Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Control Engineering

Localization and advanced control for autonomous model cars

Master's thesis

Bc. David Kopecký

Master programme: Cybernetics and Robotics
Supervisor: Ing. Michal Sojka, Ph.D.

Prague, May 2019

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Kopecký David**	Personal ID number:	**434676**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Control Engineering**		
Study program:	**Cybernetics and Robotics**		
Branch of study:	**Cybernetics and Robotics**		

II. Master’s thesis details

Master’s thesis title in English:

Localization and advanced control for autonomous model cars

Master’s thesis title in Czech:

Lokalizace a pokročilé řízení autonomního modelu vozidla

Guidelines:

1. Make yourself with F1/10 competition, ROS project and software that was used in previous competitions.
2. Review possibilities of using advanced methods for vehicle time-optimal control.
3. Implement an algorithm for vehicle localization on the track. Reliable solution will probably need to fuse results of multiple localization methods.
4. Implement an algorithm for vehicle time-optimal control on the race track using the developed localization algorithm.
5. Properly test and document the results.

Bibliography / sources:

[1] F1/10 – The Rules version 1.0, http://f1tenth.org/misc-docs/rules.pdf
[2] Jan Filip, Sledování trajektorie pro autonomní vozidla, diplomová práce ČVUT, 2018

Name and workplace of master’s thesis supervisor:

Ing. Michal Sojka, Ph.D., Embedded Systems, CIIRC

Name and workplace of second master’s thesis supervisor or consultant:

Date of master’s thesis assignment: **01.02.2019** Deadline for master's thesis submission: **24.05.2019**

Assignment valid until:
by the end of summer semester 2019/2020

Ing. Michal Sojka, Ph.D.	prof. Ing. Michael Šebek, DrSc.	prof. Ing. Pavel Ripka, CSc.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the master’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master’s thesis, the author must state the names of consultants and include a list of references.

.

Date of assignment receipt Student’s signature

Declaration

I hereby declare I have written this master's thesis independently and quoted all the sources of information used in accordance with methodological instructions on ethical principles for writing an academic thesis. Moreover, I state that this thesis has neither been submitted nor accepted for any other degree.

In Prague, May 2019

....
Bc. David Kopecký

Abstract

The goal of this thesis is to develop a solution for F1/10 autonomous driving competition. First part of the work deals with mapping and vehicle localization on the racing track with a down-scaled model car. Introduced localization use the Monte Carlo methods to process the data from LiDAR and estimate the position of the vehicle. The implemented system is able to precisely estimate the vehicle position with the rate of 25 Hz. The second part of the thesis deals with the design of the trajectory tracking control system. The presented solution uses the LQR and Model predictive control to achieve good performance with knowledge of vehicle kinematics.

Keywords: F1/10 competition, Autonomous racing, Monte Carlo Localization, Hector SLAM, trajectory tracking, Model Predictive Control

Abstrakt

Tato práce se zabývá mapováním a lokalizací na závodní dráze F1/10 autonomous driving competition pomocí zmenšeného modelu vozidla. Lokalizace využívá Monte Carlo metod pro zpracování dat LiDARu a odhad pozice vozidla. Implementovaný systém dokáže kvalitně odhadovat pozici s frekvencí 25 Hz. Druhou částí práce je návrh řídicího systému sledování trajektorie. Navržený systém využívá pokročilých metod řízení LQR, Model Predictive Control a uvažuje kinematický model řízeného vozidla.

Klíčová slova: F1/10 competition, Autonomní řízení, Monte Carlo lokalizace, Hector SLAM, Sledování trajektorie, Model Predictive Control

Acknowledgements

First, I would like to thank Michal Sojka for supervising this work and for all helpful advice he gave me during my internship in Industrial Informatics Department.

Second, I would like to thank my friend and colleague Jaroslav Klapálek for all consultations and support he gave me during difficult experiments of this work.

Finally, I must express my very profound gratitude to my parents for providing me with unfailing support and continuous encouragement throughout my years of study. This accomplishment would not have been possible without them.

List of Figures

Contents

Chapter 1

Introduction

The goal of this thesis is to develop a control system able to race with a vehicle model scaled by 1/10 on the racing track with the utilization of high-level planning on the created map. The task is divided into three major parts. The objective of the first part is to develop a mapping system, which use the sensor data to explore the unknown environment and create a map of the track. The second part deals with the vehicle localization on the track without an absolute position sensor such as GPS or any indoor localization. The third part then aims to a lateral control system which drives the vehicle over the racing track. The goal is to use advanced control methods considering the vehicle kinematics to drive the vehicle with the best performance.

The thesis follows the previous work of Martin Vajnar [1] which focus on building of the racing platform and processing of the sensor data. The design of control system is then following the work of Jan Filip [2] which deals with the task of trajectory tracking formalized as a servomechanism problem and come up with solution tested on simulations.

1.1 Motivation

The motivation of this work is to create a solution for F1/10 autonomous driving competition, which will be used in upcoming race round. In previous rounds, most of the solutions were reactive algorithms, which did not consider the track layout or vehicle kinematics structure. Even though the presented solutions were functional and shown a good performance, it turned out, that reactive control approach is not able to handle all situations efficiently. Because of that, the map-based approach is introduced, which is able to localize the vehicle on the track and gives new options for vehicle control by high-level planning.

The second motivation of this work is, that developed localization of car models creates good testing conditions for the development of applications related to autonomous driving.

Such applications have to be tested on various scenarios to prove robustness and reliability. Testing those scenarios on car models instead of real cars is then much easier and cheaper.

1.2 Work Outline

In Chapter 2, the thesis describes the motivation and rules of F1/10 autonomous driving competition, introduce the racing platform, and review the solutions presented in previous rounds of the race. In Chapter 3 the scan-matching problem is outlined as a way how to perform Simultaneous Localization and mapping (SLAM) and the Hector SLAM method will be described. In Chapter 4, the Monte Carlo Localization (MCL) will be used to achieve a vehicle localization on a 2D map with the data from LiDAR. This Chapter also compares several methods of ray-casting and introduce two extensions, which use the data from wheel odometry to improve MCL position estimation. Chapter 5 analyze the process of simple automatic trajectory planning on the racing track for testing purposes and discuss the problem of trajectory utilization for time optimal racing. Finally, the last Chapter 6 focus on advanced control of the vehicle introduced as trajectory tracking.

Chapter 2

Background

Several competitions in the field of autonomous driving have been announced in the recent past to challenge different types of tasks. The *DARPA grand challenge* in 2004 [3] and *DARPA urban challenge* in 2007 [4] were one of the first large-scale competitions which aimed to develop a driver-less vehicle able to move in different terrains and handle basic traffic rules. *The Audi autonomous driving cup* challenge participants to build fully automatic driving functions and the necessary software architectures on 1/8 scaled car models. *Roborace* deals with the task of autonomous driving car able to race manually driven vehicles on a racing track. This thesis focus on solution of F1/10 autonomous driving competition, which challenge to race with scaled vehicles by 1/10.

This Chapter gives the reader background to the competition rules and motivation in Section 2.1 and provides the reader basic overview of the racing task. Then, in Section 2.2, describes the structure and equipment of the racing platform used in this thesis with preview of abilities of its components. Section 2.3 summarizes the reactive with map-based control strategy and highlights their main advantages and disadvantages regards to already utilized solutions in previous rounds of F1/10 competition. Section 2.4 follows with review of control methods possibly used to perform vehicle steering task along the racing track.

2.1 F1/10 competition

The F1/10 is a worldwide competition of scaled autonomous cars announced by the University of Pennsylvania which deals with the task of developing a software able to race with a down-scaled vehicle model on the racing track. Racing with the scaled platforms in contrast to real cars makes development affordable, easy to test and gives an opportunity to small student teams to bring their ideas

Since the key phrase of the competition is “The battle of algorithms”, the task does not rely on building a vehicle itself but limits the teams with hardware requirements to provide similar racing conditions. The idea of having a platform with same abilities pushes participant to focus on the control structures with different kinds of approaches. Organizers also rely on providing all the functional solutions open-source to the community, thus the abilities of vehicles are getting better every round of the race. The task of racing and handling vehicle in high speeds constantly discovers new bottlenecks of algorithms and force participant to come up with more complex solutions.

2.2 Vehicle platform description

The F1/10 racing platform is originally build on a Traxxas RC rally car and customized with several components. The competition organizers provides detailed instructions of building procedure [5], same as the [1], which also focuses on processing the data from sensors.

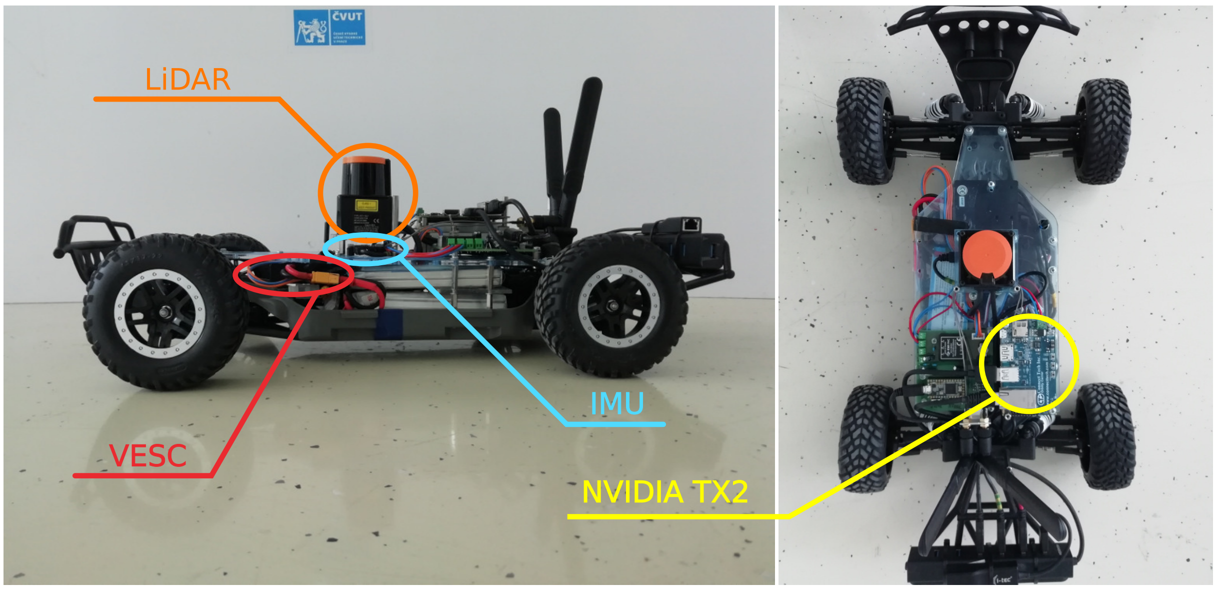
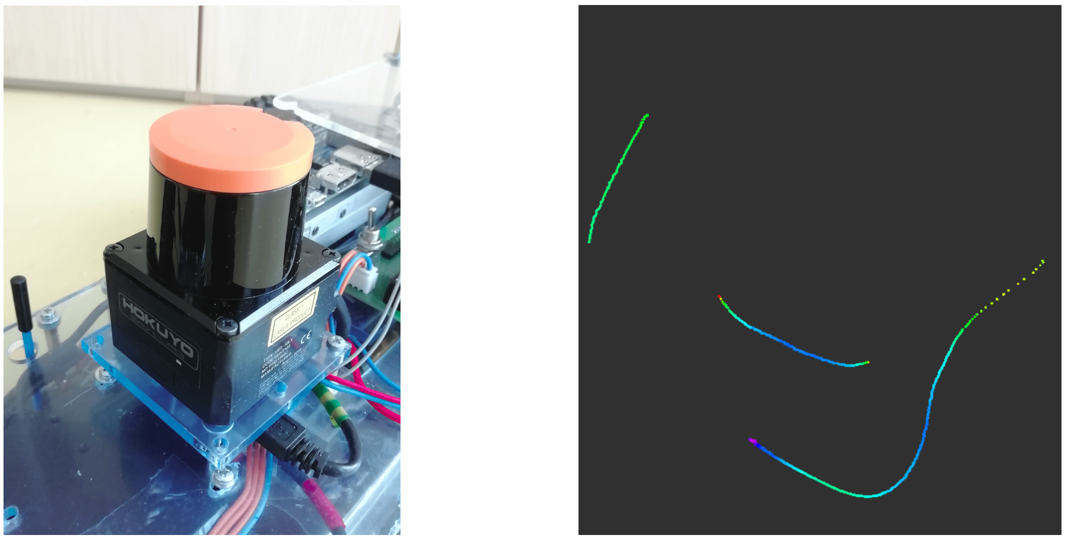


Figure 2.1: Racing platform

2.2.1 Sensors and perception

The racing car perceives the environment with several sensors. The most significant component is the LiDAR or optionally the stereo camera, which is able to measure distances from objects around the vehicle. Rules of the competition do not define the specific place where the LiDAR has to be mounted and even utilization of multiple LiDARs is allowed. However, for performing the task of localization and mapping the current configuration shown in 2.1 is sufficient.



(a) (b))

Figure 2.2: Utilized lidar Hokuyo UST-10LX and vizualized LiDAR scan

LiDAR measure the distance by illuminating the object with laser light pulse and calculating the time until the reflected beam is received again. This measurement is performed sequentially around the range of 270° with resolution of 0.25°. The resulting scan is the planar scene as shown in figure 2.2b. The LiDAR is mostly used for navigation.

Except for the LiDAR, the vehicle can use the inertial measurement unit (IMU), and the data from VESC control used to control brush-less DC motor. IMU provides the data of linear vehicle acceleration in three axes and angular velocity of a roll, pitch, and yaw motion which can be used to determine vehicle odometry. The IMU is commonly used only for measuring the actual angular velocity of yaw motion. The VESC, on the other hand, provide quite precise data about linear velocity of the vehicle and it is used for computation of vehicle odometry together with data from steering commands.

2.2.2 Computer unit

All the processes and calculations are performed on embedded system Nvidia Jetson TX2, mounted to the car on the Orbitty carrier board. The code servicing the peripherals and actuators is implemented in the Robotic Operating System (ROS) running in the Linux environment. The main advantage of Jetson Tx2 is GPU with 256 cores which allows some of time-consuming tasks, such as localization described in chapter 4, to be parallelized.

2.2.3 Platform kinematics

The used racing platform is a 4-wheel ackermann-type steering vehicle described in [6] or [7] . The geometry of this steering mechanism is outlined in Fig. 2.3

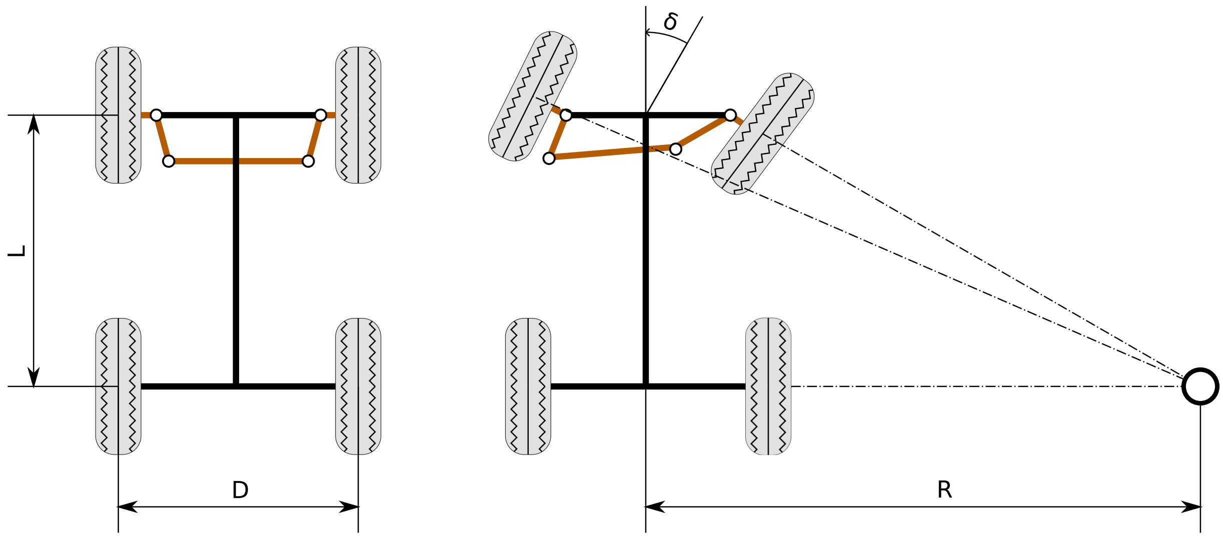


Figure 2.3: Ackermann geometry

the L denote the wheelbase of the vehicle, D is the wheel spacing, δ is the steering angle and R radius of turn. For a vehicle movement over the planar space we introduce the notation pictured in Fig. 2.4

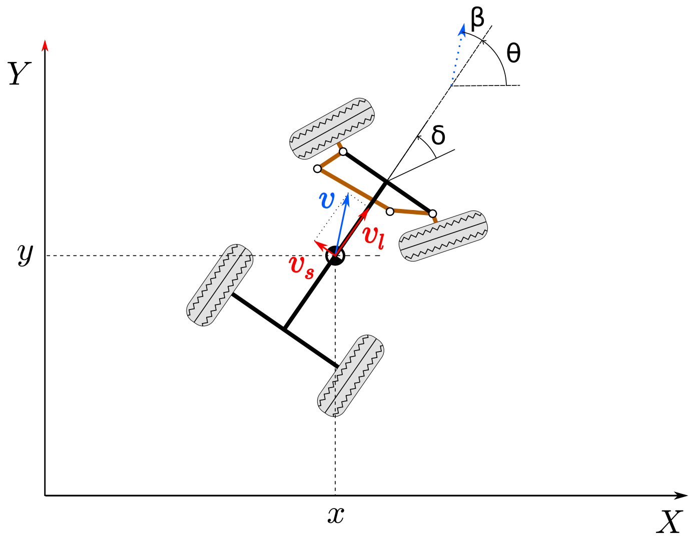


Figure 2.4: Vehicle kinematic model notation

x,y,θ denotes vehicle position and orientation on the planar world coordinates, v_l and v_s is vehicle longitudinal and lateral velocity, δ is the vehicle steering angle and β denotes vehicle slip angle.

The kinematic platform moves along the curves defined by curvature $\kappa(t)$, which could be expressed as radius inverse of circle tangent to curve as shown in Fig. 2.5.

$$k(t) = \frac{1}{R(t)} = \frac{\tan\delta(t)}{L} = \frac{d\theta}{ds} \tag{2.1}$$

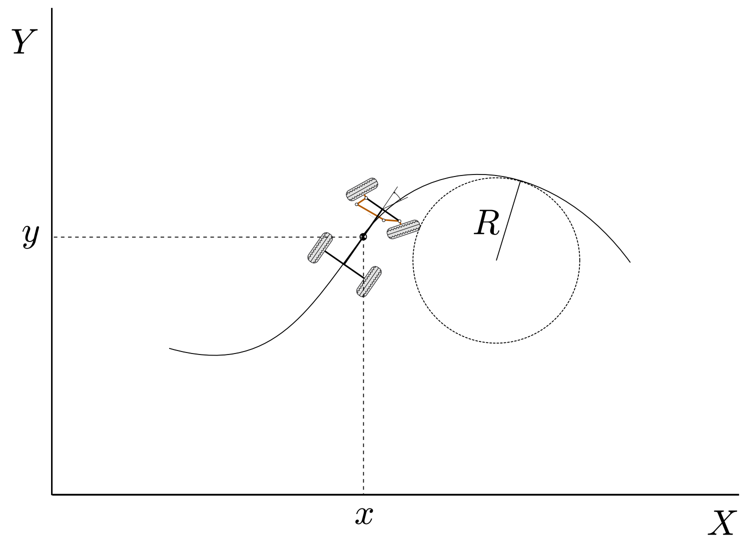


Figure 2.5: Curvature outline

The vehicle motion on the world coordinates then could be expressed as

$$\dot{x} = \frac{dx}{dt} = v(t)\cos\theta(t) \tag{2.2}$$

$$\dot{y} = \frac{dy}{dt} = v(t)\sin\theta(t) \tag{2.3}$$

$$\dot{\theta} = \frac{\theta}{dt} = v(t)k(t) = v(t)\frac{\tan\delta(t)}{L} \tag{2.4}$$

2.3 Control strategies

In previous rounds of the competition, many different control strategies have been introduced. These strategies can be generally divided into two categories – reactive and map-based strategies. Even though the map-based algorithms are assumed to have great potential, and their performance increases rapidly with every round of the competition, demonstrated solutions were not as efficient as reactive ones. The reason for this may be the fact, that map-based approach requires a much more complex solution compared to reactive approach as described further.

2.3.1 Reactive strategy

The reactive control algorithms select the outputs of the system as a response on the short-term sensor data without high level cognition aspects. In the case of racing, the vehicle navigation is performed based on current or last few LiDAR scans. From these scans the various types of errors are determined and penalized or the object avoidance task is carried out to accomplish the non-colliding driving trough the race track.

Straightforward principle of the reactive algorithms usually makes them easy to implement and tune. Nevertheless some implementations turned out very effective and robust in avoidance of different kind of static and also moving obstacles in higher speeds. The state of the art of the racing reactive algorithm is the **Follow The Gap** (FTG), which utilize the heuristics from the obstacle avoidance algorithm presented in [8].

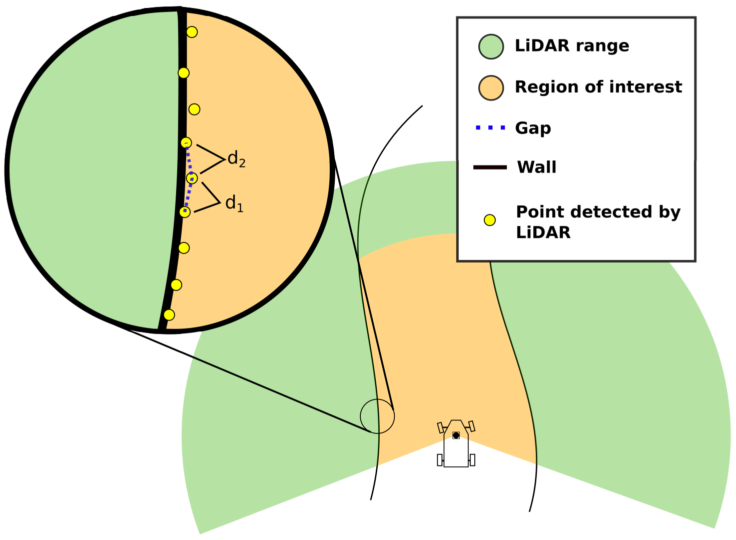


Figure 2.6: FTG Neighborhood gaps distance calculation

The FTG algorithm processes the current LiDAR scan and separates points which are out of the predefined range called Region Of Interest (ROI). Every point inside the ROI is considered as obstacle. FTG computes sequentially distances between neighbor obstacles from left to right as shown in Figure 2.6 and pick the two points with the largest distance. The center of the line connecting those points is considered as a goal point, and the steering angle of the car is set in the direction of this point. The essential function of the FTG is outlined in Fig.2.7.

FTG is usually adjusted in several ways, to gain a better performance on the racing track. For instance, the velocity of the car could be tuned by the value of steering angle, to go faster if the steering angle is close to heading angle, or the range of ROI could be adjusted to set aggressiveness of vehicle steering. The choice of tuning parameters always depends on the track layout and have to be precisely tuned before every race in trial laps.

Even tough the reactive algorithms could be very effective and fast on simple tracks, on the more complex tracks the sharp turns or dead ends could cause problems. Also the lack of information about track layout limits the vehicles to handle the most difficult part of the track. That could lead to non-efficient driving in long corridors or slow cornering in a simple turns. Because of that, the map-based approach is being introduced, thus the trajectory planning and higher level control could be performed.

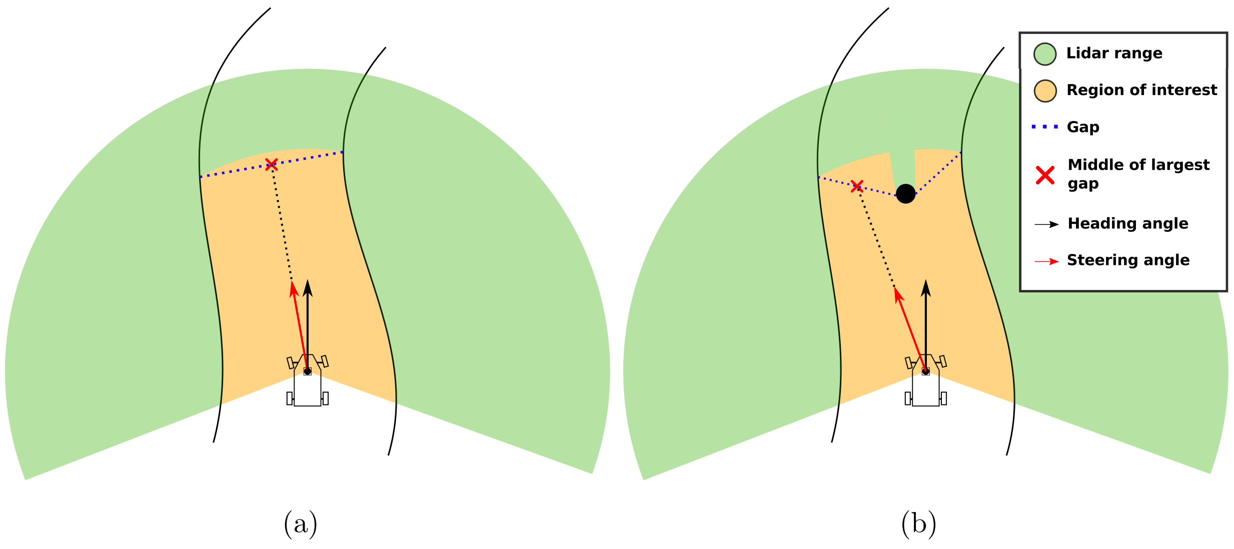


Figure 2.7: Example of FTG decision in corridor

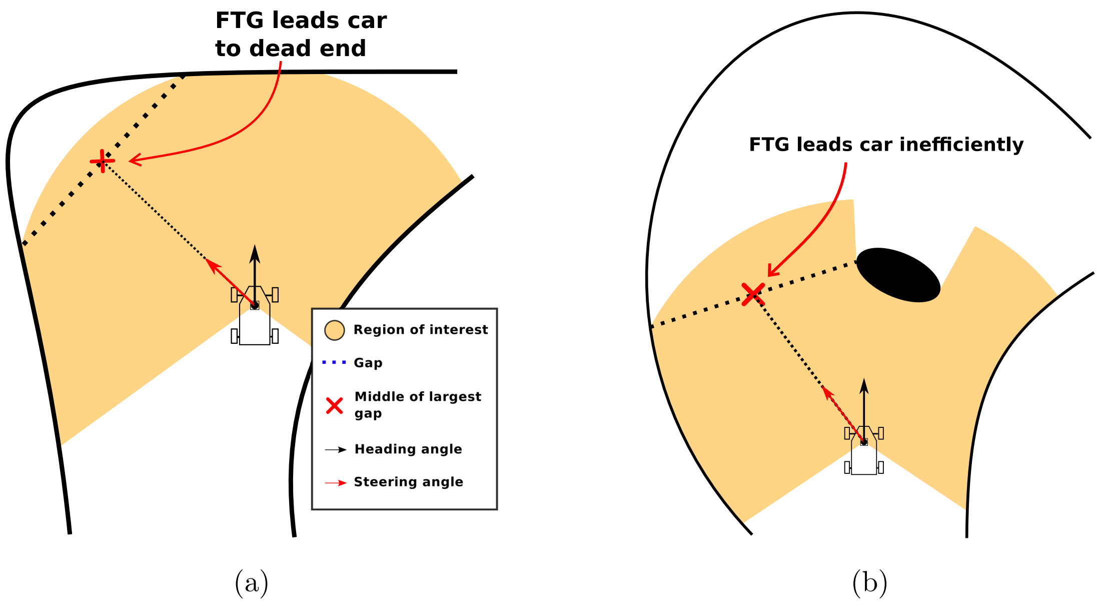


Figure 2.8: Example of problematic situations for reactive algorithms

2.3.2 Map-based strategy

Knowledge of the map of the track could bring a huge advantage to vehicle control as it could be used for higher level trajectory planing. Such planing can easily avoid situations where reactive algorithms fail (Examples shown in Fig. 2.8) and efficiently plan vehicle behaving in every part of the track. On the other hand the Map-based strategies are much more complex and need to carry out the task of mapping, localization, planing and trajectory tracking. Those task are the main focus of this thesis and will be described further.

2.4 Advanced control methods

All the previous racing solutions used in F1/10 competition were more or less functional but even the map-based solutions, when high-level trajectory planning task was involved, did not consider the kinematics or dynamics of the vehicle yet. The advanced control methods consider those aspects and try to utilize the vehicle abilities as much as possible to finish the lap in the fastest time.

To utilize the vehicle optimally the kinematic or dynamic model has to be introduced and the controller has to consider properties such as maximal acceleration and deceleration, speed limit, maximal steering angle or friction of the track. Finding those properties, which differ for every vehicle platform, is not a simple task and have to be established by several identification experiments. When the identified vehicle model is available, several control design methods could be used. In this thesis, we focus on designing the Linear-Quadratic Regulator (LQR) and Model Predictive Control (MPC) briefly introduced in next sections and explained in Chapter 6.

2.4.1 Linear-Quadratic Regulator (LQR)

LQR is a control strategy based on minimization of the defined quadratic cost function, which penalizes a final state of the system in the predicted horizon, the actual state of the system in every step and the controller input. The result of the design is the state feedback, which recalculates its penalizing constants every time step to ensure the optimal control action for preferred cost function and the system dynamics.

The adjustable cost function gives us an option to emphasize essential states of the systems and stress the control input to set the aggressivity of the system correctly. Nevertheless, the controller itself is not able to consider the discrete constraints of the system such as maximum vehicle steering angle. Hence we will investigate design of the Model Predictive Control.

2.4.2 Model Predictive Control (MPC)

The MPC is a control strategy, which solves the finite horizon open-loop optimal control problem. The main advantage of the MPC is that the design could consider the set of discrete constraints and include them into the optimization. Practically that means, that the controller is able to optimize the action on the predicted horizon with the knowledge of the vehicle limits.

The output of the MPC is the open-loop sequence of control inputs that minimizes the reference error of the system. The feedback is introduced by applying only the first input from the sequence and repeating the calculation in every step. Even though the MPC is the powerful state of the art control structure, the analytical solution of constrained MPC does not exist. Hence the optimization method called Quadratic-Programming have to be used which could be demanding for the vehicle computer unit. The detailed design of the MPC is explained further in this thesis in Chapter 6.

Chapter 3

Mapping

This chapter describes the method of mapping a track with the racing car using the data from the LiDAR scan and known scan-matching techniques of 2D-SLAM. Creating a map and cognition of the track environment is necessary for further higher-level planning task and map-based approach control.

The process of mapping is being developed to comply with the F1/10 competition rules. Those rules allow participants to make a manual or semiautomatic mapping lap, where the car is able to map the track. This mapping stage is performed directly before the race since the track could slightly change a layout due to crashes of other cars in previous rounds. Task of mapping the unknown environment with any other reference localization method is called Simultaneous Localization and Mapping (SLAM). For the racing task, only the planar layout is needed. This exploration of the environment represented by the planar map is often referred to as 2D-SLAM.

In the first part of this Chapter, the work focuses on 2D-SLAM problem formulation and scan-matching methods. Then Section 3.3 introduces the Hector slam method and describes its properties and features. The third part of this chapter focuses on mapping process tuning, and in the last section, the autonomous mapping process and its benefits are discussed.

3.1 2D-SLAM problem formulation

The Simultaneous localization and mapping (SLAM) is the difficult task in the area of mobile robotics which tries to handle environment exploration with the robot without any prior information about examined area or position. To be able to create a map, the robot has to be equipped with a proper sensor such as a LiDAR or stereo camera. Such sensors are able to approximate the layout or shapes of the surrounding environment in sufficient range as is reviewed in [9]. Those scans are processed sequentially and based on the changes in the scans robot tries to determine his relative movement (translation and rotation). This could be approached in two different ways. The first approach is to find special features in the scans such as sharp corners or specially shaped objects and determined the robot movement by position changes of these features. This approach is called feature-based SLAM. The second approach tries to find a transformation between following scans, which successfully fit them on each other, this method is called scan-matching, and since the track has no specific features and only planar scans from LiDAR are used, it's much more suitable for the task of this thesis.

3.2 Scan-matching problem

The task of the 2D scan-matching is to find a proper rigid transformation between following sensors scans to determine the robot relative motion, as shown in Fig. 3.1.

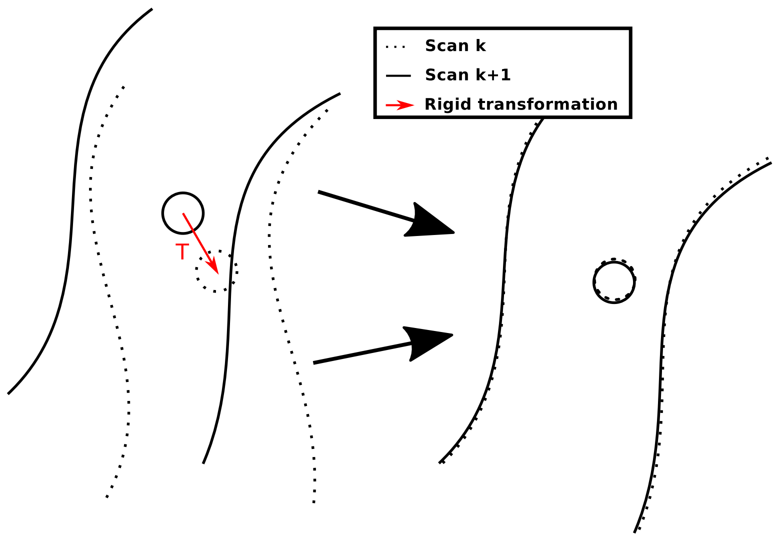


Figure 3.1: Scan matching transformation

The rigid transformation T consists from rotational matrix R and translation vector t, which map the same object from the scan in time $k+1$ to scan in time k with relation of the rigid transformation

$$x_k = Rx_{k+1} + t. \tag{3.1}$$

which in 2D representation can be rewritten to

$$\begin{bmatrix} x_k \\ y_k \end{bmatrix} = \begin{bmatrix} \cos\Phi & -\sin\Phi \\ \sin\Phi & \cos\Phi \end{bmatrix} \begin{bmatrix} x_{k+1} \\ y_{k+1} \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \tag{3.2}$$

3.2.1 Methods review

The several methods were introduced to perform scan-matching task with different heuristics. The basic method of Iterative Closest Point (ICP) introduced in [10] tries to fit scans with the usage of the nearest neighbor point heuristics. That results in an expensive search and possibility to stuck in a local minimum. The Polar Scan Matcher (PSM) [11] tries to utilize the natural polar coordinate system of LiDAR scanner, and even tough is faster than ICP, still is not efficient enough for real-time map construction. The method of a Flexible and scalable SLAM introduced by [12] formalize the scan-matching as the occupancy grid interpolation with the approximation of map gradients. This approach is suitable for sensors with high scan rates such as the Hokuyo LiDAR used by vehicle platform and its suitable for application of racing since it's usable without any other sensor data from IMU or good odometry.

3.2.2 Drawbacks of Scan-matching

Scan-matching methods are generally able to perform the SLAM but occasionally suffer in difficult situations. We can recognize those situations in two general cases. The first case shown in Fig. 3.2 captures the situation when in the following scan a substantial part of the new obstacle appears.

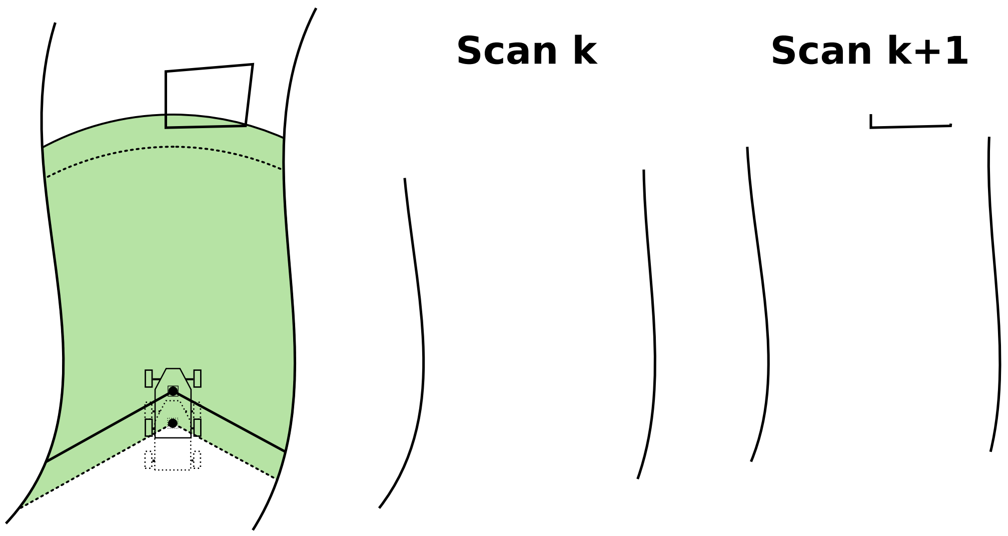


Figure 3.2: Problematic situation for scan-matching

This might lead methods with simple heuristics such as ICP to stuck in local minimum and wrong map construction. The second case is difficult even for more complex scan-matching methods and captures a moment when two following scans of moving robot are unrecognizable from each other. This situation is shown in Fig. 3.3.

In this situation, scan-matching results in zero transformation same as the robot would

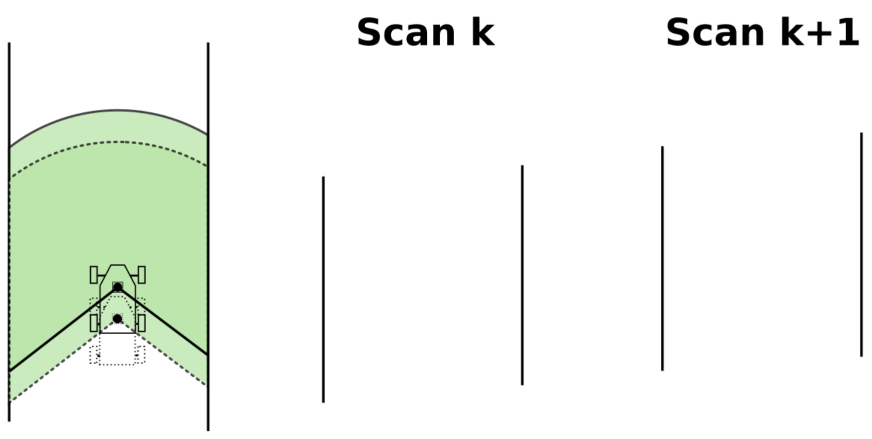


Figure 3.3: Straight corridor as a problematic situation for scan-matching

be stopped. To avoid this, the scan-matching have to be able to consider data from odometry or ensure, that sensors range is larger than the length of the longest possible corridor of explored environment.

3.3 Hector slam

Since the used LiDAR has sufficient range of $30m$, the used SLAM method should not suffer from the second problematic scenario of straight corridors. That also means, that no special integration of odometry to SLAM is needed.

Because of that, we decided to choose the Hector SLAM algorithm based on Flexible and Scalable SLAM [12], which is leveraged with a high scan rate of Hokuyo LiDAR and shown excellent results in hand-held mapping scenario. The Hector SLAM is also provided as an open-source ROS package. Thus it is documented and easy to integrate [13].

Hector slam perform mapping on the occupancy grid represented by resolution. LiDAR scans are firstly interpolated into this grid as shown in Figure 3.4 and then scan-matched to already created map. Since the Hector SLAM works with probabilities, every point of the occupancy grid represents the probability of present obstacle in the area which changes with every following scan. The final map is the result of probability thresholds which allows mapping to correctly reconstruct the map in the case when some slight layout changes or noise measurement occurs.

The final map is constructed from cells of occupancy grid, where each cell represents one of three mapping state.

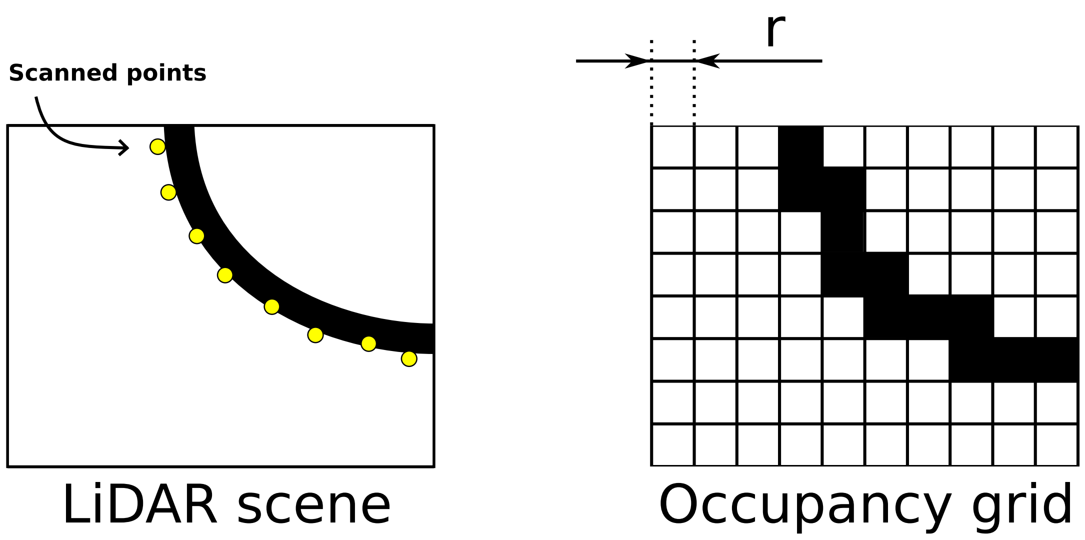


Figure 3.4: Occupancy grid interpolation (r - resolution of the occupancy grid)

3.3.1 Map resolution

Hector SLAM is adjustable with several parameters. The most important parameter is the resolution of the map, which affects either the mapping process and localization. Having a map with low resolution can lead to a bad approximation of the track environment. On the other hand, using a map with high resolution, could be computationally demanding.

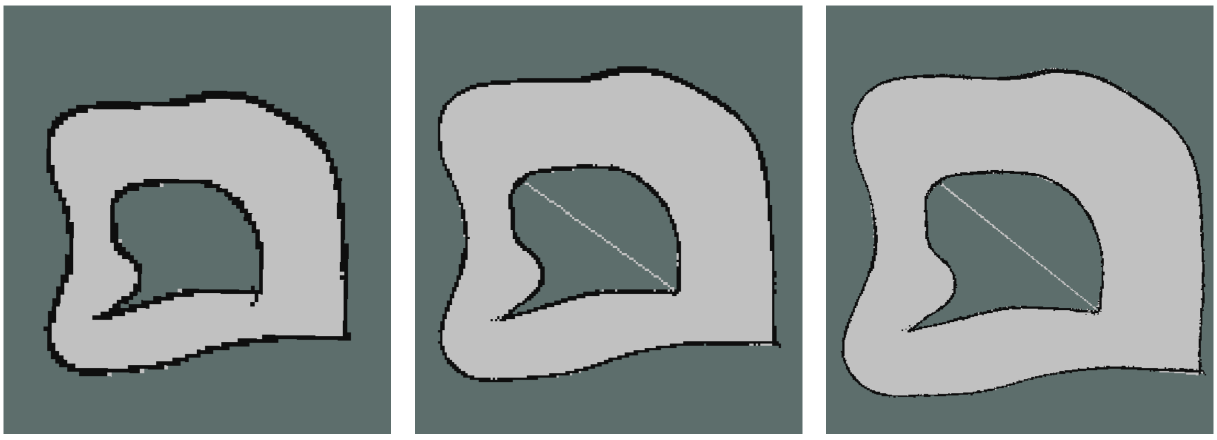


Figure 3.5: Maps of the track with different resolution (0.1m, 0.05m, 0.025m)

The resolution of the occupancy grid should always be selected based on the mapped environment. For racing track mapping, the resolution $0,05m$ is a good compromise.

3.3.2 Influence of the scan rate

As was mentioned at the beginning of this Section, the Hector SLAM method is utilizing the high scan rate of the sensors. The algorithm is minimizing the criterion function [12](Eq. 7), which is aligning the scan on the known map, considering slight movement between last known and current position. Result of this optimization is the rigid transformation described by Eq. 3.2. Making those moves finer between each scan is helping the algorithm to find an optimal solution. Hence with the higher rate of the scan sensor, we are able to create a map with a robot going at a higher speed.

To verify this, the experiment was conducted. The robot was set to move with slow constant speed around the track, and the data from sensors were recorded. After that, the mapping was performed several times on the data record, where the data from the scan were continuously down-sampled. Examples of results are shown in Fig. 3.6

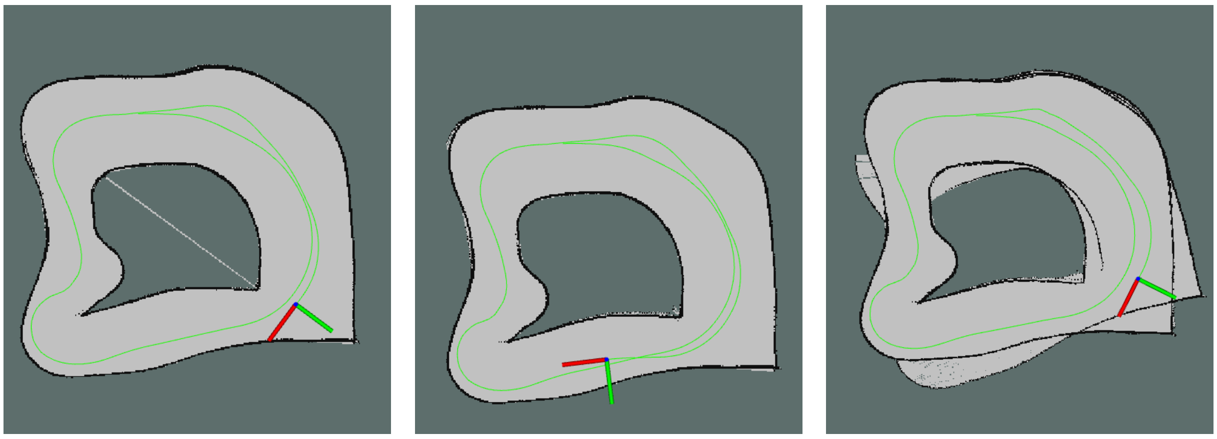


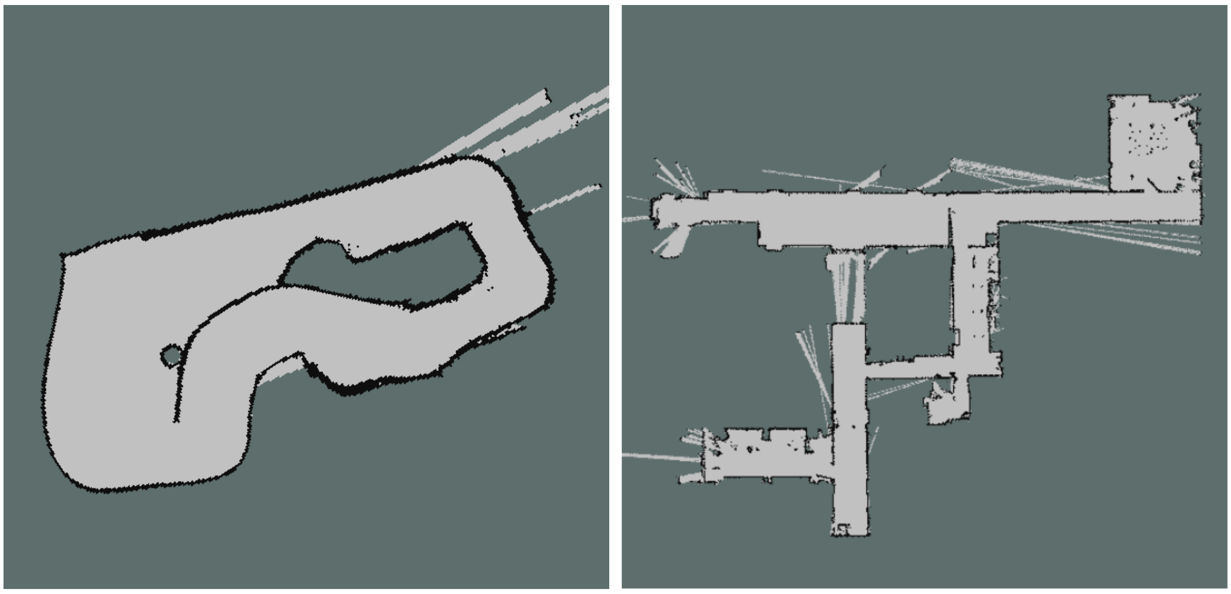
Figure 3.6: Mapping with scan different rate of LiDAR sensor (40Hz, 20Hz, 13Hz)

The result showed, that scanner at the frequency of 13Hz was not able to construct the map correctly, even if the speed of the car was very slow. Because of that, the choice of used LiDAR could be essential regarding fact, that some equally expensive LiDARs offers only 15Hz scan rate.

3.4 Mapping experiments

With the integrated Hector SLAM, several different environments were mapped to verify the function. Firstly, the map of the small track shown in Fig. 3.6 was made. The track is characterized with narrow rounded corridors, which could be problematic for the scan-matching. However, the mapping was successful. The second examined case was the large track with either narrow and spacious corridors. Since the mapping algorithm does not perform any backward corrections, mapping suffers to additive error. That could be crucial at the moment when the vehicle is about to finish the mapping lap, and the algorithm should connect the walls of the large loop. The result of the experiment is shown in Fig. 3.7(a).

The last experiment tries to map a more complex environment, and it was conducted in the office area. The result is shown in Fig. 3.7(b). It turned out, that office area with



(a) (b)

Figure 3.7: Constructed maps of different environment

a lot of straight walls and features was easy to map and even if the car moved with large accelerations the mapping algorithm was able to handle it.

3.5 Autonomous mapping

As was stated at the beginning of this Chapter, the rules of the competition allow mapping the track manually. However, the automatic mapping has important benefits. The mapping procedure could suffer when the car accelerates and tilt. The usage of a reactive algorithm in this stage with slow and constant velocity provides optimal conditions for mapping algorithm. If the map is during the mapping stage corrupted, the slower speed could be set, and mapping could be repeated until the map is constructed successfully.

Chapter 4

Localization

In this Chapter, the localization task in the known environment will be introduced. The sensor-based mobile robot localization or pose estimation is a challenging task, and it is recognized as a key problem in mobile robotics. Even though the plenty of approaches has been introduced either in 2D and 3D space, finding a robust solution for specific mobile robot or vehicle is never a simple task.

The localization of the vehicle is necessary for decision making and trajectory tracking. In this stage is assumed, that the racing track map is known and its layout has not changed significantly since the mapping stage. As the vehicle is not able to use GPS, Indoor localization or any other absolute position localization method, the localization has to be done primarily by LiDAR.

In this Chapter, the localization methods overview will be provided regarding to vehicle platform sensor equipment. Then in Section 4.2, the Monte Carlo Localization is described together with ray casting methods. Section 4.3 provide the wheel odometry calculation concerning vehicle Ackermann steering kinematics and in the last Section 4.4 approaches to pose filtering and estimation rate increasing are introduced.

4.1 Method overview

The several methods for indoor localization in a known environment have been introduced so far. Those methods are usually divided into groups of **Filtering techniques** and **Probabilistic techniques** as reviewed in [14]. The Filtering techniques such as [15] or [16] often assume usage of absolute position sensors or knowledge of information from which the absolute robot position could be estimated. In that case, the filtering is performed to process the measurement noise or to provide optimal position estimation from multiple data sources. In the second case, when the robot sensors are used to perform relative motion localization (sometimes stated as track keeping), the filtering tries to process the sensor data to minimize the uncertainty of robot relative motion. This localization is then provided by estimation with knowledge of those relative motion steps and the initial position of the robot. Since the racing car is not equipped with any absolute position sensor, the filtering methods could be used only to determine the relative movements of the vehicle. However, estimation of those movements always suffers from additive error and uncertainty of the estimated position grows with every next measurement. Hence, this type of localization could not be utilized for longer times and its not suitable for the racing task.

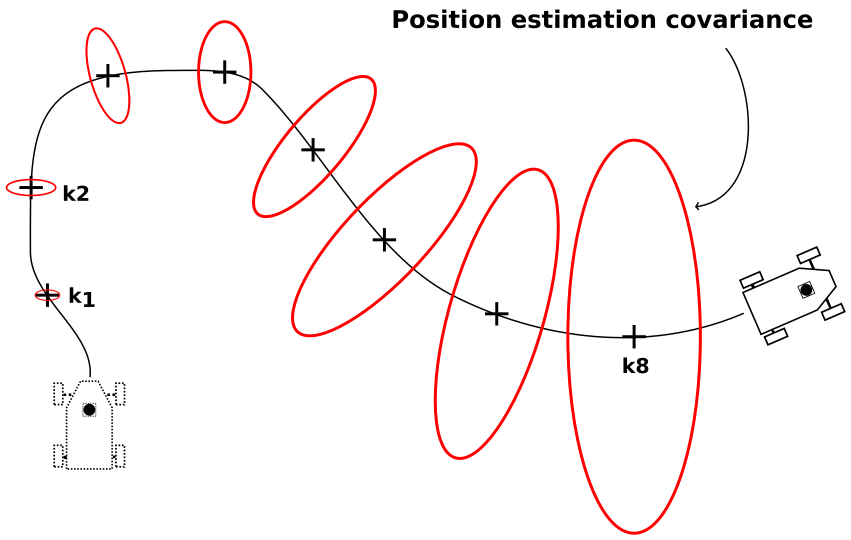


Figure 4.1: Growing covariance of global position estimation base on relative movements with additive error

The Probabilistic techniques are using sensors data to estimate the position on the map, by computation of the likelihood for measured data and randomly posed hypothesis (or particles) on the map. This technique refers to particle filter, which is together with the Markov localization the background of Monte Carlo Localization (MCL) methods generally introduced in [17]. The MCL method is able to localize the robot without prior knowledge of the initial position using the LiDAR. Hence, its suitable solution for localization of racing platform.

4.2 Monte Carlo Localization

Let us introduce the known map (the occupancy grid with a given resolution) as a set of states M, on which the robot position could be represented by

$$l = \langle x, y, \theta \rangle, \quad l \in M \tag{4.1}$$

where x, y denotes the robot coordinates in map Cartesian reference frame and θ is the robots heading angle. Then let us consider the motion of robot formulated as a conditional probability function given by

$$P(l'|l,a) \tag{4.2}$$

which denotes the robot movement from position l to position l' with performed action a. Note that action a could represent the velocity command, steering command, or any other variable inducing the robot change of position. Eq. 4.2 is called motion model. Finally, let us assume the sensor model as a conditional probability function

$$P(s|l) \tag{4.3}$$

which represents the likelihood, that measured data s are the result of a robot being at the position l.

4.2.1 Markov Localization

The Markov Localization (ML) is the probabilistic approach of robot pose estimation based on the measured data s and performed motions caused by action a. The Markov localization is introducing the belief distribution

$$B(l) \in (0,1), \tag{4.4}$$

which is giving the probability of robot being in position l for any $l \in M$. Initially, when the robot has no prior knowledge of robot position, $B(l)$ is represented by uniform distribution, giving the same probability for every state l. This Belief is then updated in two stages – Robot motion stage and Sensor readings stage.

The robot motion stage is performed when the robot is being commanded with action a and changes the position. The Belief $B(l)$ is updated as

$$B(l) \leftarrow \int P(l|l',a)B(l')dl'. \tag{4.5}$$

The sensor reading stage use the Bayesian rule to update the belief $B(l)$ with motion model 4.3 when sensor data s are received as

$$B(l) \leftarrow \alpha P(s|l)B, \tag{4.6}$$

where α is normalizing factor which ensure that

$$\sum_{l \in M} B(l) = 1 \tag{4.7}$$

This update process is applicable only if the environment is *Markovian.* Thus the past sensor readings are conditionally independent of future readings. Belief update is repeated with every following sensor readings s and robot action a. The state with the largest probability $B(l)$ is then picked as a position estimate.

Working with the belief $B(l)$, which has to keep information about the probability of every state l in the large discrete domains M such as occupancy grids, would be very demanding. Hence the solution of ML has to be extended.

4.2.2 Monte Carlo method

The MCL key idea is to approximate the ML belief $B(l)$ by a set of N weighted, random samples distributed over the domain M. Those samples are called particles and are represented by

$$\langle l, p \rangle = \big\langle \langle x, y, \theta \rangle, p \big\rangle, \tag{4.8}$$

where l denotes some position on the map and p is a numerical weighting factor. For all samples have to apply

$$\sum_{n=1}^{N} p_n = 1, \tag{4.9}$$

thus the weighting factors are analogous to discrete probability.

Initially, when we do not have any prior information about real robot position, all N particles are distributed uniformly over the occupancy grid. The goal of MCL is to optimally update the prior belief (Position of particles and their weights) based on robot movement caused by action a and the received sensor data s. The procedure is the same as for the ML, and it's structured into Robot motion stage and sensor readings.

The Robot motion is performed when the robot perform movement with action command a. Each particle is shifted from position l to different position l', which is randomly picked from the condition probability 4.2. Weighting factors p of all shifted particles are set to value N^{-1} and the **Sensor readings** is then incorporated. **Sensor readings** uses the current measurement data s to recompute each particle weighting factor p with sensor model 4.3

$$p = \alpha P(s|l'), \tag{4.10}$$

where α is the normalization factor that ensures the condition 4.9. The N particles with their weighting factors then create the new approximation of belief for the next generation, and a new sample of N particles is randomly picked from this belief.

It can be shown, that estimation of the robot position is getting better with every next sample and particles converge to the real vehicle position with some covariance. From all particles, the one with the highest likelihood is being picked and considered as an estimated position. The size of the particles sample N could influence the efficiency of the MCL, but since the MCL is a demanding process, increasing the number of particles could lead to slow pose estimation.

4.2.3 Ray casting

In the Sensor reading step, the MCL has to be able to generate virtual range sensor data for every particle position in the map, to be able to perform likelihood calculation 4.3. This process is called ray casting, and its essential principle is shown in Fig. 4.2.

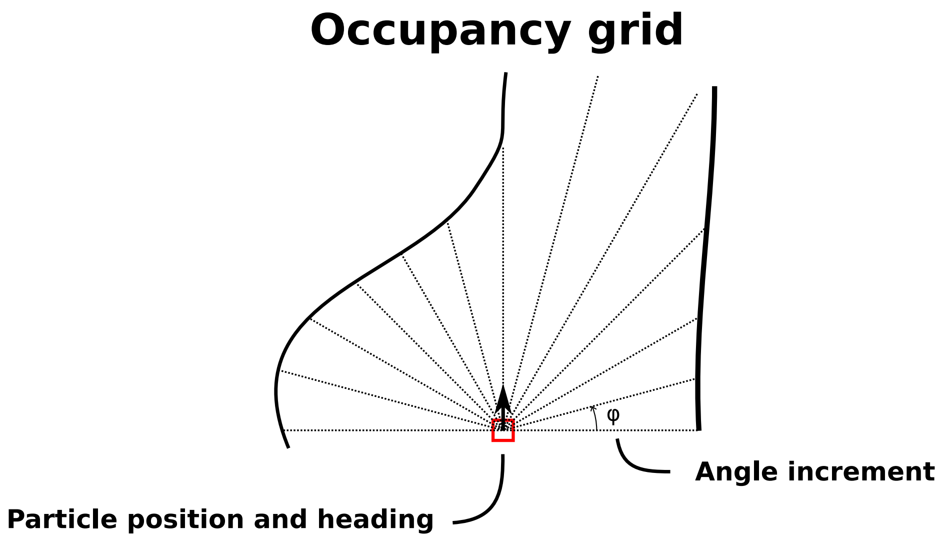


Figure 4.2: Particle virtual range sensor approximation by ray casting

Ray casting algorithm works on the provided occupancy grid of the environment and a sample of N particles. Given the algorithm, the so-called casting query $(x, y, \theta)_{query}$, the algorithm finds the closest obstacle $(x, y)_{colide}$ in the desired direction θ_{query} and returns the Euclidian distance d

$$d = \sqrt{(x_{colide} - x_{query})^2 + (y_{colide} - y_{query})^2)}. \tag{4.11}$$

The algorithm is determining the particle distances from the obstacles by casting multiple rays in the given range of directions and angle increment ϕ. Since the dozens of rays have to be cast for each particle and thousands of particles has to be maintained every Sensor readings. The choice of the ray casting method is crucial for MCL performance.

Bresenham's Line (BL) casting method

Bresenham's Line is the basic method of line approximation in grid environment [18]. BL ray casting use this line approximation to iteratively search along given direction for obstacles. The BL main advantage is that the initialization time of the algorithm is almost none compared to other algorithms since the BL method uses the pure map with no other adjustments.

Ray marching (RM) casting method

Ray marching method introduced in [19] is also using the BL algorithm for line approximation, but before the algorithm is initialized, RM method generates the look-up table, where the every occupancy grid cell is assigned the distance to the closest obstacle. When the RM is then performing the ray casting, the line is not searched one by one cell, but algorithm iteratively jumps for the number of cells of closest obstacle given by look-up table. This principle is shown in Figure 4.3.

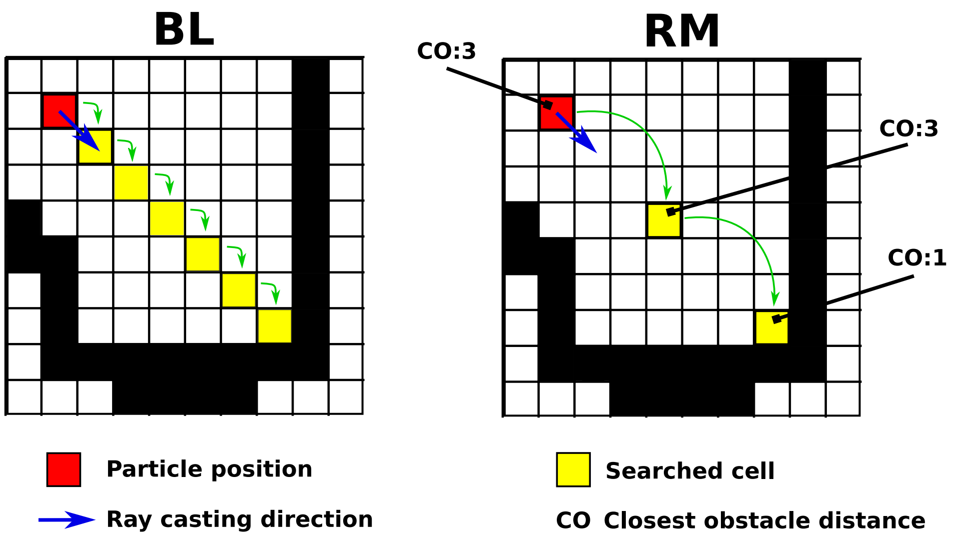


Figure 4.3: Comparison of BL and RM function

The RM method is generally faster than BL method and in the worst case (Ray is heading close and along the wall) is as good as BL method. The initialization of the algorithm is slower because of the look-up table computation, but that is not an issue since the map is not changing during the localization task.

Compressed Directional Distance Transform (CDDT) casting method

The CDDT casting method introduced in [20] utilizes the three-dimensional look-up table. This look-up table stores the closest distance for given particle coordinates and direction, hence in 2D grid map no more searching is needed, and the casting query return the distance of obstacle in given direction immediately.

4.2.4 Ray casting comparison

The three discussed ray casting methods were tested in several conditions. The few test drives were made on different types of tracks, and the data from sensors were recorded. On the recorded data samples, the MCL was performed with the usage of different ray casting methods and the varied number of particles. The result of the measurement is seen in Fig. 4.4

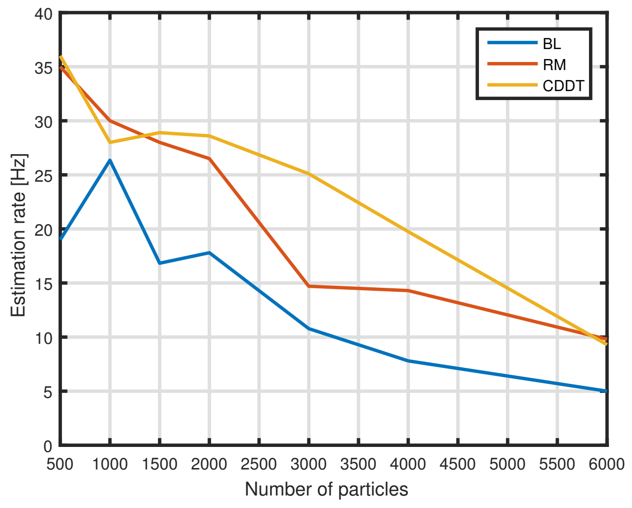


Figure 4.4: The estimation rate of MCL with different ray casting method

From the result could be seen, that CDDT ray casting method has the best performance, especially in the area of 3500-4000 particles, where the localization provides the best performance in the sense of losing the position. The 6000 particles is the MCL limit, for which any of ray-casting method was not able to localize the vehicle robustly for higher speeds

4.3 Odometry

The odometry of the robot use the data about vehicle velocity and steering to estimate robot relative motion. As was already mentioned, the odometry suffer to additive error and its not possible to use it for pure localization. However, the knowledge of odometry could be used as an action a in MCL **Robot motion** stage for robot realtive motion estimate. To determine the odometry, the data from vehicle ESC (VESC) unit will be used in the combination with the data about the steering command.

4.3.1 Velocity identification

The forward rolling motion of the racing platform is provided by DC-brushless motor controlled by low-level velocity controller VESC. The VESC is controlled by PWM signal from the computer unit, thus the duty cycle of the PWM signal is the velocity action command a_l.

From experiments was found out, that the maximum forward velocity was reached with duty cycle 11.96%, minimum forward speed needs duty cycle $9,56\%$, minimum backward speed is set with duty cycle 8.54%, and maximum backward speed require duty cycle 5.98%. Between minimum forward and backward speed is the deadzone, where car is being stopped.

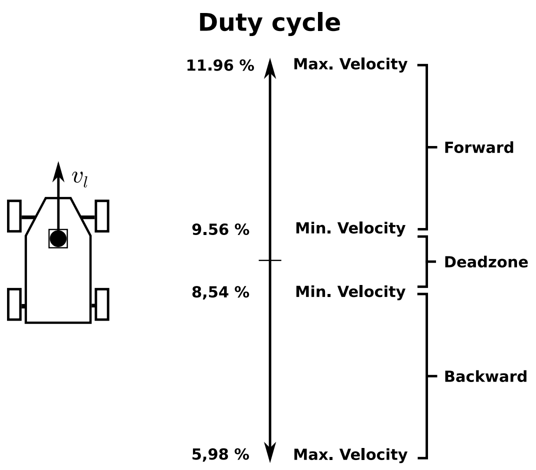


Figure 4.5: Forward and backward velocity duty cycle limits

To identify the vehicle velocity characteristics, the simple experiment was conducted. The Constant duty cycle has been set for a given interval and the average speed of the car was derived from travelled distance. From this experiments we get the results shown in Figure 4.6

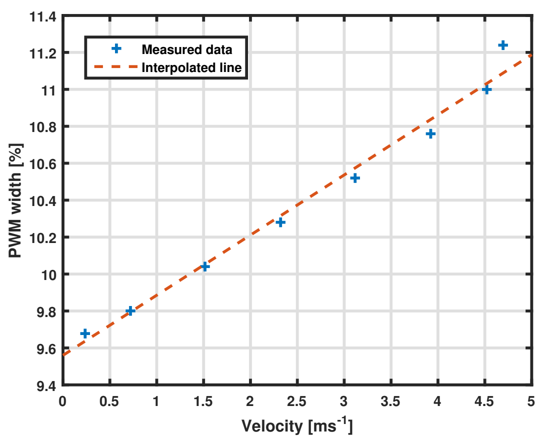


Figure 4.6: Forward velocity identification

Regarding to data pattern, the relation between duty cycle and vehicle velocity was determined as linear. The data were interpolated with function

$$a_l = v_{gain} v_l + v_{off} = 0.33 v_l + 9.56 \tag{4.12}$$

4.3.2 Steering identification

Similarly as the velocity, the steering identification has to be done for correct odometry evaluation. The Vehicle is steered by servomotor also controlled with duty cycle of PWM signal. The servomotor is setting the steering angle δ with action command a_s.

Values of duty cycles for maximum and minimum values of left and right steering are shown in Figure 4.7

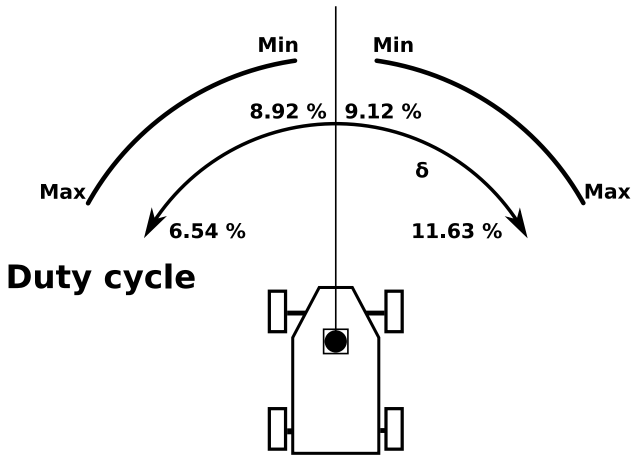


Figure 4.7: Steering duty cycle limits

The goal of the identification is to find a function describing the relation between the action command a_s and the steering angle δ. Because of that, the following experiment was conducted. The constant duty cycle a_s was set to the vehicle together with low velocity command a_v. The vehicle starts to drive along the fixed sized circle with radius R. The radius is measured and the steering angle could be then derived from equation 2.1 as

$$\delta = \arctan \frac{L}{R} \tag{4.13}$$

The data from the measurement shown in the Figure 4.8 could be then interpolated with linear equation

$$a_s = s_{gain} \delta + st_{off} = 5.91 \delta + 9.02 \tag{4.14}$$

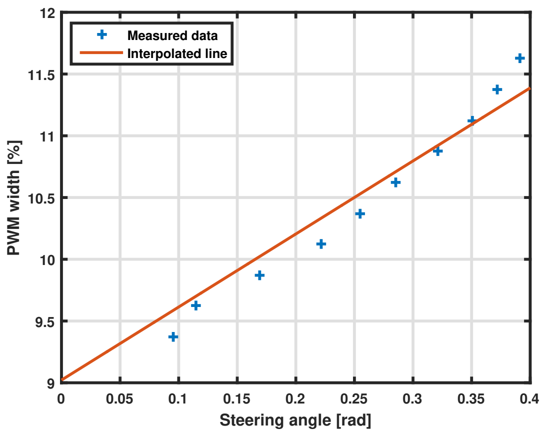


Figure 4.8: Steering duty cycle limits

4.3.3 Odometry calculation

With the derived relations 4.12 and 4.14 the odometry could be evaluated. The goal of this calculation is to determine the most accurate estimate of the current longitudinal velocity v_l and the angular velocity $\dot{\theta}$. The VESC unit is able to provide a feedback about the output of brushless motor velocity low-level control in the form of duty cycle. Unfortunately, the servomotor is not able to provide any feedback, hence the current steering command provided to the servomotor has to be used. Taking steering command as the input information means, that we are neglecting the servomechanism dynamics, however, since we consider only small relative changes of steering angle during the control process, we will deal with that.

The velocity of the car v_l is directly computed from the action command a_l and equation 4.12. The angular velocity $\dot{\theta}$ is determined from equation 4.14, vehicle steering command a_s and vehicle kinematic equation 2.4 as

$$\dot{\theta} = v_l \frac{\tan\delta}{L} = v_l \frac{\tan(st_{gain} a_s + st_{off})}{L} \tag{4.15}$$

4.3.4 Odometry testing

Regarding to the fact, that odometry identification was performed with simple techniques, which could contain lot of uncertainty. The testing experiment was conducted. The car is driven along the taped cross on the floor and command to make several maneuvers essentially outlined in the Figure 4.9

The goal of the test is to tune odometry constants in equations 4.12 and 4.14, thus the additive error is decreased to minimum and in final position the odometry pose estimation will be as close as possible to initial position.

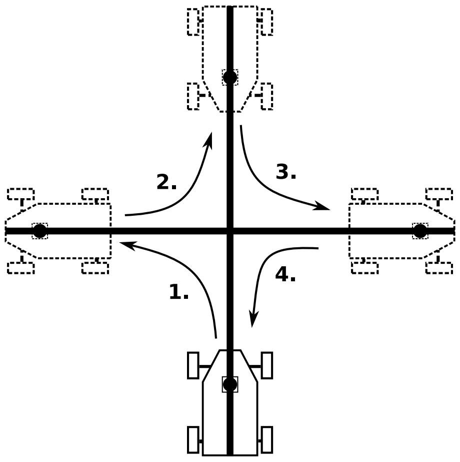


Figure 4.9: Odometry testing maneuver

The best result was obtained, when the steering interpolation gain was changed to

$$a_s = s_{gain}\delta + st_{off} = 5.5\delta + 9.02. \tag{4.16}$$

Then the odometry provided pose estimetion shown in Fig. 4.10.

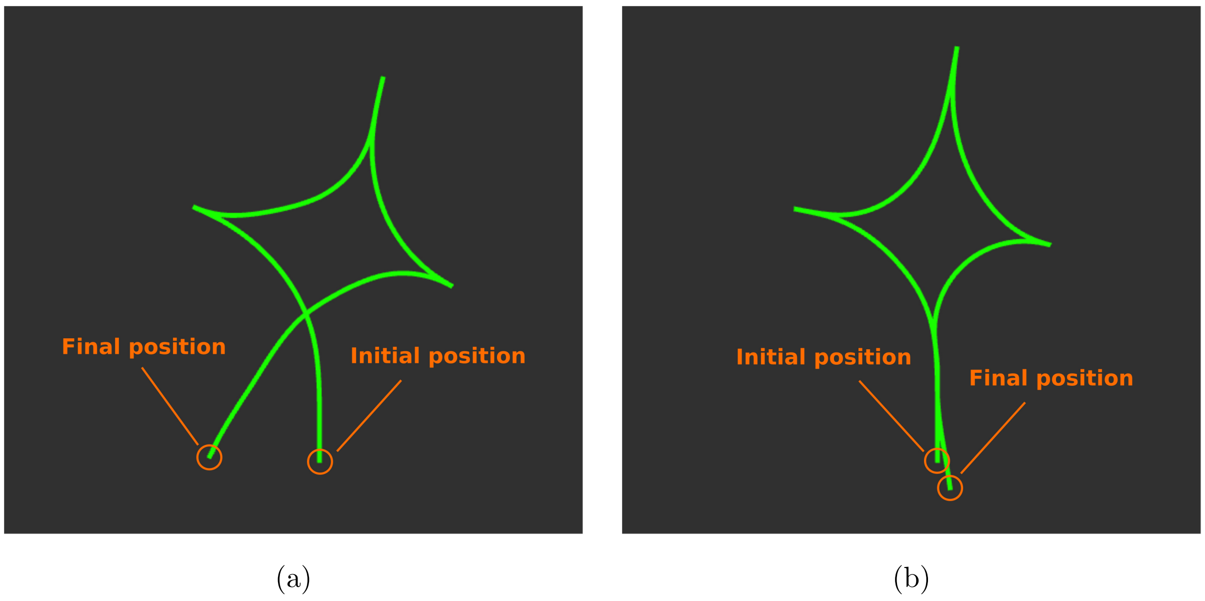


Figure 4.10: The odomerty position estimation of testing maneuver before correction (a) and after correction (b)

4.4 Increasing pose estimation rate and filtering

The MCL performs position estimation with the different frame rate based on number of used particles and ray casting method. Hence the data from odometry are usually available more often, the idea of data fusion is to make the other estimation of position regarding to position estimate from MCL and data from odometry, The goal of the data fusion is to enlarge rate of the position estimation, which could be used for better control. or provide the best pose estimation in case when the data from particle filter are not available.

4.4.1 Relative pose estimator

In this section the Relative pose estimator algorithm will be introduced as a simple method how to increase the rate of vehicle pose estimations using the data from odometry and knowledge of vehicle kinematics. The essential function of the algorithm is shown in the Figure 4.11

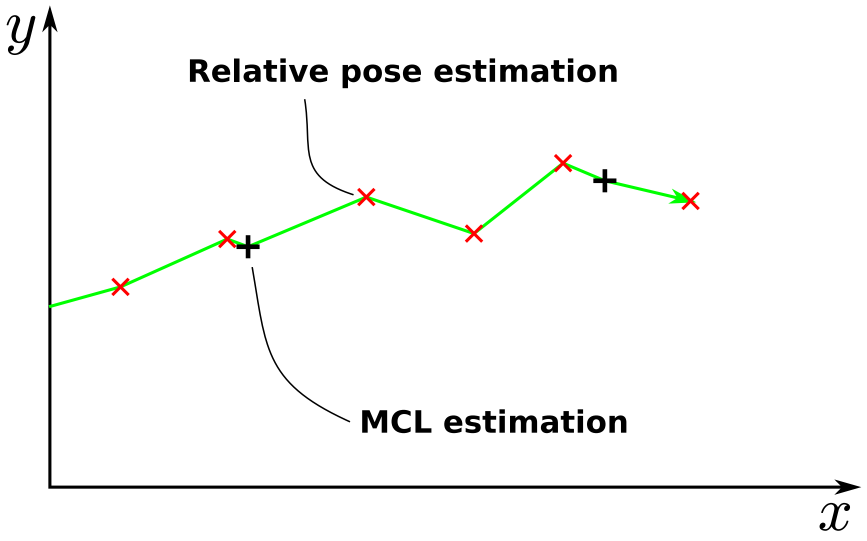


Figure 4.11: Relative pose estimation from odometry

Let the p_{est} to be a position estimation of the Relative pose estimator algorithm denoted as

$$p_{est} = \langle x_e, y_e, \theta_e \rangle, \tag{4.17}$$

where the x_e, y_e and θ_e are the estimated coordinates of the vehicle in world frame. The algorithm is initiated with the incoming odometry data o_k in the form

$$o = \langle v_k, \dot{\theta}_k \rangle \tag{4.18}$$

and the incoming data from MCL as a pose estimation p_{MCL}. The algorithm set the estimated position regarding to MCL estimation

$$p_{est} = p_{MCL} \tag{4.19}$$

and wait for next odometry data available in time $k+1$. When the data from odometry o_{k+1} are available the dt is introduced as time difference between time of last estimation update and time $k + 1$. The estimation of vehicle pose is then updated with kinematic model as

$$\begin{aligned} p_{est} =& \langle x_e + v_k \cos(\theta_e + \dot{\theta}_k dt)dt, \\ & y_e + v_k \sin(\theta_e + \dot{\theta}_k dt)dt, \\ & \theta_e + \dot{\theta}_k dt \rangle \end{aligned} \tag{4.20}$$

This procedure repeats when the next odometry data are available until the MCL perform next estimation and the p_{est} is corrected again

$$p_{est} = p_{MCL} \tag{4.21}$$

The result of this estimation could be seen in Figure

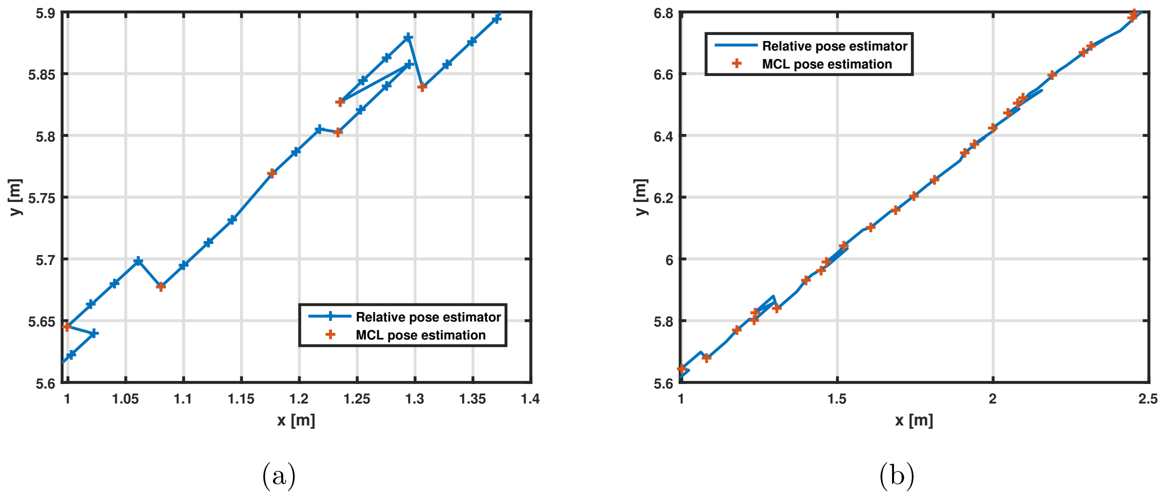


Figure 4.12: Result of relative pose estimation on real data

4.4.2 EKF for ackermann platform kinematics

In this section, method of position filtering will be introduced to provide an option for noisy data case of MCL estimation. The MCL could provide a noisy data in several cases. In the first case, the small number of MCL particles could result in noise caused by worse probabilistic properties. The second case could be caused by difficult structure of the surrounding environment with similar patterns. The filter use the knowledge of odometry, vehicle kinematics and given statistic properties to perform the optimal estimate.

For the filtration the extended kalman filter will be used, which could handle the nonlinear kinematics of ackermann platform. The discrete dynamic system of vehicle could be defined as

$$x_k = f(x_{k-1}, u_{k-1}) + w_{k-1} \tag{4.22}$$
$$y_k = h(x_k) + v_k, \tag{4.23}$$

where x_k is the inner state of the system in discrete time k, u_k is input to the system, $f(x_k, u_k)$ is the nonlinear state equation of the system and w_k is the process noise. y_k then denote the output of of the system, $h(x_k)$ is the nonlinear output equation and v_k is the measurement noise. The process and measurement noise are modeled as a white noise with the covariances

$$E[w_k {w_k}^T] = Q \tag{4.24}$$
$$E[v_k {v_k}^T] = R \tag{4.25}$$
$$\tag{4.26}$$

and random vectors w_k and v_k are assumed to be uncorrelated, thus apply

$$E[w_k {v_j}^T] = 0 \quad \text{for all } k \text{ and } j \tag{4.27}$$

The Extended Kalman Filter is divided into two steps – Model Forecast step and Data assimilation step. Hence the probability properties are in most cases unknown, matrixes Q_k and R_k are usually considered as an adjustable part of the filtering and are set manually to gain the best filtering performance. The state vector x_k is the state of the vehicle considered as vehicle position coordinates and heading angle

$$x_k = \begin{bmatrix} xc_k \\ yc_k \\ \theta_k. \end{bmatrix} \tag{4.28}$$

Since the equation $f(x_k)$ is nonlinear, Extended Kalman Filter use the Tyler expansion of first order to approximate the forecast and next estimation of x_{k+1}.

The filtering is intiated with state x_0 and initial covariance P_0 such as x_0 equals to last known position from MCL and $P_0 = Q$. Then the Model Forecast Step is performed

Model Forecast Step (Predictor)

The Model Forecast step propagates the current estimated state and covariance throw state equation. The nonlinear state equation of the vehicle kinematics is

$$x_k = f(x_{k-1}) = \begin{bmatrix} xc_{k-1} + v_l \cos(\theta_{k-1} + \dot{\theta}dt)dt \\ yc_{k-1} + v_l \sin(\theta_{k-1} + \dot{\theta}dt)dt \\ \theta_{k-1} + \dot{\theta}dt \end{bmatrix}, \tag{4.29}$$

where v_l, $\dot{\theta}$ is the longitudinal velocity and angular velocity of the vehicle, considered as last known data from the odometry. The sampling time of the filter is denoted as dt and its set to match the rate of MCL estimations. The state forecast ${x_k}^f$ is than performed as

$$x_k^f = f(x_{k-1}^a) \tag{4.30}$$

$$P_k^f = J_f(x_{k-1}^a) P_{k-1} J_f^T(x_{k-1}^a) + Q, \tag{4.31}$$

where x_{k-1}^a and P_{k-1} denotes the optimal estimation and covariance from last step (Initially considered as x_0 and P_0) and J_f states the Jacobian of nonlinear state equation $f(x_k)$

$$J_f = \begin{bmatrix} \frac{\partial f_1(x_k)}{\partial x_c} & \frac{\partial f_1(x_k)}{\partial y_c} & \frac{\partial f_1(x_k)}{\partial \theta} \\ \frac{\partial f_2(x_k)}{\partial x_c} & \frac{\partial f_2(x_k)}{\partial y_c} & \frac{\partial f_2(x_k)}{\partial \theta} \\ \frac{\partial f_3(x_k)}{\partial x_c} & \frac{\partial f_3(x_k)}{\partial y_c} & \frac{\partial f_3(x_k)}{\partial \theta} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -v_l \sin(\theta + \dot{\theta}dt)dt \\ 0 & 1 & v_l \cos(\theta + \dot{\theta}dt)dt \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.32}$$

Data Assimilation Step (Corrector)

The Data Assimilation Step uses the Linear Mean Square estimate to perform the estimation between the new measured data y_k and the forecast prediction x_k^f with the following equations

$$K_k = P_k^f J_h^T(x_k^f) \Big(J_h(x_k^f) P_k^f J_h^T(x_k^f) + R \Big)^{-1} \tag{4.33}$$

$$x_k^a = x_k^f + K_k(y_k - h(x_k^f)) \tag{4.34}$$

$$P_k = (I - K_k J_h(x_k^f)) P_k^f. \tag{4.35}$$

The J_h is generally the Jacobian of output nonlinear equations $h(x_k)$, however, since measured data are equal to the inner state of the system, the Jacobian J_h is the identity matrix

$$J_h = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.36}$$

The computed state x_k^a is the final estimation used as the output from the filter and as a the initial state for the next round of the filtering process. The whole process is repeated with the next measurement. Result of filtration by EKF is seen in Fig.

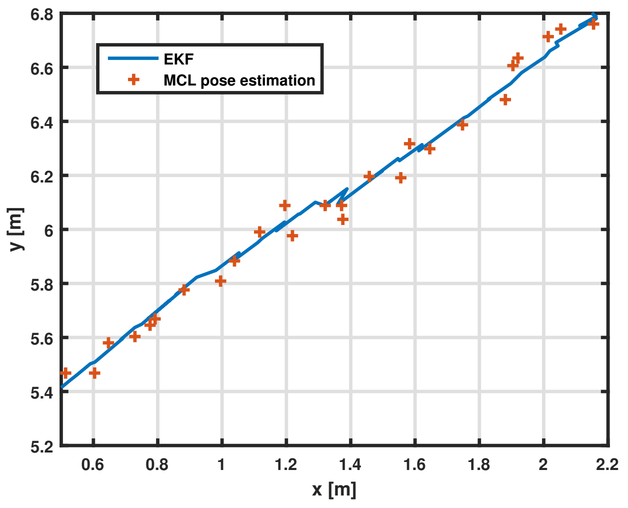


Figure 4.13: Pose filtering by EKF

4.4.3 Result discussion

From the experiments of MCL localization could be seen, that wheel odometry has the huge impact on the precision of pose estimation, thus it is important to perform the wheel odometry tuning. In some cases the user could be pushed to lower the number of MCL particles due to lack of computational power or have to localize the robot in difficult environment. In both cases, the MCL could provide noisy estimations or provide the data with insufficient rate. Therefore, the Relative pose estimator or EKF could be utilized to improve the localization process.

On the tested scenarios the MCL localization worked well in configuration of 4000 MCL particles and *CDDT* ray casting method. The result of vehicle localization on the race track is shown in Fig. 4.14

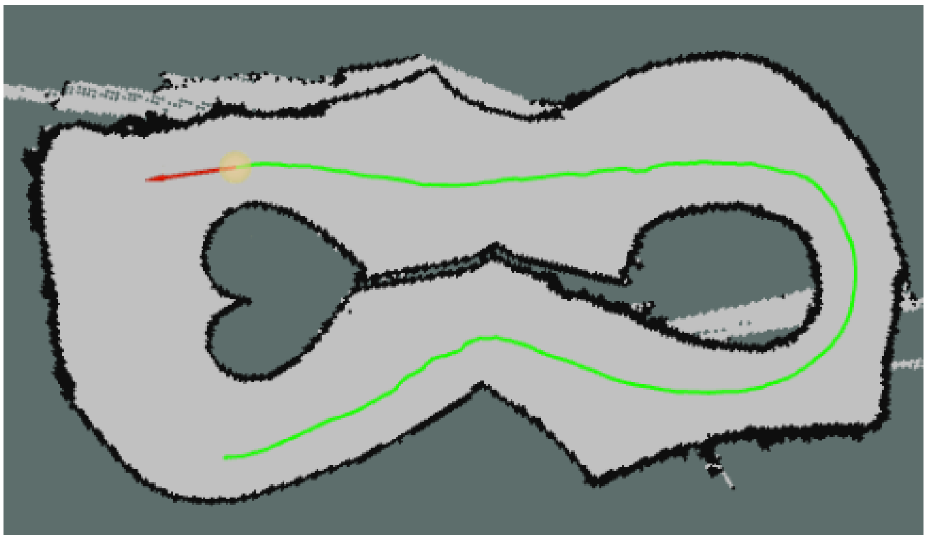


Figure 4.14: Recorded trajectory of localized vehicle

Chapter 5

Trajectory generation

When we are able to localize the car on the known map, the high-level trajectory planning could be established. The benefits of this approach are, that planned trajectory could be optimized in the way of vehicle dynamics limits and lap time. This optimization task is usually called Optimal Racing Line planning.

The trajectory is in all calculations considered as a curve, however in the discrete world is being approximated as a sequence of points, where every point gives the vehicle reference for the desired position, heading angle and speed or acceleration. To be able to construct the trajectory leading the car inside the racing circuit, the several features have to be recognized from the map such as left and right track boundary, track starting line or racing direction. The planning algorithm also have to consider the size properties of the vehicle, to avoid collisions caused by planing the trajectory near the track boundaries.

This thesis focuses on the recognition of basic track features from the map and planing of simple trajectory leading the car through the lap. The optimal racing line problem is briefly reviewed, however, due to the complexity of the task is not further examined or implemented.

5.1 Optimal Racing line problem

The Optimal Racing line is the trajectory allowing vehicle to travel through a given track in minimum time. This trajectory is not necessarily the shortest one, but have to allow the vehicle to utilize its abilities as much as possible in turns and long corridors. Task of racing line optimization dates in early 1960s, when first gradient descent and shooting methods, were used for optimizing minimum time manoeuvring. Since then, lot of other approaches has been introduced. In the control systems point of view, the task could be stated as a two-point boundary problem as described in [21]. However, most of the algorithms generally split the track into short segments (or Bezier curves) and try to use techniques of evolutionary algorithms [22],[23],[24] to find a most suitable solution. The interesting approach is also introduced in [25], where the optimizing algorithm parameters are directly adjusted to handle limits of single track dynamic model.

5.2 Central Trajectory algorithm

The following algorithm is the method how to automatically plan a simple trajectory through the track. The method was introduced to discover problems related to recognition of basic track features and to create a simple trajectory on which the tracking control algorithms could be tested.

5.2.1 Walls recognition

The track is assumed as a corridor with left and right wall connected as two loops. To lead the trajectory inside the track, walls has to be recognized. The occupancy grid is represented by three numbers, -1 denotes unexplored cell, 0 is the explored unoccupied cell and 1 is explored and occupied cell. The algorithm consider wall as a continuous set of points (occupied cells). The first occupied cell on the map is found randomly and added to the set. The neighborhood around this point is searched and from the searched area all the occupied points are added into the continuous set until no more neighbors are found. All found continuous sets are sorted by size and two larges sets are considered as walls.

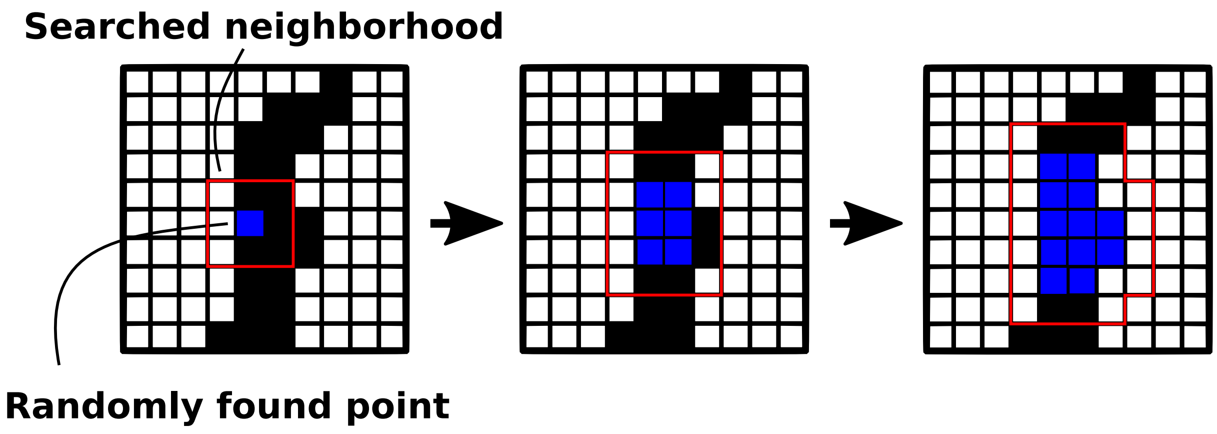


Figure 5.1: Iteratively growing continuous set

5.2.2 Center of the track

To find the center of the track, the analogy to flood-fill algorithm is used. Two continuous sets recognized as a walls are marked with different numbers, thus the cell number does not have a meaning of grid occupancy anymore, but recognize to which set of the two walls the particular cell belong. In the every iteration, the algorithm searches around neighborhood of walls and expand the set to the unoccupied cells. This process is repeated until all cells in the map are occupied.

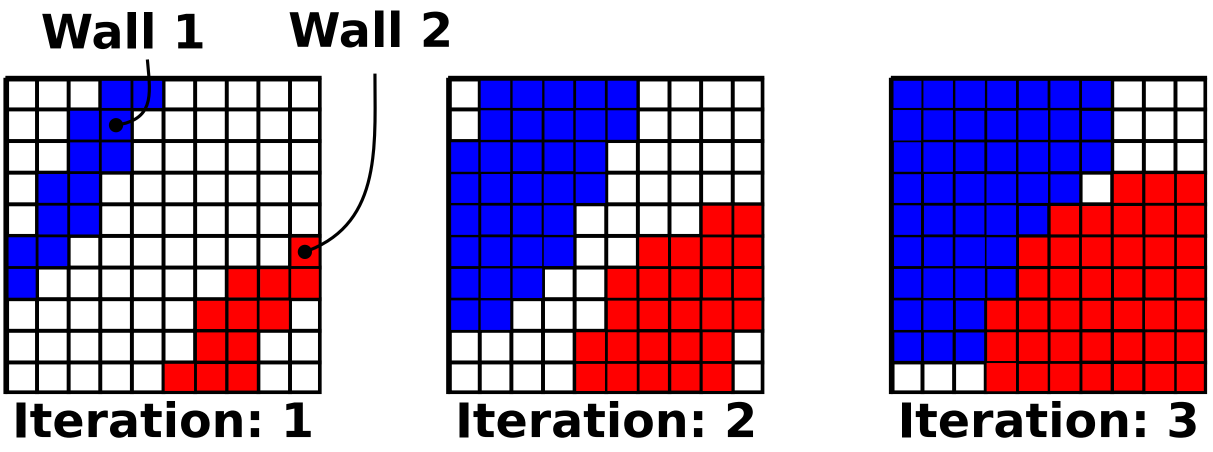


Figure 5.2: Map flooding

Flooded map creates the interface between two sets. The points lying on this interface are taken out and considered as an approximation of the center between walls of the track. Since the trajectory has to be a sequence of following points, the sorting has to be done to put points in order and create a loop. Hence the algorithm worked on the occupancy grid the created trajectory is non-smooth. The non-smooth trajectory like that would generate a non-smooth reference for the vehicle controller, hence the smoothing is performed, by taking every N-th point of the created sequence, which makes the final trajectory much easier to track.

5.2.3 Resulting trajectory

The final cut-off trajectory is then interpolated with spline of the third order, to get a smooth trajectory leading approximately in the center of the track. The examples of trajectories automatically generated by Central Trajectory algorithm are shown in Figure 5.3

5.3 Velocity profiling

The trajectory generated by Central Trajectory algorithm is the sequence of points giving the vehicle a position reference. To perform a vehicle utilization in turns and straight corridors, the trajectory speed reference has to be introduced. This reference has to consider the vehicle kinematics and dynamics to prevent slipping in sharp turns and handle vehicle to maximum speed in long corridors.

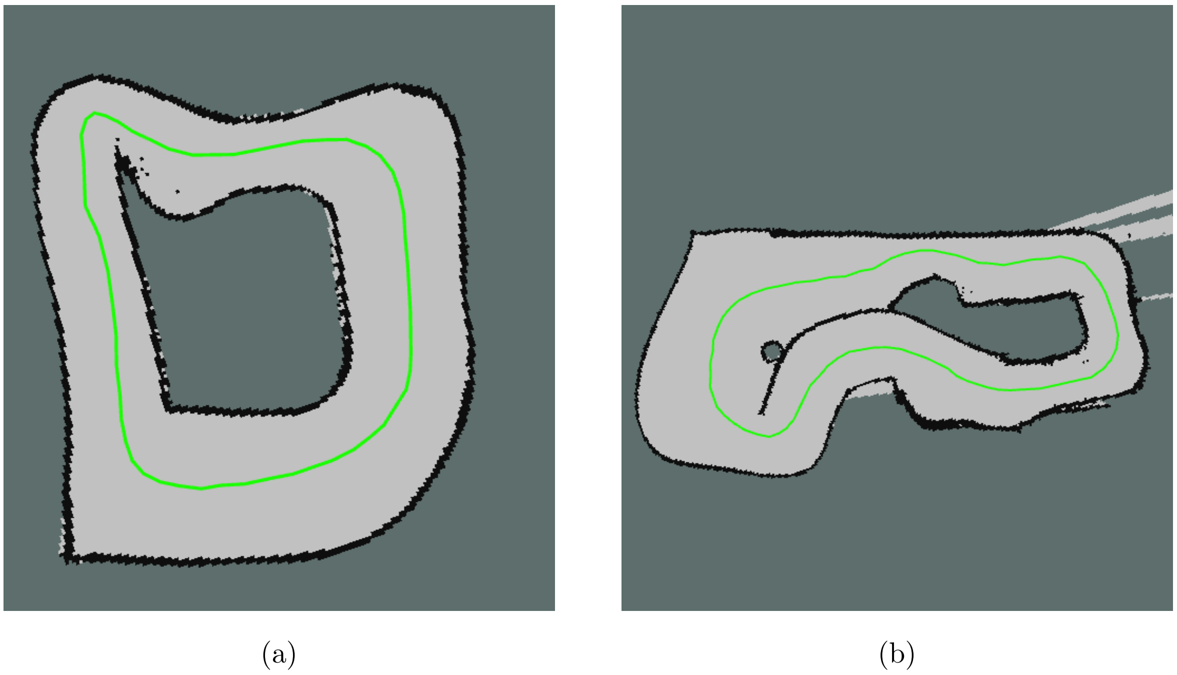


Figure 5.3: Trajectories generated by Central Trajectory algorithm

The vehicle utilization could be understood as driving with maximum possible speed without slipping and was introduced in [2] and [26]. That is usually affected by several aspects like, mass of the vehicle, friction of the track or curvature of the followed trajectory. For the purpose of the given racing task the following dynamic model will be used

$$F_x = ma_x - bv \tag{5.1}$$

$$F_y = mv^2\kappa \tag{5.2}$$

$$F_z = mg, \tag{5.3}$$

where m is the vehicle mass, a_x denotes vehicle longitudinal acceleration, b is the coefficient of linear rolling resistance and κ is the curvature of followed path. Eq. 5.3 denotes the same dynamic model formulated in [2], however the affect of aerodynamic forces was neglected.

To model a tire adhesion limits the friction circle model will be used, which assumes that slipping is prevented when

$$F_x^2 + F_y^2 \leq (\mu F_z)^2 \tag{5.4}$$

The task of Velocity profiling is to maximize the transferable force, but meet the criteria of the condition given by Eq. 5.4.

5.3.1 Curvature approximation

In the vehicle dynamics Eq. 5.3 is the lateral force F_y denoted as a product of longitudinal velocity and the trajectory curvature. Since the Central Trajectory algorithm provides the trajectory in form of sequence of positions, the curvature of trajectory has to be carried out.

Menger curvature

The Menger curvature is an approximation given by three point lying on the circle tangent to an approximated curve. Those points are creating a triangle shown in Figure 5.4.

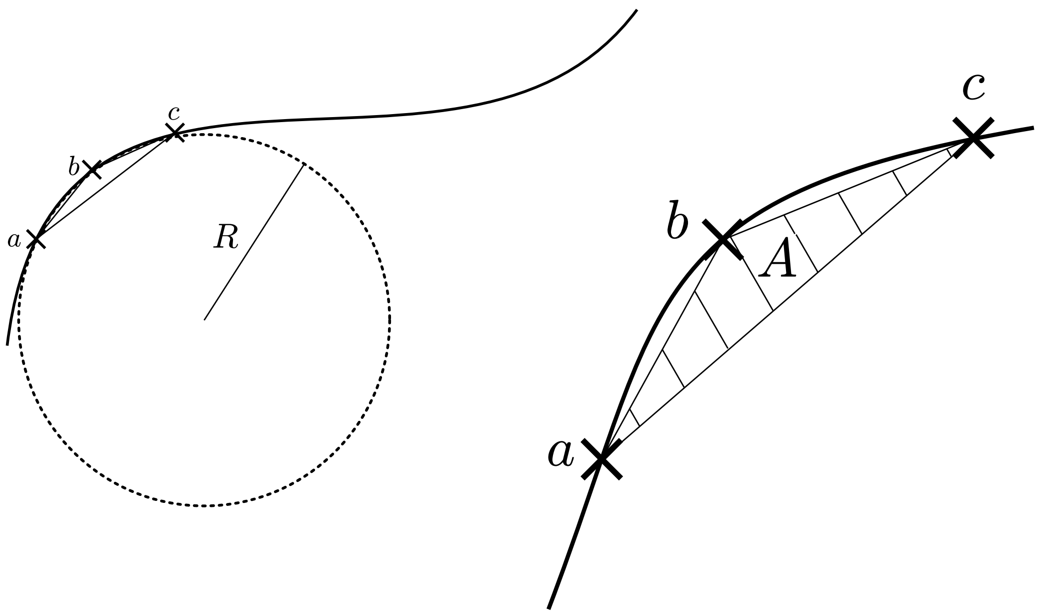


Figure 5.4: Menger curvature approximation by three points

the Menger curvature in point b is then formalized as

$$\kappa = \frac{4A}{|a-b||b-c||c-a|}, \tag{5.5}$$

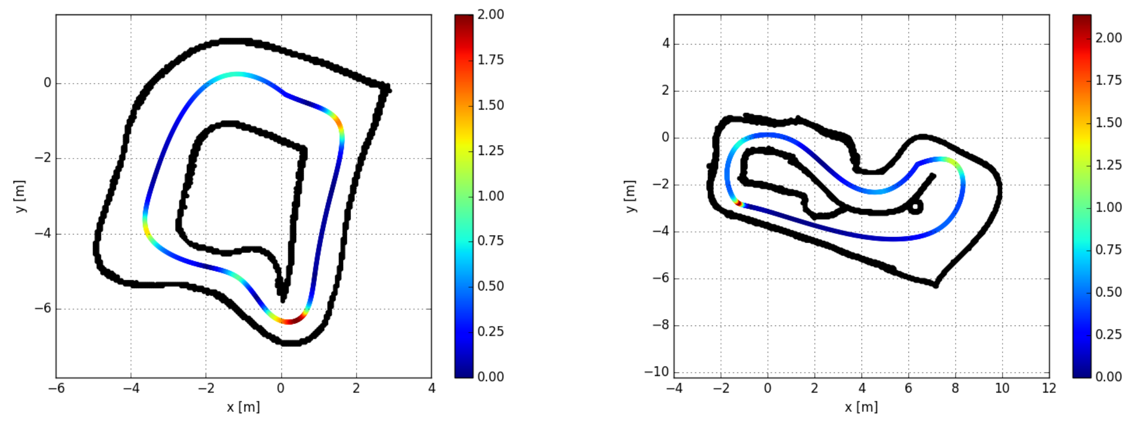
where A is the inner area of the given triangle and could be calculated with **Heron's formula** as

$$A = \sqrt{s(s-|a-b|)(s-|b-c|)(s-|c-a|)} \tag{5.6}$$

with semi-parameter of the triangle

$$s = \frac{(|a-b|+|b-c|+|c-a|)}{2} \tag{5.7}$$

The curvature is computed for every point on the trajectory. The result for large and small track is shown in Figure 5.5.



(a) (b)

Figure 5.5: curvature of predefined trajectories $[m^-1]$

5.3.2 Velocity profiling algorithm

The velocity profiling algorithm used by [2] is the two-pass iterative algorithm, which is precomputing the trajectory velocity reference in order to meet the criteria defined by Eq. 5.4. The algorithm is divided into backward and forward pass. In backward pass algorithm solves the necessary deceleration for optimal breaking before sharp turns and in forward pass the necessary accelerations are determined.

The velocity profiling algorithm is well described in [2](Chapter 3). However, in this thesis will be rewritten to keep the structure and clarify the principle.

Backward pass

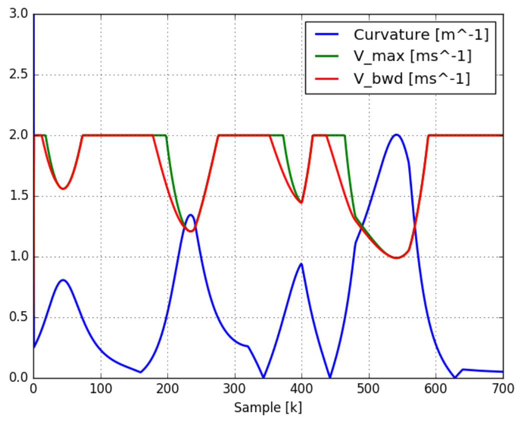
The trajectory is defined as sequence of N points with given curvatures κ. The algorithm is initiated with the trajectory data, v_{lim} as a maximum velocity limit of the vehicle and Δs, which denotes the distance between trajectory points.

The backward pass computes the maximum velocity at each step for previous sample based on the solution of Eq. 5.4 for $\kappa(k-1)$ and $a_x = 0$. The result is saturated with user-defined parameter v_{lim} and the deceleration required to achieve velocity $v_{max}(k-1)$ with utilization of maximum available breaking function

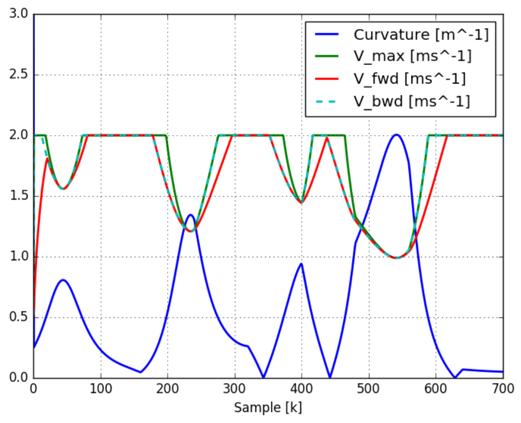
$$h(\kappa, v, d) = \begin{cases} \frac{1}{m}(\sqrt{\mu^2 F_z^2(v) - F_y^2(\kappa, v)}), & \text{if } d = 1 \\ \frac{1}{m}(-\sqrt{\mu^2 F_z^2(v) - F_y^2(\kappa, v)}), & \text{if } d = -1 \end{cases} . \tag{5.8}$$

The result of the algorithm is shown in Figure 5.6(a)

```
Algorithm 1: Backward pass
  Data: $\kappa(k), v_{lim}, \Delta s$
  Result: $v_{bwd}(k), v_{max}(k)$
  $k \leftarrow$ length(N);
  $v_{bwd}(k) \leftarrow v_{lim}$;
  while $k > 1$ do
      $v_{max}(k-1) \leftarrow$ solution of Eq. 5.4 for $\kappa(k-1)$;
      $v_{max}(k-1) \leftarrow \min(v_{max}(k-1), v_{lim})$;
      $a_{lim} \leftarrow [v_{max}^2(k-1) - v_{bwd}^2(k)]/(2\Delta s)$;
      $a(k-1) \leftarrow -h(\kappa(k), v_{bwd}(k), -1)$;
      $a(k-1) \leftarrow \min(a(k-1), a_{lim})$;
      $v_{b}wd(k-1) \leftarrow \sqrt{v_{bwd}^2(k) + 2a(k-1)\Delta s}$;
      $k \leftarrow (k-1)$;
```



(a)



(b)

Figure 5.6: Generated speed profile of backward (a) and forward (b) pass

Forward Pass

The forward pass utilize the computed sequence from backward pass algorithm and calculates the vehicle maximal accelerations to achieve the optimal speed profile, the algorithm is described in 2. The final speed profile is shown in Fig. 5.6(b)

Algorithm 2: Forward pass

```
**Data:** $\kappa(k), v_{bwd}(k), \Delta s, v_0$
**Result:** $v_{fwd}(k), a(k), t(k)$
$k \leftarrow 1$;
$t(k) \leftarrow 0\ v_{fwd}(k) \leftarrow v_0$;
**while** $k < N$ **do**
    $a_{lim} \leftarrow [v_{bwd}^2(k+1) - v_{bwd}^2(k)]/(2\Delta)\ a(k) \leftarrow h(\kappa(k), v_{fwd}(k), 1)$;
    $a(k) \leftarrow \min(a(k), a_{lim})$;
    $v_{fwd}(k+1) \leftarrow \sqrt{v_{fwd}^2(k) + 2a(k)\Delta}$;
    $t(k+1) \leftarrow t(k) + 2\Delta s[v_{fwd}^2(k+1) + v_{fwd}(k)]^{-1}$;
    $k \leftarrow (k+1)$;
```

Chapter 6

Control

In the following Chapter, several vehicle control strategies of trajectory tracking will be introduced. The problem of designing such control system does not rely in the design of controller itself, but also on correct method of trajectory preview and mathematical modelling of the given system.

6.1 Tracking error definition

The error is the value given by the reference signal and feedback of the dynamic system provided to input of controller. The shape of this error signal directly affects the controller behavior and its formalization has to be approached carefully. The tracking error as a value giving the controller feedback depends on the trajectory preview method. Therefore in following section several trajectory preview methods will be introduced as an approaches of tracking error computation.

6.1.1 Closest point ahead error

The Closest point ahead method briefly introduced in [27] assumes the trajectory of discrete points and do not provide any interpolation of the curve between them. The vehicle reference is taken as a closest point ahead (with non-negative error x_e) on the defined trajectory, as shown in Figure 6.1.

The error is computed from coordinates of the vehicle (x, y, θ) and coordinates of the reference point (x_r, y_r, θ_r) as follows

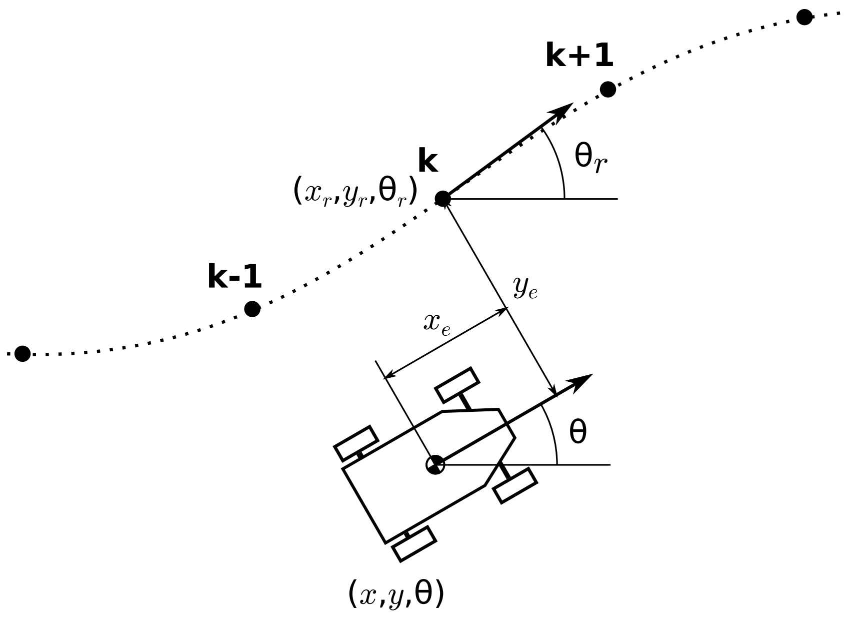


Figure 6.1: Closest point ahead error

$$x_e = \cos(\theta)(x_r - x) + \sin(\theta)(y_r - y) \tag{6.1}$$

$$y_e = -\sin(\theta)(x_r - x) + \cos(\theta)(y_r - y) \tag{6.2}$$

$$\theta_e = \theta_r - \theta. \tag{6.3}$$

The error y_e and θ_e is then usually used for a lateral control of the vehicle. For the longitudinal controller the speed given by velocity profile in point k is taken as a reference v_{ref} and velocity error v_e is computed by

$$v_e = v_{ref} - v_l, \tag{6.4}$$

where v_l is current longitudinal velocity of the vehicle

6.1.2 Crosstrack error

The Crosstrack error well described in [28], is the method which use the interpolation to a line connecting the two closest trajectory points. The interpolation is outlined in figure 6.2

To preview the trajectory, the two closest point (O_1, O_2) are found by Euclidean distance. The line connecting these points could be expressed in vector form as

$$ax + by + c = 0, \tag{6.5}$$

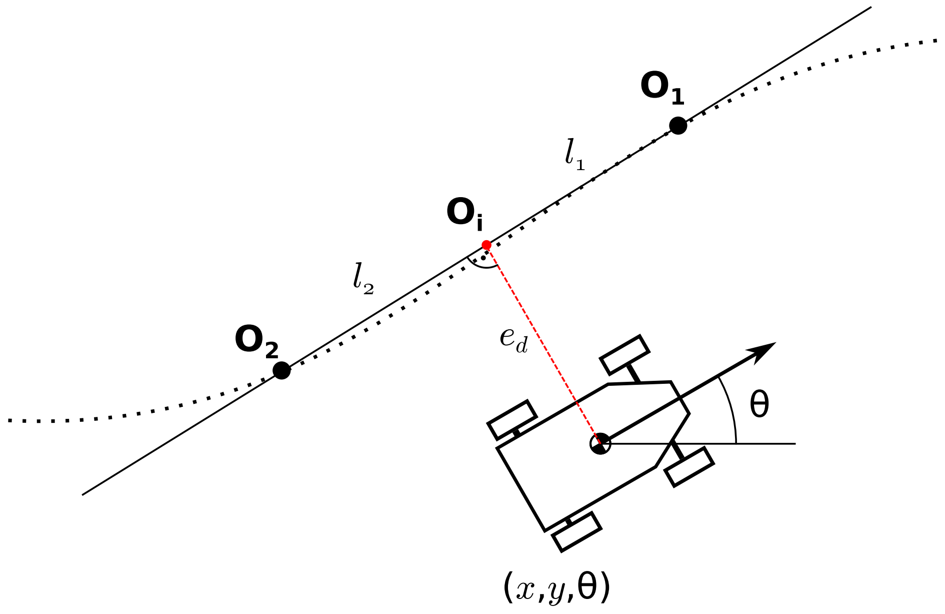


Figure 6.2: Crosstrack error

where

$$a = \frac{O_{2y} - O_{1y}}{O_{2x} - O_{1x}} \tag{6.6}$$

$$b = -1 \tag{6.7}$$

$$c = (O_{1y} - O_{1x})a. \tag{6.8}$$

The interpolation point O_i is the closest point to vehicle current position (x, y, θ) lying on the line described by Eq. 6.5. The coordinates of O_i are given by

$$O_{ix} = \frac{b(bx - ay) - ac}{a^2 + b^2} \tag{6.9}$$

$$O_{iy} = \frac{a(-bx + ay) - bc}{a^2 + b^2}. \tag{6.10}$$

From the interpolation point the error of distance from the trajectory is given by norm

$$e_d = \sqrt{(}(O_{ix} - x)^2 + (O_{iy} - y)^2), \tag{6.11}$$

Other variables defined in the trajectory points O_1, O_2, such as reference velocity or reference heading angle, are interpolated with ratio of distances l_1 and l_2

$$l_1 = \sqrt{(O_{ix} - O_{1x})^2 + (O_{iy} - O_{1y})^2} \tag{6.12}$$

$$l_2 = \sqrt{(O_{ix} - O_{2x})^2 + (O_{iy} - O_{2y})^2}. \tag{6.13}$$

The interpolation of the reference heading θ_r is then given in direction of movement from point O_2 to O_1 as

$$\theta_{ref} = \theta_{r2} + (\theta_{r1} - \theta_{r2})\frac{l_2}{(l_1 + l_2)} \tag{6.14}$$

6.1.3 Lookahead

Since the controlled vehicle has holonomic constrains due to ackerman steering, its not able to make sharp turns on relatively small area. Referring the vehicle error to closest points on the trajectory as is done by the Closest point ahead (CPA) error or crosstrack error may lead for large overshoots in sharp turns. Regarding to that, the lookahead approach was introduced. The lookahead discussed in [29] and used in [30] is not relating the reference to the vehicles position, but to the lookahaed point L_h which is considered at certain distance in front of the vehicle. This approach could bring the information about turn earlier and avoid the overshoots in sharp cornering.

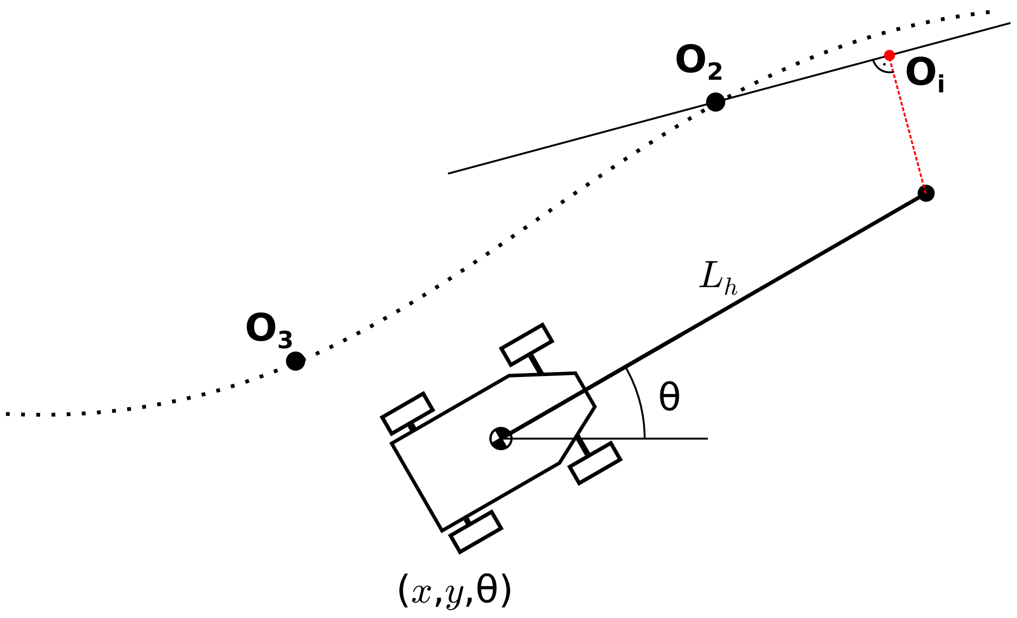


Figure 6.3: Crostrack error with lookahead

Static lookahead

The static lookahead assumes the lookahead distance L_h as a constant user-defined parameter. The lookahead point position x_L, y_L is then computed as simple transformation from vehicle coordinates

$$x_l = x + L_h \cos(\theta) \tag{6.15}$$

$$y_l = y + L_h \sin(\theta). \tag{6.16}$$

The choice of the distance is then crucial to track layout and vehicle velocity range

Adaptive lookahead

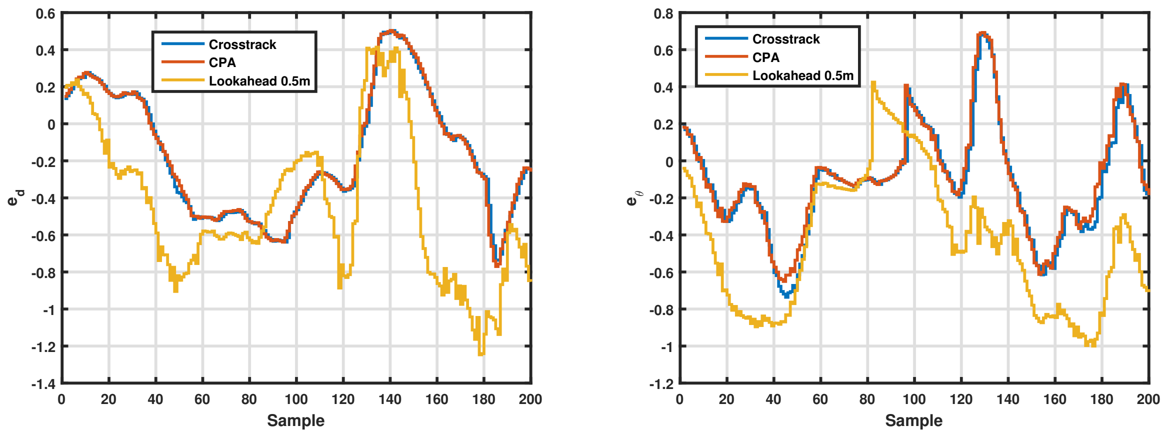
The Adaptive lookahead considers the variable lookahead distance L_h as a function of vehicle longitudinal velocity v_l. Note the task is then not formalized as a constant distance preview anymore, but could be considered as constant time preview, hence the lookahead distance L_h is computed as

$$L_h = t_l v_l, \tag{6.17}$$

where t_l denotes the constant user-adjusted time parameter. The position of the lookahead point is then computed by Eq. 6.16.

6.1.4 Experiment

To compare the tracking error method the experiment was conducted. The vehicle was manually driven around the track and trajectory errors from all methods were recorded. the result is shown in the Fig. 6.4.



(a) (b)

Figure 6.4: Comparison of tracking error methods

From the result could be seen, that on the fine sampled trajectory used in the experiment the CPA error and crosstrack error provides the same result. However, the crosstrack error is always the better option, hence it gives the smooth reference to the controller in every occasion. The lookahead error provides the controller with trajectory preview in advance as it could be seen in Fig. 6.4(b). However, the reference

6.2 Lateral control

The main task of trajectory tracking problem is the vehicle lateral control, which goal is to continuously set vehicle steering δ to follow the predefined trajectory by penalizing the tracking error. The basic approach of trajectory tracking is to use a error penalization, which directly provides the vehicle steering action. Such control could be developed without prior knowledge of vehicle dynamics or kinematics and fine tuned to work properly on a simple tracks. However, this simple control method is usually efficient under static conditions such a constant longitudinal velocity or trajectory with homogeneous curvature. The other approach is the model-based control, which utilize the platform mathematical model to determine the optimal steering action.

6.2.1 Lateral control without mathematical model

The following approach of lateral control without mathematical model was introduced in [27] to perform a task of trajectory tracking of two-wheel robot. Such robot has different holonomic constrains and kinematics then the ackermann platform. However, when the non-zero longitudinal velocity is considered, behavior of both platforms is similar.

Let us assume that the trajectory is previewed with crosstrack error defined in 6.1.2. The preview in time k gives vehicle reference position on the trajectory defined by coordinates $x_r(k)$, $y_r(k)$, $\theta_r(k)$ from which the lateral error y_e and heading error θ_e are determined. From last two samples of trajectory preview we are able to determine the angular velocity of reference heading by

$$\omega_r = \frac{\theta_r(k) - \theta_r(k-1)}{T_s}, \tag{6.18}$$

where T_s denotes the sampling period. The control law then could be introduced as

$$\omega_a = k_1\omega_r + k_2\theta_e + k_3\frac{sin(\theta_e)}{\theta_e}y_e, \tag{6.19}$$

where k_1,k_2 and k_3 are the adjustable tuning constants. The output of control law is the vehicle reference angular velocity, which determines the steering angle δ as relative change by integration

$$\delta = \delta + \frac{\omega_a}{T_s} \tag{6.20}$$

6.2.2 Model-based controller structure

The model-based control is considering the task of trajectory tracking as a linear control system design. In this thesis the trajectory tracking task will be designed as a servomechanism problem as introduced in [2](Chapter 4). Servomechanism problem its suitable for reference tracking and provides a unique structure for further LQR and MPC design.

Structure of servomechanism problem

Let us consider the controled system as a Linear Time Invariant (LTI) discrete system given by state equations

$$\begin{aligned} x_1(t+1) &= A_1 x_1(t) + B_1 u(t) \\ y(t) &= C_1 x_1(t) + D_1 u(t), \end{aligned} \tag{6.21}$$

where $x_1 \in \mathbb{R}^{n_x}$ is the control system state vector, $u \in \mathbb{R}^{n_u}$ is the system input and $y \in \mathbb{R}^{n_y}$ is the output. Note, that (A_1,B_1) are considered stabilizable. Then let us consider a second LTI discrete system as a reference generator

$$\begin{aligned} x_2(t+1) &= A_2 x_2(t) \\ r(t) &= C_2 x_2(t), \end{aligned} \tag{6.22}$$

where $x_2 \in \mathbb{R}^{n_x}$ is the state of reference generator with observable (A_2,C_2) and $r \in \mathbb{R}^{n_y}$ is the generated reference. With the controlled system '6.21 and the reference generator 6.22 the tracking error could be introduced as

$$e(t) = r(t) - y(t). \tag{6.23}$$

Since we do not consider any feedthrough, the zero matrix D does not affect the error which could be the formalized as

$$e(t) = C_2 x_2 - C_1 x_1. \tag{6.24}$$

The extended servomechanism system could be then introduced as

$$\begin{aligned} x(t+1) &= \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix} x(t) + \begin{bmatrix} B1 \\ 0 \end{bmatrix} u(t) \\ e(t) &= \begin{bmatrix} -C_1 & C_2 \end{bmatrix} x(t) \end{aligned} \tag{6.25}$$

where,

$$x(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}. \tag{6.26}$$

The system 6.25 is the servomechanism structure, where independent parts of matrix A are split into controlled system and reference generator. Therefore the output of this

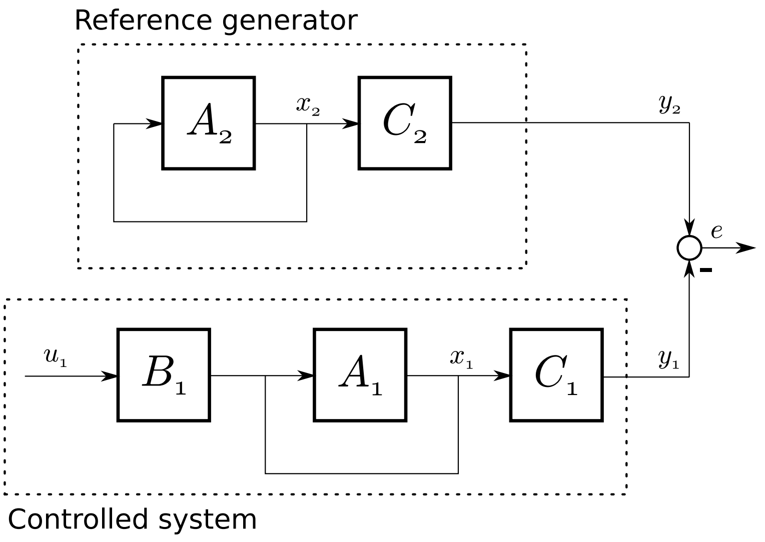


Figure 6.5: Servomechanism structure

system could be organized as a tracking error $e(t)$. Optimal minimization of this error lead to optimal trajectory tracking.

Structure with known disturbance

The previously introduced structure of servomechanism problem 6.25 could be extended of known disturbance d of reference signal. This disturbance change the structure of the sevomechanism problem to

$$\begin{aligned} x(t+1) &= \underbrace{\begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix}}_{\hat{A}} x(t) + \underbrace{\begin{bmatrix} B1 \\ 0 \end{bmatrix}}_{\hat{B}} u(t) + \underbrace{\begin{bmatrix} 0 \\ E_1 \end{bmatrix}}_{\hat{E}} d(k) \\ e(t) &= \underbrace{\begin{bmatrix} -C_1 & C_2 \end{bmatrix}}_{\hat{C}} x(t) \end{aligned} \tag{6.27}$$

where,

$$x(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}, \tag{6.28}$$

where matrix E_2 feed the disturbance into the reference generator system. This structure is giving an option to utilize future information about trajectory shape in MPC control design

Vehicle lateral model

The servomechanism structure considers that either controlled system and reference generator consist of states from which the output error $e(t)$ could be determined by substraction

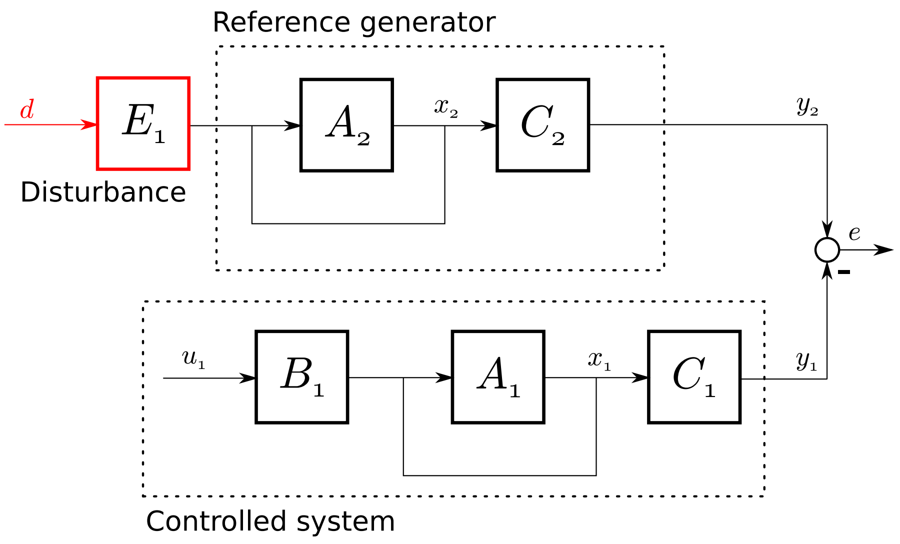


Figure 6.6: Servomechanism structure with known disturbance

of their outputs as outlined in Eq. 6.24. In this thesis the control system configuration use the error e_d as a distance of vehicle from the trajectory and error of the vehicle heading angle θ_e. Therefore the projection of those errors rates has to be included into the controlled system. The rate of crosstrack error with assumption of small heading error e_θ could be linearized as

$$\dot{e_d} = v_l(\theta - \theta_{ref}) = v_l e_\theta, \tag{6.29}$$

where v_l is the vehicle longitudinal velocity. As the non-zero error rate is caused by the vehicle motion at certain direction and change of trajectory reference, the error rate could be split into two parts

$$\dot{e_d} = -\dot{d_1} + \dot{d_2} = -\underbrace{(v_l\theta + v_l\frac{l_f}{L}\delta)}_{d_1} + \underbrace{v_l\theta_{ref}}_{d_2}. \tag{6.30}$$

The heading error of the vehicle e_θ evolves based on the steering angle δ, and reference heading angle is changing according the trajectory curvature. The result error rate could be then determined as

$$\dot{e_\theta} = -\frac{v_l}{L}\theta + v_l\theta_{ref}. \tag{6.31}$$

With this assumption the controlled system is formulated as

$$\begin{aligned}
\begin{bmatrix} \dot{d}_1 \\ \dot{\theta} \\ \dot{\delta} \end{bmatrix} &= \underbrace{\begin{bmatrix} 0 & v_l & v_l\frac{l_f}{L} \\ 0 & 0 & \frac{v_l}{L} \\ 0 & 0 & 0 \end{bmatrix}}_{A_1} \begin{bmatrix} d_1 \\ \theta \\ \delta \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}}_{B_1} u(t) \\
y(t) &= \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}}_{C_1}
\end{aligned} \tag{6.32}$$

an the reference generator is defined as

$$\begin{aligned}
\begin{bmatrix} \dot{d}_2 \\ \dot{\theta}_{ref} \\ \dot{\kappa}_{ref} \end{bmatrix} &= \underbrace{\begin{bmatrix} 0 & v_l & 0 \\ 0 & 0 & v_l \\ 0 & 0 & 0 \end{bmatrix}}_{A_2} \begin{bmatrix} d_1 \\ \theta \\ \delta \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}}_{E_2} u(t) \\
y(t) &= \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}}_{C_2}
\end{aligned} \tag{6.33}$$

The servomechanism problem with known disturbance is then modeled as Eq. 6.27

$$\begin{aligned}
\begin{bmatrix} \dot{d}_1 \\ \dot{\theta} \\ \dot{\delta} \\ \dot{d}_2 \\ \dot{\theta}_{ref} \\ \dot{\kappa}_{ref} \end{bmatrix} &= \underbrace{\begin{bmatrix} 0 & v_l & v_l\frac{l_f}{L} & 0 & 0 & 0 \\ 0 & 0 & \frac{v_l}{L} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & v_l & 0 \\ 0 & 0 & 0 & 0 & 0 & v_l \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_{\hat{A}} \underbrace{\begin{bmatrix} d_1 \\ \theta \\ \delta \\ d_2 \\ \theta_{ref} \\ \kappa_{ref} \end{bmatrix}}_{x(t)} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}}_{\hat{B}} u(t) + \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}}_{\hat{E}} d(t) \\
\begin{bmatrix} e_d \\ e_\theta \end{bmatrix} &= \underbrace{\begin{bmatrix} -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 \end{bmatrix}}_{\hat{C}} \begin{bmatrix} d_1 \\ \theta \\ \delta \\ d_2 \\ \theta_{ref} \\ \kappa_{ref} \end{bmatrix}
\end{aligned} \tag{6.34}$$

the whole system is then discretized with corresponding sampling period T_s

6.2.3 LQR

The first model-based control strategy used to perform trajectory tracking is Linear-Quadratic Regulator (LQR), which minimize the quadratic criterion function J on given horizon of length N

$$J = \frac{1}{2}x^T(N)Q'_N(N)x(N) + \frac{1}{2}\sum_{t=0}^{N-1} e^T(t)Q'e(t) + u^T(t)R'(t). \tag{6.35}$$

The Q_N and Q_e denote the positive semi-definite weighting matrices of the terminal cost and R is the positive definite weighting matrix of the control input. Since the controller does not perform the regulation task, the criterion function does not depend on the system state x. However, penalize the system output error $e(t)$. This task is called **Linear-Quadratic optimal servomechanism**.

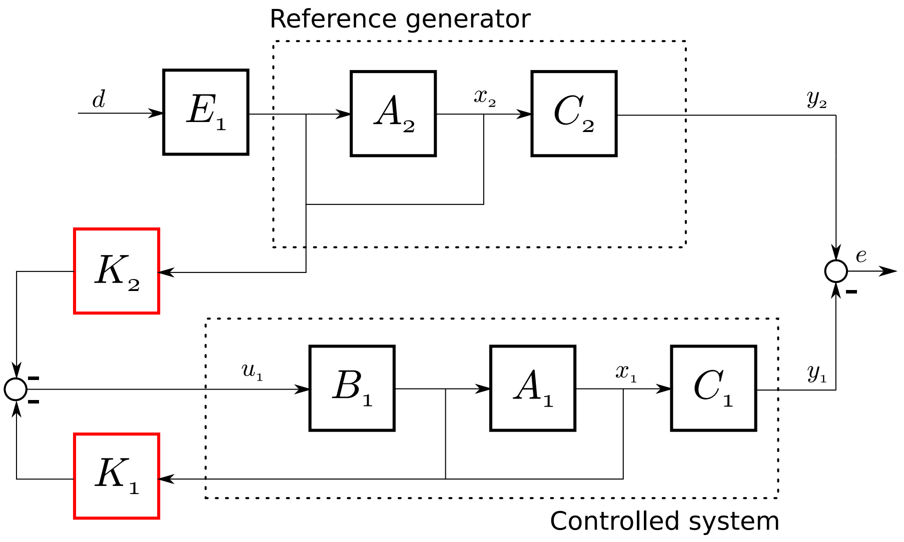


Figure 6.7: LQ optimal servomechanism structure with state feedback

The solution of LQ servomechanism is a state feedback obtain by *Joseph's stabilized Riccati equation*. The control law given by the solution could be written as

$$u(t) = -K_1(t)x_1(t) - K_2(t)x_2(t) \tag{6.36}$$

The matrices K_1 and K_2 are the result of iteration over N steps described in Algorithm 3. The tuning parameters of the controller are the weighting matrices R and Q_e which are used to compute initial matrices.

The action u is the resulting steering angle set to the vehicle servomotor.

Algorithm 3: LQ servomechanism

```
**Data:** $(A_1, B_1, C_1, D_1), (A_2, B_2, C_2, D_2), Q_e, R, N$
**Result:** $u$
$Q_1 \leftarrow C_1^T Q_e C_1$;
$Q_2 \leftarrow -C_1^T Q_e C_2$;
$P_1 \leftarrow Q_1$;
$P_2 \leftarrow Q_2$;
$k \leftarrow N$;
**while** $k > 0$ **do**
    $K_1(k-1) \leftarrow (R + B_1^T P_1(k) B_1)^{-1} B_1^T P_1(k) A_1$;
    $K_2(k-1) \leftarrow (R + B_1^T P_1(k) B_1)^{-1} B_1^T P_2(k) A_2$;
    $G \leftarrow A_1 - B_1 K_1(k-1)$;
    $P_1(k-1) \leftarrow G^T P_1(k)(G) + K_1^T(k-1) R K_1(k-1) Q_1$;
    $P_2(k-1) \leftarrow (G^T P_2(k) A_2 - P_1(k) B_1 K_2(k-1)) + K_1^T(k-1) R K_2(k-1) + Q_2$;
    $k \leftarrow (k-1)$;
$u \leftarrow -K_1(t) x_1(t) - K_2(t) x_2(t)$
```

6.2.4 MPC

The Model Predictive Control approach is minimizing the same criterion function as the LQ servomechanism. However, the result of the optimization is the open loop control sequence $\bar{u}$ minimizing the output error e on prediction horizon given by N_p samples. The feedback is obtained by applying only the first action input from the open loop sequence and performing the optimization again in the next time step. The benefit of MPC is, that optimization allows to extend the problem for nonlinear constrains such as maximal steering angle. The solution of MPC is described in detail in [2](Chapter 5) and the problem is stated as follows

$$\begin{aligned} &\min_{\bar{u},} \frac{1}{2} \bar{x}^T \bar{Q} \bar{x} + \bar{u}^T \bar{R} \bar{u} \\ &\text{s.t.} \bar{A}\bar{x} + \bar{B}\bar{u} = \bar{b}, \quad . \\ &\quad b_{iq} < A_{iq}[\bar{x}^T, \bar{u}^T] \end{aligned} \tag{6.37}$$

Let us write response predictions of dynamic system in following matrices

$$\bar{A} = \begin{bmatrix} -I & & & \\ \hat{A}_{n+1} & -I & & \\ & \hat{A}_{n+2} & -I & \\ & & \ddots & \ddots \end{bmatrix}, \quad \hat{B} = \begin{bmatrix} \hat{B}_n & & & \\ & \hat{B}_{n+1} & & \\ & & \hat{B}_{n+2} & \\ & & & \ddots \end{bmatrix} \tag{6.38}$$

$$\hat{b} = \begin{bmatrix} -\hat{A}_n x_n - \hat{E}_n d_n \\ -\hat{E}_{n+1} d_{n+1} \\ -\hat{E}_{n+1} d_{n+1} \\ \vdots \end{bmatrix}. \tag{6.39}$$

The criterion function given in 6.37 use the matrices Q and R

$$\bar{Q} = \begin{bmatrix} Q_{n+1} & & & \\ & Q_{n+2} & & \\ & & \ddots & \\ & & & Q_{n+N_p} \end{bmatrix}, \quad \bar{R} = \begin{bmatrix} R_n & & & \\ & R_{n+1} & & \\ & & \ddots & \\ & & & R_{n+N_p-1} \end{bmatrix} \tag{6.40}$$

, where

$$Q_n = \hat{C}^T Q_e \hat{C} \tag{6.41}$$

and R is a scalar, since the system has only one input u. The algorithm enter the current state vector x_n to a matrix $\bar{b}$ together with the disturbance preview sequence d, which serves as an optimization initial conditions. Optimization is also provided with vector

$$h = \begin{bmatrix} \delta_{l+1} \\ \delta_{l+2} \\ \vdots \\ \delta_{l+Np} \end{bmatrix}, \tag{6.42}$$

which use the constant δ_l to limit the steering action u. Since the problem has the nonlinear constrains, the problem has to be solved numerically. For that purpose the solver *cvxpy* was used. The optimization was done by following code.

```
import cvxpy as cv

u = cv.Variable((Np,1))
x = cv.Variable((Np*5,1))
obj = cv.Minimize((1 / 2) *(cv.quad_form(x, Q_bar) + cv.quad_form(u,R_bar)))
prob = cv.Problem(obj,[A_bar*x + B_bar*u == b_bar, u <= h, -h <= u])
prob.solve()
```

The result of the optimization is the open loop sequence of control input δ as shown in Fig. 6.8 From the sequence $\bar{u}$, the first input is taken and considered as control action.

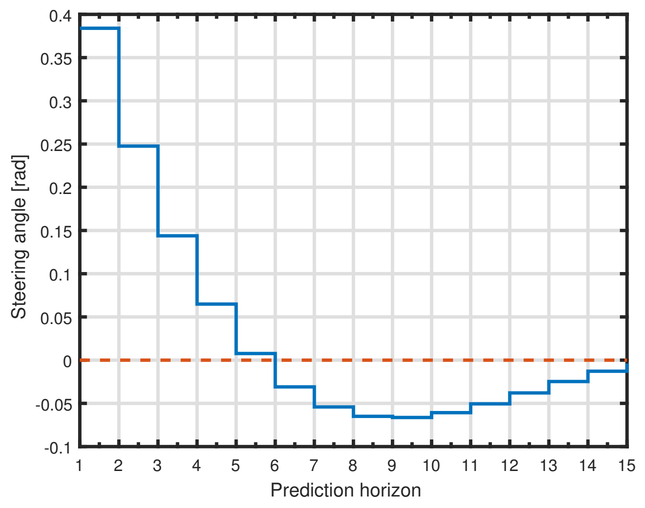


Figure 6.8: Result of optimization - open loop control sequence $\bar{u}$

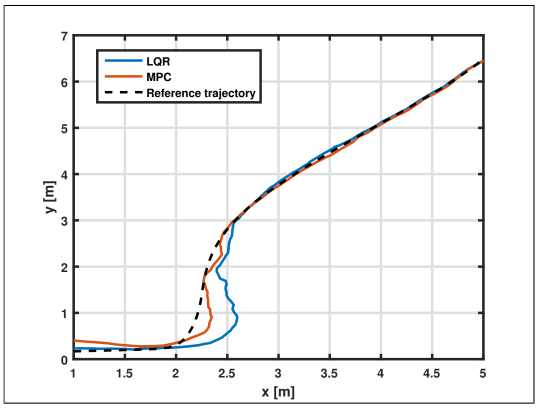
6.2.5 Experiments

Several experiments have been made to compare the performance of LQR and MPC lateral controller. Since both controllers use same criterion function, the weighting matrices Q and R did not differ and were set as

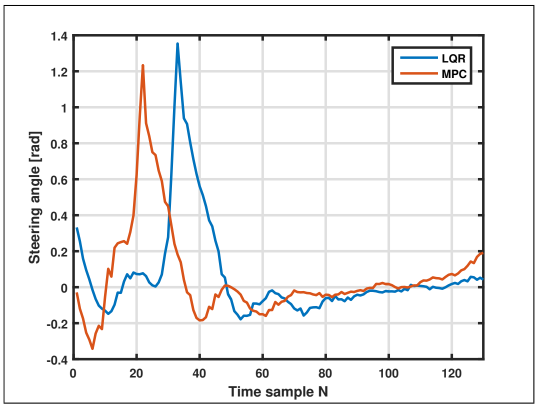
$$Q_e = \begin{bmatrix} 1 & 0 \\ 0 & 0.1 \end{bmatrix} \qquad R = \begin{bmatrix} 10 \end{bmatrix}. \tag{6.43}$$

This configuration emphasize the error e_d and adjustment of R then set the aggressivity of control. The result of control is shown in Fig. 6.9

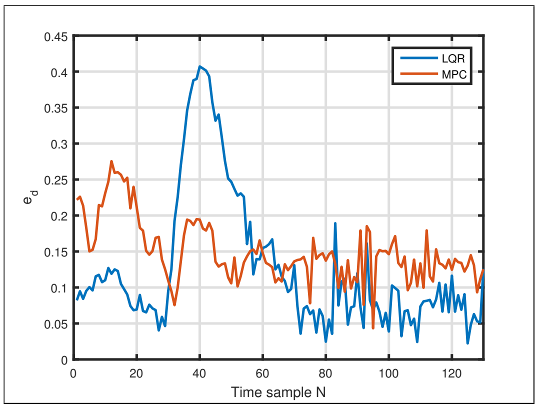
From the Figure could be seen, that both controllers have good performance on straight sections, however in the sharp turn the MPC is reacting more quickly due to trajectory preview, which brings the information about upcoming turn in advance. Therefore the steering action is performed sooner and cornering is handled with smaller overshoot. The main disadvantage of MPC is the necessity of a numerical solver, which performance depends on the solver implementation together with the used length of prediction horizon. In conducted experiments the solver was set to perform optimization over prediction horizon of 20 steps (1 second), which limits the control loop to $20Hz$.



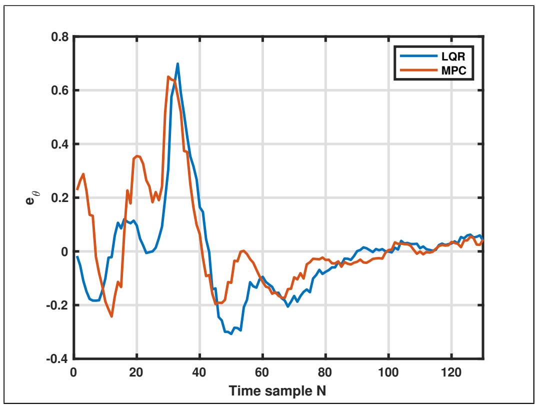
(a)



(b)



(c)



(d)

Figure 6.9: Experiments of trajectory tracking with LQR and MPC

Chapter 7

Conclusion

This thesis successfully deals with the design of a system able to localize the vehicle model on the racing track and develop a control system performing the trajectory tracking. In Chapter 3 thesis deals with the design of a mapping system able to create a planar map of a track from sensor data collected during the manual drove trial lap. The task was formulated as a 2D SLAM and the successful mapping was achieved with a Hector SLAM algorithm. Hector SLAM utilizes the scan-matching methods to perform environment exploration and map construction from LiDAR scans. With the knowledge of the map, the task of vehicle localization was done in Chapter 4 by Monte Carlo Localization (MCL) method. The MCL utilize the LiDAR scans, wheel odometry and probabilistic approach of Markov Localization to perform the best position estimation. The integrated localization was tested in several scenarios and further extended of Relative pose estimator and Extended Kalman Filter (EKF). Relative pose estimator fuse the MCL estimation with data from wheel odometry to increase the rate of vehicle pose estimation. The EKF, on the other hand, use the nonlinear kinematic equations of vehicle, to predict the vehicle motion and filter the position in case of noisy data from MCL. The last part of the work is dealing with the design of the control system performing the trajectory tracking task with LQ optimal servomechanism structure. The goal of this part was to integrate the already examined methods introduced in [2] and verify their performance on real hardware. For that purpose, the LQR and MPC controller were implemented and tested with the usage of developed mapping and localization system.

7.1 Future work

7.1.1 Optimal racing line planning

This thesis introduced only simple techniques of trajectory planning on the racing track for testing purposes. However, the task of racing relies on planning a trajectory, which is optimal in the sense of lap time. Such a trajectory has to consider the vehicle kinematics together with vehicle limits and surface adhesion. The future work is to implement an algorithm able to plan the optimal racing line on the created map of the track with the use of nonlinear programming as is introduced in [25].

7.1.2 Reactive and map-based algorithm fusion

The map-based approach of vehicle control on the racing track has the advantage of complex knowledge used for vehicle trajectory planning. However, struggles when the track change the layout or localization provides misleading data. The future work is to fuse the map-based approach with reactive algorithms to create a robust system able to perform high-level planning together with agile obstacle avoidance.

Bibliography

[1] M. Vajnar, "Model car for the f1/10 autonomous car racing competition", PhD thesis, Jun. 2017.

[2] J. Filip, "Trajectory tracking for autonomous vehicles", PhD thesis, May 2018. DOI: 10.13140/RG.2.2.36089.93288.

[3] D. A.R. P. Agency. (2004). Grand challenge 2004 final report, [Online]. Available: https://www.esd.whs.mil/Portals/54/Documents/FOID/Reading%20Room/DARPA/15-F-0059_GC_2004_FINAL_RPT_7-30-2004.pdf (visited on 05/15/2019).

[4] R. Schedel, "Darpa urban challenge 2007", *ATZ worldwide*, vol. 110, no. 1, pp. 10–12, 2008, ISSN: 2192-9076. DOI: 10.1007/BF03224975. [Online]. Available: https://doi.org/10.1007/BF03224975.

[5] (2019). F1tenth racing platform build manual, [Online]. Available: http://f1tenth.org/build.html (visited on 05/15/2019).

[6] J.-S. Zhao, Z.-J. Feng Xiang Liu, and J. Dai, "Design of an ackermann type steering mechanism", *Journal of Mechanical Engineering Science*, vol. 227, Nov. 2013. DOI: 10.1177/0954406213475980.

[7] M. A.N.P. B. Prof Mr. Aniket Kolekar Mr. Sumir Mulani, "Review on steering mechanism", *International Journal for Science and Advance Research In Technology*, vol. 3, Apr. 2017.

[8] M. Demir and V. Sezer, "Improved follow the gap method for obstacle avoidance", in *2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, 2017, pp. 1435–1440. DOI: 10.1109/AIM.2017.8014220.

[9] T. Chong, X. Tang, C. Leng, M. Yogeswaran, O. Ng, and Y. Chong, "Sensor technologies and simultaneous localization and mapping (slam)", *Procedia Computer Science*, vol. 76, pp. 174 –179, 2015, 2015 IEEE International Symposium on Robotics and Intelligent Sensors (IEEE IRIS2015), ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2015.12.336. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050915038375.

[10] Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces", *Int. J. Comput. Vision*, vol. 13, no. 2, pp. 119–152, Oct. 1994, ISSN: 0920-5691. DOI: 10.1007/BF01427149. [Online]. Available: http://dx.doi.org/10.1007/BF01427149.

[11] A. Diosi and L. Kleeman, "Fast laser scan matching using polar coordinates", *The International Journal of Robotics Research*, vol. 26, no. 10, pp. 1125–1153, 2007. DOI: 10.1177/0278364907082042. eprint: https://doi.org/10.1177/0278364907082042. [Online]. Available: https://doi.org/10.1177/0278364907082042.

[12] S. Kohlbrecher, O. von Stryk, J. Meyer, and U. Klingauf, "A flexible and scalable slam system with full 3d motion estimation", in *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, 2011, pp. 155–160. DOI: `10.1109/SSRR.2011.6106777`.

[13] (2014). Hector slam ros package documentation, [Online]. Available: `http://wiki.ros.org/hector_slam` (visited on 05/15/2019).

[14] S. Malagon-Soldara, M. Toledano-Ayala, G. Soto-Zarazua, R. Carrillo-Serrano, and E. Rivas-Araiza, "Mobile robot localization: A review of probabilistic map-based techniques", *IAES International Journal of Robotics and Automation (IJRA)*, vol. 4, Mar. 2015. DOI: `10.11591/ijra.v4i1.6548`.

[15] L. Teslić, I. Škrjanc, and G. Klančar, "Ekf-based localization of a wheeled mobile robot in structured environments", *Journal of Intelligent & Robotic Systems*, vol. 62, no. 2, pp. 187–203, 2011, ISSN: 1573-0409. DOI: `10.1007/s10846-010-9441-8`. [Online]. Available: `https://doi.org/10.1007/s10846-010-9441-8`.

[16] S. Panzieri, F. Pascucci, and R. Setola, "Multirobot localisation using interlaced extended kalman filter", in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 2816–2821. DOI: `10.1109/IROS.2006.282065`.

[17] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte carlo localization: Efficient position estimation for mobile robots", Jan. 1999, pp. 343–349.

[18] J. Bresenham, "Algorithm for computer control of a digital plotter", *IBM Systems Journal*, vol. 4, pp. 25–30, 1965.

[19] M. N. Gamito and S. C. Maddock, "Ray casting implicit fractal surfaces with reduced affine arithmetic", *The Visual Computer*, vol. 23, no. 3, pp. 155–165, 2007, ISSN: 1432-2315. DOI: `10.1007/s00371-006-0090-7`. [Online]. Available: `https://doi.org/10.1007/s00371-006-0090-7`.

[20] C. H. Walsh and S. Karaman, "Cddt: Fast approximate 2d ray casting for accelerated localization", May 2018, pp. 1–8. DOI: `10.1109/ICRA.2018.8460743`.

[21] V. Cossalter, M Da Lio, R. Lot, and L. Fabbri, "A general method for the evaluation of vehicle manoeuvrability with special emphasis on motorcycles", *Vehicle System Dynamics - VEH SYST DYN*, vol. 31, pp. 113–135, Feb. 1999. DOI: `10.1076/vesd.31.2.113.2094`.

[22] L. Cardamone, D. Loiacono, P. L. Lanzi, and A. P. Bardelli, "Searching for the optimal racing line using genetic algorithms", Sep. 2010, pp. 388 –394. DOI: `10.1109/ITW.2010.5593330`.

[23] M. Bevilacqua, A. Tsourdos, and A. Starr, "Particle swarm for path planning in a racing circuit simulation", May 2017, pp. 1–6. DOI: `10.1109/I2MTC.2017.7969735`.

[24] M. Botta, V. Gautieri, D. Loiacono, and P. L. Lanzi, "Evolving the optimal racing line in a high-end racing game", in *2012 IEEE Conference on Computational Intelligence and Games (CIG)*, 2012, pp. 108–115. DOI: `10.1109/CIG.2012.6374145`.

[25] G. Perantoni and D. J. Limebeer, "Optimal control for a formula one car with variable parameters", *Vehicle System Dynamics*, vol. 52, no. 5, pp. 653–678, 2014. DOI: `10.1080/00423114.2014.889315`. eprint: `https://doi.org/10.1080/00423114.2014.889315`. [Online]. Available: `https://doi.org/10.1080/00423114.2014.889315`.

[26] J. Subosits and J. C. Gerdes, "Autonomous vehicle control for emergency maneuvers: The effect of topography", in *2015 American Control Conference (ACC)*, 2015, pp. 1405–1410. DOI: `10.1109/ACC.2015.7170930`.

[27] G. Karer, M. Kolbe, J. Leskovec, V. Logar, and P. Drago Matko, "Robot ballet", May 2019.

[28] J Rounsaville, J Dvorak, and T. Stombaugh, "Methods for calculating relative cross-track error for asabe/iso standard 12188-2 from discrete measurements", *Transactions of the ASABE*, vol. 59, pp. 1609–1616, Dec. 2016. DOI: `10.13031/trans.59.11902`.

[29] J.-B. Park, S.-H. Bae, B.-S. Koo, and J.-H. Kim, "When path tracking using look-ahead distance about the lateral error method and the velocity change method tracking comparison", Oct. 2014, pp. 1643–1647. DOI: `10.1109/ICCAS.2014.6987822`.

[30] D.-h. Kim, C.-S. Han, and j. Lee, "Sensor-based motion planning for path tracking and obstacle avoidance of robotic vehicles with nonholonomic constraints", *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 227, pp. 178–191, Jan. 2013. DOI: `10.1177/0954406212446900`.