust planning solutions. In their implementation, the authors utilize Voronoi diagrams to span the environment and discover as many homotopy classes as possible. They then use some clever mathematical analysis based on the related concept of path homology to refine and filter these sets of paths, followed by parallel optimization to select the optimal solution. This approach offers improved scalability for real-time applications. However, it does not inherently support smooth path following without relying on waypoints and is computationally limited due to the exhaustive nature of Voronoi diagram generation. Bhattacharya et al. [64] perform a similar exhaustive homotopy class search, instead using the graph-based search algorithm A* [65] to improve the scalability.

Unlike the previous architectures, we elect to use a sample-based planner to identify strong candidate paths that avoid obstacles and combine this result with MPC. The first is a direct tracking MPC approach where the output of the planner is used as a reference trajectory for the cost function. The second approach is to implement a homotopy guided MPC with corridor constraints defined by the current planner solution. These approaches have some important advantages compared to the methods presented to this point. Firstly, sample-based planners such as Rapidly Exploring Random Trees (RRT*) [66], Fast Marching Trees (FMT) [67], Probabilistic Roadmaps (PRM) [68], or D* [69], can generate solutions more tractably in large unstructured environments. Unlike deterministic search algorithms that exhaustively explore the configuration space, sample-based planners take a probabilistic approach by sampling random points in the space. This sampling strategy allows them to efficiently explore the configuration space without being constrained by its size or complexity [70]. By randomly sampling configurations, these planners can effectively navigate through complex and high-dimensional spaces, searching for feasible paths that connect the start and goal states. The probabilistic nature of the sampling ensures that the planner explores a diverse set of configurations, covering a wide range of possibilities. This random exploration helps in avoiding local optima and discovering feasible paths even in challenging and cluttered environments.

We are not the first to try to combine sample-based planners with MPC. Zhou et al. [71] run a variation of the sample-based planner Informed RRT* [72] to generate shortest-distance paths avoiding obstacles, and opt to track this path directly with MPC under the assumption that it will be collision free. Similar to this work, Al-Moadhen et al. [73] uses Batch Informed Trees (BIT*) as the sample-based motion planner with the addition of a B-Spline smoothing post-processing step to generate kinematically feasible paths to be tracked with MPC.

A fundamental difference between previous works and this research lies in the design of the sample-based path planner. By introducing specific modifications to the planner's fundamental structure, including the incorporation of a specialized configuration space and a novel edge cost metric that prioritizes minimizing lateral deviation from a reference path, we demonstrate the ability to customize this architecture to best align with the path-following problem structure. This tailored approach allows us to effectively leverage the available prior information in VT&R3 and effectively fulfill the proposed planner principles.

The comparison table in Fig. 3.2 provides a comprehensive overview of how the different local obstacle-avoidance methodologies discussed thus far align with the requirements of our planning problem. It highlights the particular niche that our work aims to address and illustrates the distinctive contribution we bring to this field.

Papers \ Principles	1. Immediate Obstacle-free Solutions	2. Preservation of Reference Path	3. Balance Path Length and Lateral Deviation	4. Any-time Global Planning	5. Incremental Update for Dyamic Obstacles	6. Probabilistic Completeness
Adaptive Route Following [5]	✗	✗	✓	✗	✗	✗
Driving on Point Clouds [27]	✓	✗	✗	✓	✗	✓
Learning Composable Behavior Embeddings [59]	✗	✓	✗	✓	✓	✗
Optimization-Based Autonomous Racing [49]	✓	✗	✓	✗	✗	✓
Timed Elastic Band [57]	✓	✗	✗	✓	✓	✗
Hybrid Obstacle Avoidance [51]	✗	✗	✗	✓	✓	✓
Superting BIT*-Based Planning [53]	✗	✗	✗	✓	✓	✓
Path Homotopy Invariants [65]	✓	✓	✗	✗	✗	✓
Along Similar Lines (Ours) [93]	✓	✓	✓	✓	✓	✓

Figure 3.2: A comparison table illustrating how other researchers' local obstacle solutions address the proposed planning principles in Chapter 3.1. Our approach, [51], is the only method identified that meets all of proposed principles.

3.3 Methodology

3.3.1 Assumptions

Before describing our path planning method, it is important to state our assumptions about the environment and discuss their importance.

Assumption 1 *A maximum corridor with place-dependant width is specified at all points along the taught reference path that constrains the size of the planning domain.*

Assumption 1 is a critical aspect that significantly influences the path-planning process. By having access to a predefined maximum corridor with a width that varies based on the location along the taught reference path, we can effectively limit the search space when generating planning solutions. This constraint is particularly important in the teach-and-repeat context, where the generated local plans should closely follow the taught path, typically deviating by approximately 5 m or

less due to the potential for localization failures beyond this range. Moreover, in certain scenarios, it is advantageous to impose a manual, place-dependant corridor along the taught path. This additional constraint allows us to account for non-traversable terrain that may not be detectable through our obstacle perception system, such as the edge of a cliff, or a nearby body of water. By incorporating this constraint, we can further restrict the planner's solutions to consider only feasible paths within the cleared terrain, ensuring safe and reliable navigation during the autonomous traversal.

For applications where this information is not directly available and is less relevant for the operational domain, this value can instead suitably be specified by a constant corridor width as a user defined parameter.

Assumption 2 *The environment can be approximated by local planar surfaces and a projected 2D obstacle map representation of the local environment is available for reliable collision checking.*

Assumption 2 is based on the understanding that our primary focus in this work lies in path planning rather than perception. We assume that we have access to accurate 2D OGM (Occupancy Grid Maps) that project the local 3D obstacle structures detected by LiDAR onto the vehicle's frame. These maps divide the space into occupied and obstacle-free cells that can be queried by the planner to perform collision checks.

In our application, we make a trade-off by utilizing the simpler 2D OGM representation instead of more computationally expensive 3D collision checking methods. Since our objective is to identify and avoid obstacles in close proximity to the robot along the taught path, this 2D representation suffices. Although this assumption may appear restrictive, we demonstrate that even in highly unstructured non-planar operational environments for UGVs, the manageable local perception errors allow for reliable collision checking.

Assumption 3 *The robot's state estimate is accurate and available to query at any given moment.*

Assumption 3 builds upon Assumption 1, assuming that as long as the robot remains within the safe place-dependant corridor, the planner will consistently have access to an accurate estimate of the robot's current state when interacting with the navigation system. This is accomplished by utilizing the last known state of the robot with respect to the localization system and extrapolating the state forward in time using a constant velocity model, if necessary.

In practice, within the VT&R3 framework, the localization system provides frequent updates at the rate of the LiDAR (often 10 Hz or better), reducing the need for extensive extrapolation. By relying on this assumption, we can ensure that the planner operates based on reliable and up-to-date information about the robot's position and velocity, allowing for precise and accurate path planning.

3.3.2 Batch Informed Trees

While the selection of path planner is arbitrary for the application of our extensions, we employ BIT* [74] as our baseline planner. BIT* is probabilistically complete, asymptotically optimal, and can be adapted to re-plan or *rewire* itself in the presence of moving obstacles similar to the Lifelong Planning A* (LPA*) algorithm [75], making it an ideal candidate for our current and future applications.

BIT* performs as follows. A Random Geometric Graph (RGG) with *implicit* edges is defined by uniformly sampling the free space around a start and goal position. An *explicit* tree is constructed from the starting point to the goal using a heuristically guided search through the set of samples. Given a starting state $\mathbf{x}_{\text{start}}$ and goal state $\mathbf{x}_{\text{goal}}$, the function $\hat{f}(\mathbf{x})$ represents an admissible estimate (i.e., a lower bound) for the cost of the path from $\mathbf{x}_{\text{start}}$ to $\mathbf{x}_{\text{goal}}$, constrained through $\mathbf{x} \in \text{X}$.

Admissible estimates of the cost-to-come and cost-to-go to a state $\mathbf{x} \in \text{X}$ are given by $\hat{g}(\mathbf{x})$ and $\hat{h}(\mathbf{x})$, respectively, such that $\hat{f}(\mathbf{x}) = \hat{g}(\mathbf{x}) + \hat{h}(\mathbf{x})$. Similarly, an admissible estimate for the cost of creating an edge between states $\mathbf{x}, \mathbf{y} \in \text{X}$ is given by $\hat{c}(\mathbf{x}, \mathbf{y})$. Together, BIT* uses these heuristics to process and filter the samples in the queue based on their ability to improve the current path solution. The tree only stores collision-free edges and continues to expand until either a solution is found or the samples are depleted.

A new batch begins by adding more samples to construct a denser RGG. Had a valid solution been found in the previous batch, the samples added are limited to the subproblem that could contain a better solution. Given an initial solution cost, c_{best}, we can define a subset of states, $X_{\hat{f}} := \{\mathbf{x} \in \text{X} \,|\, \hat{f}(\mathbf{x}) \leq c_{\text{best}}\}$ that have the possibility of improving the solution. When the metric for the edge-cost computation is Euclidean distance, the region defined by $\hat{f}(\mathbf{x}) \leq c_{\text{best}}$ is that of a hyperellipsoid with transverse diameter c_{best}, conjugate diameter $\sqrt{{c_{\text{best}}}^2 + {c_{\text{min}}}^2}$, and focii at $\mathbf{x}_{\text{start}}$ and $\mathbf{x}_{\text{goal}}$ [72].

A high level outline of the BIT* algorithm taken from the original work is shown in 3.3 for context.

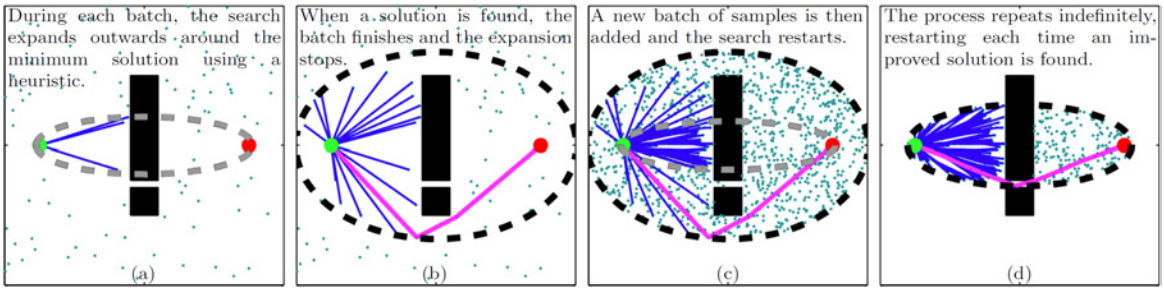


Figure 3.3: An illustration of the informed search procedure used by BIT* [74]. Start and goal states are shown as green and red dots, respectively. The current batch path solution is shown in magenta and is used to define the geometry of the black dashed heuristic ellipse that constrains the sampling volume of the subproblem in each subsequent batch.

The original implementation of BIT* is designed for shortest-distance point-to-point planning and is not customized for path following. In the remaining sections, we describe two modifications to adapt a generic sample-based planner for the desired path-following structure outlined in Chapter 1.

3.3.3 The Curvilinear Configuration Space

Our first extension adds natural path-following by using an orthogonal curvilinear planning domain [76]. A reference path is composed of a set of discrete 3 Degree of Freedom (DOF) poses $P = \{\mathbf{x}_{\text{start}}, \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{\text{goal}}\}$ with $\mathbf{x} = (x, y, \psi)$ describing the Euclidean position and yaw. We define a curvilinear coordinate, (p, q), representation of the path such that the p-axis, $p \in [0, p_{\text{len}}]$ describes the longitudinal distance along the path, and the q-axis, $q \in [q_{\min}, q_{\max}]$ is the lateral distance perpendicular to each point p on the path. $q_{\min}$ and $q_{\max}$ describe the lateral place-dependant bounds of the curvilinear space at each segment of the path.

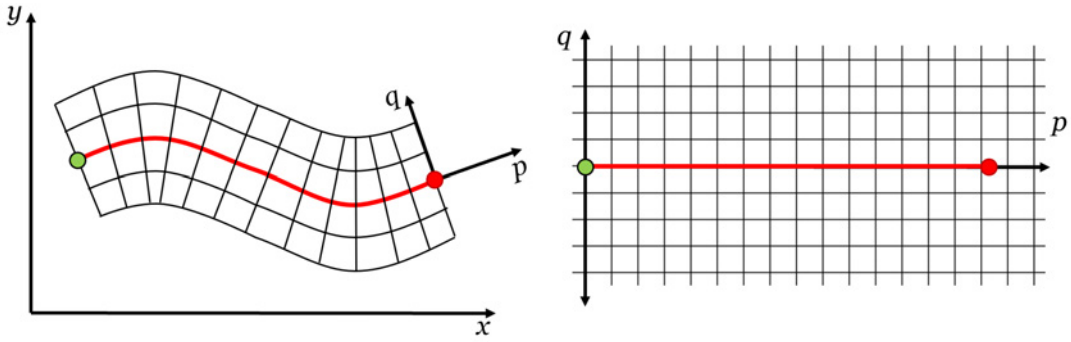


Figure 3.4: Left: A reference path in Euclidean coordinates shown in red, with the longitudinal and lateral components extended in a grid. Right: The corresponding representation of the path in curvilinear coordinates.

A change in distance between subsequent poses, Δp, is computed as

$$\Delta p = \sqrt{\Delta x^2 + \Delta y^2 + a\Delta\psi^2}. \tag{3.1}$$

An aggregated p value is stored for each discrete pose in a pre-processing step up to the total length of the path, p_{len}. It is important to note that as part of (3.1), we incorporate a small term for changes in yaw along the repeat path tuned by a constant parameter a. This allows us to avoid singularities in the curvilinear coordinate space in the event of rotations on the spot by distinguishing between poses with identical positions but changing orientations. This insight combined with our strategy for handling singularities proves to be a major distinction between our method and other curvilinear coordinate planner implementations [77–79]

A key observation from this definition is that all paths in Euclidean space become straight lines in (p, q) space, automatically handling paths that self-intersect without requiring waypoints. By storing the p values associated with each Euclidean pose from the reference path, we can uniquely

map an arbitrary curvilinear point (p_i, q_i) to its corresponding Euclidean point (x_i, y_i) by interpolating to find a pose on the reference path closest to the target point and applying some basic trigonometry.

While a unique map always exists from (p, q) to Euclidean space, generally the reverse can not be guaranteed due to singularities. This proves to be problematic when considering the collision checking of obstacles. Our solution is intuitive: we run BIT* in (p, q) space as normal, and perform all collision checks in Euclidean space by discretizing edges and mapping the individual points back to Euclidean space for query with the obstacle costmaps.

The full process for collision checking and converting points between the curvilinear planning domain and Euclidean space is shown in Fig. 3.5

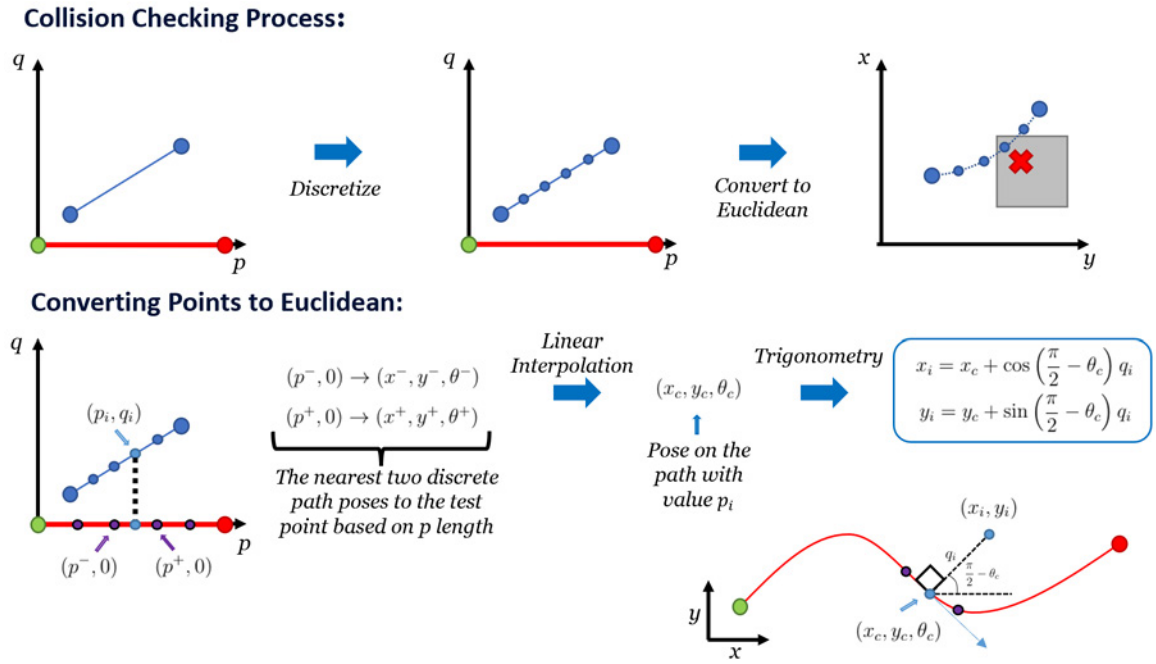


Figure 3.5: An illustration of the proposed collision checking scheme using curvilinear coordinates. Given an edge in curvilinear space, we can perform a collision check by finely discretizing the line into points and converting each point to Euclidean space using some basic interpolation and trigonometry. We then perform a check to see if any of the points are located inside the current obstacle map, and return a boolean that determines if the edge should be created in curvilinear space.

After planning a successful path in (p, q) space, we use this same unique map to convert the plan back to Euclidean space for tracking with the controller. It is worth noting that this process improves the *smoothness* of our solutions compared to Euclidean planners, as straight line edges in the curvilinear planner are inherently mapped back to curves in Euclidean space.

Using curvilinear coordinates for the planner configuration space provides several important advantages for path-following applications. It is obvious that the shortest-distance path solution in (p, q) space will be the horizontal line connecting $(0, 0)$ to $(p_{\text{goal}}, 0)$. Consequently, this solution is equivalent to the nominal teach path and we can exploit this to satisfy Principle 1). As we know the form of this nominal solution in advance we can generate a small subset of pre-seeded samples in

BIT* during each batch on the p-axis. This has the effect of accelerating the BIT* convergence, as if there are no obstacles, BIT* will immediately find the nominal solution without having to process the random samples in the space.

Second, as we plan longitudinally along the reference path, it is impossible to ever plan a path that shortcuts the reference, thus satisfying Principle 2).

A final positive by-product of the curvilinear configuration space is the fact that straight lines in (p, q) space generally convert to smooth curves in Euclidean space. This implies that that after converting the BIT* result to Euclidean space, we will have a relatively smooth path to serve as the target for a tracking controller.

In the following subsection we explore how we can exploit the simplified geometry of the curvilinear coordinate planning domain to derive a more sophisticated laterally weighted edge-cost metric for BIT*.

3.3.4 Laterally Weighted Batch Informed Trees

Consider the 2D Euclidean planning problem where the teach path is composed of a path connecting $\mathbf{x}_{\text{start}}$ to $\mathbf{x}_{\text{goal}}$. The usual cost of an edge connecting two arbitrary points in space (x_1, y_1) to (x_2, y_2) can be expressed generally as the length c_{21}:

$$c_{21} = \int_{x_1}^{x_2} \sqrt{1 + \left(\frac{dy}{dx}\right)^2}\, dx. \tag{3.2}$$

For our work, we incorporate an additional coefficient such that the cost of an edge increases as the lateral y deviation over the length of the edge grows, scaled by a tuning parameter α:

$$c_{21} = \int_{x_1}^{x_2} (1 + \alpha y^2) \sqrt{1 + \left(\frac{dy}{dx}\right)^2}\, dx. \tag{3.3}$$

For a straight line, this integral becomes

$$c_{21} = \left(1 + \frac{\alpha(y_2^3 - y_1^3)}{3(y_2 - y_1)}\right) \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}. \tag{3.4}$$

As Δy approaches zero (horizontal edges) we have

$$\lim_{\Delta y \to 0} \left(1 + \frac{\alpha(y_2^3 - y_1^3)}{3(y_2 - y_1)}\right) \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} = (1 + \alpha y^2)|x_2 - x_1|, \tag{3.5}$$

where $y_1 = y_2 = y$.

In (3.4) we obtain the Euclidean distance metric scaled by a coefficient to apply a penalty for lateral path deviation. This edge-cost metric plays a significant role in shaping the output of path planners, particularly in the context of path following. To illustrate its impact, we consider the scenario depicted in Fig. 3.6. In this example, the robot begins deviated from the intended path

while attempting to follow the horizontal red reference line. Two distinct strategies emerge as potential extremes.

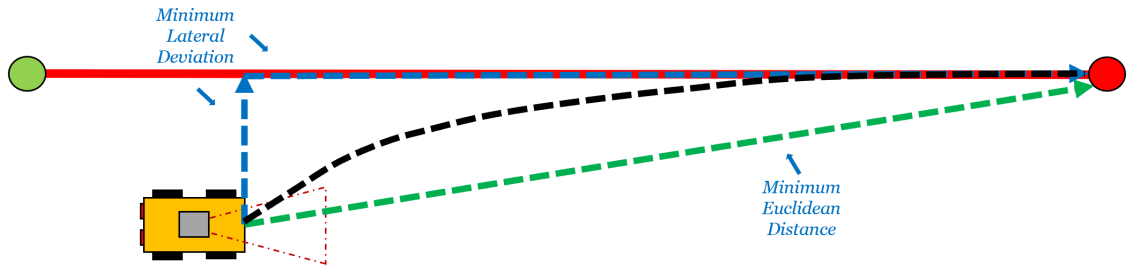


Figure 3.6: A comparison of path-following strategies given a reference path to follow (red). One low energy approach is to follow the line of shortest Euclidean distance (green). A more aggressive, but potentially safer strategy is to minimize the cumulative lateral deviation from the reference (blue). In practice, we argue that a hybrid approach is most desirable (black).

The first strategy (green) involves the robot taking the shortest Euclidean path to reach the final goal quickly and efficiently. However, we contend that this option is undesirable due to the considerable distance and time spent far away from the intended path. Alternatively, the second strategy (blue) aims to minimize lateral path deviation immediately by executing two abrupt rotations to swiftly realign with the reference path. Nevertheless, pursuing this extreme approach may lead to excessive rotational maneuvers and compromise efficiency.

We argue that the best solution lies between these two extremes. By fine-tuning the parameter α in (3.4), we can effectively adjust the planner's behaviour to strike a better balance (black). When $\alpha \to 0$, the Euclidean distance metric dominates. Conversely, as $\alpha \to \infty$, the focus shifts to pure lateral minimization. In our practical experiments, we have found that setting α to 0.5 strikes a desirable middle ground, encapsulating the benefits of both strategies.

While this edge-cost metric as defined works in this simple example of a straight-line path, the result is difficult to generalize when considering arbitrarily complex reference paths in Euclidean space. In curvilinear space, however, all reference paths become horizontal lines on the p-axis, allowing us to directly apply this idea for the edge-cost metric in BIT* by letting $y_i \to q_i$ and $x_i \to p_i$, respectively, in (3.4).

Before using this new metric in BIT*, we must first evaluate the influence of the edge-cost to the informed sampling region that constrains the RGG sub-problem following an initial solution. Consider the estimated total cost, $\hat{f}(\mathbf{x}) = \hat{g}(\mathbf{x}) + \hat{h}(\mathbf{x})$, to incorporate an arbitrary sample, $\mathbf{x} = (p, q)$, into the path solution in curvilinear coordinates as in Fig. 3.7.

The cost is

$$\hat{f}(\mathbf{x}) = \left(1 + \frac{\alpha}{3} q^2\right) \left(\sqrt{(p - p_{\text{start}})^2 + q^2} + \sqrt{(p - p_{\text{goal}})^2 + q^2} \right). \tag{3.6}$$

If $\mathbf{x}$ is to improve the quality of a current solution cost, c_{best}, we require that $\hat{f}(\mathbf{x}) \leq c_{\text{best}}$. Rear-

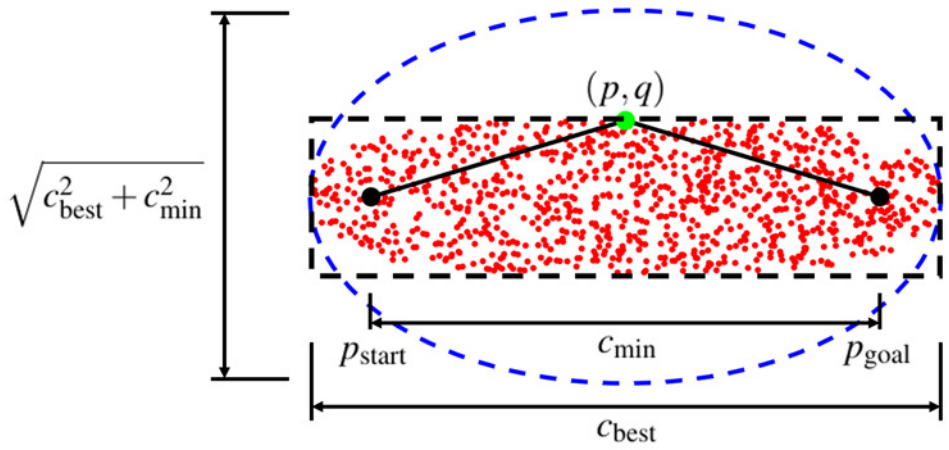


Figure 3.7: The informed sampling domain, $X_{\hat{f}}$, for the Euclidean distance edge-cost metric (blue ellipse), and the conservatively bounded laterally weighted edge-cost metric (black box) for $\alpha = 0.5$. Samples shown in red, were populated using rejection sampling and illustrate the true 'eye-ball' distribution of the informed sampling region. As α tends to zero, the domains coincide.

ranging the inequality, we have

$$\left(\sqrt{(p - p_{\text{start}})^2 + q^2} + \sqrt{(p - p_{\text{goal}})^2 + q^2}\right) \leq \frac{c_{\text{best}}}{\left(1 + \frac{\alpha}{3} q^2\right)} \leq c_{\text{best}}. \tag{3.7}$$

We note that the lateral scaling factor is always ≥ 1 for all q and that once again the left-hand term is simply the Euclidean distance edge-cost metric. This result implies that, conservatively, we could sample from within the informed ellipse defined by the usual Euclidean edge cost (no lateral penalty) as in [72], and the probabilistic path convergence guarantees will remain satisfied.

On the other hand, (3.7) indicates that the lateral penalty causes our true informed sampling region to become a denser subset within the original ellipse. While difficult to describe geometrically, we can visualize this region by randomly sampling from within the outer ellipse and rejecting points with heuristic costs, $\hat{f}(\mathbf{x})$, larger than the bounds. As shown in red in Fig. 3.7, we see the laterally weighted informed sampling region takes the shape of an 'eye-ball' and for this problem has significantly smaller volume than the Euclidean distance ellipse.

In practice, direct sampling from the eye-ball region is difficult and rejection sampling can be inefficient. However, it is possible to calculate the height of a conservative rectangle to bound the true informed sampling region and perform direct sampling from within the bounding box. Sampling from the smaller region increases the likelihood of generating sample points that improve the current solution in each BIT* batch, thereby enhancing the planner's convergence rate compared to Euclidean BIT*. Additionally, paths generated using this method naturally prioritize reducing lateral deviation from the reference path during obstacle-avoidance scenarios.

By considering a sample on the boundary at the mid-point between the start and goal, $p_{\text{eye}} = \frac{p_{\text{start}} + p_{\text{goal}}}{2}$, we can compute the maximum height of the eye-ball, q_{eye}, by exploiting the fact that

the cost-to-come to this point, $\hat{g}(\mathbf{x}_{\text{eye}})$, is exactly equal to $\frac{c_{\text{best}}}{2}$:

$$\left(\frac{c_{\text{best}}}{2}\right)^2 = \left(1 + \frac{\alpha}{3} q_{\text{eye}}^2\right)^2 \left(\left(\frac{p_{\text{start}} + p_{\text{goal}}}{2}\right)^2 + q_{\text{eye}}^2\right). \tag{3.8}$$

At the end of each batch of BIT*, we apply Pythagoras' theorem solving (3.8) for q_{eye}, to conservatively approximate the informed sampling region for our weighted lateral edge-cost metric as the encapsulating rectangular bounding box shown in Fig. 3.7. We then perform direct sampling from this bounding box in our BIT* implementation and the algorithm proceeds as normal.

3.3.5 Teach & Repeat Integration

While the core of the planner is presented in Chapters 3.3.2 to 3.3.4, in this section we discuss a number of design modifications we make to both improve performance and tailor the planning algorithm for our specific VT&R3 application.

In a typical path-following problem set-up, you may only have access to a loosely constrained sparse set of key points along the reference path that you are trying to follow. This is sufficient to generate the curvilinear space and apply the methods presented in this work. In VT&R3 however, we have access to a dense collection of Euclidean poses that anchor our reference path at regular intervals and make the curvilinear configuration space well defined.

In an obstacle-free environment, it is true that BIT* will converge to the taught path in curvilinear space in a relatively short amount of time. The limiting factor is the generation of random continuous samples in the configuration space that fall exactly on the $q = 0$ axis. We can expedite this process significantly by initializing BIT* with a number of pre-populated samples on this axis. This allows BIT* to immediately find the optimal lowest cost solution that follows the taught path exactly during the first batch in a matter of milliseconds, even for paths of significant length (in excess of kilometers).

This modification also drives our second integration change. In the presence of obstacles, our focus and detection capability are primarily limited to the robot's immediate surroundings. Therefore, sampling the configuration space within a sliding window, spanning from the robot's current state to the upcoming path, is sufficient for finding obstacle collisions. By exploiting this insight, we can leverage the pre-seeded samples to promptly generate solutions for the entire path, while accounting for local obstacles. Consequently, the number of samples required to find a solution is significantly reduced.

Our final modification takes advantage of the sliding window implementation by adopting a more efficient planning direction: from the end of the path to the current robot state, rather than the conventional approach of planning from the robot towards the goal. This decision is based on the observation that the section of the path between the end of the path and the edge of our sliding window typically represents the largest portion of the planning tree. When re-planning becomes necessary due to the appearance of new obstacles, we can save valuable time by restoring only the relevant segments of the tree within the sliding window.

3.3.6 Tree Repair

During the planning process, it is common for the current plan to become invalid as new obstacles are discovered or existing obstacles move within the environment. In such cases, we are faced with two options: either restart the planning from scratch or *rewire* the existing solution to navigate around the new obstacles while preserving the processed tree to avoid redundant computations. In the interest of computational efficiency, we choose the latter approach. The full tree-repair process is illustrated in Fig. 3.8 for reference, and is subsequently explained in detail.

At the end of each planning batch, we have a spanning tree composed of interconnected vertices and edges distributed throughout the configuration space. This tree also includes a lowest cost path of adjoining vertices leading from the end of the path to the current state of the robot in curvilinear coordinates. To validate the solution, we convert this path to Euclidean space and perform collision checks through the connected branch of vertices using the process described in Chapter 3.3.3. If the path encounters no collisions with the observed local obstacles, we store the result for use in the navigation system. However, if a collision occurs, we take note of the cost-to-come value of the vertex immediately preceding the edge that leads to the collision.

Next, we proceed to prune the tree by removing any vertices and edges with larger cost-to-come values. These branches can no longer guarantee valid paths due to the presence of obstacles and their stored cost-to-come values now may be incorrect. Subsequently, we resample the configuration space using the default maximum admissible sampling heuristic region (i.e, the maximum corridor) since we no longer have an upper bound on the current solution cost and need to consider all possible options once again.

Although we lose the portion of the planning tree between the obstacle and the robot, we are able to maintain the portion of the tree between the end of the path and the local obstacle collision point. This segment generally represents the bulk of the tree and serves as one of the motivating factors for why we elect to plan from the end of path to the robot state.

3.4 Curvilinear Singularities and Corner Cases

Throughout this work, we have been operating under the assumption that converting continuous paths in curvilinear coordinates to Euclidean space is a seamless process, free from singularities and discontinuities. In the vast majority of scenarios, this assumption holds true. However, there are situations, particularly during sharp inside turns, where certain regions in the curvilinear configuration space can introduce problematic singularities, challenging the validity of our assumption. In this section, we delve into a detailed exploration of how we can adapt our planning algorithm to address this issue effectively.

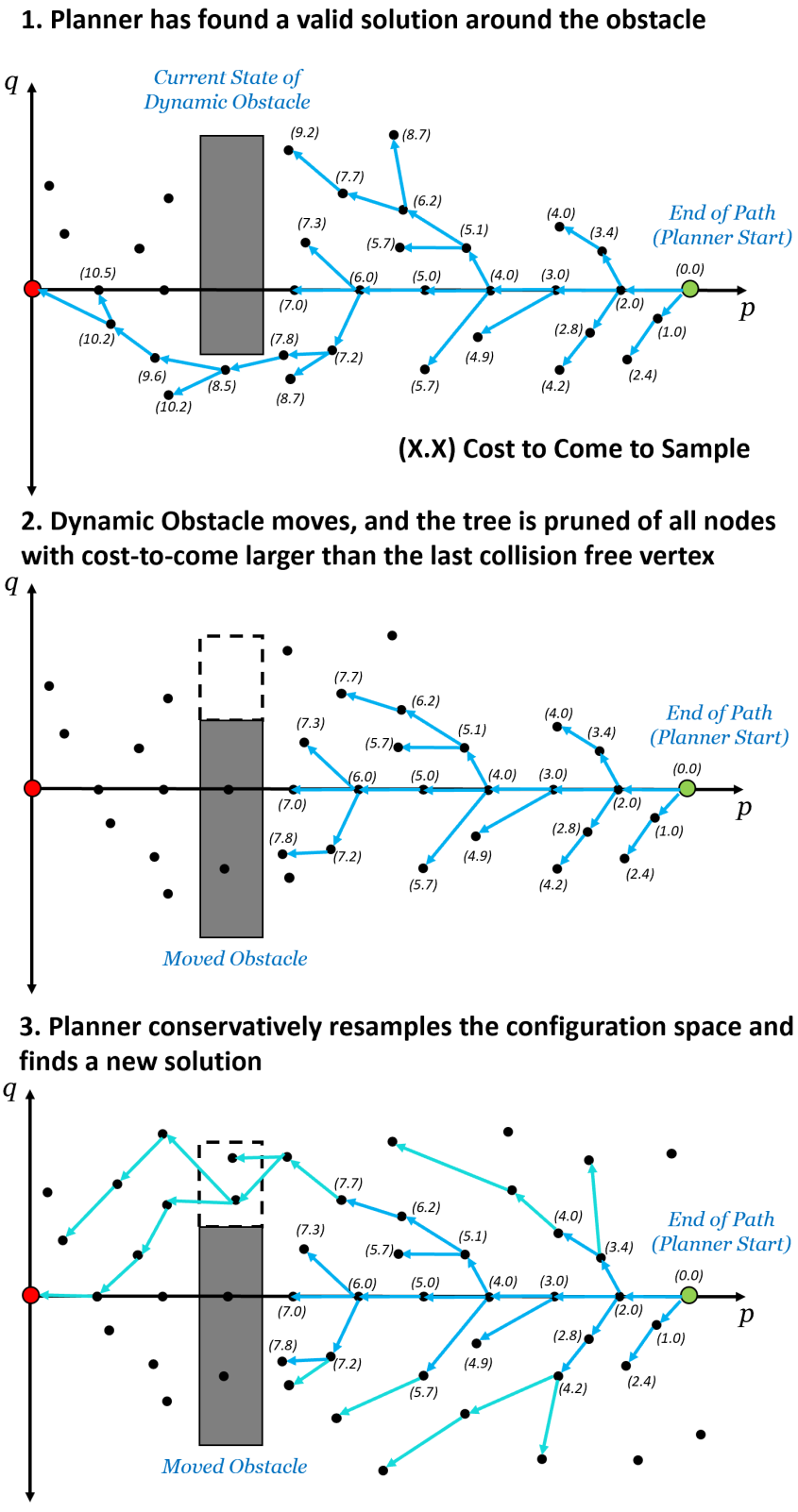
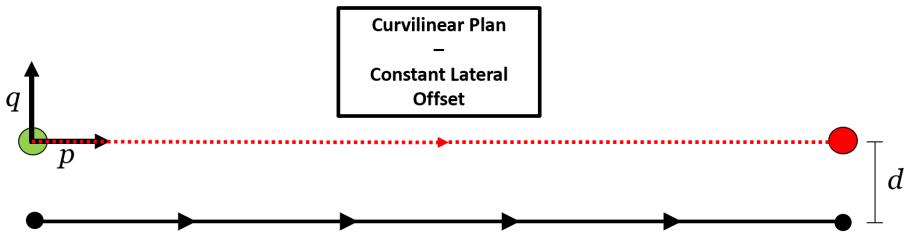


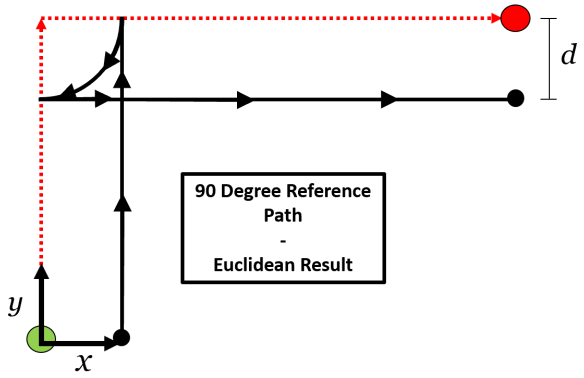
Figure 3.8: In this figure we illustrate the tree-repair process allowing us to handle moving obstacles.

3.4.1 Singularity Regions

To better understand the singularity problem, let us consider the toy problem depicted in Figure 3.9. Imagine a reference path in Euclidean space that consists of a straight line path that takes a sharp 90-degree turn to the right halfway through. In curvilinear representation, this path becomes a straight horizontal line. Now, if we attempt to follow a line with a fixed lateral offset from this initial path in curvilinear space and convert it back to Euclidean space, we will end up with the irregular path shown in Fig. 3.9b.



(a) Curvilinear plan with constant offset.



(b) Corresponding Euclidean reference path and planner result after conversion to Euclidean space.

Figure 3.9: In this figure we illustrate a toy problem to help conceptualize the curvilinear space singularity issue. Consider a red Euclidean reference path with a sharp 90 degree turn and its curvilinear straight line representation. In curvilinear space, if we wished to follow the reference path with a constant lateral offset of d [m] we simply generate the black horizontal path solution in (a). After being converted to Euclidean space in (b), we find that taking this path results in an undesirable three-point turning behaviour.

While this path is technically valid, such behaviour is generally not desirable. The occurrence of this "three-point-turn" behaviour arises when the planned path in BIT* traverses the curvilinear space with a non-zero lateral q component over regions where the reference Euclidean path exhibits excessive inside curvature. A common scenario where this can occur is when an obstacle is placed near the inside corner of a sharp turn, as illustrated in Fig. 3.10.

Although the occurrence of excessive curvature is typically manually avoided by the operator during the teach phase of VT&R3 due to the wear imposed on a large differential drive robot and struggles with localization accuracy, it is still important to address this issue comprehensively. Our proposed solution involves identifying these problematic regions during an offline pre-computation

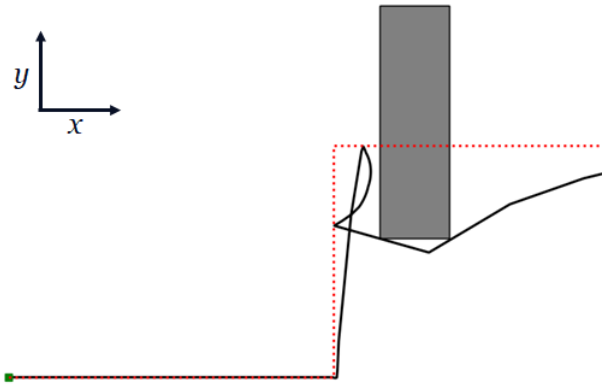


Figure 3.10: In this scenario, we have another sharply turning reference path. The presence of the obstacle near to the corner necessitates the planner to cross a similar singularity region as was demonstrated in Fig. 3.9, resulting in the odd path result shown in black.

step. Then, during runtime, we modify our planning algorithm to naturally avoid these regions.

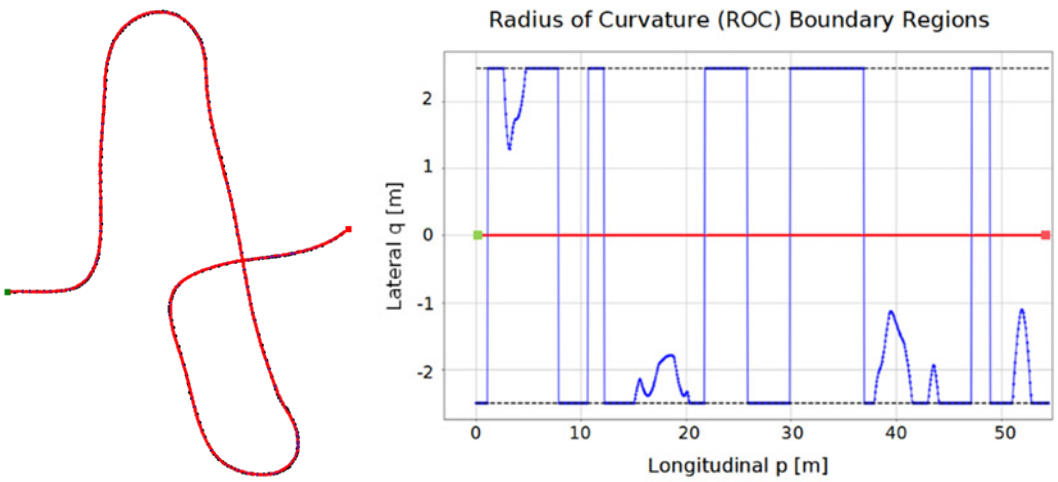
To define the singularity regions, we consider an arbitrary point in curvilinear space, denoted as (p, q), with the corresponding projected Euclidean space point as (x_p, y_p). The singularity regions can be described by the inequality

$$q_{\max}(p) \geq q > q_{\min}, \tag{3.9}$$

where $q_{\max}(p)$ represents the maximum lateral bounds of the corridor defined by the reference path at each point p, and $q_{\min}$ is defined as the distance to the nearest point on the reference path from the Euclidean point (x_p, y_q). In other words, a curvilinear point falls within a singularity region if, when converted to Euclidean space, the q value does not match the distance to the closest Euclidean point on the reference path.

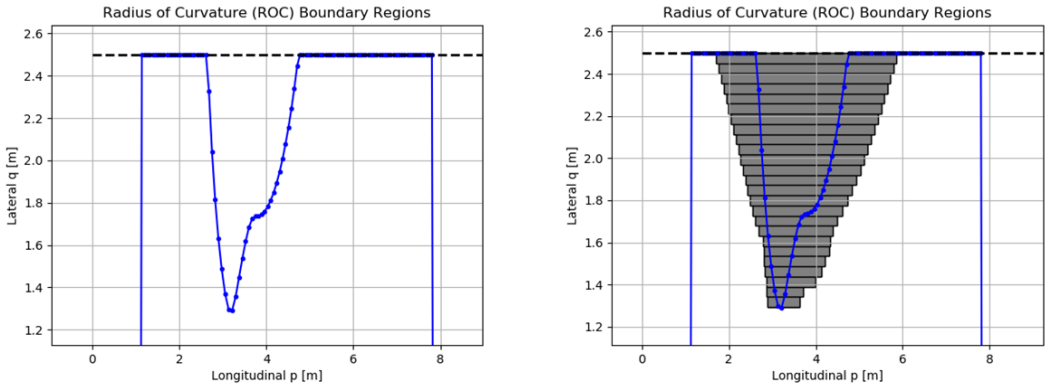
While defining the singularity regions is a straightforward task, discretizing the configuration space to compute the interior boundary (where $q = q_{\min}$) is neither practical nor elegant. Instead, we can rely on another observation: these regions are closely linked to the curvature of the reference path, specifically the instantaneous Radius of Curvature (ROC). Our key insight is to notice that all points where $q > \mathrm{ROC}$ also satisfy the equality in equation (3.9). Although the ROC does not fully describe the singularity regions, it provides a promising starting point for our search. By pre-computing the ROC at each point p along the path using the reference path, we can expand along lines of constant lateral distance and test for condition $q = q_{\min}$ to be satisfied.

A typical plot depicting the ROC of a reference path is illustrated in Fig. 3.11, and Fig. 3.12 demonstrates how to generate the corresponding singularity region. To ensure tractability, we discretize the lateral and longitudinal directions with a resolution of 10 cm and conservatively approximate the singularity regions using a series of adjoining rectangles. We do not bother calculating the singularity regions outside of the maximum lateral boundary for our reference path as we never intend to plan in these regions. It is worth observing that for segments of the path with constant ROC, the singularity regions form symmetric trapezoids.



(a) Euclidean reference path. (b) Curvlinear path representation with instantaneous radius of curvature overlay.

Figure 3.11: In this figure we show an intricate reference path in (a) and its corresponding curvilinear representation in (b). At each point along the path, we compute an instantaneous ROC, plotting the result in blue. Note that we clamp the ROC magnitudes at the maximum corridor bounds, $\pm q_{\max}$ for readability.



(a) Instantaneous ROC in (p, q) space. (b) The resulting singularity boundary region.

Figure 3.12: In this figure we illustrate the generation of the curvilinear singularity regions. Using the ROC plot as a starting pot, we descretize the space laterally and expand from the ROC plot to the left and right along the p-axis, testing until condition (3.9) is invalidated. Eventually, we are left with the singularity region shown in (b).

3.4.2 Wormhole Generation

Now that we have a clear understanding of the singularity regions and how they appear in the planner, it is crucial to discuss how we can effectively avoid them. One possible approach would be to treat these regions as obstacles and prevent the planner from generating paths that pass through them. While this approach does work, it would eliminate the option of taking the inside corner to circumvent an obstacle on a sharp turn, as shown in Figure 3.10. However, it is clear in this example that taking the inside corner could still be the best choice. Therefore, solely relying on this approach should be avoided if possible.

An important insight we can exploit is that the boundary points of the singularity regions, denoted as (p_l, q) and (p_r, q) for the left and right boundary points of q, respectively, correspond to the same Euclidean position after being converted using the process described in Chapter 3.3.3. The distinction is in their headings, which can vary significantly.

In other words, by traversing from the leftmost boundary point to the rightmost boundary point, we effectively have an edge that, when followed, executes a turn on the spot in Euclidean space. It is worth noting that the intermediate points along the fictitious edge created between (p_l, q) and (p_r, q) do not exhibit this continuous turning behaviour; only the boundary pairs possess this property. However, the length of the edge is indeed proportional to the amount of rotational offset between the two corresponding Euclidean poses and can be taken into account when planning.

This realization provided the intuition that instead of planning through the singularity regions we can try to leap past them. To achieve this, we introduce the concept of *wormholes*. In this context, a wormhole refers to pairs of vertices in the curvilinear planning domain that, when connected by the planning tree, enable *teleportation* across the singularity regions. Similar to pre-seeding the reference path with samples along the $q = 0$ axis, we can also pre-populate several wormhole edges that serve as passageways, opening up the possibility of traversing inside corners during planning. An illustration of the pre-seeded wormholes for the previous reference path problem is depicted in Fig. 3.13 and we illustrate a planning example using wormholes in Fig. 3.14.

We do not wish to allow the planner to use these means of transporting the configuration space for free, however, as we recall that taking a path through a wormhole results in a plan that turns on the spot; a generally undesirable feature. As such we can impose a cost for these edges that is proportional to the amount of rotation that would be incurred and a tunable parameter that allows the user to adjust how willing they want the planner to be to take these passage ways. If a small turn on the spot behaviour yields a significant path cost reduction compared to a long outside corner path, then in some cases it may be acceptable to allow this behaviour.

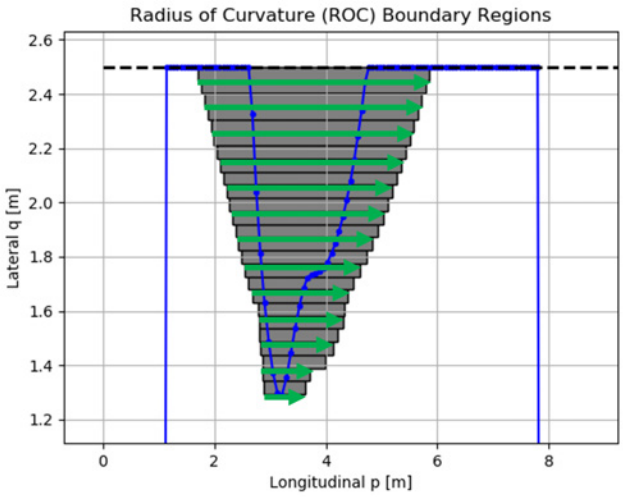


Figure 3.13: Curvilinear singularity regions with pre-seeded wormhole edges shown in green. The BIT* planner is able to use these edges to find valid path solutions, however is not allowed to create new edges within the regions.

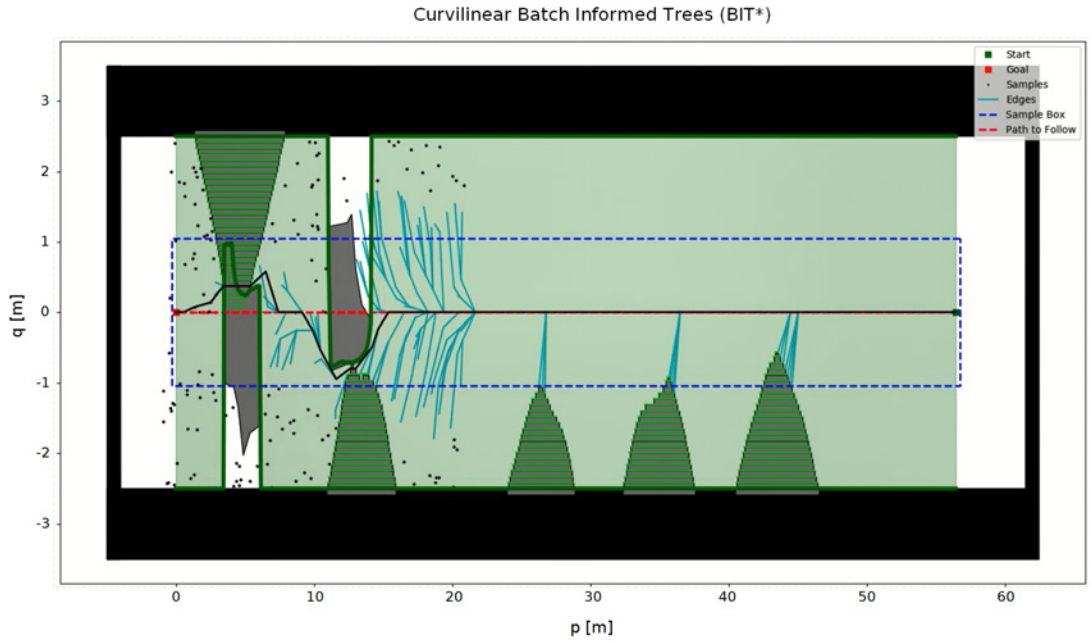
3.5 Evaluation

3.5.1 Evaluation Strategy and Metrics

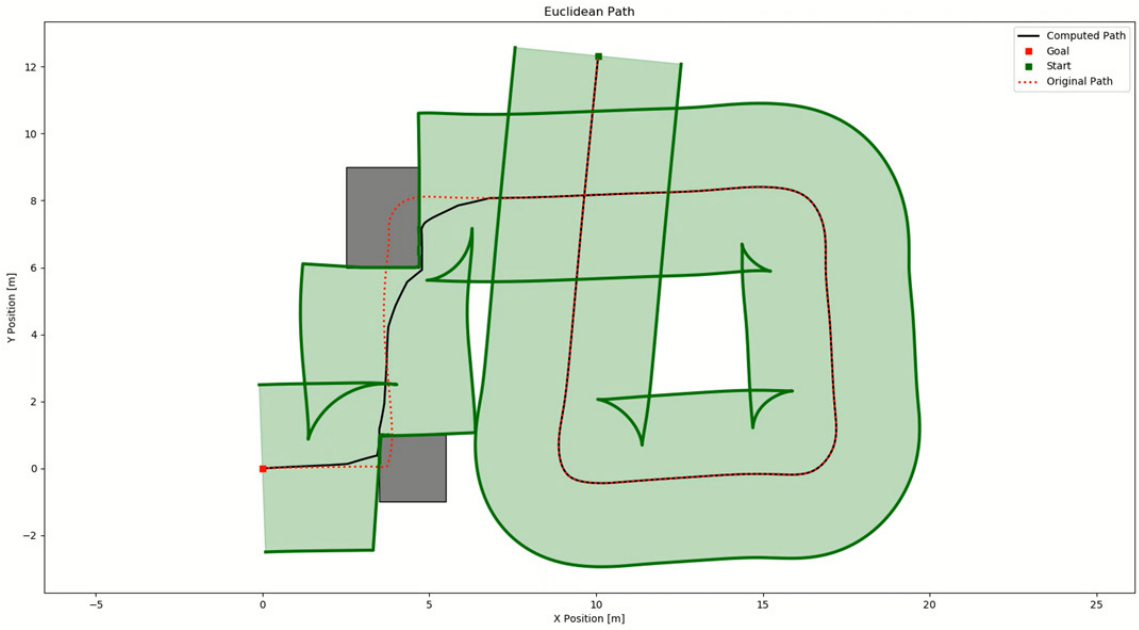
In teach-and-repeat applications, it is critical that when avoiding obstacles we limit lateral path error to best exploit the prior knowledge on terrain assessment. It is also important to maintain a similar sensor viewing angle throughout the trajectory for localization purposes. With these characteristics in mind, we propose two metrics to compare the relative quality of the path solutions produced over the course of our experiments. We compute the Root Mean Square Error (RMSE) for both the translation (lateral) and rotation (heading) components relative to the reference path over 15 m segments of obstacle interactions and average the result across all trials. In the context of this study, obstacle interactions refer to any event where a newly appeared obstacle (as identified by terrain change detection) causes the robot to deviate from the original reference path to avoid a collision.

The planner's path convergence time is an additional crucial aspect to consider. Sample-based planners often depict this relationship by plotting the normalized solution cost against the computation time allocated to the problem. For an online local planning system, three specific target points from this plot hold interest:

1. *Time to Initial Solution:* This refers to the time required for a single batch of BIT* to be completed, acting as a measure for how swiftly the system can respond to environmental changes, such as the sudden appearance of an obstacle around a previously hidden corner or the movement of a dynamic obstacle. While the solution may be coarse at this stage, the ability to rapidly locate this starting point is crucial for maintaining effective control behaviour.



(a) Lateral Batch Informed Trees with wormholes.



(b) Corresponding Euclidean reference path and output plan.

Figure 3.14: In this figure we show a representative planning scenario using wormholes to find better path solutions. The obstacles, (grey), in Euclidean space were added manually in their distorted form to the (p, q) space for visualization purposes only. We find that the use of wormholes allows us to generate paths that tightly hug obstacles, opting for turns on the spot to further reduce lateral path deviation. These turns on the spot allow us to generate safe-corridor constraints, (green), that the controller can use to smooth out the trajectory while maintaining collision avoidance properties.

2. *Time to 97% Convergence:* The time to 97% convergence represents the duration needed to reach a solution cost that is within 97% of the optimal solution cost. In practical terms, we approximate the upper bound cost by evaluating the solution generated by the planner after running for 100,000 samples. This metric provides insight into how quickly the planner can approach a nearly optimal solution, allowing for efficient trajectory planning.

3. *Time for Correct Homotopy Class Identification:* The time to identify the correct path homotopy class signifies the duration required to generate a path solution that belongs to the same homotopy class as the one defined by the converged path solution.

The latter criterion has more nuanced value, and a more detailed discussion of homotopy class theory is provided in Chapter 4 due to its relevance in the second of our MPC formulations. In short, the time to identify the correct path homotopy class can be considered the time it takes to find the globally optimal collision avoidance trajectory when applied to our architecture. For our purposes, it represents a measure of the typical motion planning lag for the online implementation.

We demonstrate the benefits of the lateral edge-cost metric by testing BIT* on several representative obstacle-avoidance problems, both with and without the proposed extensions. For fair comparison, we implemented our own C++ version of BIT* in accordance with [74], using parameters of 150 samples per batch and an RGG constant of 1.1. Our extended implementation, Lateral BIT*, uses the laterally weighted edge-cost metric (3.4) with $\alpha = 0.5$ and the rectangular approximation of the informed sampling region. All experiments were conducted on a Lenovo Thinkpad laptop with 11th Gen Intel(R) Core(TM) i7-11800H @ 2.3GHz. For the standard BIT* implementation, we can force the use of a pure Euclidean distance edge-cost metric by simply adjusting $\alpha = 0$, and switch to using the ellipsoidal sampling region described in [72]. Online robot experiments with the entire local obstacle-avoidance system are provided in Chapters 4 and 5.

3.5.2 Results

To study the influence of the new edge-cost metric in isolation, we posed a series of 10 simulated planning problems where the reference path is composed of a horizontal line 15 m in length on the x-axis, making the curvilinear and Euclidean path representations identical. We allow both versions of the algorithm to find converged path solutions connecting the starting pose to the goal and evaluate the solution both quantitatively and qualitatively. A representative example of the output paths and associated BIT* trees on one test problem is shown in Fig. 3.15.

In analyzing Fig. 3.15, we see some important differences in the exploration strategies of the two methods. Using our lateral edge-cost metric, BIT* tends to naturally generate smoothly curving solutions, spending the most time exploring paths near to the reference while balancing forward progress with reducing lateral error. In contrast, the standard BIT* algorithm settles on the direct shortest-distance solution. While efficient, in practice this path could be a higher-risk manoeuvre due to the additional localization and terrain-assessment uncertainty incurred when away from the

reference path. We calculate the RMSE metrics for both implementations and summarize the results in Table 3.1.

As we would expect, the use of our laterally weighted edge-cost metric considerably reduces the average lateral error with respect to the reference path from 25.50 cm to 9.83 cm. We also see a small improvement on the heading error, likely due to the fact the lateral planner tends to spend more time exactly following the reference path.

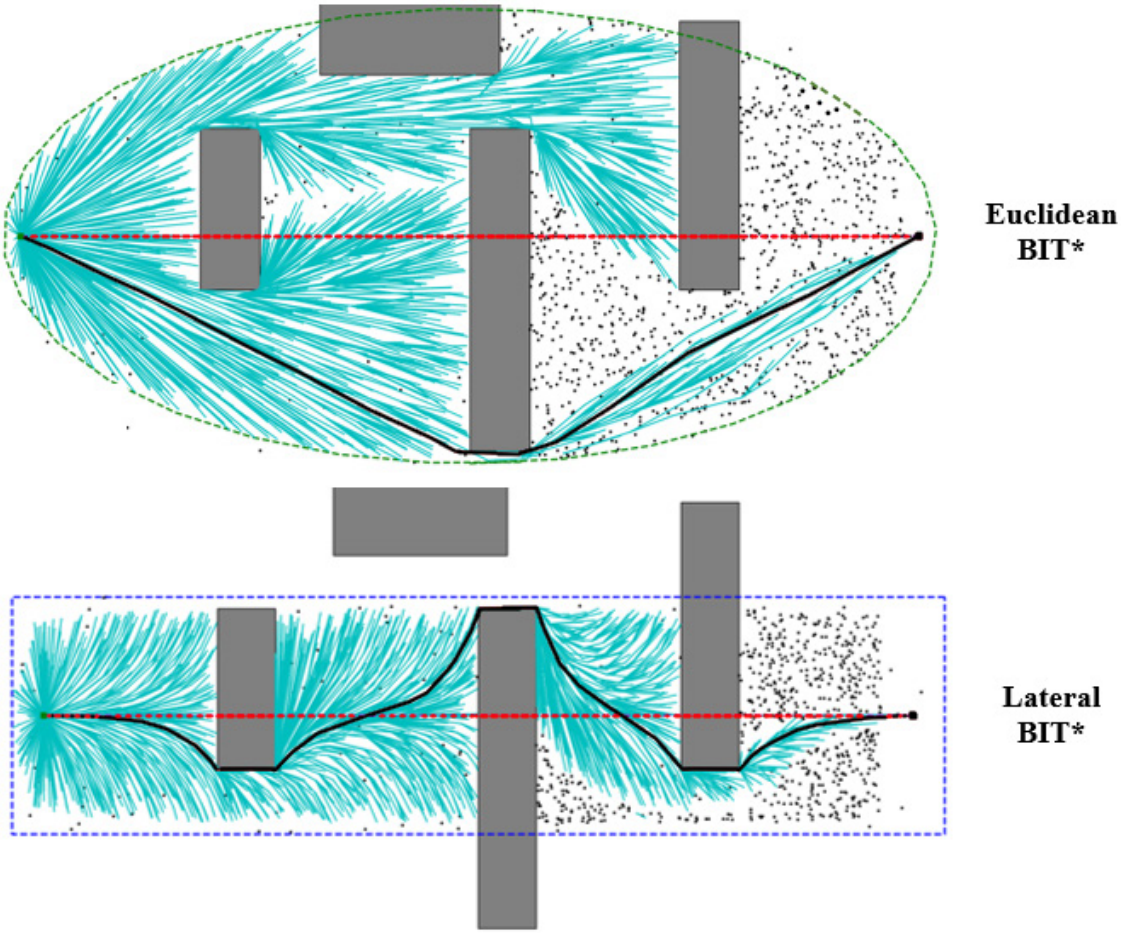


Figure 3.15: A comparison of the planning trees generated running BIT* with the Euclidean edge-cost metric (top) and with the laterally weighted metric (bottom) in a representative test environment. Our edge-cost metric encourages the final plan, shown in black, to avoid obstacles while remaining close to the red reference path on the p-axis.

Regardless of path-following capability, real-time performance is crucial for the path planning module. In Fig. 3.16, we analyze the runtime performance and observe that our extended version of BIT* generally generates initial solutions and converges to an optimal result at a slower pace compared to the Euclidean version of BIT*. This outcome was anticipated based on the qualitative path results discussed in the previous section, where a higher number of samples per batch were required to generate the observed "weaving" plans. Intuitively, a high sample density is necessary to refine the smooth curves that are characteristic of our extensions, which leads to slower convergence.

Table 3.1: Path Planner Error Analysis For Simulation Experiments

Exp. #	Lateral RMSE [cm] (Lateral BIT*)	Lateral RMSE [cm] (Euclidean BIT*)	Heading RMSE [deg] (Lateral BIT*)	Heading RMSE [deg] (Euclidean BIT*)
1.	8.52	26.80	29.46	35.71
2.	10.11	22.18	32.95	29.61
3.	9.75	24.73	30.25	34.74
4.	9.90	31.03	25.02	31.87
5.	7.16	22.61	23.80	25.39
6.	11.24	25.10	41.31	32.73
7.	10.81	26.79	29.11	35.42
8.	9.51	24.96	30.70	31.12
9.	8.90	27.03	27.74	43.67
10.	12.37	23.72	35.78	28.90
Mean:	**9.83**	**25.50**	**30.61**	**32.92**

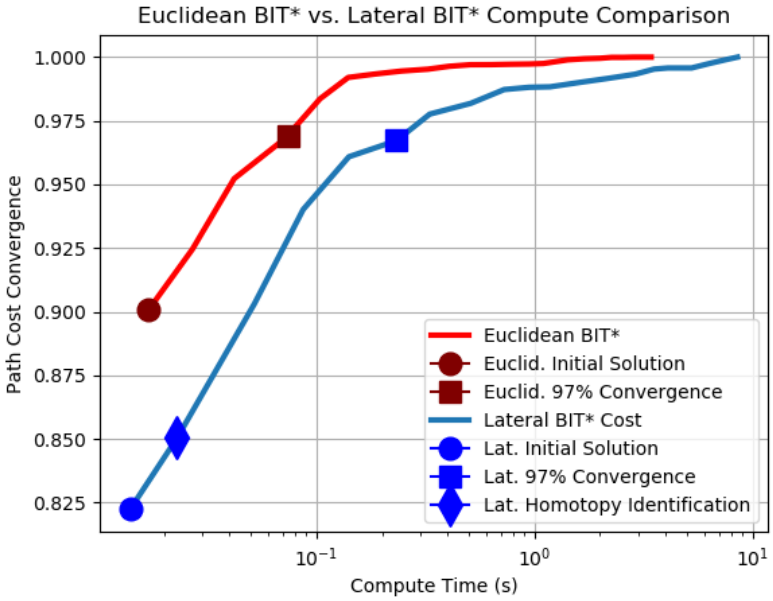


Figure 3.16: In this figure we compare the compute characteristics of the Euclidean BIT* planner to the Lateral BIT* planner. We find that generally the Euclidean edge-cost metric converges faster with marginally higher early solution quality. However, it is important to recognize that for the key compute metrics that most heavily influence the performance our architecture, the initial solution time and the homotopy class identification time, Lateral BIT* performs comparably. This is in addition to the other advantages the Lateral planner holds in terms of desirable solution characteristics.

However, it is important to highlight that the differences in compute times are relatively small. Despite this, our Lateral BIT* implementation still manages to find initial solutions to challenging planning problems in just 17ms. Furthermore, it achieves convergence to approximately 97% of the optimal solution cost in just 233ms on average, ensuring the generation of smooth output paths in a

timely manner.

The true strength of our planner lies in its average time for correct homotopy class identification. This metric effectively measures the time taken by the planner to generate a solution that weaves through obstacles and can be continuously deformed into the optimal solution without colliding with obstacles. On average, our planner is able to identify these correct homotopy solutions in just 23 ms, which is nearly three times faster than the Euclidean planner can approach a converged solution suitable for tracking. As discussed further in Chapter 4, we leverage this homotopy class solution in our second MPC implementation to generate smooth optimal trajectories for obstacle avoidance that surpass the quality of solutions produced by direct tracking of more converged plans. Consequently, our motion planner is able to exhibit remarkable responsiveness, allowing it to promptly react to dynamic obstacles without explicitly planning in a temporal dimension.

While it is easy to see the desirable output path properties the lateral edge-cost metric encourages using straight-line reference paths, we can further exploit the curvilinear coordinate space to produce similarly smooth plans on more intricate reference paths. In Fig. 3.17, we initialize the planner on a complex path taken from real teach data that includes a variety of sharp curves, a path crossing, and several difficult obstacles. Despite the challenges, our planner is able to converge to a desirable solution using only a single goal and no intermediate waypoints. As Euclidean BIT* tends to generate solutions that traverse singularity regions in curvilinear space (this subsequently produces discontinuous Euclidean paths), we are not able to directly provide a comparison to this result.

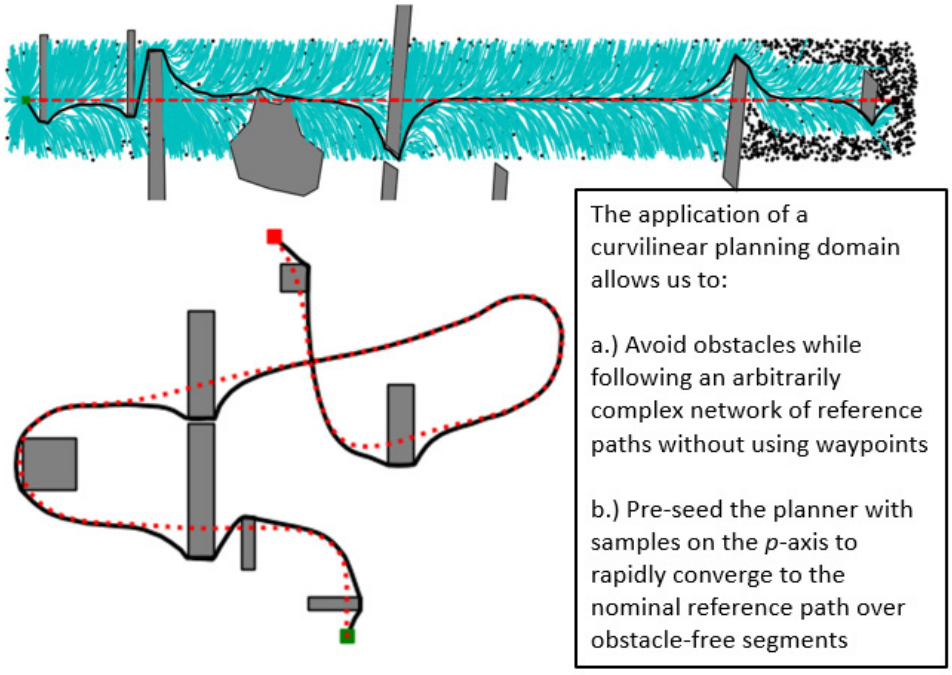


Figure 3.17: Lateral BIT* planning in a curvilinear representation of the space (top) to find an obstacle-avoidance path solution in Euclidean space (bottom) along a complex reference path with many obstacles.

3.6 Summary

In this chapter, we introduced two key advancements in sample-based planning, enabling the transformation of a conventional point-to-point planner into one optimized for path following with local obstacle avoidance. By leveraging curvilinear coordinates and a novel laterally weighted edge cost metric, our planner excels in generating smooth paths from the robot to the path's endpoint while prioritizing minimal lateral deviations from a predefined reference path. We discussed how these properties, along with other desired planner principles, align well with the requirements of robotics applications featuring a teach-and-repeat problem structure, such as VT&R3.

To demonstrate the benefits of our proposed planner, we conducted experimental comparisons against a state-of-the-art sample-based planner employing a Euclidean edge-cost metric. The results clearly indicate that our method produces more desirable path-following solutions while maintaining reasonable computation times. In the subsequent chapters, we will introduce the paired MPC for the motion planner and conduct a series of online experiments to further validate the system's efficacy and robustness.

Chapter 4

Model Predictive Control Trajectory Generation

This chapter introduces the second major contribution of this work: a trajectory planning architecture based on MPC. Building upon the path planner presented in Chapter 3, our goal is to design a controller that can effectively execute the planned path on a range of robotic platforms. By employing MPC, we aim to complete an autonomous repeat objective while safely avoiding local obstacles in the environment.

4.1 Introduction

With the path planner providing a potentially viable solution, our focus shifts to the physical execution of the planned trajectory. The MPC-based trajectory generation approach serves as the bridge between the abstract path representation and the actual robotic platform, ensuring that the planned path is followed accurately and robustly. By integrating control algorithms into the planning process, we can adapt the trajectory in real time, accounting for dynamic factors such as robot kinematics, actuator limitations, and robot acceleration.

MPC is a control strategy that involves solving an optimization problem repeatedly over a finite time horizon. Unlike classical control methods that rely on predefined control policies or feedback laws, MPC operates by predicting the system's future behaviour based on a dynamic model and optimizing control inputs to meet a given performance criterion. This predictive nature allows MPC to explicitly consider system constraints and adapt the control strategy in real-time, making it particularly well-suited for complex robotic tasks.

One key advantage of MPC is its ability to handle nonlinear and time-varying systems. By employing a model of the robot dynamics, MPC can accurately capture the system's behaviour and make predictions about its future states. This predictive capability enables MPC to account for dynamic changes in the environment, such as variations in obstacle positions or disturbances, and adjust the control inputs accordingly. As a result, MPC exhibits robustness and adaptability, which

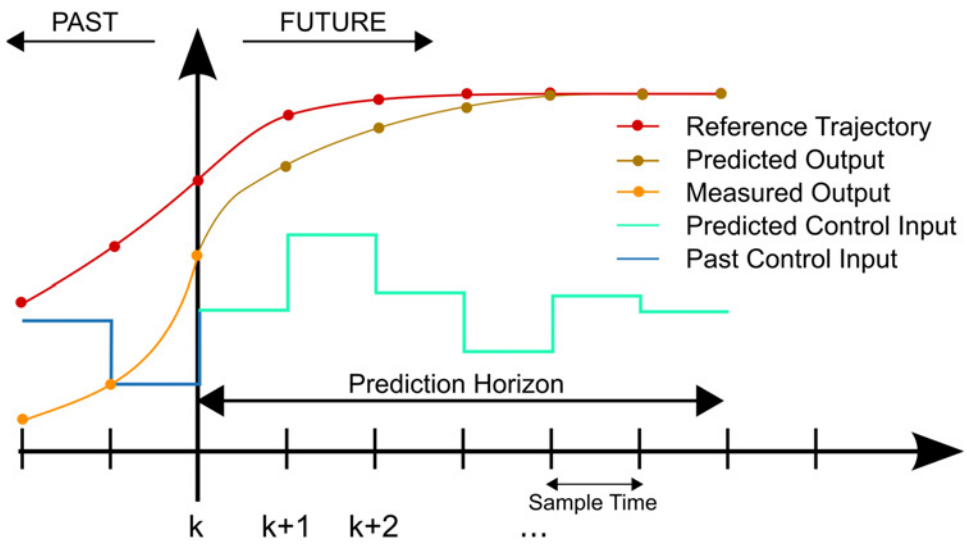


Figure 4.1: A traditional MPC scheme: given a desired reference trajectory to track(red) and a robot model, the goal is to find the optimal sequence of k control inputs (teal) that propagate the modelled state forward in time to reduce the error between the predicted output (brown) and the reference. At each new time step, we reinitialize the system using the latest state measurements (orange) and repeat the process.

are crucial for reliable and responsive robotic control. Furthermore, MPC provides a systematic framework for incorporating various objectives and constraints into the control formulation. By formulating the control problem as an optimization task, MPC can find control inputs that not only achieve accurate tracking but also optimize other performance criteria, such as minimizing energy.

Additionally, MPC offers the advantage of inherent feedback control. As the system progresses over time, MPC solves the optimization problem repeatedly at each time step, generating updated control inputs based on the most recent system measurements. This feedback loop allows MPC to actively correct for errors and disturbances, ensuring accurate and responsive tracking performance. Compared to alternative control methods, such as Proportional-Integral-Derivative (PID) control or open-loop approaches, MPC excels in its ability to handle complex dynamics, incorporate constraints, and provide real-time adaptability. The predictive nature, flexibility, and feedback capabilities of MPC make it a preferred choice for path following and tracking tasks in various robotic applications, including autonomous navigation, mobile robotics, and unmanned aerial vehicles [80].

In the remainder of this chapter, we will delve into the details of the MPC framework and its integration with the path planner. We will explore the formulation of the cost function, constraints, and prediction models, all aimed at generating trajectories that safely avoid obstacles and safely complete the repeat task. Overall, the MPC-based trajectory generation component enhances the capabilities of the path planner, allowing us to seamlessly transition from path planning to trajectory execution. By combining the strengths of both approaches, we can achieve robust and adaptable robotic systems that can autonomously navigate complex environments and successfully accomplish repeat tasks while avoiding obstacles.

4.2 Related Work

Historically, Model Predictive Control (MPC) has been widely used in the chemical processing industry due to its ability to effectively handle multivariable control problems and accommodate external constraints with ease [81]. With advancements in computational capabilities over the past decades, the application of MPC, also known as Receding Horizon Control (RHC), has expanded to the field of mobile robotics. In its simplest form, the nonlinear kinematic model of a robot is employed to predict the robot's motion over time based on a series of discrete input velocity commands. This predicted motion is then compared to a time-varying reference trajectory to calculate an error cost function. To efficiently solve the real-time MPC problem with practical forward simulation horizons, it is common practice, as demonstrated in [82], to linearly approximate the vehicle kinematics and formulate the problem as a Quadratic Programming (QP) task.

Researchers have made significant progress in extending the basic kinematic MPC formulation to incorporate full vehicle dynamics, going beyond the scope of linearized kinematics. Dong et al. [46] argue that neglecting system dynamics may lead to inconsistent predicted trajectories compared to those executed by the control module. Their work shows that in applications involving self-driving cars operating at speeds exceeding 20 m/s, incorporating vehicle dynamics in MPC yields superior tracking capability compared to using kinematics alone. Similar studies [83] have explored this analysis for non-holonomic robots that track trajectories at lower speeds more typical of off-road navigation. The research findings indicate that for velocities below 2.5 m/s, using kinematic models alone leads to only marginal tracking performance deterioration, suggesting that analytically and computationally efficient kinematic methods are suitable in certain applications.

When applying MPC to mobile robot control, it is crucial to consider several desirable constraints in the optimization process. Lindqvist et al. [47] utilize MPC for a six-degrees-of-freedom Unmanned Aerial Vehicle (UAV) and impose linear inequality constraints on the maximum control input range and rates of change to ensure feasible altitude stabilization within finite thrust. Furthermore, they model dynamic obstacles approaching the UAV as spheres and incorporate quadratic equality constraints to ensure that the UAV position remains outside the obstacle's vicinity. Since the problem formulation is nonlinear and non-convex, they employ the Optimization Engine (OpEn) [84] to solve the optimization. Although the algorithm can be executed in real-time on a laptop, performing the optimization on-board the UAV is not feasible.

Hence, it is often desirable to transform the MPC optimization problem into a convex form, ensuring that every local minimum is also a global minimum. Dong et al. [46] apply a similar set of control input and obstacle constraints to a self-driving car. Additionally, they include lateral path constraints to confine the vehicle's maneuvers within the boundaries of the road. Despite representing the obstacle constraints as non-linear and non-convex Euclidean distance functions, they successfully "convexify" them using a procedure that decomposes the constraints into the intersection of several subsets of smooth convex functions. By employing this method, standard convex interior point algorithms can be utilized to enforce the constraints, significantly reducing computation time compared to Artificial Potential Field methods employed by other researchers [45].

Our approach, as discussed in Chapter 3, utilizes a sample-based planner to address the challenge of solving a tractable MPC problems in obstacle-dense environments. In our initial approach, we prioritize tracking the reference path generated by the planner while enforcing kinematic constraints to ensure a smooth trajectory, similar to other existing methods [85, 86]. This implementation is straightforward and effective in avoiding obstacles when the reference path is collision-free. However, a drawback of this method is that if the reference path is not explicitly kinematically feasible, it can lead to small tracking errors that may result in collisions. Unlike the comparable work done by Xu et al. [86] that requires a post-processing step to smooth plans, our planner naturally generates smooth trajectories, mitigating some of these limitations. Nevertheless, we propose a second alternative MPC architecture that addresses this issue by utilizing the planner to generate dynamic lateral corridor state constraints. The MPC then uses these constraints to generate trajectories that guarantee obstacle avoidance, as long as the constraints are maintained. We then compare this approach with the more commonly used direct tracking method.

Both of our MPC formulations are based on matrix Lie Group theory. Similar to the implementation by Teng et al. [87] and Chang et al. [88], we represent the robot's state using the Special Euclidean Group $\mathrm{SE}(3)$, which allows us to encapsulate both the position and orientation of the robot in a single mathematical object. The $\mathrm{SE}(3)$ Lie Group has a well-defined structure that enables efficient optimization algorithms to be applied. By formulating the optimization problem directly on the SE(3) Lie Group, we can leverage specialized optimization techniques tailored to Lie groups, resulting in improved computational efficiency and more accurate solutions. Additionally, the $\mathrm{SE}(3)$ Group naturally incorporates constraints such as position bounds, orientation bounds, and velocity bounds into the control framework. Specifically, we exploit the ability of Lie Groups to generate highly generalized kinematic constraints, allowing our MPC to easily support a library of diverse robotic platforms.

Learning-based MPC has become popular in recent years, and has even previously been applied to teach-and-repeat applications. Ostafew [89, 90] developed a Learning-based Nonlinear Model Predictive Control (LB-NMPC) algorithm which can exploit the repeated path traversals to learn a hybrid kinematic and disturbance model. Disturbances are modelled using Gaussian processes, which are subsequently updated using experiences from previous trials to enhance the tracking capability. We opt not to employ learning MPC methods in our current work. These methods add an additional layer of complexity to the problem, and we aim to first establish the validity of the underlying obstacle-avoidance architecture.

The remaining sections of this chapter will describe the specific implementation details of our controllers and provide experimental evaluations.

4.3 Methodology

4.3.1 Tracking Model Predictive Control

In this section, we will delve into the mathematical formulation of our Model Predictive Control (MPC) schemes. To ensure generality and versatility across various robotic platforms, we adopt a representation that leverages matrix Lie Group theory, drawing upon established notation from Barfoot [91, 92]. This choice allows us to create an MPC framework that can seamlessly accommodate a wide range of vehicles, from 6 Degrees of Freedom (DOF) drones to differential drive unmanned ground vehicles (UGVs) and beyond. VT&R3 inherently supports navigation for diverse robot configurations. By formulating the control problem in a general manner, we enable the straightforward substitution of the robot's kinematics without the need for extensive equation refactoring or altering the underlying control solution method. This flexibility proves advantageous as we seek to apply VT&R3 to different robot platforms, catering to their specific characteristics and requirements.

Given a robot in a moving coordinate frame $\underrightarrow{\mathcal{F}}_v$ with respect to a global frame $\underrightarrow{\mathcal{F}}_i$, we define the generalized 6 DOF velocity vector ϖ in the moving frame as

$$\varpi = \begin{bmatrix} \mathbf{v}_v^{vi} \\ \boldsymbol{\omega}_v^{vi} \end{bmatrix}, \tag{4.1}$$

composed of the linear and angular velocity vectors, $\mathbf{v}$ and $\boldsymbol{\omega}$, respectively. The pose of the vehicle in the global frame is subsequently defined by the transformation matrix

$$\mathbf{T} \triangleq \mathbf{T}_{vi} = \begin{bmatrix} \mathbf{C}_{vi} & \mathbf{r}_v^{iv} \\ \mathbf{0}^T & 1 \end{bmatrix}, \tag{4.2}$$

henceforth denoted as $\mathbf{T}$. By constraining the generalized velocity in (4.1), we can enforce different robot kinematic models using a projection matrix, $\mathbf{P}$, to isolate the non-zero components of ϖ. In our specific application, we will be working with a unicycle model with only forward linear and angular yaw velocities v and ω, respectively. However, it is clear to see that by specifying a different projection matrix, we can enforce alternative kinematic constraints. We make the following substitution in terms of a lower dimension velocity vector $\mathbf{u}$, such that

$$\varpi = \begin{bmatrix} v & 0 & 0 & 0 & 0 & \omega \end{bmatrix}^T = \mathbf{P}^T \mathbf{u}, \tag{4.3}$$

where we have:

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4.4}$$

$$\mathbf{u} = \begin{bmatrix} v \\ \omega \end{bmatrix}. \tag{4.5}$$

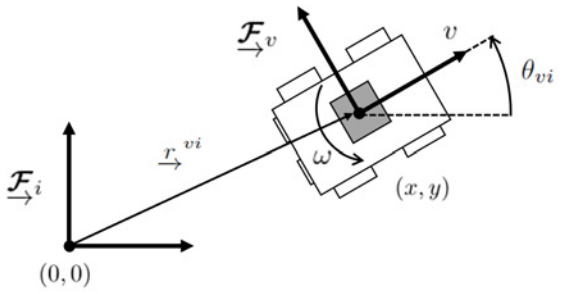


Figure 4.2: The definition of our robot state variables for a differential drive robot.

We wish to derive a model predictive controller to compute the optimal sequence of k control inputs, out to a horizon of K time steps with a period of h seconds, to minimize the error between some reference path, $\mathbf{T}_{\text{ref},k}$ and the predicted trajectory of the robot. This path tracking task can be realized by solving the least squares optimization problem:

$$\operatorname{argmin} J(\mathbf{T}, \mathbf{u}) = \sum_{k=1}^{K} \ln(\mathbf{T}_{\text{ref},k}\mathbf{T}_k^{-1})^{\vee^T} \mathbf{Q}_k \ln(\mathbf{T}_{\text{ref},k}\mathbf{T}_k^{-1})^{\vee} + \mathbf{u}_k^T \mathbf{R}_k \mathbf{u}_k \tag{4.6a}$$

s.t.

$$\mathbf{T}_{k+1} = \exp\Big((\mathbf{P}^T \mathbf{u}_k)^{\wedge} h\Big)\mathbf{T}_k, \quad k = 1, 2, \ldots, K \tag{4.6b}$$

$$\mathbf{u}_{\min,k} \leq \mathbf{u}_k \leq \mathbf{u}_{\max,k}, \quad k = 1, 2, \ldots, K. \tag{4.6c}$$

Using Lie groups and notation adopted from Barfoot [91], our objective function (4.6a) aims to minimize the pose error between the trajectory we intend to generate and a reference trajectory, while simultaneously trying to minimize control effort. Equations (4.6b) and (4.6c) are our generalized kinematic constraint and actuation limit constants, respectively.

For the direct tracking control implementation, we obtain the reference poses at each time step $\mathbf{T}_{\text{ref},k}$ directly from the current densely discretized Euclidean BIT* solution that guides the robot from its current state to the path's endpoint. To determine the reference poses along the trajectory for each time step, we interpolate poses from the current planner solution with a separation distance denoted as $p_{\text{ref},k}$. This allows us to try to maintain a nominal desired path tracking velocity v_{ref} such that

$$p_{\text{ref},k} = v_{\text{ref}} h k. \tag{4.7}$$

The parameters $\mathbf{Q}_k$ and $\mathbf{R}_k$ can be tailored to the specific application, allowing adjustment of the relative importance of different tracking degrees of freedom. For example, the rotational components of the reference poses can be down-weighted to enhance positional path-following behaviour. Likewise, $\mathbf{u}_{\min,k}$, $\mathbf{u}_{\max,k}$, and v_{ref} can be fine-tuned to suit the particular robot implementation. To generate the trajectory with the lowest cost, we solve this MPC problem over the horizon K. Following the standard MPC approach, we apply the first velocity command to the robot and subsequently

reinitialize the problem at each control loop request based on the latest available information.

This straightforward control approach is effective in practice, particularly when the reference trajectory is reasonably smooth and kinematically feasible. Our curvilinear BIT* approach proves helpful in this regard, as the conversion of the path back to Euclidean space naturally yields smooth curves. However, smoothness is not guaranteed in all cases. When the BIT* solution is not directly kinematically feasible, the controller is capable of smoothing out the resulting trajectory, albeit at the expense of introducing slight path tracking error. If this tracking error is not compensated for by inflating obstacles, it could potentially lead to collisions. We delve into this issue and its practical implications in greater detail through the course of our experimental evaluations in Chapter 5.

4.3.2 Homotopy Guided Corridor Model Predictive Control

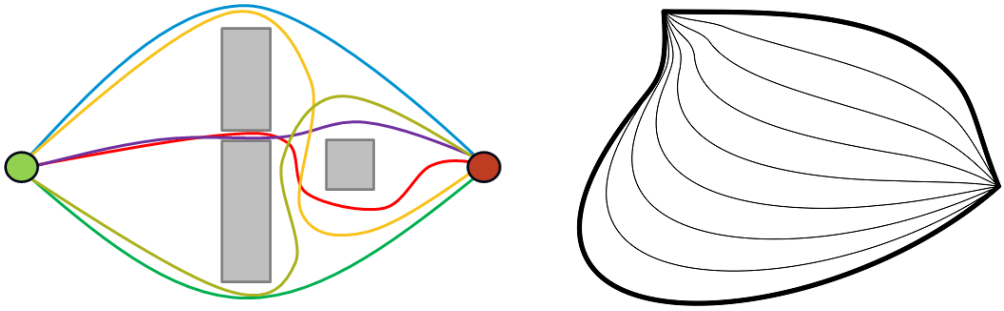
To address the limitations of the direct tracking MPC approach, we propose an alternative architecture that provides a concrete guarantee on collision avoidance. Instead of blindly following the output of the path planner, which may not always be kinematically feasible, our approach involves directly tracking the taught path that is guaranteed to be kinematically feasible. We then incorporate the planner solution into a hard spatial constraint within the MPC problem.

To understand our approach, it is crucial to discuss the concept of path homotopies. In an environment with obstacles between a robot and its goal, the presence of obstacles introduces different potential path homotopy classes. A path homotopy class represents a set of paths with common endpoints that can be continuously deformed into one another without intersecting any obstacles. In other words, paths belonging to the same homotopy class can be transformed into each other by smooth deformations while remaining obstacle-free.

The emergence of multiple homotopy classes creates local minima in the optimization problem, making it challenging to find the globally optimal solution. Each homotopy class is representative of a multitude of feasible paths through the obstacle-filled environment. However, not all paths within a homotopy class are equally desirable in terms of optimality or safety. Some paths may have shorter distances or smoother trajectories, while others may involve sharp turns.

To tackle this challenge, we draw inspiration from the work of Linigar et al. [49]. We utilize the planner's solution to identify the most promising homotopy class among the set of possible paths. By considering the characteristics of the selected homotopy class, we define a series of lateral corridor path constraints that shape the MPC trajectory planning within a convex planning space. These lateral corridors serve as spatial constraints that guide the MPC toward a globally optimal solution for the trajectory planning problem.

By initializing the new MPC problem with the current path solution and optimizing the trajectory within the constrained convex space, we achieve three significant advantages. Firstly, by utilizing the collision-free BIT* solution to generate the corridor constraints, we decouple the MPC from the planner and ensure collision avoidance with a guarantee. As the MPC controller operates within these corridors, which effectively limit its motion to safe regions of the environment. Secondly, this approach allows us to work with rough initial solutions generated by the planner to



(a) Distinctive homotopy class samples. (b) Span of paths in a homotopy class.

Figure 4.3: In this figure we provide some examples to illustrate how homotopy classes are defined. In (a), we show six representative path solutions that each belong to unique homotopy classes due to the presence of obstacles. In (b) we show a subset of a distinct homotopy class that includes any and all paths with common endpoints that can be continuously deformed into one another.

define the corridors, rather than relying on well converged planner solutions for precise obstacle avoidance. As a result, our homotopy guided corridor MPC method has the potential to operate safely at higher robot velocities compared to the direct tracking approach. Lastly, as the reference poses are always selected from the originally taught path instead of the path planner solution, it is expected that the path-following error of the taught path will be improved marginally.

The revised MPC optimization problem is similar to Equation (4.6a), but now includes an additional lateral corridor constraint:

$$\operatorname{argmin} J(\mathbf{T},\mathbf{u}) = \sum_{k=1}^{K} \ln(\mathbf{T}_{\mathrm{ref},k}\mathbf{T}_k^{-1})^{\vee^T}\mathbf{Q}_k \ln(\mathbf{T}_{\mathrm{ref},k}\mathbf{T}_k^{-1})^{\vee} + \mathbf{u}_k^T\mathbf{R}_k\mathbf{u}_k \tag{4.8a}$$

s.t.

$$\mathbf{T}_{k+1} = \exp\Big((\mathbf{P}^T\mathbf{u}_k)^{\wedge} h\Big)\mathbf{T}_k, \quad k = 1,2,\ldots,K \tag{4.8b}$$

$$\mathbf{u}_{\min,k} \leq \mathbf{u}_k \leq \mathbf{u}_{\max,k}, \quad k = 1,2,\ldots,K \tag{4.8c}$$

$$-\mathbf{d}_{k,l} \leq \mathbf{1}_2^T\mathbf{T}_{\mathrm{ref},k}\mathbf{T}_k^{-1}\mathbf{1}_4 \leq \mathbf{d}_{k,r}, \quad k = 1,2,\ldots,K \tag{4.8d}$$

Here, $\mathbf{1}_i$ represents the i-th column of the identity matrix, with $\mathbf{d}_{k,l}$ and $\mathbf{d}_{k,r}$ denoting the maximum allowable lateral deviation at each point on the left and right sides of the path that define the path homotopy corridor. The values of $\mathbf{d}_{k,l}$ and $\mathbf{d}_{k,r}$ are computed on-demand by the controller and the process is outlined as follows:

Upon obtaining the curvilinear space representation of the desired reference pose from the latest BIT* solution, $(p_{\mathrm{ref},k}, q_{\mathrm{ref},k})$, in curvilinear coordinates, we construct two test edges extending from $(p_{\mathrm{ref},k}, q_{\mathrm{ref},k})$ to the maximum safe-corridor boundaries $(p_{\mathrm{ref},k}, q_{\max})$ and $(p_{\mathrm{ref},k}, -q_{\max})$, where $q_{\max}$ is the place-dependant maximum lateral boundary set by the user at each point along the reference path. We then perform collision checks on these edges using our standard procedure and set $\mathbf{d}_{k,l}$ and $\mathbf{d}_{k,r}$ equal to the lateral component of the last collision-free vertex along the discretized

test edges. This process is illustrated in Fig. 4.4.

Just as in the previous implementation, we adaptively select the reference poses in accordance with a desired nominal robot repeating velocity; however, now the reference poses, $\mathbf{T}_{\mathrm{ref},k}$, are selected directly from the teach path instead of the BIT* solution.

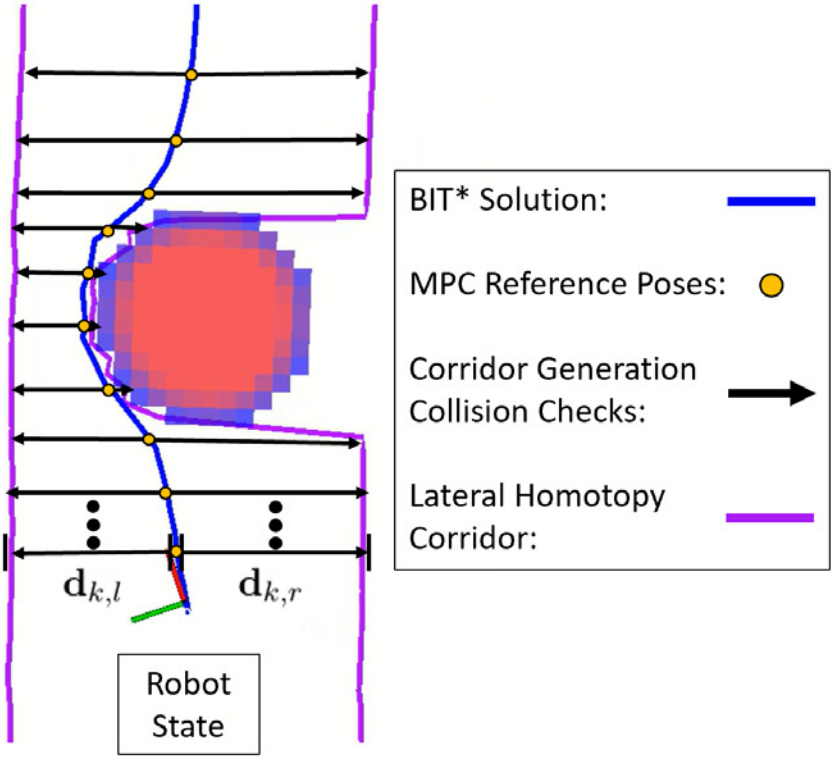


Figure 4.4: We generate a dynamic safe lateral corridor constraint (purple) using the current planner solution (blue). Given reference poses along the path, we collision check laterally in both directions, starting from the planner solution to the maximum place-dependant corridor bounds. If there is no collisions, the corridor is set to the maximum bounds. If there is a collision, the corridor is set to the vertex immediately preceeding the collision point. We repeat this process for all reference poses and for each control loop.

4.3.3 Solution Method

Given the optimization problem defined in Equation (4.8a), our next step is to devise a method for effectively enforcing the series of box constraints related to the actuator limits and lateral corridor. To address this, we employ a technique known as the Barrier Method [93].

Using the Barrier Method, we can capture the effects of constraints of the general form $\mathbf{x} \leq \mathbf{a}$ and $\mathbf{x} \geq \mathbf{a}$ as a series of logarithmic penalty terms. This transformation allows us to convert the constrained optimization problem into an unconstrained problem that can be solved efficiently. The

squared logarithmic barrier function for an arbitrary inequality constraint is written as

$$\mathbf{x} \leq \mathbf{a} \rightarrow \beta \sum_{i=1}^{\dim(\mathbf{x})} \ln^2(\mathbf{a}_i - \mathbf{x}_i), \tag{4.9}$$

$$\mathbf{x} \geq \mathbf{a} \rightarrow \beta \sum_{i=1}^{\dim(\mathbf{x})} \ln^2(\mathbf{x}_i - \mathbf{a}_i), \tag{4.10}$$

where β is a tunable scalar parameter to weight the influence of the barrier function. As the value of β decreases, the barrier function approaches zero for all states $\mathbf{x}_i$ that satisfy the inequality constraint. However, the barrier function rapidly tends towards infinity as the state variables approach the constraint boundary. By adjusting the value of β, we can control the trade-off between satisfying the constraints and optimizing the objective function.

In practice, it is common to employ an iterative reweighting scheme to fine-tune the barrier parameters. Initially, a large value is assigned to the barrier parameters to heavily penalize violations of the constraints. The optimization problem is then solved iteratively, gradually reducing the barrier parameters in each iteration. This iterative process benefits from warm-starting, where the solution from the previous iteration is used as the initial guess for the current iteration. By carefully reducing the barrier parameters and avoiding abrupt changes, the optimization process converges to an optimal solution that balances constraint satisfaction and objective function optimization.

To apply Equations (4.9), and (4.10) to the velocity and lateral path constraints, we define the vectors $\mathbf{\Phi}_{\text{vel}}(\mathbf{u})$ and $\mathbf{\Phi}_{\text{lat}}(\mathbf{y})$ to augment the original cost function as

$$\mathbf{\Phi}_{\text{vel}}(\mathbf{u}) = \begin{bmatrix} \beta \ln(\mathbf{u}_{\max,1} - \mathbf{u}_1) \\ \vdots \\ \beta \ln(\mathbf{u}_{\max,2K} - \mathbf{u}_{2K}) \\ \beta \ln(\mathbf{u}_1 - \mathbf{u}_{\min,1}) \\ \vdots \\ \beta \ln(\mathbf{u}_{2K} - \mathbf{u}_{\min,2K}) \end{bmatrix}, \quad \mathbf{\Phi}_{\text{lat}}(\mathbf{y}) = \begin{bmatrix} \beta \ln(\mathbf{d}_{1,r} - \mathbf{y}_1) \\ \vdots \\ \beta \ln(\mathbf{d}_{K,r} - \mathbf{y}_K) \\ \beta \ln(\mathbf{y}_1 + \mathbf{d}_{1,l}) \\ \vdots \\ \beta \ln(\mathbf{y}_K + \mathbf{d}_{K,l}) \end{bmatrix}. \tag{4.11}$$

We define $\mathbf{V}$ and $\mathbf{W}$ as diagonal weighting matrices, so we may write the augmented version of the cost function (4.8a) in the lifted form:

$$J(\mathbf{T}, \mathbf{u}, \mathbf{y}) = \sum_{k=1}^{K} \ln(\mathbf{T}_{\text{ref},k}\mathbf{T}_k^{-1})^{\vee^T} \mathbf{Q}_k \ln(\mathbf{T}_{\text{ref},k}\mathbf{T}_k^{-1})^{\vee} + \mathbf{u}_k^T \mathbf{R}_k \mathbf{u}_k \tag{4.12a}$$

$$+ \mathbf{\Phi}_{\text{vel}}(\mathbf{u})^T \mathbf{V} \mathbf{\Phi}_{\text{vel}}(\mathbf{u}) + \mathbf{\Phi}_{\text{lat}}(\mathbf{y})^T \mathbf{W} \mathbf{\Phi}_{\text{lat}}(\mathbf{y})$$

s.t.

$$\mathbf{T}_{k+1} = \exp\Big((\mathbf{P}^T \mathbf{u}_k)^{\wedge} h\Big) \mathbf{T}_k, \quad k = 1, 2, \ldots, K. \tag{4.12b}$$

Problem (4.12a) can be solved efficiently with a Gauss-Newton method by linearizing the problem at an operating point and applying small perturbations to the Lie Algebra to simplify terms. In practice, we solve the MPC problem directly using the Simultaneous Trajectory Estimation and Mapping (STEAM) engine [92], an iterative Gauss-Newton-style optimization library aimed at solving batch nonlinear optimization problems involving both Lie Group and Continuous-time components. STEAM is by no means the fastest way of solving these types of problems; however, it is convenient to work with for problems modelled in this form and allows us to maintain suitable control rates in excess of 30 Hz.

4.3.4 High-level Control and Speed Scheduling

In addition to the core MPC driving our motion planner, we incorporate an additional module to enhance the high-level behaviours of the controller. In the control problem as previously described, the MPC will attempt to follow the taught reference path while deviating from the path to avoid local obstacles as we desire. Nominally, the robot will repeat these paths at the user configured set-point velocity, v_{ref}, but using a constant reference velocity can introduce problems.

One challenge arises when the robot needs to navigate sharply turning paths. In such cases, maintaining the reference velocity while minimizing path tracking error becomes difficult. Moreover, when encountering curbs or challenging terrain, adhering to a fixed speed may cause large and abrupt movements for heavy and rigid vehicles. Additionally, while deviating from the path to avoid obstacles during a repeat, it is often advantageous to slow down to provide the planner and controller with more time to find optimal solutions and execute intricate trajectories.

To tackle these issues, we have developed an external module that dynamically adjusts the current set-point velocity (v_{ref}) in the controller. This module utilizes a range of criteria to determine the appropriate speed adjustments. By incorporating this speed scheduling component, we can adaptively modify the robot's velocity based on the specific requirements and challenges encountered during operation. This allows for smoother navigation around tight turns, more controlled movements over uneven terrain, and enhanced obstacle avoidance-capabilities.

Motivated by the success of an early VT&R speed scheduling implementation [9], our module operates by integrating several path-dependant criteria, each proposing individual adjustments to the initial user-provided velocity set-point. These candidate set-points are then evaluated, and the scheduler conservatively selects the lowest velocity among them. Several useful speed scheduling criteria have been proposed, including:

1. *Planar Curvature*: This criterion regulates the speed at a given point on the path based on the curvature of the upcoming trajectory, κ_{planar}. Higher curvatures correspond to lower desired speeds, allowing for smoother navigation around sharp turns.

2. *Profile Curvature*: In addition to considering curvature in the ground plane, it is important to account for curvature in the vertical plane, denoted $\kappa_{\mathrm{profile}}$. This criterion addresses rough terrain, bumps, and slopes, allowing the robot to adjust its speed accordingly.

3. *Proximity to the End of Path*: When approaching the end of the path, it becomes desirable to gradually reduce the vehicle's speed. This adjustment allows the robot to navigate more precisely, allowing for accurate parking or reaching a specific location. In our implementation, the end of path variable, d_{eop} is equal to 1.0 when the robot is within 5 m from the end of the path, and 0.0 otherwise.

4. *Proximity to the Taught Path*: As the robot deviates laterally from the taught path, terrain and localization uncertainties tend to increase. In these instances, it is essential to drive slower and exercise more caution compared to when the robot is precisely following the path. The proximity of the robot to the taught path is taken directly from the lateral component of the curvilinear coordinate q_{robot}.

5. *Proximity to New Obstacles*: The current minimum distance between the robot to nearby obstacles, d_{obs}, plays a significant role in determining an appropriate velocity. As the robot gets closer to a newly discovered obstacle, it should drive more cautiously. This criterion accounts for quick replanning requirements in order to avoid potential collisions or occlusions as the robot's viewpoint changes in close proximity to objects.

The first three conditions lend themselves to pre-computation since they are solely dependant on the path we aim to repeat. By calculating these values for any point along the path in advance, we can provide the end-user with an estimate of the expected velocity during the repeat. This assumes an obstacle-free scenario, which is the most common case. Conversely, the latter two conditions are computed online within the control loop as a preliminary step before formulating the MPC problem.

To determine the planar and profile curvatures, denoted as κ_{planar} and $\kappa_{\mathrm{profile}}$ respectively, we apply a spline pre-processing step to the taught path to ensure smooth curvature variation. While it is feasible to schedule speed based on instantaneous curvature, we found that such a reactive approach did not perform well in preliminary tests. Instead, we assign curvature values at any given point along the taught reference path by averaging the curvature over the upcoming 5 m segment. This allows us to anticipate upcoming turns in advance, enabling us to initiate speed adjustments before reaching the turn. Moreover, this method compensates for curvature fluctuations that may occur along the path.

We propose candidate reference velocity modifications for each of the speed scheduling factors

following the inverse square relationships

$$v_{\text{ref,planar}} = \max(v_{\min}, \frac{v_{\text{ref}}}{1 + \kappa^2_{\text{planar}}\gamma}), \tag{4.13a}$$

$$v_{\text{ref,profile}} = \max(v_{\min}, \frac{v_{\text{ref}}}{1 + \kappa^2_{\text{profile}}\delta}), \tag{4.13b}$$

$$v_{\text{ref,eop}} = \max(v_{\min}, \frac{v_{\text{ref}}}{1 + d^2_{\text{eop}}\epsilon}), \tag{4.13c}$$

$$v_{\text{ref,lat}} = \max(v_{\min}, \frac{v_{\text{ref}}}{1 + q^2_{\text{robot}}\zeta}), \tag{4.13d}$$

$$v_{\text{ref,obs}} = \max(v_{\min}, \frac{v_{\text{ref}}}{1 + \frac{\eta}{d^2_{\text{obs}}}}). \tag{4.13e}$$

These equations incorporate tunable weights, namely $\gamma, \delta, \epsilon, \zeta$, and η, allowing fine-tuning of the scheduler modules to meet user preferences. To ensure a minimum non-zero velocity set point in extreme cases, a configuration parameter, $v_{\min}$, is provided. We generate candidate modifications to the reference velocity set points, and then simply conservatively select the smallest value during the control loop as

$$v_{\text{ref}} = \min(v_{\text{ref,planar}}, v_{\text{ref,profile}}, v_{\text{ref,eop}}, v_{\text{ref,lat}}, v_{\text{ref,obs}}). \tag{4.14}$$

In addition to the aforementioned control modules, we incorporate an extra safety layer in our high-level control system. Following the generation of the optimal sequence of control inputs by the MPC module, we perform a final collision check on the rolled-out trajectory. If this check detects a potential collision, we immediately halt the vehicle. In the vast majority of circumstances, this operation should never be triggered. However, there are situations where rapidly moving dynamic obstacles may pose a challenge for the controller to react and replan adequately. In such cases, this final module serves as a last-resort measure to prevent a collision. By implementing this safety layer, we prioritize the protection of the robot and its surroundings, ensuring a robust and reliable control system that can handle unforeseen obstacles or challenging scenarios.

4.4 Evaluation

4.4.1 Evaluation Strategy and Metrics

We integrated our motion planning architecture into the existing VT&R3 codebase[1] and conducted a series of online experiments to validate our approach. The experiments in this chapter were performed at the Experimental Proving Grounds on Canadian Forces Base Suffield using an Argo Atlas J8 electric differential drive robot in collaboration with Defence Research and Development Canada (DRDC). We run the LiDAR VT&R3 pipeline using a single Ouster OS1 LiDAR operating in 512x64 scan mode at 10Hz on a Lenovo Thinkpad T15 Gen 2 laptop with 11th Gen Intel(R)

[1]Code at https://github.com/utiasASRL/vtr3

Core(TM) i7-11800H @ 2.3GHz. Please refer to Fig. 4.5 for a visual representation of our robot configuration.

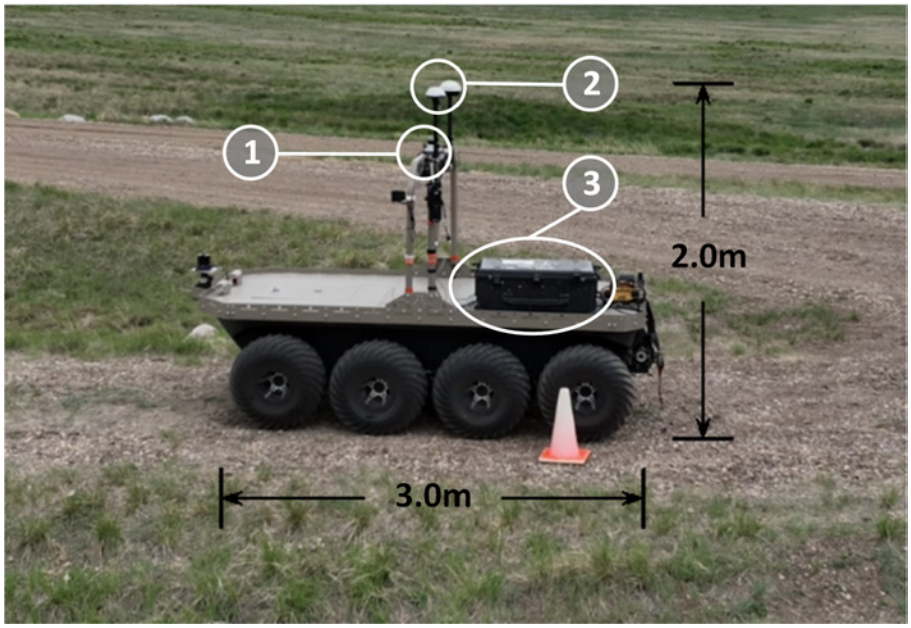


Figure 4.5: The ARGO Atlas J8 Unmanned ground vehicle used to conduct all field evaluation: (1) Ouster OS1 LiDAR 1024x64 @ 10Hz, (2) Hemisphere Dual Differential GPS, (3) Internal Lenovo Thinkpad laptop with 11th Gen Intel(R) Core(TM) i7-11800H @ 2.3GHz.

As our motion planner is ultimately deployed in VT&R3, we dedicated time to validate the proposed controller implementations in typical teach-and-repeat scenarios without obstacles. This step ensured that our new control implementations could achieve similar performance characteristics to previous teach-and-repeat controllers under normal operating conditions, as encountering obstacles on the path is considered an outlier case. Hence, our initial set of experiments focused on analyzing the path tracking performance of both the direct tracking MPC controller and the homotopy guided MPC controller using similar criteria as Ostafew et al. [89]. Our ultimate objective while repeating the previously taught reference path is to minimize the total Root Mean Squared (RMS) lateral and heading errors while limiting the magnitude of the maximum lateral and heading errors that typically occurring during sharp turns in the path and may result in VT&R3 failures.

We assess our path tracking errors by utilizing two sources of information. Firstly, we calculate the lateral and heading RMS errors based on ground truth differential GPS data obtained using a Hemisphere dual GPS. Additionally, we utilize the internal VT&R3 relative localization system to compare the teach-and-repeat routes. While reporting tracking errors typically relies on GPS ground truth, it has been demonstrated that the VT&R3 LiDAR localization system consistently maintains high localization accuracy, with errors of less than 5.2 cm for lateral position and 0.3 deg for heading [21]. In comparison, the accuracy of the available GPS for these experiments ranges from 10-30 cm, depending on satellite availability. Therefore, we provide both sources of information for comprehensive comparison.

After evaluating the path tracking performance in obstacle-free scenarios at fixed velocities, we then introduce the speed scheduler into the architecture to assess its added value. In some applications, such as mining, lawn mowing, or route patrol, the time to complete the repeat may have more of a priority over perfect path following. In these instances, the speed scheduler excels, allowing you to complete a repeat objective in a shorter period of time while still maintaining reasonable path tracking errors. These experiments attempt to explore the trade-offs between these behaviours and demonstrate how the speed scheduler enabled us to achieve the best of both worlds.

Having examined the intrinsic path tracking properties of our controller in this section, the subsequent chapter will focus on presenting the primary experiments related to local obstacle avoidance, serving as the central focus of our work.

4.4.2 Results

Path Tracking Performance

In the initial set of path tracking experiments, we manually drove the robot to teach a network of obstacle-free paths spanning approximately 1.5 km across the Experimental Proving Ground of the Suffield Research Centre in Alberta, Canada. The terrain predominantly consisted of open prairie, with scattered buildings and urban structures. The path teaching process took place in the morning, maintaining a speed of 1 m/s and encompassing a diverse range of trajectories, as depicted in the satellite view shown in Fig. 4.6.

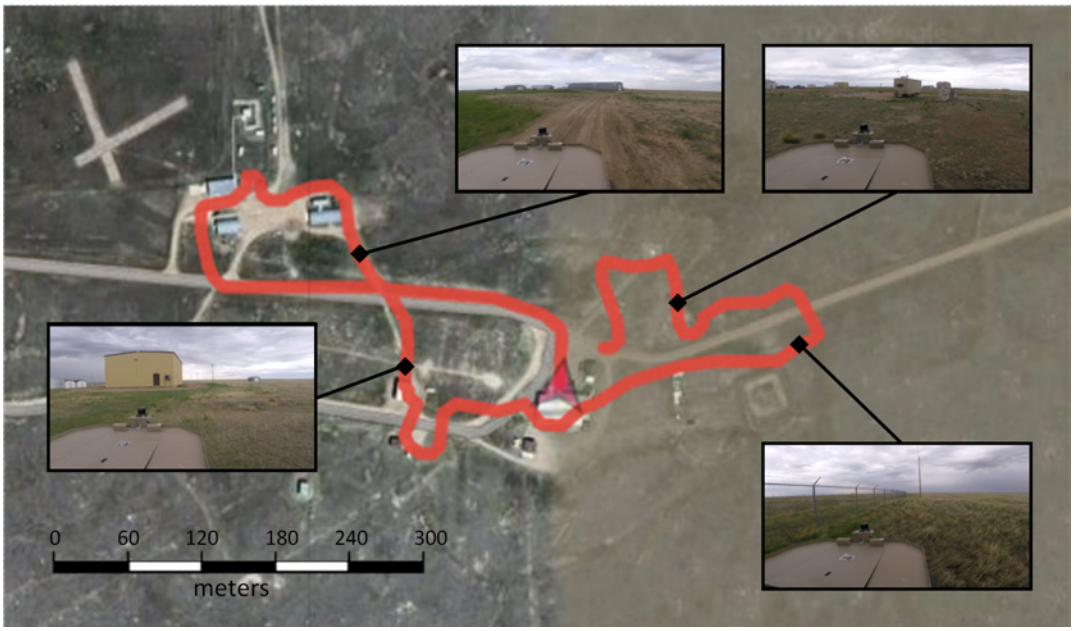


Figure 4.6: We evaluate our VT&R path tracking performance on the Experimental Proving Grounds at Canadian Forces Base Suffield, AB, Canada. The 1.5 km taught reference path is shown in red. The terrain composition was predominantly open prairie, with many small urban structures scattered throughout the taught path.

In the afternoon, we repeated the entire path network using the direct tracking MPC introduced

in Chapter 4.3.1 and the homotopy guided MPC presented in Chapter 4.3.2, using a set reference speed of 1.25 m/s. In an obstacle-free environment, the distinguishing feature between these control architectures is the selection of the reference poses for tracking control. The direct tracking controller attempts to track the output path from the path planner, while the homotopy guided MPC uses reference poses from the taught path, subject to state constraints based on the output of the planner. Nominally, in an obstacle-free scenario, the path produced by the planner aligns perfectly with the taught path, resulting in identical reference poses. However, in practice, if the robot deviates from the reference path due to localization or tracking errors, the planned path may slightly differ in the segment near the robot as it guides the robot back to the path. In the homotopy guided MPC, the controller itself determines the optimal trajectory to compensate for these path-following errors, which we hypothesize will better suit the robot's kinematics, in addition to potentially improving obstacle-avoidance capabilities.

The planner settings remained consistent with the offline experiments described in Chapter 3, employing $\alpha = 0.5$ and setting the number of samples per batch to 150. As for the controller, the tuning parameters were identical for both MPC implementations, utilizing a horizon $K = 20$ with each time step spanning 0.2 seconds. It is worth highlighting that the optimization weights of our MPC were initially established through experimentation on a Clearpath Robotics Grizzly differential drive robot and tested in significantly different terrain. These weights were kept fixed for this particular experiment to showcase the controller's adaptability to various vehicles and environments.

Fig. 4.7 provides a comprehensive visualization of the lateral and heading error magnitudes along the length of the repeat using both variations of our controller. Analyzing the tracking error results with respect to the notionally more accurate VT&R3 localization estimates, we observe that for the majority of the route, the lateral and heading errors remain below 3 cm and 5 deg, respectively, for both proposed controllers. Occasional error peaks occur during sharp turns in the taught path, where, without speed scheduling, the controllers naturally prioritize maintaining a smooth trajectory and the configured repeat velocity over more precise tracking behaviour. Even in these unique and transient circumstances, the maximum tracking errors are kept below 18 cm and 15 deg for both control implementations.

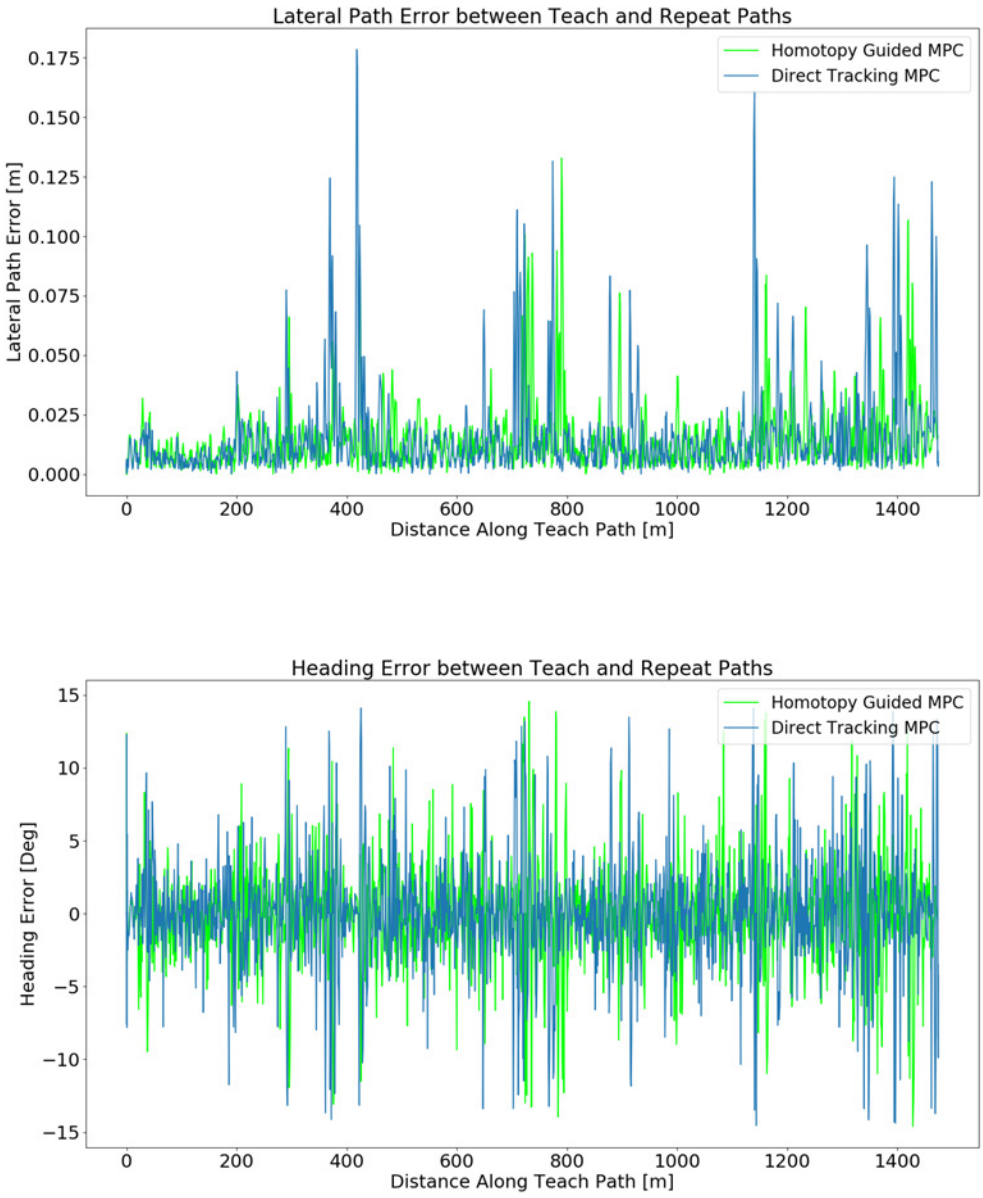


Figure 4.7: A comparison of the lateral and heading path tracking errors between of the direct tracking MPC and the homotopy guided MPC over a long obstacle-free repeat. We find that while heading errors are approximately consistent between the two methods, we see a small performance improvement in lateral path error when using the homotopy guided MPC.

Table 4.1 further supports the desirability of the repeat path trajectories by presenting the RMS errors across the repeat. The direct tracking MPC motion planner demonstrates impressive lateral and heading RMSE of 2.48 cm and 3.8 deg, respectively. On the other hand, the homotopy guided MPC achieves even further improvements with errors of 2.07 cm and 3.5 deg, respectively. As hypothesized, the homotopy guided MPC controller performs marginally better, especially when

Table 4.1: VT&R Path Tracking Error Analysis

Controller	Lateral RMSE [cm] (GPS)	Lateral RMSE [cm] (VTR3)	Heading RMSE [deg] (GPS)	Heading RMSE [deg] (VTR3)
Tracking MPC	12.27	2.48	5.46	3.81
Homotopy MPC	8.73	2.07	5.35	3.50

comparing the magnitude of peak errors, as the limitations of the direct tracking method become more apparent during these sharp turning scenarios.

Compared to previous implementations of VT&R controllers, our controller maintains similar tracking errors while repeating at over twice the velocity of previous experiments conducted by Ostafew et al. [89, 90]. This highlights the potential for the new controllers to directly replace the previous VT&R3 controller, even without considering the additional obstacle-avoidance capabilities evaluated in Chapter 5.

Speed Scheduler Evaluation

Figure 4.8: We taught a 350 m reference path loop in a parking lot at Canadian Forces Base Suffield and perform repeats at various speed profiles to see the influence of the proposed speed scheduler.

In the second controller experiment, we introduce the speed scheduler proposed in Chapter 4.3.4 to the architecture to investigate its impact. A new closed-loop path was taught at Canadian Forces Base Suffield consisting of roads and parking lot spanning a length of 350m. This path was chosen

to represent an autonomous patrol route, as shown in Fig. 4.8. To study the influence of the speed scheduler, we repeated the path multiple times at specific reference velocities of 1.5 m/s, 2.5 m/s, 3.0 m/s and 3.5 m/s. Testing beyond this point was deemed unsafe for the urban environment in which we conducted experiments. The planner and controller parameters from the previous set of experiments remained unchanged.

In the first set of speed trials, we deployed the best-performing controller architecture—the homotopy guided MPC, without the speed scheduler. We then conducted the experiments once more, this time incorporating the speed scheduler to modulate the reference velocity. The time taken to complete the loop and the RMS lateral and heading tracking errors were recorded using both differential GPS and the VT&R3 localization system and the results are reported in Table 4.2.

Table 4.2: Speed Scheduler Error Analysis

Trial	**Lap Time [s]**	**Lateral RMSE [cm] (GPS)**	**Lateral RMSE [cm] (VTR3)**	**Heading RMSE [deg] (GPS)**	**Heading RMSE [deg] (VTR3)**
1.5 m/s, No Scheduler	265.99	8.41	2.00	3.67	2.88
2.5 m/s, No Scheduler	180.84	8.81	3.27	5.09	3.71
3.0 m/s, No Scheduler	N/A	N/A	N/A	N/A	N/A
3.5 m/s, No Scheduler	N/A	N/A	N/A	N/A	N/A
1.5 m/s, Scheduler	271.40	3.12	1.82	3.44	2.55
2.5 m/s, Scheduler	194.76	8.86	3.41	4.69	3.79
3.0 m/s, Scheduler	146.62	9.71	3.91	4.78	3.96
3.5 m/s, Scheduler	125.69	9.12	5.35	5.66	4.29

While the overall tracking errors remain consistent regardless of the use of the speed scheduler and gradually increase with higher target velocities as expected, it is important to note that this result alone can be somewhat misleading. The long straight sections of the path tend to be well-tracked at high velocities, regardless of the use of a scheduler. However, the most significant difference is observed during turns, where the curvature components of the speed scheduler effectively anticipates the turns and slows down the vehicle to improve manoeuvrability.

Without the speed scheduler, we could only successfully complete the patrol route repeat at a maximum speed of 2.5 m/s. Any attempt to exceed this reference velocity consistently resulted in loss of control and localization failures during sharp turns. This observation is supported by analyzing the lateral path error plot in Fig. 4.9. Both with and without the scheduler, the trend shows that peak lateral errors gradually increase as the speed increases. However, when employing the speed scheduler, these peaks are significantly reduced, enabling us to achieve a much higher successful repeat velocity. Compared to the somewhat high repeat time of 3 minutes and 0.84 seconds at 2.5 m/s without the scheduler, the speed scheduler allowed us to complete the path in just 2 minutes and 5.69 seconds, while maintaining a noticeably safer behaviour.

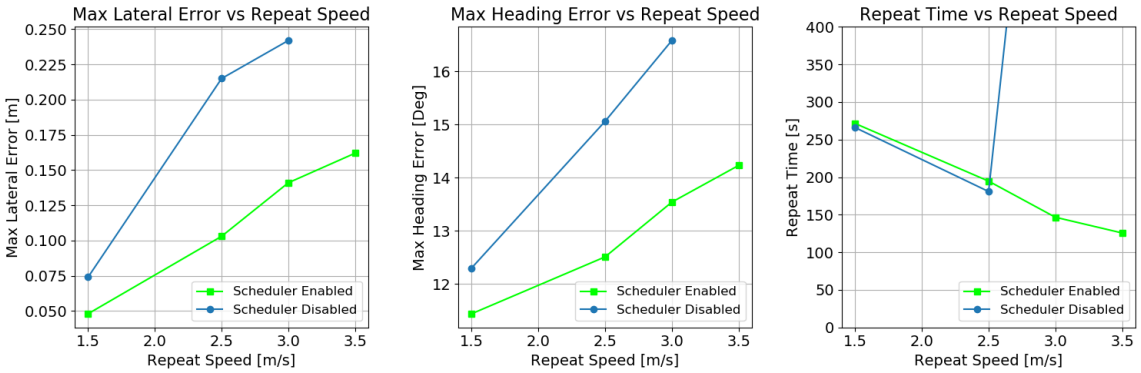


Figure 4.9: In this figure, we compare the maximum lateral errors, heading errors, and repeat completion time, both with and without the use of the speed scheduler. We see that the magnitude of the path tracking errors decreases considerably across the repeat speeds, allowing us to complete the repeat objectives safely and in a shorter period of time. With the speed scheduler disabled, the robot was unable to complete the repeat beyond 2.5 m/s as a result of losing control on a sharp turn.

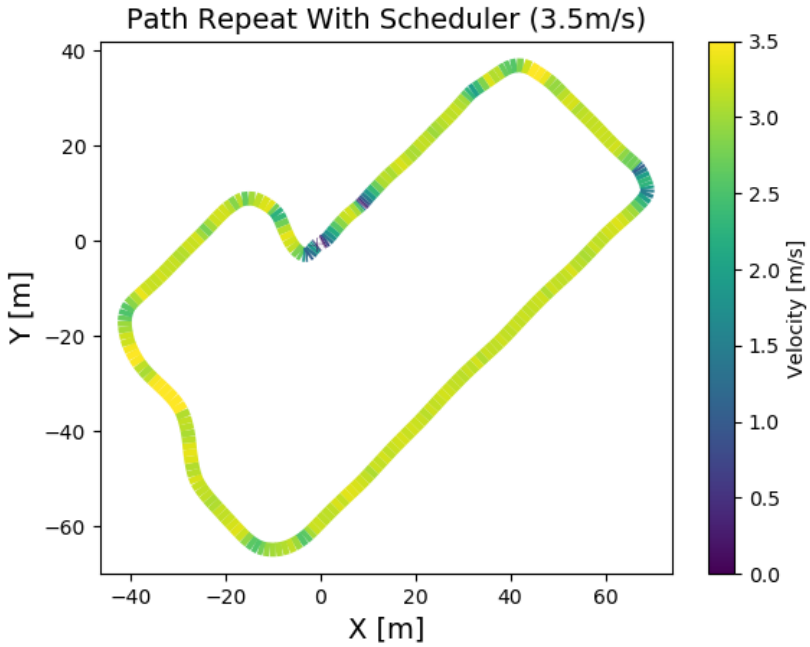


Figure 4.10: The velocity profile at all points along the teach path while repeating with a set reference velocity of 3.5 m/s and the speed scheduler. We see the speed is regulated most dramatically early on in the path when the robot is in close proximity to surrounding vehicles and on the sharp corners with high curvature.

It is worth mentioning that the speed scheduler was originally tuned for nominal operating speeds of 1.25 m/s, and repeating at speeds exceeding 2.5 m/s is considered unusual in VT&R3. If a higher normal operating point is expected, the speed scheduler can be tuned to optimize its performance for these higher speeds.

4.5 Summary

In this chapter, we presented two controller implementations that complement our path planner and enable autonomous path following with local obstacle avoidance. Our first approach involves utilizing a tracking MPC controller that directly follows the planned path while incorporating various constraints. To enforce the kinematics on different robot platforms effectively, we formulate the problem using matrix Lie Groups and provide a solution method for these types of optimization problems. This approach proves to be straightforward and efficient; however, it does not provide a strict guarantee on collision avoidance. Hence, we introduced our second MPC approach, a path-following MPC with hard state constraints derived from the homotopy class of the planner solution. This enables us to avoid obstacles with a guaranteed level of collision avoidance.

To enhance the performance of the prospective MPC controllers, we integrate a custom speed scheduler that regulates the repeat speed based on a number of factors and then conduct a series of online experiments using a large all-terrain autonomous vehicle. Our goal was to replicate or improve upon the basic performance of previous teach-and-repeat motion planners. Through these experiments, we establish that our motion planner exhibits several desirable characteristics for teach-and-repeat applications, including reliability and minimal tracking errors. In the upcoming chapter, our attention turns to assessing the motion planner in scenarios requiring local obstacle avoidance to complete the teach-and-repeat objectives, where the primary advantages of our proposed architecture will become evident.

Chapter 5

Closed-Loop Obstacle Avoidance Experiments

5.1 Introduction

Our final set of experiments aims to assess the effectiveness of our entire system in real-world obstacle-avoidance scenarios. We successfully integrated our obstacle-avoidance architecture into the VT&R3 codebase and conducted extensive long-term autonomy tests in two diverse environments. These environments presented contrasting challenges: the first involved navigating a relatively flat prairie terrain with a simple single loop, while the second featured a more complex network of paths in a valley with varying elevations.

In each case, the robot was initially manually driven to teach an obstacle-free network of paths. Subsequently, various obstacles were introduced throughout the path network to obstruct the predefined routes. These obstacles were placed to simulate potential long-term changes that could occur along repeat paths. Through a series of experiments, we repeated the taught paths and analyzed the resulting trajectories produced by our two motion planning architectures. Ultimately, this investigation aimed to explore the suitability of these approaches in extending the long-term navigation capabilities of VT&R3 with local obstacle avoidance.

The experiments were performed in collaboration with DRDC, utilizing an ARGO Atlas J8 electric differential drive robot under identical configurations to those presented in Chapter 4. We deployed the LiDAR VT&R3 pipeline on a Lenovo Thinkpad T15 Gen 2 laptop equipped with an 11th Gen Intel(R) Core(TM) i7-11800H processor running at 2.3GHz, as illustrated in Fig. 4.5.

5.2 Evaluation Strategy and Metrics

In our evaluation, we emphasize the importance of obstacle-avoidance trajectories that strike a balance between minimizing lateral deviations from the taught path and ensuring smooth forward progress. This balance is crucial due to the prior knowledge we have about the taught path's

collision-free nature, as verified by a human operator. By closely following the reference path, we maximize our chances of avoiding unforeseen obstacles and improve localization performance in VT&R3 despite perception uncertainties.

To assess the performance, we report the RMS lateral and heading errors obtained from both the differential GPS and the VT&R3 state estimation system during the repeat experiments. However, it is important to note that these values become subjective, as they are influenced by the size and density of obstacles encountered during each repeat. To address this, we propose a new metric called the "obstacle interaction" RMS lateral and heading error that focuses on a specific obstacle's influence on path deviation. We define an obstacle interaction as the length of the path extending 5 m on either side of the obstacle. By comparing the obstacle interaction RMS error with the error in obstacle-free sections, we gain insights into the relative amount of path deviation caused by the planner to successfully avoid the obstacle.

Additionally, we consider the average maximum lateral deviation per obstacle interaction. This measure, when evaluated in the context of the lateral extent of the obstacle obstructing the reference path, provides an indication of how effectively the planner avoids obstacles while staying close to the reference path. We are particularly interested in observing the consistency of the maximum lateral deviation across obstacles with varying geometries.

When comparing the two motion planners, the direct tracking MPC and the homotopy guided MPC, we propose using the average robot path curvature as a metric to evaluate the relative smoothness of these approaches. Our rationale is that if both motion planners can maintain similar desirable lateral error characteristics during obstacle avoidance, the path with the lowest average curvature is likely to be more preferable in terms of energy efficiency. This consideration is related to our design choices and tuning parameter settings, as there is inherently a trade-off between generating smooth paths with minimal control effort and optimizing for minimal lateral path deviations and we hope to find a strong middle ground.

Finally, one of our most critical and intuitive metrics of interest is the obstacle-avoidance rate. This rate is calculated by dividing the number of successful obstacle interactions (those that are collision-free) by the total number of obstacles encountered on the route. It serves as a reliable indicator of the improvement in the long-term autonomy of the VT&R3 system, as our goal is to minimize the need for operator interventions during autonomous navigation, particularly under these difficult unanticipated obstacle-avoidance scenarios.

5.3 Experiment 1: Easy Scenario

5.3.1 Environment Setup

Our obstacle-avoidance experiments took place on the Experimental Proving Grounds of Canadian Forces Base Suffield, a vast isolated grassland area dedicated to military training and testing in Alberta, Canada. In this first experiment, we focused on a relatively flat section of the prairie and designed a large looping reference path spanning 304 m. This path simulated a patrol route within

an active exercise area, featuring scattered small urban structures. These structures, along with occasional moving pieces of equipment and debris, added variability to the path conditions. Prior to introducing obstacles, we repeated the route to establish a baseline for tracking performance, similar to the experiments conducted in Chapter 4.4.2.

Fig. 5.1 provides a satellite view of the terrain and the small patrol loop, as well as images depicting the representative terrain and the obstacles incorporated along the path. A total of 10 obstacles were placed sporadically along the loop, and we conducted 5 repetitions using both the direct tracking MPC architecture and the homotopy guided MPC. While the size and shape of the objects varied, the overall terrain exhibited a relatively uniform and planar nature, making it a suitable scenario for demonstrating the basic capabilities of the local obstacle-avoidance system in a controlled setting. To maintain consistency in evaluating the proposed motion planners, we excluded dynamic obstacles from this experiment, leaving their analysis for future work.

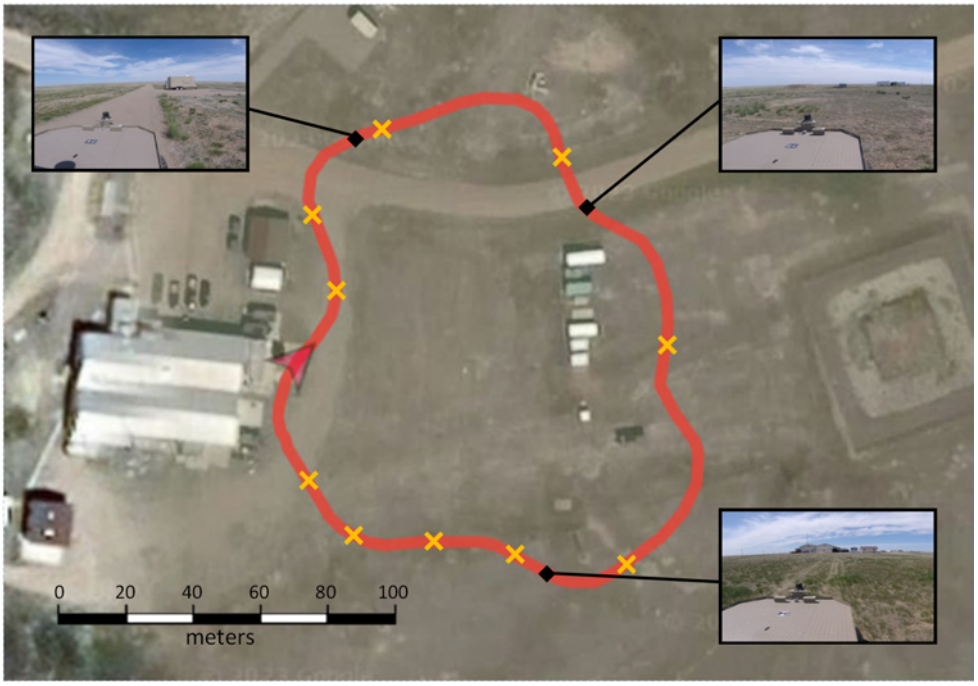


Figure 5.1: We evaluate our local obstacle-avoidance system at the Experimental Proving Ground on Canadian Forces Base Suffield. In the first scenario, we taught a short loop on reasonably flat and open terrain. We then introduced a series of 10 obstacles (yellow crosses) to the reference path, forcing the robot to deviate from the teach path to complete subsequent repeat objectives.

5.3.2 Results

Upon repeating the loop both without obstacles and with obstacles, we present the actual robot path relative to the taught reference path (as reported by GPS) across the loop in Fig. 5.2. The locations and sizes of the obstacles have been marked for reference, and some examples of obstacle interactions are provided in Fig. 5.3. In both the direct tracking MPC and the homotopy guided MPC implementations, the robot successfully completed the loop 5 times at a target reference speed of

1.25 m/s, achieving a 100% obstacle-avoidance rate. Throughout this experiment, the robot covered a cumulative distance of 1.5 km, navigating through 50 obstacles without any collisions or operator interventions. While both controllers accomplished our primary goal in this less challenging scenario, we will now delve into the relative differences and trajectory characteristics of the two methods.

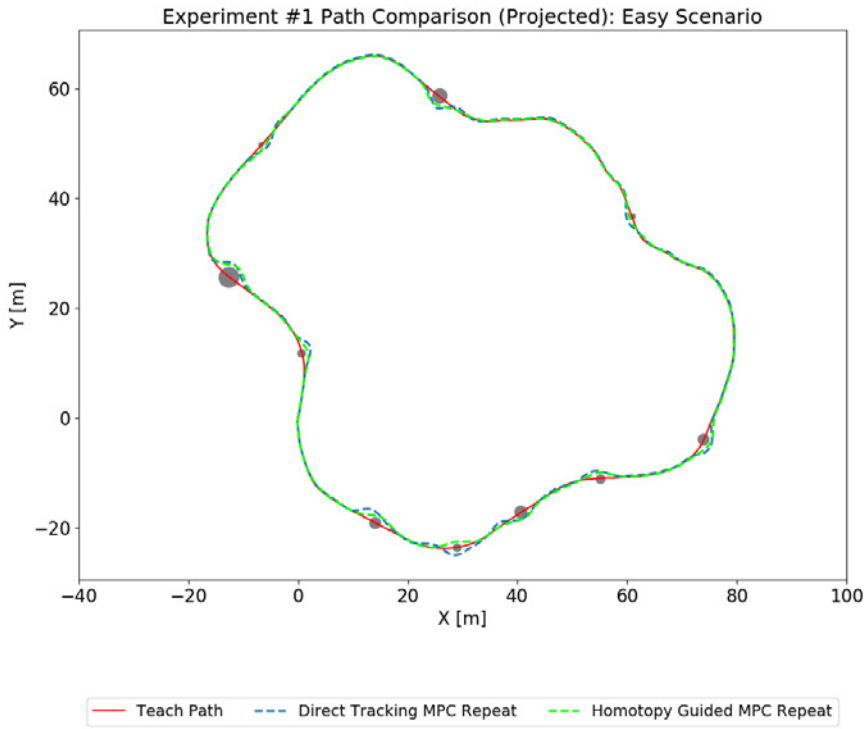


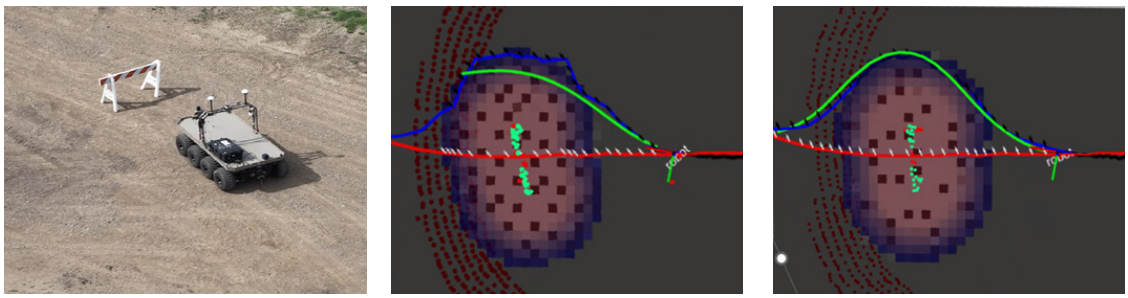
Figure 5.2: A plot of the repeat trajectories overlaid with the teach path as reported by GPS ground truth. We see that the homotopy guided MPC solution (green), generally avoids obstacles with more consistent obstacle interaction profiles and with less overall lateral path deviation then the direct tracking approach (blue). None the less, both techniques were successful 100% of the time in avoiding the set of obstacles.

Both motion planners demonstrated a tendency to closely follow the reference path and smoothly navigate around obstacles. In obstacle-free segments of the path, the planner promptly realigned with the nominal reference trajectory. To assess the performance of the planners during obstacle interactions, we calculated the RMS lateral and heading errors for obstacle-free repeats of the environment, serving as a baseline for path tracking performance. For this particular path network, the direct tracking MPC achieved a lateral RMS error of 2.50 cm, while the homotopy guided MPC yielded a slightly lower error of 2.21 cm. Similarly, the baseline heading RMS errors were 4.5 deg and 4.0 deg, respectively, agreeing well with our previous Chapter 4 tracking error results.

Next, we recalculated the RMS error metrics specifically for the obstacle-avoidance interactions, focusing on the 10 m segments of the path centered around each obstacle. By comparing the tracking errors during obstacle interactions for each motion planner, we aimed to quantify their performance based on our design guidelines. We posit that achieving lower lateral tracking errors during obstacle

(a) Small trailer (b) Wooden pallet

Figure 5.3: Some typical obstacle interaction scenarios encountered along the reference path.



(a) Avoiding a large barrier (b) Direct Tracking MPC (c) Homotopy Guided MPC

Figure 5.4: In this figure, we compare the solutions of the two controller architectures over a representative obstacle-avoidance scenario with a barrier. The current planner solution (blue path) provides an initial path leading the robot around the obstacle and back to the reference path (red path). In (b), the direct tracking MPC produces a smooth trajectory (green path), to track the BIT* solution, however, due to infeasible kinematics in the planner solution, there is some tracking error that takes the robot dangerously close to the obstacle. In contrast, the homotopy guided MPC generates a tight and safe trajectory around the obstacle using a similar planner solution.

interactions implies a tighter trajectory that closely adheres to the initially taught path, thus better leveraging our prior terrain knowledge. As long as this trajectory remains collision-free and safe, we prefer it over excessively conservative obstacle-avoidance strategies. The results of the obstacle interaction analysis for one loop repetition of each controller are presented in Table 5.1. We provide the extent of the obstacle on the path for reference as this directly influences how far the robot must deviate from the path during an obstacle interaction and depends on both the size of the robot and obstacle. Our findings indicate that both controllers exhibit comparable performance in this aspect, with the homotopy guided MPC demonstrating a reduction in obstacle interaction errors compared to the direct tracking MPC, as anticipated.

Table 5.1: Obstacle Avoidance Error Table for the Easy Scenario

Obstacle # Extent [m]		**Direct Tracking MPC**			**Homotopy Guided MPC**		
		Maximum Lateral Error [m]	**Lateral RMSE [m]**	**Heading RMSE [deg]**	**Maximum Lateral Error [m]**	**Lateral RMSE [m]**	**Heading RMSE [deg]**
1.	0.7	1.54	0.90	22.98	1.02	0.62	14.37
2.	1.8	2.17	1.33	34.69	2.22	1.46	27.38
3.	0.3	1.17	0.64	17.63	0.68	0.42	8.81
4.	1.3	2.11	1.21	21.85	1.63	1.02	21.74
5.	0.2	1.16	0.60	16.47	0.48	0.27	10.50
6.	1.0	1.70	0.95	15.39	1.36	0.87	14.87
7.	0.8	1.20	0.60	18.52	1.10	0.61	17.16
8.	1.1	1.40	0.72	18.87	1.38	0.81	18.12
9.	0.7	1.67	1.05	20.94	1.08	0.66	17.40
10.	1.0	1.53	0.91	21.23	1.30	0.84	14.94

The obtained data aligns with our qualitative observations, affirming that the planner effectively avoids significant translation and rotation errors relative to the extent of the encountered obstacles, while avoiding any adverse effects on localization. Additionally, we consider the maximum lateral deviation per obstacle interaction as an informative metric. By examining the peaks of the lateral error magnitude plot along the path for both planners in Fig. 5.5, we find that, on average, the direct tracking MPC deviates laterally from the reference path by a maximum of r + 0.675 m with a standard deviation of $\pm$ 0.254 m, while the homotopy guided MPC results in r + 0.335 m with standard deviation $\pm$ 0.048 m. In this context, r represents the lateral extent of the obstacle obstructing the reference path. Theoretically, the lower bound of this value should be exactly r for collision-free avoidance, but knowing that our perception is not infallible and to ensure a small safety buffer, we introduce an obstacle inflation tuning parameter of 0.3 m, resulting in a slight expected offset. Notably, the homotopy guided motion planner is able to achieve maximum lateral deviation characteristics close to this bound and the small uncertainty in the measure is evident of consistent obstacle-avoidance behaviour across obstacles of varying sizes and geometries. In contrast, the direct tracking MPC tends to take a more conservative approach to avoiding obstacles, with less repeatability.

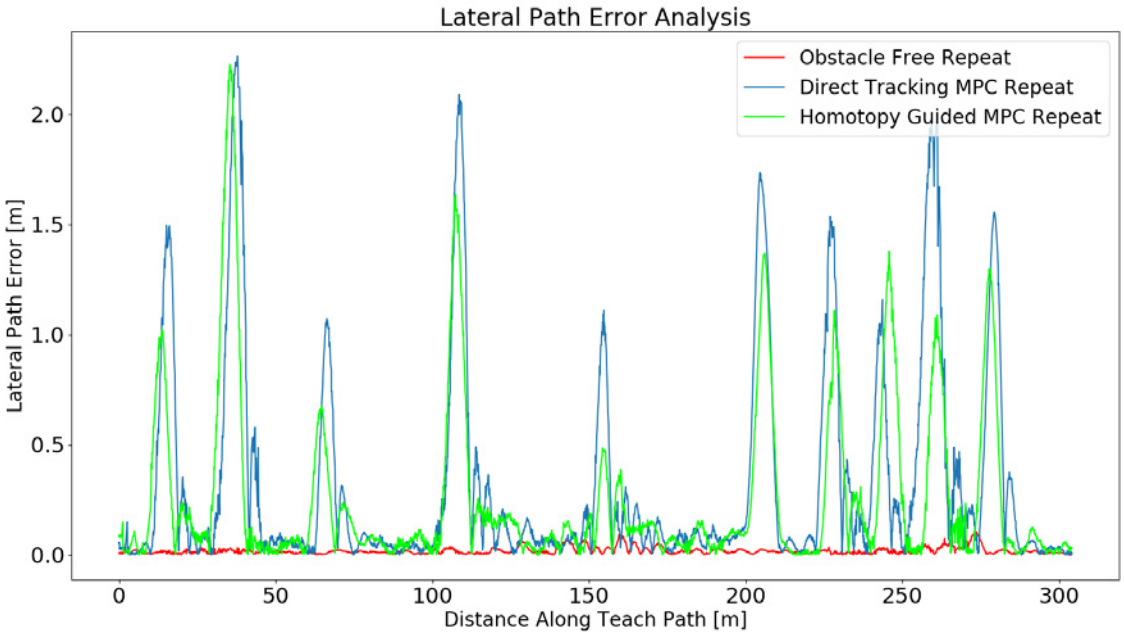


Figure 5.5: A comparison of the lateral path errors across the obstacle-avoidance repeats for the two proposed control architectures.

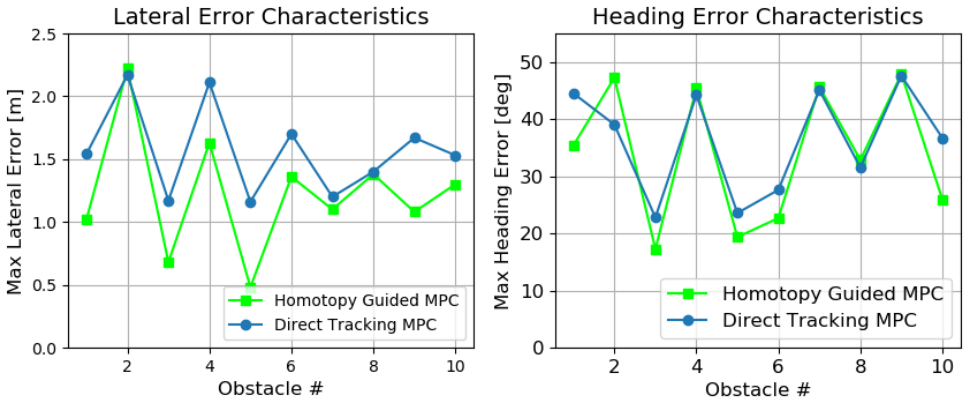


Figure 5.6: The maximum tracking error characteristics on an individual obstacle interaction basis.

Another aspect of comparison between the two motion planners is the average curvature of the robot trajectory across the repeats. Intuitively, the presence of obstacles leads to an increase in average path curvature, as the robot maneuvers to avoid collisions. However, we contend that minimizing the increase in path curvature during obstacle encounters is a desirable characteristic for the motion planner. As a baseline, we see that the obstacle-free average curvature of 0.093 m^{-1} increases to 0.112 m^{-1} for the direct tracking MPC and 0.105 m^{-1} for the homotopy guided MPC, representing a 20.4% and 12.9% increase, respectively. These findings indicate that while the tracking errors during obstacle interactions are similar, the homotopy guided MPC produces more efficient and better optimized trajectories.

5.4 Experiment 2: Hard Scenario

5.4.1 Environment Setup

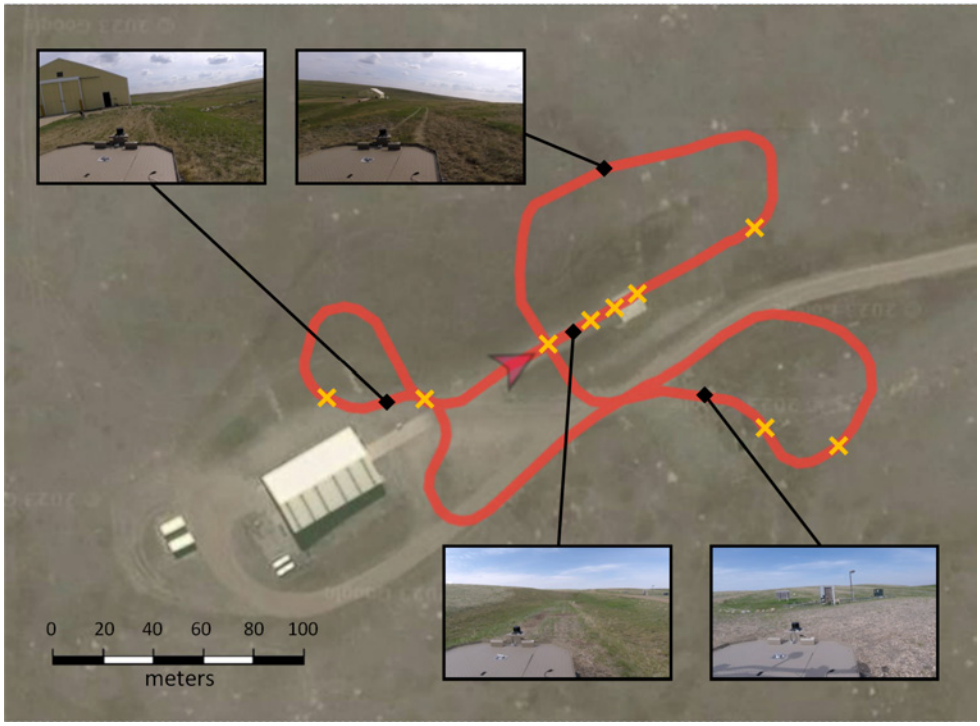


Figure 5.7: We evaluate our local obstacle-avoidance system at the Experimental Proving Ground on Canadian Forces Base Suffield. In the second scenario, we taught a long and intricate route in the middle of a valley consisting of many elevation changes and diverse terrain. We then introduced a series of 11 obstacles (yellow crosses) to the reference path, forcing the robot to deviate from the teach path to complete subsequent repeat objectives.

For our second experiment, we conducted the tests in another section of the Experimental Proving Grounds at Canadian Forces Base Suffield. This time, we selected a more remote location situated in the basin of a valley, with a prominent research station structure at its centre. This setting allowed us to demonstrate the performance of VT&R3 and our motion planners in a complex, non-planar

Figure 5.8: A subset of the obstacle interactions the robot faced while completed the path repeats. in the hard scenario.

environment. In addition to the challenging terrain, the reference path we taught using VT&R3 presented greater difficulty for repeating. The path encompassed a complex loop that traversed the many slopes of the valley, featuring several self-intersections to comprehensively explore the surrounding area before returning to the starting point.

After repeating the obstacle-free reference path to establish baseline tracking data, we introduced a series of 11 obstacle interactions, similar to those in Chapter 5.3, onto the path. Fig. 5.7 provides a satellite view of the location, accompanied by sample images showcasing the obstacle interactions. We then repeated the path using both the direct tracking MPC and the homotopy guided MPC motion planning architectures, and evaluated the results based on our selected metrics.

5.4.2 Results

A comparison between the teach-and-repeat paths for both control schemes is provided in Fig. 5.10. In this experiment, a similar set of obstacles as in the easier terrain scenario were used, but with more challenging placement, necessitating complex trajectories within a more heavily constrained safe-corridor. Additionally, the presence of elevation changes obstructed the view of some obstacles along the repeat path until the vehicle approached close enough to clear occlusions, requiring faster replanning to successfully avoid obstacles. The total path length was 550 m, repeated 5 times for each motion planner, resulting in the robot encountering a total of 55 obstacles interactions over a cumulative navigation distance of 2.75 km.

In this more difficult test case, both motion planners successfully completed the repeats without explicit operator intervention. However, the direct tracking MPC exhibited minor glancing col-

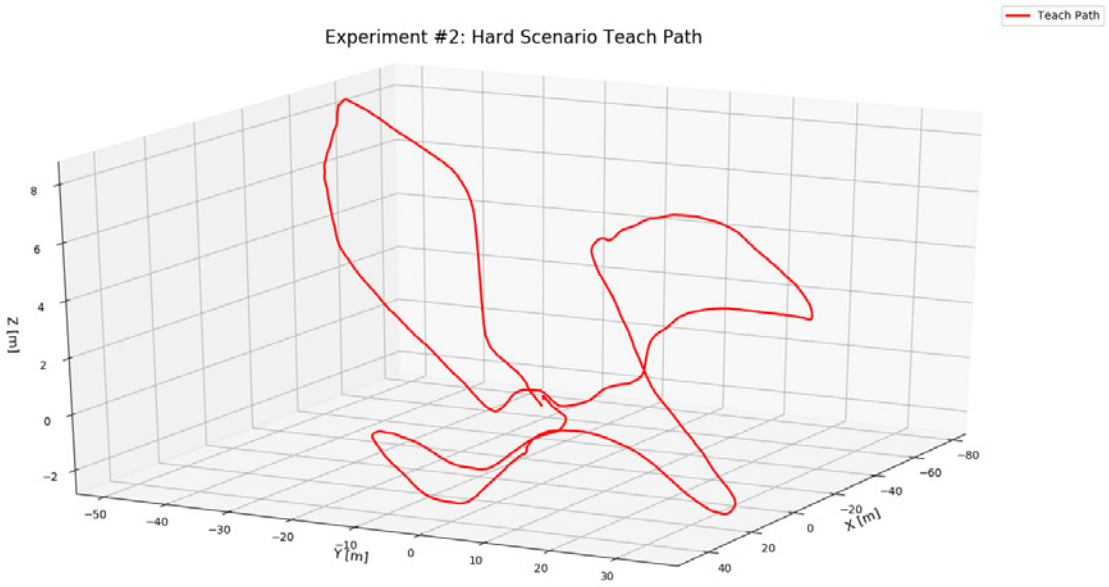
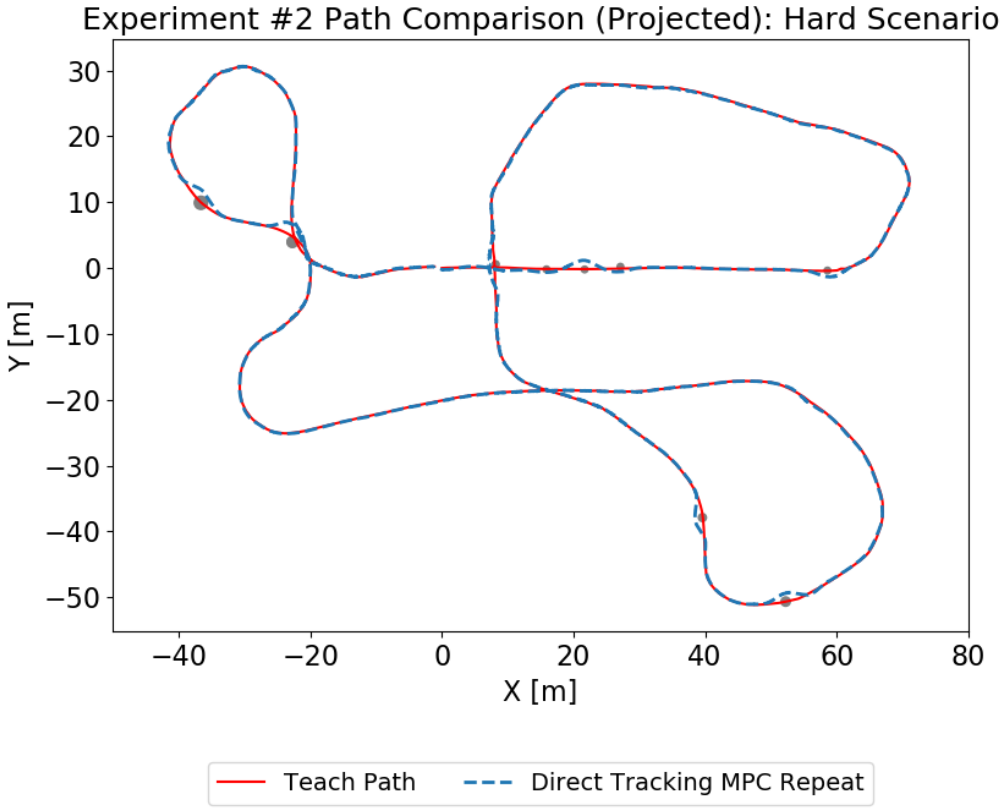


Figure 5.9: The taught reference path showcasing the large elevation changes across the loop as reported by GPS

lisions with some obstacles, resulting in an obstacle-avoidance rate of 87.27%. In contrast, the homotopy guided MPC maintained a perfect 100% obstacle-avoidance rate. Specifically, the direct tracking MPC consistently collided with obstacle #1 and occasionally with obstacle #5. Obstacle #1 was a small wooden barrier, for which the path planner quickly converged to a highly optimized tight collision avoidance path. However, due to kinematic tracking errors, the robot struggled to precisely follow this path and ended up contacting the edge of the barrier. On the other hand, obstacle #5 was situated on a steep slope with limited visibility until the robot approached within 2 m of the obstacle. Consequently, the planner needed to react swiftly to re-plan around the obstacle, and occasionally, it was unable to generate a sufficiently converged solution in time to fully maneuver around the obstacle, resulting in a glancing blow.

In contrast, the homotopy guided MPC showed no issues while navigating the aforementioned obstacles. This likely can be attributed to the decoupled nature of the planner and controller, along with the capability to generate optimized trajectories by leveraging rough initial solutions from the planner. These characteristics enable rapid reactions to unforeseen obstacles. While the more successful obstacle-avoidance rates achieved by the homotopy guided MPC provide telling conclusions, the RMS lateral and heading errors for the obstacle interactions further corroborate the earlier observations made during the easier obstacle course.

The baseline obstacle-free path tracking errors were measured as 6.12 cm for lateral errors and 6.1 deg for heading errors in the case of the direct tracking MPC. Similarly, the homotopy guided MPC exhibited obstacle-free lateral and heading errors of 5.94 cm and 6.0 deg, respectively. It is worth noting that the tracking errors increased in the baseline obstacle-free repeats compared to those of the easier environment, likely due to the more challenging reference path trajectories and



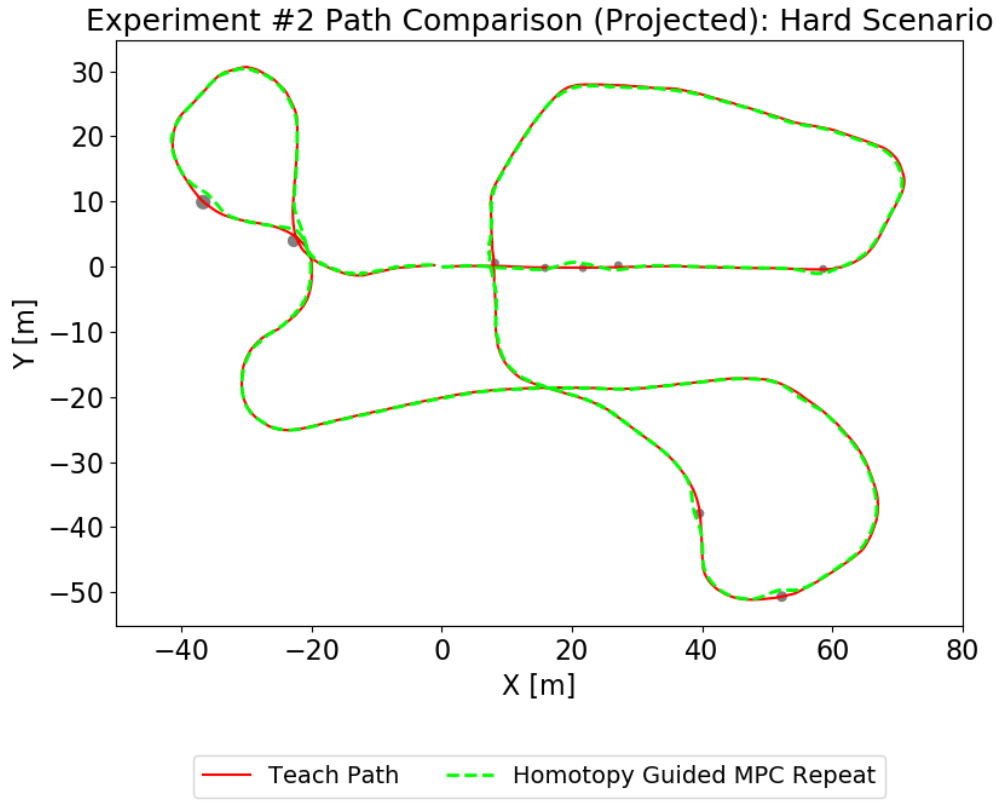


Figure 5.10: A plot of the repeat trajectories overlaid with the teach path as reported by GPS ground truth. We find that the trends from the easy scenario continue with the homotopy guided controller generating marginally more consistent trajectories with smaller lateral errors than the direct tracking MPC.

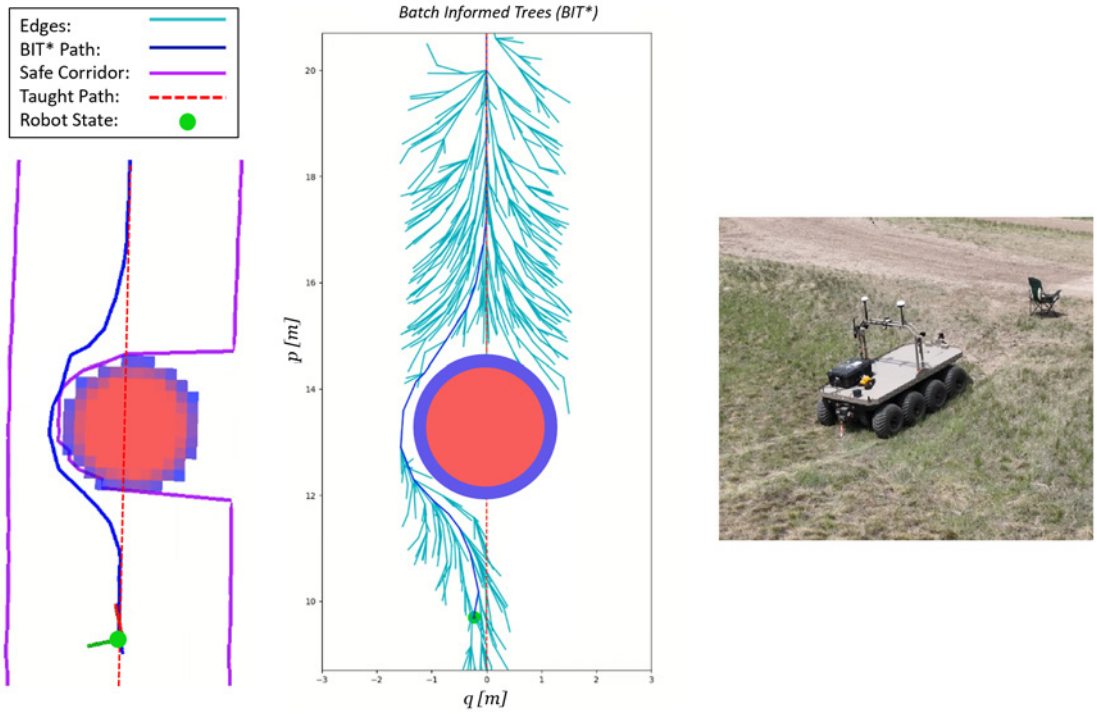


Figure 5.11: A representative BIT* path and safe lateral corridor constraint solution, taken at a snapshot along the repeat.

the rough in terrain composition. Following obstacle-free repeats, obstacles were introduced on the path and Table 5.2 reports the obstacle interaction RMS error metrics for the repeats.

In addition to achieving more reliable and safe navigation, we consistently observe that the homotopy guided MPC maintains or surpasses the quality of local obstacle-avoidance trajectories compared to the direct tracking MPC, particularly in minimizing lateral and rotational path errors. To further analyze the lateral deviations, we present an overlaid plot of the lateral path errors for both motion planners in Fig. 5.12. On average, the direct tracking MPC deviates laterally from the reference path by a maximum of r + 0.484 m, with a standard deviation of $\pm$ 0.158 m, while the homotopy guided MPC results in r + 0.309 m, with a standard deviation of $\pm$ 0.052 m. It is noteworthy that the trajectory characteristics of the repeated obstacle-avoidance paths maintain the trends observed in Chapter 5.3, where the homotopy guided MPC exhibits slight advantages in terms of the consistency of lateral deviations, despite the significantly different operating environment. Both motion planning solutions, however, generate qualitatively desirable paths that avoid obstacles while closely adhering to the reference path to complete the objective and exhibit consistent characteristics across a range of operating conditions.

Table 5.2: Obstacle Avoidance Error Table for the Hard Scenario

Obstacle #	Extent [m]	Direct Tracking MPC			Homotopy Guided MPC		
		Maximum Lateral Error [m]	Lateral RMSE [m]	Heading RMSE [deg]	Maximum Lateral Error [m]	Lateral RMSE [m]	Heading RMSE [deg]
1.	0.1	0.48	0.31	8.48	0.30	0.22	4.32
2.	0.1	0.52	0.38	13.49	0.33	0.33	10.20
3.	0.5	1.33	0.70	21.81	0.82	0.49	13.39
4.	0.1	0.49	0.40	17.40	0.41	0.28	10.18
5.	0.3	0.92	0.52	13.45	0.64	0.37	11.70
6.	0.3	0.78	0.40	15.39	0.65	0.36	11.80
7.	0.6	1.21	0.58	17.61	0.90	0.54	11.10
8.	0.7	1.26	0.69	19.40	1.04	0.63	13.01
9.	0.4	0.78	0.50	14.88	0.72	0.35	10.56
10.	1.0	1.40	0.81	17.82	1.31	0.83	14.84
11.	0.8	1.15	0.58	12.37	1.18	0.60	9.76

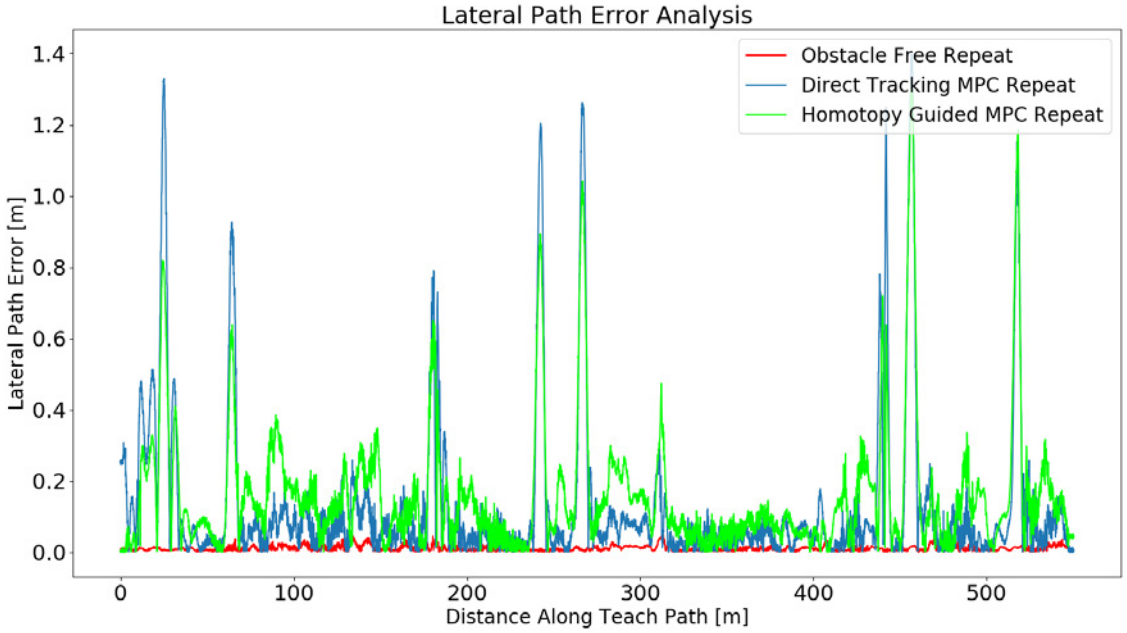


Figure 5.12: A comparison of the lateral path errors across the obstacle-avoidance repeats for the two proposed control architectures.

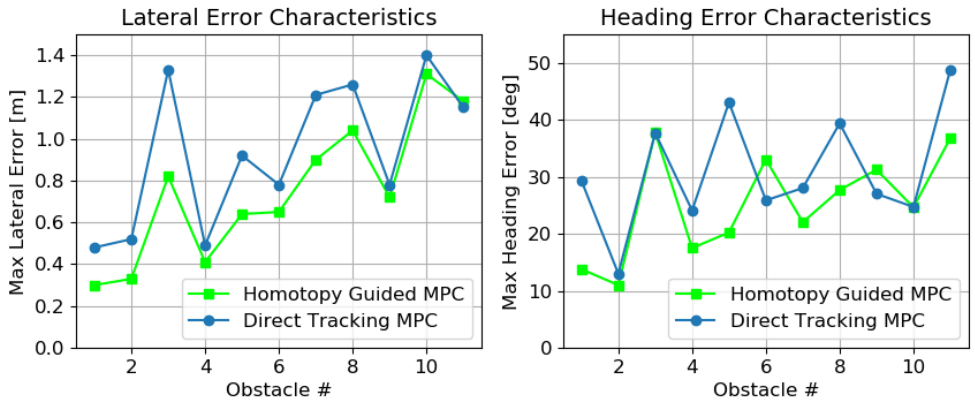


Figure 5.13: The maximum tracking error characteristics on an individual obstacle interaction basis.

When considering the average path curvature, we find further evidence supporting the superiority of the homotopy guided MPC over the direct tracking approach. With an obstacle-free average path curvature of 0.106 m^{-1}, the direct tracking method results in a robot trajectory with a curvature of 0.123 m^{-1}, while the homotopy guided method achieves 0.114 m^{-1}. This corresponds to a change of 16.0% and 7.5%, respectively.

5.5 Discussion

In this chapter, we conducted a comparative analysis of two motion planning approaches in the context of obstacle avoidance in complex environments. Through a series of experiments on both easier and more challenging terrains, we evaluated the performance of the two controllers in terms of obstacle-avoidance rate, path tracking errors, lateral deviations, and path curvature. The results consistently demonstrated the slight benefits of the homotopy guided MPC over the direct tracking MPC. The homotopy guided MPC exhibited higher obstacle-avoidance rates, achieved lower lateral and rotational path errors, and maintained more consistent maximum lateral deviations across various obstacle sizes and geometries. Additionally, it generated paths with reduced path curvature, indicating more efficient and optimized trajectories. These findings highlight the effectiveness of the homotopy guided MPC in generating safe and efficient robot trajectories in a variety of unstructured environments.

While the direct tracking MPC does indeed offer a potentially viable motion planning solution, it is important to discuss the reasons for some of its shortcomings. In the direct tracking approach, the MPC attempts to track the current path solution from the planner, but it is possible that the current plan may not be well converged or smooth during certain control iterations, particularly early on when new obstacles are detected. Consequently, this leads to larger kinematic tracking errors that can manifest as glancing collisions as we saw in the more challenging test environment. While increasing the obstacle inflation parameter from our previous set point of 0.3 m can mitigate collisions, it also amplifies the lateral and heading errors, which is not desirable. Furthermore, when

the controller tracks an unconverged path solution, the robot may deviate further from the taught path until the planner finds a better solution, resulting in increased path curvatures and lateral errors that accumulate along the way.

In contrast, the homotopy guided MPC overcomes these limitations. As we are able to identify the correct homotopy class of the optimal solution nearly 10 times faster then we can find a converged path solution, the homotopy guided motion planner is able to more quickly adapt to obstacles, and generate optimal local avoidance trajectories further in advance from obstacles. Moreover, as the controller tracks reference poses directly from the taught path, the overall path-following performance is expected to improve. Lastly, as obstacles are taken into account directly as constraints, there is much less chance that collisions occur due to kinematic tracking errors (under the assumption of perfect obstacle detection).

Regardless of the control implementation selected, these experiments provide promising evidence for the efficacy of our proposed local obstacle-avoidance path planner. In each instance, our planner serves as the underlying guide, consistently generating robot path solutions that are ideally suited for teach-and-repeat applications, as well as generalized path-following tasks that prioritize maintaining proximity to a reference path. The outcomes of these experiments yield compelling evidence supporting the practicality and effectiveness of our proposed planner as a versatile tool for finding optimized robot trajectories in local obstacle-avoidance scenarios.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this work, we presented a local obstacle-avoidance architecture designed for path-following applications such as teach and repeat. By modifying a sample-based motion planner to use a laterally weighted edge-cost metric combined with a curvilinear planning space, we show how natural path following can be achieved that exploits prior knowledge of the terrain to avoid obstacles. Our method emphasizes reducing lateral deviations from a previously taught reference path, which is expected to offer increased safety during traversal. We then explored two different MPC architectures that leverage the current path planning solution to generate smooth robot trajectories and avoid local obstacles along the reference path.

The primary objective for this research was to expand the long-term navigation capabilities of a teach-and-repeat style architecture. By integrating our obstacle-avoidance system with VT&R3 we are able to complete repeat objectives through both gradual and spontaneous environmental changes over time. In Chapter 2 we discuss the challenges associated with this objective and explore the past literature of previous teach-and-repeat development to establish a set of guiding motion planning characteristics for our algorithm design choices.

Chapter 3 presents our most significant contribution, the sample-based path planning module. We describe the complete planning algorithm and demonstrate how specialized adaptations to traditional sample-based planners enable the generation of highly desirable real-time path solutions. Simulation analysis provide empirical evidence supporting the effectiveness of our approach. Our initial work in this chapter was published in the 20th Conference on Robots and Vision (CRV) [51], and was further expanded upon in this thesis. Chapter 4 delves into the details of our proposed control modules based on MPC. We present two main MPC implementations: the direct tracking approach, which directly tracks the path planner's output while enforcing kinematic constraints, and the homotopy guided method, which formulates a constrained, convex sub-problem based on the current path planner solution, allowing the MPC to find its own globally optimal solution. A comprehensive evaluation of these architectures is performed in Chapters 4 and 5, revealing the superior reliability and minimal path-following errors achieved by the homotopy guided MPC, making

it the most promising technique for the VT&R3 application.

Although our presented experiments primarily utilize the ARGO Atlas J8 robot platform within the teach-and-repeat framework, we are pleased to report successful deployment of our motion planning algorithms on a Clearpath Robotics Grizzly robot running VT&R3 [51] and an autonomous boat for environmental monitoring that follows a global path composed of GPS waypoints [94]. These demonstrations highlight the versatility and generalizability of our approach across a wider range of robots and path-following applications.

6.2 Future Work

In this section, we discuss potential future directions to improve or extend upon the work presented in Chapters 3, 4, and 5.

6.2.1 Visual Teach & Repeat 3 Upgrades

Recovery Behaviours

The path planner generates solutions from the robot's current state to the end of the desired path. However, this default behaviour may lead to issues in scenarios where obstacles are not detected until the robot is very close to them, such as blind corners. Under these conditions, the path planner will generate a collision free solution around the obstacle; however, it may require very rough kinematic trajectories to successfully execute. To address this, incorporating high-level "recovery behaviours" in the control scheme can be beneficial. For example, including a short backup operation of 1-2 m followed by replanning could allow us to generate safer trajectories around obstacles.

Perception Improvements

While the obstacle detection system used in this thesis relies on LiDAR change detection and a geometric approach, there is room for improvement. The system is sensitive to tuning parameters and tends to produce false positives that require the use of a temporal filter to remove. Unfortunately, this introduces delay to the system that impedes reactions to newly observed and dynamic obstacles. Furthermore, LiDAR occlusions and incomplete observations pose challenges in motion planning, requiring us to impose an obstacle inflation to compensate. Future work could explore the integration of deep learning techniques to enhance obstacle detection, potentially replacing or augmenting the existing perception stack in VT&R3. Learning LiDAR change detection directly could better leverage the teach-and-repeat structure to improve performance in challenging environments, such as dense vegetation, where typical single shot terrain classification methods tend to struggle to meet the accuracy requirements of a reliable off-road navigation systems.

Another limitation of the current change detection approach is its inability to detect long-term static structures that exist off the main path, but still constrain the size of the safe configuration space

and need to be considered. In our implementation, we address this issue by incorporating a user-labelled path-dependant maximum corridor width. This ensures that the planner remains within the corridor while avoiding other obstacles in tight scenes. However, to enhance safety measures, it would be beneficial to supplement this approach with a secondary, simple planar obstacle detection system using LiDAR. This additional module would then automatically fill in the gaps in the curvilinear domain corridor constraints, providing a more comprehensive and robust obstacle-avoidance strategy.

6.2.2 Dynamic Obstacle Avoidance

Temporal Planning and Control

While the proposed path planner does demonstrate responsiveness to dynamic obstacles, its reactive nature can lead to non-ideal trajectories as obstacles move and the optimal solution changes. To address this, incorporating temporal planning and control techniques can be beneficial. By anticipating the movement of dynamic obstacles, the planner can reduce the number of replanning cycles and collaborate more effectively with other dynamic agents.

Fortunately, the proposed motion planning architecture is well suited for this adaptation in future work. BIT* has been demonstrated to perform admirably in higher dimensional planning spaces [95], and it is likely that adding a temporal dimension to the curvilinear configuration space would scale well to real-time operation. Our collision checking scheme is also malleable to temporal collision checks. Currently, when we collision check our planning edges, we convert the curvilinear points to Euclidean space and query a 2D OGM. In a curvilinear space with a temporal dimension, the approach is the same, however in this case we propose the use of a so called Spatiotemporal Occupancy Grid Map (SOGM) [96] that can be queried in both time and space for collision checking. With a perception module designed to estimate the SOGM's and a temporal axis in the planning domain, no other architectural or algorithmic changes would be required to accommodate this upgrade.

Bibliography

[1] P. Furgale and T. D. Barfoot, "Visual teach and repeat for long-range rover autonomy," *Journal of Field Robotics*, vol. 27, no. 5, pp. 534–560, 2010.

[2] M. Paton, K. MacTavish, M. Warren, and T. D. Barfoot, "Bridging the appearance gap: Multi-experience localization for long-term visual teach and repeat," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE Press, 2016, p. 1918–1925.

[3] M. Gridseth and T. D. Barfoot, "Keeping an eye on things: Deep learned features for long-term visual localization," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1016–1023, 2022.

[4] C. McManus, P. Furgale, B. Stenning, and T. D. Barfoot, "Visual teach and repeat using appearance-based lidar," in *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 389–396.

[5] P. Krüsi, B. Bücheler, F. Pomerleau, U. Schwesinger, R. Siegwart, and P. Furgale, "Lighting-invariant adaptive route following using iterative closest point matching," *Journal of Field Robotics*, vol. 32, 06 2014.

[6] A. Cherubini and F. Chaumette, "Visual navigation of a mobile robot with laser-based collision avoidance," *The International Journal of Robotics Research*, vol. 32, pp. 189–205, Feb. 2013.

[7] Y. Matsumoto, M. Inaba, and H. Inoue, "Visual navigation using view-sequenced route representation," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 1, 1996, pp. 83–88 vol.1.

[8] A. Howard, "Multi-robot mapping using manifold representations," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, vol. 4, 2004, pp. 4198–4203 Vol.4.

[9] J. Marshall and T. Barfoot, "Design and field testing of an autonomous underground tramming system," *Springer Tracts in Advanced Robotics*, vol. 42, 07 2007.

[10] J. Marshall, T. Barfoot, and J. Larsson, "Autonomous underground tramming for center-articulated vehicles," *J. Field Robotics*, vol. 25, pp. 400–421, 06 2008.

[11] G. Sibley, C. Mei, I. Reid, and P. Newman, "Vast-scale outdoor navigation using adaptive relative bundle adjustment," *I. J. Robotic Res.*, vol. 29, pp. 958–980, 06 2010.

[12] M. Paton, F. Pomerleau, K. MacTavish, C. Ostafew, and T. Barfoot, "Expanding the limits of vision-based localization for long-term route-following autonomy," *Journal of Field Robotics*, vol. 34, 10 2016.

[13] M. Paton, K. MacTavish, C. J. Ostafew, and T. D. Barfoot, "It's not easy seeing green: Lighting-resistant stereo visual teach & repeat using color-constant images," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 1519–1526.

[14] M. Paton, F. Pomerleau, and T. D. Barfoot, "Eyes in the back of your head: Robust visual teach & repeat using multiple stereo cameras," in *2015 12th Conference on Computer and Robot Vision*, 2015, pp. 46–53.

[15] K. MacTavish, M. Paton, and T. D. Barfoot, "Visual triage: A bag-of-words experience selector for long-term visual route following," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 2065–2072.

[16] C. D. McKinnon and A. P. Schoellig, "Experience-based model selection to enable long-term, safe control for repetitive tasks under changing conditions," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 2977–2984.

[17] ——, "Learn fast, forget slow: Safe predictive learning control for systems with unknown, changing dynamics performing repetitive tasks," *CoRR*, vol. abs/1810.06681, 2018. [Online]. Available: http://arxiv.org/abs/1810.06681

[18] M. Warren, M. Greeff, B. Patel, J. Collier, A. P. Schoellig, and T. D. Barfoot, "There's no place like home: Visual teach and repeat for emergency return of multirotor UAVs during gps failure," *IEEE Robotics and Automation Letters*, vol. 4, no. 1, pp. 161–168, 2019.

[19] M. Bianchi and T. D. Barfoot, "UAV localization using autoencoded satellite images," *CoRR*, vol. abs/2102.05692, 2021. [Online]. Available: https://arxiv.org/abs/2102.05692

[20] L. Clement, J. Kelly, and T. Barfoot, "Robust monocular visual teach and repeat aided by local ground planarity and color-constant imagery," *Journal of Field Robotics*, vol. 34, 06 2016.

[21] Y. Wu, "Vt&r3: Generalizing the teach and repeat navigation framework," *University of Toronto Institute for Aerospace Studies MASc Thesis*, 09 2022.

[22] K. Burnett, Y. Wu, D. J. Yoon, A. P. Schoellig, and T. D. Barfoot, "Are we ready for radar to replace lidar in all-weather mapping and localization?" *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 328–10 335, 2022.

[23] T. Nguyen, G. Mann, R. Gosine, and A. Vardy, "Appearance-based visual-teach-and-repeat navigation technique for micro aerial vehicle," *Journal of Intelligent & Robotic Systems*, vol. 84, 12 2016.

[24] T. Krajník, P. De Cristóforis, K. Kusumam, P. Neubert, and T. Duckett, "Image features for visual teach-and-repeat navigation in changing environments," *Robotics and Autonomous Systems*, vol. 88, 11 2016.

[25] T. Swedish and R. Raskar, "Deep visual teach and repeat on path networks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018, pp. 1614–161 409.

[26] L. G. Camara, T. Pivoňka, M. Jílek, C. Gäbert, K. Košnar, and L. Přeučil, "Accurate and robust teach and repeat navigation by visual place recognition: A cnn approach," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 6018–6024.

[27] P. Krüsi, P. Furgale, M. Bosse, and R. Siegwart, "Driving on point clouds: Motion planning, trajectory optimization, and terrain assessment in generic nonplanar environments: Driving on point clouds," *Journal of Field Robotics*, vol. 34, 12 2016.

[28] D. Baril, S. Deschênes, O. Gamache, M. Vaidis, D. LaRocque, J. Laconte, V. Kubelka, P. Giguère, and F. Pomerleau, "Kilometer-scale autonomous navigation in subarctic forests: challenges and lessons learned," *CoRR*, vol. abs/2111.13981, 2021. [Online]. Available: https://arxiv.org/abs/2111.13981

[29] S. Quinlan and O. Khatib, "Elastic bands: connecting path planning and control," in *[1993] Proceedings IEEE International Conference on Robotics and Automation*, 1993, pp. 802–807 vol.2.

[30] M. Mattamala, N. Chebrolu, and M. Fallon, "An efficient locally reactive controller for safe navigation in visual teach and repeat missions," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2353–2360, 2022.

[31] R. Finman, T. Whelan, M. Kaess, and J. J. Leonard, "Toward lifelong object segmentation from change detection in dense RGB-D maps," in *2013 European Conference on Mobile Robots*, 2013, pp. 178–185.

[32] R. Ambrus, J. Ekekrantz, J. Folkesson, and P. Jensfelt, "Unsupervised learning of spatial-temporal models of objects in a long-term autonomy scenario," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 5678–5685.

[33] M. Fehr, F. Furrer, I. Dryanovski, J. Sturm, I. Gilitschenski, R. Siegwart, and C. Cadena, "TSDF-based change detection for consistent long-term dense reconstruction and dynamic object discovery," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 5237–5244.

[34] D. Girardeau-Montaut and R. Marc, "Change detection on points cloud data acquired with a ground laser scanner," in *2005 ISPRS Workshop*, 2005.

[35] H. Andreasson, M. Magnusson, and A. Lilienthal, "Has somethong changed here? autonomous difference detection for security patrol robots," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007, pp. 3429–3435.

[36] P. N. Trujillo, P. L. J. Drews-Jr, R. P. Rocha, M. F. M. Campos, and J. Dias, "Novelty detection and 3d shape retrieval based on gaussian mixture models for autonomous surveillance

robotics," *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4724–4730, 2009.

[37] A. W. Vieira, P. L. J. Drews, and M. F. M. Campos, "Efficient change detection in 3d environment for autonomous surveillance robots based on implicit volume," in *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 2999–3004.

[38] L.-P. Berczi and T. D. Barfoot, "It's like déjà vu all over again: Learning place-dependent terrain assessment for visual teach and repeat," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 3973–3980.

[39] L.-P. Berczi and T. Barfoot, "Looking high and low: Learning place-dependent gaussian mixture height models for terrain assessment," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 3918–3925.

[40] K. Nagla, M. Uddin, and D. Singh, "Dedicated filter for robust occupancy grid mapping," *International Journal of Robotics and Automation*, vol. 4, pp. 82–92, 03 2015.

[41] D. Arbuckle, A. Howard, and M. Mataric, "Temporal occupancy grids: a method for classifying the spatio-temporal properties of the environment," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, 2002, pp. 409–414 vol.1.

[42] A. Šelek, M. Seder, M. Brezak, and I. Petrović, "Smooth complete coverage trajectory planning algorithm for a nonholonomic robot," *Sensors*, vol. 22, no. 23, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/23/9269

[43] A. Said, R. Talj, C. Francis, and H. Shraim, "Local trajectory planning for autonomous vehicle with static and dynamic obstacles avoidance," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, 2021, pp. 410–416.

[44] T. Faulwasser and R. Findeisen, "Nonlinear model predictive control for constrained output path following," *arXiv.org*, vol. 61, 02 2015.

[45] J. Ji, A. Khajepour, W. W. Melek, and Y. Huang, "Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 2, pp. 952–964, 2017.

[46] Z. Dong, X. Xu, X. Zhang, X. Zhou, X. Li, and X. Liu, "Real-time motion planning based on mpc with obstacle constraint convexification for autonomous ground vehicles," in *2020 3rd International Conference on Unmanned Systems (ICUS)*, 2020, pp. 1035–1041.

[47] B. Lindqvist, S. Sharif Mansouri, A.-A. Agha-Mohammadi, and G. Nikolakopoulos, "Nonlinear MPC for collision avoidance and control of UAVs with dynamic obstacles," *IEEE Robotics and Automation Letters*, vol. PP, pp. 1–1, 07 2020.

[48] A. Sathya, P. Sopasakis, R. Van Parys, A. Themelis, G. Pipeleers, and P. Patrinos, "Embedded nonlinear model predictive control for obstacle avoidance using panoc," in *2018 European Control Conference (ECC)*, 2018, pp. 1523–1528.

[49] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1:43 scale rc cars," *Optimal Control Applications and Methods*, vol. 36, pp. 628–647, 09 2015.

[50] V. Z. Patterson, F. E. Lewis, and J. C. Gerdes, "Optimal decision making for automated vehicles using homotopy generation and nonlinear model predictive control," in *2021 IEEE Intelligent Vehicles Symposium (IV)*, 2021, pp. 1045–1050.

[51] J. Sehn, Y. Wu, and T. D. Barfoot, "Along similar lines: Local obstacle avoidance for long-term autonomous path following," 2022.

[52] P. Marín, A. Hussein, D. Martín Gómez, and A. de la Escalera, "Global and local path planning study in a ros-based research platform for autonomous vehicles," *Journal of Advanced Transportation*, vol. 2018, pp. 1–10, 02 2018.

[53] X. Mai, D. Li, J. Ouyang, and Y. Luo, "An improved dynamic window approach for local trajectory planning in the environment with dense objects," *Journal of Physics: Conference Series*, vol. 1884, p. 012003, 04 2021.

[54] M. Kobayashi and N. Motoi, "Local path planning: Dynamic window approach with virtual manipulators considering dynamic obstacles," *IEEE Access*, vol. 10, pp. 1–1, 01 2022.

[55] C. Rösmann, F. Hoffmann, and T. Bertram, "Integrated online trajectory planning and optimization in distinctive topologies," *Robotics and Autonomous Systems*, vol. 88, 11 2016.

[56] ——, "Kinodynamic trajectory optimization and control for car-like robots," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 5681–5686.

[57] X. Meng, Y. Xiang, and D. Fox, "Learning composable behavior embeddings for long-horizon visual navigation," *CoRR*, vol. abs/2102.09781, 2021. [Online]. Available: https://arxiv.org/abs/2102.09781

[58] G. Kumar, N. S. Shankar, H. Didwania, R. Roychoudhury, B. Bhowmick, and K. M. Krishna, "Gcexp: Goal-conditioned exploration for object goal navigation," in *2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*, 2021, pp. 123–130.

[59] D. Dugas, O. Andersson, R. Siegwart, and J. J. Chung, "Navdreams: Towards camera-only RL navigation among humans," 2022.

[60] A. Sadek, G. Bono, B. Chidlovskii, and C. Wolf, "An in-depth experimental study of sensor usage and visual reasoning of robots navigating in real environments," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 9425–9431.

[61] R. Bellman, "On the theory of dynamic programming." *Proceedings of the National Academy of Sciences of the United States of America*, vol. 38 8, pp. 716–9, 1952.

[62] J. Park, S. Karumanchi, and K. Iagnemma, "Homotopy-based divide-and-conquer strategy for optimal trajectory planning via mixed-integer programming," *IEEE Transactions on Robotics*, vol. 31, pp. 1–15, 10 2015.

[63] S. Bhattacharya and R. Ghrist, "Path homotopy invariants and their application to optimal trajectory planning," *CoRR*, vol. abs/1710.02871, 2017. [Online]. Available: http://arxiv.org/abs/1710.02871

[64] S. Bhattacharya and M. Likhachev, "Search-based path planning with homotopy class constraints." vol. 2, 12 2010.

[65] P. E. Hart, N. J. 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, pp. 100–107, 1968.

[66] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *CoRR*, vol. abs/1105.1186, 2011. [Online]. Available: http://arxiv.org/abs/1105.1186

[67] L. Janson and M. Pavone, "Fast marching trees: a fast marching sampling-based method for optimal motion planning in many dimensions - extended version," *CoRR*, vol. abs/1306.3532, 2013. [Online]. Available: http://arxiv.org/abs/1306.3532

[68] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[69] A. Stentz, "The d* algorithm for real-time planning of optimal traverses," 1994.

[70] P. Lajevardy, A. mousavian, and M. Oskoei, "A comparison between rrt* and a* algorithms for motion planning in complex environments," 08 2015.

[71] H. Zhou, P. Feng, and W. Chou, "A hybrid obstacle avoidance method for mobile robot navigation in unstructured environment," *Industrial Robot*, vol. ahead-of-print, pp. ahead–of–print, 2022.

[72] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 2997–3004.

[73] A. Al-Moadhen, H. Kamil, and A. Khayeat, "Superting BIT*-based path planning with mpc-based motion planning of the self-driving car," *Int J Intell Syst Appl Eng*, vol. 10, no. 3, pp. 214–226, dec 2022.

[74] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Batch informed trees (BIT): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, may 2015.

[75] S. Koenig, M. Likhachev, and D. Furcy, "Lifelong planning a star," *Artificial Intelligence*, vol. 155, pp. 93–146, 05 2004.

[76] T. Barfoot and C. Clark, "Motion planning for formations of mobile robots," *Robotics and Autonomous Systems*, vol. 46, no. 2, pp. 65–78, 2004.

[77] P. Dahal, S. Mentasti, S. Arrigoni, F. Braghin, M. Matteucci, and F. Cheli, "Extended object tracking in curvilinear road coordinates for autonomous driving," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 2, pp. 1266–1278, 2023.

[78] M. Massaro and R. Lot, "A virtual rider for two-wheeled vehicles," 12 2010, pp. 5586–5591.

[79] J. Schlechtriemen, K. P. Wabersich, and K.-D. Kuhnert, "Wiggling through complex traffic: Planning trajectories constrained by predictions," in *2016 IEEE Intelligent Vehicles Symposium (IV)*, 2016, pp. 1293–1300.

[80] 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)*, 2019, pp. 64–69.

[81] P. Orukpe, "Model predictive control fundamentals," *Nigerian Journal of Technology*, Jul 2012.

[82] W. F. Lages and J. A. Vasconcelos Alves, "Real-time control of a mobile robot using linearized model predictive control," *IFAC Proceedings Volumes*, vol. 39, no. 16, pp. 968–973, 2006, 4th IFAC Symposium on Mechatronic Systems. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474667015342944

[83] K. Mondal, A. Rodriguez, S. Manne, N. Das, and B. Wallace, "Comparison of kinematic and dynamic model based linear model predictive control of non-holonomic robot for trajectory tracking: Critical trade-offs addressed," 12 2019.

[84] L. Stella, A. Themelis, P. Sopasakis, and P. Patrinos, "A simple and efficient algorithm for nonlinear model predictive control," 2017.

[85] Y. Li, X. Xu, H. Yang, H. Zhang, Q. Li, X. Wang, Z. Wang, H. Wang, S. Xu, A. Rodic, and P. B. Petrovic, "A novel hybrid path planning method for mobile robots," in *2022 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2022, pp. 821–826.

[86] P. Xu, N. Wang, S.-L. Dai, and L. Zuo, "Motion planning for mobile robot with modified bit* and mpc," *Applied Sciences*, vol. 11, p. 426, 01 2021.

[87] S. Teng, D. Chen, W. Clark, and M. Ghaffari, "An error-state model predictive control on connected matrix lie groups for legged robot control," 2023.

[88] D. E. Chang, K. S. Phogat, and J. Choi, "Model predictive tracking control for invariant systems on matrix lie groups via stable embedding into euclidean spaces," *IEEE Transactions on Automatic Control*, vol. 65, no. 7, pp. 3191–3198, 2020.

[89] C. J. Ostafew, A. P. Schoellig, and T. D. Barfoot, "Learning-based nonlinear model predictive control to improve vision-based mobile robot path-tracking in challenging outdoor environments," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 4029–4036.

[90] C. Ostafew, A. Schoellig, and T. Barfoot, "Robust constrained learning-based nmpc enabling reliable mobile robot path tracking," *The International Journal of Robotics Research*, vol. 35, 05 2016.

[91] T. D. Barfoot, *State Estimation for Robotics*. Cambridge University Press, 2017.

[92] S. Anderson and T. D. Barfoot, "Full steam ahead: Exactly sparse gaussian process regression for batch continuous-time trajectory estimation on se(3)," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 157–164.

[93] A. Schiela, "Barrier methods for optimal control problems with state constraints," *SIAM Journal on Optimization*, vol. 20, no. 2, pp. 1002–1031, 2009. [Online]. Available: https://doi.org/10.1137/070692789

[94] Y. Huang, H. Dugmag, T. D. Barfoot, and F. Shkurti, "Stochastic planning for asv navigation using satellite images," 2023.

[95] J. D. Gammell and M. P. Strub, "Asymptotically optimal sampling-based motion planning methods," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, no. 1, pp. 295–318, may 2021.

[96] H. Thomas, M. G. de Saint Aurin, J. Zhang, and T. D. Barfoot, "Learning spatiotemporal occupancy grid maps for lifelong navigation in dynamic scenes," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 484–490.