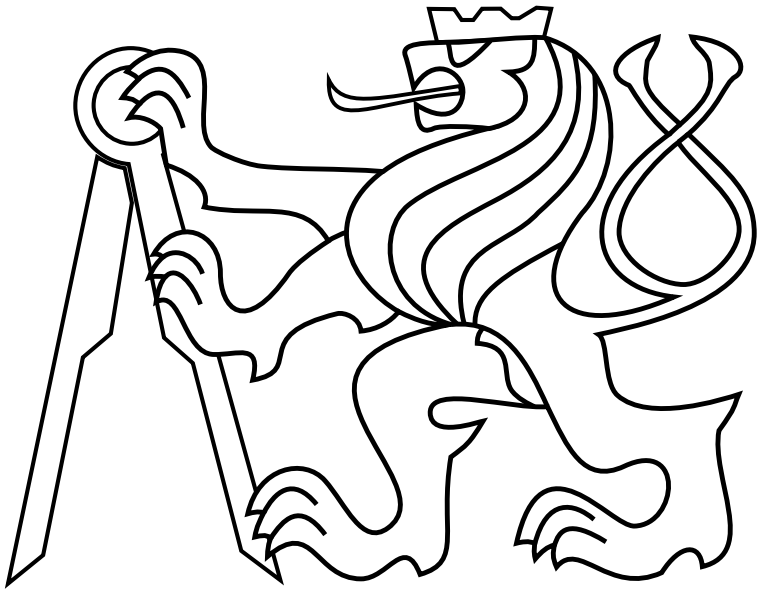
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

MASTER'S THESIS



Zdeněk Berka

Mapping of Road Marking and Localization of F1/10 Autonomous Car Using a Camera

Department of Cybernetics

Thesis supervisor: **Ing. Jaroslav Klapálek**

Prague, August 2020

MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Berka Zdeněk** Personal ID number: **456881**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Open Informatics**

Specialisation: **Computer Vision and Image Processing**

II. Master's thesis details

Master's thesis title in English:

Mapping of Road Marking and Localization of F1/10 Autonomous Car Using a Camera

Master's thesis title in Czech:

Mapování vodorovného značení a lokalizace autonomního auta F1/10 pomocí kamery

Guidelines:

1. Make yourself familiar with ROS and F1/10 autonomous racing cars.
2. Review methods of recognition and mapping of road marking.
3. Design and implement an algorithm for line detection (road marking) using a camera.
4. Create an algorithm for mapping of road marking and for localization in a created map. For SLAM method use existing solution (e.g. Google Cartographer), which may be enhanced by custom implementation.
5. Test the algorithm on F1/10 car. Discuss reliability of the solution, accuracy of the localization and identify boundaries of implemented algorithm (e.g. limits on vehicle speed, steering, track complexity).
6. Document everything thoroughly.

Bibliography / sources:

[1] M. Quigley et al., 'ROS: an open-source Robot Operating System', presented at the ICRA Workshop on Open Source Software, 2009, vol. 3.
[2] G. Papari and N. Petkov, 'Edge and line oriented contour detection: State of the art', Image and Vision Computing, vol. 29, no. 2, pp. 79–103, Feb. 2011, doi: 10.1016/j.imavis.2010.08.009.
[3] W. Hess, D. Kohler, H. Rapp, and D. Andor, 'Real-time loop closure in 2D LIDAR SLAM', in 2016 IEEE International Conference on Robotics and Automation (ICRA), 2016, pp. 1271–1278, doi: 10.1109/ICRA.2016.7487258.

Name and workplace of master's thesis supervisor:

Ing. Jaroslav Klapálek, Department of Control Engineering, FEE

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

Date of master's thesis assignment: **05.02.2020** Deadline for master's thesis submission: **14.08.2020**

Assignment valid until: **30.09.2021**

Ing. Jaroslav Klapálek Supervisor's signature	doc. Ing. Tomáš Svoboda, Ph.D. Head of department's signature	prof. Mgr. Petr Páta, Ph.D. 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 declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, 14. 8. 2020 Zdeněk Berka

Acknowledgements

Foremost, I would like to thank my supervisor Ing. Jaroslav Klapálek for his guidance and especially for his help with managing the systems on the F1/10 car whenever I hit an obstacle and needed advice. I would also like to thank everyone who supported me during my studies and while I was writing my master's thesis.

Abstract

This thesis focuses on building a navigation system for an autonomous vehicle in an unknown environment. We use a single monocular camera as a primary sensor input. We build a system for detection of road lane markings from individual camera images. We then use these detected markings as landmarks for building a map fragment of the currently perceived surroundings. To create a global map and localise within it, we use an existing 2D LIDAR-based SLAM solution (*Cartographer* [26]) which employs modern scan matching and loop closure algorithms. We build the whole system for Robotic Operating System (ROS) and verify the solution on various experimental tracks using a scaled-down model of a car called *F1/10*. In the scenario with low speed and low track complexity, the localisation shows accurate results (mean position error of 0.14 m) and loop closure manages to build an accurate and consistent map. However, the results show that the performance quickly deteriorates with a higher speed and a higher track complexity. We suggest possible causes and solutions.

Keywords: Road Detection, Simultaneous localisation and mapping, Visual navigation, Autonomous vehicles

Abstrakt

V práci se zabýváme vybudováním systému, který umožní orientaci autonomního auta v neznámém prostředí. Jako primární senzor pro tuto úlohu využíváme kameru. Náš systém detekuje vodorovné značení vozovky z jednotlivých obrázků, které posílá kamera při jízdě auta. Detekované značení používáme jako orientační body pro vybudování fragmentu mapy, který odpovídá aktuálně pozorovanému prostředí. Pomocí existujícího řešení 2D SLAMu (*Cartographer [26]*; určené pro řešení SLAMu s použitím LIDARu) náš systém vytváří globální mapu z nahromaděných lokálních fragmentů a následně v ní lokalizuje aktuální pozici auta. 2D SLAM používá moderní algoritmy párování skenů a uzavírání smyček k tvorbě konzistentní globální mapy. Celý systém je implementován pro využití v ROSu (Robotický Operační Systém). Řešení problému ověřujeme pomocí experimentů, které realizujeme na rozličných trasách se zmenšeným modelem auta *F1/10*. Při nízké rychlosti vozidla na jednoduché trase vykazuje lokalizace velmi přesné výsledky, konkrétně chyba pozice je v průměru 0.14 m, avšak se stoupající rychlostí auta a složitostí trasy přesnost systému rychle upadá. V této práci zkoumáme možné příčiny těchto problémů a navrhujeme jejich řešení.

Klíčová slova: Detekce vozovky, Současné mapování a lokalizace, Vizuální navigace, Autonomní vozidla

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The technology of autonomous cars has become a larger and larger target of interest for both the professional and general public in recent years. Even though people have tried to invent self-driving cars since the 1920s [4], in recent years, we can see this idea come to life. Nowadays, we can finally see the first few almost fully autonomous cars. Although, back-up drivers are still required to switch to manual driving mode in certain situations[14]. There is still a lot of improvement and testing to be done and also a lot of questions of non-technological character to be resolved [31] before autonomous cars can become common on the roads. Despite that, autonomous cars are predicted to become a majority on the roads by 2035 [4].

The goal of this thesis is the development and testing of a system that provides navigation to a vehicle in an unknown environment. Our system mainly depends on data from a camera mounted on the vehicle. We also explore possible enhancements by using auxiliary sensor data, e.g., an acceleration estimate provided by the IMU (Inertial Measurement Unit) and the depth landmarks provided by LIDAR. Our environment is a model of an urban road defined by lane markings. For the development and testing of the algorithms, we use a scaled-down model of a car called *F1/10*.

Navigation is achieved in three steps. First of all, our system detects lane markings in the individual camera images. Secondly, we use these detected lane markings as landmarks and build a local map fragment corresponding to the currently perceived area around the vehicle. In the last step, we provide the local map fragments in the form of LIDAR scans to an existing 2D LIDAR-based SLAM (Simultaneous Localization and Mapping) solution to build a global map and localise within it. For this purpose, we use a modern SLAM solution called *Cartographer* [26]. We chose Cartographer, because it is an effective real-time solution, which achieves better results than other popular solutions (e.g., Gmapping [23] and Hector SLAM [30]) as shown in this paper [52], which compares the accuracy of widely used 2D LIDAR-based solutions.

This thesis was inspired by an already working LIDAR-based SLAM system on the F1/10 car mentioned above. That system enabled the car to navigate itself on a track enclosed by walls, which are directly detected by LIDAR. Achieving functional SLAM for a specific robot and a specific type of an environment is extremely valuable as it allows the robot to navigate the surrounding environment autonomously [17]. SLAM is usually achieved using wide-angle range sensors like LIDAR [35]. As already mentioned, this work instead focuses on using a single monocular camera.

Our goal is to achieve precise localization and mapping of the autonomous car in an

environment defined by lane markings. This enables similar functionality as LIDAR-based SLAM but instead of mapping obstacles we map the lane markings detected from the camera images. This provides the car with navigation, which can be used for autonomous applications like autonomous driving, racing, etc., on a track similar to urban roads.

The structure of this thesis is as follows:

First, the established state-of-the-art perception methods related to our task are described and discussed in Chapter 2. Further, the assigned problem is formulated and specification of available resources for its solution is described in Chapter 3. In Chapter 4, we present and discuss the methods chosen for the solution in this thesis. Chapter 5 focuses on a brief overview of the implementation of those methods. This is followed by Chapter 6, which describes the metrics used for the evaluation and verification of the implemented solution. Finally, the experiments and their results are presented in Chapter 7 and the whole thesis is concluded in Chapter 8.

The contents of the attached CD are listed in Table 1 in the appendix. The abbreviations used throughout this thesis are listed in Table 2 in the appendix.

Chapter 2

Related Works

In this chapter, we first put the assigned task into the context of previous research work and review the state-of-the-art methods that are used for solving similar problems to those encountered. This thesis's assignment can be broken down into three subtasks: *road detection*, *local map fragment construction* and *SLAM*. The discussed methods and principles related to each subtask are presented below in separate sections.

2.1 Road Detection

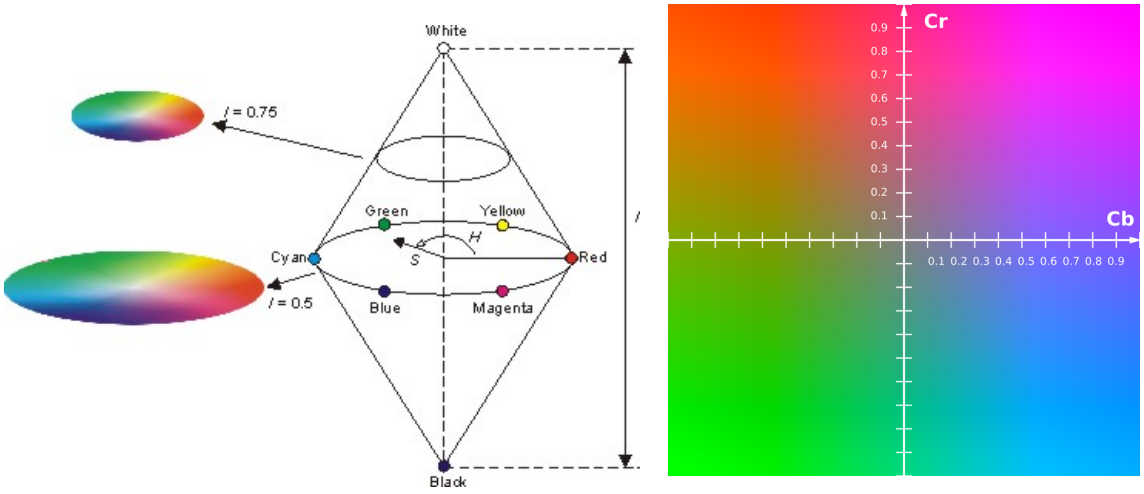
The first part of our task is the detection of the road from an independent frame taken by the camera. The detection of the road defined by clearly visible lane markings is mostly referred to as *structured road detection* or *detection of a road in urban areas* in literature. Methods used for detecting this type of road differ from methods for detecting an unmarked road or a badly marked road. In the former case, the perception applications heavily rely on detecting lane markings and using them as landmarks for navigating through the environment. Road detection is then reduced to the detection of lane markings. In the latter case, the task is much more complex. The road can be detected by dynamically estimating the color, texture and features of the road surface and trying to expand that region until it reaches discontinuities in the image that would signify naturally created road boundaries [50].

A lot of research was conducted on the topic of lane detection from a camera image in different circumstances. Most of the established methods use a combination of color-based segmentation [10, 25, 47], edge detection [46, 28, 51] and geometrical constraints [46, 54, 51] using a model of the lanes. These methods can independently vote on the classification of individual pixels or they can restrict image areas that are then used for the final segmentation by other methods. The specific methods and principles of the lane detection are introduced below.

2.1.1 Color-based Segmentation

When the lane color is distinct from the road, detection algorithms can use it to preprocess the image using color-based segmentation [10, 47, 25, 46]. This can be done in a standard RGB color space but it is not that efficient for the characterization of the lane color [46]. Other color spaces that separate components like luminance and chrominance into separate channels help reduce the effect of current light conditions [46]. Most of the reviewed relevant papers [47, 46, 32] use the HSI space (Hue, Saturation and Intensity) or the YC_bC_r space

(Luminance, blue and red chroma components). These alternative color spaces are illustrated in Figure 2.1.



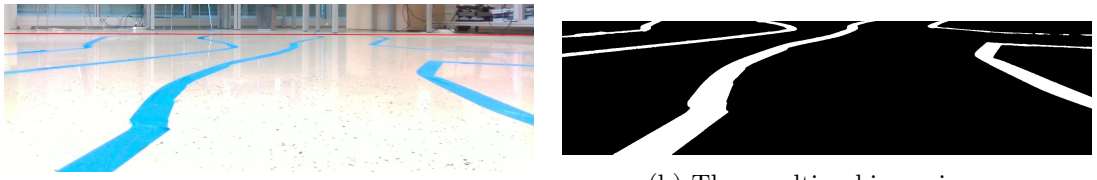
(a) HSI color space (from [5])

(b) YC_bC_r for luminance of 0.5 (from [44])

Figure 2.1: Alternative color spaces illustrations

For the segmentation of a color in regular RGB space, we need to threshold all three channels as the color information is evenly distributed among the components. In the alternative color spaces, it is possible to rely on thresholding only one or two components [47]. This saves us computation time.

The output of the color-based segmentation is a binary image that can be used to restrict areas for further analysis. An example of color segmentation of the lane markings is shown in Figure 2.2.



(a) The original image

(b) The resulting binary image

Figure 2.2: Color-based segmentation by thresholding the hue component in the HSI space example

2.1.2 Edge Detection

Edge detection is another popular approach used for the detection of lane markings in an image (as shown in, e.g., [46] and [54]). Edge detection is the process of finding continuous parts of an image where pixel intensity suddenly changes between consecutive pixels. This approach is based on the assumption that we expect high gradient values between the lane

markings and the road in the image [46]. Points that are classified as edges are good candidates to be a part of the lane markings boundary.

There are several ways to extract edges from an image. According to the survey [3] the most established ones are Canny edge detector [9] and Marr-Hildreth detector [36]. The methods of road detection [46, 54] use Canny edge detector applied to a preprocessed image. Preprocessing usually includes smoothing by convolving the image with a Gaussian kernel (Equation 2.1) to suppress edges stemming from the input noise:

$$G_{norm}(x,y) = \frac{G(x,y)}{\sum_{i=0}^{n}\sum_{j=0}^{m}G(i,j)}, \tag{2.1}$$

$$G(x,y) = e^{-\frac{(x-\frac{n-1}{2})^2+(y-\frac{m-1}{2})^2}{2\sigma^2}}, \tag{2.2}$$

where n, m are desired sizes of the kernel and σ is the standard deviation of the gaussian distribution.

The Canny edge detector uses thresholding with hysteresis to keep even the weak edge points (edge points with a lower intensity gradient) in the result as long as they are connected to strong edges (points with a high intensity gradient). This results in more continuous and robust edge detection, which can be adjusted for any application by changing the ratio between the low and the high threshold. The edge detection of Figure 2.2a can be seen in Figure 2.3. Apart from the lane markings, there are detected edges from the black granules in the ground as well as from the reflections of objects above the ground plane.

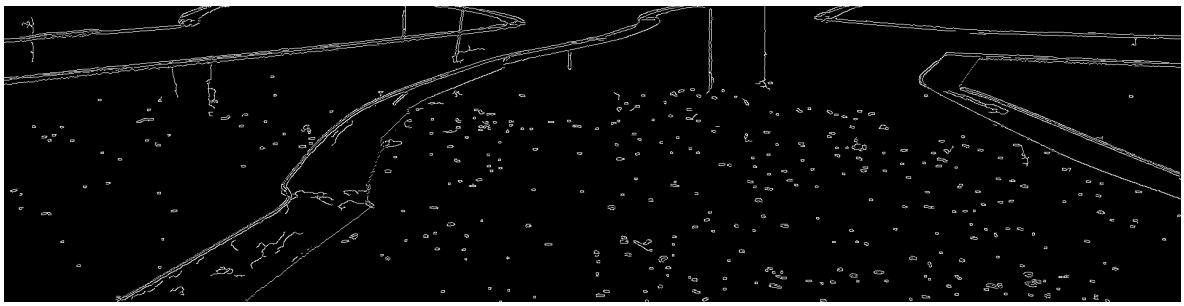


Figure 2.3: Canny edge detection example

2.1.3 Geometrical Model Fitting

When it comes to the detection of parallel lane markings on a road of a simple shape (without intersections or more complex turns, etc.), it is possible to use a parametrised geometrical model. The model can be searched for in the preprocessed image that already contains the majority of points that are likely to be lane markings (e.g., the result of edge detection or segmentation).

Some methods use a simple linear model (Equation 2.3) [28]. The linear model is very restrictive and cannot deal with higher curvature of the lane markings. It can be useful for certain applications if used in the near field of view where the road is expected to be almost

straight as shown in [28].

$$y = ax + b \tag{2.3}$$

A parabolic arc model (Equation 2.4) of the lane markings is often used [28, 54]. A circular arc model (Equation 2.5) is also useful as it can deal with the curved road [25], yet it remains simple with only 3 parameters.

$$y = ax^2 + bx + c \tag{2.4}$$

$$(x - a)^2 + (y - b)^2 = c^2 \tag{2.5}$$

Other more advanced models, e.g., cubic B-splines, have shown good results. Splines can deal with more than one curve in a single frame (e.g., an S-shaped road), which the previously presented models could not handle [51]. Using the splines, the lane markings are represented by curve segments l where each segment can be defined as a linear combination (Equation 2.6) of control points C_i [51]:

$$l(t) = \sum_i S_i(t)C_i, \tag{2.6}$$

where S_i is the spline basis function. We can then increase the complexity and flexibility of the model by adding more control points.

To fit the model to the image, the Hough transform is usually used [28, 46, 51, 54]. At first, the Hough transform enumerates all possible instances of the model. Then, it lets individual pixels vote for those instances of the model that the pixel could be a part of. Compared to the other methods for extracting the model from noisy data like RANSAC[6], the Hough transform does not return just the best-found candidate, but also a score of all instances. We can then process the results to find all instances of the model that are relevant to our task. It is computationally expensive and memory expensive, especially for the more complex models where the number of parameters is high. On the other hand, certain improvements can be done to save enough computational time to make it work even for real-time applications, e.g., the multiresolution Hough transform used in [54] or the Probabilistic Hough transform described in [22].

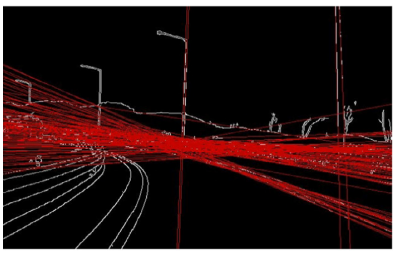
This approach is powerful when lane markings hold the assumed shape. Its great advantage in comparison to segmentation methods is that it can interpolate lane markings in cases where they are only partially visible or when they are missing completely. Geometrically constraining the road detection works well for the purpose of lane following, a lane departure warning system, etc.[28, 46, 51].

The need for parametrisation is a certain disadvantage of the method. It is not suitable for the detection of arbitrary road shapes (e.g., non-parallel lines or intersections) as the computational and memory cost significantly rises with the complexity of the model. For applications like mapping, it would require the addition of a mechanism for finding the boundaries of the modelled curves as the models naturally extend to infinity.

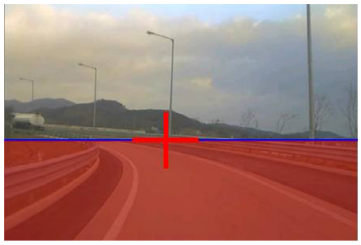
2.1.4 Region of Interest

If there is a large area of the image, which does not contain any objects of interest, the lane detection methods first cut the input image into 2 parts based on the horizon line to extract the ROI (Region of Interest) [10, 46]. There are two ways of ROI estimation. We may assume the ROI significantly changes between frames (e.g., due to vertical camera rotation caused by the car tilting) or we assume that the ROI stays the same because the relative camera pose is stable. In the former case, the extraction has to be done independently in each frame by searching for the vanishing point, e.g., using Hough line transform as proposed in [46]. This dynamic ROI search is illustrated in Figure 2.4. It is computationally expensive to do this independently in every frame during real-time application. In the latter case, we only estimate the ROI once manually and cut it out of each input frame [10] the same way.

(a) Original image of structured road scene



(b) Hough line transform applied on the detected edges from the original image



(c) Detected vanishing point and generated boundary line between the road area and the rest of the image

Figure 2.4: Dynamic ROI search using vanishing point detection (from [46])

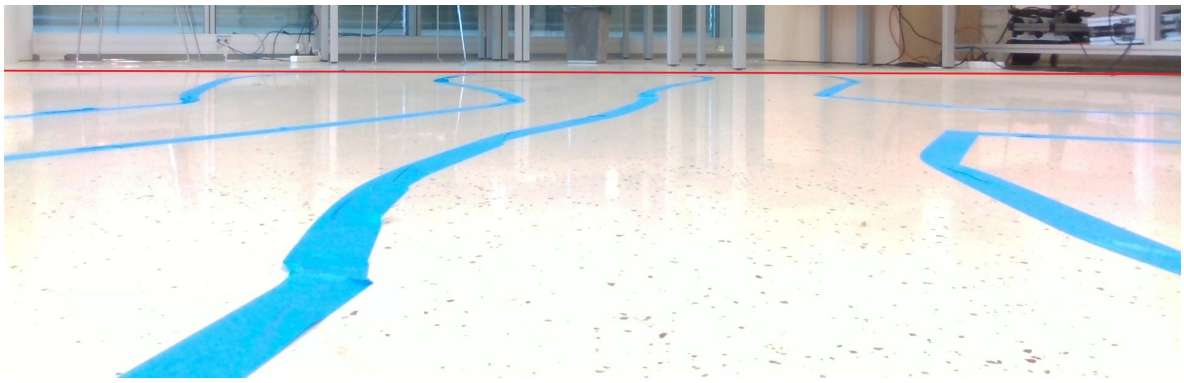


Figure 2.5: Static ROI illustration

The typical approach to get the ROI is to cut out the part of the image above the ground plane, as it contains no objects of interest. Other than that, we can also cut out the part of the ground plane that is too close to the horizon line. We can choose to do this because that part of the image is blurred and the lane markings are barely visible as illustrated in Figure 2.5. The further we look from the camera center on the ground plane, the thinner the lane markings are as there are fewer pixels per unit of length. Therefore, only the part of the image below the red line is the ROI. Above it, the lane markings cannot be accurately

detected. Even further above, the ground plane vanishes. These effects occur due to the nature of perspective projection, which causes the size of further objects to be scaled down and condensed into a gradually smaller amount of pixels in the image. Perspective projection is illustrated in Figure 2.6.

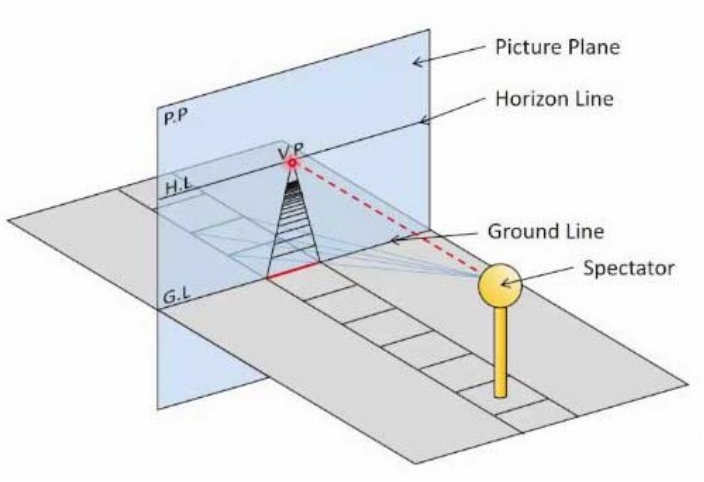


Figure 2.6: Perspective view of a plane illustration (from [12])

Jung and Kelber use the ROI approach even further in their application of lane following [28]. They propose a method of generating the ROI from the previously detected lane boundaries and processing only that part of the image to save the computational time. This can be done by extending the previously detected regions with lane markings by few pixels in each direction.

The benefit of using the ROI is that only the important part of the image is processed. This approach can save a lot of computation time.We can also take advantage of needing only that part of the image from the camera and increase the FPS by omitting reading and transferring the rest of the image (if the provided camera supports such an option). These benefits are only available if our assumptions regarding the image areas are accurate. Otherwise, we could lose valuable features.

2.2 Map Fragment Acquisition

After image processing, we have to use the detected lane markings from the independent frames to build a local map fragment of the visible surrounding environment. These map fragments can be later used for the construction of a global map, which is used for localisation.

To acquire a fragment of the map, we need to transform the coordinates of the detected points in the image plane to a coordinate system of the ground plane. The origin of this coordinate system of the ground plane moves with the car. The relation between the coordinate systems of the image and the ground plane is usually formulated by introducing a virtual camera with a birds eye view [41]. This virtual camera includes the desired ground plane coordinate system, as illustrated in Figure 2.7.

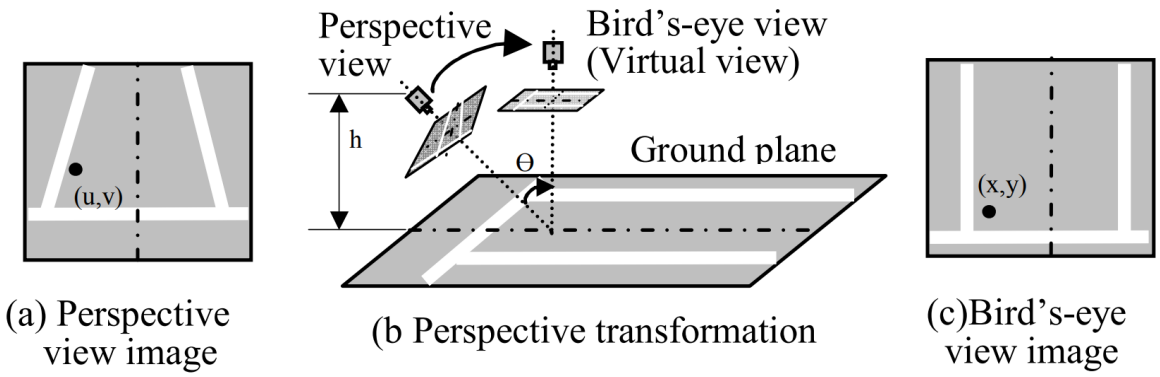


Figure 2.7: Perspective transformation illustration (from [34])

Then, we have to find the transformation between the two cameras. To derive the transformation between these coordinate systems a mathematical model (Section 2.2.1) of the cameras needs to be formulated. We then estimate the model's parameters by calibration. Doing that provides us with projection formulas defining the relation between the 3D scene and both image planes. A widely used camera model is described in the next section. We also mention the standard calibration methods. Further, in Section 2.2.2, the methods for deriving the geometrical transformation (linking the image to the ground plane) in the form of homography are presented.

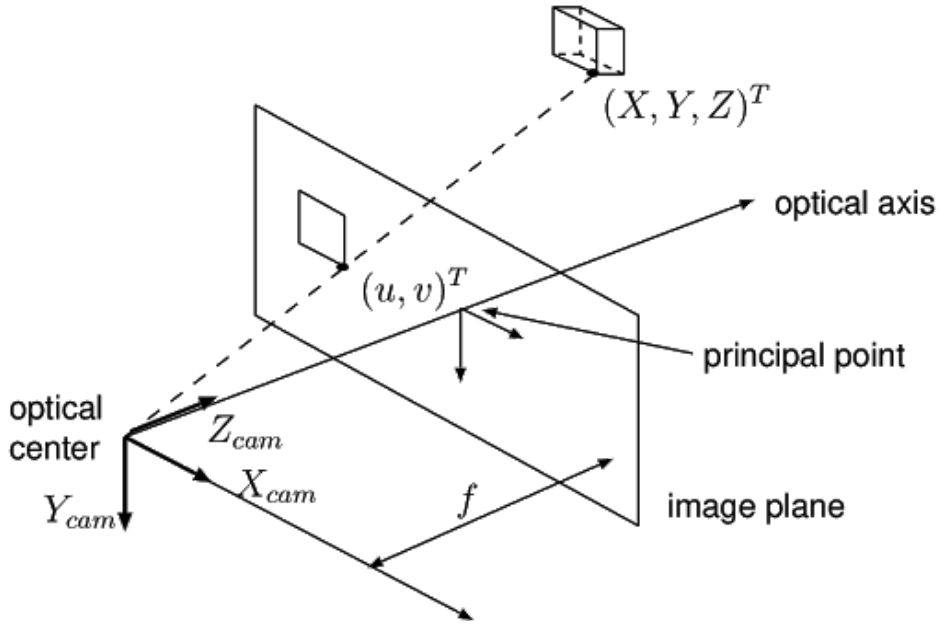


Figure 2.8: Perspective camera model illustration (from [53])

2.2.1 Camera model

A simple standard perspective camera model (also known as *pinhole camera model* illustrated in Figure 2.8) defines the projection of points from the 3D scene into the camera plane. The projection is formalized using homogenous vectors of the point's position in the world coordinate system $\vec{x_w}$, its projection to the image $\vec{x_c}$, and the projection matrix $\mathbf{P}$ as shown in Equation 2.7. The homogenous form of vectors is widely used in projective geometry as it allows for the common operations as translation, rotation, etc., to be formulated using matrix multiplication.

$$\lambda \vec{x_c} = \mathbf{P}\vec{x_w} \tag{2.7}$$

The projection matrix $\mathbf{P}$ can be decomposed into a product of matrices:

$$\mathbf{P} = \mathbf{KR}[\mathbf{I} - \vec{c}], \tag{2.8}$$

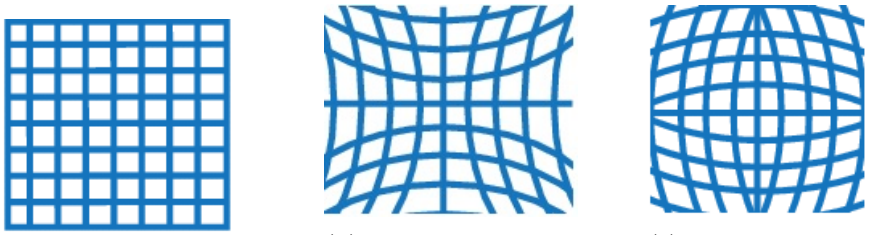
where $\mathbf{K}$ contains the individual intrinsic parameters and both $\mathbf{R}$, $\vec{c}$ contain the extrinsic parameters of the camera. $\mathbf{K}$ is usually denoted as the camera calibration matrix and has a general form [55]:

$$\mathbf{K} = \begin{pmatrix} af & -afcot\phi & u_0 \\ 0 & \frac{f}{sin\phi} & v_0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{2.9}$$

where a and ϕ are the aspect ratio and the skew angle of the digitization raster, consequently, f is the focal length and $[u_0, v_0]$ are the coordinates of the principal point in the image.

$\mathbf{R}$ is the rotation matrix defining the camera's orientation with respect to the world coordinate system. It is a 3 by 3 matrix, but it is defined only by three parameters (Euler angles of rotation).

$\vec{c}$ is the homogenous vector of camera center position in the world coordinate system. This is a widely used model in applications that require measurements of the positions of objects visible in an image [49] (e.g., in applications like visual SLAM[13], which is described further in Section 2.3.5).



(a) No radial distortion

(b) Negative radial distortion

(c) Positive radial distortion

Figure 2.9: Radial distortion illustration (from [37])

Before we can apply such a model to the raw camera images, we have to nullify the effects of possibly present distortions which usually accompany real lenses used in cameras

[49]. The radial distortion is usually the most pronounced. We can also encounter significant tangential distortion with certain lenses. We can recognize both distortions in an image easily by looking for the deformation of straight lines. Examples of radial distortion are illustrated in Figure 2.9. An illustration of tangential distortion is shown in Figure 2.10. The negative effect of these distortions can mostly be removed by applying rectification using the following popular models.

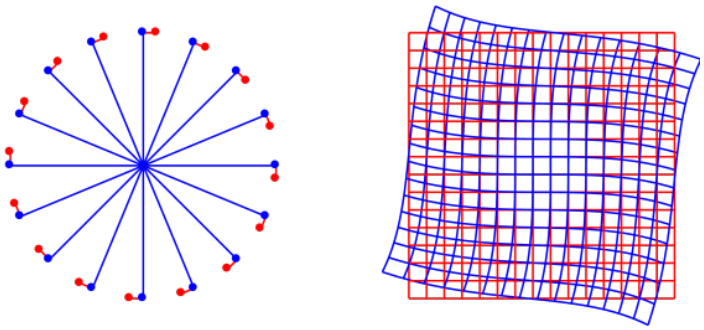


Figure 2.10: Tangential distortion illustration (from [42]; the original points are highlighted in red, the distorted points in blue)

The most common model for radial distortion is the polynomial model in following form:

$$r_c = r_d(1 + \sum_{i=1}^{N} p_i r_d^i), \tag{2.10}$$

where r_d is the distorted distance of a point in an image from the distortion center whereas r_c is the correct rectified distance of the point from the center. N is the degree of the model and p_i are the radial distortion coefficients.

Another useful model is the division model:

$$r_c = \frac{r_d}{1 + \sum_{i=1}^{N} p_i r_d^i}, \tag{2.11}$$

which is proposed in [20]. Its advantage is that it requires a lesser polynomial degree for removing high level of distortion compared to the polynomial model [49].

The popular Brown-Conrady model proposed in [8] can be used for rectification in the cases where both the radial and the tangential distortion are present in the raw image. This model can be formalized as:

$$\begin{aligned} \begin{pmatrix} x_c \\ y_c \end{pmatrix} =& (1 + k_1 r_d^2 + k_2 r_d^4 + k_3 r_d^6 + \cdots) \begin{pmatrix} x_d \\ y_d \end{pmatrix} \\ &+ \begin{pmatrix} [p_1(r_d^2 + 2x_d^2) + 2p_2 x_d y_d][1 + p_3 r_d^2 + p_4 r_d^4 + \cdots] \\ [2p_1 x_d y_d + p_2(r_d^2 + 2y_d^2)][1 + p_3 r_d^2 + p_4 r_d^4 + \cdots] \end{pmatrix}, \end{aligned} \tag{2.12}$$

where k_i are the radial distortion parameters and p_i are the tangential distortion parameters. It should be noted that this is a simplified formulation of the model as we assume that the center of distortion coincides with the origin of the coordinate system.

We can estimate the degree and the values of the coefficients of the models through calibration using image-to-scene correspondences or some known straight lines in the scene. Straight lines have the property of mapping into other straight lines under perspective transformation but are usually visibly deformed by the distortions.

2.2.2 Homography estimation

Points in a plane visible from two different perspective cameras are related by a transformation matrix **H** called homography [40]. The original camera view and the virtual camera view (which we want to transform our images into) are illustrated in Figure 2.7. Searching for the transformation from a rectified image to the ground plane can be reduced to the task of estimating the corresponding homography which is a well-examined problem with many established methods of solution.

If the camera does not move or rotate with respect to the reference point on the car, then this homography will remain constant as the car moves. However, if the camera shakes or tilts considerably during the motion of the car, we need to estimate the transformation between consecutive captured frames. This transformation can be used to correct the effect of the camera position and orientation change that is not included in the homography. This task can be solved (at least in the scenes containing mostly planar structures like urban environments) by searching for the dominant vanishing point that is determined by orthogonal lines in the scene [45].

The established methods for homography estimation usually use some apriori knowledge of the scene captured in the image, e.g., parallel straight lines (calibration via vanishing points) or easily detectable points in a scene with a measurable position in the world coordinate system (calibration via correspondences) [16, 1, 45].

The homography can be extracted from the point or line correspondences using a direct linear transformation method where the minimization of the reprojection error is solved as a nonlinear least squares problem by SVD (Singular Value Decomposition)[1]. Another approach would be to use RANSAC (RAndom SAmple Consensus) to generate homography from randomly drawn samples of correspondences and then choose the one with the most inliers (correspondences with a reprojection error lower than a certain threshold)[24, 27].

Vanishing points are intersections of the lines in the image that are parallel in the 3D scene. They are commonly found in urban scenes thanks to a high number of parallel lines formed by man-made objects[18]. They are valuable for both the estimation of the camera pose (the position and orientation of camera with respect to the world coordinate system) [18] and the reconstruction of the 3D scene [11]. As shown in [56], we can estimate homography between the plane of interest (e.g., ground plane) and the image plane using two detected orthogonal vanishing points even without previous camera calibration under the condition that our camera has a square digitization raster.

The advantage of estimating the rotation to the plane of interest using the vanishing points is that our estimation should have a consistent accuracy for the whole plane. With the correspondences approach, we need to cover most of the visual field to ensure good

calibration because the distribution of the correspondences might influence the quality of calibration with respect to certain areas of the image [24]. Possible outliers caused by mistakes in detection or some other unpredicted inaccuracies might also cause the fitting process to refuse the best model. It is a good practice to visualise the reprojection error to check for these destructive points and recalibrate after removing them. As suggested in [27] a fuzzified version of RANSAC can also be used to reduce inaccuracy caused by outliers.

2.3 Map construction and localisation

As the last step, we have to fuse the incoming map fragments together with auxiliary sensor data (e.g., IMU, LIDAR) to build a map of the surrounding environment while simultanously localising the car within it. This is an instance of a SLAM problem. We cannot separate the mapping and localisation process because the estimation error of the previously detected landmarks' position and the vehicle's pose are correlated as described in [17]. They have to be adjusted simultaneously as the new data arrives in order to reach a consistent solution and a mapping error that converges to zero upon revisiting previously mapped locations.

There are many different approaches to the SLAM problem. In the following sections, we describe the concepts and methods used in the most popular SLAM solutions.

2.3.1 Scan matching

Scan matching is a method used to find a spatial transformation of the point cloud generated from the current sensory input that alignes it with the already existing map. This is used to estimate the current vehicle position.

Many modern SLAM solutions [30, 26] need a fast and accurate scan matcher for this purpose. A simple method known as Iterative Closest Point (ICP) can be used for this purpose. It iteratively searches for correspondences between the closest input and map points, and performs least squares optimization to find the best rigid transformation fitting the correspondences. Unfortunately, the correspondence search in each iteration is too computationally expensive.

That poses a problem for the SLAM application [30]. The polar scan matching is faster in estimation of alignment, but needs the sensor data to be preprocessed first. The real-time correlative scan matching (as proposed in paper [39]) is generally more accurate and robust to noise than ICP [39], though it requires optimizations to work in real-time [30].

2.3.2 Extended Kalman Filters

Another approach is to use the Extended Kalman Filters (EKFs) as in, e.g., Hector SLAM[30]. EKFs use the measurements to create a covariance matrix of the vehicle pose and

the position of the detected points. This matrix is used to estimate the new position and orientation of the vehicle based on a state model of the vehicle's motion [21]. The results can be combined with the scan matching to optimize for the most consistent pose estimate. The disadvantage of EKFs is that they are highly dependent on the correctness of the assumed motion model and sensor noise. This can cause them to diverge when the assumptions are inaccurate [23].

2.3.3 Particle filters

Other SLAM methods (e.g., Gmapping[23]) are based on particle filters that learn grid maps. The particles are connected to specific versions of the environment map. The Rao-Blackwellized particle filters [15], which are more accurate than standard particle filters are especially successful. There is an issue with the number of particles required for the algorithm to be able to correctly build a map being too high. This makes the process unfeasible for a real-time application due to the high computational cost. Reducing the sampling is possible but it can cause the correct solution to be thrown away (this problem is also known as the particle depletion)[23]. In [23] two enhancements are proposed to deal with the particle depletion problem while keeping the complexity low: *an adaptive resampling method* and *a way of selecting dissimilar particles*. These enhancements make the approach computationally feasible for SLAM.

2.3.4 Graph-based approach and loop closure

Many modern SLAM solutions (e.g., Cartographer[26]) use a graph-based approach instead of a particle filter. It builds a graph where nodes represent vehicle poses or detected obstacles. In this graph, edges are built to represent constraints stemming from the sensor data. It matches consecutive laser scans (a structure containing data about distance of the detected obstacles from the sensor and an angle at which the detected obstacles are located) to each other to create a small submap that is accurate enough to be stored without further modifications. The currently perceived scan is matched against nearby submaps. The position of the vehicle and features is deduced by solving the local optimization problem over a probability occupancy grid map [26].

The inevitably accumulated error on localisation and features' positions is corrected via a process called *loop closing*. Loop closing is done by solving a global optimization problem for revisited locations. It also updates past estimates. Therefore, after the loop closure our path (collection of vehicle's past positions) can be more accurate [26].

2.3.5 Visual SLAM

Many visual SLAM solutions [13, 43, 29] use the Scale Invariant Feature Transform (SIFT) descriptors to keep a compact representation of an image patch for landmarks storage and comparison. The SIFT descriptors are widely used because they are invariant to

translation, scaling, rotation and partially even to affine projection, occlusions and current illumination [33]. This invariancy is valuable for applications where the landmarks are visible from different points of view due to, e.g., camera movement. The landmarks can be uniquely identified thanks to the descriptors as opposed to the usual approach with sensors like LIDAR where the map consists of indistinguishable points.

These detected landmarks can be stored in a database and connected to a probabilistic model of the map that is updated using the methods mentioned above (e.g., EKF in [13], or Rao-Blackwellized particle filters in [43]).

Chapter 3

Problem statement

Our goal is to implement a system that enables the car to perform SLAM mainly depending on camera images as a source of data. This can be broken down into the following subtasks.

1. Lane markings visible in each frame must be detected.
2. The positions of the landmarks must be transformed to the ground plane to create a map fragment of the current surrounding environment.
3. Localization within the global map must be performed using the current map fragment and pose[1] estimate.
4. The global map is augmented by the landmarks (the detected lane markings) detected in the current frame.
5. The global map also has to be optimized to keep it consistent after returning to the already perceived areas.

The solution should enable the car to first map the environment using a low speed. Then, in the already mapped space, localisation should be robust enough to be able to deal with the highest speed possible so that it can be used further for autonomous applications (e.g., autonomous driving, racing, and other more complex tasks requiring navigation through the environment).

3.1 Equipment

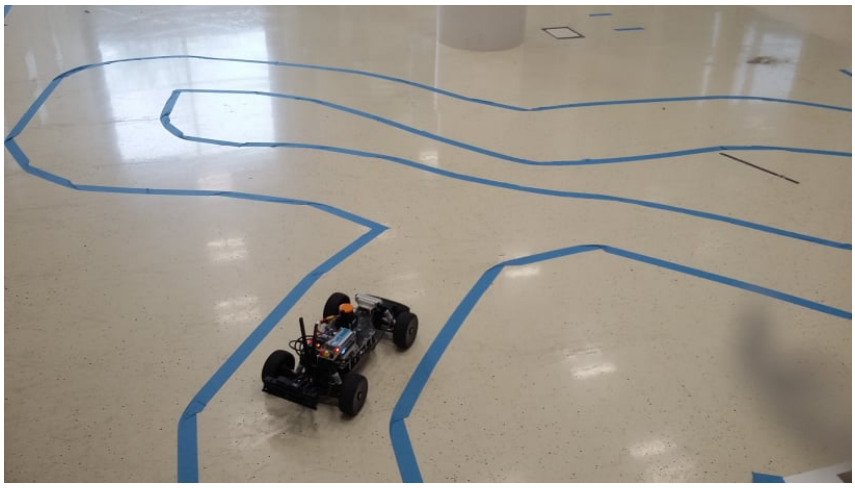
The system developed in this thesis is designed for the *F1/10 autonomous car* which can be seen in Figure 3.1a. F1/10 has standardized equipment and is designed for competition of autonomous vehicle systems. The list of relevant hardware and software equipment of F1/10 can be found in Table 3.1. The following sections focus on the components relevant to the topic of this thesis.

[1] We define pose as a collection of the extrinsic camera parameters which define the camera's (therefore also the vehicle's) position and orientation with respect to the world coordinate system.

Camera	Intel RealSense D435
IMU	Sparkfun 9DoF Razor IMU M0
LIDAR	Hokuyo UST-10LX
Computing module	NVIDIA Jetson TX2
Operating System	ROS kinetic (under Ubuntu Linux 16.04)

Table 3.1: F1/10 equipment

(a) F1/10 autonomous car



(b) Lane markings example

Figure 3.1: F1/10 and the environment

3.1.1 Camera

We use the camera to acquire a single image stream of the space in front of the car, which is then used to detect lane markings and their relative position to the car in each frame. It has an RGB resolution of 1920x1080 pixels and a framerate of 30 FPS. The camera is mounted 13 cm above the ground plane with a 0° pitch. The angle of view of the ground plane is approximately 60°. The effective visibility range of the ground plane is from 0.4 m to 2.5 m. This visual field of the ground plane is quite small when compared to area of range of LIDAR, which is a sensor usually used in SLAM applications. If we compare it to the scanning field of LIDAR, the visual field of the camera is approximately 100 times smaller. On the other hand, unlike LIDAR, it can see beyond the first detected point. The objects beyond the effective range on the ground plane are blurred as they are too close to the horizon line. Due to an incompability with the computing module mounted on *F1/10*, there are some limitations to the capabilities of the camera (these are discussed further in Section 5.2).

3.1.2 Inertial Measurement Unit

IMU provides acceleration data, which is considered to estimate the car's relative position between two discrete moments during the localisation process. Our IMU has an update

rate of 50 Hz.

3.1.3 LIDAR

Lidar provides range data of surrounding physical obstacles in a plane parallel to the ground. The angle in which LIDAR scans the plane is 270°. The update rate is 40 Hz. The angular resolution is 0.25° and the working range is from 0.06 m to 10 m.

3.1.4 ROS

The whole application is targeted at Robot Operating System (ROS), which operates on F1/10. The individual processes used in the system are represented as ROS *nodes* which communicate with each other via publishing and subscribing various data messages in so called *topics*.

3.2 Road model

The testing environment should represent a scaled-down model of a structured road, possibly including intersections. Unicolored lane markings are used as boundaries. A track example can be seen in Figure 3.1b. The used ground surface has a high light reflectance and its appearance is irregular with noise-like shapes. The ground is flat. We assume there are no other vehicles or large objects on the track that could break the line of sight between the camera and the lane markings.

For most of our experiments, we use the following two tracks. The first track is an asymmetric circuit, which is around 15.0 m long and contains long straight parts of the road. The second track is an oval circuit that is 8.8 m long and mostly consists of 90° turns.

Chapter 4

Methods

In this chapter, the methods, which we use to solve the assigned problem described in the previous chapter, are proposed. Their advantages, disadvantages and suitability in the context of our application are discussed. The structure of the solution is drawn.

At first, we focus on the methods used for identifying the lane markings from the individual images. Further, the algorithm for extracting the local map fragment from the detected landmarks is proposed. In the last section, the approach to global map creation and localisation is described.

4.1 Lane markings detection

At first, we crop the image to select our ROI, which is below the horizon line. Experiments showed that apart from the moments of high acceleration of the car the camera does not significantly change its angle of view. Because of that, we use the same pre-estimated ROI for each frame thus saving computational cost without generating a significant error (this is verified further in Chapter 6).

To detect the lane markings from the individual image frames that we want to map and use as features for localisation we use a combination of two popular image processing methods: *edge detection* and *color segmentation*.

4.1.1 Edge detection

Edge detection gives us good candidates for the edges of lane markings thanks to the big color intensity gradient between them and the road. On the other hand, edge detection also detects all kinds of other distinct objects in the image like reflections on the ground surface, or any distinct patterns that are drawn on the surface, as can be seen in Figure 4.1.

We decided to use Canny Edge Detector for edge detection with hysteresis. We use a fast and effective implementation of Canny Edge Detector from OpenCV (Open Source Computer Vision Library [7]).

We could use a geometrical constraints approach to recognize which candidates are the true edges of the lane markings. Using the Hough transform, we could restrict the shape of the lane markings, e.g., to parabolas of a certain minimal length. This could filter out a lot of

(a) The original image (b) The edge detection result

Figure 4.1: An example of distinct reflections and surface artefacts present on the track

unwanted edges from smaller objects on the surface. Unfortunately, the collected test images showed that the most common unwanted noise in the input data were reflections of straight parallel lines that are common in urban man-made environments (e.g., columns, poles, window edges). The shape of these objects is more or less similar to the desired detected object – the lane markings. This causes the edges detected on these objects or their reflections to score very high in the geometrical model fitting.

Another disadvantage of this approach is that our assignment states an arbitrary shape of lane markings. This would require a very broad model of lane markings. Unfortunately, this would result in a potentially unfeasible computational cost of the Hough transform for the real-time application. Due to these disadvantages, we decided to omit geometric constraints from the detection algorithm and instead use color segmentation.

4.1.2 Color segmentation

Color segmentation has a straightforward implementation and is computationally cheap. On the other hand, it works well only when the detected object is of dissimilar color compared to the rest of the perceived environment. In our case, lane markings have a distinct color so this method is applicable.

To make color segmentation more robust to the illumination changes (as explained in Section 2.1.1), we use the HSI color space. We transform the image from the RGB to HSI space and then apply thresholding of the individual channels. The thresholding hue is usually enough to recognize the lane markings' color from the background. There are edge cases for which thresholding other components can help to reduce or fully negate false positives (e.g., reflections of similar color or an overexposed image, for details, see Chapter 6 which focuses on testing and verifying of these methods). On the other hand, omitting other components saves us approximately $\frac{2}{3}$ of computation time.

The combination of edge detection and color segmentation is almost a pixel-wise independent process (apart from hysteresis thresholding in Canny edge detector). Therefore it is unable to deal with occlusions (lane markings hidden behind some objects or missing). This disadvantage is caused by not using a more generative approach, e.g., a geometrical model,

which could help to interpolate lane markings in the holes caused by an object breaking the line of sight or by partial erasure of lane markings.

4.2 Map fragment construction

The radial and tangential distortion in our camera is cancelled out using the Brown-Conrady model (Equation 2.12) with 3 radial distortion coefficients and 2 tangetial distortion coefficients.

We apply the perspective camera model on undistorted images. To transform the detected objects from the camera plane to the ground plane, a homography is used. Once again, we assume that the camera does not rotate enough to significantly change the transformation between planes during the motion of the car. Assuming that, we can pre-estimate the homography using calibration via image-to-scene point correspondences.

We have to generate homography from a sample of correspondences $(x_i^c, y_i^c) \leftrightarrow (x_i^g, y_i^g)$ between the camera plane and the ground plane. By following standard steps, we build a matrix of linear equations $\mathbf{L}$ that constraints the individual parameters of $\mathbf{H}$. At first, we write down the relation between correspondences and derive linear constraints from it:

$$\lambda_i \begin{pmatrix} x_i^g \\ y_i^g \\ 1 \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x_i^c \\ y_i^c \\ 1 \end{pmatrix} \implies \tag{4.1}$$

$$\lambda_i = h_{31}x_i^c + h_{32}y_i^c + h_{33} \tag{4.2}$$

$$\lambda_i x_i^g = h_{31}x_i^c x_i^g + h_{32}y_i^c x_i^g + h_{33}x_i^g = h_{11}x_i^c + h_{12}y_i^c + h_{13} \tag{4.3}$$

$$\lambda_i y_i^g = h_{31}x_i^c y_i^g + h_{32}y_i^c y_i^g + h_{33}y_i^g = h_{21}x_i^c + h_{22}y_i^c + h_{23}. \tag{4.4}$$

Then, using N correspondences, we can translate these constraints (4.3-4.4) into a system of linear equations in the form:

$$\underbrace{\begin{pmatrix} x_1^c & y_1^c & 1 & 0 & 0 & 0 & -x_1^c x_1^g & -y_1^c x_1^g & -x_1^g \\ 0 & 0 & 0 & x_1^c & y_1^c & 1 & -x_1^c y_1^g & -y_1^c y_1^g & -y_1^g \\ x_2^c & y_2^c & 1 & 0 & 0 & 0 & -x_2^c x_2^g & -y_2^c x_2^g & -x_2^g \\ 0 & 0 & 0 & x_2^c & y_2^c & 1 & -x_2^c y_2^g & -y_2^c y_2^g & -y_2^g \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & & \\ x_N^c & y_N^c & 1 & 0 & 0 & 0 & -x_N^c x_N^g & -y_N^c x_N^g & -x_N^g \\ 0 & 0 & 0 & x_N^c & y_N^c & 1 & -x_N^c y_N^g & -y_N^c y_N^g & -y_N^g \end{pmatrix}}_{\mathbf{L}} \begin{pmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{pmatrix} = \vec{0} \tag{4.5}$$

This homogenous system of linear equations Equation 4.5 contains 9 parameters and each correspondence pair generates 2 constraints. We need 4 pairs of corresponding points to restrict the space of solutions to one unique non-trivial solution. In a degenerate case when any 3 drawn correspondence points lie on a line, the linear system of equations will not yield one unique solution. We need to detect this and select another correspondence sample.

To find the best homography (supported by the most correspondences) we use a brute-force approach. We generate homography from each sample combination of acquired correspondences. Since we only use about 100 correspondences that uniformly cover the image, this approach is computationally feasible. There is no urge to use a fast stochastic solution (e.g., RANSAC) because the calibration is done only once (per camera relocation) and it is not required to run in real-time as opposed to the mapping process. To ensure consistent accuracy of transformation throughout the image, we collect uniformly spread reliable correspondences from printed chessboard patterns covering most of the visual field. For a visualisation of the reprojection error of the transformation see Section 6.2, which focuses on verification.

4.3 Mapping and localisation

For the purpose of global map construction and localisation, we use a 2D LIDAR SLAM solution called *Cartographer* [26]. Cartographer is a graph-based SLAM solution developed mainly for LIDAR-based SLAM. In Figure 4.2 a system overview of Cartographer is displayed. As the diagram shows, Cartographer consists of two main parts called *Local SLAM* and *Global SLAM*.

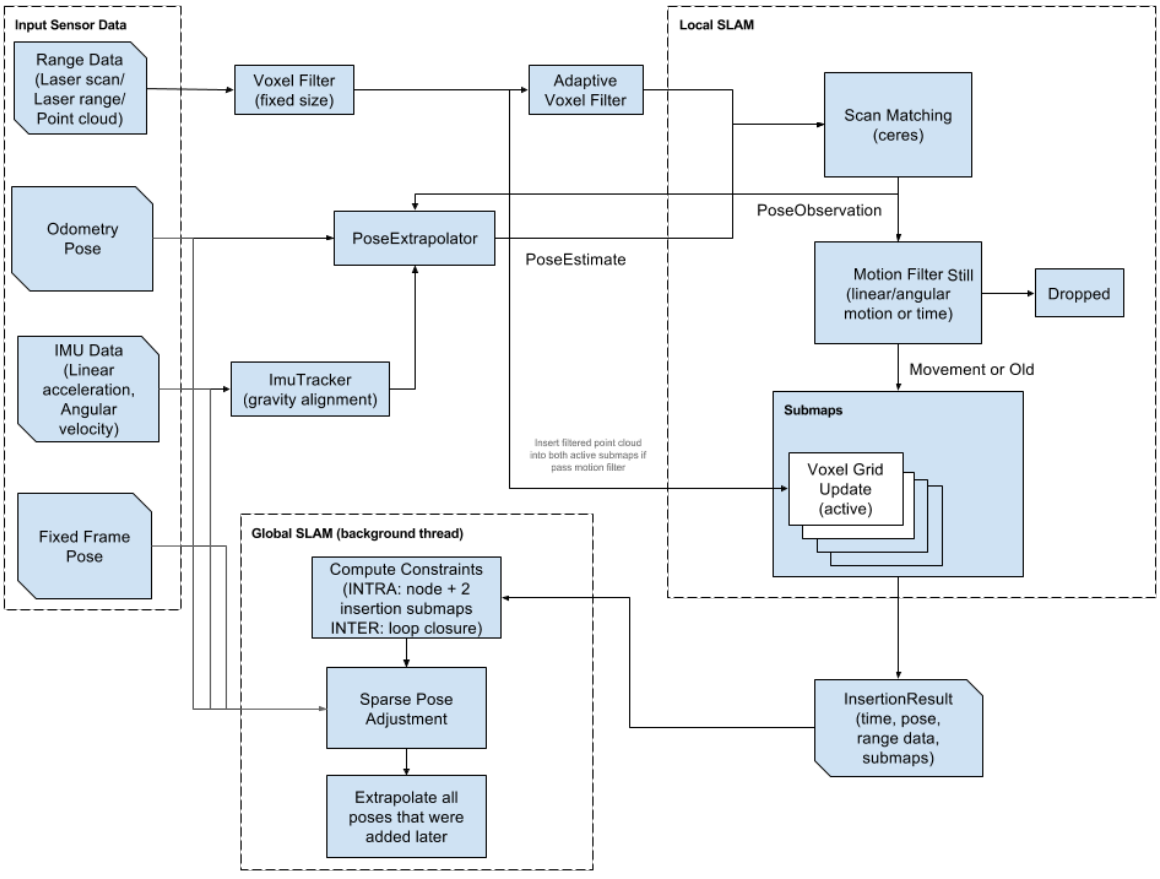
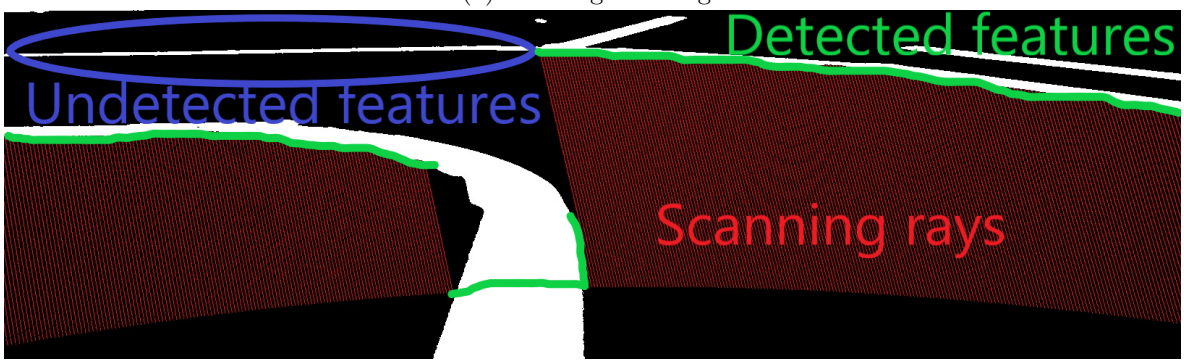


Figure 4.2: Cartographer system overview (from [48])

Local SLAM is responsible for estimating the current vehicle's pose and constructing locally accurate submaps. It proceeds by taking downsampled scans and using a scan matching optimization method with an initial estimate created from the available auxiliary sensor data (in our case IMU). These submaps and their relative alignments and positions in the fixed world coordinate system are stored to create a combined global map. The global map is represented as probability occupancy grids, i.e., grids where each cell has a value corresponding to the probability of containing a landmark (being occupied by a landmark).

The Local SLAM mapping process accumulates an error as it relies solely on scan matching against recently collected scans assembled in a submap. To remove the error, Global SLAM generates loop closure constraints between the scans and the submaps that are based on repeatably perceived areas. Both Local SLAM scan matching and Global SLAM loop closure are formulated as a nonlinear least squares optimization problem and solved using the Ceres [2] solver. Ceres is a tool for modelling and solving complicated optimization problems [2]. Scan matching simply maximalizes the sum of probabilities in the occupancy grid where the newly scanned points are placed. Loop closure minimizes Huber loss (a loss term formulated to avoid the influence of outliers) on relative alignment constraints that are built by the Global SLAM [26].

(a) The original image



(b) Visualisation of scanning rays (in a color segmented image)

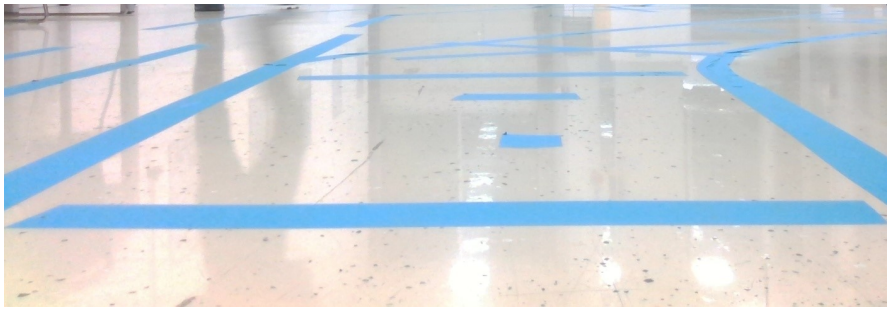
Figure 4.3: An example of the limitation of range finder-like detection

4.4 Converting data from an image to LIDAR-like data

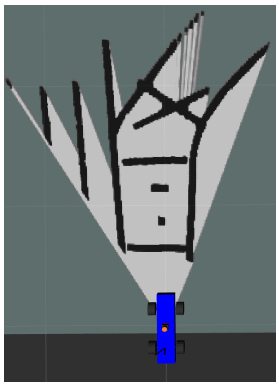
Cartographer was developed to map obstacles using mainly LIDAR scans. We use it to map lane markings by providing it with a LIDAR-like data structure containing map fragments of the lane markings that were extracted from the camera images.

Using the estimated homography, we can draw scanning rays in the image plane. Then, we can trace the first intersections of the rays with the detected landmarks to simulate LIDAR-like scanning of the environment and provide Cartographer with range data. An example of such "image scanning" is shown in Figure 4.3.

This approach limits the number of valuable feature points that can be provided to Cartographer. While the camera can see landmarks on the ground behind the first one, 2D-LIDAR cannot see behind the first obstacle in the plane of scanning. To avoid losing these valuable landmarks, we can extract a 2D point cloud containing all intersections of scanning rays with detected points from the current image and provide it to the matching algorithms (For an illustration see Figure 4.4).



(a) The original image



(b) The resulting 2D-point cloud

Figure 4.4: An example of 2D-point cloud extraction

4.5 System configurations

We experimented with various approaches with different configuration and sensor combinations to observe their influence on mapping and localisation accuracy. The explored variants are described below.

We provide Cartographer with acceleration data from IMU for the purpose of initial pose estimation of new frames.

We experimented with adding a second camera to the back of the car, which could be a valuable source of feature points for the scan matching. Unfortunately, two parallel streams of images proved to be computationally unfeasible for our equipment.

We also attempted to provide visual odometry to Cartographer as another source of initial pose estimation of new frames. Testing showed that for our camera configuration the visual odometry has problems with finding s sufficient ratio of inliers when estimating the relative position of consecutive frames. This might be caused by a low position of the camera and a forced high exposure time (as explained later in Section 5.2), which causes the features on the ground surface (which are valuable for movement estimation) to be blurred. Because it did not enhance the performance of Cartographer, we decided to exclude it from our solution.

Another possibility is to combine the camera and LIDAR approach to simultaneously build two separate maps (a camera based map of lane markings and a LIDAR based map of surrounding obstacles). These maps are linked by the same vehicle trajectory which makes the localisation process more robust as it has to take both maps into account. This is achieved by providing Cartographer with LIDAR and camera data in separate ROS topics and by establishing a dynamic *tf* (transformation) link, which separates these two maps in the plane but ties their relative localisation. This is illustrated in Figure 4.5. This combined SLAM can be used to create accurate maps of both the lane markings and the obstacles. Unfortunately, using the LIDAR and camera together forces us to keep certain Cartographer parameters set on values ideal for LIDAR. This includes the resolution of the occupancy map grid, which is required to be set above 5 cm for LIDAR. This means that acquired maps have a lower resolution than we would normally use for camera settings.

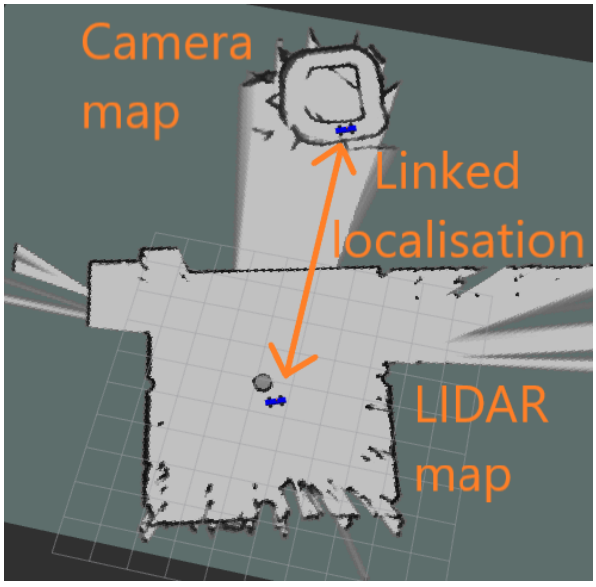


Figure 4.5: An illustration of Combined SLAM

We also use Cartographer in pure localisation mode for localisation error testing in a previously created map. In pure localisation mode, the Cartographer SLAM node remembers a limited amount of previously built submaps and compares these to a stored global map from a previous run of the system.

All alternative system configurations are realised as separate ROS launch files with their respective parameter configurations files.

Chapter 5

Implementation

In this chapter, we review the structure and specifications of the implementation of our solution. We will also note which tools we used for the realization of an effective and fast implementation of the required methods that were described in the previous chapter.

5.1 Structure

As mentioned in Chapter 3, we use ROS on the platform. The diagram in Figure 5.1 shows our ROS communication graph (including optional components related to the combined SLAM configuration).

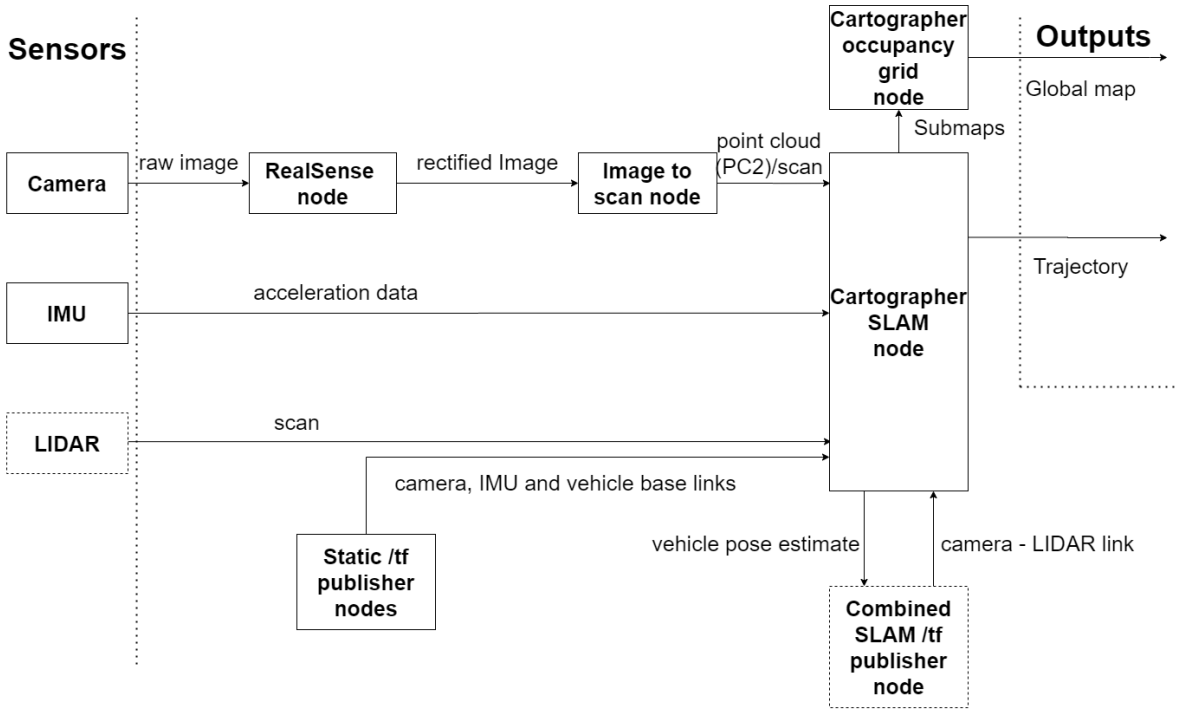


Figure 5.1: Diagram of our system

As we can see in Figure 5.1, the images from the camera are first preprocessed in *RealSense node*, which carries out the rectification of the images. Then, they are sent to our *image to scan node*, which performs the detection of lane markings and the extraction of a map fragment of the landmarks. Depending on the configuration, these landmarks are sent

to *SLAM node* either in the form of a PC2 or in the form of a LIDAR scan. Cartographer can also receive auxiliary sensor data from IMU and/or LIDAR. The submaps are merged to a global map by *Occupancy grid node*.

The transformations linking different coordinate systems are published under a */tf* topic in ROS. *Cartographer SLAM node* keeps publishing the estimated current position, path and the submaps with respect to the fixed frame as they are being constructed. The transformation links (between robot sensors and other important coordinate systems) are published by *static tf publishers*. In the LIDAR option, where we let Cartographer simultaneously build a LIDAR-based map and a camera-based map, a dynamic link simulating the localisation connection between these two maps is required. This link is generated by *combined SLAM tf publisher node* based on a current pose estimate from Cartographer.

5.2 Framerate limitation

As specified in Chapter 3, the camera used in this thesis should provide 30 FPS of images with required resolution. Unfortunately, due to the compability issues with the Jetson TX2 module that we use for computation, the camera provides lower FPS regardless of the selected resolution. In addition to this, the provided framerate is also unstable.

The framerate is an important factor for the mapping and localisation application as with a lower framerate the overlap between consecutive images can be too small for the scan matching to work accurately. This is especially prominent for a higher speed of the vehicle. There is a known parameter configuration discussed on NVIDIA support forums [38] for which the camera quite stabily provides 18-22 FPS for the mentioned computing module. This configuration requires the auto exposure function to be turned off and the exposure of the camera to be set to 40 ms, which is about 2.5 times higher than the default value. Due to this setting, the images tend to be overexposed when a strong light source is present in the scene. To reduce the negative effect of overexposure, we adjust other camera parameters; mainly we set gain to a minimum value to dim the image. As a result, we a have stable 18-22 FPS which provides enough data for the scan matching to work at least at lower speeds. As a result of the forced exposure setting, the application is sensitive to strong light reflections and certain lane markings may not be visible in the image due to the overexposure (for an example see Figure 5.2).

5.3 Computational feasibility

The detection and point cloud extraction implementation must be fast enough for a real-time application. In our case, we want the application to run 20 times per second. This corresponds with the camera's framerate of around 20 Hz on the target platform. To test the computational load, we measured the execution times of the image color segmentation, edge detection and the map fragment extraction of the incoming camera images.

(a) Local overexposure

(b) Whole frame overexposure

Figure 5.2: Examples of encountered overexposure cases due to a strong reflection from the ground surface

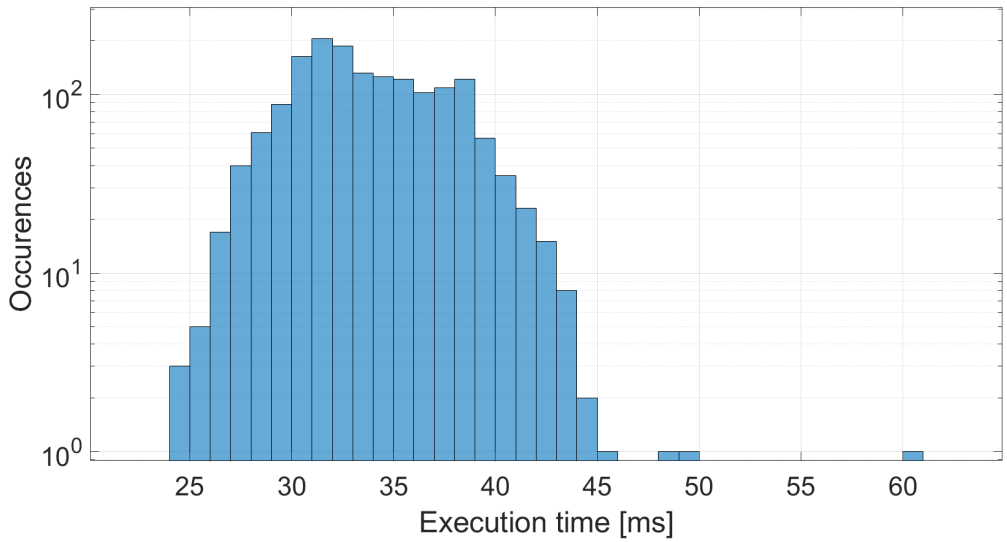


Figure 5.3: Histogram of total execution times

The histogram of results for an 80 s long experiment where the vehicle was riding through our oval circuit is plotted in Figure 5.3. The resulting average, minimum and maximum execution times for the color segmentation, edge detection and the map fragment extraction are shown in Table 5.1. The average total execution time is 33.9 ms which should allow for up to 30 Hz. The execution of our implementation is fast enough and leaves a margin because the images are published at 20 Hz.

Table 5.1 also shows a dangerously high maximum total execution time of 60 ms that

Table 5.1: Execution times analysis

	Average [ms])	**Minimum [ms]**	**Maximum [ms]**
Color segmentation	6.488	4.172	14.397
Edge detection	13.568	9.449	22.986
Map fragment extraction	13.836	8.916	30.755
Total Execution Time	33.892	24.569	60.271

could cause some images to not be processed in time for Cartographer to be able to work with them. A closer look at the measurements showed that this maximum value only occurs upon the start of the system, probably due to the initialization of required data structures. For the consecutive iterations, the execution times are always below 50 ms which is the boundary of the frequency of the incoming images. To achieve a higher framerate, the sampling of the pointcloud that is generated from the image can be reduced without accuracy decrease. This is due to the sampling currently being about 10 times higher than the occupancy grid sampling that is used by Cartographer which processes the pointcloud.

5.4 Implementation notes

For an effective and fast implementation of certain algorithms, we made use of available open source libraries. OpenCV [7] was used for calculations with matrices required for the color segmentation and map fragment extraction. For edge detection, we used an implementation of Canny edge detector from OpenCV. Cartographer [26] was used as a solution to the 2D SLAM problem.

Cartographer was not designed to work with data from a camera as discussed in Section 4.4. Fortunately, it was designed with over 50 modifiable parameters. Extensive parameter tuning was required to achieve a functional solution because data the from camera has different properties than LIDAR scans. For example, we had to: inhibit modification of scan subdivisions, adjust the sampling of the global constraint construction to give a larger weight to the local constraints and resize the resolution of various filters and of the map grid. The parameter files for individual configurations are included in the attached CD (for CD content see Table 1).

Chapter 6

Testing and verification

Through testing we acquired information about the various accuracy limitations of the individual parts of our solution. These limitations are discussed in this chapter to describe how effective our solution is when used in regular cases. We also show the problematic edge cases that cause inaccuracies or mistakes. Further, we describe the metrics that we decided to use for the evaluation of our system's performance. These metrics are used in the next chapter to evaluate the conducted experiments.

6.1 Lane markings detection testing

We conducted experiments on our testing tracks under stable light conditions. This showed us that thresholding just the hue component is enough to robustly segment the markings' color (for an example of segmentation see Figure 2.2). However, we decided to still use the other channels as they help us reduce false positives that appear in special cases. When there are strong reflections, the camera may perceive parts of the ground plane as having hue within the thresholded interval as can be seen in Figure 6.1.

(a) The original image

(b) The result of thresholding the hue component

(c) The result of thresholding hue and saturation components

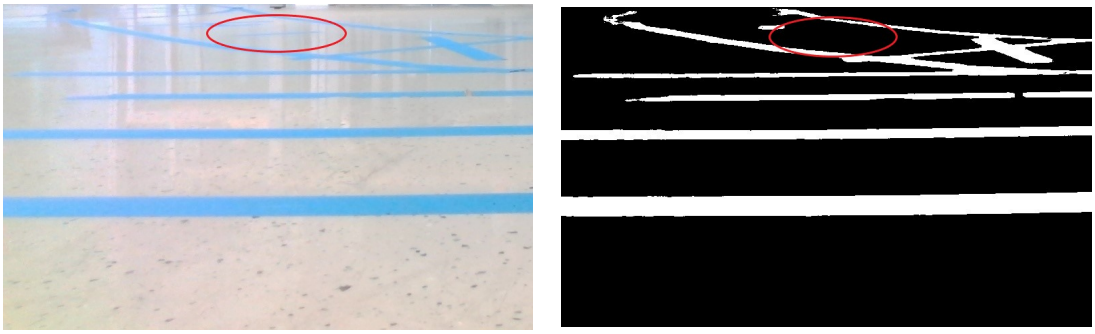
Figure 6.1: An example of strong reflection for which thresholding more components of HSI is required

Similar cases occur when part of the image is overexposed which can be caused by part of the track being affected by a strong local light source as can be seen in Figure 5.2. The overexposure problem could normally be solved by auto-exposure or setting a shorter exposure window of the camera. Unfortunately, in our case we are forced to use a fixed exposure time of 40 ms (as explained in Section 5.2).

The negative effect of both the overexposure and reflections can be reduced or even removed by thresholding the iluminance and saturation. The reduction of the false positives that can be achieved by this approach is shown in Figure 6.1.

The further we go on the ground plane from the position of the camera, the less pixels are available for a unit of length (in the direction of the camera's orientation). This was mentioned in Chapter 2 and illustrated in Figure 2.6. The accuracy of the point cloud extraction decreases with this distance aswell due to the pixel rounding.

At a certain point, the lane markings orthogonal to the vehicle's bearing are no longer detectable as they are too blurred. An illustration of an effect that causes this can be seen in Figure 2.6. The distant horizontal lines are barely visible because they are only 1-3 blurred pixels wide and the color segmentation often misses them. An example of this can be seen in Figure 6.2, where the color segmentation misses the distant line marked in the red circle. The lanes that are vertical in the image are easier to detect even in the far field because they do not merge with the background so much, especially when the vehicle is moving parallelly to them.



(a) The original image

(b) The result of color segmentation

Figure 6.2: A sample image with numerous visible lane markings to illustrate the decaying accuracy with distance from the camera

We found out that the approximate maximum distance at which the color segmentation is still capable of robust lane marking detection is 2 m (for our camera configuration described in Section 3.1). Due to this, we have limited the range of the image scanning to save computational time and to reduce false negatives stemming from the detection part of the system. On the other hand, distant features are valuable for accurate pose estimation during SLAM. Due to this, we experimented with the value of the maximum range of image scanning. As a result, we found out that the lowest mapping and localisation error is for a maximum scanning range of 2.5 m. By using a higher range, the mapping relies on inaccurate detection of distant blurred lane markings, which causes map fragments to be blurred and damages the further scan matching process. When we use a smaller range, we abandon too

many valuable features, which also results in less accurate localisation and mapping.

6.2 Map fragment acqusition testing

The map fragment accuracy is a crucial factor for the quality of the submaps, and therefore, to the successful mapping of the environment. To calibrate and test the accuracy of the map fragment, we make use of printed calibration patterns (e.g., a chessboard pattern). Using such patterns to calibrate the transformation from the camera image to the plane of interest is a common practice in applications requiring similar calibrations of extrinsic or intrinsic camera parameters (for an example, see paper [19] focused on camera self-calibration). We let the camera take images of these patterns in different scenarios. We can then extract the projections of the corner points in the images to the ground plane and compare their positions with the known measured values to quantify the projection error. The projection error is mostly caused by a limited accuracy of the calibrated homography and by a slight rotation of the camera due to the acceleration of the vehicle. To test this in the case of the moving car, we can use the relative projection error (the difference between known distance of the points in the scene and the distance extracted from projections of those points to the ground plane) of the chessboard corners as we do not have the knowledge of the camera position needed to extract the exact reprojection error.

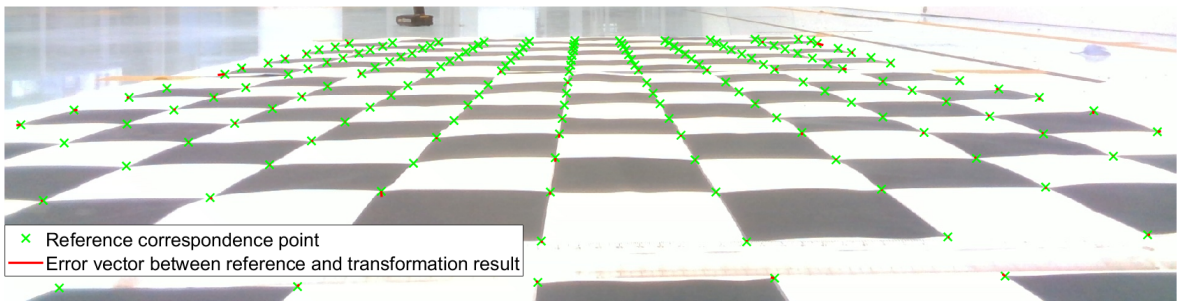
Figure 6.3 shows the error vectors of the projection while the car is stationary. The mean reprojection error is 6.7 mm and the max reprojection error is 28.7 mm in this sample image. We used an image of the chessboard pattern captured while the vehicle was moving and extracted the relative reprojection error between 100 recognized point correspondences. The average relative reprojection error of neighboring points is 6.9 mm, the maximum reprojection error is 36.1 mm. The reprojection errors are shown in Table 6.1 as well. In both cases, the projection errors satisfy our accuracy needs since the average error stays below 1 cm, which is the resolution of the map used further. We expect the projection error to be worse in moments of strong acceleration, when the camera's orientation changes with respect to the ground plane.

	Mean [mm]	**Max [mm]**
Stationary	6.7	28.7
Moving	6.9	36.1

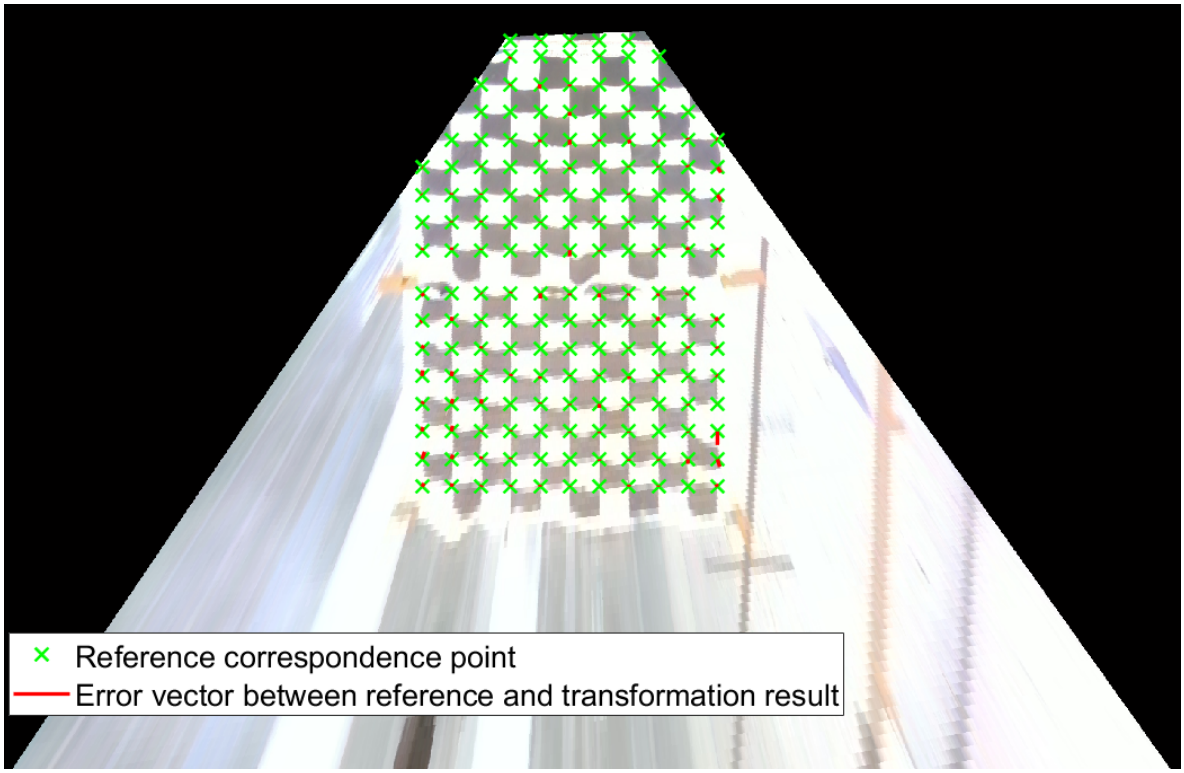
Table 6.1: The reprojection errors measured using the chessboard pattern images

6.3 Localisation and mapping evaluation metrics

The accuracy of the localisation and mapping process cannot be quantified straghtforwardly. We decided to separately measure the pose estimation error and the inacurracies of the constructed map. For pose reference, we use LIDAR SLAM which has a negligible error in the indoor small area that is used for testing. We can then compare these values to evaluate the system's overall accuracy.



(a) The original image



(b) Ground plane projection

Figure 6.3: Visualisation of the projection errors on correspondences generated by the chessboard pattern

6.3.1 Pose estimation error

The pose estimation error can be divided into a position error and an orientation error. We measure both errors with respect to the reference solution.

For the position error, we use euclidian distance between the estimated position and the reference. For the orientation error, we use the difference in the angle of the estimated bearing between the solution and the reference. We can use this approach to compare the apriori localisation error created by local SLAM to quantify the drift of the localisation caused by inaccurate submap alignment.

We can also compare the final trajectories adjusted by global SLAM to show how effective the global constraints were in correcting the localisation drift. Because of the optimization that changes original submap poses, there is no fixed link between the individual estimated poses and the original position. This means that both the reference path and our solution path may drift apart slightly and still create an accurate converging map.

Apart from global and local SLAM, we also test the pure localisation mode, which localises the vehicle in a previously built map. The pure localisation mode uses global optimization heavily to correct the alignment of its submaps with the stored global map. This helps it to not drift away from the given global map. Often, this also leads to a correction of the past path error. On the other hand, we assume, the localisation mode's purpose is to provide a real-time current pose estimate so it should be evaluated correspondingly. To quantify the quality of this process, we use the error of the pose estimation prior to the optimization.

6.3.2 Mapping error

To measure the mapping error, we placed special "X" shaped landmarks and compared their mapped relative distances to their actual measured relative distances. This can be done by comparing these two alternative measures:

- the error on consequently mapped relative distances of landmarks to create a measure of the submap error that was accumulated in the last several seconds,
- the difference in position of the same landmark during multiple lap experiments to quantify the global map consistency throughout the mapping process.

This approach is good for measuring the error accumulated by Local SLAM. For Global SLAM, it does not make sense to use any metrics collected during the experiment as it post-processes the poses later. Instead, we need to look at the final map and evaluate whether it converged to the right solution and how blurred the landmarks are.

The distances between landmarks differ from 1.5 m to 3 m. For a better comparison of mapping errors, we use a percentage error with respect to the correct relative distance. For us to be able to also compare the map consistency error for tracks of a different length, we need to take into the account that the oval circuit is almost twice as short as the asymmetric track. This is caused by the fact that local SLAM accumulates a localisation error over time, which means it will accumulate a higher error on a longer track.

6.3.3 Track limitation

As stated in Section 3.1.1, the camera has a much smaller field of view when compared to LIDAR mounted on the platform. These differences cause LIDAR to provide more feature points (in addition, they are spread throughout a wider area and in different directions) than those provided by the camera. These factors help us keep the localisation error low during the submap building in LIDAR SLAM. Whereas with the camera, there is a risk of significant

localisation drift caused by not having enough unique features. Therefore the estimation of the correct alignment in cases where there are not many lane markings visible in the field of view, e.g., during a turn (as illustrated in Figure 6.4), can be prone to error.

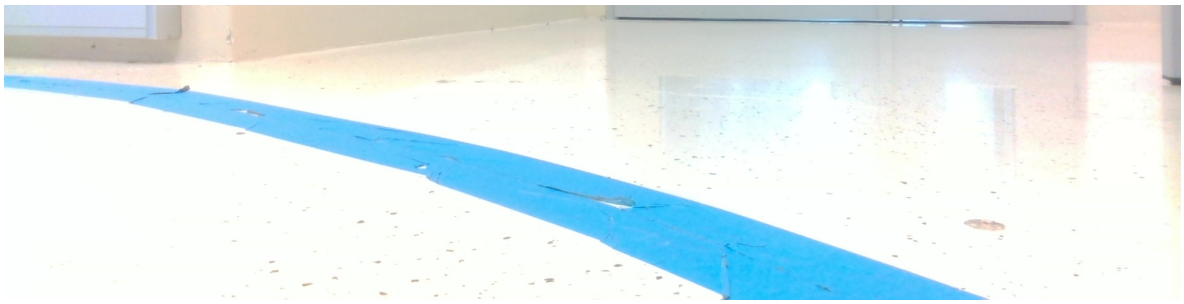


Figure 6.4: An example of unique feature absence problem

We identified problems caused by not having enough feature points for scan matching, e.g., when the car is located in a curve. This lack of landmarks causes a destructive localisation drift as the IMU error accumulates. In some cases, lane markings are visible but they hold a monotonic shape, e.g., a completely straight road where the car moves but the lane markings in the scene remain the same. In cases like this, the scan matching process has no chance to derive the movement offset from the lane markings and the surrounding submap.

We solved this problem by adding more lane markings as *support lanes* to the track. This makes our model of the environment further away from the real scenario. On the other hand, real urban roads often contain more lane markings as well (arrows, crossings, etc.). The support lanes provide enough feature points for the scan matching algorithms to work. This allowed us to continue using lane markings as the only features while not having to rely just on the data from IMU.

An alternative solution might be to use more cameras or other sensors, which have a longer range or larger angle of view. This would help with estimating the movement even when there is a lack of visual features in front of the car. Unfortunately, multiple cameras are not feasible in our case as discussed in Section 4.5.

Chapter 7

Experiments

In this chapter, we present the results of the localisation and mapping of a car on our testing tracks. First, we show the performance of local SLAM. Further, we focus on the influence of global SLAM, which adjusts the alignment of the submaps estimated by local SLAM.

For the experiments, we use two different tracks, which are described in detail in Section 3.2. The main difference is in the track length and the portion of turns on the track. The asymmetric track is twice as long as the oval circuit. The oval circuit has much higher portion of curves. At curves, the camera changes its field of view much quicker than on straight parts. Comparing the performance results on these two tracks shows us how the error accumulation is influenced by the curve frequency.

The main experiments are conducted using configuration of Cartographer with the point cloud camera input. We acquired reference values of localisation using LIDAR-based SLAM on the recorded sensor data. We also present example results from combined SLAM and camera SLAM using 2D LIDAR-like scanning of the camera images. All configurations also make use of the data from IMU for pose estimation.

We repeated each experiment at a different speed level to show how speed influences the localisation and mapping error. The average velocities for individual experiments on both tracks are presented in Table 7.1. The highest speed level for both tracks is highlighted in red. At this speed level, the localisation drift proved to be too severe for local SLAM to be able to build consistent submaps. We only show the partial results of these experiments to show how the performance deteriorates when the velocity is too high. There is no point in trying to estimate, e.g., the global mapping error, when the localisation diverges so severely. For other experiments, all results (using the metrics described in Section 6.3) are presented and discussed in the sections below.

Asymmetric circuit	
Speed level	**Average velocity [m/s]**
Low	0.45
Medium	0.75
High	1.25
Maximum	1.88

Oval circuit	
Speed level	**Average velocity [m/s]**
Low	0.44
Medium	0.71
High	1.32

Table 7.1: The average speed used in the individual experiments

The accuracy of measurements in the map is limited to centimetres due to the resolution

of the occupancy map grid. A higher resolution is not available because it causes a significant rise in the computational cost.

7.1 Local SLAM

Without corrections, local SLAM inevitably accumulates a localisation error over time but it should keep the submaps locally accurate. Below, we show the results of local SLAM performance in different conditions. The goal is to test under which conditions the local SLAM is capable of reaching its goal of building accurate submaps, which can then be post-processed and have their alignment corrected by global SLAM.

We estimate the local mapping error using the landmarks. We quantify the localisation error by comparing the results with the reference solution. We also estimate the global mapping error caused by the local SLAM localisation drift, which accumulates over time due to the incorrect prior submap alignment. We also discuss the impact of varying track complexity and the speed level of the vehicle on the accuracy.

7.1.1 Asymmetric circuit

In Figure 7.1, we can see a map of the asymmetric track including highlighted landmarks, support lanes and the car path. The landmarks are extra lane markings in an 'X' shape for the purpose of estimating the mapping error. The support lanes are the added lane markings for the purpose of generating more features for scan matching as discussed in Section 6.3.3. This is a referential map created by combined SLAM (using both the camera and LIDAR).

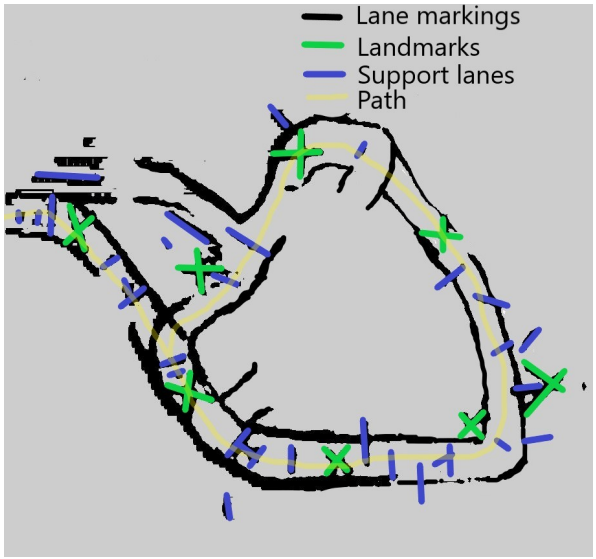
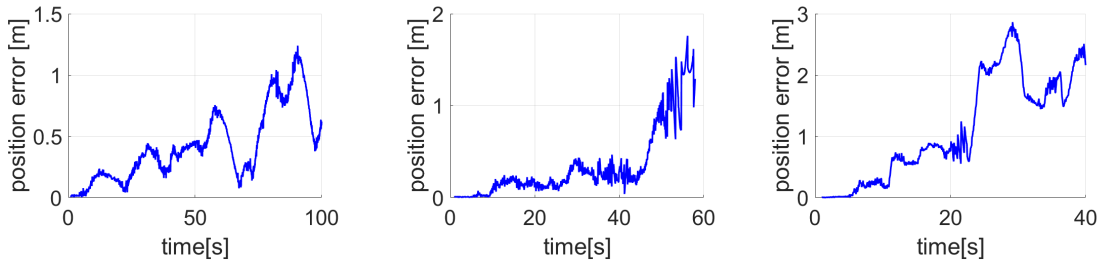


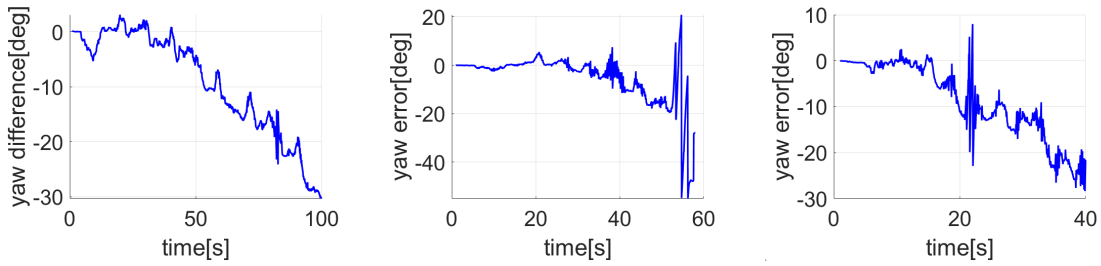
Figure 7.1: Asymmetric track - a combined SLAM map

In Figure 7.2, we present the position estimation error on the asymmetric circuit measured in experiments with different speed levels. Corresponding orientation estimation errors are shown in Figure 7.3. The presented results are of the experiments, where the car drove 3 laps.



(a) Low speed level (b) Medium speed level (c) High speed level

Figure 7.2: Position estimation error on the asymmetric circuit



(a) Low speed level (b) Medium speed level (c) High speed level

Figure 7.3: Orientation estimation error on the asymmetric circuit

We can see that over time a significant pose estimation error is accumulated (as shown in Figure 7.2 and Figure 7.3). The position estimation error is worse in cases with a higher speed level even though the travelled distance is the same. For a total travelled distance of around 45 metres, the worst position error accumulated is from 1.2 m to 2.9 m based on the speed level. The errors have a sharp development, especially for higher speed levels.

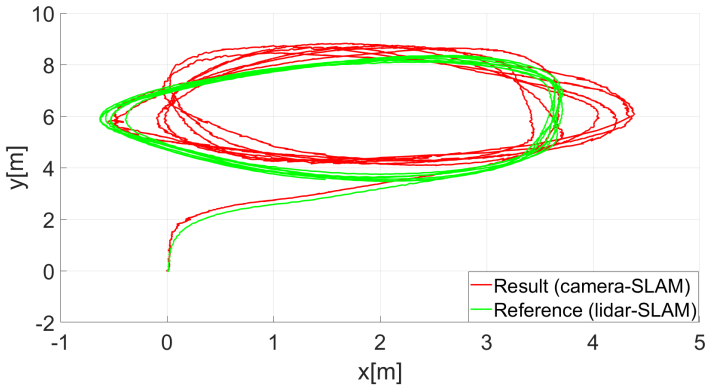
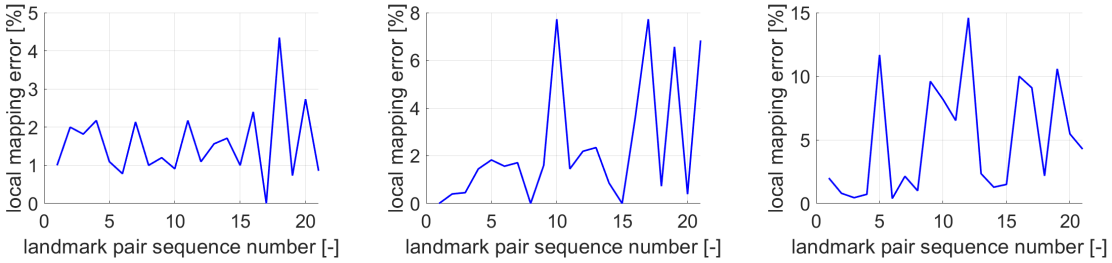


Figure 7.4: The consecutive laps drift example of Local SLAM

In Figure 7.3, the orientation estimation error shows that over time the maps begin to shift by a certain degree because of the localisation drift. This localisation drift is visualised in the form of a path comparison between the results and the reference in Figure 7.4. Since local SLAM does not correct the drift, the path of the consecutive laps shifts from the original position. This shift causes the position error to be higher in certain areas of the lap and lower in others where the estimation crosses the actual position due to its connection to the orientation error.

In Figure 7.5, we show the local mapping error development over multiple laps for different speeds.



(a) Slow speed (b) Medium speed (c) High speed

Figure 7.5: Asymmetric circuit - local mapping error

We can see that the error significantly rises with the rising speed level. There is a big difference between the mapping accuracy of the individual pairs of landmarks. This is even more visible at higher speed levels. In Table 7.2, we can see the averages and the maxima of the local mapping error. We use an occupancy grid with a resolution in centimetres. Given that, the error of few centimetres spread over a distance of meters could not cause a significant inaccuracy to the submap building process. But for the higher speed levels the maximum of 40 cm is dangerously high and may cause the submaps to be too inaccurate to achieve a consistent global map.

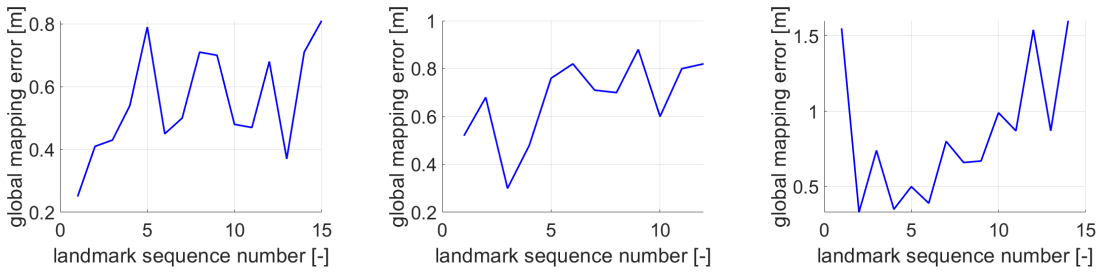
Speed level	**Average error [cm]**	**Maximum error [cm]**
Low	3.3	7.0
Medium	5.5	18.0
High	12.0	40.0

Table 7.2: Asymmetric track - local mapping error

In Figure 7.6, the global map consistency error is shown for different speeds. This error corresponds to the mapping error accumulated over a single lap of the circuit. We can see that local SLAM accumulates an error of 0.8 m to 1.5 m based on the speed level.

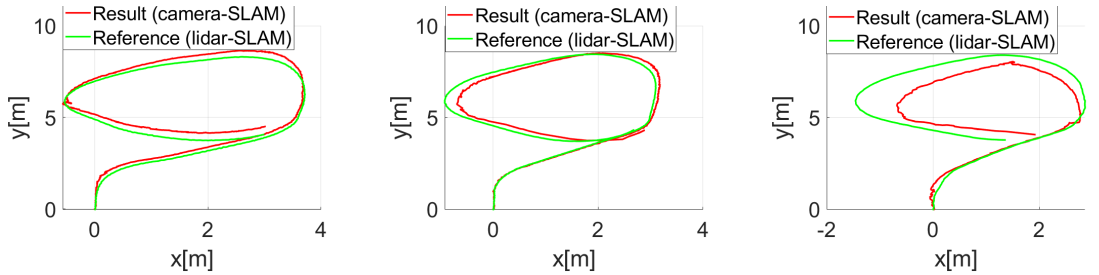
To show how the localisation drift affects the path for different speeds in a more readable format we show the paths of the first lap compared to the reference in Figure 7.7. We can see that higher speeds cause the path to shift from the reference with the increasing number of turns.

A higher speed level causes the errors to rise significantly. This is due to the quick change



(a) Low speed level (b) Medium speed level (c) High speed level

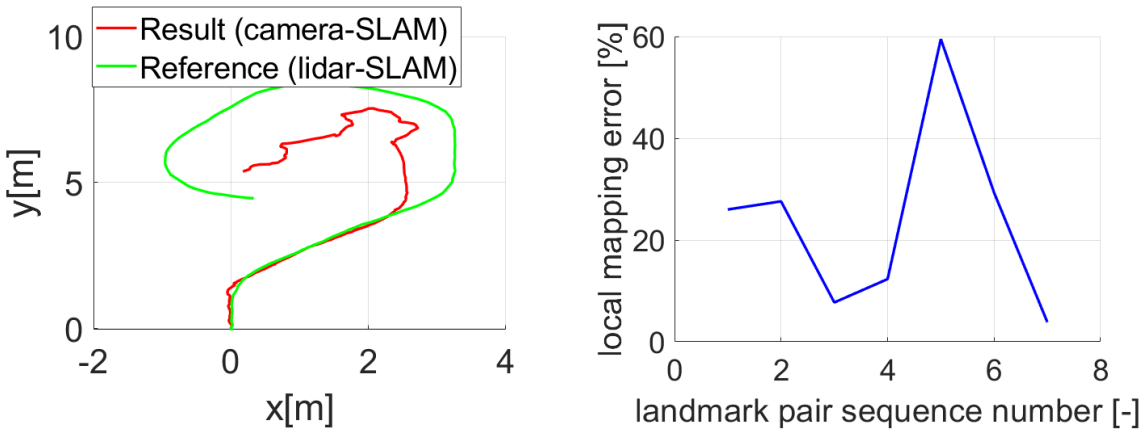
Figure 7.6: Asymmetric circuit - global mapping error



(a) Low speed level (b) Medium speed level (c) High speed level

Figure 7.7: Asymmetric circuit - visualisation of estimated path

of the field of view that comes with higher speed. A low overlap in consequent images means that the scan matching is more likely to estimate an incorrect pose and accumulate an error. At the maximum speed level, this effect is so pronounced that the localisation accumulates a big offset after the first turn as shown in Figure 7.8a. At the second turn, the localisation drift gets completely lost and is unable to form a consistent global map. At this speed, local SLAM is not even capable of producing consistent submaps, which translates into a large local mapping error in turns as shown in Figure 7.8b.

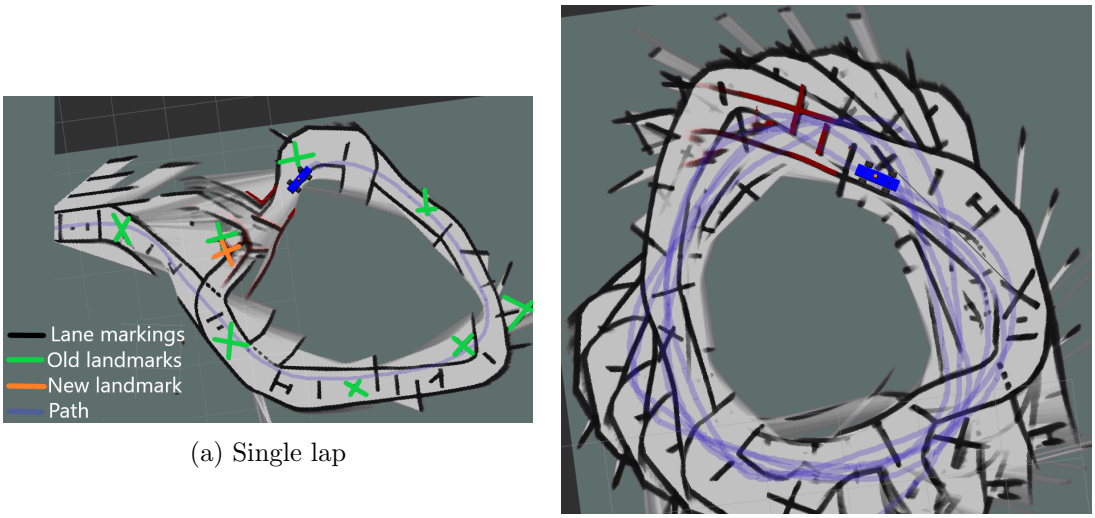


(a) Path drift visualisation (b) Local mapping error (1 lap)

Figure 7.8: Asymmetric circuit - maximum speed level

For a direct visualisation of the localisation drift effect on the map see Figure 7.9a. It shows the mapping offset of a landmark created during 1 lap. This offset is the result of the localisation error accumulated over time by local SLAM.

Without using a loop closing algorithm this offset cannot be corrected and the localisation drift continues to accumulate as can be seen in the multiple laps localisation visualisation in Figure 7.4 or in the multiple laps map drift visualisation in Figure 7.9b. We can see here that during multiple laps the submaps and the localisation drift remain almost identical. This points to the fact that the submap alignment should be able to correct this and create a consistent global map.



(a) Single lap

(b) Multiple laps

Figure 7.9: Local SLAM - a map drift example on the asymmetric circuit

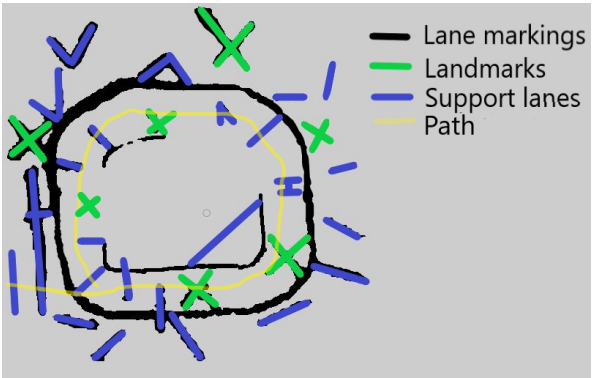


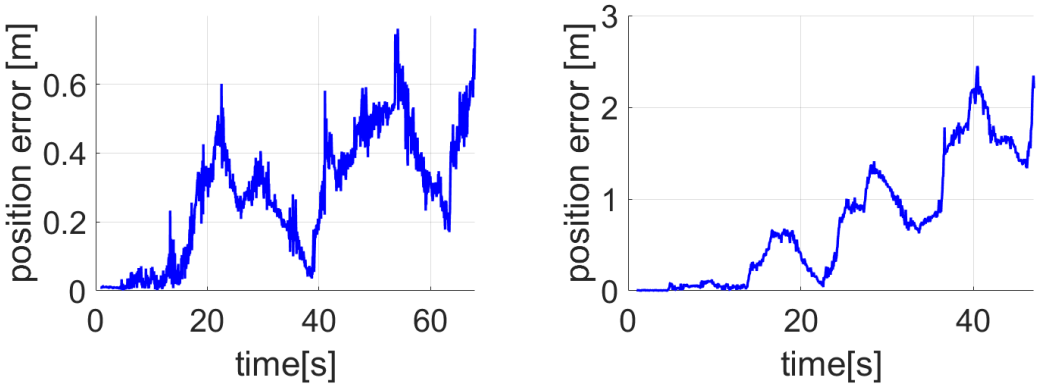
Figure 7.10: Oval track - a combined SLAM map

7.1.2 Oval circuit

In Figure 7.10, we can see a map of the oval track with highlighted landmarks. This is a referential map created by combined SLAM. We can see that a big part of the inner lane

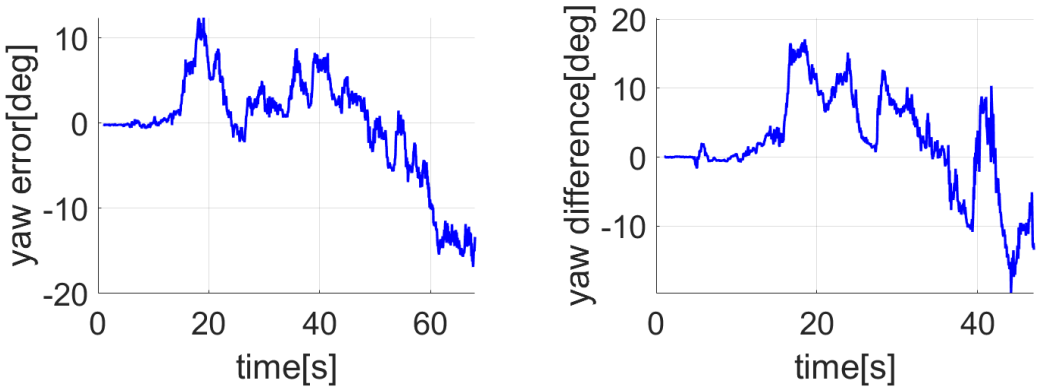
markings of the circuit never get into the field of view due to the limited field of view and the high curvature of the track.

In Figure 7.11, we present the position estimation error on the oval circuit measured during experiments with different speed levels. Corresponding orientation estimation errors are shown in Figure 7.12. The presented results are from experiments that are 3 laps long.



(a) Low speed level (b) Medium speed level

Figure 7.11: Oval circuit - position estimation error



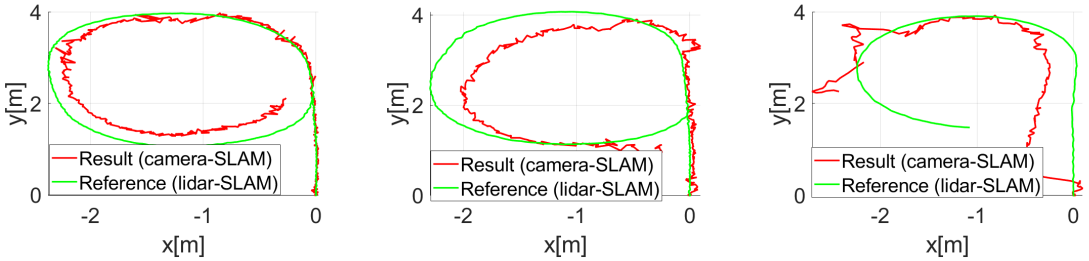
(a) Low speed level (b) Medium speed level

Figure 7.12: Oval circuit - orientation estimation error

To show how the localisation drift affects the path at different speed levels in a more easily understandable format we show the paths of the first lap compared to the reference in Figure 7.13. Similarly to the results on asymmetric circuit, we can see that higher speed causes the path to shift from the reference with the increasing number of turns.

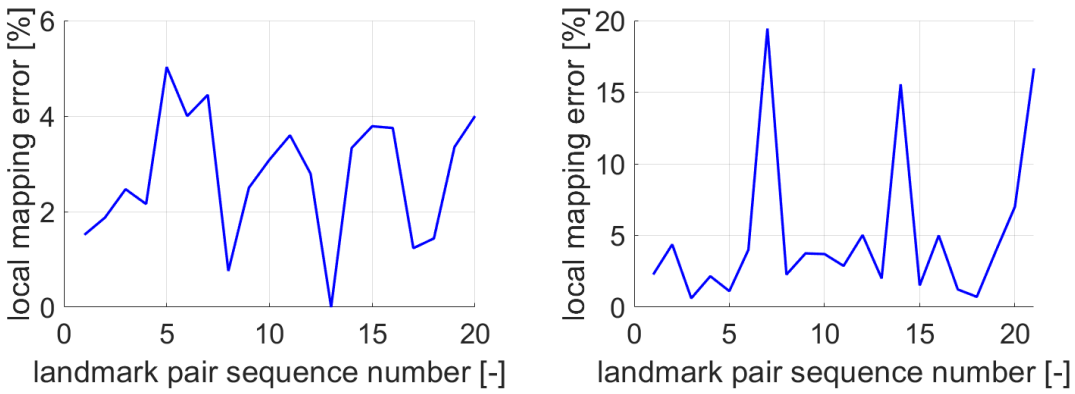
In Figure 7.14, we show the local mapping error development over multiple laps for different speeds.

Similarly to the experiments on the asymmetric track, we can see that the local mapping error significantly rises with the speed level. There is an even bigger difference between the mapping accuracy of the individual pairs of landmarks than is in case of the asymmetric



(a) Low speed level (b) Medium speed level (c) High speed level

Figure 7.13: Oval circuit - a visualisation of the estimated path



(a) Low speed level (b) Medium speed level

Figure 7.14: Oval circuit - local mapping error

circuit. In Table 7.3, we can see the averages and maxima of the local mapping error. We can see that the local mapping error of the medium speed level in experiments on the oval circuit is similar to the local mapping error of the high speed level in the asymmetric circuit.

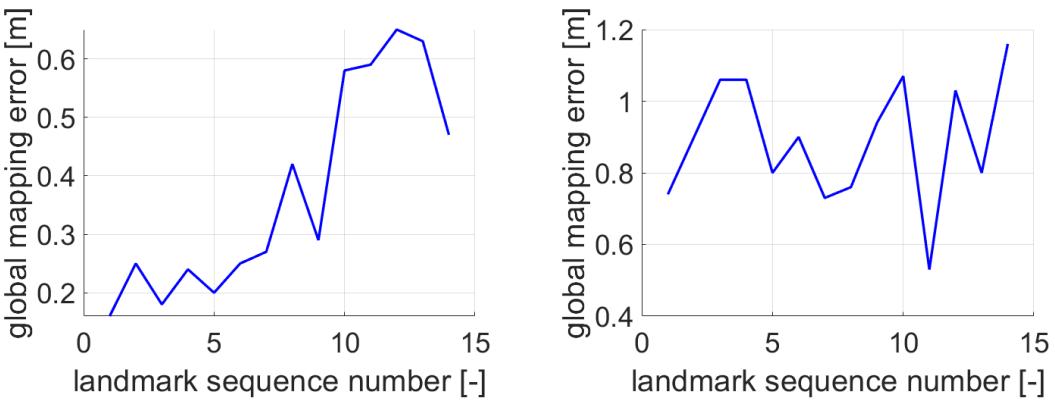
Speed level	**Average error [cm]**	**Maximum error [cm]**
Slow	4.6	12.0
Medium	8.1	35.0

Table 7.3: Oval circuit - local mapping error

We can see three big spikes of the local mapping error in Figure 7.14 that occured during the experiment at the medium speed level. A closer look shows us that all of these spikes correspond to a single pair of landmarks mapped in individual laps. There is a sharp turn between these two landmarks and the given speed level is too high for the scan matching to handle the visual field change. As a consequence, the submap of that area suffers from high inaccuracy, which translates into a high local mapping error.

In Figure 7.15, the global map consistency error is shown for different speeds. This error corresponds to a mapping error accumulated during a single lap of the circuit. We can see that local SLAM accumulates a global mapping error from 0.6 m to 1.2 m based on the speed level. Since the oval circuit is almost two times shorter, it means that local SLAM accumulated a

significantly larger mapping offset per unit of distance than on the asymmetric circuit.



(a) Low speed level

(b) Medium speed level

Figure 7.15: Oval circuit - global mapping error

In Figure 7.16, we can see how the global mapping error translates into a map offset between one area being mapped in consecutive laps. The newly mapped landmark is highlighted in orange close to the original landmark mapped in green.

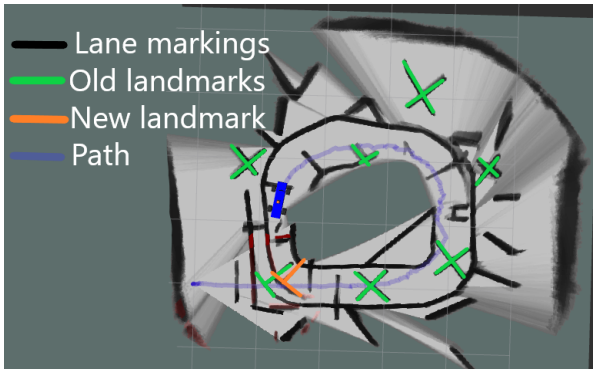


Figure 7.16: Oval circuit - Local SLAM map offset

We can see that the errors accumulated during experiments on the oval circuit are bigger than those measured on the asymmetric track when we normalize them by the total length. This is due to a higher portion of curves, which causes a quick change of the visual field that supports accumulation of the localisation error. If the change of the visual field between frames is too large, it causes local SLAM to get lost and create a significant local mapping error, which is destructive for the whole SLAM. Because of this, local SLAM's performance on the oval track is more sensitive to a higher speed than on the asymmetric track, where it has long passages that it can map ahead of the curve. On the oval circuit, consecutive images contain less overlap. This causes the performance of local SLAM on the oval circuit to be worse than the performance on the asymmetric track for the same speed level.

7.1.3 Results

When we look at the pose estimation error in all speed level and track cases, we can see that both the orientation and position error rises as the local SLAM accumulates drift. Locally, the error oscilates during individual laps. If we look at the path comparisons (Figure 7.7 and Figure 7.13), we can see that in some areas the localisation gets close to the reference. Due to, e.g., an incorrect orientation, it crosses the reference's path and then builds the position error again.

Local SLAM clearly fails to maintain a low localisation and mapping error at high speed levels in turns.

The local mapping accuracy oscilates because it is influenced by the local track difficulty (we can see landmark pairs around curves have a significantly higher local mapping error) and by current speed. The local mapping error does not accumulate over time with repetition of the laps.

The local mapping accuracy decreases with the higher speed level and higher track difficulty as shown in Figure 7.5 and in Figure 7.14. These results show us that the local mapping error stays under 5% if F1/10 is driving at the low speed level in sharper turns and up to the medium speed level in straight parts of the circuit. A low local mapping error is necessary for the loop closing algorithms to be able to create a consistent global map. The results of Local SLAM point to the fact that for the lower speed levels, the submaps are accurate enough for loop closing to be possible. This is tested further in Section 7.2.

The global mapping error (which signifies the mapping error accumulated over a whole lap) is significant and therefore we can see that Local SLAM alone is not enough to build a globally consistent map that would converge over time.

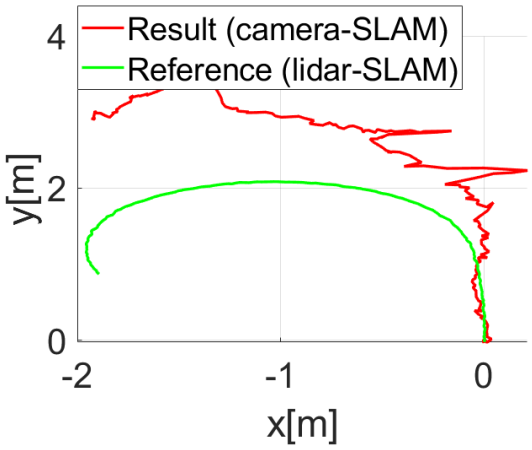


Figure 7.17: 2D LIDAR-like scanning of images - the estimated path on the oval circuit compared to the reference

We also experimented with 2D LIDAR-like scanning of images from the camera to see how a lower amount of feature points impacts the localisation and mapping process. We conducted experiments on the oval track without support lanes and landmarks (as they would

block the line of sight to further feature points, for an illustration see Figure 4.3). The results showed that the localisation suffers from an insufficient number of feature points for scan matching. Even for a low speed level, local SLAM accumulates a large localisation error in each turn as shown in Figure 7.17. This shows us that the amount of provided points is a critical factor for succesful localisation.

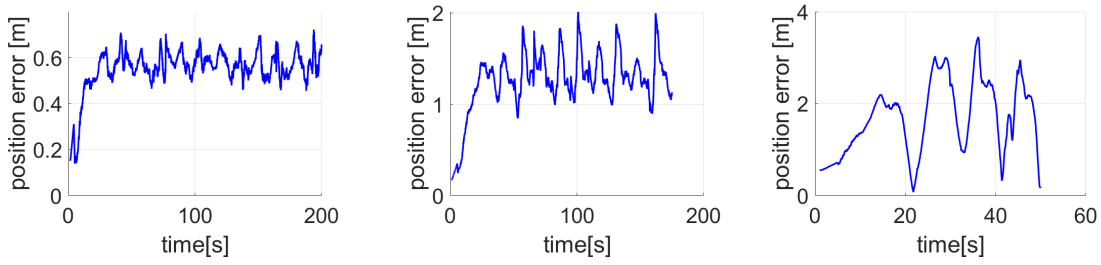
7.2 Global SLAM

In this section, we focus on testing the capability of global SLAM to correct the submap alignment estimated by local SLAM. We use the same speed levels and tracks as the ones used in local SLAM. For global SLAM experiments we add more laps to properly test the effect of loop closing on repeatedly visited areas.

To quantify the performance of global SLAM, we use the post-processed localisation error. Because of optimization, there is no point in using metrics that are measured during experiments as discussed in Section 6.3. We also present the resulting maps, as the most important sign of successful loop closure is a converging map and a converging path (by this we mean that the mapped locations and the estimated path in consecutive laps are consistent as opposed to being affected by the localisation drift). All metrics using the reference are unfortunately not exact measures because of the global optimization, which tends to shift the whole plane slightly as explained in Section 6.3.

7.2.1 Asymmetric track

Figure 7.18 shows the resulting position errors for different speed levels. Corresponding orientation errors are presented in Figure 7.19.

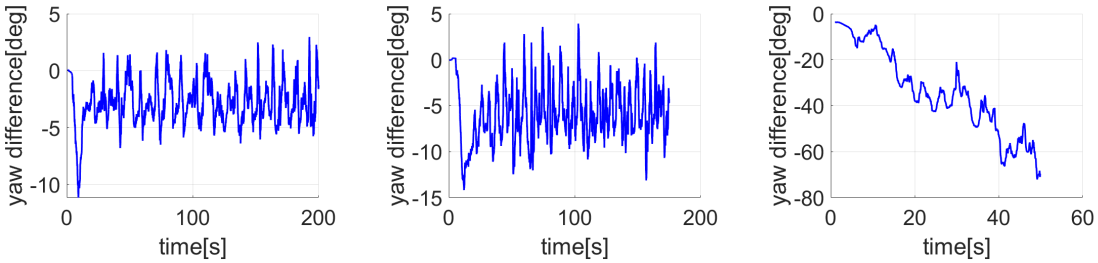


(a) Low speed level (b) Medium speed level (c) High speed level

Figure 7.18: Global SLAM - position error on the asymmetric circuit

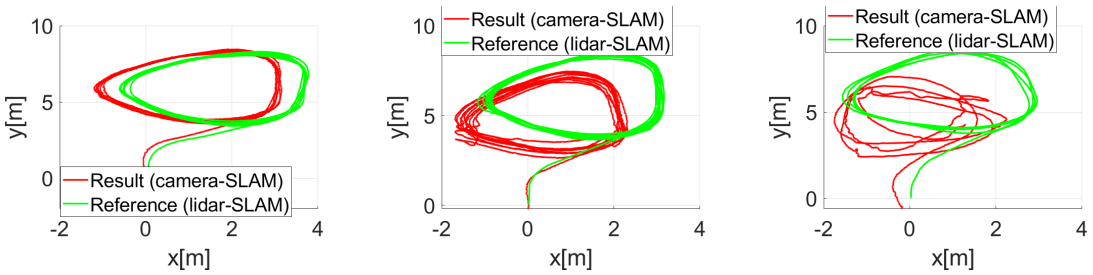
We can see that both the position and orientation error do not accumulate for low and medium speed levels compared to the results of local SLAM. This shows that the loop closure is successful in negating the effect of the localisation drift. The position error remains quite high but we cannot interpret these values without comparing the paths first.

We visualised the paths of both the reference and the solution in Figure 7.20.



(a) Low speed level (b) Medium speed level (c) High speed level

Figure 7.19: Global SLAM - orientation error on the asymmetric circuit



(a) Low speed level (b) Medium speed level (c) High speed level

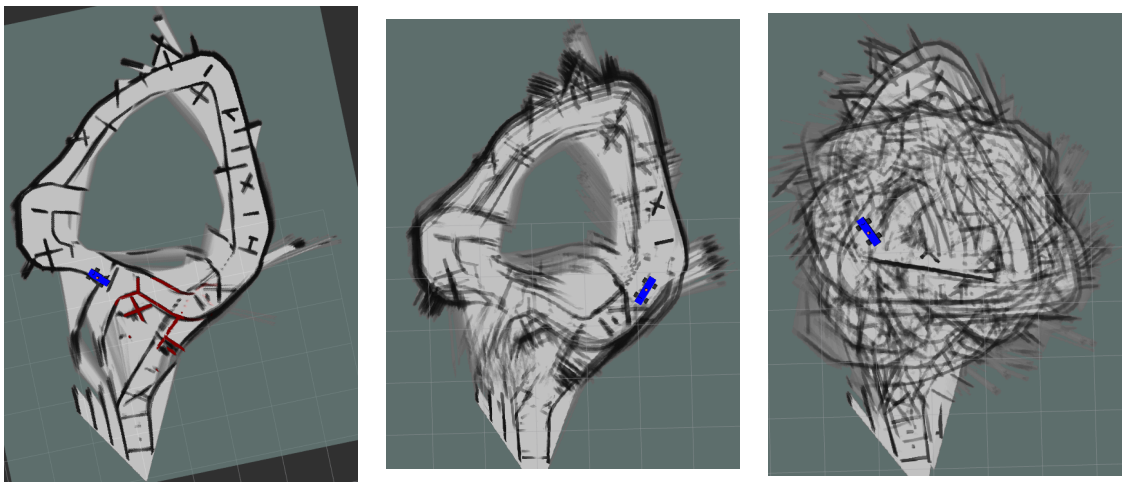
Figure 7.20: Final path the visualisation asymmetric circuit

We can see in the results that the paths converge as opposed to the results of local SLAM (for comparison see the multilap example Figure 7.4). The position and orientation error is mostly caused by the offset that was generated between our solution and the reference due to the optimalizations.

We can see that the paths of the solution and the reference look identical for the low speed level (apart from the offset). For the medium speed level, the resulting path is slightly wider spread than the reference and a closer look shows a few discontinuities in the sharp curve on the left. Otherwise, the path keeps a consistent shape. For the high speed level, we can see that loop closing no longer manages to correct paths and they diverge due to the localisation drift.

For a better illustration of the influence of a speed level on the performance of global SLAM , we compare the created maps in Figure 7.21.

We can see that the map for the low speed level is very neat compared to the maps for higher speed levels. As the speed level rises, the accuracy of the submaps decays. This makes it impossible to find a perfect alignment, which would create a consistent global map with sharply displayed landmarks. At the highest speed level, the map completely diverges as the inaccuracies of submaps are too high for loop closing to create a consistent global map.

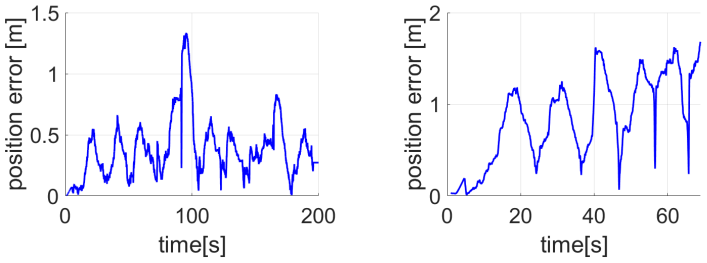


(a) Low speed level (b) Medium speed level (c) High speed level

Figure 7.21: Map samples of the asymmetric circuit constructed by Global SLAM

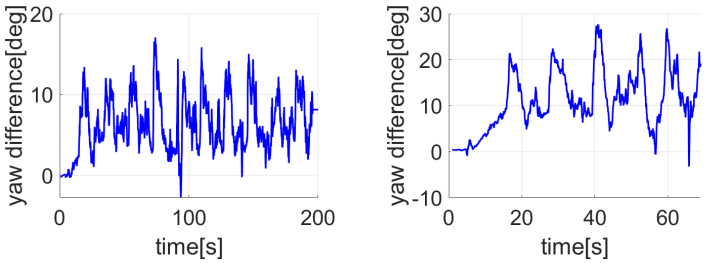
7.2.2 Oval circuit

We use the same approach to evaluate the experiments on the oval circuit. The position and orientation error are presented in Figure 7.22 and Figure 7.23.



(a) Low speed level (b) Medium speed level

Figure 7.22: Global SLAM - position error on the oval circuit



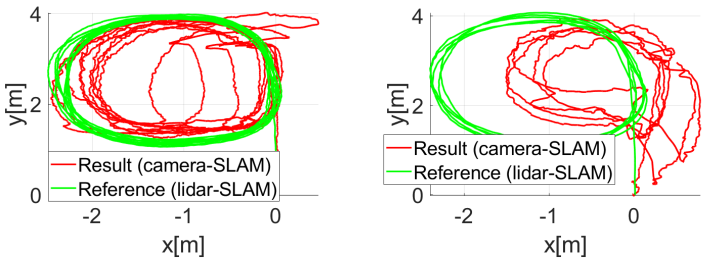
(a) Low speed level (b) Medium speed level

Figure 7.23: Global SLAM - orientation error on the oval circuit

Just like in the case of the asymmetric circuit, we can see that Global SLAM corrects

the localisation drift here. An offset between the solution and the reference remains but it no longer accumulates.

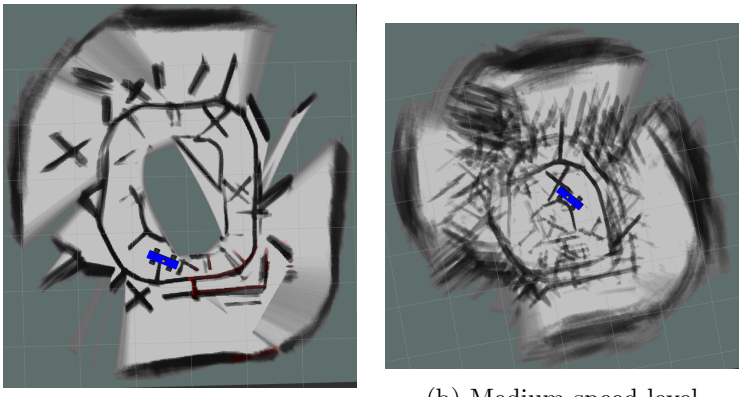
Thanks to that, the path at the low speed level remains almost consistent as shown in Figure 7.24 though path during one lap had diverged. The path at the medium speed level shows sever discontinuities even though it almost converges.



(a) Low speed level (b) Medium speed level

Figure 7.24: Final path estimation visualisation on the oval circuit

In Figure 7.25, we can see a comparison of maps created at different speed levels.



(a) Low speed level (b) Medium speed level

Figure 7.25: Map samples of the oval circuit constructed by Global SLAM

The map for the medium speed level is severely distorted. Clearly, the loop closing process is damaged by the local mapping inaccuracy, which is quite high for this combination of speed level and portion of curves as shown in Figure 7.14. The map for the low speed level is consistent. On the other hand, some of the features are slightly blurred.

7.2.3 Summary

For the combination of lower speed level and portion of curves, Global SLAM manages to correct the localisation drift and create a clear converging global map. For the medium speed level on the asymmetric track or the low speed level on the oval track, the loop closing optimization manages to keep a consistent path estimation and consistent mapping in

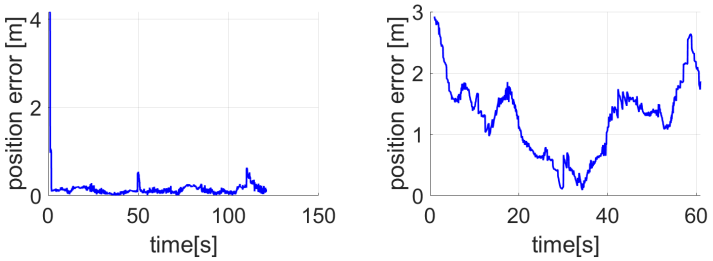
consecutive laps. But the local inaccuracies are high enough to cause the global map to be blurred. For the more difficult combinations of speed level and track difficulty, loop closing does not manage to stop the maps of consecutive laps from diverging and for the localisation to drift. We cannot clearly quantify the error of localisation because the optimization causes the global map to "float", and, therefore, it cannot be easily compared with the referential results.

The effect of loop closing is also illustrated in a demonstrative video showing the whole mapping process and the image processing, which is on the CD attached to this thesis (for CD content see Table 1).

7.3 Pure localisation mode

Finally, we test our system in a pure localisation mode in a previously obtained map. To quantify the quality of this process, we use the pose estimation error prior to the optimization as discussed in Section 6.3. The maps that we use for our pure localisation mode are the ones generated by global SLAM in experiments at the lowest speed level (the maps are depicted in Figure 7.21a and Figure 7.25a). Therefore, in this section, we also verify the usability of the maps created by camera-based SLAM. There is no trivial way of linking LIDAR-based localisation in an obstacle map to a globally optimized camera-based map. Instead of the LIDAR reference we use the path created by the camera-based global SLAM as the reference for the experiments at the low speed level, where we have it available.

Figure 7.26 and Figure 7.27 show the resulting pose estimation error throughout the localisation process.

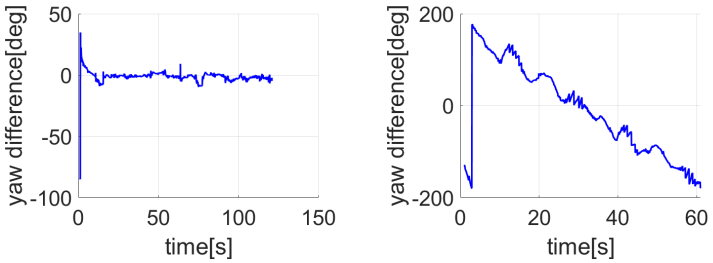


(a) Asymmetric track (b) Oval circuit

Figure 7.26: Pure localisation mode - position error at the low speed level

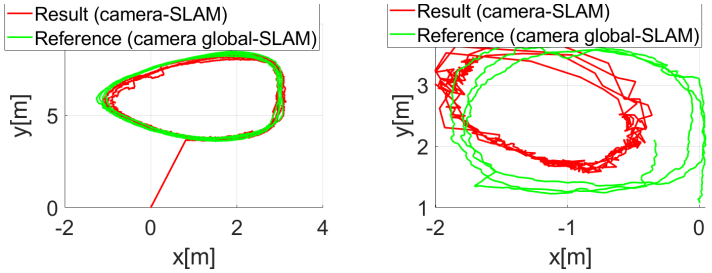
We can see that the results are very different for the asymmetric track and the oval circuit. On the asymmetric track, the localisation process managed to find the car's current position (the position error dropped under 0.2 m) on the circuit in under 2 s. The final average position error is 0.144 m with an average orientation error of 2.446° . On the other hand, on the oval circuit, the pose estimation error remained high throughout the experiment with an average position error of 1.265 m and an average orientation error of 85.21°.

We show a comparison between the solution and reference paths in Figure 7.28.



(a) Asymmetric track (b) Oval circuit

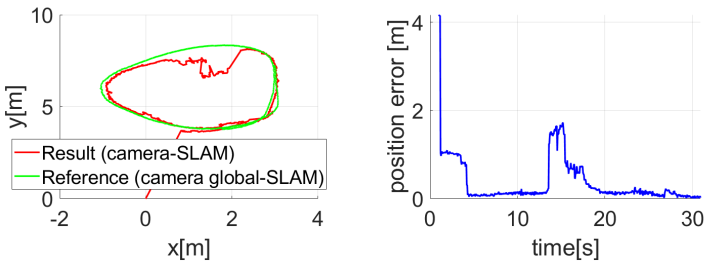
Figure 7.27: Pure localisation mode - orientation error at the low speed level



(a) Asymmetric track (b) Oval circuit

Figure 7.28: A path visualisation on the asymmetric circuit

We analyzed this striking difference in performance between the tracks and came to the following conclusion. The pure localisation mode is extremely sensitive to the local mapping accuracy. We caused this sensitivity by setting the size of the submaps to a 5 times higher value than in the rest of our camera-based Cartographer configurations. We had to do this because the pure localisation mode is set to remember only 3 submaps and that did not include enough features to keep the localisation stabilized. With smaller submaps, the localisation found the correct pose most of the time. But once the vehicle gets to a difficult turn or to a monotonous part of the circuit that does not have so many features, the localisation frequently "jumps" and creates a big pose error as shown in Figure 7.29.



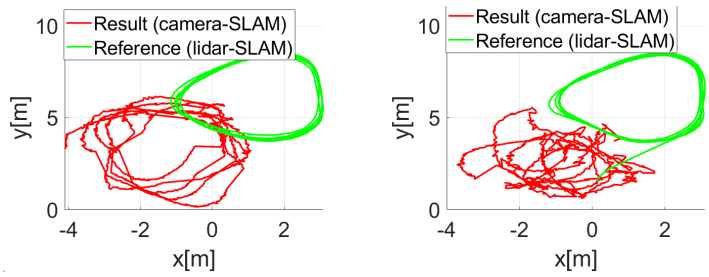
(a) Path comparison (b) Position error

Figure 7.29: Pure localisation mode - unstable behavior of the localisation in the smaller submap size configuration

As shown in Section 7.1, the local mapping accuracy is significantly worse on the oval track than on the asymmetric track. And, as shown in Section 7.2, the map created by global

SLAM on the oval circuit is not as sharp as the one built on the asymmetric track. Observation proved that these bigger submaps created during the localisation process on the oval track are inaccurate, which blocks the pure localisation mode from creating a converging path.

Figure 7.30 shows us the paths comparison with the LIDAR reference for higher speed levels. We can see that rising speed causes the localisation to shake and for the highest speed, the path diverges completely.



(a) Medium speed level (b) High speed level

Figure 7.30: Pure localisation mode - path comparison on the asymmetric circuit

7.3.1 Summary

The pure localisation mode clearly fails to estimate an accurate path for a higher speed difficulty and a higher track complexity. For the medium speed level on asymmetric track or the low speed level on the oval track, it holds a similar path in consecutive laps but it is far from accurate. Its accuracy is highely dependent on the accuracy of the submaps. For the low speed level on the asymmetric track, the localisation is accurate, converges quickly after the start, and keeps a consistent path.

Chapter 8

Conclusion

We have investigated the capability of camera-based navigation using lane markings as features for map creation. The detection of lane markings proved to be robust under normal light conditions. However, in cases of strong local illumination our camera becomes overexposed and the car loses most of its capability to orient itself in such an area. Our lane markings detection does not use a generalising approach, and, therefore, it does not interpolate the data in cases where the lane markings are partially damaged, missing, or the line of sight is blocked by other objects.

The map fragment acquisition from the detected landmarks showed the highest reprojection error on the reference correspondences of 3.61 cm and an average error of 0.69 cm while stationary or moving at constant speed. This is a low enough reprojection error for our application because our map grid has a 1 cm resolution.

We assumed that the mapping process is conducted mostly at a near-constant velocity. However, a strong acceleration of the car causes the camera's pitch change with respect to the ground plane. This causes further inaccuracies in the map fragment acquisition process as the distances in the direction of bearing would no longer be correctly reprojected using our static homography approach.

The visual field of the camera that we used is quite small (for details see Section 3.1.1) compared to usual sensory equipment used by applications striving for a functional SLAM. Due to this, we used support landmarks (additional lane markings) to gain enough features for scan matching (as discussed in Section 6.3.3).

We used Cartographer for the localisation and mapping of the detected features. The experiments showed us that the mapping accuracy is very sensitive to the sudden changes of the visual field, which is mostly connected to the velocity and track complexity. Local SLAM is capable of creating accurate submaps for lower speed levels at curves and medium speed levels in the straight parts of the track. In the cases which combine higher speed levels and curves, we observed an increase in the local mapping error, which destroys the global consistency of the submaps. As we expected, local SLAM accumulates a pose estimation error over time, which has to be corrected by loop closing after revisiting the areas of the track.

Loop closure performed by global SLAM proved to be effective in correcting the local SLAM's incorrect submap and past path alignment. For the low speed level and low track complexity, the global map perfectly converged and the estimated path shape was identical to the reference. At the medium speed level or in the case of higher track complexity combined with a low speed level, global SLAM still managed to create a converging global map, al-

though, especially the distant landmarks are visibly blurred due to submap inacurracies. The localisation did not accumulate an error. For a more difficult combination of conditions, the localisation drift was too severe and global SLAM failed to create a consistent global map.

We also tested localisation in a map (constructed by camera-based global SLAM) using Cartographer's pure localisation mode. The pure localisation mode was not able to avoid accumulating an error. On the oval track (or for higher speed levels on the asymmetric track) it did not converge into the correct path. On the other hand, we achieved accurate results for the low speed level on the asymmetric track. For the low speed level on the asymmetric track, the position estimation error converged under 0.2 m in under 2 s. The average position error was 0.14 m.

8.1 Future work

Finally, we summarize improvements and ideas for further enhancement of the current state of the system presented in this thesis.

8.1.1 Lane markings detection

The lane markings detection could be made more robust towards strong illuminance conditions, if we fixed the camera's incompability with the computing module (described in Section 5.2). This would enable us to use an automatic exposure time setting instead of a fixed one, which would decrease the occurance of overexposure. A lower exposure time could also enable us to have sharper images even at higher speed levels and possibly a higher framerate.

To make the lane markings detection generalize beyond the currently visible parts of the lane markings, it would be valuable to add a geometrical model of it using a Generalised Hough Transform, e.g., a model using B-splines similarly to [51], which are flexible enough to describe the almost arbitrary shapes of the lane markings. However, this would cause a high computational cost and would require a lot of computational power.

8.1.2 Map fragment acquisition

In this section, we discuss two main enhancements, which could enable us to gain more data from the individual frames to be capable of localisation without supporting lane markings.

Adding more cameras or a camera with a larger field of view would enable us to collect more features of the surroundings, which is valuable for localisation.

Another way of getting more data for localisation could be to use visual feature descriptors (e.g., SIFT), which can be uniquely recognized and can provide the vehicle with data about the relative orientation and its distance to them. This would mean that we would have

to store them in a database of visual landmarks (similarly to application of visual SLAM [29]) instead of just a simple occupancy map grid.

8.1.3 Localisation and mapping

To make local SLAM more robust to higher speed and track complexity, we could attempt to raise the update rate that is currently limited to 20 Hz by the camera. A higher update rate could help lower the negative effect of the sudden change of the visual field in the turns.

Extracting reliable visual odometry from consecutive camera images could also help Cartographer stabilize localisation as it would be another source of data for the vehicle's pose estimation.

Bibliography

[1] Anubhav Agarwal, CV Jawahar, and PJ Narayanan. A survey of planar homography estimation techniques. *Centre for Visual Information Technology, Tech. Rep. IIIT/TR/2005/12*, 2005.

[2] Sameer Agarwal, Keir Mierle, and Others. Ceres solver. `http://ceres-solver.org` last visited on 12.7.2020.

[3] Mitra Basu. Gaussian-based edge-detection methods-a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 32(3):252–260, 2002.

[4] Keshav Bimbraw. Autonomous cars: Past, present and future a review of the developments in the last century, the present scenario and the expected future of autonomous vehicle technology. In *2015 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, volume 1, pages 191–198. IEEE, 2015.

[5] Black Ice Software, LLC. HSI Ccolor Space. `https://www.blackice.com/colorspaceHSI.htm` last visited on 3.5.2020.

[6] Robert C Bolles and Martin A Fischler. A ransac-based approach to model fitting and its application to finding cylinders in range data. In *IJCAI*, volume 1981, pages 637–643, 1981.

[7] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[8] Duane C Brown. Decentering distortion of lenses. *Photogrammetric Engineering and Remote Sensing*, 1966.

[9] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

[10] Kuo-Yu Chiu and Sheng-Fuu Lin. Lane detection using color-based segmentation. In *IEEE Proceedings. Intelligent Vehicles Symposium, 2005.*, pages 706–711. IEEE, 2005.

[11] Roberto Cipolla, Tom Drummond, and Duncan P Robertson. Camera calibration from vanishing points in image of architectural scenes. In *BMVC*, volume 99, pages 382–391, 1999.

[12] Civil Seek. Perspective Projection. `https://civilseek.com/perspective-projection-drawing/` last visited on 3.5.2020, 2020.

[13] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007.

[14] Vinayak V Dixit, Sai Chand, and Divya J Nair. Autonomous vehicles: disengagements, accidents and reaction times. *PLoS one*, 11(12):e0168054, 2016.

[15] Arnaud Doucet, Nando De Freitas, Kevin Murphy, and Stuart Russell. Rao-blackwellised particle filtering for dynamic bayesian networks. *arXiv preprint arXiv:1301.3853*, 2013.

[16] Elan Dubrofsky and Robert J Woodham. Combining line and point correspondences for homography estimation. In *International Symposium on Visual Computing*, pages 202–213. Springer, 2008.

[17] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. *IEEE Robotics & Automation Magazine*, 13(2):99–110, 2006.

[18] Wael Elloumi, Sylvie Treuillet, and Remy Leconge. Real-time estimation of camera orientation by tracking orthogonal vanishing points in videos. volume 2, 02 2013.

[19] Ferran Espuny, Joan Aranda, and José Burgos Gil. Camera self-calibration with parallel screw axis motion by intersecting imaged horopters. volume 6688, pages 1–12, 05 2011.

[20] Andrew W Fitzgibbon. Simultaneous linear estimation of multiple view geometry and lens distortion. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I. IEEE, 2001.

[21] Udo Frese and Gerd Hirzinger. Simultaneous localization and mapping-a discussion. In *Proceedings of the IJCAI Workshop on Reasoning with Uncertainty in Robotics*, pages 17–26. Seattle, 2001.

[22] Charles Galamhos, Jose Matas, and Josef Kittler. Progressive probabilistic hough transform for line detection. In *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, volume 1, pages 554–560. IEEE, 1999.

[23] Giorgio Grisettiyz, Cyrill Stachniss, and Wolfram Burgard. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. In *Proceedings of the 2005 IEEE international conference on robotics and automation*, pages 2432–2437. IEEE, 2005.

[24] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[25] Yinghua He, Hong Wang, and Bo Zhang. Color-based road detection in urban traffic scenes. *IEEE Transactions on intelligent transportation systems*, 5(4):309–318, 2004.

[26] Wolfgang Hess, Damon Kohler, Holger Rapp, and Daniel Andor. Real-time loop closure in 2d lidar slam. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1271–1278. IEEE, 2016.

[27] Joong jae Lee and Gyeyoung Kim. Robust estimation of camera homography using fuzzy ransac. In *International Conference on Computational Science and Its Applications*, pages 992–1002. Springer, 2007.

[28] C. R. Jung and C. R. Kelber. A robust linear-parabolic model for lane following. In *Proceedings. 17th Brazilian Symposium on Computer Graphics and Image Processing*, pages 72–79, 2004.

[29] Niklas Karlsson, Enrico Bernardo, Jim Ostrowski, Luis Goncalves, Paolo Pirjanian, and Mario Munich. The vslam algorithm for robust localization and mapping. pages 24–29, 01 2005.

[30] Stefan Kohlbrecher, Oskar Von Stryk, Johannes Meyer, and Uwe Klingauf. A flexible and scalable slam system with full 3d motion estimation. In *2011 IEEE international symposium on safety, security, and rescue robotics*, pages 155–160. IEEE, 2011.

[31] Patrick Lin. Why ethics matters for autonomous cars. In *Autonomous driving - Technical, Legal and Social Aspects*, pages 69–85. Springer, Berlin, Heidelberg, 2016.

[32] X. Lin and S. Chen. Color image segmentation using modified hsi system for road following. In *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, pages 1998,1999,2000,2001,2002,2003, Los Alamitos, CA, USA, apr 1991. IEEE Computer Society.

[33] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2, 1999.

[34] Lin-Bo Luo, In-Sung Koh, Kyeong yuk Min, Jun Wang, and Jongwha Chong. Low-cost implementation of bird's-eye view system for camera-on-vehicle. pages 311 – 312, 02 2010.

[35] Marina Magnabosco and Toby P Breckon. Cross-spectral visual simultaneous localization and mapping (slam) with sensor handover. *Robotics and Autonomous Systems*, 61(2):195–208, 2013.

[36] David Marr and Ellen Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 207(1167):187–217, 1980.

[37] Blagoj Nenovski and Igor Nedelkovski. Defining a feature-rich end-to-end augmented reality platform for spatial exploration. pages 103–108, 10 2018.

[38] NVIDIA support forum. Jetson TX2 FPS Issue. `https://forums.developer.nvidia.com/t/ros-realsense-d435-on-jetson-tx2-only-15fps-fhd-availlable/62420`, last visited on 12.7.2020".

[39] Edwin B Olson. Real-time correlative scan matching. In *2009 IEEE International Conference on Robotics and Automation*, pages 4387–4393. IEEE, 2009.

[40] Tomas Pajdla. Elements of geometry for computer vision. *FEE CTU,[online].[cit. 2013-05-19], March*, 2013.

[41] Qifa Ke and T. Kanade. Transforming camera geometry to a virtual downward-looking camera: robust ego-motion estimation and ground-layer detection. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 1, pages I–I, 2003.

[42] José I Ronda and Antonio Valdés. Geometrical analysis of polynomial lens distortion models. *Journal of Mathematical Imaging and Vision*, 61(3):252–268, 2019.

[43] Robert Sim, Pantelis Elinas, Matt Griffin, James J Little, et al. Vision-based slam using the rao-blackwellised particle filter. In *IJCAI Workshop on Reasoning with Uncertainty in Robotics*, volume 14, pages 9–16, 2005.

[44] Simon A. Eugster. YCbCr. `https://en.wikipedia.org/wiki/YCbCr` last visited on 3.5.2020.

[45] Nicolas Simond and Patrick Rives. Homography from a vanishing point in urban scenes. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, volume 1, pages 1005–1010. IEEE, 2003.

[46] Jongin Son, Hunjae Yoo, Sanghoon Kim, and Kwanghoon Sohn. Real-time illumination invariant lane detection for lane departure warning system. *Expert Systems with Applications*, 42(4):1816–1824, 2015.

[47] Tsung-Ying Sun, Shang-Jeng Tsai, and Vincent Chan. Hsi color model based lane-marking detection. In *2006 IEEE Intelligent Transportation Systems Conference*, pages 1168–1172. IEEE, 2006.

[48] W. Hess, D. Kohler, H. Rapp, and D. Andor. Google Cartographer Documentation. `https://google-cartographer.readthedocs.io/en/latest/`, last visited on 12.7.2020".

[49] Aiqi Wang, Tianshuang Qiu, and Longtan Shao. A simple method of radial distortion correction with centre of distortion estimation. *Journal of Mathematical Imaging and Vision*, 35(3):165–172, 2009.

[50] Jian Wang, Zhong Ji, and Yu-Ting Su. Unstructured road detection using hybrid features. In *2009 International Conference on Machine Learning and Cybernetics*, volume 1, pages 482–486. IEEE, 2009.

[51] Yue Wang, Eam Khwang Teoh, and Dinggang Shen. Lane detection and tracking using b-snake. *Image and Vision computing*, 22(4):269–280, 2004.

[52] Rauf Yagfarov, Mikhail Ivanou, and Ilya Afanasyev. Map comparison of lidar-based 2d slam algorithms using precise ground truth. In *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 1979–1983. IEEE, 2018.

[53] Yannick Morvan. Pinhole Camera Model. `http://epixea.com/research/multi-view-coding-thesisch2.html` last visited on 3.5.2020.

[54] B. Yu and A. K. Jain. Lane boundary detection using a multiresolution hough transform. In *Proceedings of International Conference on Image Processing*, volume 2, pages 748–751 vol.2, 1997.

[55] Radim Šára. 3d computer vision: II. perspective camera. `http://cmp.felk.cvut.cz/cmp/courses/TDV/2019W/lectures/tdv-2019-02.pdf` last visited on 3.5. 2020, 2019.

[56] Radim Šára. 3d computer vision: III. computing with a single camera. `http://cmp.felk.cvut.cz/cmp/courses/TDV/2019W/lectures/tdv-2019-04.pdf` last visited on 3.5. 2020, 2019.

CD Contents

The names of all the root directories on the CD are listed in Table 1.

Directory name	**Description**
thesis	the thesis in PDF format
thesis_sources	LaTex source codes
source	C++ and MATLAB source codes, parameter configurations and launch files
video	demonstrative video of road detection, localisation and mapping

Table 1: CD Contents

List and Meaning of Abbreviations

The abbreviations used in this thesis are listed in Table 2.

Abbreviation	**Meaning**
EKF	Extended Kalman Filters
FPS	Frames per second
IMU	Inertial Measuremenent Unit
LIDAR	Light Detection and Ranging
PC2	2-dimensional Point Cloud
RANSAC	Random Sample Consensus
ROS	Robot Operating System
SIFT	Scale Invariant Feature Transform
SVD	Singular Value Decomposition
SLAM	Simultanious Localization and Mapping

Table 2: Lists of abbreviations