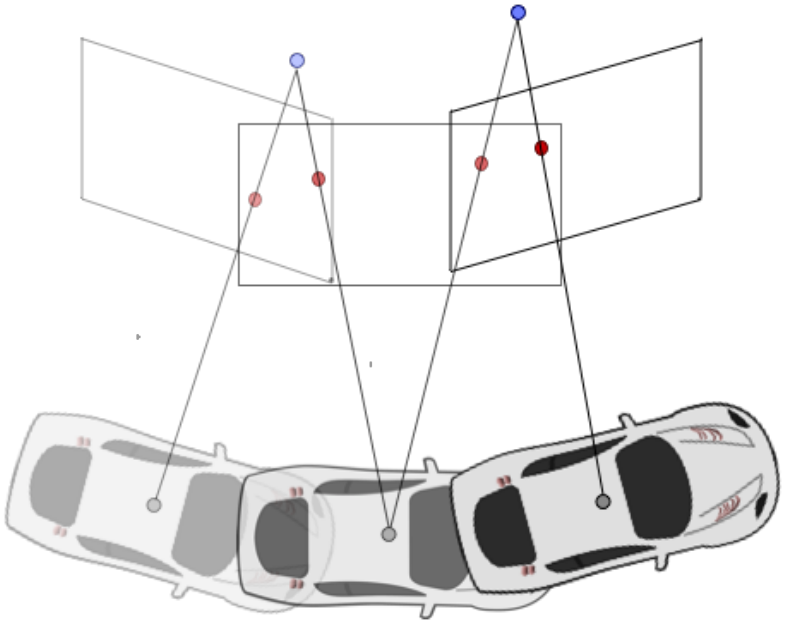
CHALMERS
UNIVERSITY OF TECHNOLOGY



Targetless calibration for vehicle mounted cameras

in planar motion using visual odometry

Master's thesis in Engineering mathematics and computational science &
Systems, control and mechatronics (Ska det vara Electrical Engineering här?)

ANDREAS ELLSTRÖM and KARIN HESSLOW

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2018

MASTER'S THESIS 2018: EENX30

Targetless calibration for vehicle mounted cameras

in planar motion using visual odometry

ANDREAS ELLSTRÖM and KARIN HESSLOW

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Signal processing and Biomedical engineering
Computer vision and medical image analysis
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2018

Targetless calibration for vehicle mounted cameras

ANDREAS ELLSTRÖM and KARIN HESSLOW

Supervisor: Christoffer Gillberg, Zenuity
Examiner and supervisor: Fredrik Kahl, Department of Electrical Engineering

Master's Thesis 2018:EENX30
Department of Electrical Engineering
Signal processing and Biomedical engineering
Computer vision and medical image analysis
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Moving vehicle with overlapping camera views for corresponding frames and estimated reconstructed 3D-points.

Gothenburg, Sweden 2018

Targetless calibration for vehicle mounted cameras
ANDREAS ELLSTRÖM and KARIN HESSLOW
Department of Electrical Engineering
Chalmers University of Technology

Abstract

The implementation of advanced driving systems has become a common feature in modern vehicles and a reassurance for the driver to help respond and intervene in critical situations. As assisting systems in vehicles continue to evolve and the need to handle more complex situations, the requirement of more data from the surrounding increases, which relies on the amount of sensors mounted on the vehicle. This however will give rise to the need of new ways to automatically calibrate cameras as of today they are being calibrated manually in a staged environment.
This thesis present an unsupervised method for extrinsic calibration of cameras mounted on a planar moving vehicle using visual odometry. The method uses hand-eye calibration by the relative motion between the camera and the vehicle. The method has been tested and verified on test data from KITTI Vision Benchmark. Real data provided by and in cooperation with Zenuity has been implanted, but needs some improvement to provide a good result.
As planar motion does not provide sufficient information for a full calibration of all 6-DoF, which represents the extrinsic calibration, other methods of estimating the remaining three parameters are briefly included in future work as a continuation of a full unsupervised calibration.
The result shows that the algorithmic approach can determine the general region of which the camera is located, but is unable to accurately find its absolute position. It is depending on a more accurate estimation of the scale parameters and probably also the estimation from the visual odometry.

Keywords: targetless, extrinsic calibration, hand-eye calibration, planar motion, visual odometry

Acknowledgements

During this thesis work many people have been helping out a lot. In general we would like to thank many of the employees at Zenuity.
Explicitly, we want to thank Christoffer Gillberg for daily support and supervising. We want to thank the 3D-reconstruction team at Zenuity for inputs of ideas, especially Daniel Philquist for helping us starting the project and keeping us in the right direction. Viktor Runemalm has been a great person to discuss our ideas along the way. We also want to thank Peter Starke and Fredrik Kahl for support and supervising.

Andreas Ellström and Karin Hesslow, Gothenburg, June 2018

Acronyms and abbreviations

GNSS - Global navigation satellite system
IMU - Inertial measurement unit
ADAS - Advanced driver assistance systems
BA - Bundle adjustment
GBA - Global bundle adjustment
LBA - Local bundle adjustment
SLAM - Simultaneous location and mapping
RANSAC - Random sample consensus
SfM - Structure from motion
LMA - Levenberg–Marquardt algorithm
DLT - Direct linear transformation algorithm
DoF - Degrees of freedom
PnP - Perspective-n-Point
VO - Visual Odometry
EBA - Emergency break assist

Contents

1 Introduction **1**
- 1.1 Advanced driver-assistance systems (ADAS) . . . 1
- 1.2 Image sensors . . . 2
- 1.3 Calibration . . . 2
- 1.4 Thesis aim . . . 3
- 1.5 Limitations . . . 3
- 1.6 Verification and ground truth . . . 3
- 1.7 Related work . . . 4

2 Theory **5**
- 2.1 Camera model . . . 5
  - 2.1.1 Camera projection . . . 5
  - 2.1.2 Lens distortion . . . 7
- 2.2 Computer vision . . . 8
  - 2.2.1 Visual odometry . . . 8
  - 2.2.2 SLAM . . . 9
  - 2.2.3 Feature extraction . . . 10
  - 2.2.4 Feature matching . . . 11
  - 2.2.5 Pose estimation . . . 12
  - 2.2.6 Kalman filter . . . 14
  - 2.2.7 Bundle adjustment . . . 15
- 2.3 Camera calibration . . . 15
  - 2.3.1 Intrinsic calibration . . . 15
  - 2.3.2 Extrinsic calibration . . . 16
- 2.4 Calibration approaches . . . 16
  - 2.4.1 Direct linear transformation (DLT) . . . 17
  - 2.4.2 Zhang's method . . . 17
- 2.5 OxTS . . . 17
  - 2.5.1 Global Navigating Satellite System . . . 17
  - 2.5.2 Inertial Measurement Unit . . . 18
- 2.6 Coordinate frames of reference . . . 18
  - 2.6.1 Global frame . . . 18
  - 2.6.2 Vehicle frame . . . 18
  - 2.6.3 Camera frame . . . 18
  - 2.6.4 Homogeneous coordinates . . . 19
  - 2.6.5 Transformation matrices . . . 19

2.6.6 Convert 3D to 2D rotation matrix . . . 19
2.7 Fit plane to trajectory . . . 20
2.8 Projection of a 3D-plane to xy-plane . . . 21
2.9 3D point cloud . . . 22

3 Methods **23**
3.1 Test data . . . 24
3.1.1 Synthetic data . . . 24
3.1.2 Real data . . . 24
3.1.2.1 KITTI Vision Benchmark . . . 24
3.1.2.2 Zenuity data . . . 26
3.2 Camera trajectory estimation from odometry . . . 26
3.2.1 Scale estimation . . . 26
3.2.1.1 Projection of a trajectory . . . 27
3.3 Calibration approach . . . 28
3.3.1 Calibration model . . . 29

4 Results **31**
4.1 Verification of calibration method . . . 31
4.2 KITTI data . . . 32
4.2.1 Feature detection and matching . . . 33
4.2.2 Scale estimation . . . 34
4.2.3 Plane estimation and trajectory projection . . . 35
4.2.4 Calibration results . . . 37
4.3 Collected data by Zenuity . . . 38

5 Discussion **43**
5.1 Calibration model . . . 43
5.2 Zenuity data . . . 43
5.3 Calibration result . . . 44
5.4 Future work . . . 44
5.4.1 Estimation of roll, pitch and z . . . 44
5.5 Conclusion . . . 44

Bibliography **47**

1

Introduction

Autonomous vehicles have long been a sought after and futuristic concept to relive drivers from long and tiresome travels while at the same time improve traffic flow and prevention of critical traffic situation. As of today, the concept of a fully implemented autonomous vehicle is not yet available to the public but research and testing has come a long way trying to make this a soon reality. The most crucial element of autonomous driving is to be able to observe and respond to what is happening around the vehicle. Much like the human senses, the vehicle requires sensors to gather data as a mean to perceive its surrounding. However, these sensors must be gather data accurately at all time and also be reliable at high velocities and unstable conditions to ensure safe performance.
Zenuity is a young company developing software for autonomous cars. To develop a vehicle software structure that is capable of perceiving its surrounding environment, navigate without human input and reacting to traffic situations will be the greatest challenges for autonomous driving within the coming years. To meet these problems data from many different sensors are required, such as radar, lidar, GNSS (Global Navigation Satellite System), IMU (Inertial Measurement Unit), cameras etc.

1.1 Advanced driver-assistance systems (ADAS)

ADAS are systems developed in purpose to help the driver in the driving process and to increase the safety. ADAS uses input from many sensors to be able to make decisions by estimating the environment. A visualization of a typical environment and a vehicles sensor reach can be seen in figure 1.1. By minimizing the human error by assistance systems, the fatal accidents reduces, since the majority of car accidents is due to the human factor [1]. In modern vehicles there are several safety features implemented, both technologies that can warn the driver for potential problems and also by implementing safeguards. Among these are systems such as emergency breaking assist (EBA) which uses radar and will automatically break if an inevitable collision is detected.

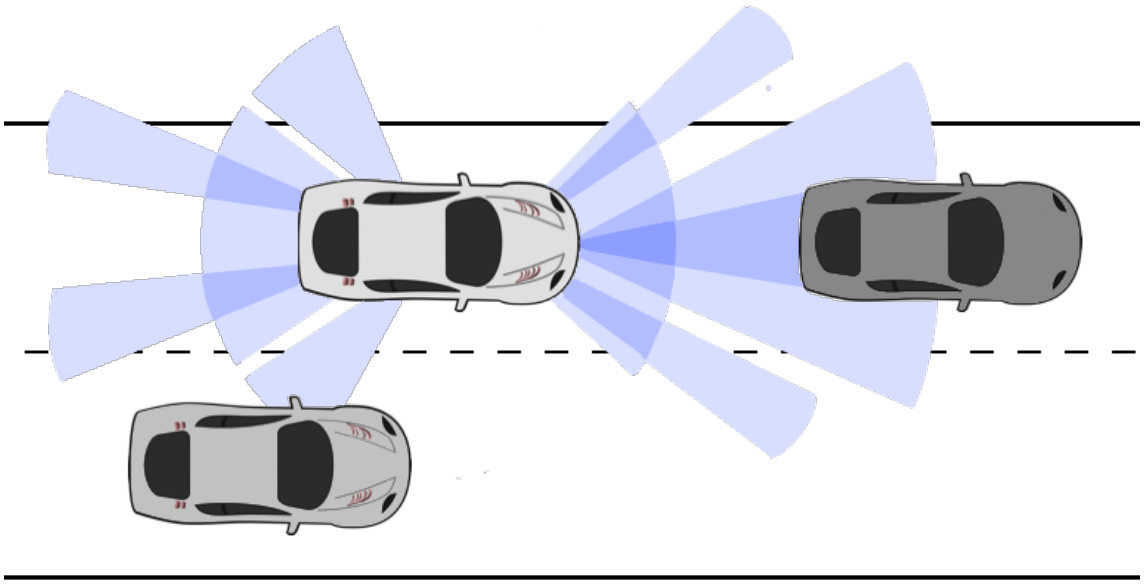


Figure 1.1: ADAS, illustration of how a sensor set up can look like. With both overlapping long and short range sensors the system is capable to reliably perceive its surroundings.

1.2 Image sensors

Image sensors, such as cameras, are offering a rich source of visual information of its perceptive view and is a good complement to other sensors, such as lidar and radar. This has made cameras a crucial part of many robotic systems where texture and features are involved and a great tool for recognition and tracking. However, as image sensors only captures momentary visual scenes, no metrical information, the absolute distance to objects is not acquired, which other sensors such as lidar are able to estimate. There exists several alternative ways to compensate for this flaw, both practical and with additional external information. Among these alternatives are rgb-d cameras, which also measures depth of the scene, or stereo vision, where two cameras are used with a fixed and known distance between them. In a similar fashion as with stereo vision, a single camera or monocular vision, can also be used to measure distance to an object by triangulation, if the distance between two frames are known. To extract the information from images computer vision has to be implemented, see section 2.2.

1.3 Calibration

In order to achieve self-driving vehicles, it is essential to have accurate sensor calibration. For example, imagine we have a sensor incorrectly calibrated by two degrees for the angle in the planar motion. Then an object that is located 60 meters in front of the vehicle is estimated to be located two meters away from its actual location. That could cause a lot of wrong made decisions for an autonomous vehicle.

Since this thesis topic is camera calibration, the continuing will resolves around that. Camera calibration can be divided into two subgroups; intrinsic and extrinsic calibration. While intrinsic calibration resolves around finding the internal parameters of the camera such as image distortion and other lens aberrations, extrinsic calibration aims to find the mounted position of the camera, in the coordinate system of the vehicle.
When it comes to vehicle mounted cameras within the application of ADAS, cameras are most often used as a tool for road estimation and identify other vehicles and pedestrian in traffic. Furthermore, without knowledge of where the camera is mounted it becomes impossible to know where exactly the identified object is in relation to the vehicle. Most camera calibration methods today uses a priori knowledge of the surrounding and are done by an operator. Most commonly this is a fixed object within a known environment. However, as the systems are approaching fully autonomous driving, the number of cameras mounted on the vehicles is substantially increasing. This makes traditional camera calibration time consuming and perhaps infeasible in the future.

By standard assumptions a vehicle can be expected to be bound to planar motion. This however limits the location of the vehicle in height above the ground as well as rotation around its x and y axis will not occur. Hence, the only parameters able to estimate under this conditions is x, y and yaw. Other methods for calibration of the left out parameters exist, but require different approaches which will be left for *future work*, section 5.4.

1.4 Thesis aim

The aim of this thesis is to develop a method for targetless extrinsic calibration of cameras mounted sparsely around a vehicle, which not only assumes no a priori knowledge about the surrounding but also involves as little human supervision as possible. The goal is to find an accurate calibration between the vehicle and the cameras from their relative motion.

1.5 Limitations

For this calibration, overlapping fields of camera views were not used in the calibration algorithm, i.e each camera was treated as it was the only camera mounted on the vehicle. In addition, no other sensors, than GNSS and IMU, were used in the method to improve the accuracy of the result, since the calibration should be independent of performance of other sensors.
No limitation on computation time were set but the calibration should ideally be done within a feasible time frame while also requiring as little supervision as possible. The calibration will also be limited to planar motion with estimation of its 3 DoF.

1.6 Verification and ground truth

In order to assure the reliability and accuracy of ADAS and calibrations the systems need to be thoroughly tested. There exists numerous ways to set up tests and scenarios to verify

results and in turn evaluate if any problems or errors exist in the systems, as these may be critical for the system to work correctly. In particular when it comes to calibration, an inaccurate calibration of a sensor can have devastating consequences for the entire perceptive system of the vehicle. As a redundancy in autonomous vehicles, sensor fusion is often used as a tool to ensure that what one sensor detects also can be found across more sensors to reliably trust the detection and afterward the necessary action. In the case of an inaccurate calibrated sensor, this secondary detection may not happen, or discrepancy may occur, which will lead to incorrect responses due to inconstancy of the detection.
A good way to verify a result is to compare the result to a ground truth as a reference to what is expected. A ground truth can be acquired in several different ways, depending on the accuracy requirements and what data is available. To use fabricated data is one solution to produce a ground truth, which is a created ideal environment with all known parameters. This is normally referred to as synthetic data and can be used as test data in almost every part of the system to verify that the systems respond in a correct manner. However, synthetic data is often not enough to ensure that the system works in practice, since real data is a subject to noise and disturbance. It can be difficult to create ground truth for real data and it requires several different approaches. One option is to use other, more accurate sensors to first estimate for example a trajectory in their own frame and then transform it to the frame of the sensor of interest.

1.7 Related work

While the topic of calibrating cameras with respect to a freely moving robot has been well explored, [2], [3], the concept of vehicle mounted cameras presents a different problem as it is restricted to planar motion. A few different approaches have been presented over the recent years to solve this problem but to our knowledge, all has pre-given information about the set up or the surrounding.
In [4] the calibration of intrinsic and extrinsic parameters for the camera are estimated at the same time together with odometry from a wheel speed encoder for a differential drive robot, but requires fixed known landmarks as world references. Other similar hand-eye calibration works have been done for camera to camera calibrations such as in [5] but only up to scale as no reference to the system is given in that case. Förläng detta stycke

2
Theory

This chapter gives a brief background of the theory behind this thesis work. It introduces the concepts of how cameras work and how a trajectory can be estimated from a sequence of images using different strategies. Furthermore, concepts which are important for the calibration are covered and explained.

2.1 Camera model

Camera models are commonly based on the pinhole camera type. It is a simple camera without a lens but with a pinhole, i.e a small hole in one side of a light-proof box, where rays of light passes through a small pinhole and creates an upside down reflection of the front view, see figure 2.1.

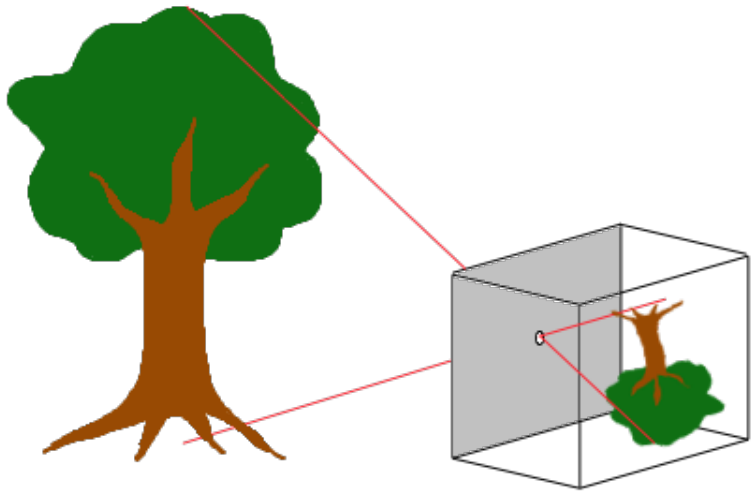


Figure 2.1: Visualization of how a pinhole camera works with an upside down reflection.

2.1.1 Camera projection

In computer vision a camera projection describes the mapping of a pinhole camera from 3D points in the world to 2D points in an image. This can mathematically be described

as,

$$\lambda u = PU, \tag{2.1}$$

which is commonly referred to as the camera equation. In this P is a 3x4 matrix, the so called camera matrix which maps the 3D points to the 2D image plane, see figure 2.2. This equation is composed by a translation vector and orientation matrix which is warped by the intrinsic parameters, see section 2.3.1. Furthermore, U is a 3D point given in homogeneous coordinates, see section 2.6.4 while u is the corresponding 2D point, also homogeneous, found in the image plane. Finally λ is referred to as the scale factor for the image point.

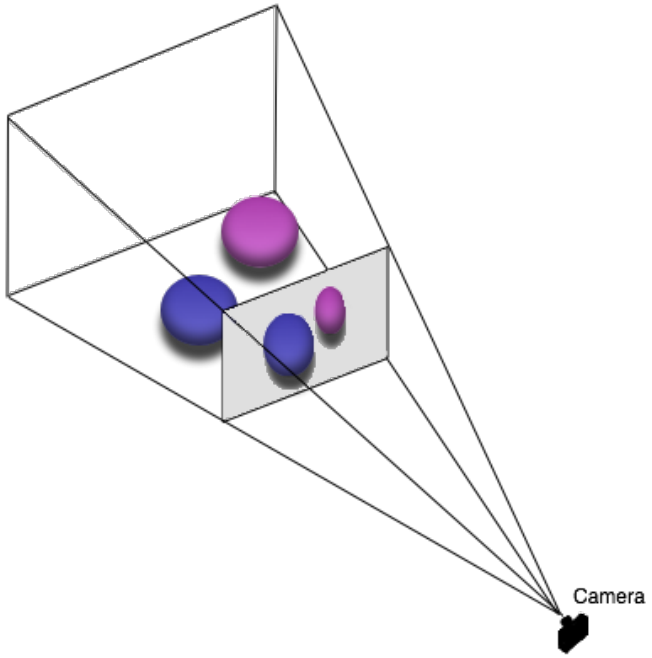


Figure 2.2: Camera projection. A 3D scene are projected onto a 2D plane.

If a pose or camera matrix, P, has be estimated correctly it should match with the true camera projection. To see how well it fits, the proposed camera matrix's 3D points can be re-projected back into the image and be compared with the original measurement in 2D. To start with, a triangulation of corresponding 3D points is estimated between the previous pose, P_1 and the proposed camera matrix P_2 as,

$$\begin{aligned} \lambda_1 u_1 &= P_1 U \quad \text{with } \lambda_1 > 0 \\ \lambda_2 u_2 &= P_2 U \quad \text{with } \lambda_2 > 0. \end{aligned} \tag{2.2}$$

This however presents 6 equation and 5 unknowns, λ_1, λ_2 and the 3D point $\mathbf{U}$, which easiest is solved by disregarding one of the equations. This now yields a linear quadratic system which the unknowns can be solved for. The estimation of 3D-points $\hat{\mathbf{U}}$ can now be re-projected back into all the images they have been found in and see how well it matches

with the original measured image points. Let,

$$P = \begin{pmatrix} \leftarrow a^T \rightarrow \\ \leftarrow b^T \rightarrow \\ \leftarrow c^T \rightarrow \end{pmatrix} \tag{2.3}$$

where a^T, b^T and c^T are the row vectors of the hypothetical camera matrix. For each estimation of *hat***U** the corresponding point in each image can be denoted as $\hat{u}$ and be described in the camera equation as,

$$\lambda \hat{u} = \lambda \begin{pmatrix} \hat{x} \\ \hat{y} \\ 1 \end{pmatrix} = P\hat{U} = \begin{pmatrix} a^T\hat{U} \\ b^T\hat{U} \\ c^T\hat{U} \end{pmatrix} \text{ with } \lambda > 0. \tag{2.4}$$

Rewriting this from matrix form to equations and solving for $\hat{x}$ and $\hat{y}$ yields,

$$\lambda = c^T\hat{U}, \ \hat{x} = \frac{a^T\hat{U}}{c^T\hat{U}}, \ \hat{y} = \frac{b^T\hat{U}}{c^T\hat{U}}, \ \text{ if } c^T\hat{U} > 0. \tag{2.5}$$

Now the hypnosis of P can be compared as the residual vector, r, see equation below, can be computed.

$$r = \begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} - \begin{pmatrix} x \\ y \end{pmatrix}, \ \text{ if } c^T\hat{U} > 0 \tag{2.6}$$

The absolute residual, $|r|$, is often referred to as the re-projection error as it measures how far the re-projected 2D point is from the original measurement.

2.1.2 Lens distortion

Even if camera models commonly are based on a pinhole camera, most images are not captured with such camera, which results in a lens distortion. Lens distortion is when a lens produces a straight line as a curved line. The most common distortions are barrel and pincushion distortion, shown in figure 2.3

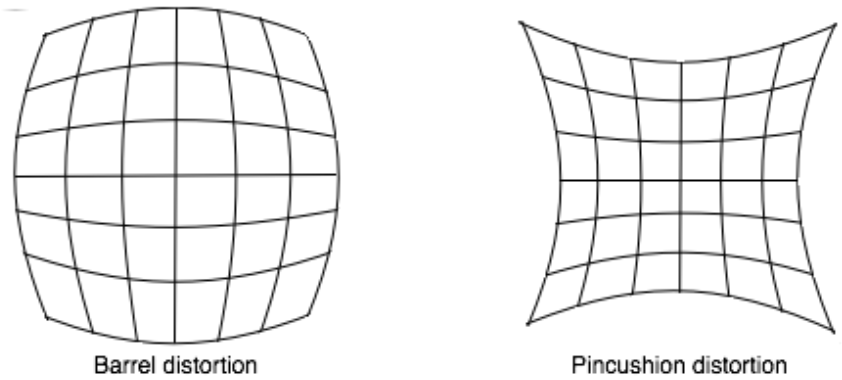


Figure 2.3: The most common lens distortion. To the left: Barrel Distortion. To the right: Pincushion Distortion.

2.2 Computer vision

In the scope of robotic engineering, computer vision is the subfield of artificial intelligence which tries to mimic the human visual system as a tool for machines to "understand" 3-dimensional scenes. Typical problems in computer vision involves recognition and identification of specific objects, features or activity perceived from images. In this regard, pose and motion estimation also becomes a topic which falls under this subject and represents the fundamental principle for camera trajectory estimation in this thesis. In general, most of the existing visual based motion estimation approaches can be summarized into four steps:

1) **Image sequence** can be produced by different types of cameras and setups, and can be used to satisfy the requirements of the situation. This can range from a monocular frontal looking camera to 360° view around the vehicle, consisting of several cameras with overlapping views depending on what is needed by the system.
2) **Feature extraction** is needed to find correspondences between images. There exists a verity of different types of correspondences but which one to select comes down to the trade of between whether accuracy, robustness or computation time is prioritized.
3) **Feature matching** these correspondence correctly and tracking them over sequential frames is the core part of understanding the motion of camera.
4) **Pose estimation and optimization** estimates the relative pose, i.e. translation and orientation, between two images. As more images, and thereby more information, are added these poses can be more accurately estimated. Also, some type of optimization is preformed as a final step. There are two main approaches to this optimization, filter based methods, such as Kalman and particle filters and Bundle Adjustment, further described in section 2.2.6 and 2.2.7.

2.2.1 Visual odometry

In computer vision, visual odometry (VO) is the process of estimating the egomotion of the camera and aims to recover the trajectory incrementally, pose after pose followed by preforming local optimization. The egomotion of a camera describes its 3D motion relative to the environment and has been used in the past within a verity of projects [6], [7]. VO is based on the technique known as Structure From Motion (SFM) which focuses on 3D reconstruction of both surrounding environmental structure and camera poses from 2D images. The first real-time long-ruining implementation of VO were done by Nistér et al. [8], which became one of the largest and name-founding contributions to the field. The "odometry" part was coined due to being very similar to the concept of wheel speed odometry [9]. It can be implemented for most types of camera setups but suffers from scale estimation and drift in the case of monocular vision unless additional information, such as known objects or complementing data from other sensors are used. The motion part in SFM can be considered as views from different cameras or by moving a single camera in order to observe a scene from different angles, as can be seen in figure 2.4.

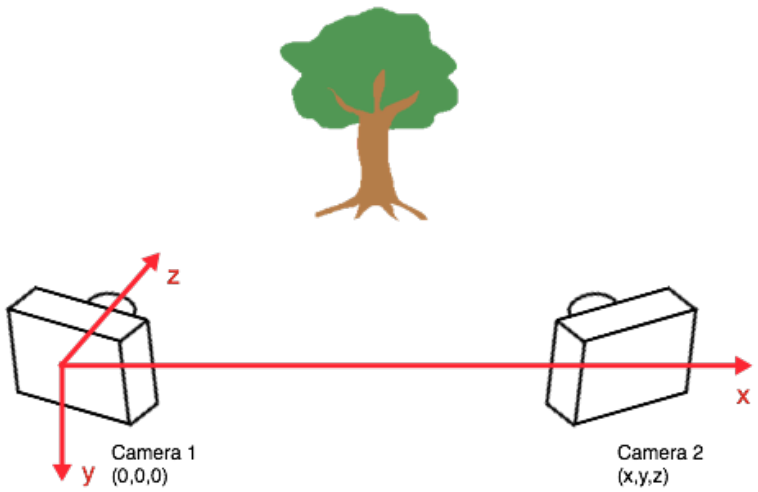


Figure 2.4: Visualization of structure from motion. Camera 1 and camera 2 represents the first and the second view, respectively.

2.2.2 SLAM

Simultaneous Location And Mapping (SLAM) or Visual SLAM (vSLAM) for cameras, is a process where a robot is required to locate itself in an unknown environment and at the same time construct the map of the surrounding area. This problem is a so called chicken-and-egg problem and has been extensively studied in the past decades due to being able to be implemented for many different types of sensors [10], [11]. In large, the main difference between VO and vSLAM is where each of the methods focus lies. In the case of VO, the focus lies on local consistency from pose to pose whereas vSLAM tries to maintain a global consistency of the map and camera trajectory it creates. One way for vSLAM to do this is to detect previously visited areas. This is called loop closure, which is reducing the drift in its estimation, see figure 2.5.

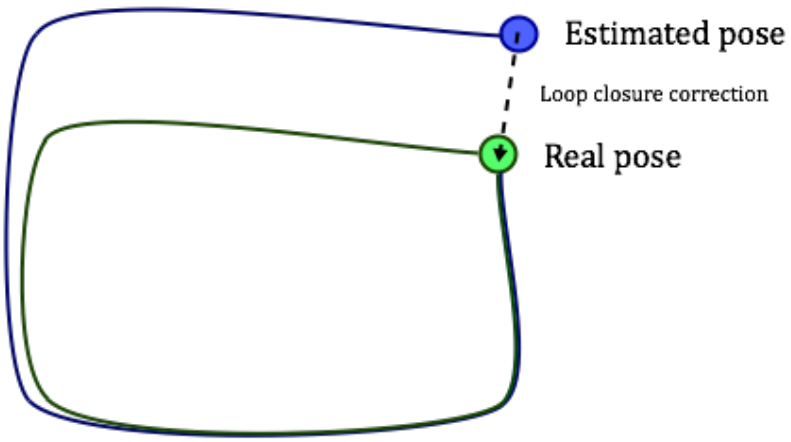


Figure 2.5: Loop closure. The blue estimated line is corrected and the final estimated pose is moved to the green spot instead.

2.2.3 Feature extraction

A feature, or key point, is an "interesting" part of an image, see figure 2.6, which ideally can be uniquely found in several images which are taken from different prospective of the same feature. Features are generally detected at large differentials, such as edges and corners of objects. Depending on the circumstances features can be difficult to detect, for example if there is lack of light. During the last decades many methods of detecting features has been developed and today a diversity of detectors and descriptors for different purposes are available. As calibration favours precision the choice came down to using SIFT (Scale-Invariant Feature Transform) or SURF (Speeded-Up Robust Features), since these have in the past proven high accuracy due to being invariant to scale change, rotation and illumination [12], [13]. In this work, SURF were chosen as its implementation and support in MATLAB makes it both easy and practical to use, while also running significantly faster then SIFT which decreases calibration time [13].

Both the SIFT and SURF algorithms are based on the same principles but the details of how each step is preformed differ. The method used in this thesis is SURF, hence, it will be explained more detailed. As method to detect points of interests, SURF uses a basic approximation of the Hessian matrix by using the integral image. In order to then describe the feature, a distribution of Haar-wavelet is described that responses within the interest point neighbourhood. For SURF-points to be invariant to rotation, the orientation of the point of interest needs to be estimated which is done within the neighbourhood of the feature.

Figure 2.6: Feature extraction. Detected features in the image are marked with green marks.

2.2.4 Feature matching

In order to find correspondences between images to be able to determine the camera's trajectory, matching points between sequential images need to be found, see figure 2.7. This is done by trying to compare and find matching feature descriptors between frames. However, as features are described locally, mismatches are a common problem due to repeating patterns or similar looking descriptors in the image.

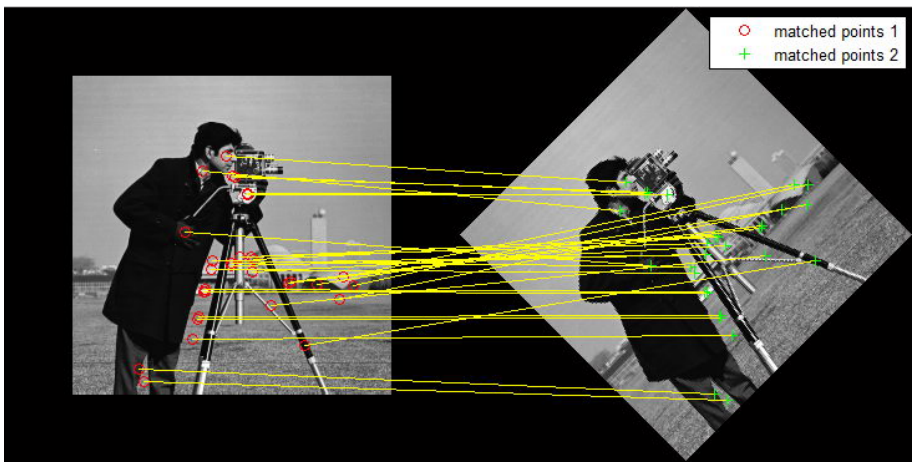


Figure 2.7: Feature matching. Detected features in the left image are matched with the detected features in the right image.

Methods for trying to detection and remove of outliers, i.e. miss-matches, are thus important to use, as miss-matched features are undesirable. One of the most commonly used algorithms for this purpose is Random sample consensus (RANSAC) which has proven to work well in the past [14]. Methods for pose estimation, see section 2.2.5, only need a very small subset of correctly matched feature points. Hence, the chance of randomly selecting a subset without outliers depends on how well the matching has been done, in

other words the ratio between in- and outliers. The strategy of RANSAC is to iteratively selecting a random subset, estimate the corresponding pose and see how well this theoretical pose supports the re-projection of the other feature points, within a certain threshold, see figure 2.8 for visualization. Those points which falls within the threshold of the re-projection are seen as inliers and those who does not are considered outliers. RANSAC then saves the estimated pose with the highest amount of inliers and uses it for the final optimization step. This has proven to work well in practice if a sufficient amount of iterations is ran with respect to the outlier ratio.

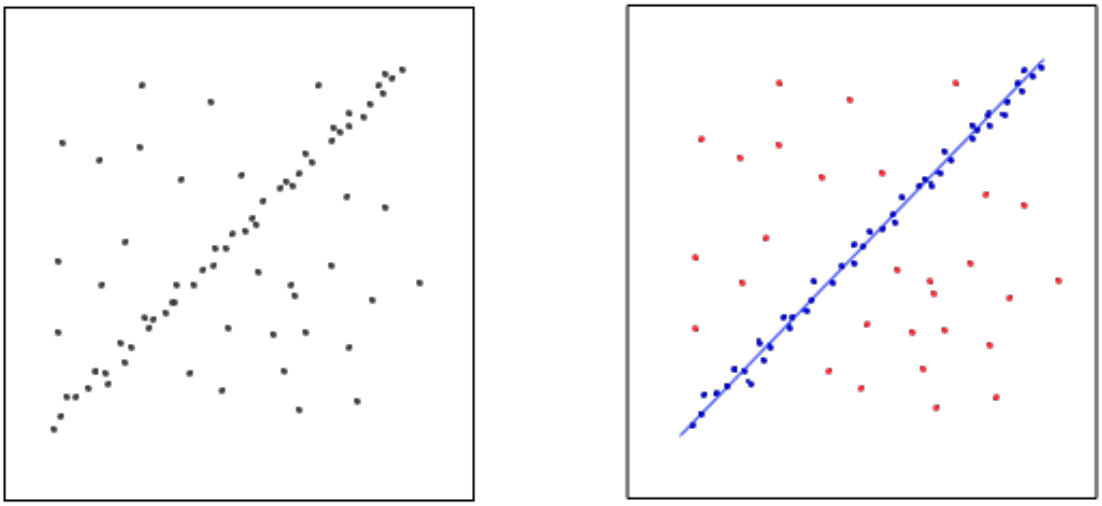


Figure 2.8: Outlier detetion. The blue dots in the right image are detected inliers and the red dots are outliers.

2.2.5 Pose estimation

When it comes to estimation of the relative pose of the camera, a few main approaches are used depending on the dimension of the correspondences are available.

2D-2D

When no 3D points has been estimated, 2D to 2D correspondences between images for estimation of the relative pose is the preferred approach. Depending on situation and geometry limitations different algorithms can be used which mostly differ in the amount of correspondences needed for pose estimation. One of the most used algorithms is Nister's *Five Point Algorithm*[15] which, as the name applies, uses five corresponding points. By using epipolar geometry which assumes geometric relations between 3D points and there projections, constraints can be formed and the relative pose between views can be found [16]. This is normally done by estimating the essential matrix, **E**, that relates the image of a point, p_0, in one frame to its corresponding point, p_1, in the other frame, which upholds the relation $p_0^T E p_1 = 0$. The essential matrix is defines as:

$$\mathbf{E} = [t]_x R. \tag{2.7}$$

where $[t]_x$ is the translation described as a 3-by-3 skew-symmetric matrix and R the rotation which describes the pose between the two views. However, the scale of the translation in this equation is not known and somehow needs to be estimated.

3D-2D

When corresponding points between images has been determined, triangulation of the matched 2D points can be done, which result in 3D points in the world coordinate frame. By matching the next image within these 3D points, the new image can be projected from 3D to 2D and its relative pose to the world can be found. This is done by minimizing the 2D re-projection error and can be described by the cost function:

$$\mathbf{T} = arg\min_{\mathbf{T}} \sum_{i} |\mathbf{z} - f(\mathbf{T}, \hat{\mathbf{X}}_i)|^2, \tag{2.8}$$

where $\mathbf{T}$ represents the transforms between the current frame and the previous, with $\mathbf{z}$ being the observed 2D points in the current frame and $f(\mathbf{T}, \hat{\mathbf{X}}_i)$ being the re-projection function of the corresponding 3D points triangulated from the previous frame. This type of pose estimation is called Perspective-n-Point (PnP) [17] where i represents the number of feature pairs, depending on which PnP method is used.

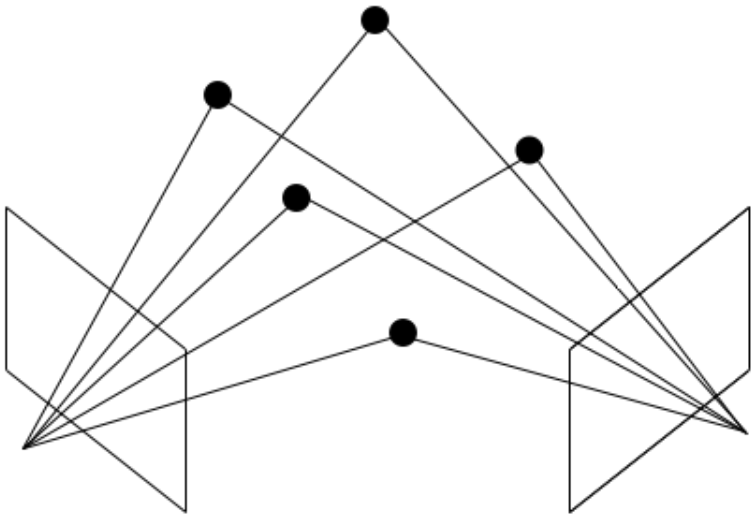


Figure 2.9: *Five Point Algorithm* determine the relative camera pose between two views given five corresponding points.

To increase the performance of the pose estimation the matched features should be selected uniformly over each image. In general, this increases the robustness of the pose

estimation as features are selected at different distances from the camera and estimates a more accurate ego-motion [18]. However, as features are detected in images at large differentials, such as edges, it is not likely for features to be uniformly spread, but instead often clustered around outstanding objects. In order to acquire a good distribution of features the image is split into small grids and non-maximum suppression is used to find the feature with local maximum.

2.2.6 Kalman filter

Kalman filter is a recursive filter, or algorithm, estimating the state in a dynamic system by data normally containing noise, such as motion blur for cameras. The algorithm is very useful in many different situations. The filter is composed by two steps, prediction and update. The prediction step takes the prior states and the uncertainty of those states, a covariance matrix, and runs it through a motion model to predict the next state. Meanwhile a new measurement of the next state is done and becomes processed by a measurement model to fit the state model. The update step is then preformed and weights the uncertainties from both models and updates the states accordingly. The updated step then becomes the prior in the next iteration. For example, the behaviour of a car can be modeled can be models such that only can move forward and turn sideways but not do a rotation without a velocity. If that behaviour is taken into consideration, such wrong estimates can be compensated for if proposed by the sensors. A visualization of Kalman filtering is shown in figure 2.10.

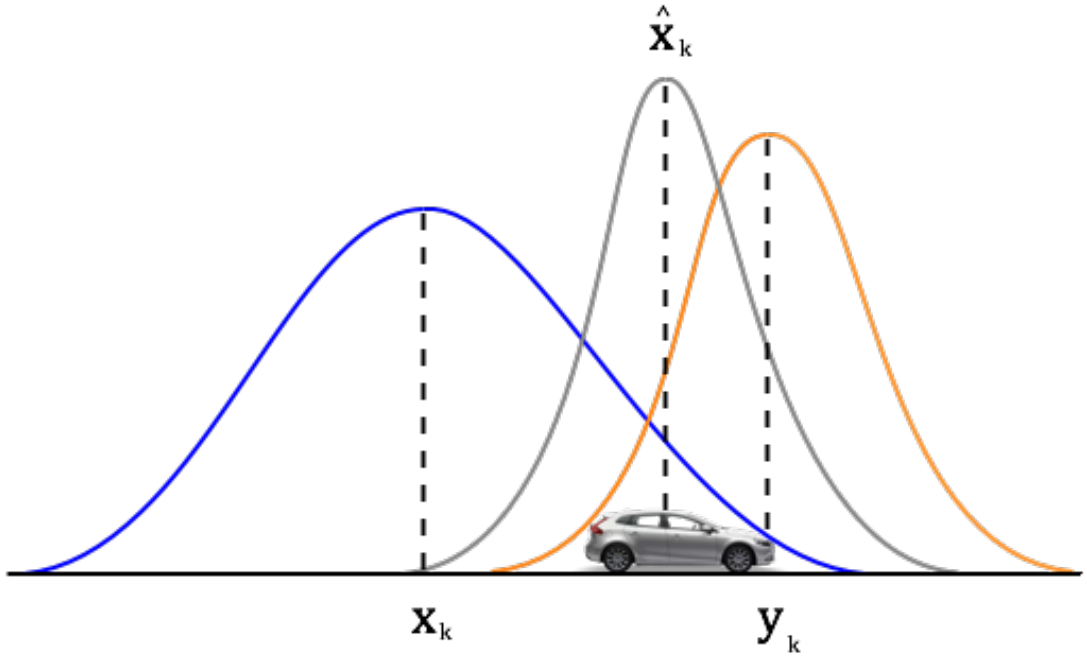


Figure 2.10: Visualization of a Kalman filter. The cars position is predicted to be at x_k while the measurements from the sensors y_k suggests another position. The filter then weights the uncertainties against each other and gives a updates the state accordingly.

2.2.7 Bundle adjustment

Bundle adjustment is a well known method used to solve non-linear least square problems for SfM. The method is almost always used as a final step of feature-based 3D reconstruction. It jointly estimates poses and landmarks positions, see figure 2.11. The goal is to minimize the re-projection error between the image locations of observed and predicted images points. To do this for all poses and points becomes a very computationally heavy process which only grow as more images are added and is thus mostly suited for offline usage. This is referred to as global bundle adjustment (GBA) [19]. An alternative strategy which improves the efficiency of this BA is to only use a limited number of previously added images, a sliding window, for optimization of each new image. By doing so, the number of parameters which are needed to be estimated over time does not increase beyond the size of the window. This is called local bundle adjustment (LBA) and is more fit for real time applications.

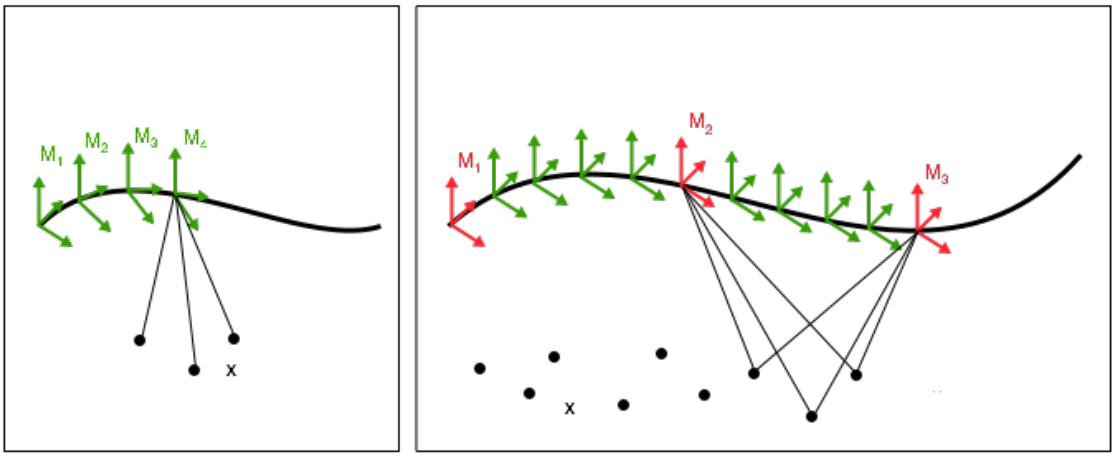


Figure 2.11: The image to the right shows how Bundle Adjustment minimizing the error between the image locations of observed and predicted images points.

2.3 Camera calibration

Camera calibration resolves around estimating the parameters of the lens and location and orientation of the camera relative to another sensor. This has made camera calibration a critical part in subjects such as navigation, robotics and machine vision since these parameters are essential for their functionality.

2.3.1 Intrinsic calibration

The intrinsic calibration compensates for how the lens bends the light when traveling through the pinhole. This can often be seen in images as supposed straight lines bends around the center of the camera due to the lens either being convex or concave. The compensation is done to the camera matrix P, see equation 2.1, which is constructed as,

$$P = K \begin{bmatrix} R & t \end{bmatrix} \tag{2.9}$$

where R is a 3x3-dimensional rotation matrix, t a 1x3 translation vector relative its origin and K the intrinsic matrix and is defined as follows:

$$K = \begin{bmatrix} \alpha_x & \gamma & u_0 & 0 \\ 0 & \alpha_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{2.10}$$

K contains five intrinsic parameters, which are dependent on the focal length, image sensor format and principal point. The parameters, α_x and α_y, are the focal length in terms of pixels, i.e the focal length, f, scaled by the pixel size in x and y direction. γ is the skew coefficient between x and y axis (often equal to 0). The principal point ideally in the centre of the image represents by u_0 and v_0.
As the intrinsic parameters depends on the physical construction of the lens it tends to not change much over time if one calibrated.

2.3.2 Extrinsic calibration

The extrinsic parameters are the translation, t, and the rotation, R (described further in section 2.6.3), which define the position and orientation of the camera in the global frame, see figure 2.12.

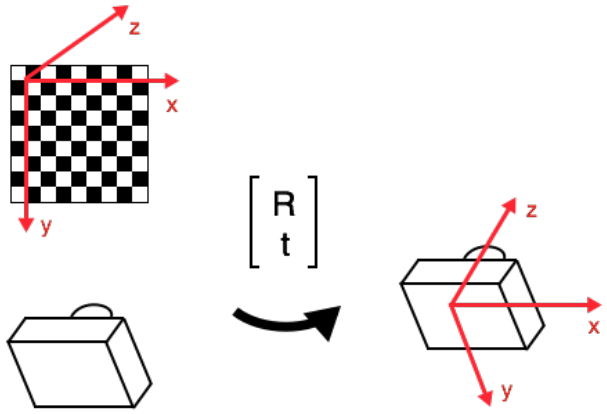


Figure 2.12: Illustration of how the extrinsic parameters are determined. A rotation, R, and a translation, t, describe the transformation between the camera and an object.

2.4 Calibration approaches

Since camera calibration is demanding these days there are several methods developed. The most commonly used methods for extrinsic calibration are presented in section 2.4.1 and 2.4.2.

2.4.1 Direct linear transformation (DLT)

This algorithm solves a set of variables from a set of similarity relations. It is a combination of rewriting the similarity equations as homogeneous linear equations and solving them by standard methods. The similarity equation looks like follows

$$\mathbf{x}_k \sim \mathbf{A}\mathbf{y}_k \quad \text{for } k = 1,...,N \tag{2.11}$$

where **x** and **y** are known vectors and **A** is the calibration matrix (a linear transformation between **x** and **y**).

2.4.2 Zhang's method

This method uses traditional calibration techniques with known targets and self-calibration techniques using the relation between the calibration points from different positions. For this calibration at least two different images of the target are required, either by moving the camera or moving the target. If no intrinsic parameters are given the required number of different images of the target increases to at least three.

2.5 OxTS

The position of the vehicle is given by the measurement unit OxTS[20], shown in figure 2.13. The OxTS records a complete 3D motion and dynamics profile with the GNSS and IMU sensor fusion, see section 2.5.1 and 2.5.2 for more details.

Figure 2.13: OxTS RT3003. Including IMU and GNSS.

2.5.1 Global Navigating Satellite System

GNSS is the standard generic term for satellite navigation system and includes e.g. the GPS, GLONASS, Galileo, Beidou and other regional systems. The benefit of having access to multiple satellite systems is accuracy and availability. If one system fails, other systems can be used instead. The GNSS acquires the coordinates in every position (longitude, latitude and altitude) and by given time stamps the velocity is indirectly given.

2.5.2 Inertial Measurement Unit

An IMU is an electronic device that uses a combination of accelerometer and gyroscope and a magnetometer to estimate the vehicles specific force, angular rate and the magnetic field surrounding the vehicle. Typically an IMU contain one accelorometer, gyroscope and magnetometer per axis which fecilitates estimation of the rotation angles.

2.6 Coordinate frames of reference

In order to properly work with input and output data from different sensors and algorithms it is important that the coordinate systems are interacting correctly with each other. In this scope the following frames are introduced: *global frame, vehicle frame* and *camera frame*.

2.6.1 Global frame

The global frame, G, is a fixed frame in the world and origo is chosen to where the data for estimated position of the camera is initialized.

2.6.2 Vehicle frame

The vehicle frame, V, is fixed to the car and the origin is placed at the centre of the rear wheel axis.

2.6.3 Camera frame

The camera frame, C, is fixed to the camera and origo is the optical centre of the camera. Coordinate systems Measured data can be sampled in different frames, but for comparison between those frames a coordinate transformation of the transformation matrices is required. In order to transform the coordinates, translation and rotation are needed. The transformation of point $\boldsymbol{p}$ is expressed as follows

$$T(\boldsymbol{p}) = R \cdot \boldsymbol{p} + t \tag{2.12}$$

where $T(\boldsymbol{p})$ is the transformed point. The rotation matrix R describes the angles: yaw-pitch-roll and is created by multiplying the rotation around the three axis as follows

$$R(r_z, r_y, r_x) = R_z(r_z) \cdot R_y(r_y) \cdot R_x(r_x) \tag{2.13}$$

where the rotation is counter-clockwise with angles r_z, r_y, r_x are defined as follows

$$R_z(r_z) = \begin{pmatrix} cos(r_z) & -sin(r_z) & 0 \\ sin(r_z) & cos(r_z) & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad R_y(r_y) = \begin{pmatrix} cos(r_y) & 0 & sin(r_y) \\ 0 & 1 & 0 \\ -sin(r_y) & 0 & cos(r_y) \end{pmatrix}$$

$$R_x(r_x) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(r_x) & -sin(r_x) \\ 0 & sin(r_x) & cos(r_x) \end{pmatrix}. \tag{2.14}$$

The coordinate system is body-fixed, which means that the axes are shifted and rotated to a different position. Also known as a Euclidean transform[16]. To calculate the rotation matrix correctly, the multiplying of the rotations has to be done in a particular order. The first rotation is around the z-axis, secondly around the y-axis and at last around the x-axis. Rotations happen in the body-fixed coordinate system.

2.6.4 Homogeneous coordinates

When different transformations are used in the same method it is preferable to use homogeneous coordinates. To get the transformed coordinates $T(\boldsymbol{p})$ we want to multiply the rotation and add the transformation $\boldsymbol{t} = \begin{bmatrix} t_x, t_y, t_z \end{bmatrix} \boldsymbol{p}$. A simple way to do this is to create a transformation matrix M that could be used for all transformations between two specific coordinate systems. $\boldsymbol{p}$ is rewritten as $\tilde{\boldsymbol{p}} = \begin{bmatrix} x, y, z, 1 \end{bmatrix}^T = \begin{bmatrix} \boldsymbol{p}, 1 \end{bmatrix}^T$ and the transformation in equation 2.12 looks like follows

$$T(\tilde{\mathbf{p}}) = M \cdot \tilde{\mathbf{p}} = \left(\begin{array}{ccc|c} & R & & t \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \begin{pmatrix} \mathbf{p} \\ 1 \end{pmatrix} = \begin{pmatrix} R \cdot \mathbf{p} + \mathbf{t} \\ 1 \end{pmatrix} \tag{2.15}$$

where M is the transformation matrix described in section 2.6.5 below. From here on the point $\tilde{\boldsymbol{p}}$ will be denoted as $\boldsymbol{p}$.

2.6.5 Transformation matrices

To transform between different coordinate frames a transformation matrix is used, denoted as follows

$$M^{(frame\ 1)}_{(frame\ 2)} \tag{2.16}$$

where *frame 1* is the current frame and *frame 2* is the frame transforming to. The transformation between camera and global frame can be calculated as

$$M^G_C = M^G_V \cdot M^V_C \quad \textit{(indexes defined in section 2.6).} \tag{2.17}$$

M^V_C contains the vector $\begin{bmatrix} x, y, z, r_x, r_y, r_z \end{bmatrix}$, also called the calibration vector. This vector describes the translation and rotation parameters for the camera with respect to the vehicle. M^G_V, varies over time and is estimated by GNSS and IMU data, which are measured locations of vehicle given as $\begin{bmatrix} x(t), y(t), z(t), r_x(t), r_y(t), r_z(t) \end{bmatrix}$.

2.6.6 Convert 3D to 2D rotation matrix

A rotation in three dimensions can be expressed in two dimensions as a plane rotating around one axis. In two dimensions the rotation matrix around the z-axis is defined as follows

$$R_{z,2D} = \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix} \tag{2.18}$$

where θ is the yaw angle. A rotation matrix in three dimensions can be transformed to Euler angles, i.e *roll, pitch, yaw*, by standard translation algorithms.

2.7 Fit plane to trajectory

To project the trajectory to the globally defined xy-plane, a fitted plane to the trajectory has to be estimated. Figure 2.14 shows a curve in relation to the xy-plane.

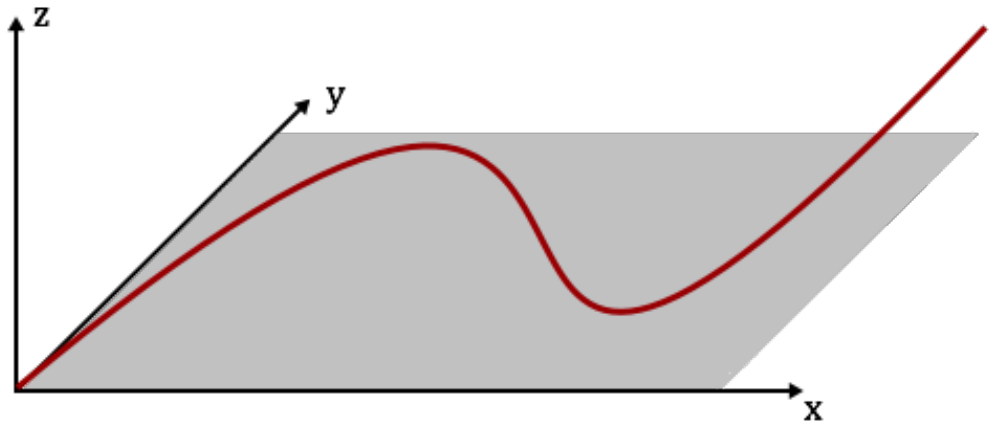


Figure 2.14: Example of a curve in three dimensions, related to the xy-plane.

We have a set of points in 3D and want to fit a plane to them [21]. A plane is generally described by a normal vector $n = [a,b,c]^T$ and a point $p = [x,y,z]^T$ with the distance d on the plane $n \cdot p + d = 0$. This could be written as:

$$ax + by + cz + d = 0. \tag{2.19}$$

Notable is that equation 2.19 is overdetermined and one component needs to be removed. This problem can be solved by assigning the z-component of the plane normal to one, i.e $c = 1$. That gives the following

$$ax + by + z + d = 0$$

$$ax + by + d = -z \tag{2.20}$$

and in matrix form:

$$\begin{pmatrix} x_0 & y_0 & 1 \\ x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots \\ n_n & y_n & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ d \end{pmatrix} = \begin{pmatrix} -z_0 \\ -z_1 \\ \vdots \\ -z_n \end{pmatrix} \tag{2.21}$$

In figure 2.15 the estimated plane is visualized.

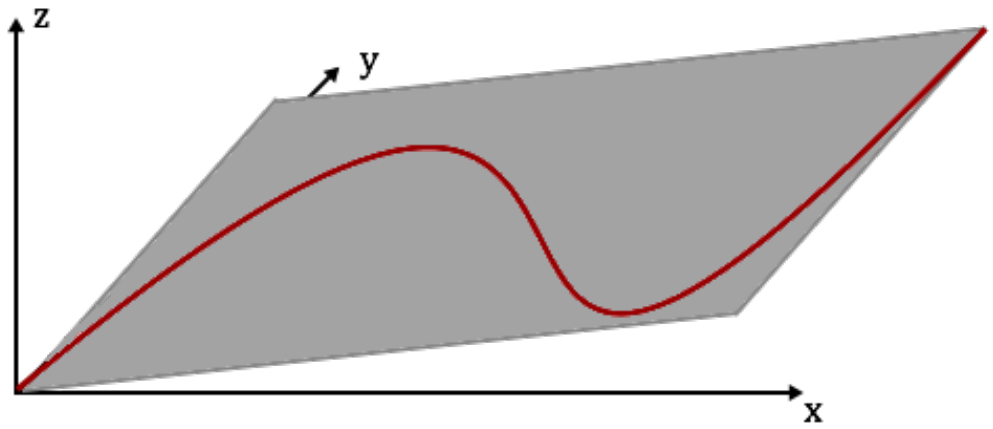


Figure 2.15: Estimated plane fitted to trajectory.

2.8 Projection of a 3D-plane to xy-plane

To project a plane, A, to plane, B, the normal vector, n_A, should be rotated to be parallel to the normal vector,n_B, for plane, B, i.e. the xy-plane. Plane A and B with corresponding normal vectors can be seen in figure 2.16.

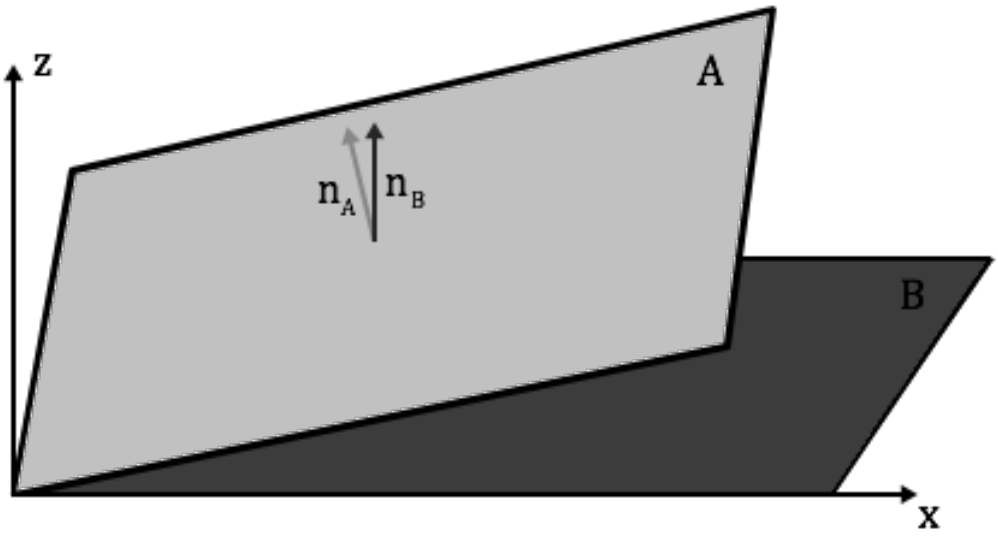


Figure 2.16: Plane A and B with corresponding normal vectors n_A and n_B.

To rotate a plane the rotation matrix, R_v, is needed and defined as:

$$R_v = \begin{pmatrix} x \cdot x \cdot C + c & x \cdot y \cdot C - z \cdot s & x \cdot z \cdot C + y \cdot s \\ y \cdot x \cdot C + z \cdot s & y \cdot y \cdot C + c & y \cdot z \cdot C - x \cdot s \\ z \cdot x \cdot C - y \cdot s & z \cdot y \cdot C + x \cdot s & z \cdot z \cdot C + c \end{pmatrix} \tag{2.22}$$

where

$$c = cos\theta = \frac{M \bullet N}{\|M\| \, \|N\|}, \; s = \sqrt{1 - cos^2\theta}, \; C = 1 - cos\theta$$

and x, y, z are the components representing the axis of the unit vector u, where $u = M \times N / \|M \times N\|$.

2.9 3D point cloud

A point cloud is a set of data points in space [22]. Feature matching and bundle adjustment estimate 3D points that can be visualized in a 3D point cloud. By observing the point cloud, it is easy to see if the 3D points are correctly estimated. The environment is estimated by reconstructed 3D points and in figure 2.17 a 90 degrees curve to the right can be seen.

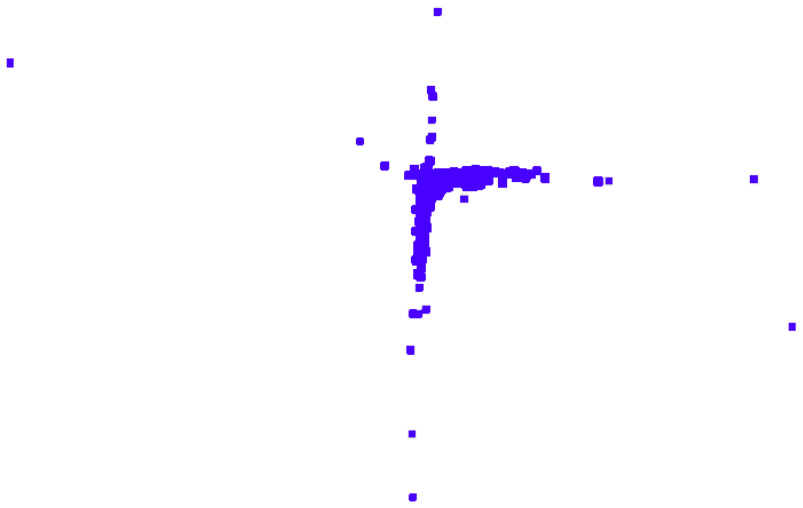


Figure 2.17: 3D point cloud of the estimated environment for the first curve in the KITTI data.

3
Methods

In this chapter the calibration approach is presented. A short argumentation for some of the choices made in an early phase of the thesis, followed by an overview of the entire calibration process will be shown. Thereafter the data sets for testing and verification and finally how the scaling and calibration were done will be presented.

The method chosen is based on functions that are well developed and implemented in Matlab. The frequently used and more complex method, SLAM, was deselected since it was more complicated to control the algorithm and find where to improve it for our approach, step by step. We also chose to not use "loop closure" since we drew the conclusion that it affects the result in an undesirable way. The method is visualized in figure 3.1.

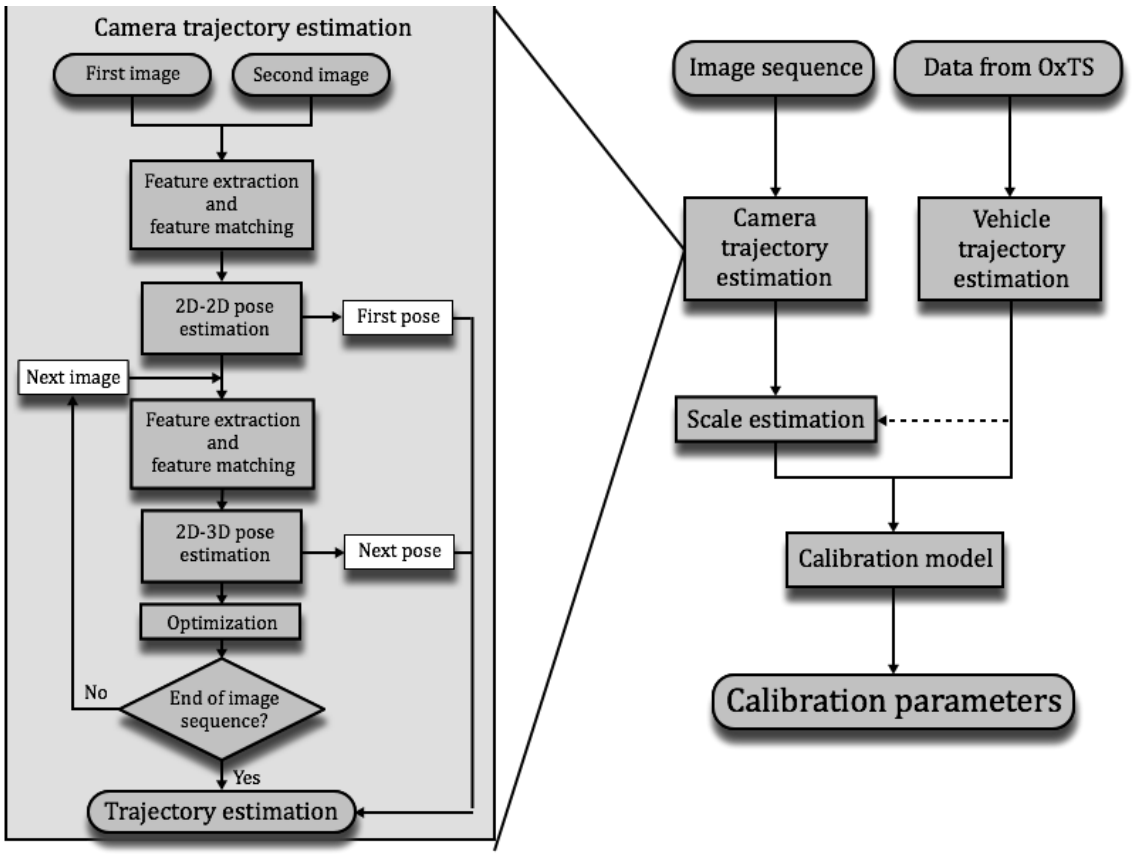


Figure 3.1: Flow chart of the calibration process. The block to the left is describing the block "Camera trajectory estimation" more detailed.

3.1 Test data

During development of a model in Matlab it is preferable to use different types of data in different phases of the process. There are two different types of data, synthetic and real data, presented in section 3.1.1 and 3.1.2. The idea was to use synthetic data in the beginning of the development, but unfortunately the only synthetic data available contained too much motion blur. Thus, the calibration method was developed by using only real data.

3.1.1 Synthetic data

Synthetic data is data created to meet specific needs or certain conditions as realistic behavior profiles. In many situations it is facilitating to use synthetic data, because of the simplicity of verifying the performance of algorithms in an ideal and predetermined environment as ground truth.

3.1.2 Real data

Real data is data measured by a sensor and contains more or less noise.

3.1.2.1 KITTI Vision Benchmark

KITTI Vision Benchmark Suite is a website offering real test data for downloading. It includes a kind of ground truth for every camera, created by other sensors, such as lidar and IMU, which simplifies the verification of the algorithm. However, it does not include ground truth for the vehicle. Therefore we had to create that in some sense. Since we know the extrinsic calibration parameters between the vehicle and the camera, these parameters could be added to the trajectory representing the ground truth for the camera. The ground truth for the vehicle were then used for representing the GPS data from OxTS. The KITTI data is available through the web page: *http://www.cvlibs.net/datasets/kitti/* and the setup of the system is shown in figure 3.2. This thesis will consider only one camera from the KITTI data.

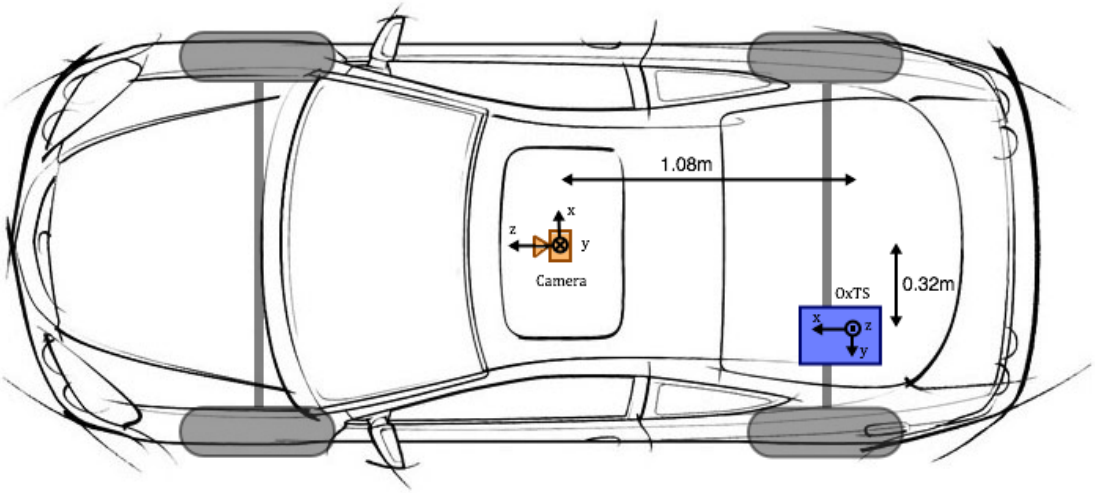


Figure 3.2: Setup for the KITTI data. The distances between and the coordinate systems for the units are defined in the figure.

Figure 3.3 shows the trajectory for the camera in the sequence used from KITTI data.

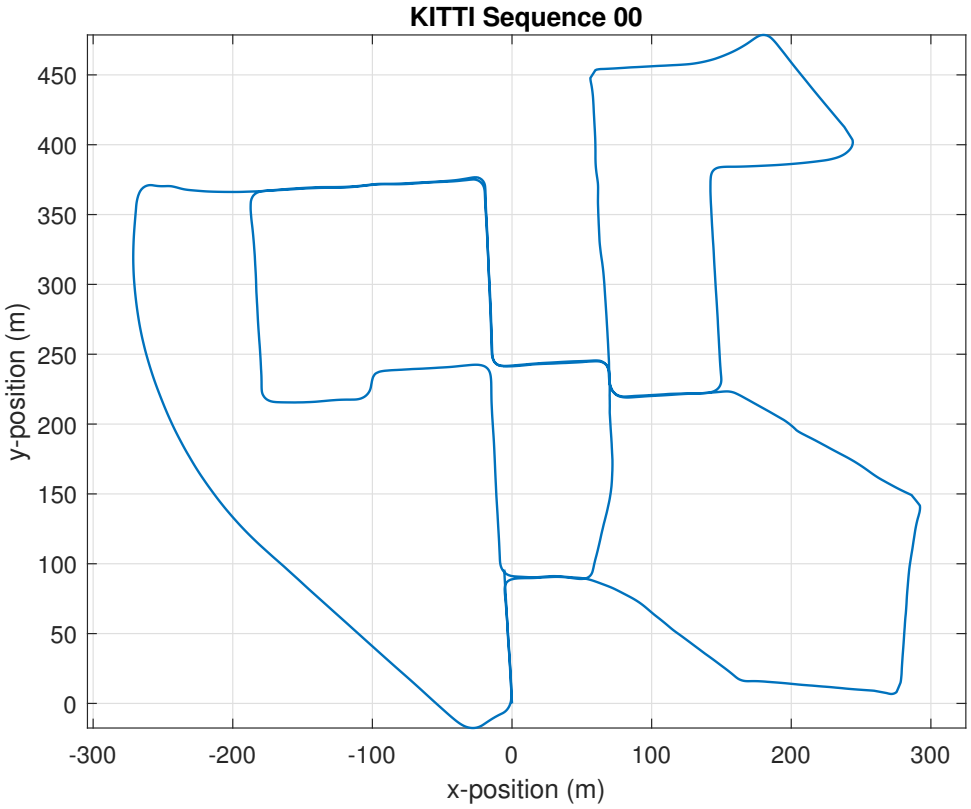


Figure 3.3: Full sequence of the vehicle trajectory. The sequence starts and finishes in origin.

3.1.2.2 Zenuity data

The data, a stream of images and positions in longitude and latitude, provided from Zenuity for this thesis were gathered from a test run where five U-turns were preformed, see figure 4.12. The data were post processed for a more accurate trajectory estimation and could also give the uncertain of each measurement. While the camera and OxTS did not have the same sampling frequency the difference in time between the time steps did not exceed 10 micro seconds which was assumed to be good enough to not do any interpolation of the data.

3.2 Camera trajectory estimation from odometry

The relative pose estimation of the camera is the estimation of the extrinsic parameters, described in section 2.3. From odometry estimation there will always be more or less drift. The longer a trajectory is, the more drift occurs. In order to reduce drift only the curves were considered in the calibration algorithm, and every curve was implemented independently of each other in the calibration algorithm. The method for estimation of the relative pose can be divided into the steps described in section 2.2.3 - 2.2.5.

3.2.1 Scale estimation

In comparison to other sensors, cameras does not make direct numerical measurements of its observations, hence it is impossible to estimate the absolute scale and distance to estimated 3D points and its own trajectory without any sort of reference. This also causes further problem as the scale tends to drift, as a result of small errors in pose estimation which accumulates over time. To compensate for this, continuous information needs to be feed into the system to prevent the drift. To solve this we used the OxTS data was used as a reference of distance traveled over a couple of frames. This, however, could only be used as both the camera and car travels along a straight path since the distance traveled when turning will differ due to different locations in the curve. Thus, the scale was estimated by data before and after a curve. An approximated linear scale parameter, visualized in figure 3.4, was implemented to estimate the camera trajectory in the curve. Linearization was done due to the scale tends to drift approximately linearly.

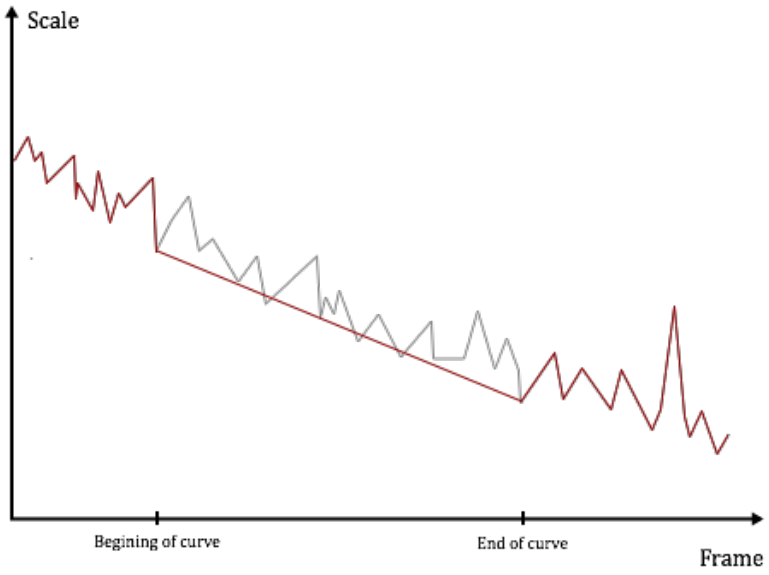


Figure 3.4: Approximated linear scale estimation.

The linear approximation of the scale is given by $y = kx + m$. The gradient $k = \Delta y / \Delta x$. $\Delta y = y_2 - y_1$, where y_1 and y_2 are the mean of the scale for the ten frames before and after the curve, respectively. Δx is the number of frames and $m = y_0$, i.e. the starting value of y.

3.2.1.1 Projection of a trajectory

From the odometry the estimated trajectory for the camera is not parallell to the xy-plane, which is preferable. To project a trajectory to the xy-plane, the coordinates for every frame has to be transformed by the rotation matrix R_v, see section 2.8. The rotated trajectory can be seen in figure 3.5.

$$\hat{t_i} = R_v \cdot t_i \tag{3.1}$$

where $\hat{t_i}$ is the projection of the trajectory t_i from every frame i.

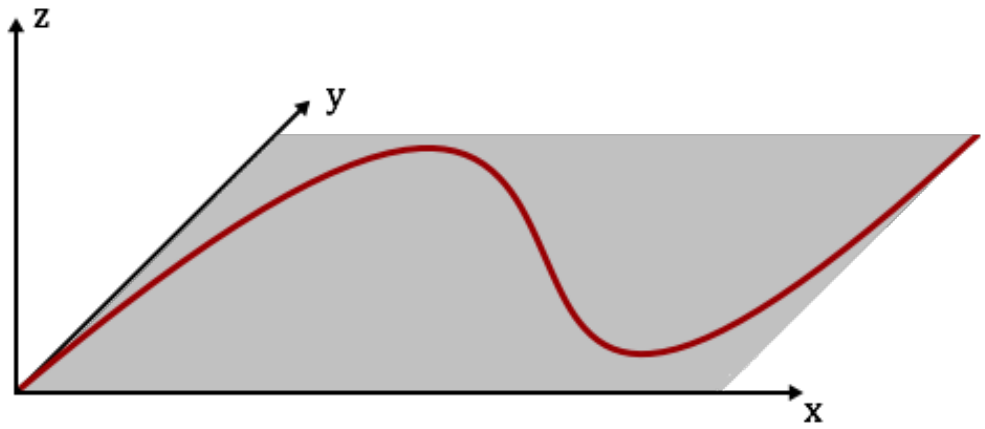


Figure 3.5: Camera trajectory projected to the xy-plane.

3.3 Calibration approach

The fundamental principle of this calibration method resolves around the fact that as the vehicle move, different sensors will register different relative trajectories depending on where on the vehicle said sensors are located. For example, a sensor placed in the front end of the vehicle will start to register a turn before one placed in the back. However, all sensors are fixed to the vehicle and thus the transformation between them is rigid, see figure 3.6. A common choice when modeling a vehicle is to place the origin at the center back axis. In an ideal and simple world a vehicle will move along a perfectly flat road and only make right and left turns. From this simplification, no information about the height (z), pitch nor roll can be extracted since no changes will occur in respect to these parameters.
To estimate the parameters for 3DoF (x, y and yaw) the vehicle has to move in some direction and turn in some sense, i.e there has to be a difference in the variables to be able to estimate the trajectory.

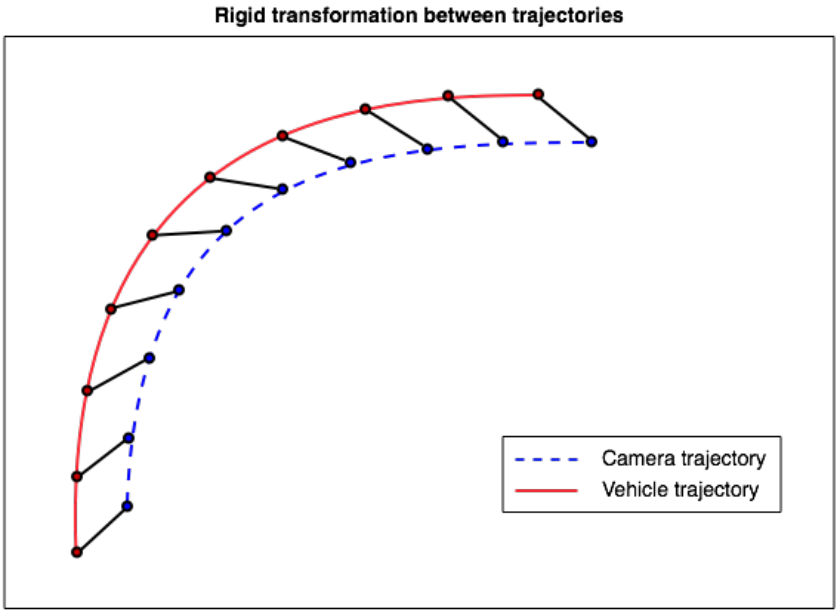


Figure 3.6: The relation between the trajectories. The position and orientation of the camera will always be at the same distance from the

3.3.1 Calibration model

The goal of the calibration is to find a rigid body transformation, i.e. the rotation and translation between the camera frame and vehicle frame. The chosen approach to attempt to solve this problem is commonly referred to as hand-eye calibration. This approach utilizes the fixed transformation between the two homogeneous coordinate frames. By assuming that the transformation from one point in the camera trajectory $P_{C,j}$ can reach the next point in the vehicle trajectory $P_{V,j+1}$ by two different transforms, see figure 3.7. These two possible transforms, which reaches the same frame can be written as:

$$T_{C,i}T = TT_{V,i} \tag{3.2}$$

where T is the transformation matrix i.e. the calibration matrix, described by its translation and rotation. $T_{C,i}$ and $T_{V,i}$ are the transformatiosn between the current and the next frame. This equation can be used for both 2D and 3D calibration as long as it is represented in homogeneous coordinates. For 2D motion, T is expressed as:

$$T = \begin{pmatrix} cos(\theta) & -sin(\theta) & x \\ sin(\theta) & cos(\theta) & y \\ 0 & 0 & 1 \end{pmatrix} \tag{3.3}$$

where x, y and θ are the translation and the yaw angle between the camera and the vehicle. The transformations $T_{C,i}$ and $T_{V,i}$ describe the relation between frames in there own coordinate systems and are calculated as:

$$T_{C,i} = P_{C,j+1} \cdot P_{C,j}^{-1} \tag{3.4}$$

$$T_{V,i} = P_{V,j+1} \cdot P_{V,j}^{-1} \tag{3.5}$$

where $P_{C,j}$ and $P_{V,j}$ are the frames for the camera's and the vehicle's trajectories, respectively.

Since the equation, 3.2 should always hold, it can represented as a minimization problem for each pair of sequential frames. To find the optimal calibration between an estimated trajectory for the camera and the trajectory for the vehicle the following minimization problem needs to be solved:

$$\min_{T} \sum_{i=1}^{N} \|T_{C,i} \cdot T - T \cdot T_{V,i}\|_F^2. \tag{3.6}$$

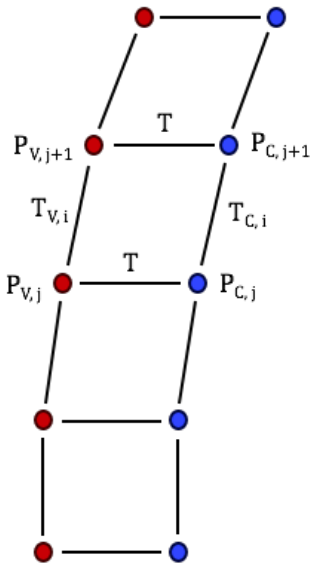


Figure 3.7: The Red dots represent the camera's trajectory and blue dots the vehicle's. T is the transformation between the two trajectories. $T_{C,i}$ and $T_{V,i}$ are the transformation between the current and the next frame for the camera and the vehicles trajectory, respectively and $P_{C,j}$ and $P_{V,j}$ are the frames for the trajectory for the camera and the vehicle, respectively.

4

Results

The results of this thesis is presented in this following chapter.

4.1 Verification of calibration method

The calibration method was verified using ground truth for the KITTI-data. A subsection of the two corresponding trajectories for the vehicle and the camera are shown in figure 4.1. The algorithm gives the transformation, i.e the calibration, between the vehicle and the camera, which is $[x, y, yaw] = [1.08, 0.32, 0]$ as output.

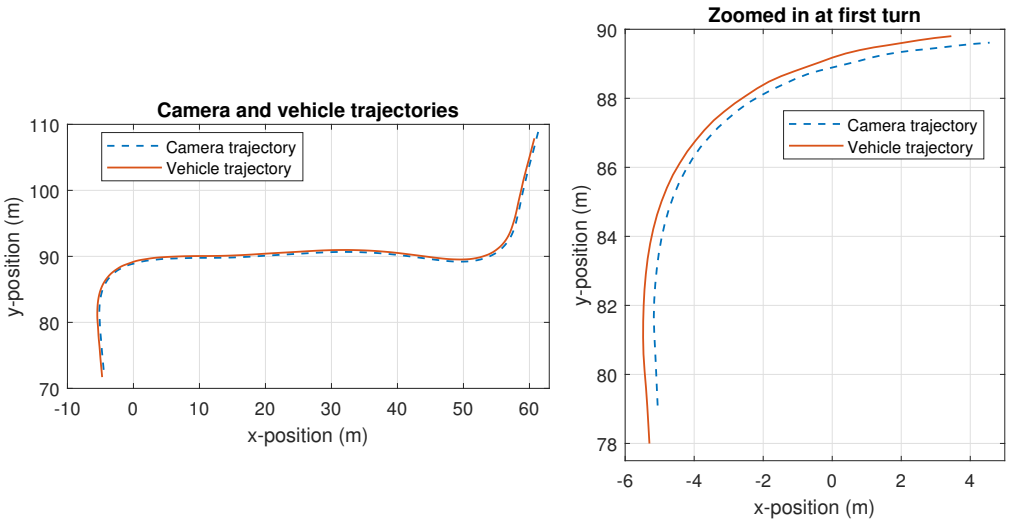


Figure 4.1: Left image: Illustration of the behaviour between camera and vehicle. The rigid transformation enforces a fixed relation between them as the position of the camera will always be at the same distance and angle in comparision to the direction of the vehicle. **Right image:** Zoomed in at the first turn. The fixed relation can more clearly be seen close up and how the camera is positioned in front of the vehicle as it starts and ends ahead of it.

In figure 4.1 the behaviour of the rigid relation between the camera's and the vehicle's trajectory can be seen. Note however that both of these trajectories are described in the same coordinate system which is illustrated by the fixed relation between the two trajectories. This fixed relation would not be seen if the trajectories were define by their

own respective coordinate systems, as both would start at zero which is the case in reality since the unknown relation between them is define by the extrinsic parameters. Nonetheless, the implemented hand-eye calibration method is not affected by global translation nor rotation since the algorithm compares relative trajectories which relates its motion compared to its own relative orientation. For example if one curve is both translated and rotated, the minimization problem can still be solved for the best match. An example of the transformation is shown in figure 4.2.

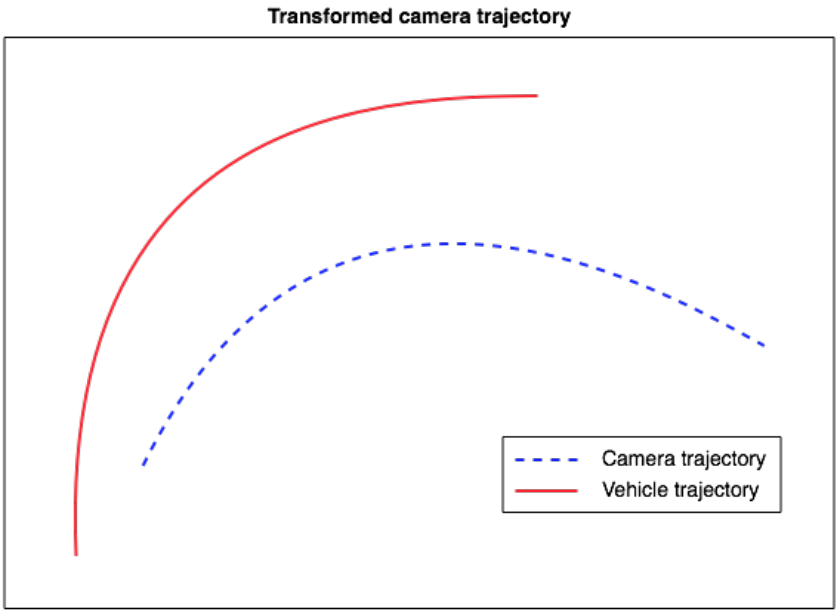


Figure 4.2: Caption

4.2 KITTI data

The trajectory for the test data used for the calibration can be seen in figure 3.3. The full sequence is composed by 4540 images with a resolution of 1241x376 pixels. As mentioned in *Methods* "turn detection" was implemented in the model. Within the sequence a total of 27 identifiable turns could be found, see figure 4.3, which all gave an independent suggestion of the calibration parameters. The majority of the environment throughout the entire sequence is static, i.e. very little motion in traffic and no pedestrians.

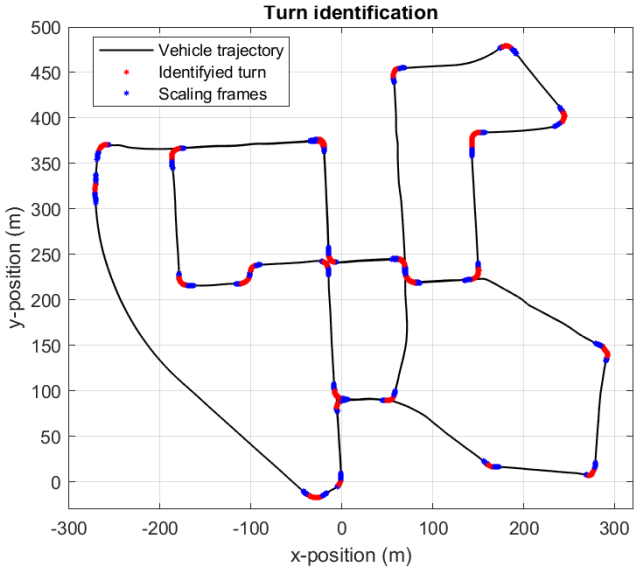


Figure 4.3: Estimation of each frame which lies within a turn. From the initial frames which were deemed to have a change of circa 3 degrees over five frames (red) an additional 10 frames were added to both start and finish of each turn to be used for scale estimation (blue).

4.2.1 Feature detection and matching

Feature detection of the KITTI data can be seen in figure 4.4. A sufficient amount of features were detected in all images and could easily be matched between different frames, see 4.5 and 4.6.

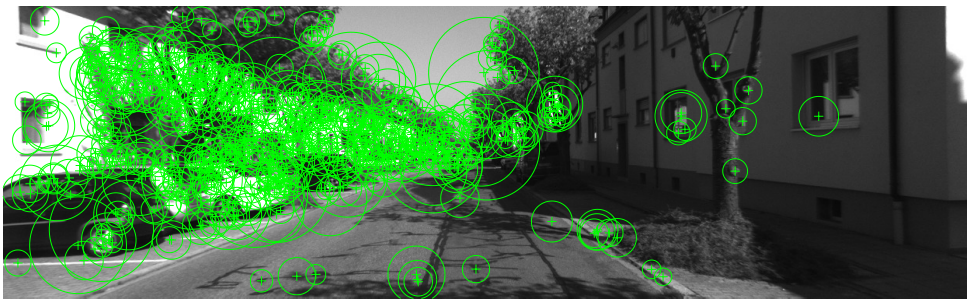


Figure 4.4: Detected SURF-features, the circles sizes are proportional to the scale of the detected feature.

Figure 4.5: Feature matching without outlier rejection. A few noticeable miss matches can be seen as they do not behave correctly in regard to the motion.

Figure 4.6: Feature matching with outlier rejection. The miss matches seen in figure 4.5 are removed, as well as a few other matches that did not fall within the threshold of the re-projection error of the estimated pose.

4.2.2 Scale estimation

The result of the estimated scale compared to the true scale can be observed in figure 4.7. From the result of the calibration it is obvious that the estimated scale in the plot to the right gave a much more accurate result for the calibration parameters, compare to the estimated scale in the left plot.

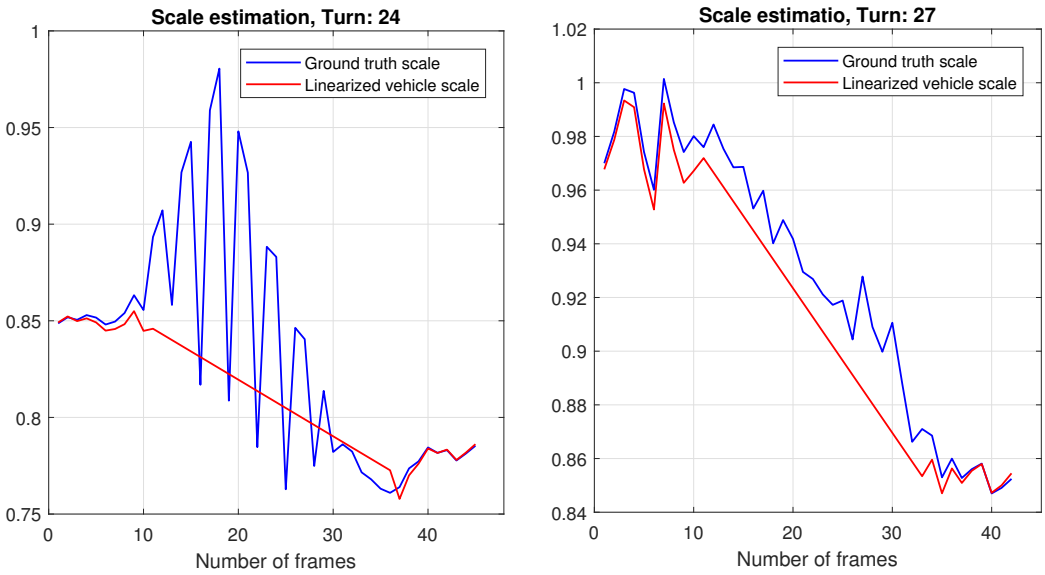


Figure 4.7: Scale estimation. The red curve approximated to be linear when the vehicle is in the curve.

Visualization of some scaled curves and ground truth can be observed in figure 4.8.

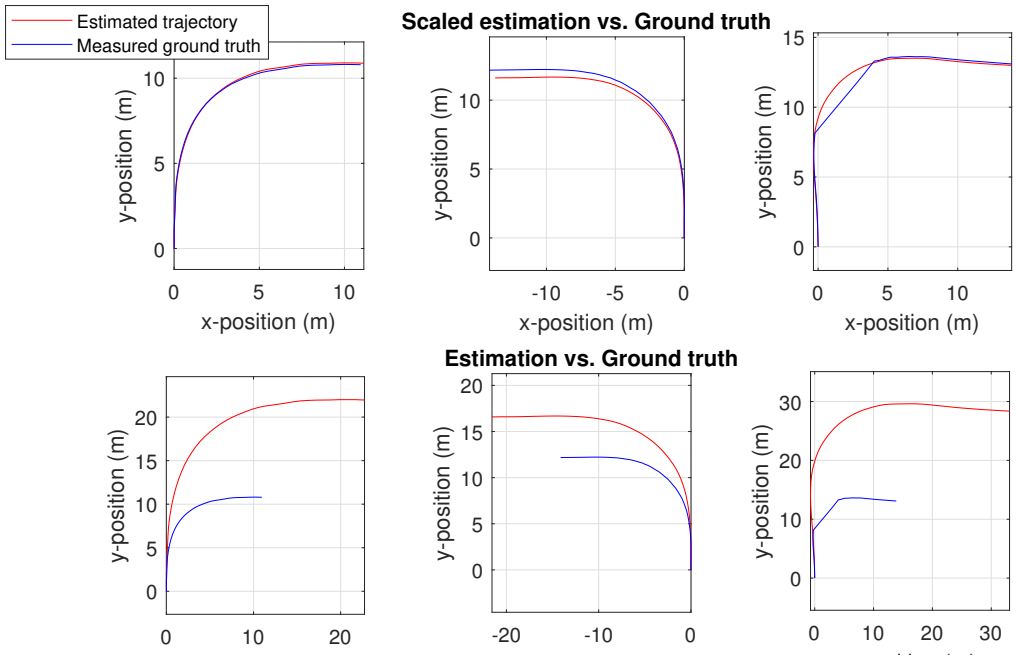


Figure 4.8: The two plots to the left is a good result of a scaled curve, the two plots in the middle are an example of when the scale is not equally good and the plots to the right are an example of when the ground truth data is wrongly estimated.

4.2.3 Plane estimation and trajectory projection

Under the assumption of planar motion, the change along the z-axis of both trajectories should be neglectable. However, as can be seen in figure 4.9, this is not the case for

the ground truth, as difference between the highest and lowest estimated point for the hole sequence ranges up to almost 20 meters, but the motion still seems to be planar. To compensate for this the plane which best represents the trajectory were estimated and projected together with the trajectory to be parallel with the xy-plane. Hence, the coordinate system of the ground truth was orientated in the same way as the estimated plane, even if the original trajectory lied in a small uphill. To be consistent the the trajectory estimated by the visual odometry, also were rotated to the xy-plane. Visulaization of the rotation of the trajectory can be seen in figures 4.9 and 4.10.

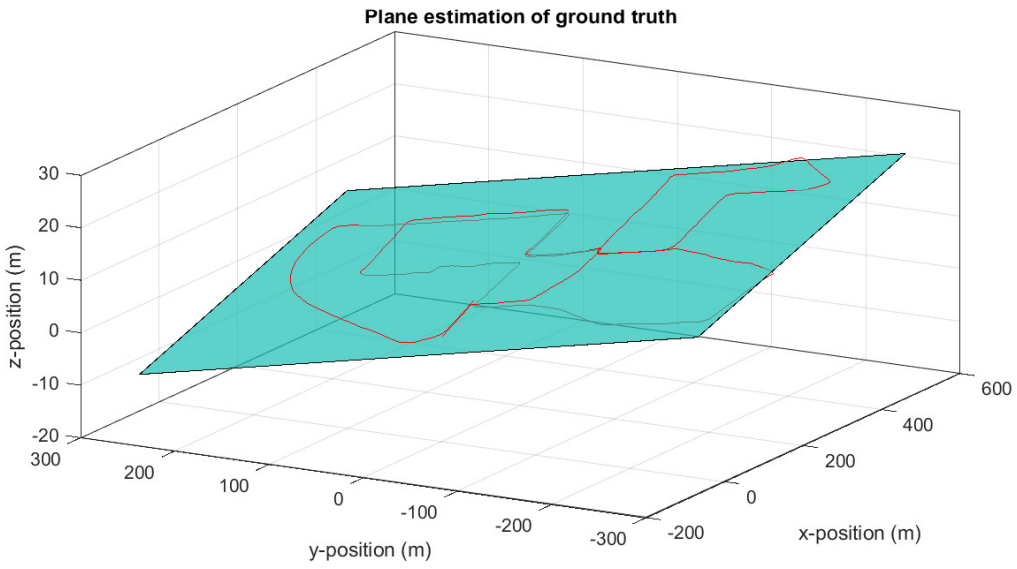


Figure 4.9: 3D plot of ground truth and estimated corresponding plane. From the image sequence a slight uphill could be perceived which seemed to support the change of height throughout the sequence.

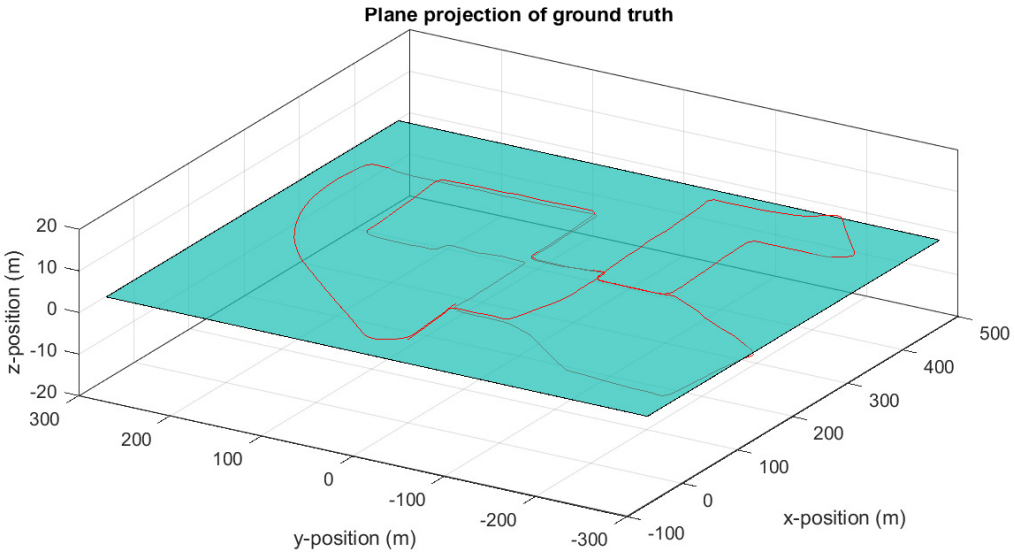


Figure 4.10: Trajectory projected to the xy-plane to support the assumption of planar motion. Over the whole sequence the height never exceeds more then ± 1.2 meter

4.2.4 Calibration results

From each of the turns a separate suggestion for the calibration parameters is given. Histograms illustrating the result for the calibration parameters for x, y and yaw, are shown in figure 4.11 and numerical values are presented in table 4.1. Note that the ground truth sometimes is faulty, as seen in the plots to the right in figure 4.8, the resulting calibration for the x parameter are the staples that falls far away from the mean value, as seen in figure 4.11.

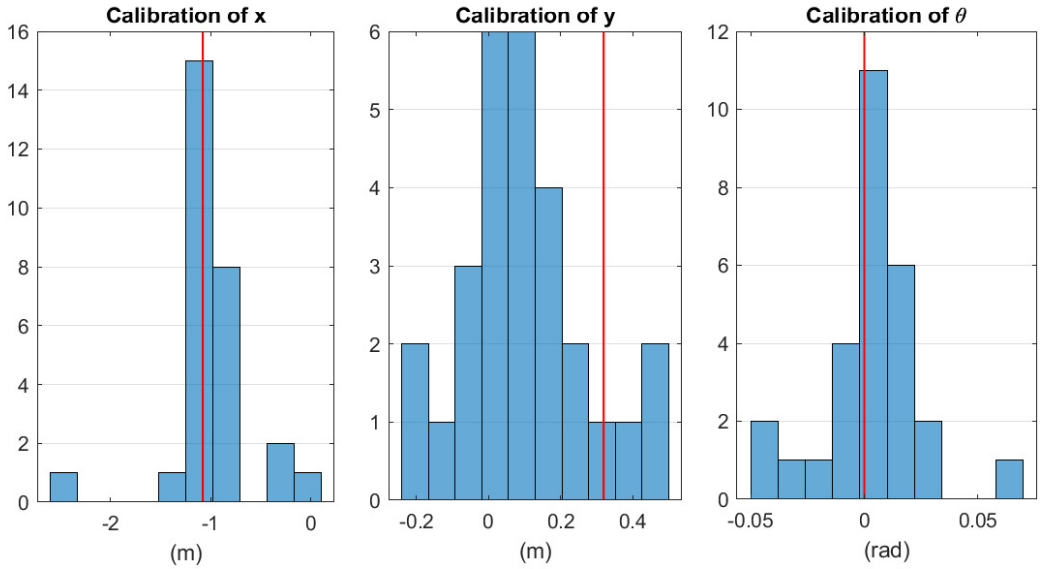


Figure 4.11: Histograms of the results from calibration over 28 curves. In most of the cases the calibration of the x parameter falls fairly close to the true value. In the case of the y parameter the result is slightly skewed to the right compared to the true value while for the yaw angle the calibration falls within less than a tenth of a degree.

From the full sequence, three turns of the ground truth were found to be broken similar to what which can be seen in the right most figures in 4.8. These were, together with the the miss identified turn, removed by hand for the calibrated result presented in table 4.1.

Table 4.1: Results from calibration

Parameters	Mean	Variance
x	-0.963	0.030
y	0.090	0.024
yaw	-0.002	$2.5 * 10^{-4}$

4.3 Collected data by Zenuity

The data of the vehicles trajectory provided by one of Zenuity's own test vehicle can be seen in figure 4.12. After inspecting the corresponding image sequence only three out of the five turns were deemed usable as another car was driving in front of the camera for two of the turn sequences which would interfere with the pose estimation.

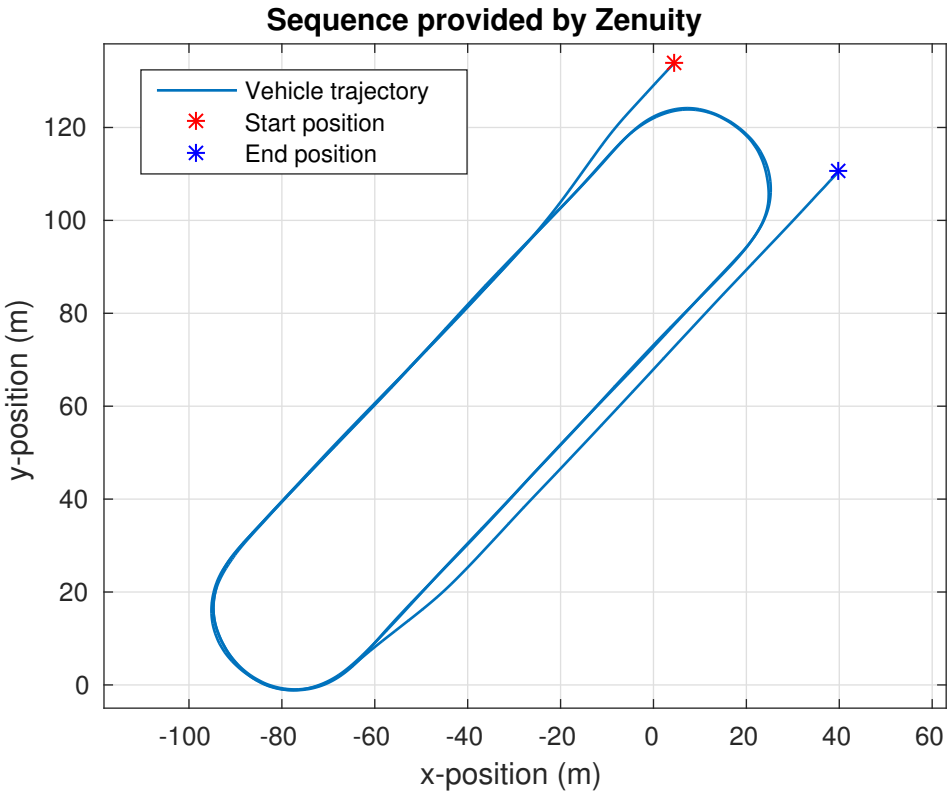


Figure 4.12: Vehicle trajectory estimated from data collected by Zenuity's test vehicle. The trajectory contains five turns before leaving the area.

The estimated camera trajectory for on curve can be observed in figure 4.13. The result should look a smooth U-shaped curve, similar to figure 4.12. As can be observed, the estimation of the curve collapse approximately half way trough the curve. This was the case for all of the three turns which were used which made the full camera trajectory unobtainable.

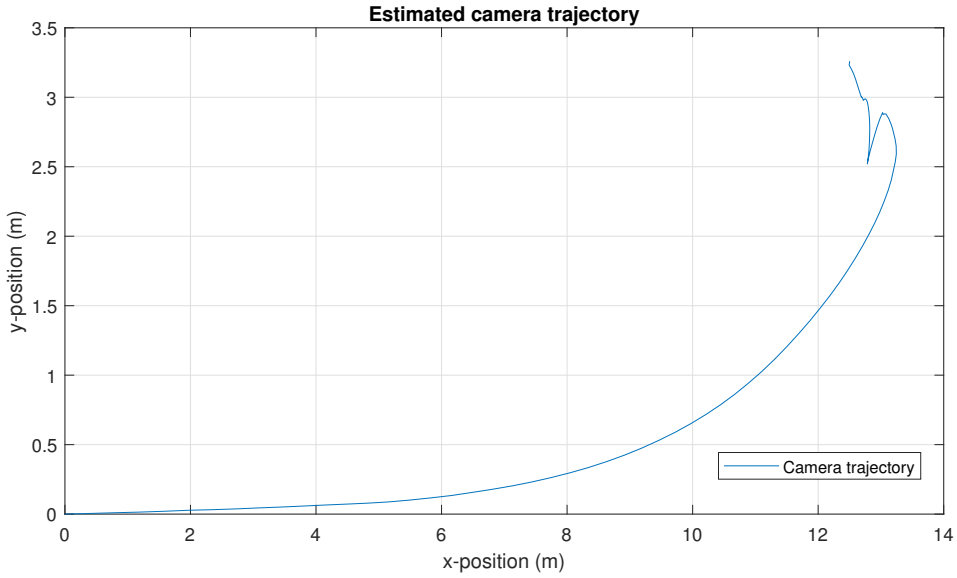


Figure 4.13: Estimated trajectory from Zenuity data. VO manages to estimate the trajectory of the camera about half way but fails thereafter. A similar outcome was seen in the other turns as well.

To be able to analyze the result figures 4.14-4.16 were produced. The figures shows how the environment looks like, how well the result of feature detection and matching was done and at last the 3D point cloud of the reconstructed 3D points.

Figure 4.14: The view of the frame where the algorithm could not perform an reasonable trajectory any further.

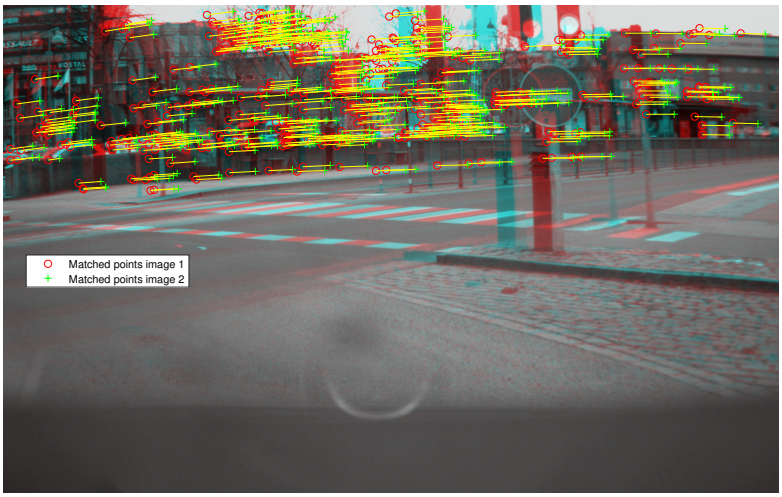


Figure 4.15: Feature detection in the frame where the estimation of the trajectory collapse.

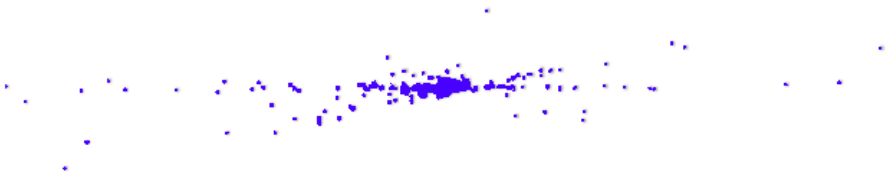


Figure 4.16: 3D point cloud constructed by the data from the estimated camera trajectory from image sequence collected by Zenuity.

5
Discussion

In this chapter the result is discussed and conclusions are drawn. Examples of possible improvement and reasons of impact of the result are presented.

5.1 Calibration model

[kommentera numeriska värden från tabell] The result shows that the calibration method works properly, since the verification shows that it provides the correct calibration when using ground truth data. The result from KITTI data does not represent the reality completely, since the uncertainty from the OxTS were not considered. The idea was to estimate the vehicles trajectory from the OxTS from the data produced by Zenuity, but since the trajectory estimation gave a unusable result, the trajectory for the vehicle was never estimated. To evaluate the estimated trajectory for the vehicle, ground truth data is also required and not available.

5.2 Zenuity data

The estimation of the trajectory for the camera from the image sequence collected by Zenuity did not give a good result, observed in section 4.3. To analyze why the algorithm for estimating the trajectory did not perform a good result the following questions were stated:

1. At which frame does the algorithm collapse and how does it look?
2. Are there enough features detected in the images and are feature matching possible?
3. How does the 3D-point cloud look?

Answers:

1. In figure 4.14 we can obtain the frame where the algorithm are not able to further estimate the trajectory correctly. The frame seems to be in good condition but some dynamic objects can be found in all three sequences but with little movement.
2. Figure 4.15 shows that the features detected and matched should perform a fairly good result. Hence, the feature detection should not be the reason for the bad result.
3. The 3D-point cloud in figure 4.16 shows that something is wrong with the reconstructed 3D-points. If the 3D-point where correctly estimated we should have been obtaining a point cloud formed as a curve.

The above stated observations indicates that something is wrong with the camera. Information were given that changes of the lens had been done manually and that could be a fair reason why the result cannot be performed any better than in figure 4.13.

5.3 Calibration result

In section 4.2.4 it can be seen that x and θ are estimated fairly good, while y is not estimated with the same accuracy. The reason for this could be that the movement along the x axis is much larger compared to what it is along the y axis. A small error in the estimation does affect the result relatively much more in this case.

5.4 Future work

There are multiple ways to improve the result. Below some of them has been stated:

1. The most fundamental and relevant things to complete in future work would be for Zenuity to use a different camera, alternatively find out what is wrong with the currently used one, and also to be able to estimate the ground truth position of the vehicle.
2. To improve the scale estimation one could for example identify objects or landmarks, known by there size, to improve the accuracy of the reconstructed 3D-points.
3. Visual odometry causes drift that we are not able to compensate for completely. One possible way to reduce the drift could be to improve the input data, by for example using better cameras that generate images of higher quality.
4. Another method to improve the algorithm could be to only do the calibration in static environments as dynamically moving objects such as other vehicles can make correct pose estimation unfeasible if not identified and removed correctly.
5. To analyze the calibration method further one could use cameras mounted on other locations on the vehicle. A view sideways gives perhaps more complex information about the movement of the vehicle.

5.4.1 Estimation of roll, pitch and z

The method presented in this thesis was only adapting three parameters out of six. However, there are methods for estimating the other three parameters, but these methods uses different approaches. To estimate the roll and pitch angle the horizon line can be used. The horizon is a fixed reference, thus, the relative roll of the camera can be estimated. Since the rotation angles are fixed to each other, the pitch angle can be estimated as well. The parameter z can be estimated from triangulation of lane markings.

5.5 Conclusion

The results show that this algorithmic approach can determine the general region of which the camera is located but is unable to accurately find its absolute position. This however is done without any knowledge about its surrounding and with very little human input. The

estimation of the scale has shown to play a large part of how accurate the parameters can be estimated, and another method, reliable in turns as well, might be required. Further testing of different types of setups, environments and conditions, such as velocity and turn rate, is required as this thesis became limited by time and the data sets available.
The conclusion we drew about the data from Zenuity is that the camera used was in bad condition and for further investigation ground truth data is required as a reference to verify the targetless calibration.

Bibliography

[1] K. L. Movig, M. Mathijssen, P. Nagel, T. Van Egmond, J. J. De Gier, H. Leufkens, and A. C. Egberts, "Psychoactive substance use and the risk of motor vehicle accidents," *Accident Analysis & Prevention*, vol. 36, no. 4, pp. 631–636, 2004.

[2] R. Y. Tsai and R. K. Lenz, "A new technique for fully autonomous and efficient 3d robotics hand/eye calibration," *IEEE Transactions on robotics and automation*, vol. 5, no. 3, pp. 345–358, 1989.

[3] R. Horaud and F. Dornaika, "Hand-eye calibration," *The international journal of robotics research*, vol. 14, no. 3, pp. 195–210, 1995.

[4] G. Antonelli, F. Caccavale, F. Grossi, and A. Marino, "Simultaneous calibration of odometry and camera for a differential drive mobile robot," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 5417–5422.

[5] F. Pagel, "Motion adjustment for extrinsic calibration of cameras with non-overlapping views," in *Computer and Robot Vision (CRV), 2012 Ninth Conference on*. IEEE, 2012, pp. 94–100.

[6] Y. Cheng, M. Maimone, and L. Matthies, "Visual odometry on the mars exploration rovers," in *Systems, Man and Cybernetics, 2005 IEEE International Conference on*, vol. 1. IEEE, 2005, pp. 903–910.

[7] M. Dunbabin, J. Roberts, K. Usher, G. Winstanley, and P. Corke, "A hybrid auv design for shallow water reef navigation," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE, 2005, pp. 2105–2110.

[8] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 1. Ieee, 2004, pp. I–I.

[9] M. O. Aqel, M. H. Marhaban, M. I. Saripan, and N. B. Ismail, "Review of visual odometry: types, approaches, challenges, and applications," *SpringerPlus*, vol. 5, no. 1, p. 1897, 2016.

[10] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[11] M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, "A

solution to the simultaneous localization and map building (slam) problem," *IEEE Transactions on robotics and automation*, vol. 17, no. 3, pp. 229–241, 2001.

[12] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[13] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *European conference on computer vision*. Springer, 2006, pp. 404–417.

[14] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," in *Readings in computer vision*. Elsevier, 1987, pp. 726–740.

[15] D. Nistér, "An efficient solution to the five-point relative pose problem," *IEEE transactions on pattern analysis and machine intelligence*, vol. 26, no. 6, pp. 756–770, 2004.

[16] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[17] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o (n) solution to the pnp problem," *International journal of computer vision*, vol. 81, no. 2, p. 155, 2009.

[18] B. Kitt, A. Geiger, and H. Lategahn, "Visual odometry based on stereo image sequences with ransac-based outlier rejection scheme," in *Intelligent Vehicles Symposium (IV), 2010 IEEE*. IEEE, 2010, pp. 486–492.

[19] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment—a modern synthesis," in *International workshop on vision algorithms*. Springer, 1999, pp. 298–372.

[20] O. T. S. Ltd, "Precision reference for oxts inertial+gnss, rt3000," 1998. [Online]. Available: http://www.oxts.com/products/rt3000/?lang=sv

[21] P. Skulimowski, M. Owczarek, and P. Strumillo, "Door detection in images of 3d scenes in an electronic travel aid for the blind," in *Image and Signal Processing and Analysis (ISPA), 2017 10th International Symposium on*. IEEE, 2017, pp. 189–194.

[22] A. Golovinskiy, V. G. Kim, and T. Funkhouser, "Shape-based recognition of 3d point clouds in urban environments," in *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 2154–2161.