Trajectory Optimization for Autonomous Navigation in Dynamic Environments

B.C. Floor

Master of Science Thesis

Delft University of Technology

Cognitive Robotics Department

Trajectory Optimization for Autonomous Navigation in Dynamic Environments

Master of Science Thesis

For obtaining the double degree of Master of Science in
Systems and Control & Embedded Systems
at Delft University of Technology

B.C. Floor

November 20, 2018

Faculty of Mechanical, Maritime and Materials Engineering (3mE)
Faculty of Electrical Engineering, Mathematics and Computer Science (EWI)
· Delft University of Technology

Delft University of Technology

Cognitive Robotics

Delft University of Technology
Department of
Cognitive Robotics (CoR)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE)
Faculty of Electrical Engineering, Mathematics and Computer Science (EWI)
for acceptance a thesis entitled

Trajectory Optimization for Autonomous Navigation in Dynamic Environments

by

B.C. Floor

in partial fulfillment of the requirements for the degree of

Master of Science
Systems and Control & Embedded Systems

Dated: November 20, 2018

Supervisor: ______________________
Dr. J. Alonso-Mora

Committee members: ______________________
Prof. Dr. R. Babuška

____
Dr. ir. C.J.M. Verhoeven

Abstract

Motion planning for Autonomous Ground Vehicles (AGVs) in dynamic environments is an extensively studied and complex problem. State of the art methods provide approximate solutions that make conservative assumptions to provide safety and feasibility. We aim to outperform current methods by following a trajectory optimization-based approach, providing a Local Model Predictive Contouring Control framework. Our method allows AGVs to execute reactive motion while tracking a locally parametrized reference path, anticipating on the predicted evolution of the environment. Given the static environment configuration in an occupancy grid map and dynamic obstacles represented by ellipses, we formulate explicit collision avoidance constraints. Well-informed planning decisions are made through a cost function with trade-offs between competing performance variables such as tracking accuracy, maintaining the reference velocity, and clearance from obstacles.

An efficient implementation of the method is presented that satisfies the real-time constraint of online navigation tasks. Furthermore, we present an implementation of a complete navigation system to emphasize our ability to deal with real sensor data and onboard processing. We show that the general definition of the framework applies to both unicycle and bicycle kinematic models, commonly used to represent mobile robots and autonomous cars, respectively. Simulation results for a car and experimental results with a mobile robot show that our method is a feasible and scalable approach. Proposed improvements of the method include *1)* considering obstacle velocities and positioning with respect to the AGV in the penalty term that creates clearance, *2)* incorporating prediction uncertainty of obstacles, and *3)* improving our method that deals with infeasible solutions of the optimal control problem.

Table of Contents

Preface **ix**

1 Introduction **1**

1-1 Research Aim . . . 3

1-2 Thesis Report Content and Outline . . . 4

2 Scientific Paper **5**

3 Discussion and Conclusion **21**

3-1 Discussion . . . 21

3-1-1 Local Reference Path Parametrization . . . 22

3-1-2 Collision Avoidance . . . 22

3-1-3 Experimental results . . . 23

3-1-4 Scalability of the Method . . . 24

3-2 Conclusion . . . 25

3-3 Recommendations for Future Work . . . 26

A Additional Experimental Results **29**

A-1 Comparing the Two Variations of Static Obstacle Avoidance . . . 29

A-1-1 Interpretation of Planning Behavior . . . 29

A-1-2 Comparison of Computational Effort . . . 31

A-2 Robustness of Static Environment Collision Avoidance . . . 31

B Implementation of the Optimal Controller Framework **33**

B-1 Designing the ROS Package . . . 33

B-1-1 Linking the MoveIt! Plugin . . . 35

B-1-2 Computing the Local Reference Path . . . 35

B-1-3 Search Routine of the Static Environment Collision Constraint . . . 37

B-1-4 `RViz` Visualization . . . 42

B-1-5 Dynamic Reconfigure Server . . . 42

B-2 Solving the Optimal Control Problem with `ACADO` . . . 43

C Experimental Setup **47**
C-1 Mobile Robot 47
C-1-1 Skid-Steering Locomotion 48
C-1-2 Equipment of the Mobile Robot 48
C-1-3 Kinematic Motion Model 52
C-1-4 Motor Control Scheme 54
C-2 Setting up the Experiment 55

D Dealing with Obstacle Prediction Uncertainty **57**
D-1 Method 57
D-2 Preliminary Results 58
D-3 Conclusion 60

E Derivation of the Minkowski Sum Bounding Ellipse **63**

F Submitted Conference Paper **67**

Bibliography **75**

Glossary **81**
List of Acronyms 81
List of Symbols 82

List of Figures

1-1 Complexity of advanced transport systems expressed in lines of code [1]. 1

1-2 Example service robot applications ***(left)*** package delivery[1] ***(right)*** Surveillance[2]. 2

A-1 Equivalent simulated navigation scenario for both constraint variations. 30

A-2 Search times of the two constraint variations as a function of the search distance. 31

A-3 Simulation results of traversing an environment with a static barrier on the reference path. 32

A-4 Simulation results of traversing an environment with static obstacles on the reference path. 32

B-1 Occupancy grid map representation of the environment with incrementally expanded search boundaries defined in the local frame. 38

B-2 Incremental search for the circular collision free area in an inflated occupancy grid map with the Autonomous Ground Vehicle heading ***(left)*** aligned with the grid and ***(right)*** not aligned with the grid. 38

B-3 Parametrization of the rectangular collision free area. 40

B-4 `RViz` visualization. 42

B-5 Dynamic reconfigure server interface. 43

C-1 Jackal robotics platform, designed by Clearpath™ Robotics. 47

C-2 Snapshot of the onboard controller board. 48

C-3 Human Machine Interface panel. 49

C-4 CAD drawing of the designed sensor mount 51

C-5 Schematic of the onboard power distribution circuit. 51

C-6 Joystick controls layout. 51

C-7 Free body diagram of a four-wheeled skid-steering Autonomous Ground Vehicle with generalized planar velocities [2]. 52

C-8 Control scheme of the actuating motors. 54

C-9 Communication diagram of the two experimental setups. 55
C-10 Constructed navigation scenario in the OptiTrack environment. 55

D-1 Visualization of the resulting constraint-ellipse. 58
D-2 Uncertainty growth of the obstacle position for different p_ϵ. 58
D-3 Simulated overtaking scenario with prediction uncertainty on the obstacle position. 59
D-4 Simulated pedestrian avoidance scenario with prediction uncertainty on the obstacle positions. 60

List of Tables

B-1 Number of required Online Data Variables per prediction step. 35
B-2 Explained aspects of the RViz environment. 43
B-3 Generated C files by ACADO [3]. 45

C-1 Computer specifications. 49
C-2 Sensor specifications. 50

Preface

A year ago, I was offered the opportunity to start a graduation project in robotics. I embraced the project gratefully and I am still considering this as one of the best decisions in my life. I am passionate about developments that contribute to a world where robots are becoming part of our daily lives. Perfectly in line with that passion, my research contributes to the complex task of planning motion for autonomous ground vehicles in dynamic environments. It excites me that robotics is a highly competitive and diverse research area, demanding me to consult all the knowledge obtained during my academic career.

Being able to work among the highly inspiring people of the Department of Cognitive Robotics (CoR) brought me a lot of satisfaction and has been an important motivation for me. Simultaneously, I have been privileged to have access to tools during my thesis that allowed me to bring my theoretical research into practice. I have spent many hours and late evenings in the lab getting acquainted with the robot and other equipment to ensure that I would make use of all the possibilities to their full extent. The practical experience that I have been able to acquire is of great value to me.

The thesis is carried out under the supervision of Dr. Javier Alonso-Mora to obtain the MSc. degree in both Systems and Control and Embedded Systems. Even though I usually tend to struggle to perform according to my personal satisfaction thresholds, I am honestly proud to conclude my academic journey of six years at the Delft University of technology with this thesis.

First and foremost, I would like to thank Javier for his excellent guidance during my thesis work. I greatly admire how you combine your expertise and professionalism with your kind and honest attitude. Meanwhile, Bruno has been a great mentor to me and inspired me to aim for ambitious achievements. Our collaboration is very valuable to me and taught me to have a high level of respect for your personality.

Knowing that I have requested a huge amount of empathy, I owe a big debt of gratitude to Thirsa, who has been standing by my side during the whole process. Also, I want to thank my parents and all of my family and friends that have been understanding and supportive, even though my effort of staying in contact with them might have experienced a global minimum during my study.

Now that you, the reader, managed to get to the bottom of this page, I thank you for your interest in my work and I kindly invite you to continue reading.

Boaz Floor
Delft, November 20, 2018

Chapter 1

Introduction

Nowadays, autonomous navigation of numerous types of vehicles is the subject of research for many companies and research institutions [4]. Driven by safety [5], road infrastructure efficiency, pollution reduction [6, 7], and connected mobility [8, 9], development of intelligent road vehicles is progressing at a fast pace [10, 11]. Additionally, autonomous driving can provide a new transportation system to non-driving demographic groups [12, 13]. Essential challenges remain yet to be solved [14] and it is not yet clear how autonomous vehicles will be deployed in the future [15, 16, 17]. On top of that, autonomous systems are increasingly becoming more complex and vulnerable to system failures that can cause severe accidents, which is emphasized by Fig. 1-1 [18, 19]. It is, however, widely accepted that autonomous vehicles have the potential to introduce a new transport paradigm and that their impact will be significant in our daily lives [20, 21].

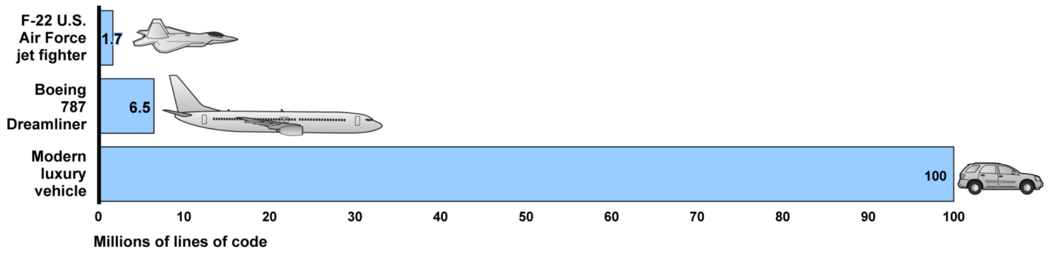


Figure 1-1: Complexity of advanced transport systems expressed in lines of code [1].

Concurrently, research on autonomous vehicles has diverged from the well-structured road environments, considering cluttered, off-road, unknown and unstructured environments [22, 23, 24]. These characteristics do particularly appear in environments that mobile robot platforms have to operate in. The employment of mobile robots is mainly covered by the class of service robots of which two examples are given in Fig. 1-2. Service robots cover a wide range of application areas such as package delivery, security, agriculture, personal assistance, construction, health care, entertainment, and cleaning. Being triggered by mobile robot competitions [25] and industrial applications [26], developments in mobile robot applications enhance the research of required and related research areas. According to a recent market research report, the global market of service robotics was valued at USD 10.36 billion in 2017 is expected to reach USD 28.65 billion by 2023, at a compound annual growth rate of 17.9% between 2018 and 2023 [27].

Figure 1-2: Example service robot applications ***(left)*** package delivery[1] ***(right)*** Surveillance[2].

In order to be able to perform real-world tasks with mobile robots or autonomous cars, autonomous navigation is one of the key features that is, throughout all applications, persistently required for them to operate properly. Autonomous Ground Vehicles (AGVs) will operate in environments where they have to interact intensively with humans and they will have to deal with an open and uncertain world. Typically, outside of closed and controlled industrial environments, the online navigation task only has onboard sensor-based information available, resulting in limited knowledge of the environment [28]. As a result of the limited view through its sensors, an AGV has to integrate over consecutive sensor measurements to build a model of its environment [29]. Correct data association is essential, even though the perception systems through which we receive information about the precarious environment introduces new uncertainties [30]. Following a modular approach, it is mentioned by [31] that the autonomous navigation problem can roughly be divided into three main research topics:

- Mapping of the environment and localization within that environment.
- Detection, characterization, and behavior prediction of the potential moving obstacles.
- Online motion planning and behavioral decision making.

Partially due to its complexity and importance within the navigation task, the field of motion planning has been extensively studied [32, 33]. Moreover, the motion planning problem is fundamental to the field of robotics and has a long history [34]. The off-line motion planning problem, which assumes full knowledge of the environment, is generally well understood [35]. In real-world applications of autonomous vehicles, however, these methods are not applicable due to their incapability of dealing with (partially) unknown environments, uncertainty and dynamic behavior of obstacles. Many solutions have already been proposed and implemented, increasingly yielding better performance in real-world applications [36, 37, 38]. The complex nature of the problem resulted in the development of approximate solutions or solutions that

[1]https://www.cnet.com/news/startup-bets-its-wheeled-robots-not-airborne-drones-will-deliver-your-groceries/ (accessed October 28, 2018)
[2] https://securitybrief.co.nz/story/advanced-security-brings-autonomous-surveillance-robot-nz-market (accessed November 11, 2018)

aim to solve different subsets of the motion planning problem. Successful methods decompose or discretize the continuous workspace in a smart way to obtain a tractable problem [36, 39]. Often, conservative assumptions have to be made to achieve feasibility and safety guarantees [10]. Therefore, it is still a challenging problem to plan collision-free, time-efficient and optimal trajectories for an AGV in unstructured dynamic environments.

To overcome the increasing dimensionality of the planning problem, some real-time approaches treat dynamic environments as static and use replanning techniques as dynamic obstacles move [40, 41, 42]. Under the same static assumption, [43] takes motion primitives into account in the planning stage to obtain smooth and achievable trajectories. However, by not anticipating on the dynamic behavior of the objects, valuable information about the environment is lost. Reactive, control-oriented methods such as Artificial Potential Field methods [44] or the Dynamic Window Approach [45] provide promising solutions for collision avoidance but lack global convergence guarantees and advanced decision-making capabilities. Model Predictive Control (MPC) approaches account for the future evolution of the environment and of the vehicle state to generate anticipatory motions and compute the corresponding optimal control commands [46, 47]. MPC is a trajectory optimization-based control technique that allows dealing with dynamic and physical constraints while optimizing a desired performance index [48, 49]. A MPC strategy known as Model Predictive Contouring Control (MPCC) [50] allows one to track a reference path (rather than a trajectory parameterized in time) and explicitly penalize the deviation from it (in terms of contouring and lateral errors). We build on the concept Model Predictive Contouring Control to produce planning decisions based on the future evolution of the environment while creating reactive motion plans. Furthermore, this optimization-based approach allows us to explicitly define collision constraints and trade-offs between competitive performance variables in a cost function. We present a new planning framework that is generally applicable to any AGV. Simulation results for a car and experimental results with a mobile robot are presented.

1-1 Research Aim

Motion planning for Autonomous Ground Vehicle is an unresolved problem in general and a frequently researched topic in both industry and academia. Therefore, this thesis seeks to improve motion planning techniques in dynamic environments using a trajectory optimization approach. In particular, we will rely on a Model Predictive Contouring Control approach to develop a new planning framework. The research will be conducted by a design stage, implementation work, and experiments in both simulations and on a mobile robot for different vehicle types. The following research questions are set as guidelines to fulfill this objective.

1. How can the concept of Model Predictive Contouring Control be utilized to navigate autonomous vehicles, traversing real-world dynamic environments?

2. How can static and dynamic obstacle collision avoidance be incorporated in the motion planning stage?

3. How can the proposed motion planning framework be implemented and adopted within an autonomous navigation system such that it satisfies real-time operation?

1-2 Thesis Report Content and Outline

The main results will be covered in Chapter 2 of the thesis, where the main findings and contribution of the thesis work are presented in a scientific paper format. In Chapter 3, a more in-depth discussion and conclusion is made and recommendations for future research work are considered.

Supplementary content to the paper will be discussed in the appendices according to the following structure:

- Appendix A – Additional experimental results.
- Appendix B – A chapter about the implementation of our planning framework, its attributes and comments on the utilized software tools.
- Appendix C – The mobile robot and its equipment, simulation environment and experimental setup will be presented in this chapter.
- Appendix D – An extension to the planning framework to deal with dynamic obstacle uncertainty.
- Appendix E – The derivation of the Minkowski sum bounding ellipse for obstacle avoidance.
- Appendix F – Submitted conference paper.

Chapter 2

Scientific Paper

Local Model Predictive Contouring Control for Dynamic Environments

Abstract— **We present a local motion planner, namely, a Local Model Predictive Contouring Control design, for an Autonomous Ground Vehicle (AGV) traversing dynamic environments. Our design allows the AGV to execute reactive motion while tracking a global plan, thanks to the local parametrization of the path. In addition, our framework allows for avoidance of static obstacles (given in an occupancy grid map) and moving obstacles represented by ellipses. Furthermore, we provide a new bound to correct the approximation of the Minkowski sum of an ellipsoid obstacle and the union of discs representation of the controlled vehicle to guarantee collision avoidance. We show that the general definition of the framework applies to both unicycle and bicycle kinematic models, commonly used to represent robots and autonomous cars, respectively. Simulation results for a car and experimental results with a mobile robot are presented.**

I. INTRODUCTION

This paper proposes a general framework to safely navigate Autonomous Ground Vehicles (AGVs), such as mobile robots and cars (Fig. 1), in dynamic environments. Motion planning and control for AGVs is usually addressed as two separate problems (with the planner and controller running on two different modules) [1], [2]. In particular, the motion planner generates safe, smooth, and feasible paths, while the motion controller typically aims to track this planned path (directly acting on the AGV's actuator). Motion planning techniques, however, do not usually take into account that the path-following controller relies on the smoothness and kinodynamic feasibility of the reference path. This can compromise the safety of the vehicle when the controller is unable to follow the planned trajectory. Commonly, motion planning procedures rely on graph-search or on randomized sampling-based techniques [3]. To overcome the increasing dimensionality of the planning problem, some real-time approaches treat dynamic environments as static and use replanning techniques as dynamic obstacles move [4], [5], [6]. Under the same static assumption, [7] considers incorporating constraints in the planning stage to obtain smooth and achievable trajectories. By not anticipating on the dynamic behavior of the objects, valuable information about the environment is lost.

In order to generate appropriate motion in highly dynamic and uncertain environments, local and immediate motions become more important to avoid collisions. Therefore, reactive, control-oriented methods are often more promising. However, purely reactive methods such as Artificial Potential Field methods [8] or the Dynamic Window Approach [9] often lack global convergence guarantees and advanced decision-making capabilities. Motion planning techniques that do consider kinematic constraints as well as a changing environment such as [10], have to rely on sampling-based techniques to deal with the complexity of the problem. These sampling-based approaches result in jerky motion [11].

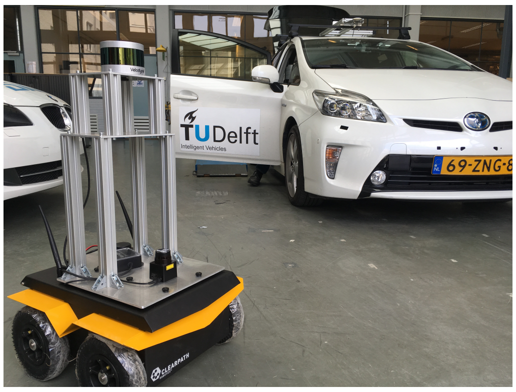


Fig. 1: Example AGVs: A mobile robot and an autonomous car.

We use a Model Predictive Control (MPC) approach to the trajectory planning problem to overcome the disconnection between the planning and control stage. This method allows us to compute safe, smooth and kinodynamic feasible trajectories, while simultaneously retrieving the corresponding optimal control inputs. Whereas path planning only considers the generation of a consecutive set of geometric specifications of the positions and orientations of a robot, trajectory planning parametrizes time in the problem formulation by including the generation of linear and angular velocities as well [12]. Furthermore, this controller approach allows us to explicitly define collision constraints and trade-offs between competing performance variables in a cost function.

A. Related Work

A method based on Stochastic trajectory optimization for motion planning (STOMP) [13], updates a candidate solution by evaluating the simulated cost of sampled trajectories around an initial guessed trajectory. Although this approach is successfully implemented, no guarantee is available on the convergence to a safe solution. FasTrack proposes a safe controller based on pre-computed safety bounds [14], [15]. Real-time trajectory planning and tracking is done using a set of simplified kinematic or dynamic planning models. The proposed method, however, can only deal with static obstacles. It is shown by [16] how continuous-time trajectories can be represented by a small number of states using sparse Gaussian process models. Next, they solve the motion planning problem through probabilistic inference on a factor graph. The downside of this approach is that obstacles are assumed to be static and that replanning is required when the environment changes.

A planning paradigm that incorporates the controller stage into the planning problem relies on Model Predictive Control (MPC). MPC is an optimization-based control technique that allows dealing with dynamic and physical constraints, while optimizing a desired performance index [17], [18]. MPC approaches allow to account for the future evolution of the environment and of the vehicle state to generate anticipatory motions and compute optimal control commands [19], [20]. The authors of [21] rely on a MPC scheme for cruise control and overtaking maneuvers. The authors of [22] design a MPC controller to ensure vehicle stability using differential braking and active steering. The aforementioned approaches, however, only tackle specific driving situations and do not consider the link between the global planner and local controller. Our approach addresses this issue by linking the global planner and the controller. Similarly to our approach, the authors of [23] build a clothoidal constrained path from irregular GPS waypoints and employ MPC to locally track the generated path. Collision avoidance constraints, however, are not taken into account.

A MPC strategy known as Model Predictive Contouring Control (MPCC) [24] allows one to track a reference path (rather than a trajectory parameterized in time) and explicitly penalize the deviation from it (in terms of contouring and lateral errors). Following the MPCC paradigm, the authors of [25] propose a Nonlinear MPCC (NMPCC) to handle static and dynamic obstacles for a driving scenario. They define the static collision constraints as deviation limits with respect to the reference path and assume constrained driving scenarios, such as highways. This assumption is too restrictive for mobile robots that usually navigate in unconstrained scenarios. [26] proposes to parametrize a reference path for MPCC using third-order spline polynomials. Their method, however, requires a full path to be computed beforehand, making it not applicable to realistic navigation scenarios. Several approaches have been presented that aim to incorporate obstacle avoidance into the MPC formulation. Specifically targeted at traversing unknown environments, [27] adopts a measure of obstacle thread in the cost function, using the parallax angle. [28] introduces the use of a repulsive potential to avoid obstacles in the environment. Both methods, however, only penalize being near obstacles and cannot guarantee obstacle avoidance. To solve this problem, [29] and [30] incorporate their collision avoidance strategy as inequality constraints on the AGV states. Both methods suffer from a limited feasible set of vehicle states since one local convex description of the collision-free area around the current vehicle position is used for the complete prediction horizon.

Efforts have been made to increase the computational efficiency of solving MPC problems for real-time solutions on embedded platforms, of which an overview is given by [31], [32]. One solution is an explicit approach, in which the solution for all possible problem instances is pre-computed, requiring a lot of memory. Explicit methods are limited to small-scale systems since the look-up table of solution can grow exponentially in the number of states. Contrary to the explicit approach, Interior-Point methods [33] are popular for providing solutions to a class of MPC problems online. Available open-source packages [34], [35] provide fast solvers for MPC problems, relying on Interior-Point methods. Another approach that gained attention for the computation of solutions online are active set methods [36]. [37] implements a fast Quadratic Programming (QP) solver, based on the online active set strategy [38], specifically targeted at MPC applications.

B. Contribution

We solve the motion planning and control problem by combining the strengths of global re-planning techniques with the reactive behavior of an MPC approach. In particular, we propose a new formulation, namely, a Local Model Predictive Contouring Control (LMPCC) approach, building on MPCC. By relying on [24], [25], this work makes the following contributions:

1) *A local formulation of the contouring control problem.* LMPCC supports global replanning without the need of creating a new full path parametrization. In contrast with the original MPCC scheme, LMPCC does not require an analytical path representation as a reference.
2) *A static obstacle avoidance strategy.* This strategy explicitly constrains the AGV's dynamics at time t along the prediction horizon to an approximation of the collision-free area around the robot. The approximation is obtained by exploiting the predicted behavior of the vehicle at time $t-1$.
3) *A bound to define dynamic collision avoidance constraints.* In particular, we correct the approximation of the Minkowski sum of the ellipsoid and a circle (previously used for dynamic collision avoidance) such that, if the constraints are satisfied, collision avoidance is guaranteed.
4) *Performance results in simulation (using a mobile robot and an autonomous car) and with real-world experiments (using a mobile robot).* Our strategy supports real sensor data and onboard localization. Furthermore, we show that our strategy can be fully implemented onboard the mobile robot in a real navigation scenario, as our experiments show, and run in real time.

The method relies on an open-source solver [39] and will be released.

II. PRELIMINARIES

Let B denote an Autonomous Ground Vehicle (AGV) on the plane $\mathcal{W}=\mathbb{R}^2$. B can be represented by the following discrete nonlinear system:

$$\boldsymbol{z}(t+1)=f(\boldsymbol{z}(t),\boldsymbol{u}(t)), \tag{1}$$

where $\boldsymbol{z}(t)$ and $\boldsymbol{u}(t)$ represent the state and the command at time $t \geq 0$ [1], respectively. The configuration of the AGV is denoted in configuration space $\mathcal{C}=\mathbb{R}^2\times\mathcal{S}$ by $\boldsymbol{z}(t)\in\mathcal{C}$. The area occupied by the AGV at state $\boldsymbol{z}$ is $\mathcal{B}(\boldsymbol{z})$. $\mathcal{B}(\boldsymbol{z})$

[1]In the remainder of the paper we omit the time dependency when it is clear from the context.

is approximated by a union of circles, that is, $\mathcal{B}(\boldsymbol{z}) \subseteq \bigcup_{c\in\{1,\dots,n_c\}} \mathcal{B}_c(\boldsymbol{z}) \subset \mathcal{W}$, where n_c is the number of circles.

We consider static and dynamic obstacles. In particular, the static obstacle environment is assumed to be captured in an occupancy grid map. The area occupied by the static obstacles is $\mathcal{O}^{\text{static}} \subset \mathcal{W}$. Furthermore, each moving obstacle A_i is represented by an ellipse of area $\mathcal{A}_i$. Consider a set of moving obstacles A_i with $i \in \mathcal{I} := \{1,\dots,n\}$ in $\mathcal{W}$, where n can vary over time. The area occupied by all moving obstacles at time instant t is given by $\mathcal{O}_t^{\text{dyn}} = \bigcup_{i\in\{1,\dots,n\}} \mathcal{A}_i(\boldsymbol{z}^{A_i}(t))$, where $\boldsymbol{z}^{A_i}(t)$ denotes the state of A_i at time t. The size of an ellipsoid associated with the obstacle is defined by a and b, the semi-major and semi-minor axes of the ellipse, respectively.

The objective is to generate collision-free motion for B through $\mathcal{W}$, from its current state to a desired end configuration. After parametrizing a reference path, this can be formulated as an optimization problem as follows:

$$J^* = \min_{\boldsymbol{z}_{0:N},\boldsymbol{u}_{0:N-1}} \sum_{k=0}^{N-1} J(\boldsymbol{z}_k,\boldsymbol{u}_k,\theta_k) + J(\boldsymbol{z}_N,\theta_N) \tag{2a}$$

$$\text{s.t.} \quad \boldsymbol{z}_{k+1} = f(\boldsymbol{z}_k,\boldsymbol{u}_k), \tag{2b}$$

$$\mathcal{B}(\boldsymbol{z}_k) \cap \left(\mathcal{O}^{\text{static}} \cup \mathcal{O}_k^{\text{dyn}}\right) = \emptyset, \tag{2c}$$

$$\boldsymbol{u}_k \in \mathcal{U},\ \boldsymbol{z}_k \in \mathcal{X}\,, \tag{2d}$$

where $\mathcal{X}$ and $\mathcal{U}$ are the set of admissible states and inputs, respectively. $\boldsymbol{z}_{1:N}$, $\boldsymbol{u}_{0:N-1}$ are the predicted state and control, respectively over a prediction horizon T_{horizon} divided into N prediction steps. θ_k denotes the predicted progress along the reference path. J is a cost function defining the planner objectives. By solving the optimization problem above, we can obtain the optimal sequence of commands $\boldsymbol{u}^*_{0:N-1}$ to guide the AGV along the reference path.

In the remainder of the paper, we show how to locally parametrize the reference path, formulate the objectives and how to define the collision avoidance constraints that together form our LMPCC framework.

III. METHOD

The goal of the LMPCC is to generate feasible and optimal motion with respect to the defined cost along the constructed local reference path. The LMPCC framework contains the following contributions:

1) A local parametrization of a global reference path through the environment (Section III-A).
2) A strategy to select the length of the local reference path representation (Section III-B).
3) The application of a search routine to solve the approximation of the path progress estimation (Section III-C).
4) The use of a transition function between waypoints to eliminate the non-differentiable representation of the reference path (Section III-D).
5) Static obstacle avoidance by explicitly constraining the control problem to an approximation of the collision-free area as a feasible set. (Sections III-E.1 and III-E.2).
6) A correcting bound on the Minkowski sum of a circle and an ellipse for moving obstacles (Section III-E.3).

In Section III-F, we present how the general concept of MPCC applies to our locally formulated control problem and we conceptualize the complete planning framework.

A. Global planning / Generation of Reference Path

Given a goal position $\boldsymbol{p}_{\text{goal}}$, first, we use a global planning method (such as RRT [40]) to compute a collision-free path through the static environment representation from an initial position $\boldsymbol{p}_0$. This global reference path $\mathcal{P}$ consists of M points defined as $\boldsymbol{p}_m = [x_m^p, y_m^p] \in \mathcal{W}$ with $m \in \mathcal{M} := \{1,\dots,M\}$. Then, we split the path into segments delimited by $[x_m^p, y_m^p]$ and $[x_{m+1}^p, y_{m+1}^p]$. We use cubic spline interpolation to obtain an analytical expression of the reference path for each segment, connecting each of the global reference path points with a polynomial of length s_m. The efficient implementation of [41] allows splines to be generated in $\mathcal{O}(q)$ and evaluation of the spline at a single point to be performed in $\mathcal{O}(\log(q))$, where q is the number of input data points. This efficiency strengthens our choice of local spline fitting during execution. Hence, we define the piece-wise spline segments as a function of the traveled distance along the reference path θ. This path parameter equals zero at the start of each path segment and continuously increases along the segment. Fig. 2 shows three reference path segments and the projected robot position on the reference path. Appendix B-1-2 elaborates on the method used to fit the local reference path.

Each path segment $\boldsymbol{R}_m$ is composed of the analytical spline expression as a function of θ, $\boldsymbol{p}_m^r(\theta)$, concatenated with a corresponding velocity reference, that is, $\boldsymbol{R}_m := [{\boldsymbol{p}_m^r}^{\mathrm{T}}, v_{\text{ref},m}]^{\mathrm{T}}$. This velocity reference depends on the environment and should be provided by the route planning module. The cubic polynomials that define the reference path segments are given by

$$\boldsymbol{p}_m^r(\theta) = \begin{bmatrix} x_m^r \\ y_m^r \end{bmatrix} = \begin{bmatrix} a_{m,1}^x\theta^3 + a_{m,2}^x\theta^2 + a_{m,3}^x\theta + a_{m,4}^x \\ a_{m,1}^y\theta^3 + a_{m,2}^y\theta^2 + a_{m,3}^y\theta + a_{m,4}^y \end{bmatrix}, \tag{3}$$

where $[a_{m,1}^x,\dots,a_{m,4}^x]$ and $[a_{m,1}^y,\dots,a_{m,4}^y]$ are the coefficients of the cubic polynomials at segment m of both splines. We consider a limited set of path segments $\boldsymbol{R}_m \subset \mathcal{R}$

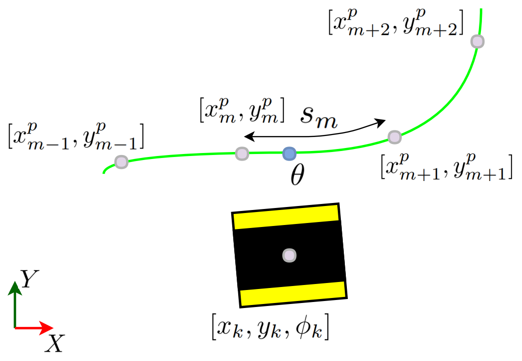


Fig. 2: Reference path representation

of the reference path to form our local representation. This allows reducing the computational complexity of the planning problem and segments outside of the local reference path to change over time. We define the local reference path $\boldsymbol{L}^r$ at the current time by linking η path segments, starting from the m-th closest path segment, that is,

$$\boldsymbol{L}^r = \{R_i \in \mathcal{R} | i = [m, \dots, m+\eta]\}. \tag{4}$$

B. Selecting the Number of Path Segments

We provide a strategy to select η to guarantee that the local reference path representation captures enough information of the global path to be tracked by the LMPCC. In particular, we provide a strategy to select the length of the local reference path with respect to the prediction horizon. The number of path segments η to be included in the local reference path is a function of the prediction horizon length, the individual path segment lengths, and the speed of the AGV at each time instance, as described below:

$$\underbrace{\sum_{i=m}^{m+\eta} s_i}_{\text{Local reference path length}} \geq \underbrace{\tau \sum_{j=0}^{N} v_j}_{\text{Traveled distance in horizon}}, \tag{5}$$

where τ is the length of the discretization steps along the horizon. We use an upper bound on the condition in Eq. (5) by considering maximum longitudinal velocity $v_{\max}$ and select η such that

$$\sum_{i=m}^{m+\eta} s_i \geq T_{\text{horizon}} \cdot v_{\max}. \tag{6}$$

C. Progress on Reference Path

MPCC keeps track of the path progress along the total reference path using the path parameter θ. Finding the corresponding path parameter, however, involves solving another optimization problem which would increase the computational cost of the algorithm [25]. If the distance between waypoints is small in relation to their curvature, spline parametrization can be regarded as reasonable approximations of arc-length parametrizations. Therefore, conventional MPCC assumes that the evolution of path parameter can be approximated by the traveled distance of the robot:

$$\theta_{k+1} \approx \theta_k + v_k \tau, \tag{7}$$

where v is the forward velocity of the controlled vehicle. This estimation of the path parameter, however, has shown to be quite coarse, especially when the AGV must deviate from the reference path during an avoidance maneuver. During such a maneuver, the path parameter starts drifting. LMPCC resolves the problem of the drifting path parameter by performing line search around the estimated path parameter, as Algorithm 1 describes, resulting in the refined path variable estimation $\tilde{\theta}$. In particular, note that the evolution of the path variable in the prediction horizon still matches Eq. (7), while the initialization of the path parameter is done with the refined approximation $\tilde{\theta}_0$ at each iteration.

Algorithm 1 Refined path variable estimation

```
1: $\theta_0 = \theta_{previous} + v_k \tau$
2: window = $[\theta_0 - L_{\text{window}} : \theta_0 + L_{\text{window}}]$
3: for each $\theta_{\text{sample}}$ in window do
4:     Compute distance to $\theta_{\text{sample}}$
5:     if distance < distance$_{\text{min}}$ then
6:         distance$_{\text{min}}$ = distance
7:         $\tilde{\theta}_0 = \theta_{\text{sample}}$
8:     end if
9: end for
```

D. Maintaining Continuity Over the Local Reference Path

To concatenate the parametrized reference path segments and reference velocities of separate segments into the local reference path that is tracked by the LMPCC, we provide a differentiable expression of the corresponding parameters over the prediction horizon. We use a sigmoid activation function σ to link η path segments $\boldsymbol{R}_m$ as in Eq. (4). In order to connect the analytical expressions of the reference path segments (Eq. (3)), we multiply the piece-wise spline sections by their corresponding activation function $\sigma_m(\theta, s_m)$, as follows:

$$\bar{\boldsymbol{L}}^r(\theta) = \sum_{i=m}^{m+\eta} [\boldsymbol{p}^r(\theta)^{\mathrm{T}}, \bar{v}_{ref}]^{\mathrm{T}} \cdot \sigma_m(\theta, s_m), \tag{8}$$

resulting in $\bar{\boldsymbol{L}}^r(\theta)$, a differentiable replacement for $\boldsymbol{L}^r$.

E. Collision Avoidance

Our collision avoidance strategy is separated for the static environment and the dynamic obstacles. The static environment is taken into account by defining a set of feasible positions of the AGV within the occupancy grid map representation as inequality constraints. Dynamic obstacle avoidance is achieved by introducing inequality constraints on the position of the AGV with respect to obstacle positions. Additional clearance from dynamic obstacles is obtained through a repulsive penalty in the cost function. Figure 3 provides an overview of our collision avoidance strategy that we discuss in more details below.

The occupied area by the AGV is represented by n_{disc} discs centered in $\boldsymbol{p}_j^B$, where $\boldsymbol{p}_j^B$ is the position on the j-th disc in the AGV body-fixed frame and n_{disc} is the number of discs used to bound the AGV, $j \in \mathcal{J}^{\text{disc}} := \{1, 2, \dots, n_{\text{disc}}\}$. The position of disc j in the world fixed frame, $\boldsymbol{p}_j^W$, is obtained through the transformation $\boldsymbol{p}_j^W = \boldsymbol{T}_B^W(\boldsymbol{z})\boldsymbol{p}_j^B$, where the transformation $\boldsymbol{T}_B^W(\boldsymbol{z})$ consists of a rotation by heading ψ and a translation by position $\boldsymbol{p}$ of the AGV. An inflated static obstacle environment is obtained by inflating the obstacles with r_{disc}.

First, static environment collision avoidance will be discussed, where Sections III-E.1 and III-E.2 present two different variations of our approach. Next, the position constraints on the dynamic obstacles are introduced in Section III-E.3. Finally, the repulsion from dynamic obstacles is described in Section III-E.4.

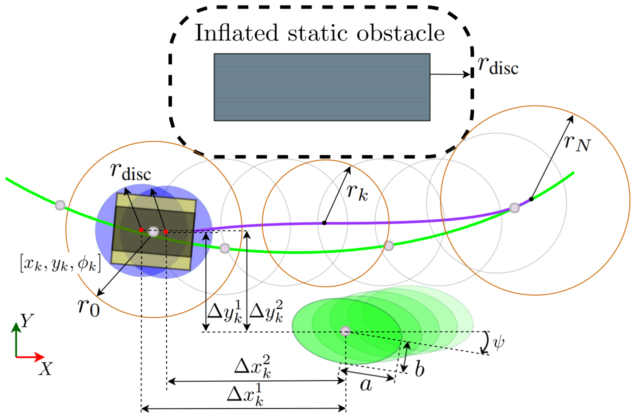


Fig. 3: Overview of collision avoidance parameters.

1) Static Obstacle Avoidance Using Circular Feasible Regions: It is assumed that the reference path is free of static obstacles. However, it should be guaranteed that static obstacles are avoided. Fig. 3 shows how we approximate a circular region free of collisions with static obstacles for each point in the prediction horizon. We overcome static environment collisions by defining a feasible set of solutions, where the approximated collision-free area is introduced as an inequality constraint on the current AGV position such that $\mathcal{B}(\boldsymbol{z}_k) \cup \mathcal{O}^{\text{static}} = \emptyset \, \forall \, k \in \{0, \ldots, N\}$. In particular, we use the optimal state sequence computed at the previous time instance $t-1$, namely, $\boldsymbol{z}^*_{0:N|t-1}$. This sequence of states contains the positions $\boldsymbol{p}^*_{0:N|t-1}$ which are used to center the circles depicted in Figure 3. The radii of the collision-free region are found by expanding squares in the inflated occupancy grid map of the environment until an occupied cell is detected, starting from the AGV positions $\boldsymbol{p}^*_{0:N|t-1}$. The radius of the collision-free circle at timestep k is denoted by r_k. Fig. 4 describes the method used to find r_k of the circle. A square is expanded in the occupancy grid map, with step size Δ^{search}, to search for the nearest occupied cell. The grid map is indexed by $\bar{\boldsymbol{p}} = [\bar{x}, \bar{y}]$, having its origin at $\boldsymbol{p}^{\text{map}}_{\mathcal{O}}$ in $\mathcal{W}$, and has a cell size of $\gamma \times \gamma$. A more detailed description of this procedure and its implementation is presented in Appendix B-1-3.

Given $\boldsymbol{z}^*_{0:N|t-1}$, we enforce that the positions of the discs $\boldsymbol{p}^W_j$, may never enter the inflated static obstacle environment by constraining their translation to r_k. This constraint is given by Eq. (9) for disc j, where the positions $\boldsymbol{p}^W_{j,k}$ are obtained through their corresponding transformations $\boldsymbol{T}^W_{B,k}(\boldsymbol{z}_k)$.

$$c_k^{\text{stat},j}(\boldsymbol{z}_k) = r_k - \|\boldsymbol{p}^*_{k|t-1} - \boldsymbol{T}^W_{B,k}(\boldsymbol{z}_k)\boldsymbol{p}^B_j\|_2^2\Big|_k > 0 \qquad \forall k \in \{0, \ldots, N\} \tag{9}$$

Using Eq. (9) to avoid static obstacles will result in n_{disc} quadratic inequality constraints in the LMPCC problem formulation.

2) Static Obstacle Avoidance Using Polygon Feasible Regions: In our second approach of representing the area free of static environment collisions, we use a polygon representation. More specifically, a convex four-sided polygon, or convex quadrilateral, will represent the feasible region of solutions such that the AGV will not collide with the static environment. This representation has as advantage that it can cover a much larger approximation of the actual collision-free area compared to the circle. Especially close to static obstacles, the polygon region provides a safe way out, whereas the circular area will remain bound by the closest obstacle as shown in Fig. 5. Furthermore, the constraints on static obstacle avoidance can now be written in linear form, in contrast with the quadratic form of Eq. (9). On the other hand, searching for the convex quadrilateral collision-free area in the occupancy grid map is more difficult. The feasible area is again defined as the set of feasible positions of the AGV, such that the individual positions of the occupied area representing discs do not collide with the inflated static obstacle environment. r

The size and position of the quadrilateral is expressed with the coordinates $\boldsymbol{p}^q_u = [x^q_u, y^q_u]$, with $u := \{1, \ldots, 4\}$, as shown in Fig. 6. The unit vector along each side of the quadrilateral is given by $\vec{t^u}$ and the normal inward unit

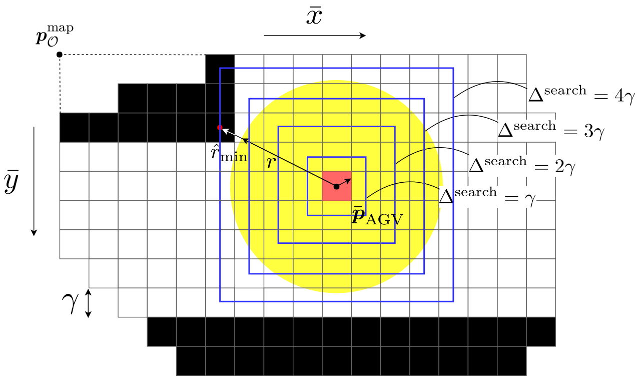

$\bar{x}$
$\boldsymbol{p}^{\text{map}}_{\mathcal{O}}$
$\Delta^{\text{search}} = 4\gamma$
$\Delta^{\text{search}} = 3\gamma$
$\Delta^{\text{search}} = 2\gamma$
$\Delta^{\text{search}} = \gamma$
$\bar{y}$
$\hat{r}_{\text{min}}$
r
$\bar{\boldsymbol{p}}_{\text{AGV}}$
γ

Fig. 4: Search for the nearest occupied cell by expanding a square in the occupancy grid map.

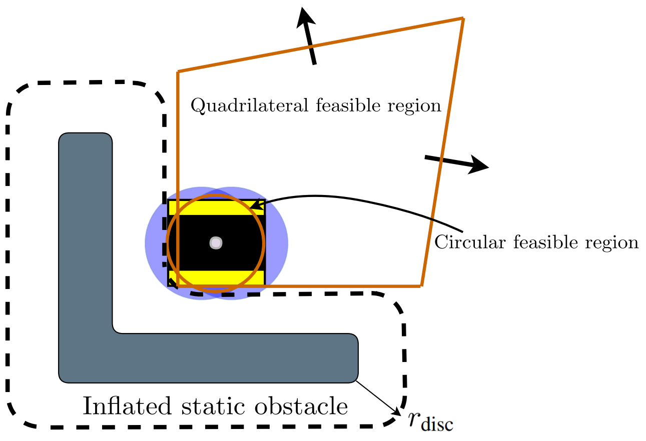


Fig. 5: Comparing the collision-free area representations.

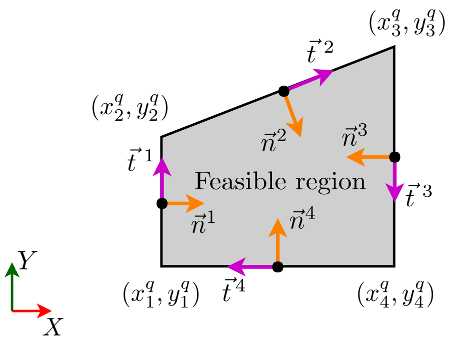


Fig. 6: The collision-free area as a convex quadrilateral feasible region.

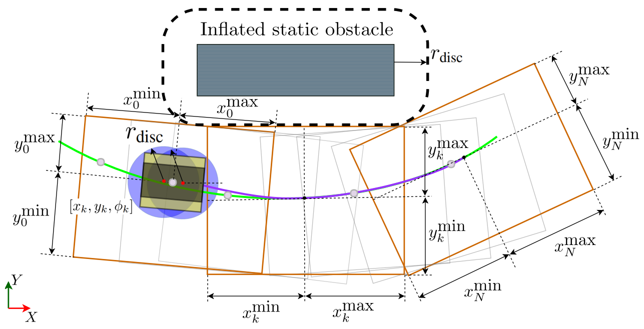


Fig. 8: Expressing the feasible region of solutions using rectangles.

vector by $\vec{n}^u$. The feasible region that is represented by this particular shape is expressed as a union of four linear constraints, $c_k^{\text{stat,j}}(\boldsymbol{z}_k) = \bigcup_{u=1}^{4} c_k^{\text{stat},u,j}(\boldsymbol{z}_k)$, for each disc j representing the occupied area of the AGV. Each individual linear constraint expresses that position $\boldsymbol{p}_j^W$ should remain on the inward side of the quadrilateral that is found for each optimal position in the prediction horizon, obtained at the previous time instance $t-1$. This means that the set of feasible solution positions with respect to $\boldsymbol{p}^*_{k|t-1}$ is given by $\{\boldsymbol{p}_k | \vec{n}_k^u \cdot (\boldsymbol{p}^*_{k|t-1} - \boldsymbol{T}^W_{B,k}(\boldsymbol{z}_k)\boldsymbol{p}_j^B) < a_k^u\}$, where a_k^u is a constant. Given $\boldsymbol{p}_u^q$, we can compute the corresponding $\vec{n}^u$ through $\vec{t}^u$, after which a_k^u is found by solving $\vec{n}_k^u \cdot \boldsymbol{p}_u^q = a_k^u$. The resulting constraint for disc j is given by Eq. (10), where the positions $\boldsymbol{p}^W_{j,k}$ are obtained through their corresponding transformations $\boldsymbol{T}^W_{B,k}(\boldsymbol{z}_k)$.

$$c_k^{\text{stat},u,j}(\boldsymbol{z}_k) = a_k^u - \vec{n}_k^u \cdot \left(\boldsymbol{p}^*_{k|t-1} - \boldsymbol{T}^W_{B,k}(\boldsymbol{z}_k)\boldsymbol{p}_j^B\right)\Big|_{k,u} > 0$$
$$\forall k \in \{0,\dots,N\}, \quad \forall u \in \{1,\dots,4\} \tag{10}$$

In our implemented search routine, we limit the set of quadrilateral polygons to rectangles. Our search routine expands the sides of a vehicle aligned rectangle simultaneously in the inflated occupancy grid environment with steps of Δ^{search}, until either an occupied cell is found or the maximum search distance $\Delta^{\text{search}}_{\max}$ is reached. Once an expanding rectangle side is fixed as a result of an occupied cell, the rest of the rectangle sides are still expanded to search for the largest possible area. The result of the search routine are the rectangle dimensions $[x_k^{\min}, x_k^{\max}, y_k^{\min}, y_k^{\max}]$ with $k := \{0,\dots,N\}$, defined in the AGV coordinate frame, as visualized by Fig. 7. A more detailed description of this search routine and its implementation is presented in Appendix B-1-3. The parametrized dimensions of Fig. 7 express the quadrilateral coordinates $\boldsymbol{p}^q_{u,k}$, originated at the optimal positions obtained from the solution at the previous time instance, shown by Fig. 8.

Using the inequality constraint of Eq. (10), the quadrilateral constraint to avoid static obstacles will appear as $4 \cdot n_{\text{disc}}$ linear inequality constraints in the LMPCC problem formulation.

3) Position Constraints on Dynamic Obstacles: Recall that each moving obstacle A_i is represented by state $\boldsymbol{z}^{A_i}(t)$, and ellipse dimensions a and b. The collision avoidance constraints can be defined as an inequality constraint for each j-th disc bounding the robot with respect to the distance of each obstacle $i \in \{1,\dots,n\}$ at time k as depicted in Fig. 3.

Omitting i for simplicity, the inequality constraint on each disc of the AGV with respect to the obstacles is given by

$$c_k^{\text{obst},j}(\boldsymbol{z}_k) = \begin{bmatrix}\Delta x_k^j \\ \Delta y_k^j\end{bmatrix}^{\text{T}} R(\psi)^{\text{T}} \begin{bmatrix}\frac{1}{\alpha^2} & 0 \\ 0 & \frac{1}{\beta^2}\end{bmatrix} R(\psi) \begin{bmatrix}\Delta x_k^j \\ \Delta y_k^j\end{bmatrix}\Bigg|_{k,j} > 1. \tag{11}$$

The distance from disc j to the obstacle is split into its Δx^j and Δy^j components as shown in Fig. 3. $R(\psi_k)$ is the rotation matrix corresponding to the heading of the obstacle and α and β are the resulting axes of the ellipse constraint.

It is important to notice that previous approaches assumed that the Minkowski sum of an ellipse with a circle is an ellipsoid with semi-major axis $\alpha = a + r_{\text{disc}}$ and semi-minor axis $\beta = b + r_{\text{disc}}$, represented by the grey area in Fig. 9 [25]. This assumption, however, is not correct and collision can still occur [42]. In order to ensure collision-free motions the radius is enlarged by a factor λ:

$$\begin{bmatrix}\alpha \\ \beta\end{bmatrix} = \begin{bmatrix}a + \lambda r_{\text{disc}} \\ b + \lambda r_{\text{disc}}\end{bmatrix}, \tag{12}$$

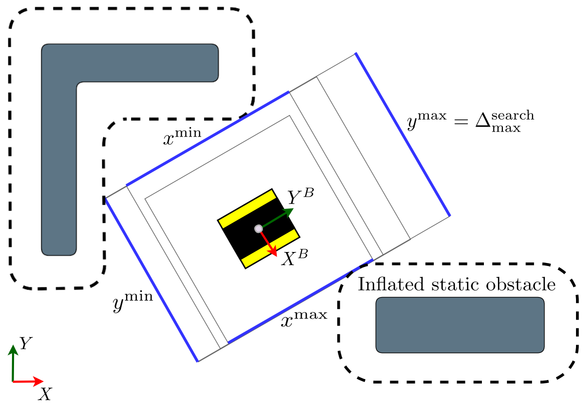


Fig. 7: Vehicle aligned search routine.

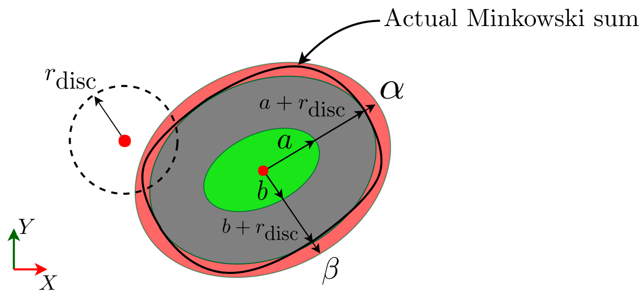


Fig. 9: True Minkowski sum and the bounding ellipse.

where λ is computed such that the curvature of the constraint ellipsoid is a lower bound of the curvature of the Minkowski sum, visualized by Fig. 9, as follows:

$$\lambda = \min \left[\begin{array}{c} \frac{(a^2+br)\left(1-\frac{2ab}{a^2+br}+\sqrt{\frac{(a-2b)^2+br)}{a^2+br}}\right)}{2b} \\ \frac{(b^2+ar)\left(1-\frac{2ab}{b^2+ar}+\sqrt{\frac{(-2a+b)^2+ar)}{b^2+ar}}\right)}{2a} \end{array} \right]. \quad (13)$$

This guarantees that the constraint ellipsoid entirely bounds the collision space, represented by the red area in Fig. 9. The derivation of λ is given in Appendix E.

4) Repulsion from Dynamic Obstacles: A penalty in the cost function that resembles repulsion from dynamic obstacles ensures clearance between the AGV and the dynamic obstacles [28]. The penalty is defined as follows:

$$J_{\text{repulsive}}(\boldsymbol{z}_k) = Q_R \sum_{i=1}^{n} \left(\frac{1}{(\Delta x_k)^2 + (\Delta y_k)^2 + \kappa} \right), \quad (14)$$

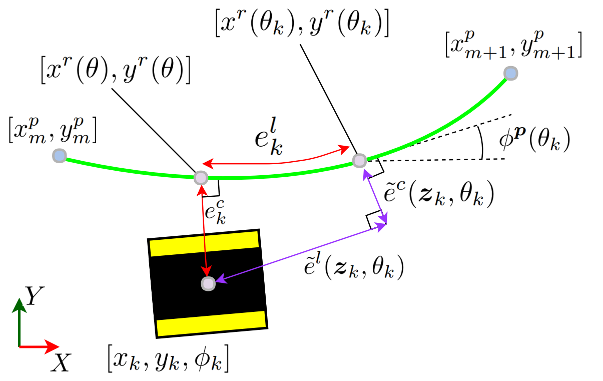
where Q_R is the weight on the repulsive forces . The distance from the AGV to the dynamic obstacles is given in its separate Δx_k and Δy_k components as in Fig. 3. A small value κ is introduced to ensure numerical stability of the solver [39].

F. Local Model Predictive Contouring Control

As introduced in Section III-A, the global reference path $\mathcal{P}$ consists of M segments, resulting in a local reference trajectory $\bar{\boldsymbol{L}}^r(\theta)$ with η segments, starting from the m-th closest path segment. The approximated local contour error $\tilde{e}^c$ and longitudinal error $\tilde{e}^l$ are expressed as a function of θ as visualized in Fig. 10. With $\bar{\boldsymbol{p}}^r(\theta_k)$ being the local reference path evaluated at θ_k, the tracking error vector $\boldsymbol{e}_k := [\tilde{e}^c(\boldsymbol{z}_k, \theta_k), \tilde{e}^l(\boldsymbol{z}_k, \theta_k)]^{\mathrm{T}}$ is defined as follows:

$$\boldsymbol{e}_k = \begin{bmatrix} \sin\phi(\theta_k) & -\cos\phi(\theta_k) \\ -\cos\phi(\theta_k) & -\sin\phi(\theta_k) \end{bmatrix} \left(\boldsymbol{p}_k - \bar{\boldsymbol{p}}^r(\theta_k) \right). \quad (15)$$

In order to ensure progress along the reference path, the conventional MPCC scheme proposes to add a negative cost term proportional to the traveled distance. In contrast, we introduce a cost term that penalizes the deviation of the vehicle velocity v_k with respect to a reference velocity v_{ref}. The reference velocity is provided according to Eq. (8), allowing the vehicle to adapt its speed according to the path



$[x^r(\theta_k), y^r(\theta_k)]$
$[x^p_{m+1}, y^p_{m+1}]$
$[x^r(\theta), y^r(\theta)]$
$[x^p_m, y^p_m]$
e^l_k
$\phi^p(\theta_k)$
e^c_k
$\tilde{e}^c(\boldsymbol{z}_k, \theta_k)$
$\tilde{e}^l(\boldsymbol{z}_k, \theta_k)$
Y
X
$[x_k, y_k, \phi_k]$

Fig. 10: Approximated contour and lag error on the path segment.

current segment that it is tracking. Now, the LMPCC tracking cost is defined as $J_{\text{tracking}}(\boldsymbol{z}_k, \theta_k) = \boldsymbol{e}_k^T \boldsymbol{Q}_\epsilon \boldsymbol{e}_k + Q_v(v_{ref} - v_k)^2$, where $\boldsymbol{Q}_\epsilon$ and Q_v are the weights on the tracking error and the reference velocity, respectively. Additionally, the cost function also penalizes with weight $\boldsymbol{Q}_u$ the inputs, that is, $J_{\text{input}}(\boldsymbol{z}_k, \theta_k) = \boldsymbol{u}_k^T \boldsymbol{Q}_u \boldsymbol{u}_k$. We can now formulate our LMPCC control problem:

$$\begin{aligned} J^* = \min_{\boldsymbol{z}_{0:N}, \boldsymbol{u}_{0:N-1}} & \sum_{k=0}^{N-1} J(\mathbf{z}_k, \mathbf{u}_k, \theta_k) + J(\boldsymbol{z}_N, \theta_N) \\ \text{s.t.} \quad & \boldsymbol{z}_{k+1} := f(\boldsymbol{z}_k, \boldsymbol{u}_k), \\ & \theta_{k+1} = \theta_k + v_k \tau, \\ & \boldsymbol{z}_{\min} \le \boldsymbol{z}_k \le \boldsymbol{z}_{\max}, \\ & \boldsymbol{u}_{\min} \le \boldsymbol{u}_k \le \boldsymbol{u}_{\max}, \\ & c_k^{\text{stat},j}(\boldsymbol{z}_k) > 0, \\ & c_k^{\text{obst},j}(\boldsymbol{z}_k) > 1, \quad \forall j, \forall \text{obst}, \\ & \mathbf{z}_0 = z_{\text{init}},\ \theta_0 = \tilde{\theta}, \end{aligned} \quad (16)$$

where $J(\mathbf{z}_k, \mathbf{u}_k, \theta_k) := J_{\text{tracking}}(\boldsymbol{z}_k, \theta_k) + J_{\text{repulsive}}(\boldsymbol{z}_k) + J_{\text{input}}(\boldsymbol{u}_k)$. Algorithm 2 summarizes our design. Each control iteration, feedback steps are performed until either a Karush-Kuhn-Tucker (KKT) condition [43] or the maximum number of iterations is satisfied (line 10).

IV. Results

This section presents the results obtained using the proposed planner. In the following, we provide simulation results and real-world experiments the mobile robot shown in Fig. 11 [44]. In our experiments, we show that the LMPCC framework can be wrapped in an autonomous navigation system, completely running on the mobile robot using on-board localization, perception, and processing. These are the first experiments where a MPCC approach is implemented completely independent of any external components.

In order to solve the non-linear optimal control problem (Eq. (16)), highly efficient C-code is exported using ACADO [39] to provide fast solutions online. Discretization of the time-continuous differential equations is done via direct

Algorithm 2 Local Model Predictive Control

```
 1: Given $z_0$, $z_{\text{goal}}$, $\mathcal{O}^{\text{static}}$, $\mathcal{O}_k^{\text{dyn}}$, and $N$
 2: **Initialization**: $k = 0$
 3: Build global path $\mathcal{P}$ Eqs. (3) and (8)
 4: Select $\eta$ according to Eq. (6)
 5: Build $\bar{\boldsymbol{L}}^r(\theta_k)$ according to Eqs. (3) and (8)
 6: **while** $\boldsymbol{z}_k \neq \boldsymbol{z}_{\text{goal}}$ in parallel **do**
 7:     Process sensor data
 8:     Estimate $\tilde{\theta}_0$ according to Algorithm 1
 9:     Compute $r_k$ along $\boldsymbol{z}^*_{0:N|t-1}$
10:     **while** $\texttt{iter} < \texttt{iter}_{\max} \wedge \text{KKT} > \text{threshold}$ **do**
11:         Solve Eq. (16)
12:         $\texttt{iter} = \texttt{iter} + 1$
13:     **end while**
14:     Apply $\boldsymbol{u}_0^*$
15:     **if** $\theta_k > s_k$ **then**
16:         $k = k + 1$
17:         Re-plan global path $\mathcal{P} \setminus \boldsymbol{p}^r$
18:         Build $\bar{\boldsymbol{L}}^r(\theta_k)$ according to Eqs. (3) and (8)
19:     **end if**
20: **end while**
```

multiple shooting techniques. The resulting QP problem is condensed and solved using `qpOASES` [37]. We implemented our design in `C++` and we will release the code as an open source Robotic Operating System (ROS) package. More details on the implementation can be found in Appendix B.

Two types of experiments are performed using the mobile robot. The first experiment is performed within a closed environment, where tracking performance and planning behavior is inspected in a specific static environment set-up and interacting pedestrians. The second experiment type brings the LMPCC to real-world scenarios where the planner is integrated within a complete autonomous navigation architecture. A comparison will be made with simulation results to highlight shortcomings of the total system, where Gazebo is used as simulation framework [49]. Whereas in the first experiment a circular approximation of the static obstacle collision-free environment is used to define the feasible solution set (Section III-E.1), we aim to traverse the unstructured environments of the second experiment using the rectangular approximation (Section III-E.2). Both approximations will be compared briefly in Section IV-D and more experimental results are presented in Appendix A.

Fig. 11: Mobile robot used for experiments [44].

A. Experimental Setup

The mobile robot allows inputs to be given in the form of longitudinal and angular velocities. A non-linear unicycle kinematic motion model, as given by Eq. (17), was used to model the robot dynamics within the LMPCC formulation [45].

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} v\cos\psi \\ v\sin\psi \\ \omega \end{bmatrix} \tag{17}$$

The computed input velocity commands v and ω are directly passed to a lower level wheel controller that translates the longitudinal and angular velocity commands into wheel velocities of the differentially driven platform. Fig. 12 visualizes the layout of the interacting control scheme. The lower level wheel controller (called diff-drive controller) reacts on $[\bar{\omega}_L, \bar{\omega}_R]$, representing the measured wheel velocities by the wheel encoder, to accurately control the desired wheel velocities. The diff-drive controller updates at a frequency of 50 Hz. Appendix C-1 presents the mobile robot, including a derivation of the motion model and details on the aforementioned diff-drive controller.

The presented planner has been adopted within an autonomous navigation system for the mobile robot as a proof of concept. This means that the mobile robot is extended with perception and localization capabilities. Fig. 13 visualizes how the different functional components of the platform interact to safely navigate through a dynamic real-world environment. Note that the decomposition of functional subsystems satisfies a modular approach to fulfill the requirements of autonomous navigation capabilities [46]. Onboard localization is achieved by fusing measurements from the Inertial Measurement Unit (IMU), wheel encoders and Light Detection And Ranging (LiDAR) sensor. A Monte Carlo localization approach, as described by [47], is used to match LiDAR scans to the environmental map. The obtained positions are fused with the wheel odometry and IMU data through an Extended Kalman Filter (EKF), to aim for a more accurate measured position of the robot.

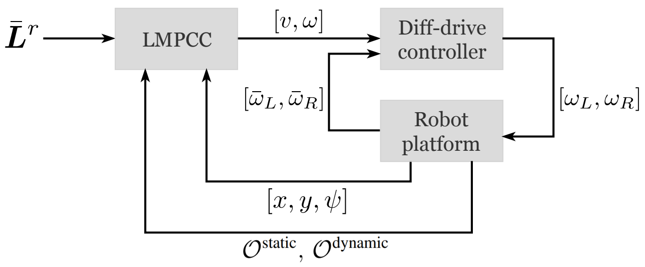


Fig. 12: Control scheme of the interacting LMPCC and differential drive controller.

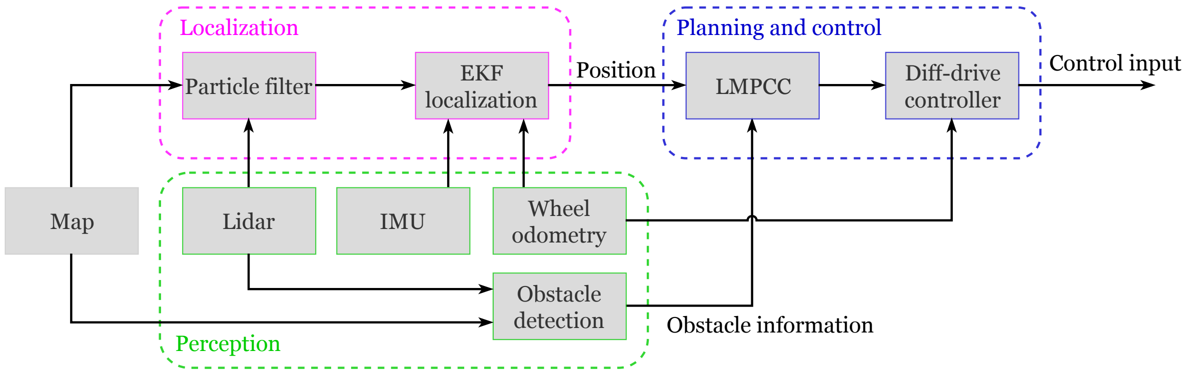


Fig. 13: Architecture of the implemented autonomous navigation system.

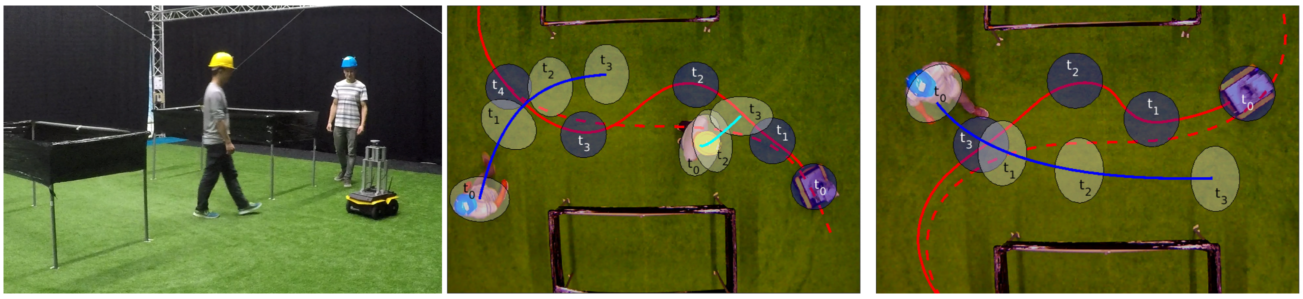
Additional to the predefined static map of the environment, updates of the static environment are incorporated within the computations of the collision-free area. In order to get a position measurement of the moving obstacles, clustered LiDAR point clouds, following the method of [48], are consulted to fit obstacle representing ellipses on the plane $\mathcal{W}$.

In addition, we implemented the global planner using the RRT-connect implementation from the Open Motion Planning Library (OMPL) [50]. A smoothened global path was constructed by building a clothoid, connecting each waypoint obtained from the RRT [51]. The waypoints of this global path were used to generate the local reference paths consisting of 3rd order polynomials as in Eq. (8). Furthermore, we used the OptiTrack system as ground-truth for the executed robot trajectories [52].

B. Closed Environment Experiments

For these experiments, the LMPCC was set to run at 20 Hz considering a prediction horizon of 5 sec with $N = 25$ steps. First, we tested tracking performance without moving obstacles at a reference velocity of $v_{\text{ref}} = 1$ m/s (Fig. 15). Second, we tested tracking and obstacle avoidance in the same environment using a reference velocity of $v_{\text{ref}} = 0.7$ m/s. The moving obstacles are two pedestrians walking around in the environment and their position is provided by the OptiTrack system. An occupancy grid map of the static environment was built a priori for the robot to localize within.

As shown by Fig. 15, the LMPCC achieved perfect tracking in simulation and on the robot by using the OptiTrack system to provide the robot state. When using onboard localization, from the perspective of the robot's believe state of its position, tracking performance is comparable to the one obtained with OptiTrack system. By looking at the ground-truth results obtained by the OptiTrack, however, the robot is not perfectly following the reference path (red line). This is mainly due to the uncertainty on the onboard localization data. It is most likely that this localization error is due to unmodeled skidding behavior in the odometry data of the wheels since the error mainly occurs at high speed turns where lateral skidding is clearly visible. We choose to analyze a time period of ten minutes of the experiment where pedestrians are present in the environment as shown in Fig. 14.a). The time period was chosen such, that most of the observed characteristic behavior of the planner was captured. In Fig. 16, we cumulate the clearance distance between the robot and the obstacles for the observed time period. In almost all instances, a minimum safe distance of 0.32 m was achieved, which corresponds to the robot radius, using a single disc. One particular avoidance maneuver resulted in an overlap between the collision radius of the robot and the static obstacle, which is most likely due to a localization error. The few detected collisions with dynamic obstacles were situations where the pedestrians violated the collision boundary of the robot and the robot could not avoid fast enough. This limitation is mainly caused by the non-holonomic nature of the robot and the acceleration limits



a) Interacting pedestrian. *b)* Interaction scenario 1. *c)* Interaction scenario 2.

Fig. 14: Two characteristic collision avoidance scenarios.

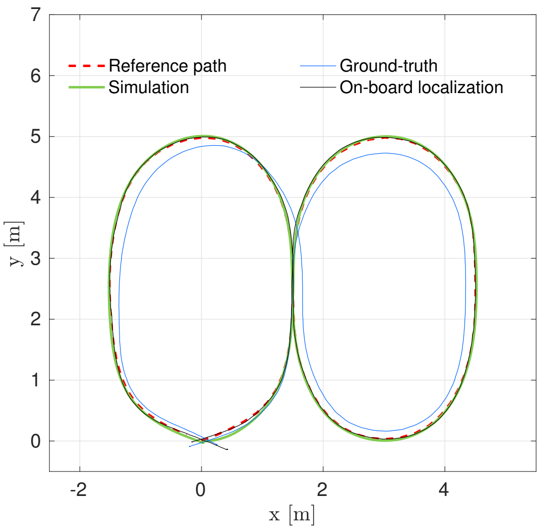


Fig. 15: Traveled path of the AGV compared to the reference.

on the actuating motors. Fig. 14.b)-c) shows two snapshots of two critical driving situations avoiding moving and static obstacles. The first scenario emphasizes the ability of the robot to safely navigate through an environment while moving persons are close by and are crossing the reference path. The second scenario shows that the robot is able to maintain clearance from static obstacles in its effort to avoid the pedestrian. More results on the robustness of static obstacle avoidance are presented in Appendix A-2.

It is observed that the repulsive penalty in the cost function effectively maintains clearance from the pedestrians. In some cases, however, the repulsive penalty causes the robot to dodge in a very conservative manner, while a less aggressive avoidance maneuver would satisfy. We argue that a better designed repulsive penalty can create a more compliant and efficient approach in navigating through dynamic environments. An improved version could for instance have an

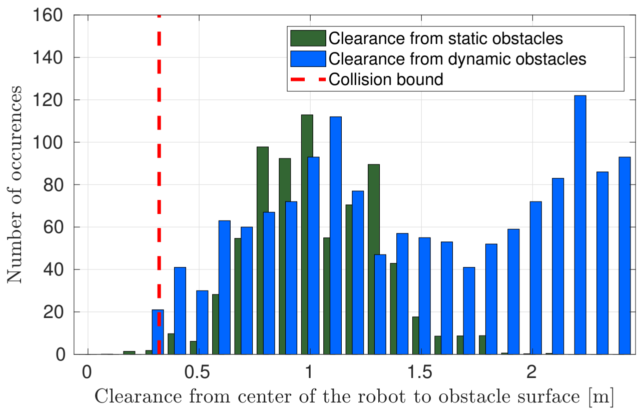


Fig. 16: Sampled clearance from the robot to static and dynamic obstacles.

TABLE I: Computation times

	Mean	Minimum	Maximum	Variance
Tracking experiment	3.2 ms	1.7 ms	15.3 ms	0.0017
Interaction experiment	11.0 ms	3.5 ms	265.2 ms	0.0151

adapting repulsion, considering the positions and velocities of the obstacle with respect to the robot.

Table I shows the computation times of solving the optimal control problem for the different experiments presented in this section. Real-time performance is satisfied for both cases. The case where the real-time constraint could not be satisfied during the experiments was caused by a pedestrian that violated the collision bound of the robot. The optimal control problem keeps iterating until the maximum number of iterations is reached, since no feasible solution can be found. The absence of a feasible solution results in a serious hazard since we are acting directly on the AGV actuating wheels. In our mobile robot experiments, it is a safe and valid solution to stand still when infeasible solutions occur. This problem does, however, require more research to come up with a proper method of dealing with infeasibility and satisfying the real-time constraint for any general AGV application.

C. Real-world Navigation Experiments

Our planner design has been adopted within the navigation framework of a mobile robot as presented in Section IV-A. All experiments in this section are performed on our mobile robot, completely independent of any external information or computation sources. For these experiments, the LMPCC was set to run at 20 Hz considering a prediction horizon of 2.5 sec with $N = 25$ steps. The reference velocity is set to $v_{\text{ref}} = 0.8$ m/s.

A first example of the working system is given in Fig. 17. The figure shows the robot in a scenario where a predefined global plan is used to traverse a corridor of our faculty back and forth. Different snapshots of separate time instances are shown in the single image, denoted by the time instances t_0, t_1, t_2, and t_3. On the right-hand side, an online available visualization of the experiment is shown. Within

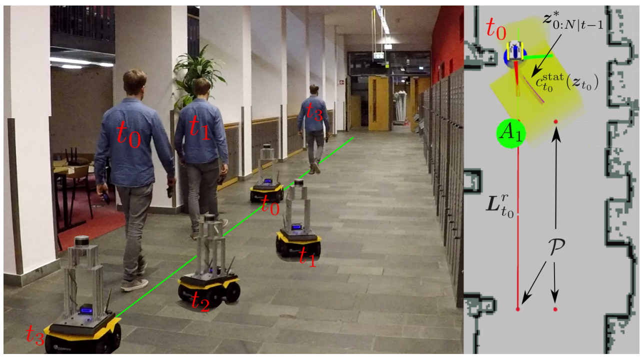
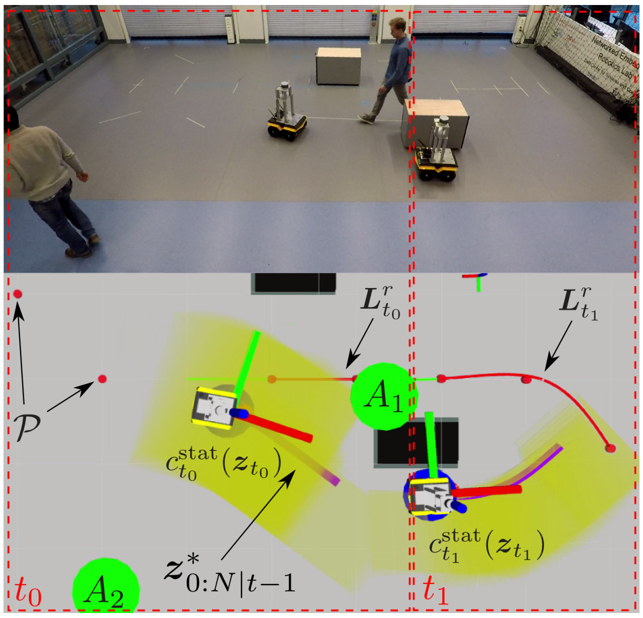


Fig. 17: Avoiding an oncoming pedestrian in a corridor. ***(left)*** real-world experiment ***(right)*** online visualization.



$\boldsymbol{L}^r_{t_0}$
$\boldsymbol{L}^r_{t_1}$
$\mathcal{P}$
A_1
A_2
$c^{\text{stat}}_{t_0}(\boldsymbol{z}_{t_0})$
$c^{\text{stat}}_{t_1}(\boldsymbol{z}_{t_1})$
$\boldsymbol{z}^*_{0:N|t-1}$
t_0
t_1

Fig. 18: Two time instances of an avoidance maneuver in an unstructured environment. ***(top)*** snapshots of the experiment ***(bottom)*** online visualization.

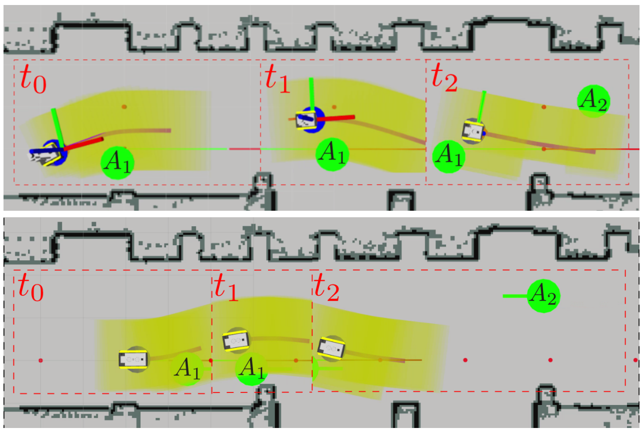


Fig. 19: Overtaking a pedestrian in a corridor ***(top)*** snapshots of the real-world experiment visualization ***(bottom)*** equivalent experiment in simulation with obstacle prediction.

the visualization, we can recognize the local reference path, $\boldsymbol{L}^r_{t_0}$, and the collision constraint with respect to the static environment, $c^{\text{stat}}_{t_0}(\boldsymbol{z}_{t_0})$, at time t_0. Furthermore, recall that the global path is denoted by $\mathcal{P}$ and the previously computed prediction horizon by $\boldsymbol{z}^*_{0:N|t-1}$. Symbols are added to the visualization for clarity. It is observed that the mobile robot avoids a collision with the oncoming pedestrian after which it returns to the reference path. It is, however, inspected that the avoidance maneuver is quite tight since no velocity information about the obstacle is available.

A second experiment emphasizes the applicability of our framework to navigation tasks in unstructured environments. Fig. 18 shows an avoidance maneuver of the mobile robot that puts static environment collision avoidance capabilities to the test. The interacting pedestrian is headed towards the robot on the reference path and forces the robot to deviate from the global plan. Caused by the constraints that express the area free of static collisions as the feasible set of solutions, LMPCC finds an alternative route around the drawer. By putting a relatively low weight on the contour error in the cost function (Eq. (15)), we allow the robot to come up with this alternative plan to traverse the environment.

In the last example, we discuss an overtaking experiment that is performed in the same corridor as in Fig. 17. A pedestrian is walking in the same direction as the robot but has a slower speed. The overtaking maneuver is invoked by the penalty devoted to the deviation from the reference velocity. Fig. 19 shows three time instances of the overtaking maneuver. Although the robot performs the maneuver without colliding with the pedestrian, the predicted return position to the reference path at t_1 does not match the true position at t_2. Although LMPCC is able to anticipate on the dynamic behavior of the pedestrian, the perception module of the mobile robot can not deliver such information in its current state. In addition to the implemented obstacle detection capabilities, an obstacle tracking module is required to make an estimate of the pedestrians' movement in the future. For example, an Extended Kalman Filter or a particle filter could be used for this purpose [53], [54]. Exemplifying this statement, Fig. 19 shows a simulated experiment, that mimics the performed real-world experiment. The only addition that distinguishes this simulation from the real-world experiment is the provided velocity estimation of the pedestrian that is overtaken. It is now observed that the actual point in space where the robot is able to return to the reference path matches the expectation in the prediction horizon of earlier time instances.

During the experiments, it is observed that uncertainties on the measured positions of the detected obstacles can cause the previously computed control horizon to become infeasible. Limited sensor data and uncertainty about obstacle behavior inevitably introduces prediction uncertainty in real-world navigation scenarios. We propose an extension to our framework where we incorporate (prediction) uncertainties of the moving obstacles to enhance safe operation. The proposed method follows [25] and will be explained together with preliminary results in Appendix D.

D. Comparing the Circular and Rectangular Static Environment Constraints

In terms of behavioral performance, we have experienced that both variations of the collision-free area constraint allowed the LMPCC to find safe solutions for traversal of unstructured static environments. The major difference in performance was observed close to static obstacles with dynamic obstacles nearby. In these particular cases, the circular constraint can be too conservative and trap the feasible solution set in a very limited space, resulting in an infeasible problem definition. We exemplify these observations by a comparing simulated navigation task in Appendix A-1.

Additionally, a computational comparison is made for the occupancy grid map search routines in a free space for a maximum search distance of 2 m. A typical search time for the circular collision-free area is 25 μs, whereas the rectangle expansion needs 360 μs. The difference of factor ten can be explained by the transformation that needs to be performed to the map fixed coordinate frame of the vehicle aligned rectangle sides.

V. Extension to an Autonomous Car

We tested our method in simulation for an autonomous vehicle, a Toyota Prius. Being able to use our framework to plan motion and the corresponding control inputs for the car, emphasizes the applicability of our approach to different types of vehicle models. The motion model used in the LMPCC to control the vehicle is the one presented in [55], that is, a kinematic bicycle model given by

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \\ \dot{v} \\ \beta \end{bmatrix} = \begin{bmatrix} v\cos(\psi+\beta) \\ v\sin(\psi+\beta) \\ \frac{v}{l_r}\sin(\beta) \\ a \\ \tan^{-1}\left(\frac{l_r}{l_r+l_f}\tan(\delta_f)\right) \end{bmatrix}, \tag{18}$$

where x and y are the coordinates of the center of mass in an inertial frame. ψ is the inertial heading and v is the speed of the vehicle. l_f and l_r represent the distance from the center of the mass of the vehicle to the front and rear axles, respectively. β is the angle of the current velocity of the center of mass with respect to the longitudinal axis of the car. a is the acceleration of the center of mass in the same direction as the velocity. The control inputs are the front and rear steering angles δ_f, and a. The LMPCC was set to run at 20Hz considering a prediction horizon of 5 sec with $N = 25$ steps.

The occupied area by the car is defined using three discs in accordance with Section III-E. The velocity reference was set to 8 m/s ($\approx$ 30 Km/h). We simulate the vehicle following a reference path, where the cost devoted to maintaining the reference velocity forces the car to overtake a cyclist. While overtaking the cyclist, the car has to avoid collision with a pedestrian that crosses the road. Fig. 20 shows that autonomous vehicle successfully avoids collision with the moving obstacles and stably converges to the reference trajectory in this particular scenario. An average computation time of 12.7 ms was required to solve the optimal control problem for the car. A solve time of 4.8 ms and 28.8 ms for the best and worst case computational scenario respectively indicates the real-time feasibility during the simulated experiment.

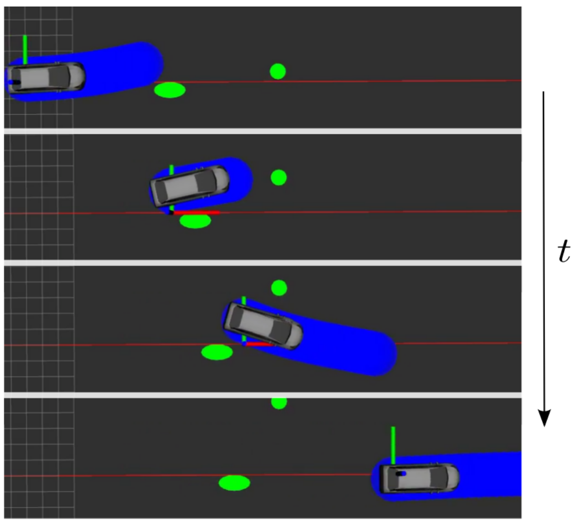


Fig. 20: Autonomous vehicle cyclist overtaking scenario.

VI. Conclusions

We proposed a trajectory optimization approach based on Local Model Predictive Contouring Control (LMPCC) to safely navigate AGVs in dynamic, unstructured environments. The LMPCC relies on a robust bound on the Minkowski sum to safely avoid dynamic obstacles. Furthermore, our design relies on a technique to compute a collision-free area to avoid collisions with the static environment. We showed the applicability of our LMPCC design in simulations for a mobile robot and an autonomous vehicle. Finally, we performed experiments using a mobile robot avoiding pedestrians in real-world environments. We showed that our motion planner satisfies the real-time constraint and can be integrated within a complete autonomous navigation system, relying on sensor data. Furthermore, the light implementation of our LMPCC allowed us to run all the algorithms onboard of the mobile robot. The next step is to test the proposed algorithm in a real autonomous car and to expand the current navigation system to incorporate prediction information about other road users. Improvements of the framework can be made by investigating a non-uniform repulsive penalty in the cost function, incorporating obstacle prediction uncertainty, and improve the method to deal with infeasible solutions of the optimal control problem.

References

[1] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, “A survey of motion planning and control techniques for self-driving urban vehicles,” *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, March 2016.

[2] W. Schwarting, J. Alonso-Mora, and D. Rus, “Planning and decision-making for autonomous vehicles,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 187–210, 2018. [Online]. Available: https://doi.org/10.1146/annurev-control-060117-105157

[3] S. M. La Valle, “Motion planning,” *IEEE Robotics & Automation Magazine*, vol. 18, no. 2, pp. 108–118, 2011.

[4] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun, “Anytime Dynamic A*: An Anytime, Replanning Algorithm,” 1995.

[5] O. Adiyatov and H. A. Varol, “A novel RRT*-based algorithm for motion planning in dynamic environments,” in *2017 IEEE International Conference on Mechatronics and Automation (ICMA)*, Aug 2017, pp. 1416–1421.

[6] L. Jaillet and T. Simeon, “A PRM-based motion planner for dynamically changing environments,” in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 2, Sept 2004, pp. 1606–1611 vol.2.

[7] M. Pivtoraiko and A. Kelly, “Fast and feasible deliberative motion planner for dynamic environments,” in *International Conference on Robotics and Automation*, Pittsburgh, PA, May 2009.

[8] Y. Wang, J. Wang, and S. Yin, “An Object-Based Path Planning Using Grids-Potential Fields for Intelligent Robot,” in *2009 Third International Conference on Genetic and Evolutionary Computing*, Oct 2009, pp. 150–153.

[9] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics Automation Magazine*, vol. 4, no. 1, pp. 23–33, Mar 1997.

[10] J. Ziegler and C. Stiller, "Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2009, pp. 1879–1884.

[11] W. Xu, J. Wei, J. M. Dolan, H. Zhao, and H. Zha, "A real-time motion planner with trajectory optimization for autonomous vehicles," in *2012 IEEE International Conference on Robotics and Automation*, May 2012, pp. 2061–2067.

[12] Y. K. Hwang and N. Ahuja, "Gross motion planning - a survey," *ACM Comput. Surv.*, vol. 24, no. 3, pp. 219–291, Sept. 1992. [Online]. Available: http://doi.acm.org/10.1145/136035.136037

[13] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 4569–4574.

[14] D. Fridovich-Keil, S. L. Herbert, J. F. Fisac, S. Deglurkar, and C. J. Tomlin, "Planning, Fast and Slow: A Framework for Adaptive Real-Time Safe Trajectory Planning," *arXiv preprint arXiv:1710.04731*, 2017.

[15] S. L. Herbert, M. Chen, S. Han, S. Bansal, J. F. Fisac, and C. J. Tomlin, "Fastrack: a modular framework for fast and guaranteed safe motion planning," in *56th IEEE Conference on Decision and Control (CDC)*. IEEE, 2017, pp. 1517–1522.

[16] M. Mukadam, X. Yan, and B. Boots, "Gaussian process motion planning," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 9–15.

[17] J. M. Maciejowski, *Predictive control: with constraints*. Pearson education, 2002.

[18] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[19] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, April 2016.

[20] A. Carvalho, S. Lefévre, G. Schildbach, J. Kong, and F. Borrelli, "Automated driving: The role of forecasts and uncertainty–a control perspective," *European Journal of Control*, vol. 24, pp. 14–32, 2015.

[21] N. Murgovski and J. Sjöberg, "Predictive cruise control with autonomous overtaking," in *54th IEEE Conference on Decision and Control (CDC),*. IEEE, 2015, pp. 644–649.

[22] M. Jalali, S. Khosravani, A. Khajepour, S.-k. Chen, and B. Litkouhi, "Model predictive control of vehicle stability using coordinated active steering and differential brakes," *Mechatronics*, vol. 48, pp. 30–41, 2017.

[23] C. M. Kang, S. Lee, and C. C. Chung, "On-road path generation and control for waypoints tracking," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 3, pp. 36–45, 2017.

[24] D. Lam, C. Manzie, and M. Good, "Model predictive contouring control," in *49th IEEE Conference on Decision and Control (CDC)*, Dec 2010, pp. 6137–6142.

[25] W. Schwarting, J. Alonso-Mora, L. Paull, S. Karaman, and D. Rus, "Safe nonlinear trajectory generation for parallel autonomy with a dynamic vehicle model," *IEEE Transactions on Intelligent Transportation Systems*, no. 99, pp. 1–15, 2017.

[26] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1: 43 scale rc cars," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 628–647, 2015.

[27] J.-M. Park, D.-W. Kim, Y.-S. Yoon, H. J. Kim, and K.-S. Yi, "Obstacle avoidance of autonomous vehicles based on model predictive control," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 223, no. 12, pp. 1499–1516, 2009. [Online]. Available: https://doi.org/10.1243/09544070JAUTO1149

[28] M. A. Abbas, R. Milman, and J. M. Eklund, "Obstacle avoidance in real time with nonlinear model predictive control of autonomous vehicles," *Canadian Journal of Electrical and Computer Engineering*, vol. 40, no. 1, pp. 12–22, winter 2017.

[29] J. Liu, P. Jayakumar, J. L. Stein, and T. Ersal, "A nonlinear model predictive control algorithm for obstacle avoidance in autonomous ground vehicles within unknown environments," 2015.

[30] D. Lenz, T. Kessler, and A. Knoll, "Stochastic model predictive controller with chance constraints for comfortable and safe driving behavior of autonomous vehicles," in *2015 IEEE Intelligent Vehicles Symposium (IV)*, June 2015, pp. 292–297.

[31] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3 – 20, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0005109801001741

[32] Y. Ding, Z. Xu, J. Zhao, K. Wang, and Z. Shao, "Embedded MPC Controller Based on Interior-Point Method with Convergence Depth Control," *Asian Journal of Control*, vol. 18, no. 6, pp. 2064–2077, 2016, rR-15-0136.R2. [Online]. Available: http://dx.doi.org/10.1002/asjc.1299

[33] A. Domahidi, A. U. Zgraggen, M. N. Zeilinger, M. Morari, and C. N. Jones, "Efficient interior point methods for multistage problems arising in receding horizon control," in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, Dec 2012, pp. 668–674.

[34] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, Mar. 2006. [Online]. Available: https://doi.org/10.1007/s10107-004-0559-y

[35] "FORCES Pro ," https://www.embotech.com/FORCES-Pro, accessed: 2018-02-02.

[36] D. Goldfarb and A. Idnani, "A numerically stable dual method for solving strictly convex quadratic programs," *Mathematical Programming*, vol. 27, no. 1, pp. 1–33, Sep 1983. [Online]. Available: https://doi.org/10.1007/BF02591962

[37] H. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.

[38] H. J. Ferreau, H. G. Bock, and M. Diehl, "An online active set strategy to overcome the limitations of explicit mpc," *International Journal of Robust and Nonlinear Control*, vol. 18, no. 8, pp. 816–830, 2008. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rnc.1251

[39] B. Houska, H. Ferreau, and M. Diehl, "ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization," *Optimal Control Applications and Methods*, vol. 32, no. 3, pp. 298–312, 2011.

[40] J. J. Kuffner and S. M. La Valle, "Rrt-connect: An efficient approach to single-query path planning," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2. IEEE, 2000, pp. 995–1001.

[41] T. Kluge, "c++ cubic spline library," 2014, accessed Jul. 5, 2018. [Online]. Available: http://kluge.in-chemnitz.de/opensource/spline/

[42] Y. Yan and G. S. Chirikjian, "Closed-form characterization of the minkowski sum and difference of two ellipsoids," *Geometriae Dedicata*, vol. 177, no. 1, pp. 103–128, 2015.

[43] H. W. Kuhn and A. W. Tucker, "Nonlinear programming," in *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*. Berkeley, Calif.: University of California Press, 1951, pp. 481–492. [Online]. Available: https://projecteuclid.org/euclid.bsmsp/1200500249

[44] Jackal small unmanned ground vehicle, Clearpath$^{\text{TM}}$ Robotics. [Online]. Available: https://www.clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/

[45] R. Siegwart and I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*, 2nd ed. The MIT Press, 2011.

[46] T. T. Mac, C. Copot, D. T. Tran, and R. D. Keyser, "Heuristic approaches in robot path planning: A survey," *Robotics and Autonomous Systems*, vol. 86, pp. 13 – 28, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889015300671

[47] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte carlo localization: Efficient position estimation for mobile robots," in *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI'99).*, July 1999.

[48] I. Bogoslavskyi and C. Stachniss, "Efficient online segmentation for sparse 3d laser scans," *PFG – Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, pp. 1–12, 2017.

[49] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, Sept 2004, pp. 2149–2154 vol.3.

[50] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, December 2012, http://ompl.kavrakilab.org.

[51] E. Bertolazzi and M. Frego, "G1 fitting with clothoids," *Mathematical Methods in the Applied Sciences*, vol. 38, no. 5, pp. 881–897, 2015.

[52] N. Point, “Optitrack,” *Natural Point, Inc*, 2011.

[53] J. H. Lee, T. Tsubouchi, K. Yamamoto, and S. Egawa, “People tracking using a robot in motion with laser range finder,” in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2006, pp. 2936–2942.

[54] H. . P. t. w. a. m. r. a. c. o. K. Bellotto, Nicola & Hu and particle filters. 388-393., “People tracking with a mobile robot: a comparison of Kalman and particle filters,” *Proceedings of the 13th IASTED International Conference on Robotics and Applications*, vol. 1, pp. 388–393, 2007.

[55] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, “Kinematic and dynamic vehicle models for autonomous driving control design.” in *Intelligent Vehicles Symposium*, 2015, pp. 1094–1099.

Chapter 3

Discussion and Conclusion

In this chapter, we conclude the thesis, starting with a discussion on the findings of the thesis work in Section 3-1. Subsequently, a conclusion is made in Section 3-2, followed by recommendations for further work in Section 3-3.

3-1 Discussion

In general, the resulting planning framework has indicated that optimization based trajectory planning is a feasible approach for Autonomous Ground Vehicle (AGV) operating online in real-world environments. Although it was mentioned by related research work that the major drawback of an optimization-based approach is its computational burden, we have shown that modern tools allow us to satisfy the real-time constraints. To our knowledge, we are the first to implement a trajectory optimization based approach using Model Predictive Contouring Control (MPCC) on a mobile robot, relying only on onboard perception and computational resources. The expressive nature of the optimal control structure is able to incorporate a detailed description of the planning problem, allowing us to craft appropriate planning behavior and anticipate on the predicted environment evolution. The optimization based approach met the expectation that collision avoidance guarantees can be crafted into hard constraints in the optimization procedure. Through these explicit constraints, safe behavior is observed for a large variety of chosen weights in the cost function. Through the adopted cost function, planning behavior can be tuned with intuitive weight parameters.

Our implementation of the planning framework makes use of the ACADO open source library that allowed us to generate efficient, problem-specific `C`-code to solve the problem online in the order of milliseconds. The planning framework and its side-components have been implemented in `c++`, relying on the Robotic Operating System (ROS) middleware. The problem specific generated code has as disadvantage that the structure of the optimal control problem cannot be changed online, meaning that only a fixed number of constraints and online parameters can be incorporated. For example, only a fixed number of dynamic obstacles can be taken into account. We have solved this particular problem by defining multiple obstacles in the framework beforehand, which are being parametrized outside the planning scope when no obstacles are present during operation. Another issue is that the number of reference path segments in the local reference path is fixed. Our implementation deals with this by considering the maximum required length, that is when the vehicles drives at maximum speed for the current chosen time horizon length.

3-1-1 Local Reference Path Parametrization

The local parametrization of the reference path in our framework resulted in a compact tracking formulation, without showing any degradation of tracking performance. Compared to conventional Model Predictive Contouring Control (MPCC), our local parametrization eliminates the redundant global path parametrization that is currently not being tracked in the optimal control problem. Furthermore, this allows us to efficiently apply replanning techniques on the global plan. By obeying the lower bound on the length of the local reference that we presented, tracking performance has shown to be equal to tracking performance using a globally parametrized reference path. Switching between local reference paths is designed to have smooth transition behavior since we have provided a method to maintain continuity with respect to the global plan. In our experiments, we have shown that choosing to diverge from the reference path is often a conceivable solution. Using our refined approximation of the path parameter update during execution, stable tracking behavior is obtained without being constrained to stay near the path.

3-1-2 Collision Avoidance

We have presented a solution to collision avoidance with respect to the static obstacle environment by explicitly defining feasible sets of solutions, introduced as inequality constraints. An inflated occupancy grid map of the environment is utilized to search for collision-free areas around the prediction horizon, exploiting previous solutions. The two variations that we presented as solution to avoid collisions with the static obstacle environment both showed safe performance. Results have shown that the circular region can be found in the order of microseconds, whereas our approximation of the quadrilateral, a vehicle aligned rectangle, requires search times of factor ten larger. An observed limitation of this approach, in general, is that solutions to the optimal control problem can only be found in the proximity of the previously computed prediction horizon, bounded by the collision-free space. Whereas the circular approximation has shown to be coarse in some situations, our current implementation of the polygon search is limited to 3 meter because of its computational cost. In most cases, the vehicle aligned rectangle provides a less conservative approximation of the collision-free area than the circle. Depending on the operation environment and available computational power/time, a choice can be made between both approximations to obtain satisfactory behavior. The rectangle approximation is a subset of the larger class of possible convex polygons that can be represented by our collision avoidance strategy, leaving possibilities for creating less conservative approximations of the collision-free area. The vehicle aligned rectangle search, where each side expanded equally, is particularly applicable to mobile robots operating in unstructured environments. For autonomous car applications, a different search routine or representation presumably results in a more effective feasible solution set.

Dynamic obstacle avoidance is achieved in the form of inequality constraints in the optimal control problem, by using a disc representation for the ego-vehicle and ellipses for the occupied area by the dynamic obstacles. We have corrected an approximation of the Minkowski sum of the ellipsoid and a circle that is used in related research, by defining an ellipse that bounds the true Minkowski sum. By providing a prediction of the future positions of the dynamic obstacles in the prediction horizon, our planning framework anticipates on the evolution of the obstacle environment. It is, however, observed that performance of our optimal controller degrades when the obstacle predictions do not match their future behavior. Therefore, we have proposed to extend our method to account for prediction uncertainty regarding the future obstacle positions within the prediction horizon.

The planning framework employs a cost specifically targeted at maintaining clearance from moving obstacles in its environment. In practice, it is observed that this penalty effectively allows us to make a trade-off between a compliant clearance from obstacles and a certain dedication to keep traversing the environment along the global plan. The current form of the repulsive penalty is not taking into account the heading and velocity of the vehicle or obstacle, nor the placement of the obstacle with respect to the robot. We argue that a more sophisticated designed repulsion can create a more efficient approach in navigating through dynamic environments.

3-1-3 Experimental results

We have presented performance results in real-world experiments using a mobile robot, showing the applicability to real navigation scenarios and compatibility with the navigation system, real sensor data, and the hardware setup. A localization system has been build that fuses LiDAR scan matches, wheel odometry data, and Inertial Measurement Unit (IMU) data to obtain an accurate position update. Furthermore, the 3D LiDAR point cloud is consulted to detect moving obstacles, using a clustering technique. Real-world experiments have shown that our current perception module lacks the ability to provide a velocity prediction to the planning framework. We have performed the real-world experiments using a static obstacle assumption since there was no velocity information available of the moving obstacles. Increasing the weight of repulsive penalty in the cost function allowed us to be more conservative and traverse real-world environments safely. As we have seen in the simulations where a velocity prediction was available, our planning framework is able to produce much more efficient plans with sufficient environmental information. The aforementioned limitations of the perception module of our navigation system can be solved by using, for example, an Extended Kalman Filter (EKF) for obstacle tracking.

It is observed that the repulsive penalties produced by interacting dynamic obstacles can cause the optimal control solutions to be trapped in local minima. Although this limits the AGV in its task to traverse an environment, staying in a local minimum is a rather safe solution for navigation tasks. Since these local minima are caused by moving obstacles in the environment, the problem is usually solved when the dynamic obstacles move on. Two causes are observed that make it more difficult to escape from local minima. The first cause is the initialization of the optimal control problem. Since we use the previously computed solution to initialize the optimal control problem, it is more likely that the solver will find a new solution close the initialized solution. Secondly, the constraint that defines the feasible set of solutions can be a very conservative approximation of the actual collision-free area. It is worth investigating whether a reset of the prediction horizon allows us to escape from these local minima easier.

Simulation results of an autonomous car have indicated the diverse applicability of the planning framework. A bicycle model was used for the car, where our optimal control planner computes the acceleration and steering angle corresponding to the prediction horizon directly. Our obstacle avoidance strategy also has shown to be effective for the car but requires differently tuned parameters. Furthermore, it is inspected that the kinematic model used to relate the inputs and outputs of the mobile robot can produce inaccurate motion in high speed turns since the skidding behavior is not modeled in the optimal control problem. Considering the mobile robot, we currently rely on the diff-drive controller that tries to produce our requested longitudinal and angular velocities. A better approach would be to act on the wheel velocities directly and account for the wheel slip in the motion model.

During experiments and simulations, we have occasionally experienced that the solver is not able to find a feasible solution, given the current problem definition. Infeasibility is a serious hazard for a controller that is directly acting on the actuators of an AGV. Although we can try to minimize the occurrences of infeasibility, there is no guarantee that an update of the environment from the perspective of the belief of an AGV satisfies all planning constraints. For instance, during the real-world experiments, people have been approaching the robot really fast and stepped in the collision bound of the robot. We have worked around this problem by inspecting the Karush-Kuhn-Tucker (KKT)-number to detect infeasible solutions. As soon as an infeasible solution was detected, the robot was controlled to stand still. This has shown to be a proper way of handling this problem in the mobile robot navigation tasks. Caution is required in this approach because this is certainly not a good solution for a car driving on the road. Also, infeasibility of the problem causes the solver to keep iterating until the maximum number of iterations is found. This could be solved by a detection mechanism that terminates the solver to still be able to satisfy the real-time constraint. Further research is needed to explore methods to deal with this problem.

3-1-4 Scalability of the Method

We have shown that the optimal control formulation can be solved in real-time for two non-linear vehicle models, using up to $n = 4$ moving obstacles, $n^{\text{disc}} = 3$ ego-vehicle discs and considering a prediction horizon up to $T_{\text{horizon}} = 5$ seconds in $N = 25$ discretization steps. Scalability and adaptability of the method depends on the complexity of the optimization problem. The complexity of the optimization problem is defined by the cost function, the number of constraints, the prediction horizon length and the number of states in the considered motion model. The number of Online Data Variables (ODVs) also adds to the computation time needed to set up the problem. We have inspected how physical design measures translate in complexity of the optimal control problem in terms of the number of parameters and constraints to obtain intuition about scalability. Dependent on the environment, an increased prediction horizon length can require more local reference path segments, each being defined by $10 \times N$ ODVs (8 coefficients, segment length, reference velocity). Each dynamic obstacle adds $5 \times N$ ODVs to the problem (position, heading, dimension). The circular collision constraint can be expressed by $3 \times N$ ODVs, whereas the polygon constraint needs $16 \times N$

ODV. Each ego-vehicle representing disc adds $1 \times N$ ODV. In terms of constraints of the optimization problem, the moving obstacles introduce $n \times n^{\text{disc}}$ constraints, whereas the static environment is represented by 4 linear constraints or 1 quadratic constraint for the rectangular and circular variation respectively. Additionally, each moving obstacles introduces a new term in the cost function.

There is no information available about how the parametric complexity of the problem translates in time complexity of solving the problem using `ACADO`. We have experienced that solve times grow quite quickly for increasing complexity. For example, the complete framework that we have used in the real-world experiments is solved in the order of 15 ms on our hardware, whereas a very basic implementation with no collision avoidance capabilities and only two local reference path segments has solve times < 1 ms. A frequency of 20 Hz has shown to produce stable planning results, leaving up to 50 ms to solve the optimal control problem. It is expected that there is still some slack for more complex environments, but we have no intuition about how complexity will grow for more advanced motion models.

3-2 Conclusion

The thesis work focussed on contributing to the field of motion planning techniques for autonomous vehicles using a trajectory optimization approach, specifically targeted at dynamic, unstructured environments. Three coherent research questions have been adopted to fulfill this aim. The first research questions of the thesis seeks to explore the use of MPCC for a motion planning framework dedicated to real-world navigation tasks of AGVs. The second research question concerns the design of collision avoidance strategies within the optimal control approach. Finally, the third research question considers the implementation of the framework and the integration within an autonomous navigation system, given the real-time constraints of online operation and real sensor measurements. Providing an answer to the research questions, a new planning framework has been developed that relies on the concept of MPCC. By defining the motion planning task as an optimal control problem, we have designed explicit collision constraints and trade-offs between competitive performance variables in a cost function. Following a locally parametrized reference path, our optimal controller creates anticipatory motion by computing control inputs corresponding to a finite optimal prediction horizon within the environment. Our implementation of the planning framework allows performing navigation tasks on a mobile robot, relying on onboard sensor data. It is expected and shown by simulations that our method will perform much better in the real-world when better predictions about the pedestrians are available. Furthermore, we have shown that the method can be used for different vehicle models. Improvements of the framework can be made by designing a non-uniform repulsive penalty in the cost function, investigating how to deal with infeasibility and by incorporating obstacle prediction uncertainty.

3-3 Recommendations for Future Work

Resulting from the thesis work, several recommendations are made concerning the continuation of the project. The implemented planning framework will be published as an open-source package after submission of the thesis since there is still work to be done. This work mostly concerns tidying the software. According to the following enumeration, we propose to:

1. *Investigate methods to deal with infeasibility of the optimization problem.* Infeasibility of the optimization problem is currently accounted for by inspecting the KKT-number and breaking the AGV to stand still in case of infeasibility. This approach has shown to work well in the experiments with the mobile robot, but there are improvements to be made. How we should control the AGV in case of infeasibility heavily depends on the vehicle type and navigation scenario. Further research is required that investigates how to act appropriately and how to satisfy the real-time constraint.

2. *Find a way to impose clearance from obstacles in a non-unified manner.* Investigate how we can consider the positions and velocities of the obstacle with respect to the robot to design a better repulsion penalty. It is worth investigating how this method can be approved to come up with less aggressive, efficient, and more compliant avoidance maneuvers.

3. *Implement onboard tracking and prediction capabilities for the mobile robot.* A suggestion might be to make use of different sensors (in combination with the LiDAR) to get more obstacle measurements since the LiDAR has a quite low number of detection points for nearby obstacles. A key advantage of the LiDAR is that we have full visibility around the robot, which is essential for navigation through unstructured environments. The LiDAR detections can possibly be fused with a (depth-)camera that can deliver more obstacle measurements in the forward driving direction.

4. *Connect the uncertainty related to the obstacle predictions from the perception module to the planning framework.* Appendix D proposed a valid approach to deal with obstacle uncertainty, which is already implemented. Once prediction uncertainties are available from an inference framework, real-world tests can be performed using our optimal controller.

5. *Implement reset capabilities to escape from local minima.* It was discussed that being trapped in local minima was enhanced by the static environment constraints and the initialization of the optimal control problem at the previously computed solution. A solution to this might be to reset the prediction horizon to the current position of the robot. A detection module for being stuck in such local minima should be designed and implemented to activate the reset.

6. *Identify and implement a more complex motion model of the mobile robot.* The current implementation of Local Model Predictive Contouring Control (LMPCC) for the mobile robot relies on the diff-drive controller that allows us to pass longitudinal and angular velocities. In contrast with the used kinematic model, a dynamic model can be considered that aims to model the robot skidding behavior and bypass the diff-drive controller. This will result in more accurate control commands and predicted state evolutions of the AGV.

7. *Build a more accessible user interface that allows easier user interaction.* The current implemented navigation system of the robot works as a proof of concept and the different sub-modules interact appropriately. A next step would be to build a more intuitive user interface for the system to allow for more flexible navigation tasks.

8. *Improve the vehicle aligned search routine for expanding rectangles.* The current implementation that searches for the vehicle aligned rectangles is a factor ten slower than a non-vehicle aligned search. This is due to the fact that for every accessed grid, a transformation to the world frame is performed, including a multiplication by a rotation matrix. It is presumable that better solutions exist that allow much faster computation times. Faster search routines would allow us to increase maximum search distances.

9. *Investigate the use of less conservative or more appropriate approximations of the area free of static environment collisions.* The currently discussed variations (circle and rectangle) have shown to work properly, but better variations are still to be investigated. A suggestion is to search for convex polygons that are not necessarily rectangular since these are already compatible with the implemented collision constraint.

Appendix A

Additional Experimental Results

In this section, we present additional experiments that support the statements made in Chapter 2. First, a behavioral and computational comparison will be made regarding the two variations presented for the static environment constraint in Appendix A-1. Next, Appendix A-2 aims to observe the robustness of the static environment collision constraint.

A-1 Comparing the Two Variations of Static Obstacle Avoidance

Our collision constraint that considers the static obstacle environment was introduced for two different variations in Chapter 2. Recall that the first variation represented the collision-free area as a circular inequality constraint, whereas the second variation formulates a rectangular area. This section shows the difference in performance of both approaches. A specific simulation environment is designed that seeks to explore the boundaries of operation. Fig. A-1 compares the two approaches in equal circumstances and shows the planning behavior over time. The optimal controller was set to run at 20 Hz, with a prediction horizon of 5 seconds in $N = 25$ steps. The presented scenario simulates the robot driving in a corridor, while an oncoming pedestrian is walking towards the robot with a velocity of 0.4 m/s. The reference velocity of the robot is 0.8 m/s. Below, we interpret the presented simulation results, followed by a computational comparison.

A-1-1 Interpretation of Planning Behavior

The difficulty of this particular navigation and avoidance task is that the pedestrian is located slightly above the reference path of the robot, such that the repulsive penalty pushes the robot towards the wall. A conflict occurs between the cost term that penalizes the progress along the path and the repulsion from the obstacle. The solution seems to be obvious from a human's perspective, but Local Model Predictive Contouring Control (LMPCC) is having difficulties in both cases to find a good solution. We can observe that the prediction horizon is bent towards the wall, making it hard for the optimal controller to find the solution that

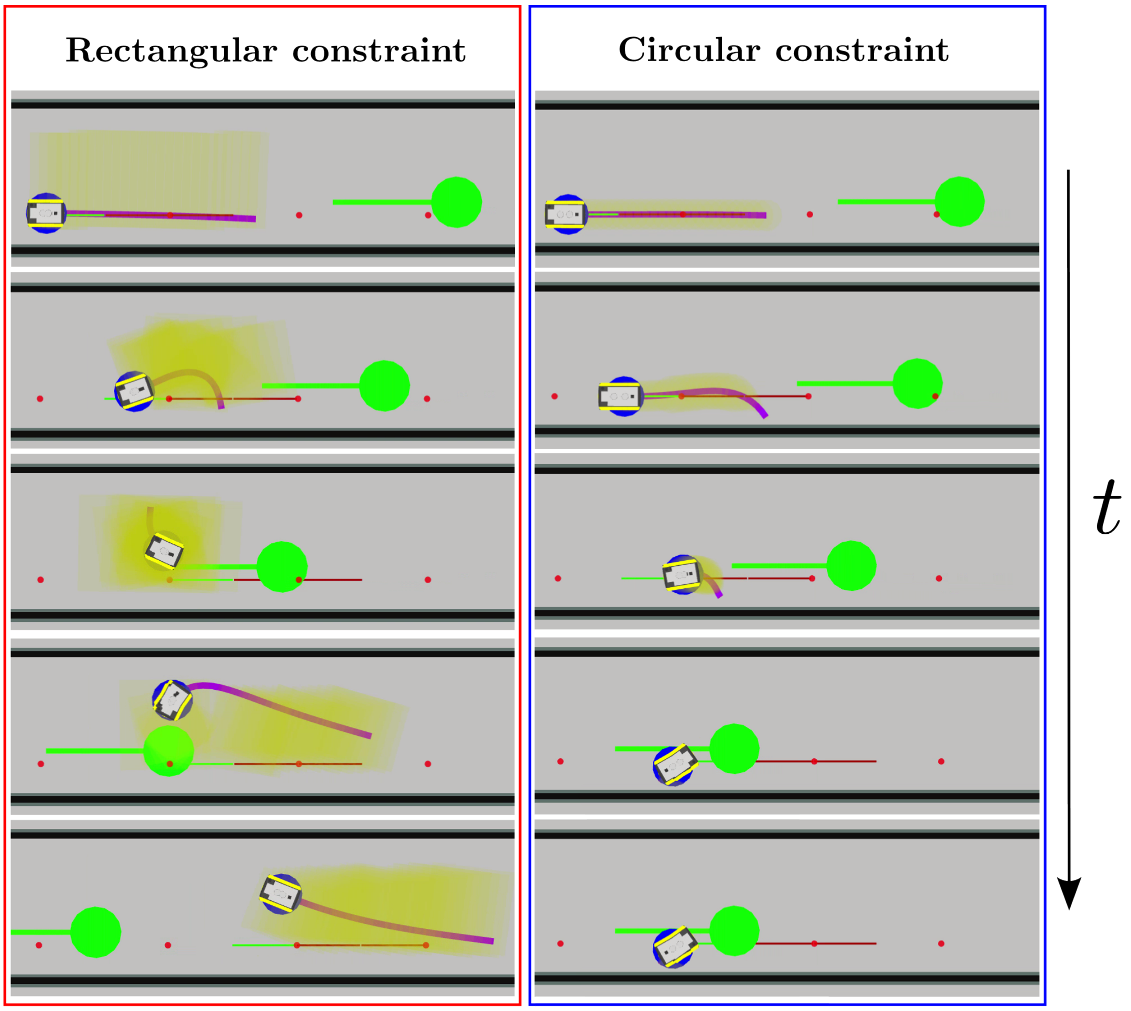


Figure A-1: Equivalent simulated navigation scenario for both constraint variations.

passes the obstacle on the left side. The reason why the robot is not able to find this solution is caused by several artifacts. First, due to the robot being trapped in this local minimum of the repulsive penalty in conflict with the lag and contour terms of the cost devoted to tracking, the prediction horizon would have to travel against a repelling force to be able to escape. Since the cost is minimized over the total horizon, traveling against this repulsive field can be an optimal solution, but the solver has difficulties finding such solutions since we are initializing the LMPCC with the previous solution, that is, on the right side of the reference path. Both constraint variations variants suffer from these complications, but there is a clear reason why the circular constraint eventually ends up in infeasible solutions and collision, and the simulation with the rectangular constraint is able to find a safe solution. Whereas the feasible solution set of the circular constraint remains very tight due to the wall that is close by, the rectangular constraint covers a large area of the corridor and allows solutions passing the obstacle on the left side. The figure shows the difference is feasible solutions along the prediction horizon by the yellow regions. Note that these regions consider the inflated static obstacle map.

A-1-2 Comparison of Computational Effort

Fig. A-2 shows the computational difference between the two discussed representations. The experiment shows the computation times for the worst case scenario: an open environment. In the case that static obstacles are present, the computation times will be lower. As expected, the search for the rectangular constraint is much slower, approximately ten times as slow. Although the search routine for both shapes is similar, the main difference comes from the fact that the incrementally expanded rectangle shape is aligned with the robot heading. In order to account for this, each searched point in the local robot frame should be multiplied by a rotation matrix to get its corresponding position in the map frame. Since a circle cannot be aligned with the heading, we only have to translate the circle to get the proper transformation. The use of the rectangular approximation at a maximum search distance of 2 m for each of the 25 positions resulted in a computation time of 9 ms of total available 50 ms each control loop.

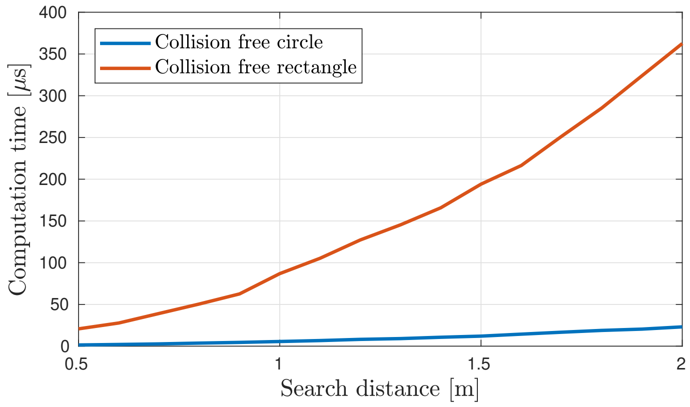


Figure A-2: Search times of the two constraint variations as a function of the search distance.

A-2 Robustness of Static Environment Collision Avoidance

In this section, we aim to get insight into the robustness of collision avoidance with respect to the static environment. We limit our analysis to the rectangular approximation of the collision-free area. Since tracking performance is defined in the cost function, it is expected to have a lower priority than the hard constraint devoted to collision avoidance. Fig. A-3 shows a simulation where it is emphasized that the constraint guarantees this safety. The mobile robot is not able to progress along the reference path since the positions that would allow the robot to do so are not adopted within the feasible solution set defined by the yellow area.

In a second experiment that we visualize in Fig. A-4, we address a larger scale simulated navigation task where the predefined global path is intentionally crossing two static obstacles of the occupancy grid map. The experiment shows that we are not constrained to the path in our effort to traverse the environment, which we will discuss according to the encircled numbers in the figure. At ②, we observe that LMPCC is able to find a clear path around the static obstacle, forced by the feasible solution set. A safe maneuver around the wall is

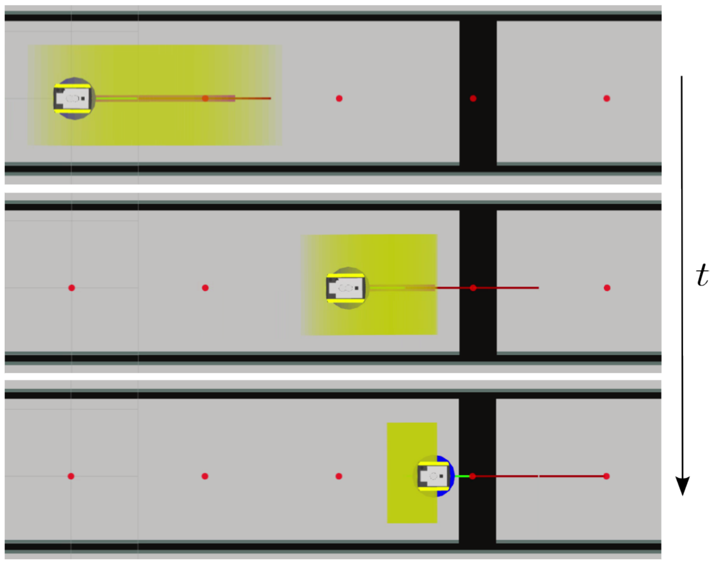


Figure A-3: Simulation results of traversing an environment with a static barrier on the reference path.

performed, after which the mobile robot returns to the reference path. The scenario at ③ shows that the reference path is going straight through a pillar. Again, the desired reference velocity along the reference path in the cost term forces the robot to diverge from the path and its forward velocity. To allow LMPCC to come up with the solutions that obediently

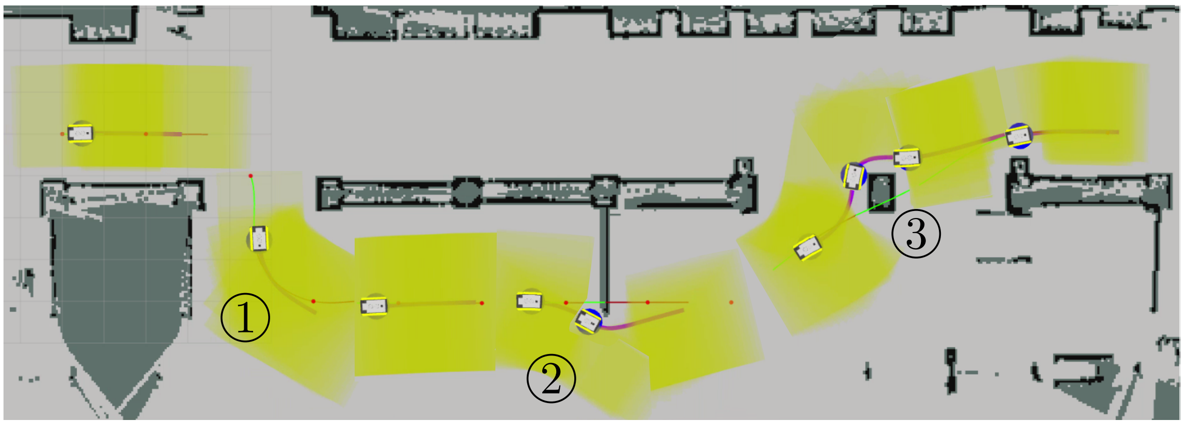


Figure A-4: Simulation results of traversing an environment with static obstacles on the reference path.

deviate from the reference path, the weight on the contour error penalty has been decreased significantly. An artifact of this choice is clearly visible in the turn at ①. It is observed that the end part of the prediction horizon is scarcely aligned with the reference path. Resulting from the low penalty on the contour error, the optimal solution yields higher velocities and less aggressive turning rather than tracking the path very accurately.

Appendix B

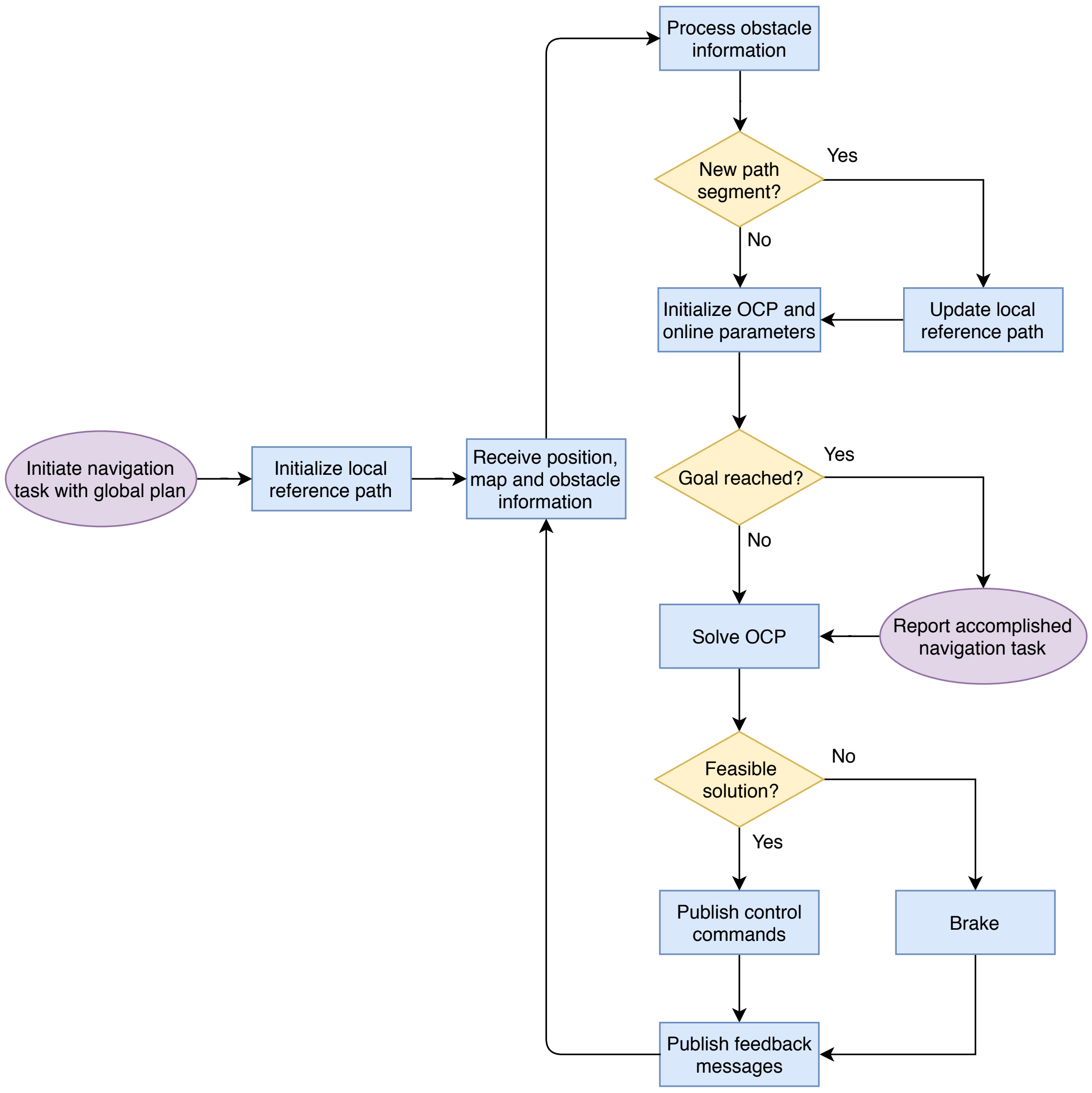
Implementation of the Optimal Controller Framework

Computational complexity is the main drawback of an optimization-based approach for motion planning. Successful efforts have been made to increase the computational efficiency of solving Model Predictive Control (MPC) problems for real-time solutions on embedded platforms, of which an overview is given by [51], [52]. One solution is an explicit approach, in which the solution for all possible problem instances is pre-computed and requires a lot of memory. We do, however, employ an approach where online solutions are provided since the look-up table of solution can grow exponentially in the number of states. The `ACADO` library [53] implements a fast Quadratic Programming (QP) solver, based on the online active set strategy [54], specifically targeted at MPC applications. In this section, we present our implementation, utilizing `ACADO`, that provides solutions to the optimization base motion planning approach online. The package is implemented in `c++` within a ROS environment. Appendix B-1 explains the architecture of our implementation and its attributes, whereas Appendix B-2 focusses on solving the MPC problem using `ACADO`.

B-1 Designing the ROS Package

The implementation of our Local Model Predictive Contouring Control (LMPCC) framework processes the global reference path, the occupancy grid map of the environment, obstacle information and vehicle positions into appropriate control commands. It does so, according to the flow diagram of Appendix B-1. The navigation task is initiated with a global plan that is used to compute the initial local reference path. In a loop, sensor data is digested to set up the Optimal Control Problem (OCP) that can be solved by `ACADO`.

As we can observe from the flow diagram, the local reference path is updated online whenever required. Furthermore, upon reaching the goal, a particular parametrization of the local reference path allows us to maintain the current position while avoiding collision with moving obstacles and the static environment. This approach ensures that the control loop remains


Initiate navigation task with global plan
Initialize local reference path
Receive position, map and obstacle information
Process obstacle information
New path segment?
Yes
No
Update local reference path
Initialize OCP and online parameters
Goal reached?
Yes
No
Report accomplished navigation task
Solve OCP
Feasible solution?
No
Yes
Publish control commands
Brake
Publish feedback messages

active and that collision avoidance is still in operation. A feasibility check on the solution of the OCP is required since we are directly acting on the actuators of the Autonomous Ground Vehicle (AGV). Currently, our method to deal with infeasibility is to break the AGV.

The problem specification that is passed to the generated OCP-solver is parametrized using Online Data Variables (ODVs). The ODVs are separately defined for each prediction step in the horizon, being are stored in one long $(N \cdot n^{\mathrm{ODV}}) \times 1$ vector, where N is the number of steps in the prediction horizon and n^{ODV} the number of ODVs. Table B-1 elaborates on the number of required ODVs for an implementation with the unicycle kinematic motion model, where n^{segments} is the number of segments in the local reference path and n^{A} the number of moving obstacles.

The remainder of this section discusses the important components of our implementation in more detail. The construction of the global plan and the initiation of the navigation task is discussed in Appendix B-1-1. Given the global plan, our framework computes the local reference path according to Appendix B-1-2. Appendix B-1-3 explains how we extract

Table B-1: Number of required Online Data Variables per prediction step.

Online Data Variable	Required number of variables
Reference path parameters	$10 \cdot n^{\text{segments}}$
Cost function weights	$4 + n^A$
Input constraints	4
Ego-vehicle disc radius	1
Dynamic obstacle parameters	$5 \cdot n^A$
Static environment collision constraint	16
Total:	$25 + 6 \cdot n^A + 10 \cdot n^{\text{segments}}$

the collision constraint with respect to the static environment from an occupancy grid map. Subsequently, Appendix B-1-4 introduces our visualization tool and Appendix B-1-5 explains how we can interact with the framework during operation.

B-1-1 Linking the MoveIt! Plugin

The link of our optimal controller with a global planning module is made through the interface of the MoveIt! ROS plugin[1]. MoveIt! is a tool that allows easy access to motion planners, such as the motion planners from the Open Motion Planning Library (OMPL). Originally designed for robotic manipulation, MoveIt! provides solutions for motion planning, manipulation, 3D perception, kinematics, control, and navigation. The planning target is represented by a planar virtual joint in $\mathcal{W}$, defined between the map fixed coordinate frame and the coordinate frame fixed to the AGV. We use a Rapidly-exploring Random Tree (RRT) to sample a global path with points $\boldsymbol{p}_m = [x_i^p, y_i^p] \in \mathcal{W}$ through the environment. As we will show in Appendix B-1-4, the planning request can be initiated through a visual `RViz` plugin. The generated global path is forwarded through the MoveIt! action server to our LMPCC framework that computes the corresponding local reference paths.

B-1-2 Computing the Local Reference Path

The waypoints of the global path are translated into the local reference paths that can be tracked by the optimal controller online. The local reference path consists of η segments, built from $\eta + 1$ local waypoints, starting from the m-th closest path segment. The x and $y-$coordinates of the path segments are separately parametrized by third order polynomials as a function of the path parameter θ, resulting in $\boldsymbol{p}^r(\theta) = [\boldsymbol{x}^r(\theta), \boldsymbol{y}^r(\theta)]^T$. [55] allows us to fit a cubic spline and to extract the polynomial coefficients. Below, we explain the method used to compute the coefficients of the $x-$coordinates of the local reference path as mentioned by [55]. The explanation omits the $y-$coordinates, since the method to obtain the corresponding coefficients is identical.

[1]https://moveit.ros.org/ (accessed November 17, 2018)

Fitting the Cubic Spline

Given the reference path points $\boldsymbol{p}_m = [x_i^p, y_i^p] \in \mathcal{W}$ and the corresponding individual segment path lengths $s_i \in \mathbb{R}$ with $i := \{m, \dots, m+\eta\}$, we define the set

$$\{[\theta_i,\, x_i]\} = \{[\sum_{m}^{i} s_i,\, x_i]\}, \quad \text{with} \quad s_i < s_{i+1} \quad \text{for} \quad i := \{m, \dots, m+\eta\}. \tag{B-1}$$

The set of local reference path segments is parametrized by θ, the cumulated segment lengths. By adopting this definition, θ equals zero at the start of each computed local reference path. Having the polynomial coefficients denoted by $[a_1^x,\, a_2^x,\, a_3^x,\, a_4^x]$, we define the piecewise cubic polynomials by

$$\boldsymbol{x}_i^r(\theta) := a_{1,i}^x(\theta - \theta_i)^3 + a_{2,i}^x(\theta - \theta_i)^2 + a_{3,i}^x(\theta - \theta_i) + a_{4,i}^x. \tag{B-2}$$

Note that $a_{4,i}^x = x_i$. The derivatives of Eq. (B-2) are given by

$$\dot{\boldsymbol{x}}_i^r(\theta) := 3a_{1,i}^x(\theta - \theta_i)^2 + 2a_{2,i}^x(\theta - \theta_i) + a_{3,i}^x, \tag{B-3}$$

$$\ddot{\boldsymbol{x}}_i^r(\theta) := 6a_{1,i}^x(\theta - \theta_i) + 2a_{2,i}^x. \tag{B-4}$$

In order to maintain continuity upto the second derivative over the individual piecewise cubic polynomials, we require that

$$\boldsymbol{x}_i^r(\theta_{i+1}) = x_{i+1} \quad \rightarrow \quad a_{1,i}^x h_i^3 + a_{2,i}^x h_i^2 + a_{3,i}^x h_i = x_{i+1} - x_i, \tag{B-5a}$$

$$\dot{\boldsymbol{x}}_{i-1}^r(\theta_i) = \dot{\boldsymbol{x}}_i^r(\theta_i) \quad \rightarrow \quad 3a_{1,i-1}^x h_{i-1}^2 + 2a_{2,i-1}^x h_{i-1} + a_{3,i-1}^x = a_{3,i}^x, \tag{B-5b}$$

$$\ddot{\boldsymbol{x}}_{i-1}^r(\theta_i) = \ddot{\boldsymbol{x}}_i^r(\theta_i) \quad \rightarrow \quad 6a_{1,i-1}^x h_{i-1} + 2a_{2,i-1}^x = 2a_{2,i}^x, \tag{B-5c}$$

where $h_i = \theta_{i+1} - \theta_i$. The requirements of Eq. (B-5a) give $3(n-1)$ equations that we rewrite by expressing a_1^x and a_3^x in terms of a_2^x and then solve for a_2^x.

$$\ddot{\boldsymbol{x}}_i^r(\theta_{i+1}) = \ddot{\boldsymbol{x}}_{i+1}^r(\theta_{i+1}) \rightarrow \quad a_{1,i}^x = \frac{a_{2,i+1}^x - a_{2,i}^x}{3h_i}, \tag{B-6a}$$

$$\boldsymbol{x}_i^r(\theta_{i+1}) = x_{i+1} \rightarrow \quad a_{3,i}^x = \frac{x_{i+1} - x_i}{h_i} - \frac{1}{3}(2a_{2,i}^x + a_{2,i+1}^x)h_i, \tag{B-6b}$$

$$\dot{\boldsymbol{x}}_{i-1}^r(\theta_i) = \dot{\boldsymbol{x}}_i^r(\theta_i) \rightarrow \quad \frac{1}{3}h_{i-1}a_{2,i-1}^x + \frac{2}{3}(h_{i-1} + h_i)a_{2,i}^x + \frac{1}{3}h_i a_{2,i+1}^x = \frac{x_{i+1} - x_i}{h_i} - \frac{x_i - x_{i-1}}{h_{i-1}}. \tag{B-6c}$$

First, we solve Eq. (B-6c), after which we use Eq. (B-6b) and Eq. (B-6a) to obtain the remaining coefficients. To obtain smooth transitions when we switch between the local reference paths, we consult the element before and after the local reference path (θ_{m-1} and $\theta_{m+\eta+1}$) to preserve continuity. We have to extrapolate to solve these appended points, resulting in one requirement less:

$$\boldsymbol{x}_{m-1}^r(\theta) := a_{2,m-1}^x(\theta - \theta_{m-1})^2 + a_{3,m-1}^x(\theta - \theta_{m-1}) + a_{4,m-1}^x, \qquad \theta \le \theta_{m-1}, \tag{B-7a}$$

$$\boldsymbol{x}_{m+\eta+1}^r(\theta) := a_{2,m+\eta+1}^x(\theta - \theta_{m+\eta+1})^2 + a_{3,m+\eta+1}^x(\theta - \theta_{m+\eta+1}) + a_{4,m+\eta+1}^x, \quad \theta \ge \theta_{m+\eta+1}. \tag{B-7b}$$

By defining zero curvature at θ_{m-1} and $\theta_{m+\eta+1}$, we require that $a_{2,m}^x = 0$ and $a_{2,m+\eta}^x = 0$.

Parametrizing the Local Reference Path Upon Reaching the Goal

In order to keep the control loop active upon reaching the goal, a special parametrization of the local reference path is required. The segment after the goal position is defined to have a reference velocity of $v_{\text{ref}} = 0$ and polynomial coefficients of

$$\boldsymbol{p}^r_{\text{goal}}(\theta) = \begin{bmatrix} \boldsymbol{x}^r_{\text{goal}}(\theta) \\ \boldsymbol{y}^r_{\text{goal}}(\theta) \end{bmatrix} = \begin{bmatrix} 0 \cdot (\theta - \theta_i)^3 + 0 \cdot (\theta - \theta_i)^2 + 0 \cdot (\theta - \theta_i) + a^x_{4,\text{goal}} \\ 0 \cdot (\theta - \theta_i)^3 + 0 \cdot (\theta - \theta_i)^2 + 0 \cdot (\theta - \theta_i) + a^x_{4,\text{goal}} \end{bmatrix} = \begin{bmatrix} x_{\text{goal}} \\ y_{\text{goal}} \end{bmatrix}. \quad \text{(B-8)}$$

B-1-3 Search Routine of the Static Environment Collision Constraint

In the scientific paper (Chapter 2), two different approaches were presented to express feasible sets of solutions, representing the area free of static obstacle collisions. This section extends the content presented in the paper and explains how the search routine is designed and implemented to find these regions in the occupancy grid map. Recall from the scientific paper that a collision-free area has to be found for each position in the optimal state sequence computed at the previous time instance $t-1$. Therefore, this section generalizes the problem by defining the method for any AGV position on the plane $\mathcal{W}$. Due to the fact that we are considering an inflated static obstacle map, all the free cells in the map correspond to positions where the AGV can be located without being in a collision. Note that this concept complicates when we use a union of discs to represent the occupied area by AGV. In that case, the inflated occupancy map is still obtained by inflating the obstacles with the radius of each disc, but all the disc positions should be located in free cells at all times. This is explicitly expressed in the collision constraint.

In our search routine, we expand AGV aligned search boundaries to access cells around the AGV to find the closest static obstacles in different directions. This principle applies to both the shapes that were presented to approximate the collision-free area. First, we present how we access the occupancy grid map to check for obstacles. Second, we will show how this can be used to build our search routine to solve the two different approaches separately.

Accessing the Occupancy Grid Map

Considering the example occupancy grid map of Fig. B-1, the red cell represents the currently occupied cell by the AGV and the black arrow its heading. A static environment is represented by the occupied black cells. The blue squares represent how we incrementally expand a search boundary to find static obstacles in the local AGV coordinate frame. Although the example shows a very coarse resolution, in reality, the occupancy grid map has a resolution of 0.05 m.

The occupancy grid map is defined on the plane $\mathcal{W} = \mathbb{R}^2$ having a size of $w_{\text{map}} \times h_{\text{map}}$, where w_{map} and h_{map} are the width and height in number of cells respectively. We define the cell size of the occupancy grid map as $\gamma \times \gamma$. The map is originated in $\boldsymbol{p}^{\text{map}}_{\mathcal{O}} = [x^{\text{map}}_{\mathcal{O}}, y^{\text{map}}_{\mathcal{O}}]$. Eq. (B-9) defines a relation between a position $\boldsymbol{p}$ on the plane $\mathcal{W}$ and the indices $\bar{\boldsymbol{p}} = [\bar{x}, \bar{y}]$ that can be used to access the occupancy grid map at the corresponding position, given that the position is within the defined region of the map.

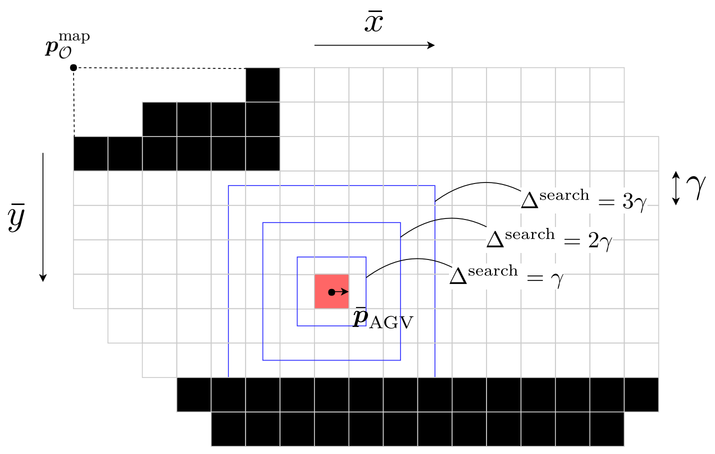

$\boldsymbol{p}_{\mathcal{O}}^{\text{map}}$
$\bar{x}$
$\bar{y}$
$\Delta^{\text{search}} = 3\gamma$
$\Delta^{\text{search}} = 2\gamma$
$\Delta^{\text{search}} = \gamma$
γ
$\bar{\boldsymbol{p}}_{\text{AGV}}$

Figure B-1: Occupancy grid map representation of the environment with incrementally expanded search boundaries defined in the local frame.

$$\bar{\boldsymbol{p}} = \begin{bmatrix} \bar{x} \\ \bar{y} \end{bmatrix} = round\left(\frac{1}{\gamma}\left(\boldsymbol{p} - \boldsymbol{p}_{\mathcal{O}}^{\text{map}}\right)\right) \tag{B-9}$$

Given AGV position $\boldsymbol{p}_{\text{AGV}}$, our search routine accesses cells around this position by expanding search boundaries in the grid map. The search iterator $\Delta^{\text{search}} \in \{\gamma, 2\gamma, \ldots, \Delta_{\text{max}}^{\text{search}}\}$ defines the expansion with steps γ of each boundary in the grid map, starting from $\boldsymbol{p}_{\text{AGV}}$.

Determining the Circular Collision Free Area

In case that we use the circular collision free area, the area is parametrized by the radius r. This means that the size of the circle is defined by the single nearest detected collision on any of the search boundaries. Fig. B-2 visualizes two scenarios where the collision-free circle is incrementally found.

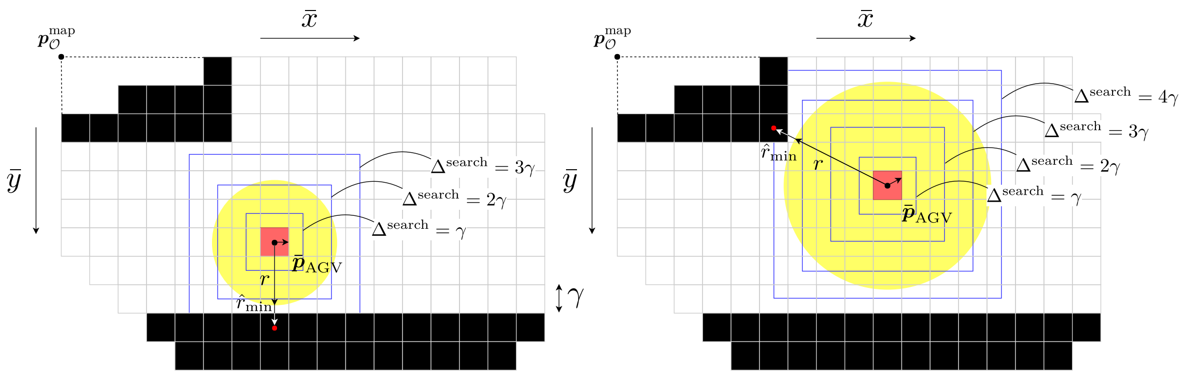

$\boldsymbol{p}_{\mathcal{O}}^{\text{map}}$
$\bar{x}$
$\bar{y}$
$\Delta^{\text{search}} = 3\gamma$
$\Delta^{\text{search}} = 2\gamma$
$\Delta^{\text{search}} = \gamma$
$\bar{\boldsymbol{p}}_{\text{AGV}}$
r
$\hat{r}_{\text{min}}$
γ
$\boldsymbol{p}_{\mathcal{O}}^{\text{map}}$
$\bar{x}$
$\bar{y}$
$\Delta^{\text{search}} = 4\gamma$
$\Delta^{\text{search}} = 3\gamma$
$\Delta^{\text{search}} = 2\gamma$
$\Delta^{\text{search}} = \gamma$
$\hat{r}_{\text{min}}$
r
$\bar{\boldsymbol{p}}_{\text{AGV}}$

Figure B-2: Incremental search for the circular collision free area in an inflated occupancy grid map with the Autonomous Ground Vehicle heading ***(left)*** aligned with the grid and ***(right)*** not aligned with the grid.

Each cell on the current search boundary, defined by Δ^{search}, is checked for occupancy and the search is stopped when an occupied cell is found or the maximum search distance is reached.

The search is, however, completed for the current Δ^{search}, as there may be multiple occupied cells for the current search distance. In the search routine, accessing $\bar{\boldsymbol{p}}^{\text{search}}$ of the occupancy grid map will reveal whether the current investigated cell is being occupied. The distance to a found occupied cell, having its position defined by $\hat{\bar{\boldsymbol{p}}}^{\text{search}}$, is denoted by $\hat{r}$, computed as:

$$\hat{r} = ||\bar{\boldsymbol{p}}_{\text{AGV}} - \hat{\bar{\boldsymbol{p}}}^{\text{search}}||_2. \tag{B-10}$$

The closest occupied cell has a distance of $\hat{r}_{\text{min}}$ and determines the radius of the collision free area. Since $\hat{r}_{\text{min}}$ expresses the distance to the closest occupied cell, a proportion of one unit cell size has to be subtracted to obtain the largest collision free circle as in Eq. (B-11). We subtract the upper bound of this proportion, half the length of the diagonal intersection of a grid cell.

$$r = \hat{r}_{\text{min}} - \frac{1}{\sqrt{2}}\gamma \tag{B-11}$$

We present the complete search routine in Algorithm 1.

Algorithm 1 Search routine to find the circular collision free area

```
1: Compute $\bar{\boldsymbol{p}}_{\text{AGV}}$ from $\boldsymbol{p}_{\text{AGV}}$ according to Eq. (B-9)
2: Initialize $\Delta^{\text{search}} = \gamma$
3: Initialize $\hat{r}_{\text{min}} = \Delta^{\text{search}}_{\text{max}}$
4: while No occupied cell is found AND $\Delta^{\text{search}} < \Delta^{\text{search}}_{\text{max}}$ do
5:     for Each cell on search boundary at $\Delta^{\text{search}}$ do
6:         if Cell is occupied at $\bar{\boldsymbol{p}}^{\text{search}}$ then
7:             Compute euclidean distance to $\bar{\boldsymbol{p}}^{\text{search}}$ according to Eq. (B-10)
8:             if $\hat{r} < r$ then
9:                 $\hat{r}_{\text{min}} = \hat{r}$
10:            end if
11:        end if
12:    end for
13:    $\Delta^{\text{search}} += \gamma$
14: end while
15: Compute $r$ from $\hat{r}_{\text{min}}$ according to Eq. (B-11)
```

Determining the Quadrilateral Collision Free Area

We approximate the convex quadrilateral collision free area by an AGV aligned rectangle in the occupancy grid map. Therefore, contrary to the circular collision free area, the search boundaries have to be expanded aligned to the AGV heading in the occupancy grid map. Recall from Chapter 2 that the collision free rectangular area is parametrized for each step in the previous optimal state sequence with $[x^{\text{min}}, x^{\text{max}}, y^{\text{min}}, y^{\text{max}}]$. An example parametrized rectangular collision free area is visualized in Fig. B-3.

During a complete search iteration of a particular Δ^{search}, we iterate over each search boundary with steps of size γ. The sampled points $\boldsymbol{p}^{\text{search}}$ on the boundaries of the current search

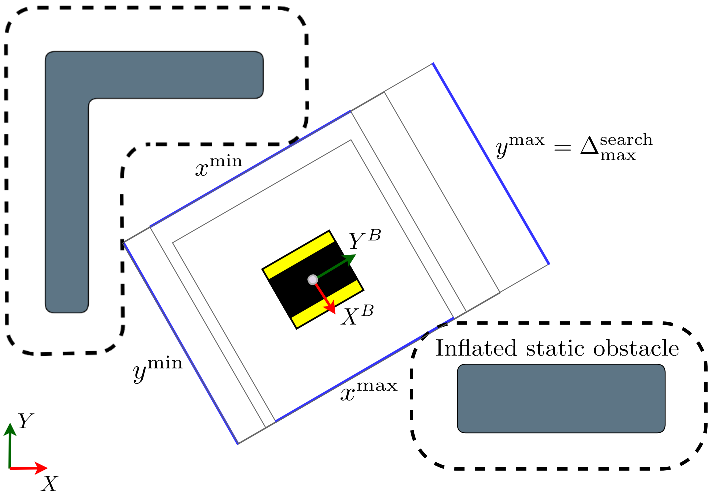


Figure B-3: Parametrization of the rectangular collision free area.

distance are translated from the local frame to grid map access indices according to Eq. (B-12). We denote the rotation matrix of the heading ψ by $R(\psi)$.

$$\bar{\boldsymbol{p}}^{\text{search}} = round\left(\frac{1}{\gamma}\Big(R(\psi)\Big(\boldsymbol{p}^{\text{search}} - \boldsymbol{p}_{\text{AGV}}\Big) + \boldsymbol{p}_{\text{AGV}} - \boldsymbol{p}^{\text{map}}_{\mathcal{O}}\Big)\right) \tag{B-12}$$

In the search routine, accessing $\bar{\boldsymbol{p}}^{\text{search}}$ of the occupancy grid map will reveal whether the current investigated cell is being occupied. Once an occupied cell on one of the search boundaries is observed, the distance of that rectangle boundary is fixed. In the following search iterations, the other search boundaries are still expanded with steps of size γ until an occupied cell is found in each direction or the maximum search distance $\Delta^{\text{search}}_{\max}$ is reached. Algorithm 2 presents the complete search routine.

Assumptions on Unknown Occupancy of Grid Cells

Earlier, we assumed that we have full knowledge of the environment in our effort to determine the nearest occupied grid cells. This does, however, not resemble a practical use case where parts of the map are not known (yet). Dependent on the application and the desired mode of operation, we can deal with unknown parts of the map in different ways. Two common approaches that we will discuss are the *free-space assumption* and the *occupied-space assumption*.

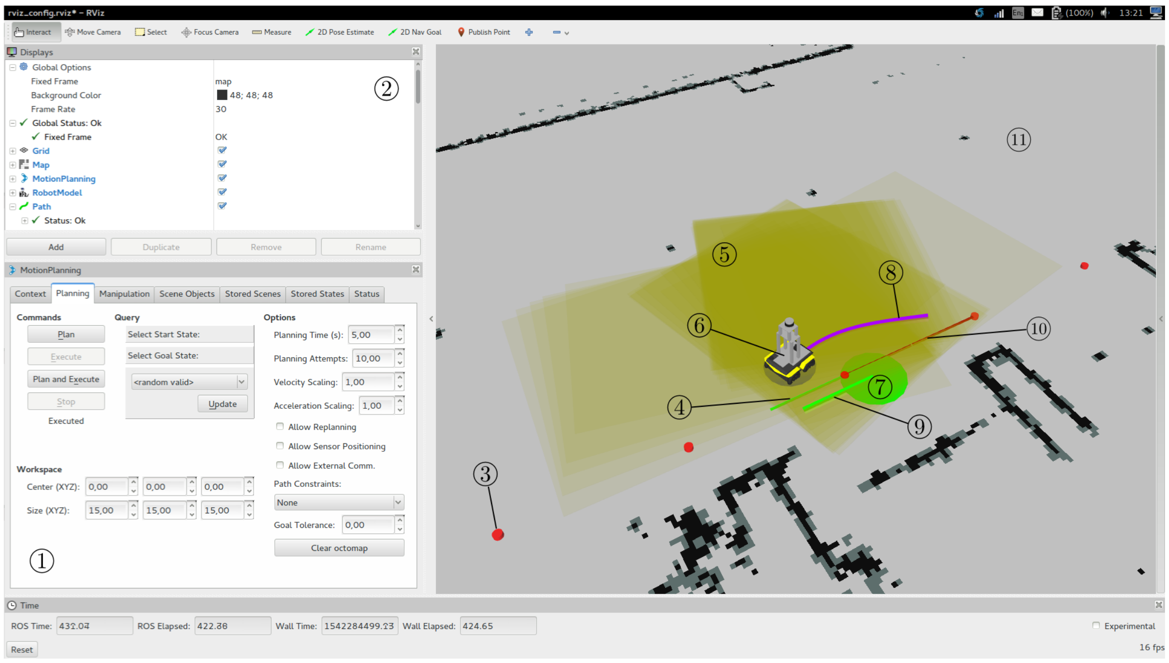
The safest way to navigate through an environment would be to assume unknown grid cells to be occupied. This approach results in feasible solution sets that always cover a known area of the map. This approach is currently being implemented in our framework since there is no guarantee that the unknown grid map cells will be updated in time during real-time operation. Entering unknown environments of the map can result in collisions if the occupancy of the cells is not updated appropriately. The free-space assumption is a less conservative approach, allowing the optimal controller to come up with solutions that are in unknown areas of the map. An advantage of this approach is that the AGV explores the environment more thoroughly.

Algorithm 2 Search routine to find the rectangular collision free area

```
1: Compute $\bar{\boldsymbol{p}}_{\text{AGV}}$ from $\boldsymbol{p}_{\text{AGV}}$ according to Eq. (B-9)
2: Initialize $\Delta^{\text{search}} = \gamma$
3: Initialize $x^{\text{min}} = y^{\text{min}} = -\Delta^{\text{search}}_{\text{max}}$, $x^{\text{max}} = y^{\text{max}} = \Delta^{\text{search}}_{\text{max}}$
4: while No occupied cell is found for each rectangle side AND $\Delta^{\text{search}} < \Delta^{\text{search}}_{\text{max}}$ do
5:     if $x^{\text{min}} = -\Delta^{\text{search}}_{\text{max}}$ then
6:         if Occupied cell is found on search boundary $x^{\text{min}}$ at distance $-\Delta^{\text{search}}$ then
7:             $x^{\text{min}} = -\Delta^{\text{search}}$
8:         end if
9:     end if
10:    if $y^{\text{min}} = -\Delta^{\text{search}}_{\text{max}}$ then
11:        if Occupied cell is found on search boundary $y^{\text{min}}$ at distance $-\Delta^{\text{search}}$ then
12:            $y^{\text{min}} = -\Delta^{\text{search}}$
13:        end if
14:    end if
15:    if $x^{\text{max}} = \Delta^{\text{search}}_{\text{max}}$ then
16:        if Occupied cell is found on search boundary $x^{\text{max}}$ at distance $\Delta^{\text{search}}$ then
17:            $x^{\text{max}} = \Delta^{\text{search}}$
18:        end if
19:    end if
20:    if $y^{\text{max}} = \Delta^{\text{search}}_{\text{max}}$ then
21:        if Occupied cell is found on search boundary $y^{\text{max}}$ at distance $\Delta^{\text{search}}$ then
22:            $y^{\text{max}} = s\Delta^{\text{search}}$
23:        end if
24:    end if
25:    $\Delta^{\text{search}} + = \gamma$
26: end while
```

B-1-4 `RViz` Visualization

Our implementation contains particular published messages that we are able to visualize within `RViz`. `RViz` is a 3D visualization tool for robots that make use of Robotic Operating System (ROS)[2]. The tool allows us to inspect planning behavior online from any computer that is linked to the ROS network, as shown by Fig. B-5. Table B-2 explains the interface windows and the specific visualized message types.



Global Options
Fixed Frame: map
Background Color: 48; 48; 48
Frame Rate: 30
Global Status: Ok
Fixed Frame: OK
Grid ✓
Map ✓
MotionPlanning ✓
RobotModel ✓
Path ✓
Status: Ok

<random valid>
Planning Time (s): 5,00
Planning Attempts: 10,00
Velocity Scaling: 1,00
Acceleration Scaling: 1,00
☐ Allow Replanning
☐ Allow Sensor Positioning
☐ Allow External Comm.
Path Constraints: None
Goal Tolerance: 0,00
Workspace
Center (XYZ): 0,00 0,00 0,00
Size (XYZ): 15,00 15,00 15,00

ROS Time: 432.04 ROS Elapsed: 422.36 Wall Time: 1542284499.23 Wall Elapsed: 424.65

Figure B-4: `RViz` visualization.

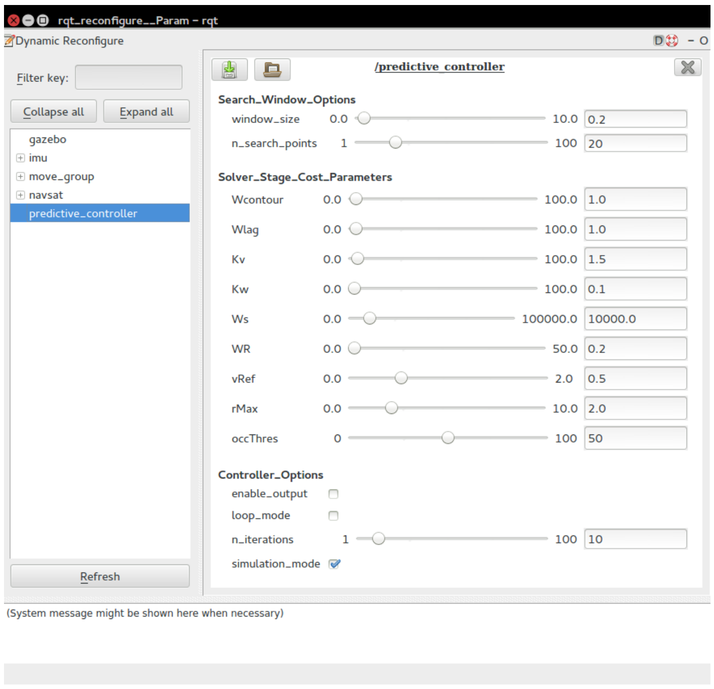
B-1-5 Dynamic Reconfigure Server

We make use of a dynamic reconfigure server[3] to be able to interact with the LMPCC online. Within this user interface, we have implemented useful functionalities that allow us to tune parameters and inspect solutions during runtime. Fig. B-5 shows the adaptable parameters in the programmed user interface. One major advantage of the available tools is the possibility to enable and disable the output. Disabling the output will stop the robot from moving, but keeps the planner running such that behavior can be inspected in `RViz`. Other parameters change weights in the cost function or influence the search routine of the path parameter.

[2]http://wiki.ros.org/rviz (accessed November 13, 2018)
[3]http://wiki.ros.org/dynamic_reconfigure (accessed 15 November, 2018)

Table B-2: Explained aspects of the `RViz` environment.

Object number	Explanation
1	MoveIt! plugin
2	Displays window to inspect available topics
3	Global path points
4	Current closest reference path segment
5	Static environment constraint
6	Mobile robot
7	Ellipsoid obstacle
8	Prediction horizon
9	Predicted obstacle position over the time horizon
10	Remaining segments of the local reference path
11	Occupancy grid map



/predictive_controller

Search_Window_Options
- window_size 0.0 – 10.0: 0.2
- n_search_points 1 – 100: 20

Solver_Stage_Cost_Parameters
- Wcontour 0.0 – 100.0: 1.0
- Wlag 0.0 – 100.0: 1.0
- Kv 0.0 – 100.0: 1.5
- Kw 0.0 – 100.0: 0.1
- Ws 0.0 – 100000.0: 10000.0
- WR 0.0 – 50.0: 0.2
- vRef 0.0 – 2.0: 0.5
- rMax 0.0 – 10.0: 2.0
- occThres 0 – 100: 50

Controller_Options
- enable_output ☐
- loop_mode ☐
- n_iterations 1 – 100: 10
- simulation_mode ☑

(System message might be shown here when necessary)

Figure B-5: Dynamic reconfigure server interface.

B-2 Solving the Optimal Control Problem with `ACADO`

`ACADO` allows us to specify the OCP for one particular use case, such that optimized code can be generated to solve that particular OCP very fast. In this section is discussed how the `ACADO` library is used to solve our problem definition of the LMPCC. Our problem definition satisfies the general form of Non-linear Model Predictive Control (NMPC), as given in Eq. (B-13), that ACADO is able to solve.

$$J^* = \min_{\boldsymbol{z}_{0:N}, \boldsymbol{u}_{0:N-1}} \sum_{k=0}^{N-1} ||h(\boldsymbol{z}_k, \boldsymbol{u}_k) - \tilde{y}_k||^2_{W_k} + ||h_N(\boldsymbol{z}_N) - \tilde{y}_N||^2_{W_N} \tag{B-13a}$$

$$\text{s.t.} \quad \boldsymbol{z}_{k+1} = F(\boldsymbol{z}_k, \boldsymbol{u}_k), \qquad \text{for } k = 0, \ldots, N-1 \tag{B-13b}$$

$$\boldsymbol{z}_k^{\texttt{lo}} \leq \boldsymbol{z}_k \leq \boldsymbol{z}_k^{\texttt{up}}, \qquad \text{for } k = 0, \ldots, N \tag{B-13c}$$

$$\boldsymbol{u}_k^{\texttt{lo}} \leq \boldsymbol{u}_k \leq \boldsymbol{u}_k^{\texttt{up}}, \qquad \text{for } k = 0, \ldots, N-1 \tag{B-13d}$$

$$\boldsymbol{r}_k^{\texttt{lo}} \leq \boldsymbol{r}_k(\boldsymbol{z}_k, \boldsymbol{u}_k) \leq \boldsymbol{r}_k^{\texttt{up}}, \qquad \text{for } k = 0, \ldots, N-1 \tag{B-13e}$$

$$\boldsymbol{r}_N^{\texttt{lo}} \leq \boldsymbol{r}_N(\boldsymbol{z}_k, \boldsymbol{u}_k) \leq \boldsymbol{r}_N^{\texttt{up}} \tag{B-13f}$$

In this general description, $\boldsymbol{z}_k \in \mathbb{R}^{n_{\boldsymbol{z}}}$ denotes the differential state and $\boldsymbol{u} \in \mathbb{R}^{n_{\boldsymbol{u}}}$ the control input. $h \in \mathbb{R}^{n_y}$ and $h_N \in \mathbb{R}^{n_{y,N}}$ are the reference functions and $W_k \in \mathbb{R}^{n_y \times n_y}$ and $W_N \in \mathbb{R}^{n_{y,N} \times n_{y,N}}$ are the weighting matrices. The (time varying) references are denoted by $\tilde{y}_k \in \mathbb{R}^{n_y}$ and $\tilde{y}_N \in \mathbb{R}^{n_{y,N}}$. Additional constraints are introduced with the constraint functions $r_k \in \mathbb{R}^{n_{r,k}}$ and $r_N \in \mathbb{R}^{n_{r,N}}$

The optimized C code that is exported is dedicated to the single OCP that is defined in the source files of the generated solver. Although the structure of the optimization problem is very strictly defined, the ACADO library allows us to pass online parameters in the form of ODV to the OCP. This allows us to have varying measurements and parameters during run-time. By allocating memory at compile time for hard-coded dimensions, the generated code uses static memory only. Also, by only allowing a constant step-size, the integrator will have a deterministic runtime. The differential equations are symbolically simplified with automatic differentiation tools. Presented details about the procedure of optimizing the code and the optimized code itself are found in the provided user manual [3]. Table B-3 gives an overview of the files that are generated by the MPCexport class.

Using the generated files in our ROS package comes down to including the acado_common.h header file in our project and calling the appropriate functions from acado_solver.c to initialize and execute the specified optimization procedure. After assigning the online parameters and the initialization for the states and control actions to an acadoVariables structure, feedback steps are performed until either a Karush-Kuhn-Tucker (KKT) condition [56] or the maximum number of iterations is satisfied. This procedure is clarified in Algorithm 3.

Algorithm 3 Solving the Optimal Control Problem with ACADO

```
1: acado_initializeSolver();
2: Initialize measured state in acadoVariables.x
3: Initialize last executed control command in acadoVariables.u
4: Assign online parameters in acadoVariables.od
5: while iter < $\text{iter}_{\max}$ ∧ acado_getKKT() > KKT threshold do
6:     acado_preparationStep()
7:     acado_feedbackStep()
8: end while
9: Extract prediction horizon states from acadoVariables.x
10: Extract prediction horizon control actions from acadoVariables.u
```

Table B-3: Generated C files by ACADO [3].

Filename	Functionality
`acado_common.h`	Contains global variable declarations and forward declarations of public API functions.
`acado_integrator.c`	Implements ODE (or DAE) and corresponding derivative evaluation and the tailored integration routine in the integrate function.
`acado_solver.c`	Implements an Gauss-Newton real-time algorithm and sets up a (condensed) QP.
`acado_qpoases_interface.hpp`	Declares an interface to call an embedded variant of qpOASES. Provides an interface to qpOASES that exploits if QP comprises only box constraints (optional).
`acado_auxiliary_functions.c`	Implements auxiliary functions for time measurements or for printing results (optional).
`test.c`	Provides a main function template to run the generated MPC or MHE algorithm. This file should serve as template that the user should modify according to her/his needs.
`MakeFile`	Provides a basic makefile to facilitate compilation of the exported code.

Appendix C

Experimental Setup

The Local Model Predictive Contouring Control (LMPCC) motion controller that has been developed during the thesis work is tested both in simulation and on a mobile robot (Fig. C-1). The setup and hardware that allowed us to do so will be presented in this section. First, Appendix C-1 presents the mobile robot. Second, Appendix C-2 explains how we set up the environment.

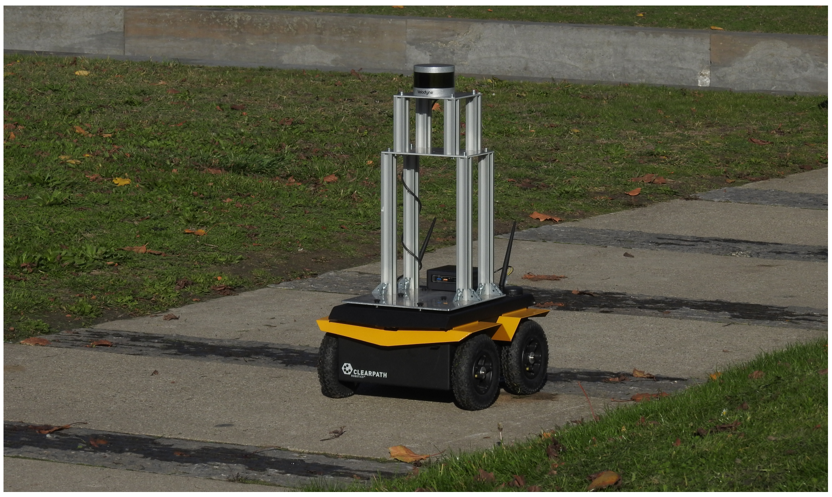


Figure C-1: Jackal robotics platform, designed by Clearpath$^{\text{TM}}$ Robotics.

C-1 Mobile Robot

Jackal, a ROS based robotic platform, designed by Clearpath$^{\text{TM}}$ Robotics, is the utilized mobile robot for the experiments. Jackal is a four-wheeled differentially driven robotic platform, making use of skid-steer locomotion. Exact measures and specifications of the robot and its internals can be found in the robot datasheet[1]. Being very flexible and compatible with a large number of accessories, Jackal is an excellent starting point for academic research on autonomous navigation.

C-1-1 Skid-Steering Locomotion

Having four powered wheels and no explicit steering mechanism, the skid-steering locomotion principle of Jackal has its mechanical simplicity as an advantage. The wheels that are mounted on the same side of the robot are driven by the same motor, mechanically connecting both wheels on each side. This means that the pairs of wheels on the left and right side always rotate at the same speed. The robust nature of the mechanical structure allows for rough terrain driving. Also, skid-steering allows the Autonomous Ground Vehicle (AGV) to rotate with a zero radius of the turn. Disadvantages of the skid-steer robot type described by [2, 57, 58] are mainly caused by the skidding principle of motion. Due to skidding, turning motion of the AGV requires significantly more power than moving straight. Furthermore, skidding causes inaccuracies in the odometry data, the tires to wear faster and less accurate motion control.

C-1-2 Equipment of the Mobile Robot

The Jackal platform comes with a dedicated internal controller board as shown in Fig. C-2, designed by the supplier. This board consists of a Micro Controller Unit (MCU), connections to the onboard sensors and a power circuit that supplies power to the system. The MCU interfaces with the Robotic Operating System (ROS) computer, the motor drivers and the available sensors. Additionally, a blue-tooth joystick controller can be used to drive the mobile robot.

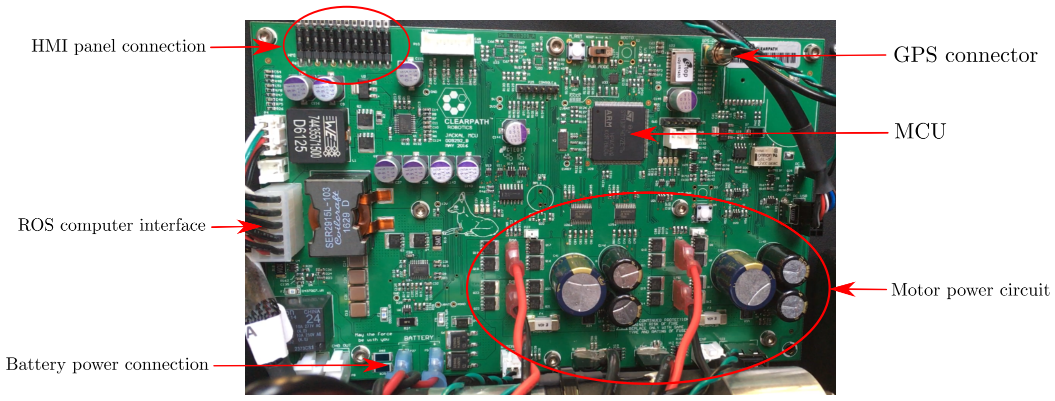


Figure C-2: Snapshot of the onboard controller board.

A Human Machine Interface (HMI) panel is available on exterior of the platform, shown by Fig. C-3. This panel allows the user to boot and shut down the robot and inspect wireless connection and battery statuses.

[1]https://www.clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/ (accessed November 14, 2018)

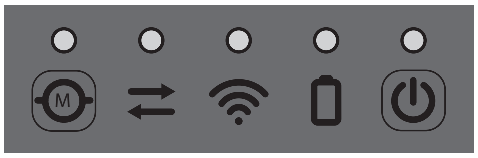


Figure C-3: Human Machine Interface panel.

Table C-1: Computer specifications.

Computer	Processor	RAM	Hard Drive
On-board computer	Celeron J1800 2.4 Ghz	2 GB	32 GB
Intel NUC	i7 7567U 4.0 GHz	32 GB	250 GB

Computational resources Jackal comes with an on-board Linux computer. Additionally, the platform has been equipped with an Intel NUC computer. Specifications of both computer are given in the table below.

Sensors The platform is equipped with several onboard sensors including Inertial Measurement Unit (IMU), Global Positioning System (GPS) and wheel encoders. On top of that, the robotic platform has been extended with exteroceptive sensors. The selected sensors are a combination of Light Detection And Ranging (LiDAR) and camera technologies, as specified in Table C-2. The available sensors can directly be integrated within the ROS framework since open-source packages are compatible with the specific sensor types. The implemented navigation system only makes use of the 3D LiDAR.

Sensor Mount We have designed and manufactured a mount to properly attach the available sensors to the robot platform. The 3D LiDAR is mounted on top to achieve full visibility with its available field of view. The 2D LiDAR is mounted in front directly on the baseplate, to perceive close range obstacles in the blind spot of the 3D LiDAR. The stereo camera is mounted just below the 3D LiDAR and allows for vertical tilting. The tilt angle of the camera can be adjusted according to the specific purpose it is to be used for. Our manufactured sensor mount is shown in the CAD model of Fig. C-4. Carefully chosen dimensions result in a sensor mount that does not limit the field of view of any of the sensors.

Power circuit The power circuit of the platform is fed by an All-Cell 26 V HE-2410 Lithium Ion battery of 27 Watt hours [5]. An internal power circuit distributes the power to the motors, the internal controller board, and the user power supply board. The user power supply board is used to power the computers and external sensors as shown in the schematic diagram of Fig. C-5.

Joystick controller The `teleop_twist_joy` [6] ROS package is used to translate the commands from the Bluetooth joystick to control commands for the mobile robot. Besides the

[5] https://www.allcelltech.com/testing/images/datasheets/naked/AllCell_Naked_24V.pdf (accessed October 31, 2018)

[6] http://wiki.ros.org/teleop_twist_joy (accessed November 15, 2018)

Table C-2: Sensor specifications.

Sensor technology	**Sensor type**	**Specifications**
2D LiDAR	Hokuyo scanning laser range finder URG-04LX[2]	Field of view (horizontal): 240° Detection distance: 20 – 5.600mm Update rate: 10 Hz Accuracy: 0.06 – 1m: ±30mm 1 – 4m: 3% of the detected distance Angular resolution (horizontal): 0.36°
3D LiDAR	Velodyne VLP-16 3D LiDAR[3]	Field of view (horizontal): 360° Detection distance: 100m Update rate: 5 – 20 Hz Accuracy: ±3cm Angular resolution (horizontal): 0.1 – 0.4° Field of view (vertical): +15.0°to −15.0°(30°) Angular resolution (vertical): 2.0°(16 channels)
Stereo camera	ZED stereo camera[4]	Field of view: 110° Depth range: 0.5 – 20m Update rate: up to 100 Hz Accuracy: ±1mm (position), 0.1°(orientation) Resolution: up to 4416 × 1242

standard functionalities in this package, we have implemented several new features as shown by Fig. C-6. These features include abilities to adapt the driving speed and commands to the motion planner. Source code of our implementation is available online[7].

[7]https://github.com/bfloor/teleop_twist_joy

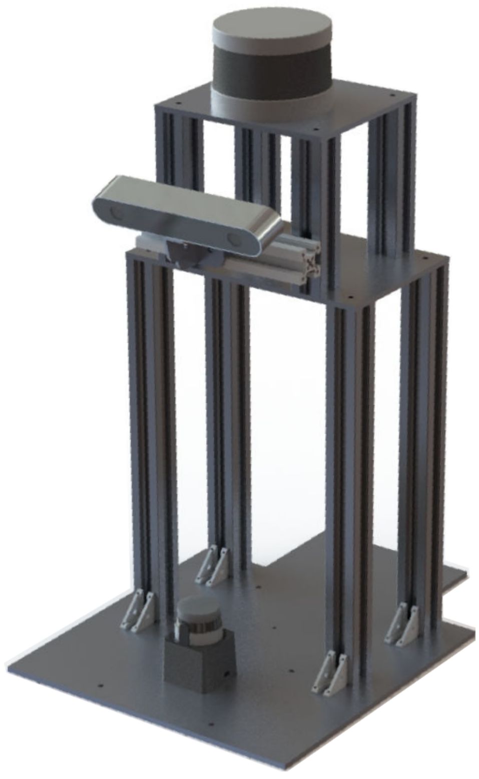


Figure C-4: CAD drawing of the designed sensor mount

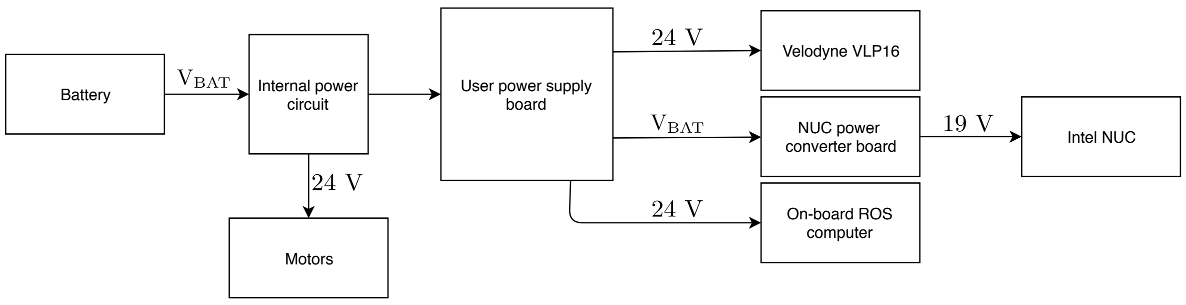
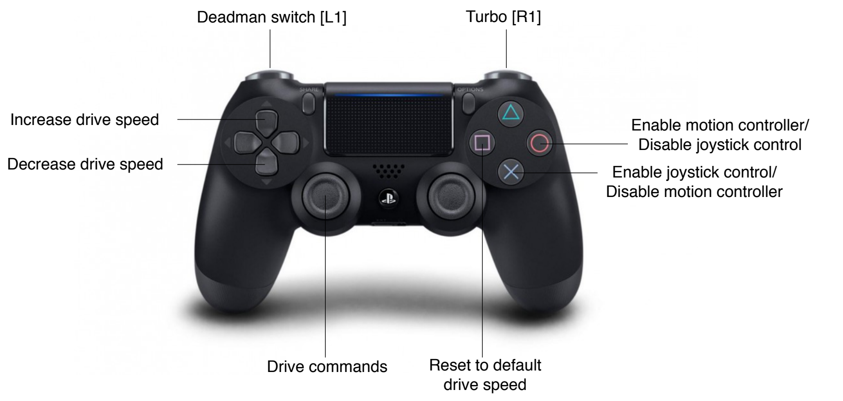


Figure C-5: Schematic of the onboard power distribution circuit.



Deadman switch [L1]
Turbo [R1]
Increase drive speed
Decrease drive speed
Enable motion controller/ Disable joystick control
Enable joystick control/ Disable motion controller
Drive commands
Reset to default drive speed

Figure C-6: Joystick controls layout.

C-1-3 Kinematic Motion Model

A mathematical model of the non-holonomic skid steering robot including the kinematics, dynamics and drive subsystems is presented in [2]. Guided by this derivation, we will derive the kinematic relations relevant to our mobile robot and controller design. A free body diagram is shown in Fig. C-7 to give an overview of the parameters used in the kinematic model.

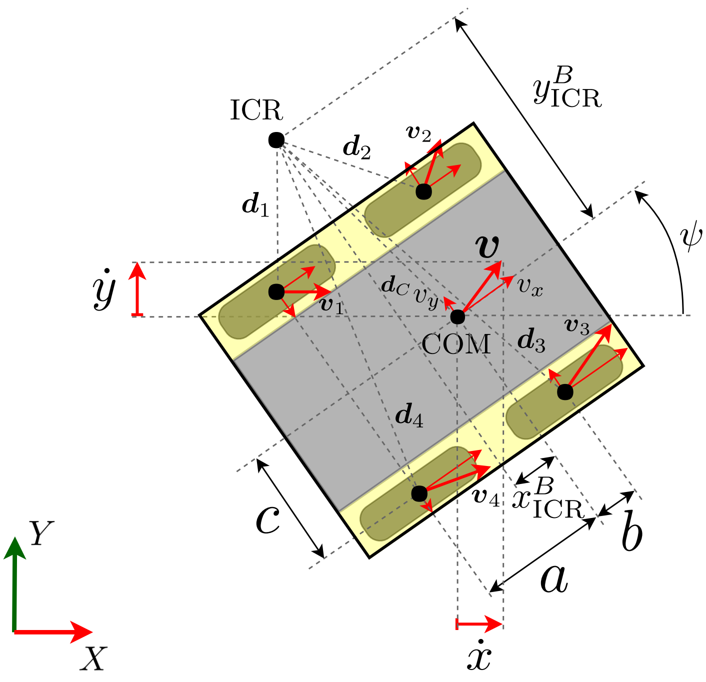


Figure C-7: Free body diagram of a four-wheeled skid-steering Autonomous Ground Vehicle with generalized planar velocities [2].

Assuming that the AGV moves on the $x-y$ plane $\mathcal{W} = \mathbb{R}^2$, we can define the state $\boldsymbol{z}$ of the AGV, describing the generalized coordinates in a three dimensional configuration space. The individual states are $\boldsymbol{z} = [x,\, y,\, \psi]^T$, where x and y correspond to the position on the plane and ψ corresponds to the heading of the AGV. In the local coordinate frame of the AGV, planar movement can be described by a vector of generalized velocities $\dot{\boldsymbol{z}} = [\dot{x},\, \dot{y},\, \dot{\psi}]^T$, or in terms of the local velocity vector $\boldsymbol{v} = [v_x,\, v_y,\, \omega]$. The mapping between the generalized velocities and the local velocity vector is given by

$$\underbrace{\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix}}_{\dot{\boldsymbol{z}}} = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \underbrace{\begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix}}_{\boldsymbol{v}}. \tag{C-1}$$

Each wheel i rotates with an angular velocity ω_i, resulting in a velocity in longitudinal direction of each wheel of $v_{ix} = r_i\omega_i$, neglecting longitudinal slip. v_{ix} is the longitudinal component of the total velocity $\boldsymbol{v}_i$ and r_i is the effective rolling radius.

Now, we define the Instantaneous Center of Rotation (ICR), which is the point in the AGV coordinate frame that has no planar movement at the given instant. Thus, all points defined within the AGV frame rotate around this point. The position of the ICR in the local frame is denoted by ICR $= [x_{\text{ICR}}, y_{\text{ICR}}]$. The distance from each wheel i to the ICR is measured by $\boldsymbol{d}_i = [d_{ix}, d_{iy}]^T$. Note that $[x_{\text{ICR}}, y_{\text{ICR}}] = [-d_{xC}, -d_{yC}]$, where $\boldsymbol{d}_C$ is the distance between the Center Of Mass (COM) and the ICR Eq. (C-2) shows how the rotational velocity around the ICR, ω, is obtained from a velocity vector perpendicular to its corresponding distance vector to the ICR. Through this relation, the separate wheel velocities can be related to each other and the COM.

$$\frac{||\boldsymbol{v}_i||}{\boldsymbol{d}_i} = \frac{||\boldsymbol{v}||}{\boldsymbol{d}_C} = |\omega| \tag{C-2}$$

The relation of Eq. (C-2) also applies to the separate components of the total velocity and distance vectors as given in Eq. (C-3).

$$\frac{v_{ix}}{-d_{iy}} = \frac{v_x}{-d_{Cy}} = \frac{v_{iy}}{d_{ix}} = \frac{v_y}{d_{Cx}} = \frac{v_x}{y_{\text{ICR}}} = -\frac{v_y}{x_{\text{ICR}}} = \omega \tag{C-3}$$

Taking a closer look at the geometric relations of the individual components of the distance vectors from the wheels to the ICR in Fig. C-7, we inspect that:

$$\begin{aligned} d_{1y} &= d_{2y} = d_{Cy} + c, && \text{(C-4a)} \\ d_{3y} &= d_{4y} = d_{Cy} - c, && \text{(C-4b)} \\ d_{1x} &= d_{4x} = d_{Cx} - a, && \text{(C-4c)} \\ d_{2x} &= d_{3x} = d_{Cx} + b. && \text{(C-4d)} \end{aligned}$$

Resulting from the symmetrically geometric proportions in Eq. (C-4) and the relations in Eq. (C-3), it is found that

$$\begin{aligned} v_L &= v_{1x} = v_{2x}, && \text{(C-5a)} \\ v_R &= v_{3x} = v_{4x}, && \text{(C-5b)} \\ v_F &= v_{2y} = v_{3y}, && \text{(C-5c)} \\ v_B &= v_{1y} = v_{4y}, && \text{(C-5d)} \\ & && \text{(C-5e)} \end{aligned}$$

where v_L and v_R are longitudinal components of the left and right wheel velocities, v_F and v_B are the lateral components of the front and rear wheel velocities. Substituting Eqs. (C-4) to (C-5) into Eq. (C-3) results in Eq. (C-6), a model that maps the COM longitudinal and rotational velocity to the longitudinal and lateral wheel velocities.

$$\begin{bmatrix} v_L \\ v_R \\ v_F \\ v_B \end{bmatrix} = \begin{bmatrix} 1 & -c \\ 1 & c \\ 0 & -x_{\text{ICR}} + b \\ 0 & -x_{\text{ICR}} - a \end{bmatrix} \begin{bmatrix} v_x \\ \omega \end{bmatrix} \tag{C-6}$$

By making the assumption that the effective rolling radius of each wheel r_i is given by r, we can rewrite this into Eq. (C-7). The longitudinal and rotational velocity of the COM is expressed in terms of ω_L and ω_R, the left and right wheel velocities, respectively. $\boldsymbol{u} = [v_x,\, \omega]$ are considered to be the inputs of the system at kinematic level.

$$\underbrace{\begin{bmatrix} v_x \\ \omega \end{bmatrix}}_{\boldsymbol{u}} = r \begin{bmatrix} \dfrac{\omega_L + \omega_R}{2} \\ \dfrac{-\omega_L + \omega_R}{2c} \end{bmatrix} \tag{C-7}$$

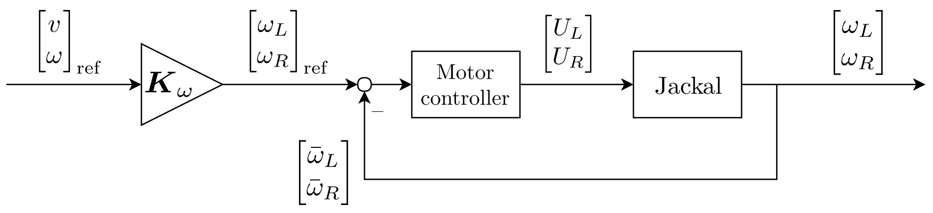
In order to include the non-holonomic property of the vehicle as a constraint in the kinematic motion model, the following equality is adopted in the model:

$$v_y + x_{\text{ICR}}\dot{\psi} = 0 \tag{C-8}$$

By assuming $x_{\text{ICR}} = 0$ (COM in center of AGV), this constraint reduces to $v_y = 0$.

C-1-4 Motor Control Scheme

The robot platform is able to follow longitudinal and rotational velocity setpoints through the use of its internal differential drive controller. Having these velocities as input, the control scheme that produces the regulated wheel velocities is shown in Fig. C-8. The internal controller can be split up into a feedforward forward kinematic translation and a motor controller. The motor controller computes the input voltage signal to the motors to achieve the desired wheel velocity. The controller is now discussed in more detail.



$\begin{bmatrix} v \\ \omega \end{bmatrix}_{\text{ref}}$
$\boldsymbol{K}_\omega$
$\begin{bmatrix} \omega_L \\ \omega_R \end{bmatrix}_{\text{ref}}$
$-$
Motor controller
$\begin{bmatrix} U_L \\ U_R \end{bmatrix}$
Jackal
$\begin{bmatrix} \omega_L \\ \omega_R \end{bmatrix}$
$\begin{bmatrix} \bar{\omega}_L \\ \bar{\omega}_R \end{bmatrix}$

Figure C-8: Control scheme of the actuating motors.

By rewriting Eq. (C-7) in Eq. (C-9), a forward gain $\boldsymbol{K}_\omega$ is obtained that directly translates the velocity setpoints v and ω into their corresponding wheel velocities ω_L and ω_R for the left and right wheel pairs, respectively. Note that this is a pure static feedforward translation and that no precautions are taken to compensate for unmodeled skidding or a kinematic model mismatch.

$$\begin{bmatrix} \omega_L \\ \omega_R \end{bmatrix} = \frac{1}{r} \begin{bmatrix} v - c\omega \\ v + c\omega \end{bmatrix} = \frac{1}{r} \underbrace{\begin{bmatrix} 1 & -c \\ 1 & c \end{bmatrix}}_{\boldsymbol{K}_\omega} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{C-9}$$

The rotational velocities of the motors that drive the left and right wheel pairs are regulated by the motor controller. A measured velocity of the left and right wheel, $[\bar{\omega}_L,\, \bar{\omega}_R]^T$, is provided by the wheel encoders as a feedback signal. The controller regulates the input voltage of the motors through a Pulse-Width Modulation (PWM) signal.

C-2 Setting up the Experiment

Two types of experiments using the mobile robot are performed. Fig. C-9 shows the communication diagram of the devices used for both experiments. The first type, represented by ①, relies on external measured positions from a motion capture OptiTrack system [59], the second type, indicated by ②, is performed using onboard measurements and processing only.

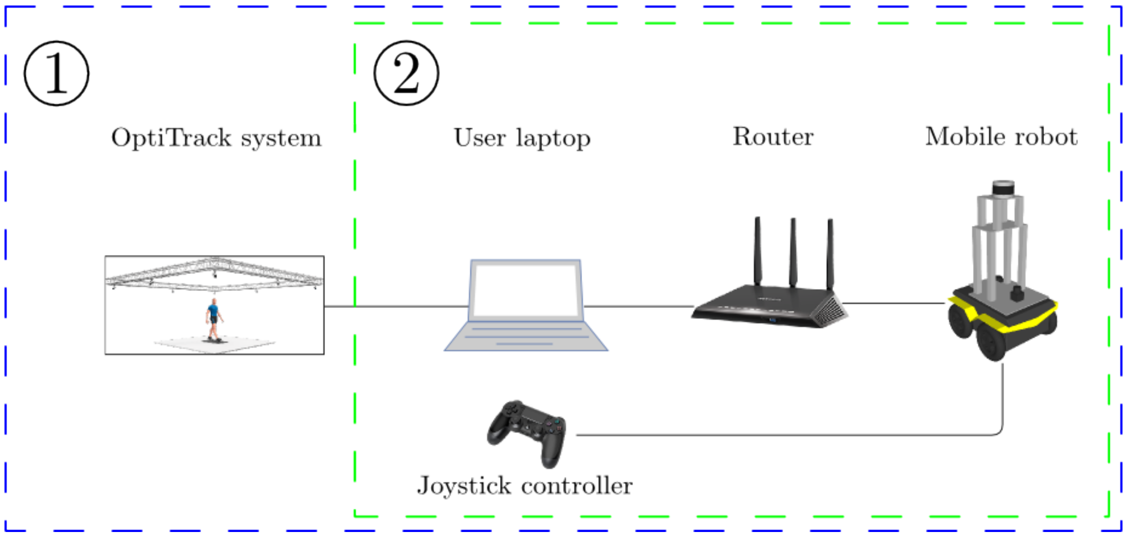


Figure C-9: Communication diagram of the two experimental setups.

For the first type of experiments, static obstacles have been constructed within the OptiTrack environment. Rectangular obstacles have been build from PVC tubes wrapped in black foil as shown in Fig. C-10. Furthermore, the interacting pedestrians are wearing helmets with markers to be detectable for the OptiTrack system.

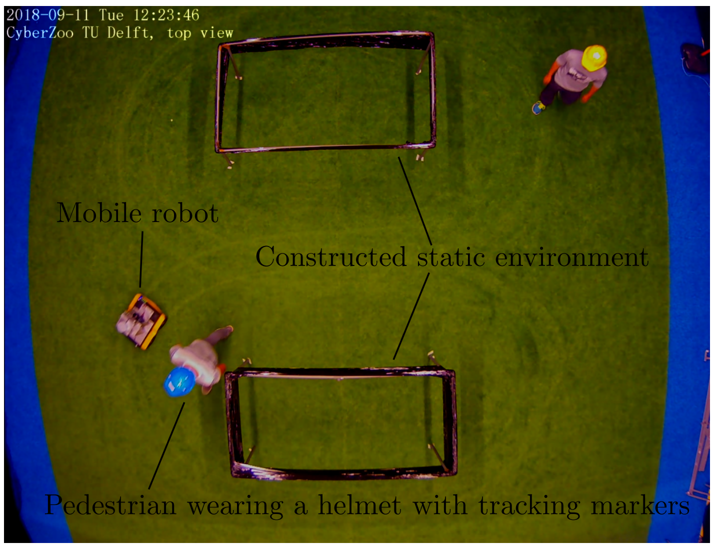

2018-09-11 Tue 12:23:46
CyberZoo TU Delft, top view
Mobile robot
Constructed static environment
Pedestrian wearing a helmet with tracking markers

Figure C-10: Constructed navigation scenario in the OptiTrack environment.

Appendix D

Dealing with Obstacle Prediction Uncertainty

This chapter presents a method to incorporate obstacle movement prediction uncertainty in the prediction horizon of our planning framework. The method is growing the occupied area of obstacles according to their prediction uncertainty over time, implemented according to [60]. Preliminary results of the implementation are presented in a simulation.

D-1 Method

The method used by [60] tries to incorporate a time-varying uncertainty related to predictions of dynamic obstacles in the optimal control problem. It is assumed that for this approach, an external inference framework is able to deliver a mean trajectory $\boldsymbol{z}_{0:N-1}^{A_i}$ of obstacle A_i with its corresponding (time varying) uncertainty $\boldsymbol{\sigma}_{0:N-1}$. Given the position uncertainty at step k, $\boldsymbol{\sigma}_k = [\sigma_k^a,\ \sigma_k^b]^T$, and the growth of uncertainty, $\boldsymbol{\sigma} = [\sigma^a,\ \sigma^b]^T$, a linear growth model for the uncertainty, $\boldsymbol{\sigma}_{k+1} = \boldsymbol{\sigma}_k + \boldsymbol{\sigma}\Delta t_k$, expresses the uncertainty over the prediction horizon of the obstacle position. Given the uncertainty model, p_ϵ denotes the probability that the position of obstacle A_i at time step k lies outside an ellipse origined at $\boldsymbol{z}_k^{A_i}$ with semi-major and semi-minor axes $a_{\boldsymbol{\sigma}_k}$ and $b_{\boldsymbol{\sigma}_k}$ respectively. The axes of the ellipse are computed as in Eq. (D-1), describing the level-set lines of the Gaussian $\mathcal{N}(0, \text{diag}(\boldsymbol{\sigma}_k))$ at level p_ϵ.

$$\begin{bmatrix} a_{\boldsymbol{\sigma}_k} \\ b_{\boldsymbol{\sigma}_k} \end{bmatrix} = \begin{bmatrix} \sigma_k^a \\ \sigma_k^b \end{bmatrix} \sqrt{\Big(-2\log(p_\epsilon 2\pi\sigma_k^a\sigma_k^b)\Big)} \tag{D-1}$$

Now, the axes of the new constraint ellipse can be found as in Eq. (D-2), adding up the inflated disc radius, the obstacle ellipse axes and the axes of the uncertainty growth. Fig. D-1 visualizes the aforementioned components of the resulting constraint ellipse.

$$\begin{bmatrix} \hat{\alpha} \\ \hat{\beta} \end{bmatrix} = \begin{bmatrix} a + \lambda r_{\text{disc}} + a_{\boldsymbol{\sigma}_k} \\ b + \lambda r_{\text{disc}} + b_{\boldsymbol{\sigma}_k} \end{bmatrix} \tag{D-2}$$

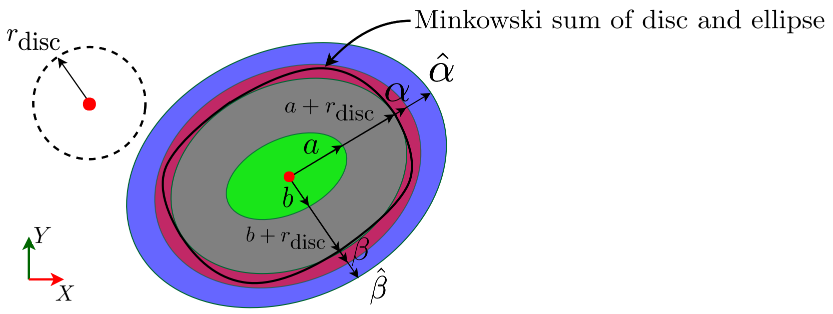


Figure D-1: Visualization of the resulting constraint-ellipse.

Adopting the new constraint axes $\hat{\alpha}$ and $\hat{\beta}$, recall from Chapter 2 that the resulting collision constraint is given by:

$$c_k^{\text{obst},j}(\boldsymbol{z}_k) = \begin{bmatrix} \Delta x_k^j \\ \Delta y_k^j \end{bmatrix}^{\mathrm{T}} R(\psi)^{\mathrm{T}} \begin{bmatrix} \frac{1}{\hat{\alpha}^2} & 0 \\ 0 & \frac{1}{\hat{\beta}^2} \end{bmatrix} R(\psi) \begin{bmatrix} \Delta x_k^j \\ \Delta y_k^j \end{bmatrix} \Bigg|_{k,j} > 1. \tag{D-3}$$

D-2 Preliminary Results

The position uncertainty has been implemented in our Local Model Predictive Contouring Control (LMPCC) framework according to the method explained in the section above. The evolution of the uncertainty over time does not need to be represented by a state within the optimal control formulation, the fixed prediction horizon time steps allow us to compute the evolution in advance for each control iteration. The resulting ellipse dimensions can be passed to the generated code through Online Data Variable (ODV) to solve the problem, resulting in negligible extra computation time.

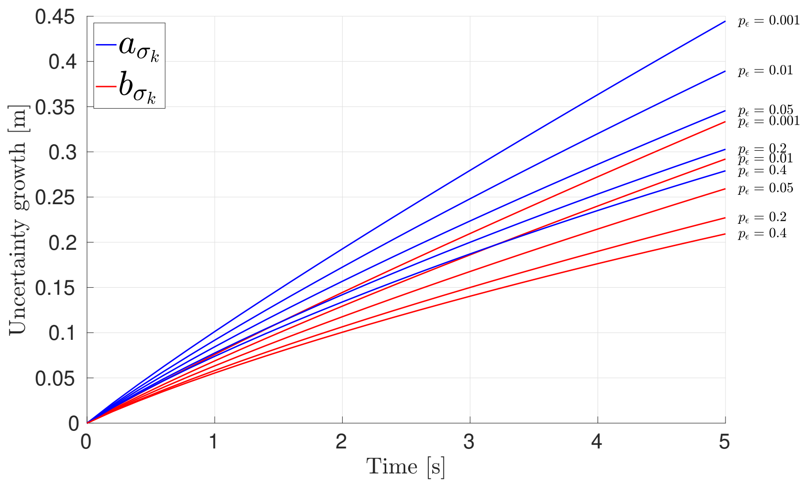


Figure D-2: Uncertainty growth of the obstacle position for different p_ϵ.

We consider an example where the uncertainty growth is given by $\boldsymbol{\sigma} = [0.02, 0.03]^T$ over a time horizon of 5 seconds in $N = 25$ steps. We want to represent the ellipsoid area occupied by obstacle A_1 with probability $1 - p_\epsilon$ using Eq. (D-1). Fig. D-2 shows the growth of the ellipsoid axes over the time horizon for several p_ϵ.

In Fig. D-3, we show a snapshot of a particular simulated overtaking maneuver of the simulated robot, considering uncertainty on the constant velocity prediction. We expand the ellipsoid obstacle according to Eq. (D-2), using $p_\epsilon = 0.05$. The robot collision space is represented by one disc, of which the evolution is given over the time horizon by the blue circles. The current obstacle position is visualized by the green ellipse and its constant velocity prediction by the green line. The red ellipses represent the growing ellipses with dimension $\hat{\alpha}$ and $\hat{\beta}$ over the time horizon. For clarity, the constraint on the static environment is not visualized. The snapshot visualizes how the optimal controller plans to pass the obstacle with more clearance because of the uncertainty growth on the ellipse. Note that the blue circles and red ellipses show to overlap between different time instances of the prediction horizon, but that this never occurs at the same time step.

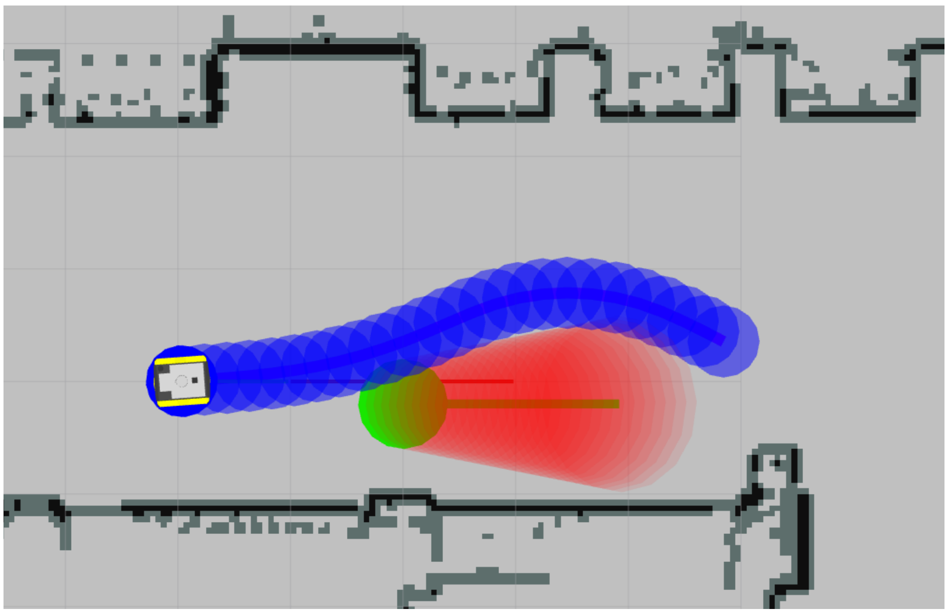


Figure D-3: Simulated overtaking scenario with prediction uncertainty on the obstacle position.

In a second scenario, shown in Fig. D-4, we simulate an oncoming pedestrian on the reference path of the mobile robot. Concurrently, a second pedestrian walking through the corridor is overtaken. In the case where no prediction uncertainty is present, the robot would be able to pass the oncoming pedestrian while simultaneously overtaking the second pedestrian. Due to the prediction uncertainty, the growing ellipses do not allow the robot to pass and LMPCC chooses to pass in a conservative manner.

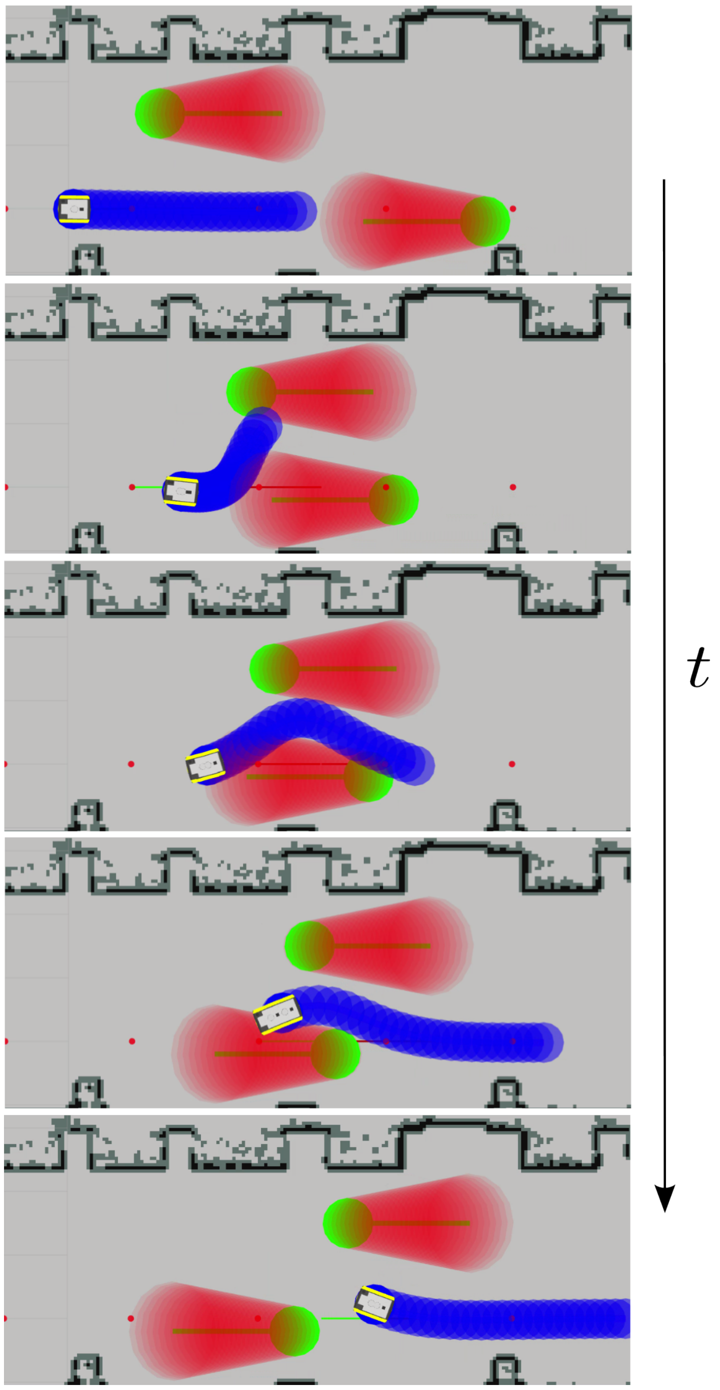


Figure D-4: Simulated pedestrian avoidance scenario with prediction uncertainty on the obstacle positions.

D-3 Conclusion

Limited sensor data and uncertainty about obstacle behavior inevitably introduces prediction uncertainty in real-world navigation scenarios. The way to deal with this uncertainty in the planning stage is to be more conservative about the places where we can safely drive. The proposed method allows us to express a time-dependent uncertainty on predicted obstacle position over the time horizon. By evaluating the uncertain position at a certain collision probability, an ellipsoid area is obtained that is adopted in our collision constraint. Simulations results have shown that our framework can deal with the implementation of prediction uncertainty, resulting in negligible additional computation time.

Appendix E

Derivation of the Minkowski Sum Bounding Ellipse

We express a closed loop form of the Minkowski sum of a circle and an ellipse, by using a bounding ellipse[1]. Our derivation of the bounding ellipse is based on the principle of curvature. More specifically, we search for an ellipse whose maximum curvature is smaller than the minimum curvature of the Minkowski sum. We start by deriving the curvature equation for the Minkowski sum.

Let us define the parametric form of the Minkowski sum of a circle with radius r and en ellipse with semi-major and semi-minor axes axis a and b, respectively as a function of τ, the angle subtended by the point at the center.

$$x_m = \cos(\tau)\left(a + \frac{br}{\sqrt{b^2\cos(\tau)^2 + a^2\sin(\tau)^2}}\right) \tag{E-1a}$$

$$y_m = \sin(\tau)\left(b + \frac{ar}{\sqrt{b^2\cos(\tau)^2 + a^2\sin(\tau)^2}}\right) \tag{E-1b}$$

By defining the first derivatives with respect to τ,

$$\frac{\partial x_m}{\partial \tau} = a\sin(\tau)\left(-1 - \frac{2\sqrt{2}abr}{\left(a^2 + b^2 + (b^2 - a^2)\cos(2\tau)\right)^{3/2}}\right), \tag{E-2a}$$

$$\frac{\partial y_m}{\partial \tau} = b\cos(\tau)\left(1 + \frac{2\sqrt{2}abr}{\left(a^2 + b^2 + (b^2 - a^2)\cos(2\tau)\right)^{3/2}},\right), \tag{E-2b}$$

[1]Hereafter, we denote the Minkowski sum of a circle and an ellipse by 'Minkowski sum'

and the second derivatives with respect to τ,

$$\frac{\partial^2 x_m}{\partial \tau^2} = \frac{a\cos(\tau)\Big(-2ab^3r + ab^3r\cos(2\tau) + 2a^3br\sin(\tau)^2 - \big(b^2\cos(\tau)^2 + a^2\sin(\tau)^2\big)^{5/2}\Big)}{\big(b^2\cos(\tau)^2 + a^2\sin(\tau)^2\big)^{5/2}}, \tag{E-3a}$$

$$\begin{aligned}\frac{\partial^2 x_m}{\partial \tau^2} =& \frac{b\Big(2ab^3r\cos(\tau)^2\sin(\tau) - \frac{1}{2}a^3br(3\sin(\tau)+\sin(3\tau))}{\big(b^2\cos(\tau)^2 + a^2\sin(\tau)^2\big)^{5/2}} \\ &- \frac{\big(b^4\cos(\tau)^4\sin(\tau) + 2a^2b^2\cos(\tau)\sin(\tau)^3 + a^4\sin(\tau)^5\big)\sqrt{b^2\cos(\tau)^2 + a^2\sin(\tau)^2}\Big)}{\big(b^2\cos(\tau)^2 + a^2\sin(\tau)^2\big)^{5/2}},\end{aligned} \tag{E-3b}$$

we can derive the curvature κ_m of the Minkowski sum:

$$\begin{aligned}\kappa_m =& \frac{\left(\frac{\partial x_m}{\partial \tau}\right)\left(\frac{\partial^2 y_m}{\partial \tau^2}\right) - \left(\frac{\partial y_m}{\partial \tau}\right)\left(\frac{\partial^2 x_m}{\partial \tau^2}\right)}{\sqrt{\left(\frac{\partial x_m}{\partial \tau}\right)^2 + \left(\frac{\partial y_m}{\partial \tau}\right)^2}^{\,3}} \\ =& \frac{ab}{\sqrt{\left(1 + \frac{2\sqrt{2}abr}{\big(a^2+b^2+(b^2-a^2)\cos(2\tau)\big)^{3/2}}\right)^2 \cdot \big(b^2\cos(\tau)^2 + a^2\sin(\tau)^2\big)^3}}.\end{aligned} \tag{E-4}$$

The maxima of the curvature can be found for the minimal radii of the Minkowski sum. Defined by the characteristic shape of the Minkowski sum, the minimal radii can be found at $\tau = 0$ and $\tau = \pi/2$. Evaluating κ_m for these τ results in

$$\kappa_m|_{\tau=0} = \frac{b}{(a^2+br)}, \tag{E-5a}$$

$$\kappa_m|_{\tau=\pi/2} = \frac{a}{(b^2+ar)}. \tag{E-5b}$$

Next, we search for the value λ that results in a lower bound on the curvature of the Minkowski sum for an ellipse whos axes are defined by $\alpha = a + \lambda$ and $\beta = b + \lambda$, solving at $\tau = 0$

$$\frac{b}{(a^2+br)} - \frac{b+\lambda}{(a+\lambda)^2} = 0, \tag{E-6}$$

and at $\tau = \pi/2$,

$$\frac{a}{(b^2+ar)} - \frac{a+\lambda}{(b+\lambda)^2} = 0. \tag{E-7}$$

Resulting in

$$\lambda|_{\tau=0} = \left\{ \frac{a^2 - 2ab + br - \sqrt{(a^2+br)((a-2b)^2+br)}}{2b}, \frac{a^2 - 2ab + br + \sqrt{(a^2+br)((a-2b)^2+br)}}{2b} \right\} \tag{E-8a}$$

$$\lambda|_{\tau=\pi/2} = \left\{ \frac{b^2 - 2ab + ar - \sqrt{(b^2+ar)((b-2a)^2+ar)}}{2a}, \right. \tag{E-8b}$$

$$\left. \frac{b^2 - 2ab + ar + \sqrt{(b^2+ar)((b-2a)^2+ar)}}{2a} \right\}. \tag{E-8c}$$

And thus, the factor λ that guarantees that the ellipse with axes $\alpha = a + \lambda$ and $\beta = b + \lambda$ has a maximum curvature smaller than the minimum curvature of the Minkowski sum is given by

$$\lambda = min \left\{ \begin{array}{c} \frac{a^2 - 2ab + br + \sqrt{(a^2+br)((a-2b)^2+br)}}{2b} \\ \frac{b^2 - 2ab + ar + \sqrt{(b^2+ar)((b-2a)^2+ar)}}{2a} \end{array} \right\}. \tag{E-9}$$

Appendix F

Submitted Conference Paper

A conference paper has been submitted to International Conference on Robotics and Automation (ICRA) 2019[1] on September 15th 2018, containing scientific contributions that were made during the thesis work. The paper will be presented in this appendix. The video that was submitted to complement the paper can be accessed via https://youtu.be/eYHPdnKa0xg.

A substantial amount of the contribution of the thesis work is already adopted within the submitted conference paper. For the thesis, extended research is performed after submission, consisting of the following major contributions with respect to the conference paper:

- Design and implement the polygon variation of the static environment collision constraint.
- Implement the autonomous navigation system to adopt our method in real navigation tasks.
- Extended simulation work for behavioral and robustness analysis.
- Perform experiments in real-world scenarios.

[1]ICRA 2019. "Call for Papers." https://www.icra2019.org/contribute/call-for-papers (accessed October 2 2018)

Local Model Predictive Contouring Control for Dynamic Environments

Boaz Floor*, Bruno Brito*, Laura Ferranti* and Javier Alonso-Mora*

Abstract— **We present a local motion planner, namely, a Local Model Predictive Contouring Control design, for an Autonomous Ground Vehicle (AGV) traversing a dynamic environment. Our design allows the AGV to execute reactive motion while tracking a global plan, thanks to local parametrization of the path. In addition, our framework allows for avoidance of static obstacles (given in an occupancy grid map) and moving obstacles represented by ellipses. Furthermore, we provide a new bound to correct the approximation of the Minkowski sum of an ellipsoid obstacle and the union of discs representation of the controlled vehicle to guarantee collision avoidance. We show that the general definition of the framework applies to both unicycle and bicycle kinematic models, commonly used to represent robots and autonomous cars, respectively. Simulation results for a car and experimental results with a mobile robot are presented.**

I. INTRODUCTION

This paper proposes a general framework to safely navigate Autonomous Ground Vehicles (AGVs), such as mobile robots and cars, in dynamic environments. Motion planning and control for AGVs is usually addressed as two separate problems (with the planner and controller running on two different modules) [1], [2]. In particular, the motion planner generates safe, smooth, and feasible paths, while the motion controller typically aims to track this planned path (directly acting on the AGV's actuator). Commonly, motion planning procedures rely on graph-search or on randomized sampling based techniques [3]. Motion planning techniques, however, do not usually take into account that the path-following controller relies on the smoothness and kinodynamic feasibility of the reference path. This can compromise the safety of the vehicle when the controller is unable to follow the planner trajectory. FasTrack proposes a safe controller based on pre-computed safety bounds [4], [5]. The proposed method, however, can only deal with static obstacles. Reactive properties would be required in the planner to react appropriately in constantly changing and uncertain environments. Purely reactive methods, however, often lack global convergence guarantees and advanced decision making capabilities. Other approaches consider incorporating constraints in the planning stage to obtain smooth and achievable trajectories [6]. A planning paradigm that completely embraces the controller stage relies on Model Predictive Control (MPC). MPC is an optimization based technique that allows to deal with dynamic and physical constraints, while optimizing a desired performance index [7], [8]. MPC approaches allow to account for the future evolution of the environment and of the vehicle to generate anticipatory motions and compute an optimal control command [9], [10]. The authors of [11] rely on an MPC scheme for cruise control and overtaking maneuvers. The authors of [12] design an MPC controller to ensure vehicle stability using differential braking and active steering. The aforementioned approaches, however, only tackle specific driving situations and do not consider the link between the global planner and local controller. Our approach address this issue by linking the global planner and the controller. Similarly to our approach, the authors of [13] build a clothoidal constrained path from irregular GPS waypoints and employ MPC to locally track the generated path. Collision avoidance constraints, however, are not taken into account. An MPC strategy known as Model Predictive Contouring Control (MPCC) [14] allows one to track a reference path (rather than a trajectory parameterized in time) and explicitly penalize the deviation from it (in terms of contouring and lateral errors). Following the MPCC paradigm, the authors of [15] propose a Nonlinear MPCC (NMPCC) to handle static and dynamic obstacles for a driving scenario. They define the static collision constraints as limits on the reference path and assume constrained driving scenarios, such as highways. This assumption can be too restrictive for mobile robots that usually navigate in unconstrained scenarios.

We solve the motion planning and control problem by combining the strengths of global re-planning techniques with the reactive behavior of a MPC approach. In particular, we propose a reformulation of the MPCC approach, namely, a Local Model Predictive Contouring Control (LMPCC) approach. By relying on [14], [15], this work makes the following contributions:

1) *A local formulation of the contouring control problem.* LMPCC supports global replanning without the need of creating a new full path parametrization. In contrast with the original MPCC scheme, LMPCC does not require an analytical path representation as a reference. Similar to [16], we parameterize a reference path using third order spline polynomials. Compared to [16], our LMPCC requires only a local parametrization of the path making it applicable to realistic navigation scenarios.
2) *A static obstacle avoidance strategy.* This strategy explicitly constrains the AGV's dynamics at time t along the prediction horizon to an approximation of the collision-free area around the robot. The approximation

This work was supported by NOW-TTW

B. Floor and B. Brito contributed equally

*The authors are with Cognitive Robotics (CoR), department of the Faculty of Mechanical, Maritime and Materials Engineering, Delft University of Technology, 2628 CD Delft, The Netherlands {bruno.debrito,l.ferranti, j.alonsomora}@tudelft.nl

is obtained by exploiting the predicted behavior of the vehicle at time $t-1$.

3) *A bound to define dynamic collision avoidance constraints*. In particular, we correct the approximation of the Minkowski sum of the ellipsoid and a circle (previously used for dynamic collision avoidance) such that, if the constraints are satisfied, collision avoidance is guaranteed.
4) *Performance results in simulation (using a mobile robot and an autonomous car) and with real-world experiments (using a mobile robot).* Our strategy supports real-sensor data and on-board localization. Furthermore, we show that our strategy can be fully implemented on-board the mobile robot in a real navigation scenario, as our experiments show, and run in real time.

The method relies on an open-source solver [17] and will be released.

II. PRELIMINARIES

Let B denote an AGV on the plane $\mathcal{W} = \mathbb{R}^2$. B can be represented by the following discrete nonlinear system:

$$\boldsymbol{z}(t+1) = f(\boldsymbol{z}(t), \boldsymbol{u}(t)), \tag{1}$$

where $\boldsymbol{z}(t)$ and $\boldsymbol{u}(t)$ represent the state and the command at time $t \geq 0$ [1], respectively. The configuration of the AGV is denoted in configuration space $\mathcal{C} = \mathbb{R}^2 \times \mathcal{S}$ by $\boldsymbol{z}(t) \in \mathcal{C}$. The area occupied by the AGV at state $\boldsymbol{z}$ is $\mathcal{B}(\boldsymbol{z})$. $\mathcal{B}(\boldsymbol{z})$ is approximated by a union of circles, that is, $\mathcal{B}(\boldsymbol{z}) \subseteq \bigcup_{c \in \{1,\dots,n_c\}} \mathcal{B}_c(\boldsymbol{z}) \subset \mathcal{W}$, where n_c is the number of circles.

We consider static and dynamic obstacles. In particular, the static obstacle environment is assumed to be captured in an occupancy grid map. The area occupied by the static obstacles is $\mathcal{O}^{\text{static}} \subset \mathcal{W}$. Furthermore, each moving obstacle A_i is represented by an ellipse of area $\mathcal{A}_i$. Consider a set of moving obstacles A_i with $i \in \mathcal{I} := \{1, \dots, n\}$ in $\mathcal{W}$, where n can vary over time. The area occupied by all moving obstacles at time instant t is given by $\mathcal{O}_t^{\text{dyn}} = \bigcup_{i \in \{1,\dots,n\}} \mathcal{A}_i(\boldsymbol{z}_i(t))$, where $\boldsymbol{z}_i(t)$ denotes the state of A_i at time t. The size of an ellipsoid associated with the obstacle is defined by a and b, the semi-major and semi-minor axes of the ellipse, respectively.

The objective is to generate collision free motion for B through $\mathcal{W}$, from its current state to a desired end configuration. This can be formulated as an optimization problem as follows:

$$J^* = \min_{\boldsymbol{z}_{0:N}, \boldsymbol{u}_{0:N-1}} \sum_{k=0}^{N-1} J(\boldsymbol{z}_k, \boldsymbol{u}_k, \theta_k) + J(\boldsymbol{z}_N, \theta_N) \tag{2a}$$

$$\text{s.t.} \quad \boldsymbol{z}_{k+1} = f(\boldsymbol{z}_k, \boldsymbol{u}_k), \tag{2b}$$

$$\mathcal{B}(\boldsymbol{z}_k) \cap \left(\mathcal{O}^{\text{static}} \cup \mathcal{O}_k^{\text{dyn}}\right) = \emptyset, \tag{2c}$$

$$\boldsymbol{u}_k \in \mathcal{U}, \; \boldsymbol{z}_k \in \mathcal{X}. \tag{2d}$$

where $\mathcal{X}$ and $\mathcal{U}$ are the set of admissible states and inputs, respectively. $\boldsymbol{z}_{1:N}$, $\boldsymbol{u}_{0:N-1}$ are the predicted state and control, respectively over a prediction horizon T_{horizon} divided into N prediction steps. θ_k denotes the predicted progress along the reference path. J is a cost function defining the planner objectives. By solving the optimization problem above, we can obtain the optimal sequence of commands $\boldsymbol{u}^*_{0:N-1}$ to guide the AGV along the reference path.

In the remainder of the paper, we show how to formulate the objectives and how to define the collision avoidance constraints to track a local (generated) reference path in our LMPCC framework.

[1] In the remainder of the paper we omit the time dependency when it is clear from the context.

III. METHOD

The goal of the LMPCC is to generate feasible and optimal motion with respect to the defined cost along the constructed local reference path. The LMPCC framework contains the following contributions:

1) The use of a transition function between waypoints to eliminate the non-differentiable representation of the reference path (Section III-A).
2) The application of a search routine to solve the approximation of the path progress estimation (Section III-C).
3) Static obstacle avoidance by explicitly constraining the control problem to an approximation of the collision-free area (Section III-E.1).
4) A bound on the Minkowski sum of a circle and an ellipse for moving obstacles (Section III-E.2).

A. Global planning / Generation of reference path

Given a goal position $\boldsymbol{p}_{\text{goal}}$, first, we use a global planning method (such as RRT [18]) to compute a collision-free path through the static environment representation from an initial position $\boldsymbol{p}_0$. This global reference path $\mathcal{P}$ consists of M points defined as $\boldsymbol{p}_m = [x^p_m, y^p_m] \in \mathcal{W}$ with $m \in \mathcal{M} := \{1, \dots, M\}$. Then, we split the path into segments delimited by $[x^p_m, y^p_m]$ and $[x^p_{m+1}, y^p_{m+1}]$. We use cubic spline interpolation to obtain an analytical expression of the reference path for each segment, connecting each of the global reference path points with a polynomial of length s_m. The efficient implementation of [19] allows splines to be generated in $\mathcal{O}(q)$ and evaluation of the spline at a single point to be performed in $\mathcal{O}(\log(q))$, where q is the number of input data points. This efficiency motivates our choice of local spline fitting during execution. Hence, we define the piece-wise spline segments as a function of the traveled distance along the reference path θ. This path parameter equals zero at the start of each path segment and continuously increases along the segment. Fig. 1 shows three reference path segments and the projected robot position on the reference path. Each path segment is composed of the interpolated reference trajectory points concatenated with a corresponding velocity reference, that is, $\boldsymbol{R}_m := [{\boldsymbol{p}^r_m}^T, v_{\text{ref},m}]^{\text{T}}$. This velocity reference depends on the environment and should be provided by the route planning module. The cubic polynomials that define the local reference points $\boldsymbol{p}^r$ are a function of θ and are given by:

$$\boldsymbol{p}^r_m(\theta) = \begin{bmatrix} x^r_m \\ y^r_m \end{bmatrix} = \begin{bmatrix} a^x_{m,1}\theta^3 + a^x_{m,2}\theta^2 + a^x_{m,3}\theta + a^x_{m,4} \\ a^y_{m,1}\theta^3 + a^y_{m,2}\theta^2 + a^y_{m,3}\theta + a^y_{m,4} \end{bmatrix} \tag{3}$$

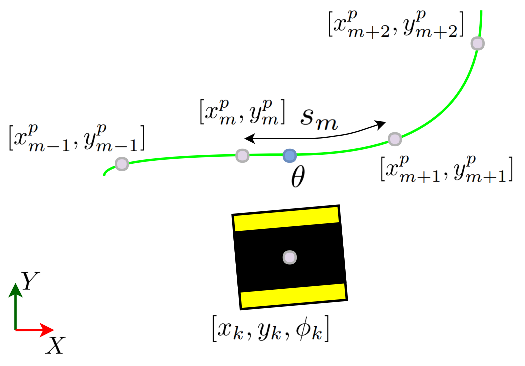


Fig. 1: Reference path representation

where $[a^x_{m,1}, \ldots, a^x_{m,4}]$ and $[a^y_{m,1}, \ldots, a^y_{m,4}]$ are the coefficients of the cubic polynomials at segment m of both splines.

We consider a limited set of path segments $\boldsymbol{R}_m \subset \mathcal{R}$ of the reference path. This allows to reduce the computational complexity of the planning problem and allows segments outside of the local reference path to change over time. We define the local reference path $\boldsymbol{L}^r$ at current time by linking η path segments, starting from the m-th closest path segment, that is,

$$\boldsymbol{L}^r = \{R_i \in \mathcal{R} | i = [m, \ldots, m+\eta]\} \tag{4}$$

B. Selecting the number of path segments

We provide a strategy to select η to guarantee that the local reference path representation captures enough information of the global path to be tracked by the LMPCC. In particular, we provide a strategy to select the length of the local reference path with respect to the prediction horizon. The number of path segments η to be included in the local reference path is a function of the prediction horizon length, the individual path segment lengths, and the speed of the AGV at each time instance, as described below:

$$\underbrace{\sum_{i=m}^{m+\eta} s_i}_{\text{Local reference path length}} \geq \underbrace{\tau \sum_{j=0}^{N} v_j}_{\text{Traveled distance in horizon}}, \tag{5}$$

where τ is the length of the discretization steps along the horizon. We use an upper bound on (5) by considering maximum longitudinal velocity $v_{\max}$ and select η such that

$$\sum_{i=m}^{m+\eta} s_i \geq T_{\text{horizon}} \cdot v_{\max} \tag{6}$$

C. Progress on reference path

MPCC keeps track of the path progress along the total reference path using the path parameter θ. Finding the corresponding path parameter, however, involves solving another optimization problem which would increase the computational cost of the algorithm [15]. If the distance between waypoints is small in relation to their curvature, spline parametrization can be regarded as reasonable approximations of arc-length parametrizations. Therefore, conventional MPCC assumes that the evolution of path parameter can be approximated by the traveled distance of the robot:

$$\theta_{k+1} \approx \theta_k + v_k \tau \tag{7}$$

where v is the forward velocity of the controlled vehicle. The estimation of the path parameter, however, has shown to be quite coarse, especially when the AGV must deviate from the reference path during an avoidance maneuver. During such a maneuver, the path parameter starts drifting. LMPCC resolves the problem of the drifting path parameter by performing line search around the estimated path parameter, as Algorithm 1 describes, resulting in the refined path variable estimation $\tilde{\theta}$. In particular, note that the evolution of the path variable in the prediction horizon still matches Eq. (7), while the initialization of the path parameter is done with the refined approximation $\tilde{\theta}_0$ at each iteration.

Algorithm 1 Refined path variable estimation

```
1: $\theta_0 = \theta_{previous} + v_k \tau$
2: window $= [\theta_0 - L_{\text{window}} : \theta_0 + L_{\text{window}}]$
3: for each $\theta_{\text{sample}}$ in window do
4:     Compute distance to $\theta_{\text{sample}}$
5:     if distance $<$ distance$_{\text{min}}$ then
6:         distance$_{\text{min}}$ $=$ distance
7:         $\tilde{\theta}_0 = \theta_{\text{sample}}$
8:     end if
9: end for
```

D. Maintaining continuity over the local reference path

To concatenate the properties of separate reference path segments into the local reference path $\boldsymbol{L}^r$ that is tracked by the LMPCC, we provide a differentiable expression of the corresponding parameters over the prediction horizon. We use a sigmoid activation function σ to link η path segments $\boldsymbol{R}_m$ as in Eq. (4). In order to connect the analytical expressions of the reference path segments (Eq. (3)), we multiply the piece-wise spline sections by their corresponding activation function $\sigma_m(\theta, s_m)$, as follows:

$$\bar{\boldsymbol{L}}^r(\theta) = \sum_{i=m}^{m+\eta} [\bar{\boldsymbol{p}}^r(\theta_k)^{\mathrm{T}}, \bar{v}_{ref}]^{\mathrm{T}} \cdot \sigma_m(\theta, s_m) \tag{8}$$

E. Collision avoidance

Figure 2 provides and overview of our collision avoidance strategy that we discuss in more details below. The AGV is represented by n_{disc} discs centered in $\boldsymbol{p}^B_j$, where $\boldsymbol{p}^B_j$ is the position on the j-th disc in the AGV body-frame and n_{disc} is the number of discs used to bound the AGV, $j \in \mathcal{J}^{\text{disc}} := \{1, 2, \ldots, n_{\text{disc}}\}$.

1) Position constraints on static obstacles: It is assumed that the reference path is free of static obstacles. When the AGV deviates from the reference path, it should be guaranteed that static obstacles are avoided. Figure 2 shows how we approximate a collision-free region along the prediction horizon. In particular, we use the shifted optimal state sequence computed at the previous time instance $t-1$,

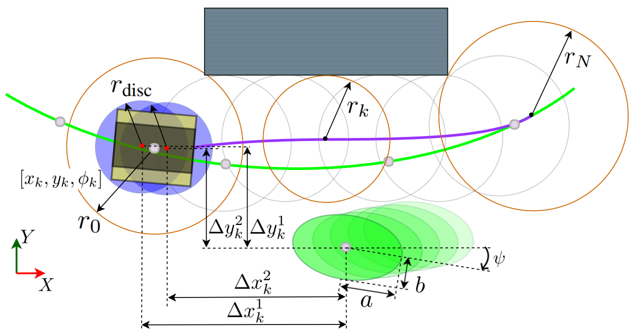


Fig. 2: Overview of our collision avoidance strategy.

namely, $\boldsymbol{z}^*_{1:N|t-1}$. Then, the N-th element to append to $\boldsymbol{z}^*_{1:N|t-1}$ is obtained by extrapolating from the last element of $\boldsymbol{z}^*_{1:N|t-1}$. The resulting vector is $\bar{\boldsymbol{z}}^*_{1:N|t-1}$. This sequence of states $\bar{\boldsymbol{z}}^*_{1:N|t-1}$ is used to center the circles depicted in Figure 2. The collision-free region is then computed as follows. The radii are found by expanding each circle in the occupancy grid map of the environment until an occupied cell is detected. The radius of the collision free circle is denoted by r_k. The approximated collision free area is introduced as an inequality constraint on the current AGV position such that $\mathcal{B}(\boldsymbol{z}_k) \cup \mathcal{O}^{\text{static}} = \emptyset \, \forall \, k \in \{0, \dots, N\}$.

Given $\bar{\boldsymbol{z}}^*_{1:N|t-1}$, we enforce that the distance of the AGV with respect to these positions must be smaller than a circle with radius $\bar{r}_{k,j} := r_k - r_{\text{disc}} - \|\boldsymbol{p}_j^B\|_2^2 \; \forall k \in \{0, \dots, N\}$:

$$c_k^{\text{env},j}(\boldsymbol{z}_k) = \bar{r}_{k,j} - \|\bar{\boldsymbol{p}}^*_{k|t-1} - \boldsymbol{p}_k\|_2^2 \Big|_{k,j} > 0 \qquad (9)$$

2) Position constraints on dynamic obstacles: Each moving obstacle A_i is represented by $\boldsymbol{z}_{\boldsymbol{i}}^{\boldsymbol{k}}$, a, and b. The collision avoidance constraints can now be defined as an inequality constraint for each j-th disc bounding the robot with respect to the distance of each obstacle $i \in \{1, \dots, n\}$ at time k as depicted in Fig. 2.

Omitting i for simplicity, the inequality constraint on each disc of the AGV with respect to the obstacles is given by:

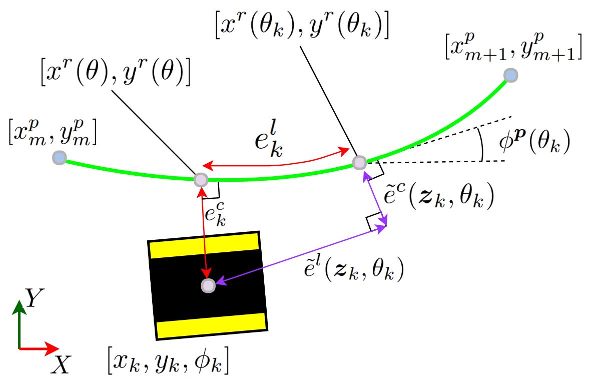
$$c_k^{\text{obst},j}(\boldsymbol{z}_k) = \begin{bmatrix} \Delta x_k^j \\ \Delta y_k^j \end{bmatrix}^{\text{T}} R(\psi)^{\text{T}} \begin{bmatrix} \frac{1}{\alpha^2} & 0 \\ 0 & \frac{1}{\beta^2} \end{bmatrix} R(\psi) \begin{bmatrix} \Delta x_k^j \\ \Delta y_k^j \end{bmatrix} \Bigg|_{k,j} > 1 \quad (10)$$

The distance from disc j to the obstacle is split into its Δx^j and Δy^j components as shown in Fig. 2. $R(\psi_k)$ is the rotation matrix corresponding to the heading of the obstacle and α and β are the resulting axes of the ellipse constraint.

It is important to notice that previous approaches assumed that the Minkowski sum of an ellipse with a circle is an ellipsoid with semi-major $\alpha = a + r_{\text{disc}}$ and semi-minor axis $\beta = b + r_{\text{disc}}$ [15]. This assumption, however, is not correct and collision can still occur [20]. In order to ensure collision-free motions the radius is enlarged by a factor λ:

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} a + \lambda r_{\text{disc}} \\ b + \lambda r_{\text{disc}} \end{bmatrix}, \qquad (11)$$

where λ is computed such that the curvature of the constraint ellipsoid is a lower bound of the curvature of the Minkowski



$[x^r(\theta_k), y^r(\theta_k)]$
$[x^r(\theta), y^r(\theta)]$
$[x^p_{m+1}, y^p_{m+1}]$
$[x^p_m, y^p_m]$
e_k^l
$\phi^{\boldsymbol{p}}(\theta_k)$
e_k^c
$\tilde{e}^c(\boldsymbol{z}_k, \theta_k)$
$\tilde{e}^l(\boldsymbol{z}_k, \theta_k)$
Y
X
$[x_k, y_k, \phi_k]$

Fig. 3: Approximated contour and lag error on the path segment.

sum, as follows:

$$\lambda = \min \begin{bmatrix} \dfrac{(a^2 + br)(1 - \frac{2ab}{a^2+br} + \sqrt{\frac{(a-2b)^2+br)}{a^2+br}})}{2b} \\ \dfrac{(b^2 + ar)(1 - \frac{2ab}{b^2+ar} + \sqrt{\frac{(-2a+b)^2+ar}{b^2+ar}})}{2a} \end{bmatrix} \qquad (12)$$

This guarantees that the constraint ellipsoid entirely bounds the collision space.

3) Repulsion from dynamic obstacles: A penalty in the cost function that resembles repulsion from dynamic obstacles ensures clearance between the AGV and the obstacles. The penalty is defined as follows:

$$J_{\text{repulsive}}(\boldsymbol{z}_k) = Q_R \sum_{i=1}^{n} \left(\frac{1}{(\Delta x_k)^2 + (\Delta y_k)^2 + \gamma} \right) \quad , \quad (13)$$

where Q_R is the weight on the repulsive forces [21]. The distance from the AGV to the dynamic obstacles is given in its separate Δx_k and Δy_k components as in Fig. 2. A small value γ is introduced to avoid a division by zero, for numerical stability of the solver [17].

F. Local Model Predictive Contouring Control

As introduced in Section III-A, the global reference path $\mathcal{P}$ consists of M segments, resulting in a local reference trajectory $\bar{\boldsymbol{L}}^r(\theta_k)$ with η segments, starting from the m-th closest path segment. The local contour error ϵ^c and the longitudinal error ϵ^l are expressed as a function of θ as visualized in Fig. 3. With $\bar{\boldsymbol{p}}^r(\theta_k)$ being the local reference path points, the contouring and longitudinal error vector $\boldsymbol{e}_k := [\tilde{\epsilon}^c(\boldsymbol{z}_k, \theta_k), \tilde{\epsilon}^l(\boldsymbol{z}_k, \theta_k)]^{\text{T}}$ is defined as follows:

$$\boldsymbol{e}_k = \begin{bmatrix} \sin \phi(\theta_k) & -\cos \phi(\theta_k) \\ -\cos \phi(\theta_k) & -\sin \phi(\theta_k) \end{bmatrix} \left(\boldsymbol{p}_k - \bar{\boldsymbol{p}}^r(\theta_k) \right). \qquad (14)$$

In order to ensure progress along the reference path, the conventional MPCC scheme proposes to add a negative cost term proportional to the travelled distance. In contrast, we introduce a cost term that penalizes the deviation of the vehicle velocity v_k with respect to a reference velocity v_{ref}.

The reference velocity is provided according to Eq. (8), allowing the vehicle to adapt its speed according to the path segment that it is tracking. Now, the LMPCC tracking cost is defined as $J_{\text{tracking}}(\boldsymbol{z}_k, \theta_k) = \boldsymbol{e}_k^T Q_\epsilon \boldsymbol{e}_k + Q_v(v_{ref} - v_k)^2$, Additionally, the cost function also penalizes with weight Q_u the inputs, that is, $J_{\text{input}}(\boldsymbol{z}_k, \theta_k) = \boldsymbol{u}_k^T Q_u \boldsymbol{u}_k$. We can now formulate our LMPCC control problem:

$$\begin{aligned} J^* = \min_{\boldsymbol{z}_{0:N}, \boldsymbol{u}_{0:N-1}} & \sum_{k=0}^{N-1} J(\mathbf{z}_k, \mathbf{u}_k, \theta_k) + J(\boldsymbol{z}_N, \theta_N) \\ \text{s.t.} \quad & \boldsymbol{z}_{k+1} := f(\boldsymbol{z}_k, \boldsymbol{u}_k), \\ & \theta_{k+1} = \theta_k + v_k \tau, \\ & \boldsymbol{z}_{\min} \leq \boldsymbol{z}_k \leq \boldsymbol{z}_{\max}, \\ & \boldsymbol{u}_{\min} \leq \boldsymbol{u}_k \leq \boldsymbol{u}_{\max}, \\ & c_k^{\text{env}}(\boldsymbol{z}_k) > 0, \\ & c_k^{\text{obst},j}(\boldsymbol{z}_k) > 1, \quad \forall j, \forall \text{obst} \\ & \mathbf{z}_0 = z_{\text{init}}, \, \theta_0 = \tilde{\theta} \end{aligned} \tag{15}$$

where $J(\mathbf{z}_k, \mathbf{u}_k, \theta_k) := J_{\text{tracking}}(\boldsymbol{z}_k, \theta_k) + J_{\text{repulsive}}(\boldsymbol{z}_k) + J_{\text{input}}(\boldsymbol{u}_k)$. Algorithm 2 summarizes our design. Each control iteration, feedback steps are performed until either a Karush-Kuhn-Tucker (KKT) condition [22] or the maximum number of iterations is satisfied (line 10).

Algorithm 2 Local Model Predictive Control

```
1:  Given $z_0$, $z_{\text{goal}}$, $\mathcal{O}^{\text{static}}$, $\mathcal{O}_k^{\text{dyn}}$, and $N$
2:  Initialization: $k = 0$
3:  Build global path $\mathcal{P}$ Eqs. (3) and (8)
4:  Select $\eta$ according to Eq. (6)
5:  Build $\bar{\boldsymbol{L}}^r(\theta_k)$ according to Eqs. (3) and (8)
6:  while $\boldsymbol{z}_k \neq \boldsymbol{z}_{\text{goal}}$ in parallel do
7:      Process sensor data
8:      Estimate $\tilde{\theta}_0$ according to Algorithm 1
9:      Compute $r_k$ along $\bar{\boldsymbol{z}}^*_{1:N|t-1}$
10:     while $\texttt{iter} < \texttt{iter}_{\max} \wedge$ KKT > threshold do
11:         Solve Eq. (15)
12:         $\texttt{iter} = \texttt{iter} + 1$
13:     end while
14:     Apply $\boldsymbol{u}_0^*$
15:     if $\theta_k > s_k$ then
16:         $k = k + 1$
17:         Re-plan global path $\mathcal{P} \setminus \boldsymbol{p}^r$
18:         Build $\bar{\boldsymbol{L}}^r(\theta_k)$ according to Eqs. (3) and (8)
19:     end if
20: end while
```

IV. Results

This section presents the results obtained using the proposed planner. In the following, we provide simulation results for a mobile robot and autonomous car. Furthermore, we present real-world experiments on the mobile robot [23].

TABLE I: Computation times

	Mean	Minimum	Maximum	Variance
Autonomous car	12.7 ms	4.8 ms	28.8 ms	0.0367
Robot	3.2 ms	1.7 ms	15.3 ms	0.0017
Robot and obstacles	11.0 ms	3.5 ms	265.2 ms	0.0151

A. Experiment and Simulation Setup

We relied on ACADO [17] to solve the proposed LMPCC and Gazebo [24] as simulation environment. In addition, we implemented the global planner using the RRT-connect from the Open Motion Planning Library (OMPL) [25]. A smoothened global path was constructed by building a clothoid [26], connecting each waypoint obtained from the RRT. The waypoints of this global path were used to generate the local reference paths consisting of 3rd order polynomials as in Eq. (8).

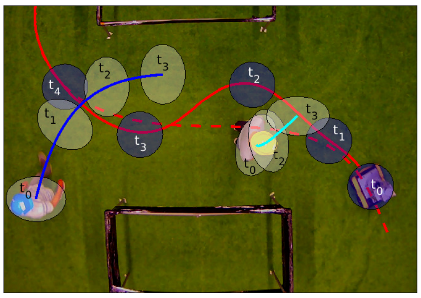
The experiments on the mobile robot were performed using the on-board localization and all algorithms were running on-board. Furthermore, we used the OptiTrack system [27] as ground-truth for the executed robot trajectories. We implemented our design in C++ and we will release the code as an open source ROS package. A video demonstrating the results accompanies this paper. Finally, the LMPCC was set to run at 20Hz considering a prediction horizon of 5 sec with $N = 25$ steps. Table I shows the computation times for the simulated car and the robot platform. Real-time performance is satisfied for the both cases. The only situation where the computation times exceeded the real-time constraint during the experiment was when interacting obstacles (pedestrians) violated the collision bound of the mobile robot.

B. Mobile robot

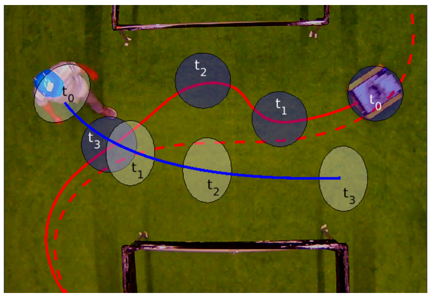
Two experiments are performed using a mobile robot platform. These are the first experiment a motion controller is implemented with only on-board perception and processing. A unicycle kinematic motion model was used to model the robot dynamics [28]. First, we tested tracking performance without obstacles at a reference velocity of $v_{\text{ref}} = 1$ m/s (Figure 5). Second, we tested tracking and obstacle avoidance in the same environment using a reference velocity of $v_{\text{ref}} = 0.7$ m/s. The obstacles are two pedestrians walking around (Figure 4.a)-b)). An occupancy map of the static environment was built a priori and only the position of the pedestrians is provided by the OptiTrack system.

As the Figure 5, the LMPCC achieved perfect tracking in simulation and on the robot by using the OptiTrack system to provide the robot state. From the perspective of the robot's believe state of its position, tracking performance is comparable to the one obtained with OptiTrack system. By looking at the ground-truth results obtained by the OptiTrack, however, the robot is not perfectly following the reference path (red line). This is mainly due to the uncertainty on the on-board localization data.

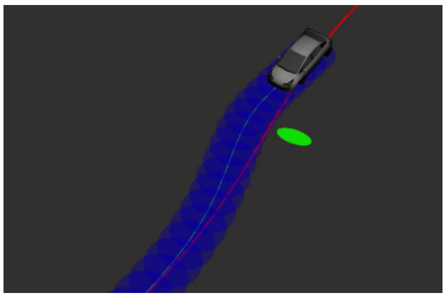
In Fig. 6, we cumulate the clearance distance between the robot and the obstacles for a characteristic time period of the second experiment. In almost all instances a minimum safe



a) Robot scenario 1



b) Robot scenario 2



c) Autonomous vehicle scenario

Fig. 4: Three characteristic collision avoidance scenarios

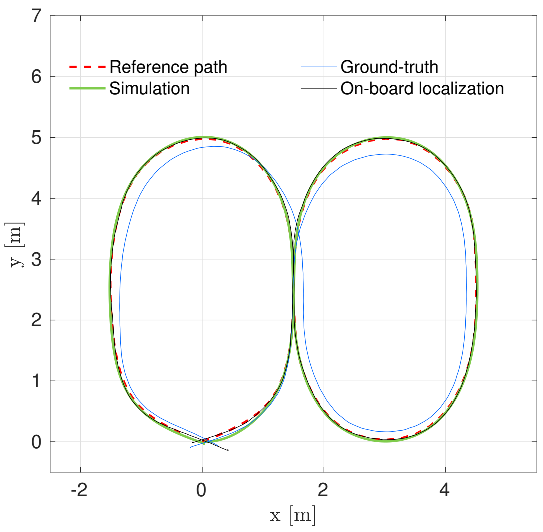


Fig. 5: Traveled path of the AGV compared to the reference.

distance of 0.32 m was achieved, which corresponds to the robot radius, using a single disc. One particular avoidance manoeuvre resulted in an overlap between the collision radius of the robot and the static obstacle, which is most likely due to a localization error. The few dynamic collision detected were situations where the pedestrians violated the collision boundary of the robot and the robot could not avoid fast enough. Fig. 4.a)-b) shows two snapshots of two critical driving situations avoiding moving and static obstacles.

C. Autonomous vehicle

We tested our algorithm in simulation results for an autonomous vehicle, a Toyota Prius. The motion model used in the LMPCC to control the vehicle is the one presented in [29], that is, a kinematic bicycle model. We simulate the vehicle following the same reference path used for the mobile robot scaled by a factor of 10. In addition, a dynamic obstacle moving at 0.5 m/s was simulated to cross the vehicle path, as presented in Fig. 4.c). The vehicle boundaries were defined using three discs in accordance to Section III-E. The velocity reference was set to 8 m/s ($\approx$ 30 Km/h). The autonomous vehicle successfully avoids collision with the obstacle and stably converges to the reference trajectory.

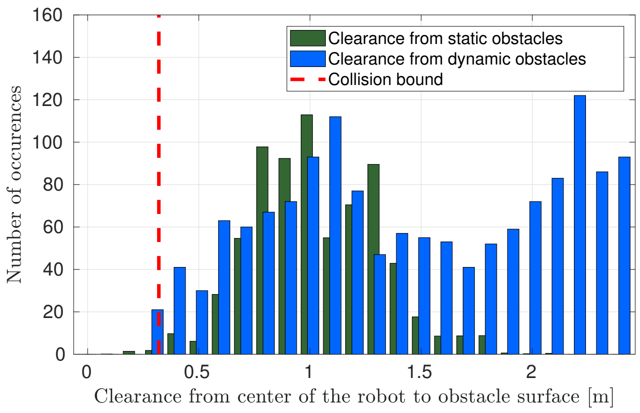


Fig. 6: Sampled clearance from the robot to static and dynamic obstacles

V. CONCLUSIONS

We proposed a local planning approach based on Local Model Predictive Contouring Control (LMPCC) to safely navigate AGVs in dynamic, unstructured environments. The LMPCC relies on a robust bound on the Minkowski sum to safely avoid dynamic obstacles. Furthermore, our design relies on a technique to compute a collision-free area to avoid static obstacles. We showed the applicability of our LMPCC design in simulations for a mobile robot and an autonomous vehicle. Finally, we performed real experiments using a mobile robot avoiding pedestrians and static obstacles. We showed that our motion planner satisfies the real-time constraint. Furthermore, the light implementation of our LMPCC allowed us to run all the algorithms on-board of the mobile robot. The next step is to test the proposed algorithm in a real autonomous car and to expand the current algorithm to incorporate prediction information about others road users.

REFERENCES

[1] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving

urban vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, March 2016.

[2] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and decision-making for autonomous vehicles," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 187–210, 2018. [Online]. Available: https://doi.org/10.1146/annurev-control-060117-105157

[3] S. M. La Valle, "Motion planning," *IEEE Robotics & Automation Magazine*, vol. 18, no. 2, pp. 108–118, 2011.

[4] D. Fridovich-Keil, S. L. Herbert, J. F. Fisac, S. Deglurkar, and C. J. Tomlin, "Planning, fast and slow: A framework for adaptive real-time safe trajectory planning," *arXiv preprint arXiv:1710.04731*, 2017.

[5] S. L. Herbert, M. Chen, S. Han, S. Bansal, J. F. Fisac, and C. J. Tomlin, "Fastrack: a modular framework for fast and guaranteed safe motion planning," in *56th IEEE Conference on Decision and Control (CDC)*. IEEE, 2017, pp. 1517–1522.

[6] J. Ziegler and C. Stiller, "Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2009, pp. 1879–1884.

[7] J. M. Maciejowski, *Predictive control: with constraints*. Pearson education, 2002.

[8] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[9] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, April 2016.

[10] A. Carvalho, S. Lefévre, G. Schildbach, J. Kong, and F. Borrelli, "Automated driving: The role of forecasts and uncertainty–a control perspective," *European Journal of Control*, vol. 24, pp. 14–32, 2015.

[11] N. Murgovski and J. Sjöberg, "Predictive cruise control with autonomous overtaking," in *54th IEEE Conference on Decision and Control (CDC),*. IEEE, 2015, pp. 644–649.

[12] M. Jalali, S. Khosravani, A. Khajepour, S.-k. Chen, and B. Litkouhi, "Model predictive control of vehicle stability using coordinated active steering and differential brakes," *Mechatronics*, vol. 48, pp. 30–41, 2017.

[13] C. M. Kang, S. Lee, and C. C. Chung, "On-road path generation and control for waypoints tracking," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 3, pp. 36–45, 2017.

[14] D. Lam, C. Manzie, and M. Good, "Model predictive contouring control," in *49th IEEE Conference on Decision and Control (CDC)*, Dec 2010, pp. 6137–6142.

[15] W. Schwarting, J. Alonso-Mora, L. Paull, S. Karaman, and D. Rus, "Safe nonlinear trajectory generation for parallel autonomy with a dynamic vehicle model," *IEEE Transactions on Intelligent Transportation Systems*, no. 99, pp. 1–15, 2017.

[16] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1: 43 scale rc cars," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 628–647, 2015.

[17] B. Houska, H. Ferreau, and M. Diehl, "ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization," *Optimal Control Applications and Methods*, vol. 32, no. 3, pp. 298–312, 2011.

[18] J. J. Kuffner and S. M. La Valle, "Rrt-connect: An efficient approach to single-query path planning," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2. IEEE, 2000, pp. 995–1001.

[19] T. Kluge, "c++ cubic spline library," 2014, accessed Jul. 5, 2018. [Online]. Available: http://kluge.in-chemnitz.de/opensource/spline/

[20] Y. Yan and G. S. Chirikjian, "Closed-form characterization of the minkowski sum and difference of two ellipsoids," *Geometriae Dedicata*, vol. 177, no. 1, pp. 103–128, 2015.

[21] M. A. Abbas, R. Milman, and J. M. Eklund, "Obstacle avoidance in real time with nonlinear model predictive control of autonomous vehicles," *Canadian Journal of Electrical and Computer Engineering*, vol. 40, no. 1, pp. 12–22, winter 2017.

[22] H. W. Kuhn and A. W. Tucker, "Nonlinear programming," in *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*. Berkeley, Calif.: University of California Press, 1951, pp. 481–492. [Online]. Available: https://projecteuclid.org/euclid.bsmsp/1200500249

[23] Jackal small unmanned ground vehicle, Clearpath^{TM} Robotics. [Online]. Available: https://www.clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/

[24] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, Sept 2004, pp. 2149–2154 vol.3.

[25] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, December 2012, http://ompl.kavrakilab.org.

[26] E. Bertolazzi and M. Frego, "G1 fitting with clothoids," *Mathematical Methods in the Applied Sciences*, vol. 38, no. 5, pp. 881–897, 2015.

[27] N. Point, "Optitrack," *Natural Point, Inc*, 2011.

[28] R. Siegwart and I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*, 2nd ed. The MIT Press, 2011.

[29] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design." in *Intelligent Vehicles Symposium*, 2015, pp. 1094–1099.

Bibliography

[1] GAO-16-350, “Vehicle Cybersecurity: DOT and Industry Have Efforts Under Way, but DOT Needs to Define Its Role in Responding to a Real-world Attack,” *General Accounting Office*, Apr 2016.

[2] K. Kozlowzki and D. Pazderski, “Modeling and control of a 4-wheel skid-steering mobile robot,” *International Journal of Applied Mathematics and Computer Science*, vol. 14, no. 4, pp. 477–496, 2004.

[3] D. Ariens, M. Diehl, H. J. Ferreau, B. Houska, F. Logist, R. Quirynen, and M. Vukov, “Acado toolkit user’s manual.” http://www.acadotoolkit.org, 2014.

[4] M. Daily, S. Medasani, R. Behringer, and M. Trivedi, “Self-driving cars,” *Computer*, vol. 50, pp. 18–23, December 2017.

[5] S. Singh, “Critical reasons for crashes investigated in the national motor vehicle crash causation survey.” National Highway Traffic Safety Administration, Washington, D.C., USA, Rep. DOT HS 812 115, 2014.

[6] J. Greenblatt and S. Saxena, “Autonomous taxis could greatly reduce greenhouse-gas emissions of us light-duty vehicles,” *Nature Climate Change*, vol. 5, p. 860âĂŞ863, 2015.

[7] H. Igliński and M. Babiak, “Analysis of the potential of autonomous vehicles in reducing the emissions of greenhouse gases in road transport,” *Procedia Engineering*, vol. 192, pp. 353 – 358, 2017. 12th international scientific conference of young scientists on sustainable, modern and safe transport.

[8] D. J. Fagnant and K. M. Kockelman, “The travel and environmental implications of shared autonomous vehicles, using agent-based model scenarios,” *Transportation Research Part C: Emerging Technologies*, vol. 40, pp. 1 – 13, 2014.

[9] P. Jittrapirom, V. Caiati, A.-M. Feneri, S. Ebrahimigharehbaghi, M. GonzÃąlez, and J. Narayan, “Mobility as a service: A critical review of definitions, assessments of schemes, and key challenges,” *Urban Planning*, vol. 2, no. 2, 2017.

[10] T. Luettel, M. Himmelsbach, and H. J. Wuensche, “Autonomous Ground Vehicles - Concepts and a Path to the Future,” *Proceedings of the IEEE*, vol. 100, pp. 1831–1839, May 2012.

[11] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, “A survey of motion planning and control techniques for self-driving urban vehicles,” *IEEE Transactions on Intelligent Vehicles*, vol. 1, pp. 33–55, March 2016.

[12] J. Yang and J. F. Coughlin, “In-vehicle technology for self-driving cars: Advantages and challenges for aging drivers,” *International Journal of Automotive Technology*, vol. 15, pp. 333–340, Mar 2014.

[13] C. D. Harper, C. T. Hendrickson, S. Mangones, and C. Samaras, “Estimating potential increases in travel with autonomous vehicles for the non-driving, elderly and people with travel-restrictive medical conditions,” *Transportation Research Part C: Emerging Technologies*, vol. 72, pp. 1 – 9, 2016.

[14] W. Schwarting, J. Alonso-Mora, and D. Rus, “Planning and decision-making for autonomous vehicles,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 187–210, 2018.

[15] D. J. Fagnant and K. Kockelman, “Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations,” *Transportation Research Part A: Policy and Practice*, vol. 77, pp. 167 – 181, 2015.

[16] H. Abraham, C. Lee, S. Brady, C. Fitzgerald, B. Mehler, B. Reimer, and J. Coughlin, “Autonomous vehicles and alternatives to driving: Trust, preferences, and effects of age,” *Transportation Research Board 96th Annual Meeting*, 2017.

[17] P. Bansal and K. M. Kockelman, “Are we ready to embrace connected and self-driving vehicles? a case study of texans,” *Transportation*, vol. 45, pp. 641–675, Mar 2018.

[18] J. Claybrook and S. Kildare, “Autonomous vehicles: No driver. . . no regulation?,” *Science*, vol. 361, no. 6397, pp. 36–37, 2018.

[19] T. Litman, “Autonomous Vehicle Implementation Predictions: Implications for Transport Planning,” *Victoria Transport Policy Institute*, July 2018.

[20] A. Millard-Ball, “Pedestrians, autonomous vehicles, and cities,” *Journal of Planning Education and Research*, vol. 38, no. 1, pp. 6–12, 2018.

[21] S. Pettigrew, L. Fritschi, and R. Norman, “The Potential Implications of Autonomous Vehicles in and around the Workplace,” *Int J Environ Res Public Health*, vol. 15, Aug 30 2018.

[22] R. Siegwart and I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*. The MIT Press, 2004.

[23] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, “Path Planning for Autonomous Vehicles in Unknown Semi-structured Environments,” *The International Journal of Robotics Research*, vol. 29, no. 5, pp. 485–501, 2010.

[24] N. E. D. Toit and J. W. Burdick, "Robot motion planning in dynamic, uncertain environments," *IEEE Transactions on Robotics*, vol. 28, pp. 101–115, Feb 2012.

[25] T. Braunl, "Research relevance of mobile robot competitions," *IEEE Robotics Automation Magazine*, vol. 6, pp. 32–37, Dec 1999.

[26] G. Virk, "Industrial mobile robots: the future," *Industrial Robot: An International Journal*, vol. 24, no. 2, pp. 102–105, 1997.

[27] Mordor Intelligence, "Service Robotics Market Size - Segmented by Type (Personal Robots, Professional Robots), Areas (Aerial, Land, Underwater), Components (Sensors, Actuators, Control Systems, Software), Verticals (Healthcare, Logistics & Transportation, Agriculture & Mining, Government, Military & Defense), and Region - Growth, Trends, and Forecast (2018 - 2023)," March 2018.

[28] M. Hoy, A. S. Matveev, and A. V. Savkin, "Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey," *Robotica*, vol. 33, no. 3, pp. 463–497, 2015.

[29] R. Benenson, S. Petti, T. Fraichard, and M. Parent, "Integrating Perception and Planning for Autonomous Navigation of Urban Vehicles," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 98–104, Oct 2006.

[30] D. Lenz, M. Rickert, and A. Knoll, "Heuristic search in belief space for motion planning under uncertainties," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2659–2665, Sept 2015.

[31] C. Laugier, S. Petti, D. Vasquez, M. Yguel, T. Fraichard, and O. Aycard, *Steps Towards Safe Navigation in Open and Dynamic Environments*, pp. 45–82. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[32] S. M. LaValle, *Planning algorithms.* Cambridge University Press, 2006.

[33] E. Masehian and D. Sedighizadeh, "Classic and Heuristic Approaches in Robot Motion Planning A Chronological Review," 2007.

[34] J.-C. Latombe, "Motion planning: A journey of robots, molecules, digital actors, and other artifacts," *The International Journal of Robotics Research*, vol. 18, no. 11, pp. 1119–1128, 1999.

[35] N. Sariff and N. Buniyamin, "An Overview of Autonomous Mobile Robot Path Planning Algorithms," in *2006 4th Student Conference on Research and Development*, pp. 183–188, June 2006.

[36] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *IEEE Access*, vol. 2, pp. 56–77, 2014.

[37] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially Aware Motion Planning with Deep Reinforcement Learning," *CoRR*, vol. abs/1703.08862, 2017.

[38] C. Laugier and R. Chatila, *Autonomous Navigation in Dynamic Environments.* No. 35 in Springer Tracts in Advanced Robotics, Springer, 2017.

[39] D. Ferguson, M. Likhachev, and A. T. Stentz, "A guide to heuristic-based path planning," in *Proceedings of the International Workshop on Planning under Uncertainty for Autonomous Systems, International Conference on Automated Planning and Scheduling (ICAPS)*, (Pittsburgh, PA), June 2005.

[40] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun, "Anytime Dynamic A*: An Anytime, Replanning Algorithm," 1995.

[41] O. Adiyatov and H. A. Varol, "A novel RRT*-based algorithm for motion planning in dynamic environments," in *2017 IEEE International Conference on Mechatronics and Automation (ICMA)*, pp. 1416–1421, Aug 2017.

[42] L. Jaillet and T. Simeon, "A PRM-based motion planner for dynamically changing environments," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 2, pp. 1606–1611 vol.2, Sept 2004.

[43] M. Pivtoraiko and A. Kelly, "Kinodynamic motion planning with state lattice motion primitives," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2172–2179, Sept 2011.

[44] Y. Wang, J. Wang, and S. Yin, "An Object-Based Path Planning Using Grids-Potential Fields for Intelligent Robot," in *2009 Third International Conference on Genetic and Evolutionary Computing*, pp. 150–153, Oct 2009.

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

[46] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, pp. 1135–1145, April 2016.

[47] A. Carvalho, S. Lefévre, G. Schildbach, J. Kong, and F. Borrelli, "Automated driving: The role of forecasts and uncertainty–a control perspective," *European Journal of Control*, vol. 24, pp. 14–32, 2015.

[48] J. M. Maciejowski, *Predictive control: with constraints.* Pearson education, 2002.

[49] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems.* Cambridge University Press, 2017.

[50] D. Lam, C. Manzie, and M. Good, "Model predictive contouring control," in *49th IEEE Conference on Decision and Control (CDC)*, pp. 6137–6142, Dec 2010.

[51] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3 – 20, 2002.

[52] Y. Ding, Z. Xu, J. Zhao, K. Wang, and Z. Shao, "Embedded MPC Controller Based on Interior-Point Method with Convergence Depth Control," *Asian Journal of Control*, vol. 18, no. 6, pp. 2064–2077, 2016. RR-15-0136.R2.

[53] H. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.

[54] H. J. Ferreau, H. G. Bock, and M. Diehl, "An online active set strategy to overcome the limitations of explicit mpc," *International Journal of Robust and Nonlinear Control*, vol. 18, no. 8, pp. 816–830, 2008.

[55] T. Kluge, "c++ cubic spline library," 2014. Accessed Jul. 5, 2018.

[56] H. W. Kuhn and A. W. Tucker, "Nonlinear programming," in *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, (Berkeley, Calif.), pp. 481–492, University of California Press, 1951.

[57] B. Shamah, "Experimental comparison of skid steering vs. explicit steering for a wheeled mobile robot," Master's thesis, Robotics Institute , Carnegie Mellon University, Pittsburgh, PA, March 1999.

[58] J. Y. Wong, *Theory of Ground Vehicles.* John Wiley & Sons, Inc., 4th ed., 2008.

[59] N. Point, "Optitrack," *Natural Point, Inc*, 2011.

[60] W. Schwarting, J. Alonso-Mora, L. Paull, S. Karaman, and D. Rus, "Safe nonlinear trajectory generation for parallel autonomy with a dynamic vehicle model," *IEEE Transactions on Intelligent Transportation Systems*, no. 99, pp. 1–15, 2017.

Glossary

List of Acronyms

AGV	Autonomous Ground Vehicle
COM	Center Of Mass
CoR	Cognitive Robotics
EKF	Extended Kalman Filter
GPS	Global Positioning System
HMI	Human Machine Interface
ICRA	International Conference on Robotics and Automation
ICR	Instantaneous Center of Rotation
IMU	Inertial Measurement Unit
KKT	Karush-Kuhn-Tucker
LiDAR	Light Detection And Ranging
LMPCC	Local Model Predictive Contouring Control
MCU	Micro Controller Unit
MPC	Model Predictive Control
MPCC	Model Predictive Contouring Control
NMPC	Non-linear Model Predictive Control
OCP	Optimal Control Problem
ODV	Online Data Variable
OMPL	Open Motion Planning Library

PWM	Pulse-Width Modulation
QP	Quadratic Programming
RRT	Rapidly-exploring Random Tree
ROS	Robotic Operating System