Visual Perception for Autonomous Driving

Dissertation zur Erlangung des Grades
eines Doktors der Naturwissenschaften (Dr. rer. nat.)
am Fachbereich Mathematik und Informatik
der Freien Universität Berlin

von

Tobias Langner

Berlin
November 15, 2019

Betreuer: Prof. Dr. Rául Rojas

Erstgutachter: Prof. Dr. Rául Rojas

Zweitgutachter: Prof. Dr. Manfred Hild

Disputation am 29.01.2020

Summary

This thesis describes my work in the AutoNOMOS project, dedicated to the advancement of autonomous driving. Our team of researchers demonstrated autonomous driving on the German Autobahn and in the inner city traffic of Berlin, mostly relying on LIDAR scanners and an Applanix POS LV 220 for self-localization. These sensors are highly precise, yet expensive. This prompts two questions: Can we achieve reliable environmental perception at hardware costs suitable for mass-market applications? And secondly, can we achieve human-like or even superior sensing capabilities, yet use only passive human-like sensing modalities?

Therefore, this thesis presents computer vision algorithms based on an embedded stereo camera that was developed in our research group. It processes stereoscopic images and computes range data in real-time. The sensor's hardware is inexpensive, yet powerful.

This dissertation covers four important aspects of visual environmental perception. First, the camera's pose must be precisely calibrated in order to interpret visual data within the car's coordinate system. For this purpose, an automatic calibration is proposed that does not rely on a calibration target, but rather uses the image data from typical road scenes.

The second problem solved is free-space and obstacle recognition. This thesis presents a generic object detection system based on the stereo-camera's range data. It detects obstacles using a hybrid 2D/3D segmentation method and employs an occupancy grid to compute drivable and occupied space.

Traffic lights detection at intersections is the third problem solved. Our autonomous car was equipped with two monocular cameras behind the windshield to cover a maximal field of view. The detection is steered by the vehicle's planning module and digital maps annotated with the GPS positions of oncoming traffic lights. The traffic light detection system has also been tested with the stereo camera in order to map the locations of yet unknown traffic lights.

The fourth and last problem solved is estimating the car's ego-motion based on the stereo camera's range measurements and sparse optical flow. The presented algorithm yields the car's linear and angular velocity at reasonable accuracy, offering a low-cost alternative to some parts of the Applanix POS LV 220 reference system.

The integral parts of all presented algorithms rely only on visual data. Thus, the methods are applicable not only to our vehicle but also to other autonomous cars and robotic platforms.

Zusammenfassung

Diese Dissertation fasst meine Arbeit im AutoNOMOS-Projekt zusammen, dass sich der Erforschung des autonomen Fahrens widmet. Unsere Forschungsgruppe demonstrierte autonomes Fahren auf der Autobahn und in der Innenstadt von Berlin, wobei sich das Fahrzeug größtenteils auf LIDAR-Sensoren und das Applanix POS LV 220 zur Lokalisierung verließ. Diese Sensoren sind sehr präzise, aber auch sehr teuer. Daher stellen sich zwei Fragen: Kann man zuverlässige Umfeldwahrnehmung auch mit kostengünstiger Sensorik erreichen? Und zweitens, kann man mit passiven Kamerasensoren menschlichliche Fähigkeiten erreichen oder sogar übertreffen?
Deshalb präsentiert diese Arbeit Computer-Vision-Algorithmen basierend auf einer intelligenten Stereokamera, die in unserer Forschungsgruppe entwickelt wurde. Die Kamera nimmt Stereobilder auf und berechnet Entfernungsdaten in Echtzeit. Dieser Sensor ist preisgünstig und doch leistungsfähig.
In dieser Dissertation werden vier wichtige Aspekte der Umweltwahrnehmung aufgegriffen. Erstens, muss die Kamerapose bezüglich des Fahrzeugs kalibriert werden, um Sensordaten korrekt interpretieren zu können. Deshalb wird ein Kalibrierungsverfahren vorgestellt, dass ohne Kalibriermuster auskommt und stattdessen Bilder von normalen Fahrten verwendet.
Als Zweites wird ein generisches System zur Hindernis- und Freiraumerkennung vorgestellt, das auf den Tiefendaten der Stereokamera basiert. Es erkennt mittels einer hybriden 2D/3D-Segmentierung Hindernisse und ermittelt mit einer Belegtheitskarte den Freiraum und nichtbefahrbare Flächen.
Das dritte bearbeitete Problem ist die Ampelzustandserkennung. Unser autonomes Fahrzeug wurde zu diesem Zweck mit zwei monokularen Kameras ausgestattet, um einen möglichst großen Blickbereich abzudecken. Das hier präsentierte Verfahren wird durch das Fahrpanungsmodul und kartierte GPS-Positionen der vorausliegenden Ampeln angesteuert. Das System wurde auch mit der Stereokamera getestet, um noch nicht kartierte Ampeln zu vermessen und einzutragen.
Das vierte und letzte Problem ist die Abschätzung der Eigenbewegung anhand der Stereotiefendaten und des optischen Flusses. Das vorgestellte Verfahren berechnet lineare Geschwindigkeit und Winkelgeschwindigkeiten des Fahrzeuges in akzeptabler Genauigkeit und bietet damit eine kostengünstige Alternative für Teile des Applanix POS LV 220 Referenzsystems.
Die Hauptbestandteile der präsentierten Algorithmen benötigen nur visuelle Daten und sind daher auch auf andere autonome Fahrzeuge und mobile Robotikplattformen anwendbar.

Acknowledgment

I want to thank my advisor Prof. Dr. Rául Rojas for the continuous support of my Ph.D study and related research, for his patience, and motivation without I would not have finished this thesis.

Also, I want to thank all my colleagues and fellow Ph.D students for inspiration and insightful discussions about mathematical and algorithmic problems, in no particular order: Andreas Philipp, Bennet Fischer, Claas Norman Ritter, Daniel Goehring, Daniel Neumann, Daniel Seifert, David Latotzky, Ernesto Tapia, Fritz Ulbrich, Georg Bremer, Manuel Schwabe, Miao Wang, Michael Schnürmacher, Nicolai Steinke, Ricardo Carillo, Robert Spangenberg, Stephan Sundermann, Tinosch Ganjineh, Zahra Boroujeni.

Last but not the least, I would like to thank my family and friends for supporting me throughout writing this thesis and my life in general.

Contents

Chapter 1

Introduction

1.1 Motivation

MadeInGermany, the autonomous car of Freie Universität Berlin, was purchased in 2010 to pioneer the research on autonomous driving in the city of Berlin. Perception is one of the key tasks for an autonomously driving robotic car and necessary to plan and follow efficient and safe trajectories. The car is equipped with a wide variety of sensors, i.e. LIDARs, radars, GPS, IMU, accelerometer, odometry, and cameras. My thesis addresses several important applications of visual perception, which have proven useful for our demonstration of autonomous driving in the city of Berlin:

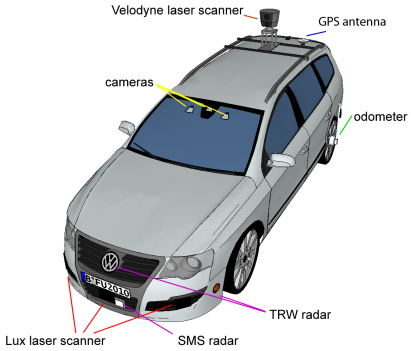
1. **The extrinsic calibration of cameras from normal driving through urban streets.**
   The fusion and interpretation of all sensor measurements requires an exact calibration of the sensor-to-car transformation. Especially the calibration of the rotation angles is of high importance, because angular errors have a larger impact on the accuracy of long-range measurements. There are various techniques for the calibration task that use a dedicated calibration target, e.g. a chessboard pattern. However, there were cases when cameras where mounted ad-hoc on *MadeInGermany* and the specific camera pose was lost after the experiment. Here a calibration method must work with the available image data. Chapter 3 introduces a novel method to calibrate the rotation angles of a forward-facing camera. It uses optical flow to perform the calibration without a target pattern.

2. **The detection of other traffic participants using stereoscopic vision.**

A vital part of autonomous driving is the avoidance of collision with other traffic participants or obstacles. Most of the intelligent vehicles rely on LIDAR-based obstacle detection and tracking systems. LIDARs offer high accuracy but the sensors are quite expensive compared to cameras and they lack spatial resolution. Our research group has developed an embedded stereo camera which, like a LIDAR, delivers 3D data in real-time. An onboard hybrid FPGA and CPU system performs the image processing and provides disparity maps with 30 frames per second. Chapter 4 presents my approach to the detection of generic obstacles and estimation of free space on basis of this data. The results are compared to measurements from LIDAR sensors mounted on *MadeInGermany*.

3. **The recognition of traffic light states.**
   Safe autonomous driving requires reliable and early detection of the state of oncoming traffic lights. Chapter 5 presents my image-processing-based approach. It is assisted by an annotated map, that delivers the exact position and height of traffic lights. With the knowledge of intrinsic and extrinsic parameters and the car's global position, the traffic lights are projected to regions of interest in the image. This approach facilitates the detection task and effectively reduces the false positive rate. Maps with annotated positions of each single traffic light are not readily available. Therefore, the second part of this chapter uses the obstacle detection from Chapter 4 to detect objects whose shape resembles those of traffic lights. These objects are projected to the camera image to generate 2D regions of interest for the traffic light detection module. If a traffic light has been successfully detected and tracked over a certain time, the tracked positions are filtered and stored in a map.

4. **Ego-motion estimation using optical flow and disparity maps.**
   The calibration method from Chapter 3 relied on the optical flow caused by the vehicle's movement. However, the movement of non-static objects also induces optical flow and its detection and removal is an important task of visual ego-motion estimation. Even the most robust estimators may fail if large objects cross the cameras field of view in close range. Chapter 6 presents a novel approach which uses the disparity maps from our embedded stereo-camera to facilitate the segmentation of optical flow and robustly estimate the vehicle's linear and angular velocity from the optical flow.



(a) *MadeInGermany* is equipped with various sensors for perception and localization. Source: [2]

(b) *MadeInGermany* on the test area at the former airport Tempelhof.

Figure 1.1: The autonomous car *MadeInGermany*.

1.2 MadeInGermany - The Automonous Vehicle at Freie Universität Berlin

My thesis is based on research and experiments conducted in the AutoNOMOS project (and follow-up projects) with the research group *Intelligent Systems and Robotics* (now known as *Dahlem Center for Machine Learning and Robotics*) at Freie Universität Berlin. After the successful participation in the DARPA Urban challenge in 2007, the research group had acquired a new robotic car in 2010 funded by the German Ministry for Education and Research (BMBF). The car was equipped with a redundant sensor setup such that many different combinations could be combined and compared to each other.

Our car *MadeInGermany* (depicted in Fig. 1.1) is a series Volkswagen Passat equipped with a very precise localization system from Applanix, the Position and Orientation System for Land Vehicles (POS LV 220)[1]. It consists of an odometer, an intertial sensor for linear accelerations and angular rates (IMU) and a GPS receiver. An embedded computing unit combines all sensor data to provide smooth and accurate values for angular and linear position, velocity and acceleration. Additionally, we use a mobile data connection to receive DPGS correction data which reduces, under optimal conditions, the localization error to a few centimeters.

Integrated into the front and rear bumper, there are six Ibeo Lux laser scanners[3], together providing a 360° surround view. Each scanner has 4 lasers rotated by an internal mirror giving a horizontal field of view of 110°

Figure 1.2: Two AV Guppy F-036C cameras were mounted with suction cups behind the windshield. The camera are pointed upwards and sideways to maximize the covered field of view for close-range traffic light detection.

and a vertical field of view of 3.2°. An embedded computing unit merges the sensor data and provides a black-box object tracking and classification.
On top of the roof is the rotating Velodyne HDL-64E S2 laser scanner[4]. It has 64 lasers arranged vertically and rotates with a frequency of 10 Hz, delivering 1.3 million points per second. The scanner covers a vertical field of view of 26.8°. The higher position allows this scanner to detect objects behind other cars. The Velodyne scanner provides 1.333 million points per second and measures distances up to 120 m with a precision of 0.2 mm.
For computer vision different camera setup had been tested. For traffic light detection we have used two Allied Vision Guppy F036-C cameras[5]. These are two monocular camera mounted with suction cups behind the windshield. As can be seen in (Fig. 5.1), the cameras were rotated to the top left and top right corner of the front window. This is necessary to see traffic lights when the car stands in the front row at an intersection. The cameras deliver images at a resolution of 752×480 and support high dynamic range to mitigate the problem of over-saturation.
The embedded stereo camera developed in our work group (Fig. 1.3) uses, like the Guppy cameras, an HDR imaging sensor with a 752×480 resolution. The stereo camera used for the experiments in this thesis has a baseline of 25 cm and a horizontal field of view of 60°. It provides stereo images and depth images at a rate of 30 Hz. An internal ARM Cortex-A9 processor and a Xilinx ZYNQ FPGA perform the data processing and consume only about 6 W which makes it an excellent choice for automotive and other mobile applications. The camera removes lens distortion and rectifies the images before creating a dense depth image by running a block-matching algorithm on census-transformed images[6]. Technical specifications and details of the algorithms can be found in the PhD thesis of Bennet Fischer[7].

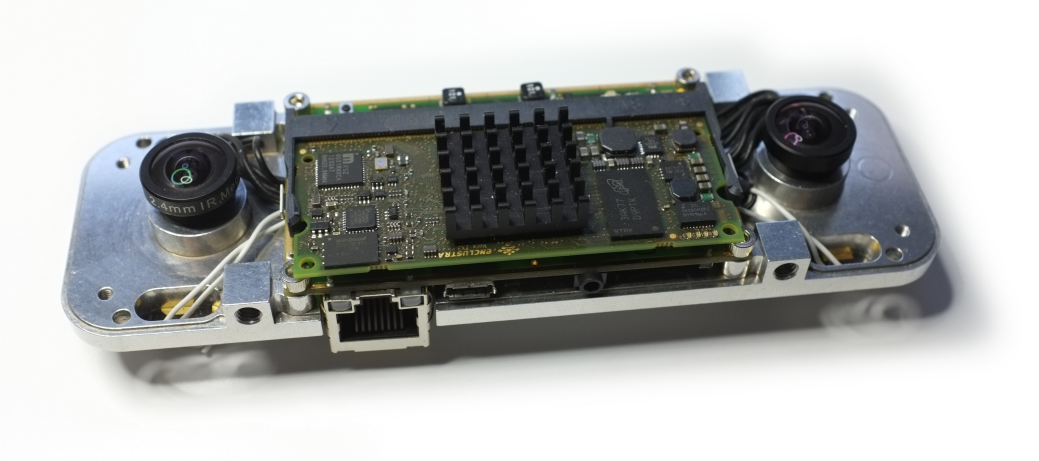


Figure 1.3: An embedded stereo camera with onboard FPGA and ARM CPU was developed in our research group. It provides images and depth maps at 30 frames per second.

1.3 Contributions

The key contributions of my thesis are:

- A real-time online calibration method to determine the rotation matrix of cameras mounted on a vehicle. Different to most algorithms known from literature it does not rely on a dedicated calibration pattern.
- A 3D obstacle detection method based on an embedded stereo camera developed by our research group.
- A traffic light detection system using two separate monocular cameras and a digital map with annotated global traffic light positions. It has contributed to successful public demonstrations in Berlin[1] and also to the win of the Dubai World Challenge for Self-Driving Transport in 2019[2].
- An application of my obstacle detection and traffic-light detection to the task of traffic light mapping.
- A method to estimate the linear and angular velocity of the ego-vehicle using the stereo camera's depth measurements and optical flow.

[1] `https://autonomos.inf.fu-berlin.de/media/videos/`
[2] `https://sdcongress.com/finalists`

1.4 Outline of the Thesis

The thesis is structured as following. Chapter 2 will provide a concise overview of the start-of-the-art methods and algorithms relevant to my contributions. Chapters chapters 3 to 6 contain the mathematical concepts, algorithmic details and experimental results for the aforementioned contributions. Finally, Chapter 7 will conclude my thesis with a summary. The Appendix contains pseudo-code for various algorithms.

Chapter 2

State Of The Art

This chapter summarizes the state of the art related to my contributions presented in chapters 3 to 6. In particular, this section will cover:

1. Online extrinsic calibration for monocular cameras on ground vehicles.
2. Obstacle detection for stereoscopic cameras with focus on occupancy grids and disparity map segmentation.
3. Traffic light detection for monocular and stereoscopic cameras.
4. Ego-motion estimation for forward-facing stereoscopic cameras.

2.1 Extrinsic Camera Calibration Using Optical Flow

Extrinsic camera calibration estimates the camera pose with respect to a reference coordinate frame. Chapter 3 presents an algorithm which calibrates the extrinsic parameters while the car is driving. Under these conditions

- calibration must use what is visible, i.e. there is no calibration target like a chessboard pattern,
- the camera's movement is bound to the car, and
- visible objects may move independently of the camera.

State-of-the-art methods derive perspective cues from edges forming straight parallel lines or relate the vehicle's ego-motion to the observed optical flow.

Perspective Cues from Parallel Lines

Urban scenes often look like so-called Manhattan worlds in which buildings, walls, roads, and other structures are aligned to three orthogonal directions. The projection of lines along these directions converges in vanishing points from which the original 3D directions can be inferred.
For example, Cipolla et al.[8] proposed a semi-automatic method to recover vanishing points from several images showing a building from different angles. Combining the computed projection matrices, they were able to compute all camera poses and generate a full 3D reconstruction of the building. Zhang et al.[9] report the full calibration of a static traffic surveillance camera. The proposed method exploits typical movement patterns of vehicles and pedestrians. Usually, they follow a network of parallel or perpendicular roads and sidewalks. A statistical analysis of the movements reveals two principal directions, while the third (up) direction can be estimated from the principal component of walking pedestrians. The authors reported results comparable to classical methods based on correspondences of interest points.
Front-facing cameras on vehicles can observe lane markings to recover the principal direction of a straight road. Hold et al.[10] propose an online calibration method which recovers pitch and roll angles, and the camera's height. Their algorithm measures the frequency of dashed line markings and combines it with the velocity provided by the car's odometry. The authors report calibration results with standard deviations less than $0.07°$ for the rotation angles respectively less then 1.5 cm for the camera's height. De Paula et al.[11] calibrate the camera's orientation and the height from detected lane boundaries and the vehicle speed. The rotation is reconstructed from vanishing points of parallel lane markings. The authors report errors of less than $0.5°$ for the camera angles, and an average height error of 10%.

Perspective Cues from Motion Flow

The movement of the 3D world relative to the camera is known as motion field and its projection in the image as optical flow. The two vector fields are linked by the equations of central projection. If the observed scene is static the optical flow is caused only by the car's movement and it can be used to recover the extrinsic parameters. Practical problems arise because:

1. The vehicle does not move exactly on a plane.
2. Independently moving objects also cause optical flow.
3. Optical flow is error prone due to the aperture problem.

Additional sensors may increase robustness of optical flow algorithms. For example, Hwangbo et al.[12] improved the tracking of KLT (Kanade-Lucas-Tomasi) feature points with a gyroscope. Fast rotations and translations, violated the KLT assumption of slowly changing optical flow. Predicting the flow with the gyroscope significantly increased the tracking robustness as the tracking rate over 50 frames increased from 33% to 50%.
Many calibration methods assume a flat world and estimate a homography between the ground plane and the image. For example, Miksch et al.[13] developed a method which uses epipolar constraints, known vehicle speed and yaw rate, and the assumption of a fixed camera height, to derive a formulation with only one homography parameter left to be optimized. The authors report a calibration error of approximately 0.35° with respect to the ground truth, and estimate an additional error of 1° when the assumed camera height is off by 5%.
Ruland et al.[14] propose a method to calibrate the position of a fish-eye lens camera mounted on a vehicle. Given the height of the camera and a known circular ego-motion, the method estimates the two remaining camera position coordinates. The authors reported an average birds-eye-view back-projection error of 1.2 mm given the correct height and 53 mm using a wrong camera height off by 0.7%.
Tapia et al.[15] proposed a method to compute the camera orientation from optical-flow-induced vanishing points and estimate the camera height by relating the vehicle speed to the projected velocity of points on the road.
Heng et al.[16] proposed a framework to calibrate multiple non-overlapping fish-eye cameras on a vehicle only by odometry and image data. It is an offline bundle adjustment method which uses all available data at once, and optimizes them jointly by finding a parameter set which maximizes the likelihood of the data. Such methods are computationally expensive but tend to yield more accurate results. The authors tested their method with four surround-view fish-eye cameras on a vehicle and report an average reprojection error of 0.5 pixel for over 450000 points.

Calibration with Stereo Depth Measurements

Stereo cameras capture both images synchronously and therefore disparity measurements do not suffer from independently moving objects. Spangenberg et al.[17] calibrated roll and pitch angles of a stereo camera by sampling the space of rotations and using the stereo matching costs as optimization criterion. Rehder et al.[18] reported a full intrinsic and extrinsic calibration up to scale using stereo matching, odometry and bundle adjustment.

This class of algorithms requires to perform stereo matching with varying parameters in every iteration of the optimization process. Since the embedded stereo camera was used as a black-box system such methods were not pursued in this thesis.

2.2 Stereoscopic Obstacle Detection with Occupancy Maps

Occupancy grids are a well-established method for the accumulation and clustering of 3D point clouds. The grid approach is easy and robust because it does not require the detection of object boundaries. The depth measurements of stereo cameras have an uncertainty which increases with the distance. Perrollaz et al.[19] modeled sensor noise with a Gaussian distribution, which computes for each 3D point a weighted update distributed over several occupancy grid cells.
An occupancy grid is a projection of 3D points onto a grid of cells on a reference plane. Heights of potential obstacles are inferred from the accumulated height of the grid cells with respect to the reference plane. When the flat-world assumption is inadequate, more complex models of the road surface are necessary. Wedel et al.[20] approximated the height profile of a road with B-Spline, while Oniga et al.[21] employed a quadratic surface.
Schauwecker et al.[22] avoided the notion of a reference plane and proposed a volumetric method, which creates Voxels from the measurements. They also considered visibility aspects and updated a volume's occupancy probability according to a Gaussian distribution model.
Classical algorithms for LIDAR sensors work on unstructured 3D point clouds and update the cells for each 3D point independently. Disparity images are structured because 3D proximity implies proximity in image coordinates. Therefore, it is often beneficial to perform a segmentation on the disparity image and then update the cells for each segment. This way, correlation of neighboring points can be used to filter out noise and errors introduced by the stereo matching algorithm.
The disparity segmentation is especially simple, if the stereo camera is mounted parallel to the road surface. Then, objects whose side face is orthogonal to the road will have constant disparity. The u-v-disparity method was the first to analyze disparity maps in this way. It detects the ground plane and vertical obstacles by computing histograms of the disparity distributions along both image axes. Li et al. [23] improved the u-v-approach by tracking feature points across consecutive frames to segment independently moving objects,

yielding velocity information for occupied grid cells. Marín-Plaza et al. [24] combined u-v-disparity with ray-tracing to compute occlusion probabilities for every grid cell. Yu et al. [25] fuse u-disparity and v-disparity space using the Dempster-Shafer theory in order to model sensor uncertainty. Cordts et al.[26] presented the so-called stixel world, a segmentation which finds vertical stripes of (nearly) constant disparities. The stixels define the border of drivable space and single outliers are smoothed out by a dynamic programming algorithm. Harms et al. [27] use the vertical disparity gradient and the estimated angle deviation from the ground plane to construct a probabilistic occupancy grid. A ray search is used to detect occupied cells according to a height threshold and curb cells according to a normal vector deviation threshold. Badino et al.[28] track the velocity of disparities and generate stochastic occupancy grid. Dynamic programming on the a polar occupancy grid yields the corridor of drivable free space.

2.3 Traffic Light Detection

Traffic lights must be detected as early as possible to give the path planning module sufficient time to react. Although traffic lights are designed to be easily visible through bright monochromatic light, early detection remains challenging because the projected size of a traffic light in the image is inversely proportional to its distance. Small bright and colored image patches can easily be mistaken as distant traffic lights. Ambiguities can only be resolved by context, e.g. a traffic light can only appear at an intersection and at certain positions relative to the road.
Algorithms differ by the set of features and the order in which different features are evaluated. Color is a typical first choice because the traffic light state is encoded by distinct monochromatic colors. However, in environments with many other bright light sources or with camera images suffering from over-saturation it may be beneficial to explore shape features first, e.g. circular symmetry. The following sections will give a brief overview of methods focusing on color, followed by a selection of shape-focused algorithms. The section will be concluded by a review of perspective constraints and machine-learned features.

Color Focused Algorithms

This class of algorithms searches for pixels of certain color and brightness and groups them to light blob clusters. These are by far the most distinct features, because traffic lights emit bright monochromatic light and in many

countries there are precise specifications on the spectrum and the luminance. Cameras measure a mixture of the light emanated from the traffic signal and the ambient light from the sun and other light sources. The former is affected by distance and viewing angle, while the latter depends on daytime and weather. As a result, the monochromatic light can appear in different hues and intensities because the ambient light mixes with the colored light and makes it appear less saturated. Therefore many published methods start with a transformation to a suitable color space.
Diaz et al.[29] determine colors using normalized RGB color space and different fuzzy-logic membership functions derived from data sets. They propose to use two different set of rules for day-time and night-time and perform clustering with a simple sequential scan. Omachi et al.[30] classify colors by applying fixed interval thresholds on RGB and normalized RGB values. The clustering of pixels is achieved by a variant of the Hough transform. Tae et al.[31] threshold colors in the HSI color space and find connected components by means of a blob labeling algorithm. Cai et al.[32] convert the image to the YCbCr color space, apply a threshold filter, and use morphological filters to find clusters. The authors propose to detect different arrows on those clusters by applying a Gabor wavelet transform followed by an independent component analysis. Sooksatra et al.[33] compute an intermediate image from the CIELab converted color image on which they search for circular blobs with the radial symmetry transform. Hosseinyalamdary et al.[34] propose a Bayesian framework to model color classes in a probabilistic fashion using mixture of Gaussians and Chi distributions. Model parameters are learned from sample images of traffic lights. Siogkas et al.[35] apply the fast radial symmetry detector to remove all non-circular elements from color-thresholded images. Omachi et al.[36] employ a modified Hough transform to find circular regions in color edge images.

Shape Focused Algorithms

Looking for colors first can be problematic when images are over-saturated and traffic lights appear partially white. Therefore, some methods start out by searching for bright circular blobs and rectangular dark boxes. Later on, colors are only evaluated within such areas of interest.
Cai et al.[32] scan for black pixels and then compute connected components and filter them by their height-to-width ratio in order to find potential traffic light boxes. De Charette et al.[37] detect bright spots in the grayscale image by means of morphological operations. Afterwards an adaptive template matcher evaluates potential box regions around the spots. They propose a

flexible XML format to define the templates, which can contain a variety of properties to be tested, e.g. width/heigh ratio, relative position of lights, HSV color space thresholds. Jie et al.[38] find rectangular areas and circle shapes on gray-scale images. Afterwards, they evaluate the colors of pixels based on pre-defined thresholds in the RGB and the HSI color space.

Perspective Constraints

The size of traffic lights and their placement at intersections can be translated to perspective constraints, i.e. they can only appear at certain regions of the image and with a certain size. For example, Diaz et al.[39] compute for each image row a lookup-table for the expected size of a traffic light at a certain height. Lights blob that are too small or too large will be rejected.
If the exact global positions of traffic lights and the vehicle's location are known, the 3D position can be projected into the image and only a small region of interest must be evaluated. Its size depends on the accuracy of the map and of the vehicle's localization with respect to the map [40, 41].
Stereo cameras can directly measure the height and size of traffic lights. Since their 3D width and height are known beforehand, false positives can easily be rejected[42]. Fregin et al.[43] report three ways to improve traffic light detection through a stereoscopic setup. They test if the area of the traffic light has a homogeneous distance, if the 3D position of a traffic light is within reasonable expectations, and if the 3D structure matches the region of interest in the image. If the third condition is not met, they have found it beneficial to adapt the region according to the 3D object boundaries.

Machine-learned Features

Some researchers have used machine learning to find effective combination of features. Kim et al.[44] proposed to train an SVM classifier on a set of primitive features, like color, size, brightness and geometric moment features. Shi et al.[45] proposed an adaptive background suppression filter to discover image regions of interest. Then a cascaded linear SVM trained on RGB and HOG features, detects circular and arrow traffic lights.
In recent years, deep convolutional neural networks (CNNs) have been very successful with image recognition tasks. The strength of CNNs is that they learn relevant image features directly from the data. John et al.[46] propose to generate regions of interest using GPS localization, and learn from given detections a salience map, i.e. a probability distribution for traffic lights. After training only the saliency map is used to generate the regions of interest. Behrendt et al.[47] modified a YOLO network to tune it specifically for the

detection of traffic lights. For example, they used a finer grid to detect smaller objects and created a separate specialized network for the classification of traffic light states. Weber et al.[48] designed and trained a fully convolutional neural network which is able to detect traffic lights from single images without using any priors. The network generates multi-class probability maps which are grouped to determine the most likely traffic light type and state.

2.4 Ego-Motion Estimation with Stereo Cameras

Visual odometry estimates the vehicle motion from a sequence of image observations. The main challenge of monocular ego-motion estimation is the presence of independent movement, i.e. other vehicles or pedestrians. Stereo cameras capture two images simultaneously which allows easy separation of the ego-motion from the general motion field.
Nister et al, [49] extract Harris corners from stereo images and match them with normalized cross-correlation windows. The triangulated 3D points are tracked over time and poses are generated using a RANSAC scheme and iterative refinement. The authors conducted ground vehicle tests and report relative errors between 1% and 2% for the measured distance. Agrawal et al.[50] proposed a localization method for a stereo camera combined with a low-cost GPS receiver. The authors generate motion hypotheses from triplets of tracked points and evaluate them with RANSAC. A transformation is considered as an inlier, if the disparity space homography yields a low average reprojection error on the second view. Experimental results for the raw visual odometry indicate relative distance errors from 2% to 5%. Howard et al.[51] propose an algorithm which estimates the ego-motion between consecutive frames of stereo images and report a relative distance error of 0.25% on a 400 m long path. Talukder et al.[52] present a method which combines dense stereo disparity maps with dense optical flow. Using the equations which link the general 3D motion to optical flow, they predict ego-motion induced optical flow whose difference to the observed optical flow field serves to segment independently moving objects. The results for visual odometry show a 6.5% relative displacement error. Badino et al.[53] constructed a 3D motion field from disparity maps by tracking 3D points over time. The frame-by-frame motion estimates are registered with previous frames to improve the robustness and accuracy. The presented test result show a relative displacement error of 4.5% on 700 m traveled distance.

Chapter 3

Extrinsic Camera Calibration Using Optical Flow

This chapter presents an automatic online calibration of a camera's extrinsic rotation matrix from images captured while driving on streets. The algorithm uses optical flow to estimate the epipolar points induced by the car's movement and deduces the camera's rotation angles from the distribution of epipolar points.
This chapter will start with a short recall of the basics of projective geometry and central projection. Then follows an introduction to extrinsic calibration from parallel lines and their vanishing points from single images. Finally, I will present my algorithm for extrinsic calibration from optical flow followed by experimental results and their discussion.

3.1 Basics of Projective Geometry

In the following sections matrices will be denoted in bold capital letters. e.g. $\mathbf{A}$, vectors in bold letters, e.g. $\mathbf{x}$. If non-homogeneous and homogeneous vectors appear together homogeneous vectors will be written in bold letters with a tilde on top, e.g. $\tilde{\mathbf{x}}$.

3.1.1 Homogeneous Coordinates

A homogeneous point is a Cartesian point augmented by a 1. For example in 2D space $\binom{x}{y} \in \mathbb{R}_2$ is equivalent to the projective point $\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \in \mathbb{P}_2$.

Homogeneous points are defined up to scale. A multiplication with a non-zero scalar λ is the same point, i.e. $\tilde{\mathbf{x}} \equiv \lambda\tilde{\mathbf{x}}$. The cartesian representation can be recovered through division by the last component, e.g. $\begin{pmatrix} x \\ y \\ w \end{pmatrix} \in \mathbb{P}_2$ is

equivalent to $\frac{1}{w} \cdot \binom{x}{y} \in \mathbb{R}_2$.
Homogeneous coordinates whose last component is 0 do not have a counterpart in real space. These points are also called *points at infinity* or *vanishing points* because they appear at the horizon where projected parallel lines intersect. We will see later that a vanishing point is the projection of parallel lines' common direction vector.
By virtue of homogeneous coordinates it is possible to define a translation as linear mapping, i.e. a matrix. The translation of a point $\mathbf{x} \in \mathbb{R}_3$ by a vector $\mathbf{t}$ is equivalent to the homogeneous transformation

$$\tilde{\mathbf{x}'} = \begin{pmatrix} 1 & 0 & 0 & t_1 \\ 0 & 1 & 0 & t_2 \\ 0 & 0 & 1 & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \tilde{\mathbf{x}}.$$

Similarly, a rigid transformation of a point $\mathbf{x}$ consisting of a rotation $\mathbf{R}$ followed by a translation given by a vector $\mathbf{t}$ can be written as a homogeneous matrix operation $\tilde{\mathbf{x}'} = \left(\begin{smallmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{smallmatrix}\right) \cdot \tilde{\mathbf{x}}$. A sequence of rigid transformations can be combined by the matrix product and represented as a single matrix. This property has been used in Alg. 4 (see Appendix).

3.1.2 The Duality of Projective Space

Duality in projective geometry refers to the fact that homogeneous coordinates have two interpretations. The homogeneous vector $\begin{pmatrix} x \\ y \\ w \end{pmatrix} \in \mathbb{P}_2$ can be a point $\begin{pmatrix} \frac{x}{w} \\ \frac{y}{w} \end{pmatrix} \in \mathbb{R}_2$ or a line with a normal vector $\binom{x}{y}$ and distance w to the origin.

The dot-product of a point $\mathbf{p}$ and a line $\mathbf{l}$ computes the point-line distance. It is the (signed) euclidean distance if the point and the line are represented in their canonical form, i.e. $\mathbf{p} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$ and $\mathbf{l} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$ where the line's normal is a unit vector with $a^2 + b^2 = 1$. The third component c is the (signed) euclidean distance of the line to the origin $\mathbf{O}$. Its sign is negative if the line's normal vector points away from the origin.
Taking the dot product of $\mathbf{p}$ and $\mathbf{l}$ yields

$$d = \underbrace{x \cdot a + y \cdot b}_{\text{distance } \mathbf{O} \to \mathbf{p}} + \underbrace{1 \cdot c}_{\text{distance } \mathbf{l} \to \mathbf{O}}$$

The point is located on the line if the terms add up to 0, which means both, the line and the point, have the same distance to the origin. This holds true for any point on the line. Thus, the cross-product of two points $\in \mathbb{P}_2$ will yield

the line through these points. Likewise, the cross-product of two lines will yield their intersection point. This is especially useful for the computation of the optical flow's focus of expansion (see Alg. 1 in the Appendix).
The same can be achieved in higher-dimensional space using the general form of the cross-product. For example, in $\mathbb{P}_3$ the dual of a point is a 3D plane. Given three points $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{R}_3$ one can compute the plane through these points as

$$\mathbf{p}_\perp = \det \begin{vmatrix} \mathbf{e_1} & a_x & b_x & c_x \\ \mathbf{e_2} & a_y & b_y & c_y \\ \mathbf{e_3} & a_z & b_z & c_z \\ \mathbf{e_3} & 1 & 1 & 1 \end{vmatrix}$$

where $\mathbf{e_i}$ is a canonical unit vector with 1 at position i.

3.2 The Pinhole Camera Model

The pinhole camera model, depicted in Fig. 3.1, is also known as central projection model. It assumes that all incoming light rays pass through a single point before they hit the image plane. The pinhole camera's coordinate system is centered at the pinhole, hence the name central projection. The image plane is orthogonal to the optical axis (the z-axis) at $z = f$, where f is the focal length of the camera. For now, let us assume $f = 1$. Then, central projection maps a 3D point $\mathbf{p} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \in \mathbb{R}_2$ to a 2D image point $\frac{1}{z} \cdot \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{P}_2$.

This formula is strikingly similar to the conversion between real points and homogeneous points. All points along the line connecting $\mathbf{p}$ and $\mathbf{0}$ are multiples of $\mathbf{p}$ and are projected to the same 2D point on the image plane. Thus, a 3D point can be interpreted as the homogeneous equivalent of its 2D projection.
Real-world cameras use optical lenses to collect and focus light, and often they do not adhere to the pinhole projection model. The most important aberration is radial distortion, where incoming light rays are bent towards the center. The effect is radially symmetric and is often modeled as polynomial over the distance to the optical center.
The distortion breaks the linearity of homogeneous projection. This can be compensated by the process of un-distortion which applies the inverse of the distortion model to rectify image coordinates. Lens distortion of thin lenses is often calibrated with the "plumb bob" model[55]. With this distortion model a point $p = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ projected to $p' = \begin{pmatrix} \frac{x}{z} \\ \frac{y}{z} \\ \frac{z}{z} \end{pmatrix}$ is distorted to:

$$p'' = \begin{pmatrix} x'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 x' y' + p_2(r^2 + 2x'^2) \\ y'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + p_1(r^2 + 2y'^2) + 2p_2 x' y' \end{pmatrix}$$

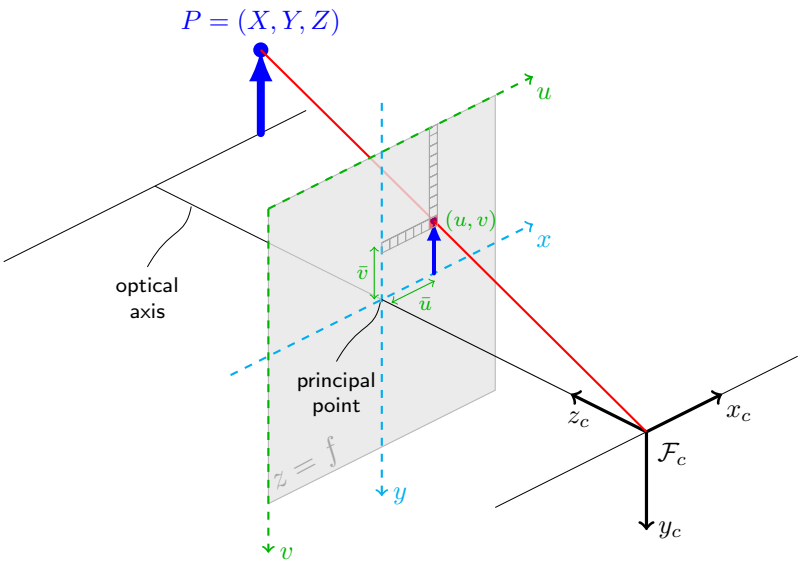


Figure 3.1: The pinhole camera model. (Source: [54])

with distortion parameters k_1, k_2, k_3, p_1, p_2 and $r^2 = x'^2 + y'^2$.

Projected (and distorted) coordinates are then scaled and translated according to the camera matrix, also called intrinsic matrix. It is defined as

$$\begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix},$$

where f is is the focal length in pixels units and (c_x, c_y) is the offset which puts the image origin to the top-left corner. The camera matrix is the homogeneous version of a 2D scaling followed by a 2D translation.
The central projection model assumes that the coordinate system is centered at the focal point and its axes are aligned to the image plane. When 3D points are given in another coordinate system, e.g. that of the car or of another sensor, they must be transformed to the camera's coordinate frame before projection equations can be applied. This transformation is a rigid transformation, a rotation followed by a translation. The rotation matrix consists of orthogonal unit vectors which span a Cartesian coordinate system. A rotation matrix is a redundant representation since a 3D rotation has only 3 degrees of freedom. However, the matrix representation can be easily interpreted by visual inspection, because the columns are the image of

the domain's basis vectors. For example, the rotation matrix

$$R = \begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{pmatrix}$$

describes the coordinate rotation which maps the vector $\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$ (x-axis) to the vector $\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$ (z-axis). The second column maps $\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$ (y-axis) to the vector $\begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix}$ (negative x-axis) and the third column maps $\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$ (z-axis) to $\begin{pmatrix} 0 \\ -1 \\ 0 \end{pmatrix}$ (negative y-axis).
Any set of three orthogonal unit vectors will be a valid rotation matrix. As will be shown in the following section, a vanishing point is the projection of a direction vector and vanishing points from orthogonal directions can be used to construct a rotation matrix.
Finally, the full projection transformation (for distortion-free projection) can be combined into one matrix:

$$P = K_{3x3} \cdot T_{3\times4}$$

where $T_{3\times4}$ contains first three rows of the 4×4 rigid transformation from world to camera coordinates. It maps a homogeneous 3D point $\mathbf{P} = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \in \mathbb{P}_3$ to a homogeneous image point $\mathbf{p} = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \in \mathbb{P}_2$.

3.3 Extrinsic Calibration with a Single Image

The extrinsic parameters define the rigid body transformation between the car's and the camera's coordinate system. The car's coordinate frame has its origin at the ground below the front axle midpoint. The axes are aligned with the car's body: x points forward, y is directed to the left and z goes up. The camera's coordinate frame is centered the camera's focal center, the forward direction is the z axis and x- and y-axis span the image plane.
The calibration method presented in this chapter computes the camera's rotation matrix. Its accurate calibration is crucial for sensor fusion because a attitude errors have a bigger impact on distance measurements. The camera's position can be measured manually with acceptable accuracy of approximately 1 cm which has little impact on the accuracy of distance measurements.
Sets of parallel lines in perpendicular directions appear often in cities and can be directly used to calibrate the rotation matrix from a single image.

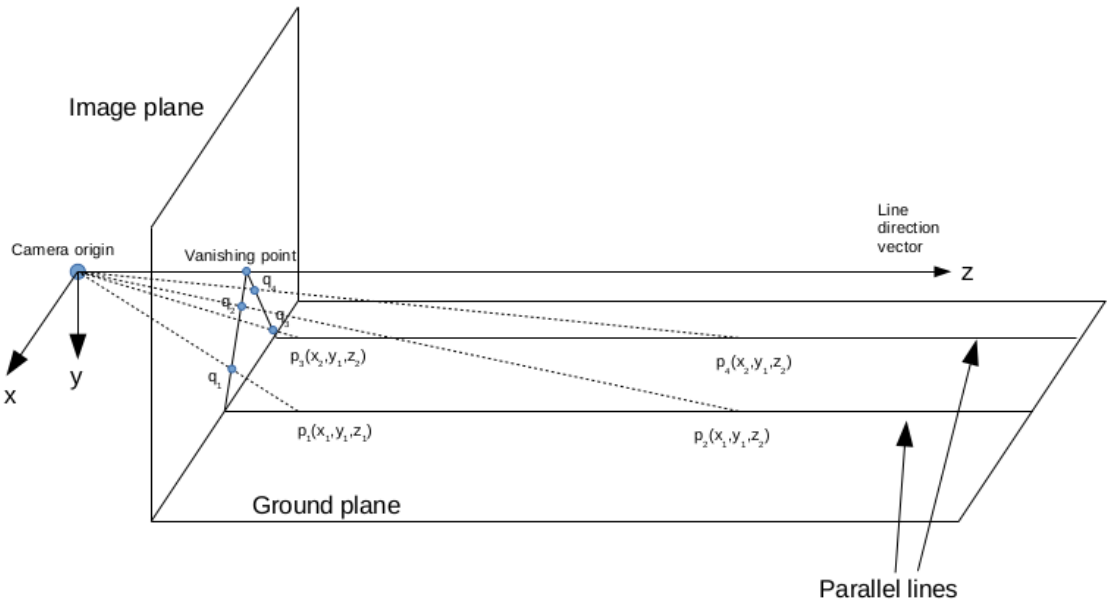


Figure 3.2: The images of two parallel lines intersect at their vanishing point. All parallel lines share a common vanishing point. It is the projection of the direction vector.

Criminisi et al. [56] offer a good introduction into the topic of measuring geometrical quantities from parallel and orthogonal lines and planes. We will now see how this can be applied to the calibration problem.

3.3.1 Vanishing Points and their Geometric Interpretation

Under central projection, two parallel lines appear to converge and intersect in the so-called vanishing point. It can be shown that this point is the projection of the common direction vector of parallel lines.
Let there be two parallel lines $\mathbf{l_1}$ and $\mathbf{l_2}$ as shown in 3.2) with a direction vector $\mathbf{v}$. A point on $\mathbf{l_1}$ can be parameterized by a parameter λ_1 as follows: $\mathbf{l_1} = \tilde{\mathbf{p_1}} + \lambda_1 \cdot \mathbf{v}$. Given a camera matrix

$$\mathbf{K} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

the line $\mathbf{l_1}$ is mapped to

$$\mathbf{K} \cdot \mathbf{l_1} = \begin{pmatrix} f_x \cdot (x_1 + \lambda_1 \cdot (x_2 - x_1)) + c_x \cdot (z_1 + \lambda_1 \cdot (z_2 - z_1)) \\ f_y \cdot (y_1 + \lambda_1 \cdot (y_2 - y_1)) + c_y \cdot (z_1 + \lambda_1 \cdot (z_2 - z_1)) \\ z_1 + \lambda_1 \cdot (z_2 - z_1) \end{pmatrix}$$

$$= \begin{pmatrix} \frac{f_x \cdot x_1 + f_x \cdot \lambda_1 \cdot (x_2 - x_1)}{z_1 + \lambda_1 \cdot (z_2 - z_1)} + c_x \\ \frac{f_y \cdot y_1 + f_y \cdot \lambda_1 \cdot (y_2 - y_1)}{z_1 + \lambda_1 \cdot (z_2 - z_1)} + c_y \\ 1 \end{pmatrix}$$

.

As λ_1 approaches infinity, the terms with λ_1 will dominate. The limit is the homogeneous point

$$\begin{pmatrix} f_x \cdot \frac{(x_2 - x_1)}{z_2 - z_1} + c_x \\ f_y \cdot \frac{(y_2 - y_1)}{z_2 - z_1} + c_y \\ 1 \end{pmatrix}$$

which is the projection of the line's direction vector. The projection of line l_2 will give the same result because with the distance approaching infinity only the direction vector $\mathbf{v}$ matters. Thus, all parallel lines share a common vanishing point. Conversely, we can re-project a vanishing point $\tilde{\mathbf{p}}$ to retrieve the direction vector $\mathbf{v} = K^{-1} \cdot \tilde{\mathbf{p}}$.

3.3.2 Construction of a Rotation Matrix from Vanishing Points

A rotation matrix consists of three orthogonal unit vectors, the three axis directions. Urban scenes often contain straight parallel lines and if we find such lines for three orthogonal directions we can compute their vanishing points $\tilde{\mathbf{p_x}}$, $\tilde{\mathbf{p_y}}$ and $\tilde{\mathbf{p_z}}$, and recover the directions vectors

$$\mathbf{d_x} = K^{-1} \cdot \tilde{\mathbf{p_x}}$$

$$\mathbf{d_y} = K^{-1} \cdot \tilde{\mathbf{p_y}}$$

$$\mathbf{d_z} = K^{-1} \cdot \tilde{\mathbf{p_z}}$$

After normalizing the vectors to unit length the rotation matrix is constructed as

$$R = \begin{bmatrix} \mathbf{d_x}^T \\ \mathbf{d_y}^T \\ \mathbf{d_z}^T \end{bmatrix}.$$

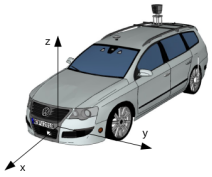


Figure 3.3: MadeInGermany's coordinate system.

Figure 3.4: Single-image calibration from parallel lines. The lane markings yield the car's x-axis (red), the building's roof and windows the y-axis (green), and the lamp posts the z-axis (cyan).

This rotation matrix is orthogonal if the observed pairs of parallel lines were exactly orthogonal to each other. However, even if that was true, the projected lines usually cannot be measured accurately and the resulting vectors will not be orthogonal. Therefore, in a final refinement step these vectors must be rectified to create a valid rotation matrix. A straight-forward solution is the cyclic cross-product

$$z := x \times y$$

$$x := y \times z$$

$$y := z \times x$$

yielding three orthogonal vectors. Since there is no preferred axis, one can compute this in three different ways, starting with each axis and take the average of the rotations [57]. Obviously, the method also works if only lines of two directions are visible because the third direction can be computed by the cross-product. Figure 3.4 shows an example scene suitable for a single-image calibration. Assuming the car is aligned with the road lane markings, all three axis can determined using parallel lines from lane markings, buildings and lamp posts.

The advantage of this method is the minimal requirement of a single image view, its disadvantage the relatively strong assumption that the observed lines are indeed perpendicular and aligned to the vehicle's body axes.

3.4 Extrinsic Calibration from Ego Motion

3.4.1 Vanishing Points from Optical Flow

Although human-made structures contain many parallel and orthogonal lines, they may not always be visible. Therefore, my calibration method is based on the epipolar geometry of consecutive frames. If the camera is moving in a static environment, the optical flow moves along epipolar lines. All epipolar lines intersect in the epipole, the projection of the other frame's camera center. The optical flow vectors will converge in the epipole because all observed points move into the same direction, the inverse of the camera's motion. Thus, re-projection of the epipole will recover the direction vector of the movement. For example, if the car drives straight ahead all optical flow vectors will converge in the epipole which will be the projection of the car's forward direction.
The effect of different camera motions is depicted in Fig. 3.5. The first column shows optical flow patterns for pure axis-aligned translations. The movement along the depth axis yields converging vectors, while orthogonal movement creates parallel vectors. The top row of Fig. 3.5 shows optical flow patterns of pure rotational movements. A rotation around the depth axis causes a circular optical flow field which has no point of convergence. Rotations around the other axes yield, similar to translations in that direction, parallel flow vectors. Rotation around an arbitrary axis will create a combination of these patterns, where divergence increases the more the rotation axis aligns with the depth axis. A camera pointed to the car's forward direction will observe patterns similar to the ones shown in the bottom row.
Given enough epipolar points and a mixture of driving straight and in curves, the set of epipolar points will look like in Fig. 3.6. Since driving forward usually dominates, there will be a dense central cluster. This center is the epipolar point of forward movement and is located at the intersection of two stretched clusters. The large horizontal cluster stems from curve driving, while the sparse vertical cluster is created when the car stops and pitches up and down because the remaining kinetic energy is damped by the shock absorbers.

The epipolar point from forward movement and the horizon vector provide enough information to construct the rotation matrix. We pick another arbitrary epipolar point p_a from the horizon and compute the vectors $\mathbf{d_a} = K^{-1}\tilde{\mathbf{p}_a}$ and $\mathbf{d_x} = K^{-1}\tilde{\mathbf{p}_x}$. Since we know that both vectors are parallel to the x-y-plane, we can construct the z-direction by taking the cross-product

$$\mathbf{d_z} := \mathbf{d_x} \times \mathbf{d_a}$$

and then construct the y-direction with

$$\mathbf{d_y} = \mathbf{d_z} \times \mathbf{d_x}$$

After normalization the rotation matrix R is given by

$$R = \begin{bmatrix} \mathbf{d_x}^T \\ \mathbf{d_y}^T \\ \mathbf{d_z}^T \end{bmatrix}$$

.

3.4.2 Implementation Details

The implementation performs several steps:

1. Optical flow computation.
2. Computation of the epipolar point from optical flow.
3. Estimation of the horizon and the central point from all collected epipolar points.
4. Construction of the rotation matrix.

Optical Flow Computation

Optical flow is the apparent movement of image points in consecutive images. It is deduced from intensity change patterns under the assumption of constant brightness. The optical flow's relation to the projection equations is discussed in detail in Chapter 6. For now, it is sufficient to know that an optical flow algorithm returns this vector field of apparent image point movement.
In my algorithm the optical flow is computed using the pyramidal Lukas-Kanade implementation (`calcOpticalFlowPyrLK()`) provided by the C++ computer vision library OpenCV[1]. As input I have used interest points provided by OpenCV's function `goodFeaturesToTrack()`. These interest points are located at locations of high contrast and therefore suitable for optical flow computation. The extraction of feature points is steered by an image mask which enforces an equal distribution of feature points. If flow vectors are close to each other they tend to be almost parallel. Finding the intersection point of flow vectors is more stable if the vectors are scattered throughout

[1] `http://www.opencv.org`

the image.
The output of the optical flow method are two lists of matched interest points. After applying un-distortion, these points are passed to the next step where we want to find the best possible intersection point given the lines defined by the matched points.

Epipolar Point Computation

The optical flow's point of convergence, the epipolar point, is robustly estimated using a RANSAC scheme as outlined in Alg. 2 (see Appendix). RANSAC stands for Random Sample Concensus and is capable of finding patterns in very noisy data. The idea is to randomly sample a minimal number of data points to construct the pattern, and check how many of the other data points support this hypothesis. The pattern with the largest support is selected.
In our case the minimal set are two flow vectors whose lines are intersected. The algorithm performs $N = 2000$ RANSAC iterations. It randomly selects two flow vectors which are non-parallel, i.e. at least 0.5° apart. The intersection point $\mathbf{x}$ is checked against every flow vector and accepted if the support is larger than a threshold (e.g. $n = 20$). A vector $\mathbf{v}$ is a supporting inlier if the angle between $\mathbf{v}$ and the line through $\mathbf{x}$ and the base point of $\mathbf{v}$ diverges by less than 0.5°. In the end, the intersection point with the largest group of supporters is used to fit the final solution with a least-squares method.
Ideally, every couple of consecutive image frames yields an epipolar point. However, the computation may fail if none of the candidates has the required minimal support. This can happen when there are not enough flow vectors (e.g. the car stands still) or when the flow field is divergent, i.e. there is significant camera roll.

Extraction of the Rotation from the Epipolar Point Cloud

The vehicle's driving maneuvers move the epipolar point according to the optical flow field and create certain patterns. During straight driving the epipolar points will focus on a dense cluster. When the car takes a curve the epipolar points will scatter along the horizon line. Large rotational flow components will make the flow vectors increasingly parallel and create intersection points with large coordinates. Likewise, epipolar points will scatter along the up and down direction if the car abruptly pitches, e.g. when the car stops or drives over a speed bump. These types of flow patterns create a cross-shaped distribution as shown in Fig. 3.6. The next stage of the calibration algorithm fits a cross pattern into the point set. It is derived from

the MATRIX-algorithm developed by von Hundelshausen et. al [58]. The MATRIX-algorithm was originally proposed to solve the localization problem for soccer-playing robots where a set of points had to be matched to a model of the field lines. In our case the model is simply the axis cross of the coordinate system. If we find the transformation which aligns the point cloud to the coordinate system axes, then the inverse transformation will define the position and orientation of the point cloud.
The force at a point is defined as convex combination of the forces to each line segment. In our case we have two line segments, the two axes, and therefore two force components. The force vector is defined as

$$\mathbf{f}(p) = w_1 \cdot \begin{pmatrix} -p_x \\ 0 \end{pmatrix} + w_2 \cdot \begin{pmatrix} 0 \\ -p_y \end{pmatrix}$$

with weights

$$w_1 = \frac{\exp(\frac{-|x|}{k})}{\exp(\frac{-|x|}{k}) + \exp(\frac{-|y|}{k})} w_2 = \frac{\exp(\frac{-|y|}{k})}{\exp(\frac{-|y|}{k})}$$

The weights w_1 and w_2 sum up to 1 and ensure a smooth vector field. They are both 0.5 if a point is equally far away from both axes. Otherwise the weights favor the component pointing to the closer axis. If one of the distances is much larger than the other the weights will function like a minimum operator. The parameter k shapes the sharpness of the transition between the two extrema as can be seen in Fig. 3.7b. My solution uses a force field with $k = 10$ as depicted in Fig. 3.7a which creates a sharp transition to the minimum distance force component. The average force of all points is used to define the translation in an iteration. The rotation angle is proportional to the angular momentum of the force field which is defined as

$$m(p) = (p_x - c_x) \cdot f_y - (p_y - c_c) \cdot f_x,$$

where the point $\mathbf{c}$ is the center of the point cloud and also the center of rotation.
The whole procedure of fitting the cross shape is laid out in Alg. 4 (see Appendix). The resulting center point and horizon vector are then used to construct three orthogonal direction vectors which define the extrinsic camera rotation (see Alg. 5 in the Appendix).

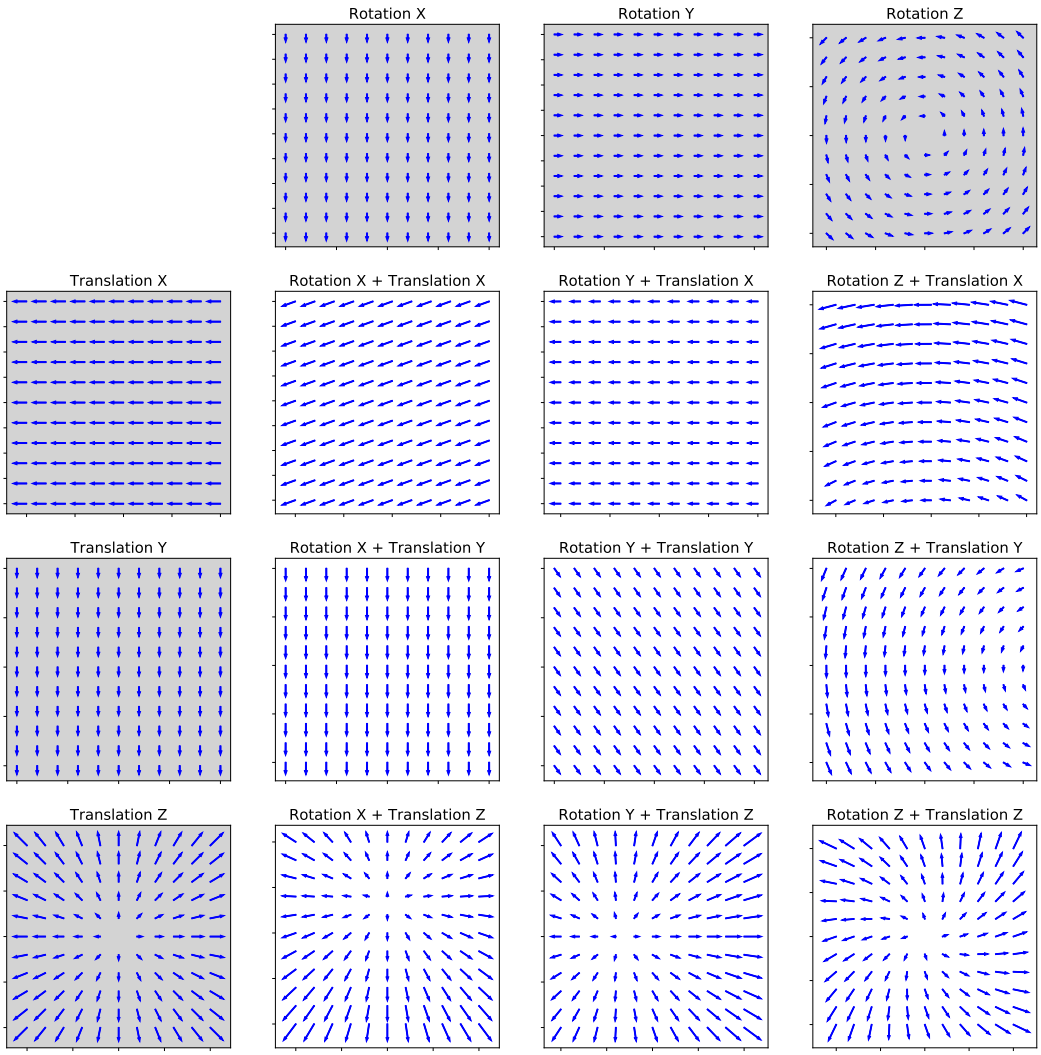


Figure 3.5: The plots show different types of optical flow patterns. Pure translations and rotations are plotted with gray background. The 3x3 matrix with white background shows the combinations of pure flow (first row) and pure translation (first column).

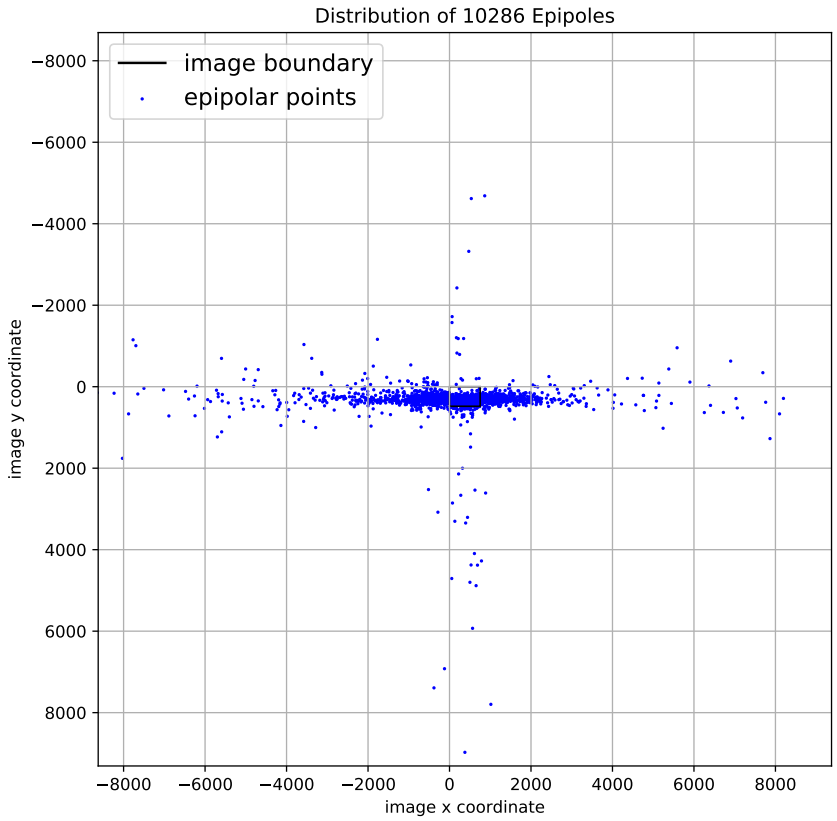
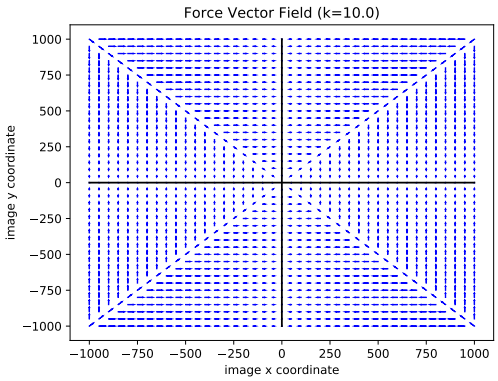
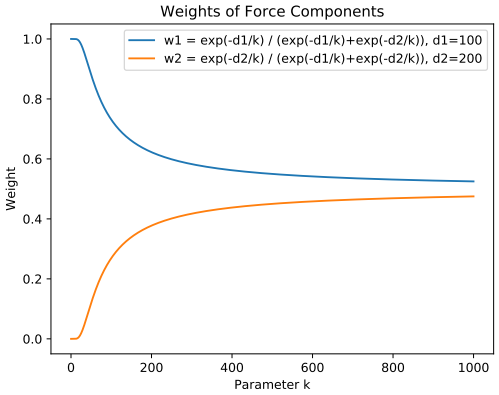


Figure 3.6: A typical cross-shaped distribution of epipolar points.



(a) Force vectors at the diagonals point to the origin. Off-diagonal vectors are deflected to the closer axis. The parameter $k = 10$ generates a force field which strongly favors the closer axis.



(b) The weights w_1 and w_2 add up to 1. Small values of k assign the higher weight to the nearest axis. Higher values of k make the weights converge to parity.

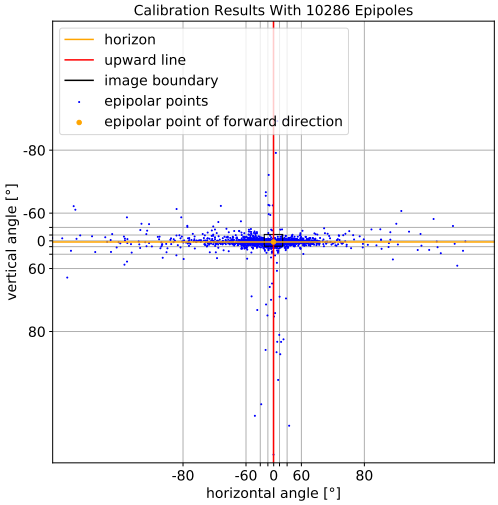
3.5 Experimental Evaluation

The presented calibration method has been tested on four different camera setups. All video sequences are a mix of straight driving with stop and go at intersections and some turns. The plots show the distribution of all epipolar points computed from the optical flow. The result of the calibration process is shown by the horizon line, the up direction line and the central epipolar point drawn on top of the point cloud. A second plot is zoomed in at the central cluster.
Besides, there the same plots showing the results computed on a quarter of the data. The subsets were split sequentially and the result for the subsets show the degree of variation given less and differently distributed data.
The evaluation is completed by an analysis of the convergence properties. The epipolar points are fed one-by-one to the calibration algorithm and the intermediate results are compared to the final calibration result computed from all epipolar points.

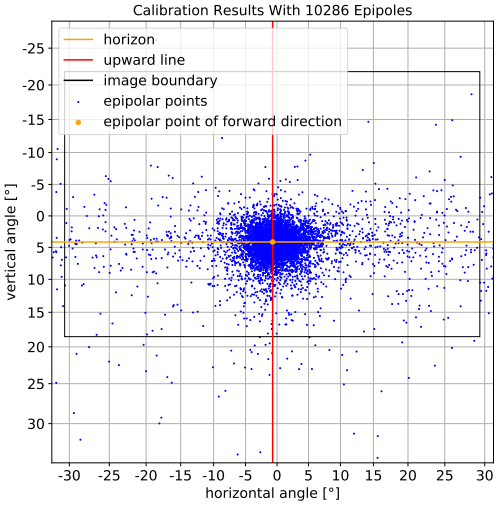
(a) Frontal view camera.

Calibration Results for a Frontal View Camera

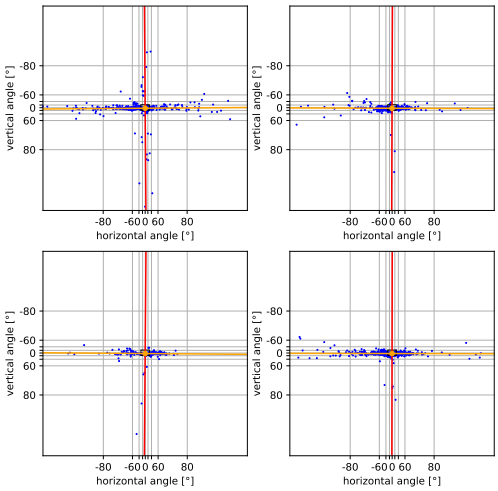
For this experiment the stereo camera was mounted on the roof tilted up by approximately 5°. Fig. 3.9 shows the epipolar points and the resulting calibration. The numerical results in Tab. 3.1 show high accuracy and low variance, except for the yaw angle, which suffers from a low number of vertical epipolar points. This also reflects in a slow convergence of the yaw angle.



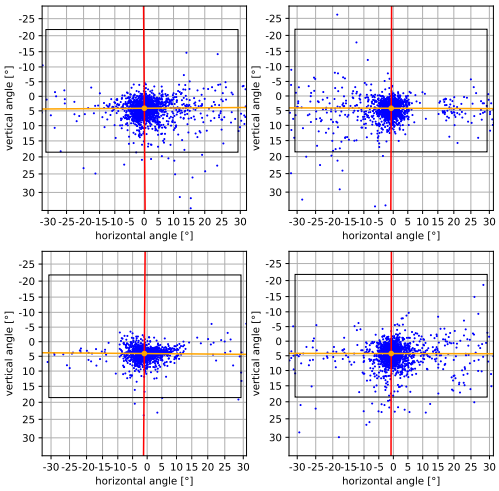
(a) When the car drives curves the epipoles scatter along the horizontal direction. When the car stops epipoles scatter along the vertical direction.



(b) The central cluster captures the forward direction.



(c) The data was split into 4 equal sequential parts. Depending on how the vehicle drove, some arms of the cross are less populated.



(d) The central cluster of the subsets.

Figure 3.9: The calibration algorithms fits a cross into the point set. The camera's roll angle is determined by the angle of the cross, while the location of the center determines pitch and yaw angles. For numerical details see Table 3.1.

	Full Data Set	Subset 1	Subset 2	Subset 3	Subset 4	RMSE
Roll	0.03°	0.46°	−0.18°	−0.41°	−0.07°	0.29°
Pitch	4.97°	4.92°	5.03°	4.88°	5.07°	0.07°
Yaw	−0.40°	−1.16°	−0.40°	−0.13°	−0.36°	0.36°

Table 3.1: The root-mean-square error is the deviation of the subset results from the full data set result. The subsets are the first, second, third, and fourth quarter of the data set.

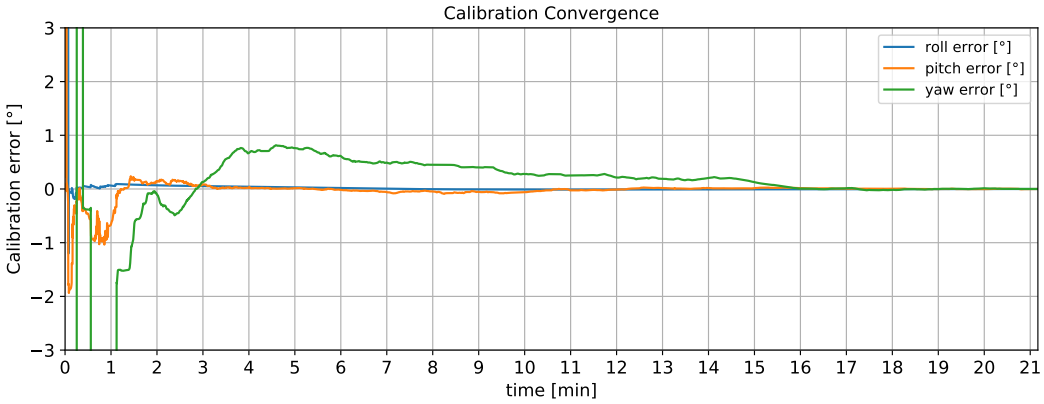


Figure 3.10: The plot shows the calibration error over time. The calibration result on the full data set (N=10000 points) served as ground truth.

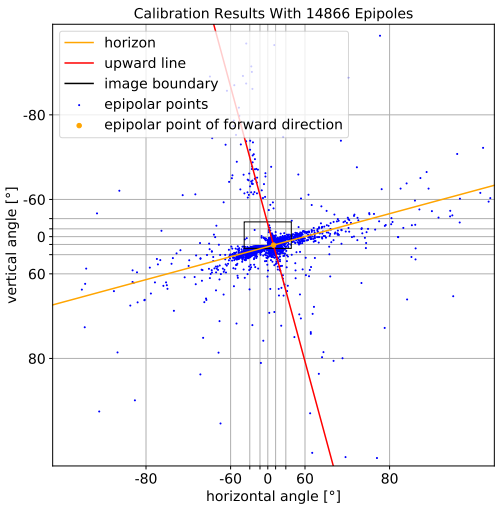
Time	60s	120s	180s	240s	300s	360s
Roll Error	0.06°	0.07°	0.05°	0.04°	0.03°	0.01°
Pitch Error	−0.66°	0.15°	0.08°	0.03°	0.01°	−0.02°
Yaw Error	−40.61°	−0.10°	0.13°	0.67°	0.76°	0.61°

Table 3.2: The calibration converges after 2 minutes. However, the yaw angle variance is relatively high. The type of track (several longs loop) may have induced a data bias.

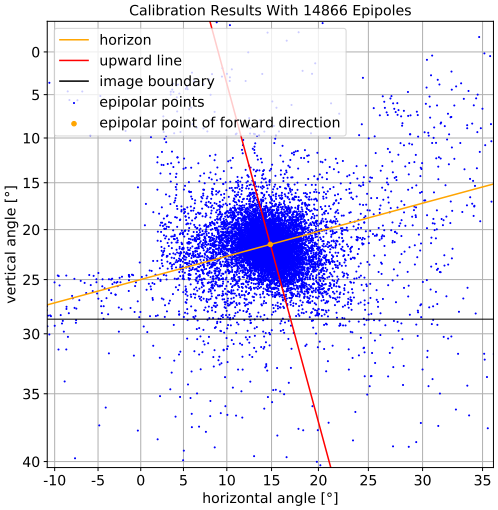
Figure 3.11: Camera with significant roll.

Calibration Results for a Rolled Camera

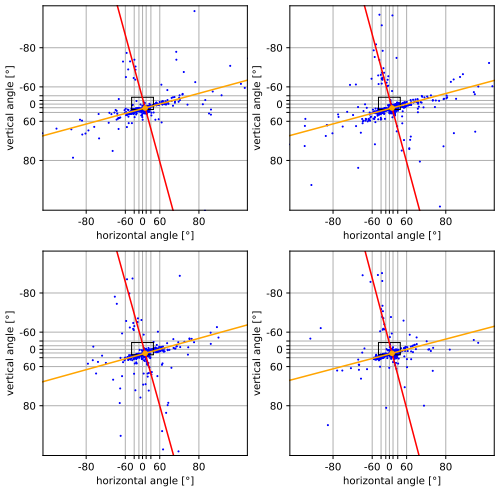
This camera was placed behind the right side of the windshield and used for traffic light detection. As can be seen in Fig. 3.11) it has significant roll and pitch with respect to the car's coordinate frame. Fig. 3.12 visualizes the distribution of the epipolar points. The numerical results in Tab. 3.1 show high accuracy and low variance, except for the roll angle. Since this camera has a significant roll angle, it will rotate around its depth axis, when the car drives a curve. Hence. the flow vectors will not exactly converge which leads to higher variance of calculated horizontal epipolar points.



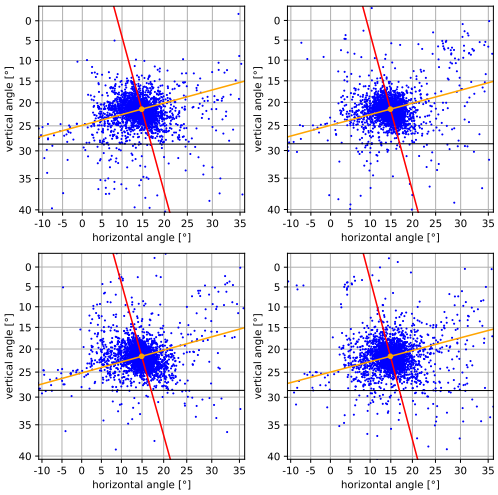
(a) When the car drives curves the epipoles scatter along the horizontal direction. When the car stops epipoles scatter along the vertical direction.



(b) The central cluster captures the forward direction.



(c) The data was split into 4 equal sequential parts. Depending on how the vehicle drove, some arms of the cross are less populated.



(d) The central cluster of the subsets.

Figure 3.12: The calibration algorithms fits a cross into the point set. The camera's roll angle is determined by the angle of the cross, while the location of the center determines pitch and yaw angles. For details see Table 3.3.

	Full Data Set	Subset 1	Subset 2	Subset 3	Subset 4	RMSE
Roll	12.49°	12.55°	12.44°	12.79°	12.19°	0.19°
Pitch	23.90°	23.79°	23.87°	24.00°	23.95°	0.07°
Yaw	−16.92°	−16.82°	−16.93°	−16.96°	−16.97°	0.05°

Table 3.3: The root-mean-square error is the deviation of the subset results from the full data set result. The subsets are the first, second, third, and fourth quarter of the data set.

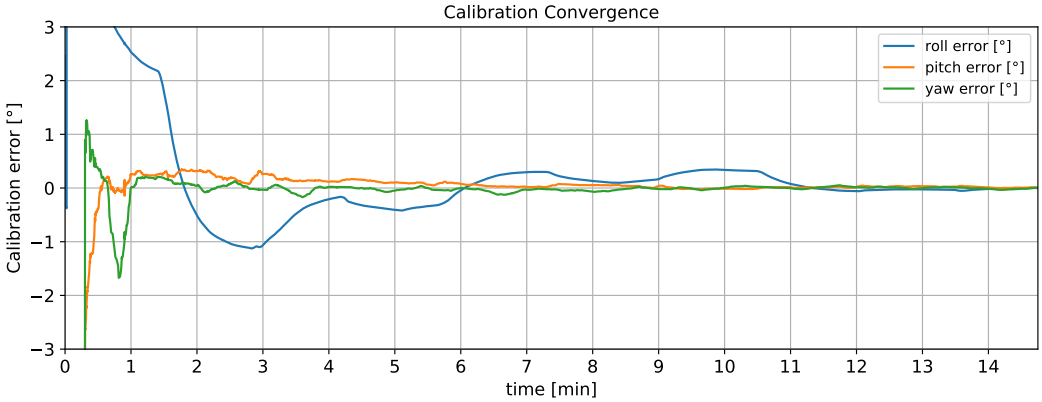


Figure 3.13: The plot shows the calibration error over time. The calibration result on the full data set (N=10000 points) served as ground truth.

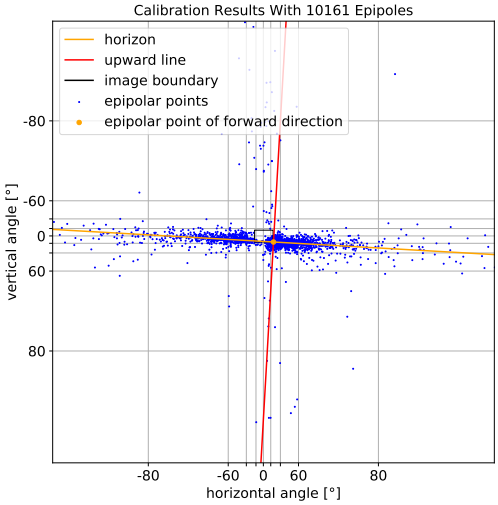
Time	60s	120s	180s	240s	300s	360s
Roll Error	2.53°	−0.51°	−1.07°	−0.21°	−0.41°	−0.04°
Pitch Error	0.25°	0.32°	0.26°	0.12°	0.10°	0.08°
Yaw Error	0.01°	0.04°	−0.03°	0.02°	−0.05°	0.00°

Table 3.4: The calibration converges after 4 minutes. Additional time decreases the variance.

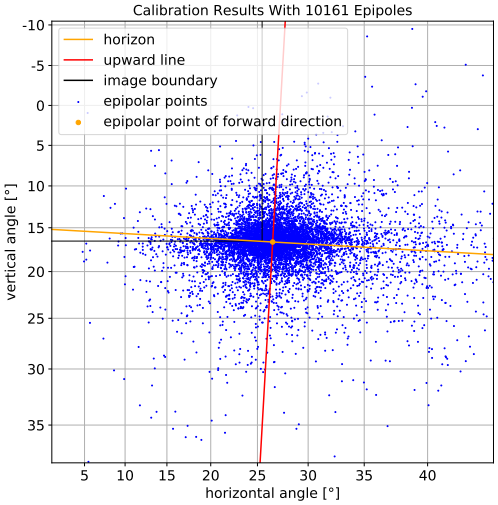
Figure 3.14: Camera rotated far to the top-left.

Calibration Results for Left-looking Traffic Light Camera

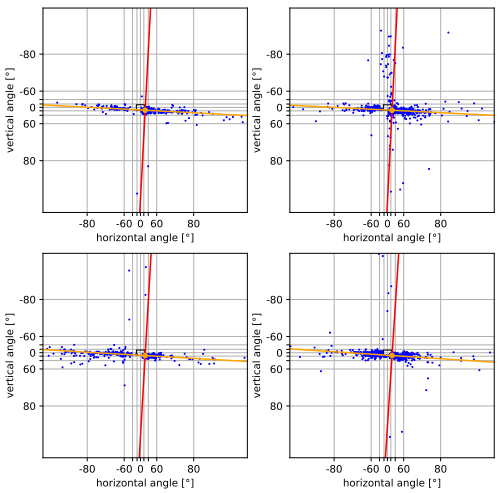
This is one of the two cameras used in the car *MadeInGermany* for traffic light detection. As can be seen in Fig. 3.14, it is pointed far up and left such that nearby traffic lights are visible. The calibration results can be seen in Fig. 3.15. It is challenging because the epipolar point of forward movement is in the bottom right corner of the image. Flow vectors will always move to the top left direction. The small angles between them makes the computation of the intersection points numerically less stable. However, the results shown in Tab. 3.5 are decent with RSME errors of circa. 0.3°.



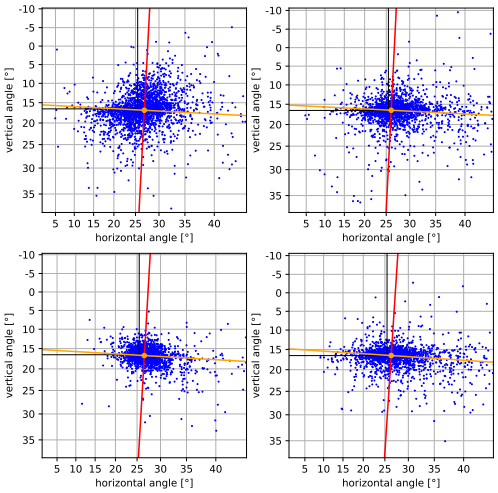
(a) When the car drives curves the epipoles scatter along the horizontal direction. When the car stops epipoles scatter along the vertical direction.



(b) The central cluster captures the forward direction.



(c) The data was split into 4 equal sequential parts. Depending on how the vehicle drove, some arms of the cross are less populated.



(d) The central cluster of the subsets.

Figure 3.15: The calibration algorithms fits a cross into the point set. The camera's roll angle is determined by the angle of the cross, while the location of the center determines pitch and yaw angles. For numerical details see Table 3.5.

	Full Data Set	Subset 1	Subset 2	Subset 3	Subset 4	RMSE
Roll	−3.49°	−3.23°	−3.05°	−3.55°	−4.00°	0.32°
Pitch	16.65°	16.89°	16.50°	16.74°	16.53°	0.14°
Yaw	−25.58°	−25.93°	−25.16°	−25.62°	−25.44°	0.25°

Table 3.5: The root-mean-square error is the deviation of the subset results from the full data set result. The subsets are the first, second, third, and fourth quarter of the data set.

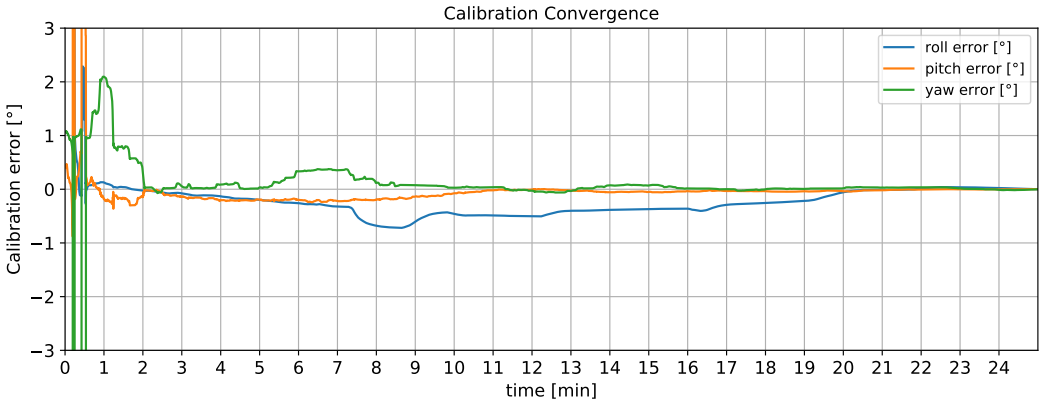


Figure 3.16: The plot shows the calibration error over time. The calibration result on the full data set (N=10000 points) served as ground truth.

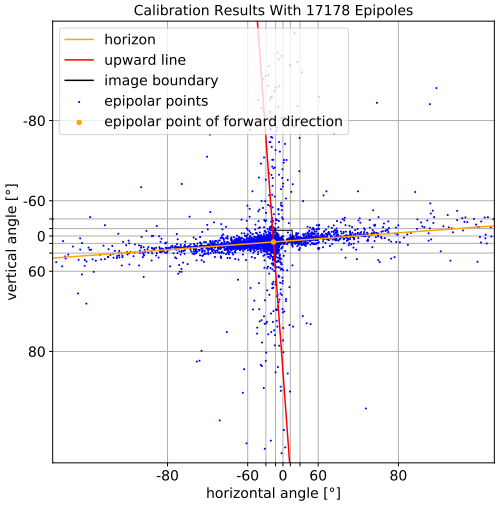
Time	60s	120s	180s	240s	300s	360s
Roll Error	0.13°	−0.03°	−0.07°	−0.13°	−0.20°	−0.26°
Pitch Error	−0.18°	−0.15°	−0.14°	−0.20°	−0.20°	−0.20°
Yaw Error	2.09°	0.28°	0.06°	0.09°	0.03°	0.33°

Table 3.6: The calibration converges after 2 minutes of driving.

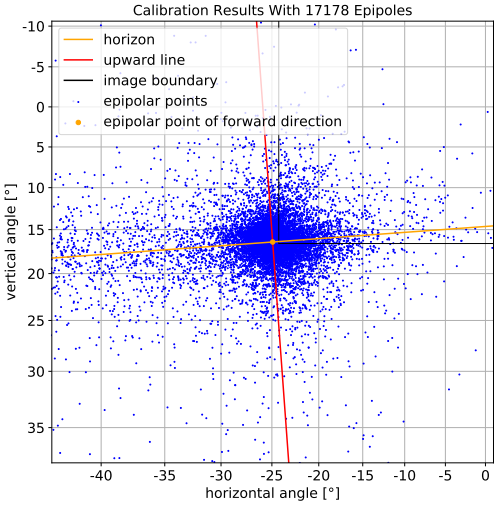
Figure 3.17: Camera rotated far to the top-right.

Calibration Results for Right-looking Traffic Light Camera

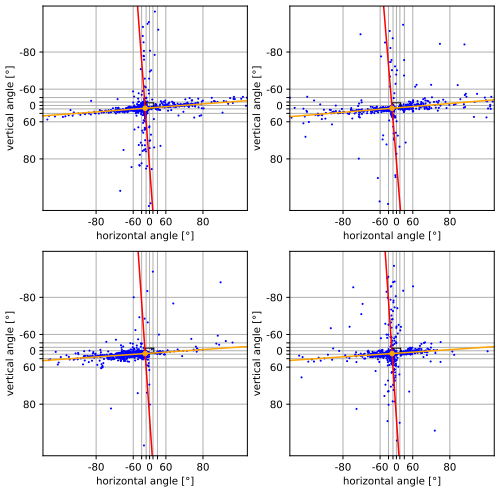
This is one of the two cameras used in the car *MadeInGermany* for traffic light detection. As can be seen in Fig. 3.17), it is pointed far up and right such that nearby traffic lights are visible. The calibration results can be seen in Fig. 3.18. It is challenging because the epipolar point of forward movement is in the bottom left corner of the image. Flow vectors will always move to the top right direction. The small angles between them makes the computation of the intersection points numerically less stable. The results are similar to the other camera.



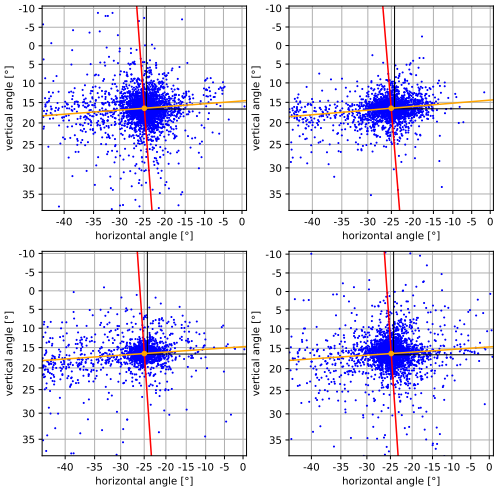
(a) When the car drives curves the epipoles scatter along the horizontal direction. When the car stops epipoles scatter along the vertical direction.



(b) The central cluster captures the forward direction.



(c) The data was split into 4 equal sequential parts. Depending on how the vehicle drove, some arms of the cross are less populated.



(d) The central cluster of the subsets.

Figure 3.18: The calibration algorithms fits a cross into the point set. The camera's roll angle is determined by the angle of the cross, while the location of the center determines pitch and yaw angles. For numerical details see Table 3.7.

	Full Data Set	Subset 1	Subset 2	Subset 3	Subset 4	RMSE
Roll	4.42°	4.52°	4.99°	4.18°	4.08°	0.32°
Pitch	16.44°	16.42°	16.48°	16.49°	16.31°	0.06°
Yaw	24.02°	23.84°	24.13°	24.06°	23.94°	0.10°

Table 3.7: The root-mean-square error is the deviation of the subset results from the full data set result. The subsets are the first, second, third, and fourth quarter of the data set.

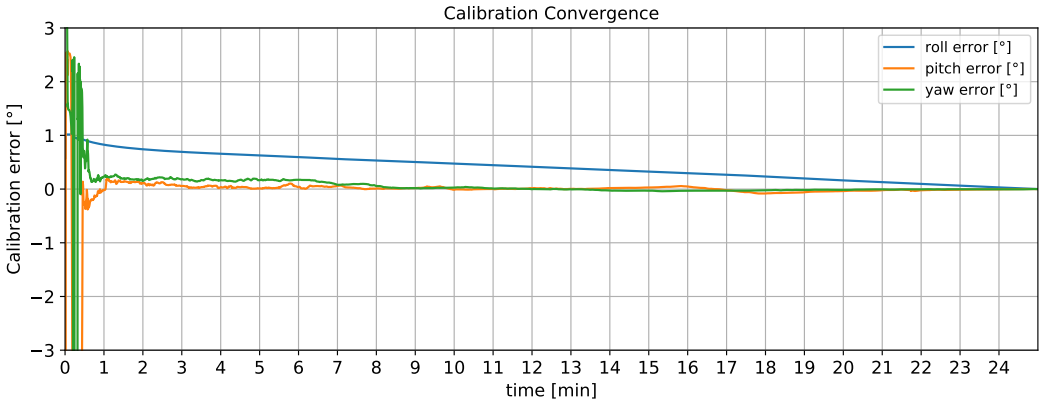


Figure 3.19: The plot shows the calibration error over time. The calibration result on the full data set (N=10000 points) served as ground truth.

Time	60s	120s	180s	240s	300s	360s
Roll Error	0.83°	0.74°	0.69°	0.66°	0.62°	0.60°
Pitch Error	−0.02°	0.11°	0.06°	0.04°	0.01°	0.03°
Yaw Error	0.23°	0.16°	0.18°	0.16°	0.17°	0.17°

Table 3.8: The calibration converges quickly for pitch and yaw, but not for the roll angle.

3.6 Conclusions

This chapter presented an algorithm to calibrate the rotation matrix of a car-mounted camera using only optical flow computed on consecutive images. The flow vectors converge in an epipolar point whose location depends on the current of direction driving. The distribution of epipolar points gathered from a video exhibits a cross-shaped pattern from which the axes of the vehicle's coordinate system and ultimately the rotation matrix is derived. The results show low variance and good convergence properties. Convergence time could benefit from more purposeful driving.

Chapter 4

Obstacle Detection with the AutoNOMOS Stereo Camera

This chapter presents my obstacle detection system for a stereo camera that has been developed by our research group. The embedded stereo camera is a hardware board with image sensors, optics, an ARM processor, and a FPGA for massive parallel processing. It captures synchronized images at a resolution of 752×480 with a frame rate of 30 Hz, rectifies the images and computes a dense disparity map.
Stereo vision 3D point clouds differ from LIDAR point clouds with respect to noise, accuracy, and precision. Therefore, the proposed obstacle detection system operates in the 2D/disparity space while also considering 3D constraints. Neighboring pixels in the disparity image are strongly correlated because it is likely they belong to the same object. In this way, outliers caused by mismatches or occlusions can be easily detected and removed.
In the following section, the underlying principles of epipolar geometry and disparity map computation are introduced. Then, the obstacle detection and segmentation algorithm will be presented, followed by an evaluation comparing the results to measurement from the Velodyne laser scanner on our test vehicle.

4.1 The Epipolar Geometry of Stereoscopic Vision

Epipolar geometry describes the geometric relationship between several camera views. Fig. 4.1 shows an example configuration with two cameras, the left camera centered at $\mathbf{O_L}$ and the right camera centered at $\mathbf{O_R}$. The cameras are related by a rigid transformation composed of a rotation $\mathbf{R}$ and a trans-

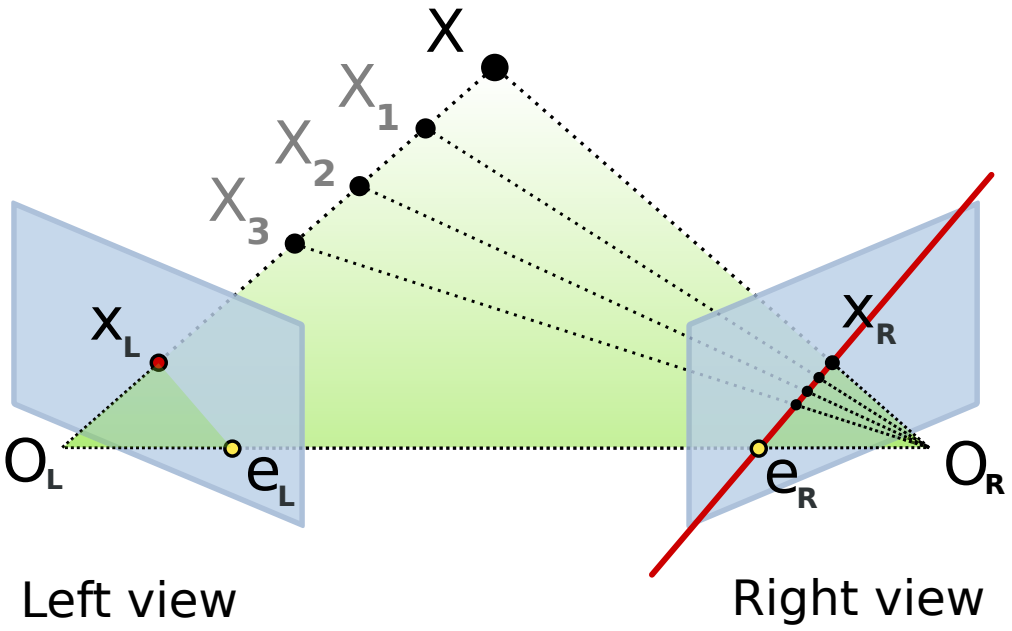


Figure 4.1: Epipolar geometry of a two-camera setup. The points X, X_1, X_2, and X_3 project to the point x_L in the left view, but to different points in the right view. All of these points in the right view are located on the same epipolar line.

lation $\mathbf{t}$. It transforms points in left camera's frame to the right camera's frame, i.e. $\mathbf{X_L} = \mathbf{R} \cdot \mathbf{X_R} + \mathbf{t}$, or in homogeneous coordinates $\mathbf{X_L} = \mathbf{T} \cdot \mathbf{X_R}$, where $\mathbf{T} = \left(\begin{smallmatrix} R & t \\ 0 & 1 \end{smallmatrix}\right)$.
The respective image planes are depicted in blue. A 3D point is projected to a location on the image plane, where the line connecting the 3D point and the camera's projection center intersects with the image plane. For example, $\mathbf{X}$ is projected to $\mathbf{x_l}$ and $\mathbf{O_r}$ to e_l. The latter point is the projection of the right camera's center. The points $\mathbf{e_l}$ and $\mathbf{e_r}$ are called epipolar points or short epipoles. Any projected line through an epipole is called epipolar line. The plane spanned by the camera centers $\mathbf{O_L}$, $\mathbf{O_R}$, and a third 3D point $\mathbf{X}$ is the epipolar plane for $\mathbf{X}$.
If $\mathbf{x_L}$ and $\mathbf{x_R}$ are known, $\mathbf{X}$ can be reconstructed by intersecting the rays $\overline{\mathbf{O_L x_L}}$ and $\overline{\mathbf{O_R x_R}}$. When only $\mathbf{x_L}$ is given, $\mathbf{x_R}$ must be located on the epipolar line, which is the projection of the ray $\overline{\mathbf{O_L x_L}}$. This is known as the epipolar constraint. It allows to search efficiently for corresponding points in the right image given points from the left image (and vice versa).
The epipoles are the blind spots of a stereo camera system. Any point $\mathbf{X}$ along the line connecting the camera centers, cannot be triangulated.

4.1.1 Algebraic Formulation of the Epipolar Constraints

The epipolar contraint for two cameras can be captured in a 3×3 matrix $\mathbf{F}$, the fundamental matrix. It has the following properties:

1. It maps a homogeneous image point from one view to the epipolar line in the other view, i.e. $\mathbf{l_R} = \mathbf{F} \cdot \mathbf{x_L}$ and $\mathbf{l_L} = \mathbf{F^T} \cdot \mathbf{x_R}$
2. The dot between a line and a point on it is 0. From $\mathbf{e_L} \cdot \mathbf{x_L} = \mathbf{0}$ and $\mathbf{e_R} \cdot \mathbf{x_R} = \mathbf{0}$ follows $\mathbf{x_L}^\mathbf{T} \cdot \mathbf{F} \cdot \mathbf{x_R} = \mathbf{0}$
3. The epipoles are mapped to $\mathbf{0}$, i.e. $\mathbf{F} \cdot \mathbf{e_L} = \mathbf{e_R}^\mathbf{T} \cdot \mathbf{F} = \mathbf{0}$
4. $\mathbf{F}$ has rank 2

For simplicity, let us assume the camera matrices are the identity matrix: $\mathbf{K_L} = \mathbf{K_R} = \mathbf{I_{3\times 3}}$. In this case, the fundamental matrix becomes the essential matrix $\mathbf{E}$. In general, they are related by

$$\mathbf{E} = \mathbf{K_R}^\mathbf{T} \cdot \mathbf{F} \cdot \mathbf{K_L},$$

respectively

$$\mathbf{F} = \mathbf{K_R}^{-\mathbf{T}} \cdot \mathbf{E} \cdot \mathbf{K_L}^{-\mathbf{1}}.$$

The essential matrix is interesting because it solely depends on the extrinsic parameters $\mathbf{R}$ and $\mathbf{t}$. Let us assume the left camera defines the reference frame, i.e. $\mathbf{O_L} = \mathbf{0}$ and the transformation of coordinates from the right camera's frame to the reference frame is given by

$$\mathbf{X_L} = \mathbf{R} \cdot \mathbf{X_R} + \mathbf{t}$$

and vice versa

$$\mathbf{X_R} = \mathbf{R^T} \cdot (\mathbf{X_L} - \mathbf{t}).$$

We know that under central projection a 3D point and its homogeneous image point are equivalent up to a scale factor $\lambda \neq 0$. Therefore, the projection of the left camera's center is a multiple of $\mathbf{O_L}$ in the coordinate frame of the right camera:

$$\begin{aligned} \mathbf{e_R} &= \lambda \cdot \mathbf{R^T} \cdot (\mathbf{O_L} - \mathbf{t}) \\ &= \lambda \cdot (\mathbf{R^T} \cdot \mathbf{0} - \mathbf{R^T} \cdot \mathbf{t}) \\ &= \lambda' \cdot \mathbf{R^T} \cdot \mathbf{t} \end{aligned}$$

Let us now consider an arbitrary point $\mathbf{X}$ and its projection $\mathbf{x_R}$. The epipolar line in the right image must pass through $\mathbf{e_R}$ and $\mathbf{x_R}$. As explained in Section 3.1.2, this line $\mathbf{l}$ can be computed by taking the cross-product

$$\begin{aligned}\mathbf{l} &= \mathbf{e_R} \times \mathbf{x_R} \\ &= \lambda' \cdot \mathbf{R^T} \cdot \mathbf{t} \times \mathbf{x_R} \\ &= \lambda' \cdot \mathbf{R^T} \cdot [\mathbf{t}] \cdot \mathbf{x_R}\end{aligned}$$

where $[\mathbf{t}]$ the formulation of the cross-product as a matrix-vector product:

$$[\mathbf{t}] = \begin{pmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{pmatrix}.$$

Thus, the essential matrix $\mathbf{E}$ is defined (up to scale) as $\mathbf{E} = \mathbf{R^T} \cdot [\mathbf{t}]$.

4.1.2 The Standard Stereo Correspondence Problem

The fundamental problem of stereo correspondence is defined as follows: Given an image feature in the reference view, find the same feature, i.e. the projection of the same 3D point, in the other view. By knowledge of the fundamental matrix the search can be restricted to the epipolar line. However, the computational burden is still high because sampled pixels along the line usually have fractional coordinates and the intensity values must be interpolated from the values at the integer coordinates. Additionally, the sampling along the line will usually not allow linear memory access which means that many cache misses will drag down the performance.
That is why most stereo cameras use a parallel setup without rotation between the cameras and a pure lateral translation. In this case the essential matrix will be

$$E = I_{3\times 3}\,[t] = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -t_x \\ 0 & t_x & 0 \end{pmatrix}.$$

Multiplying E with an homogeneous image point $p = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$ yields the epipolar line $l = \begin{pmatrix} 0 \\ -t_x \\ y \end{pmatrix}$, which is a horizontal line. This holds true for any epipolar line and allows an efficient correspondence search along memory-aligned image rows.
Since all epipolar lines are parallel the epipolar point must be a point at infinity. In the standard setup the center of the other camera is at $Z = 0$ and central projection would require a division by 0.

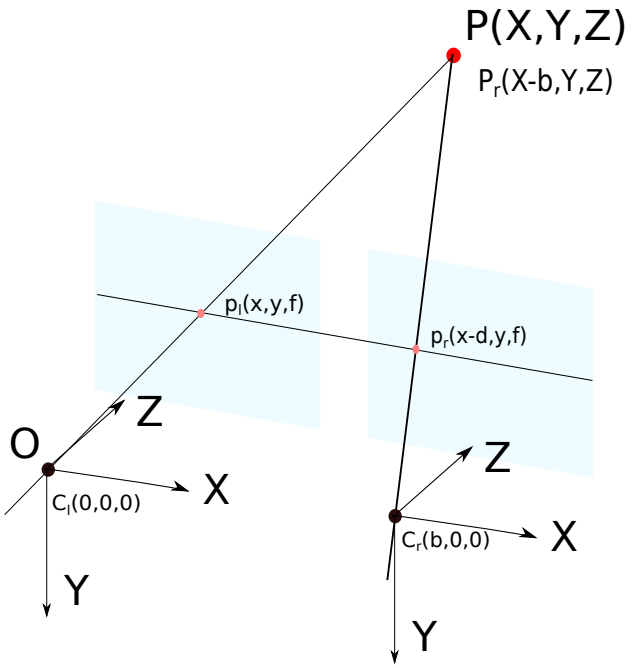


Figure 4.2: Two perfectly aligned cameras with a common focal length f are separated by a baseline b. They observe a point at depth Z. From the equations of central projection follows that the disparity d is computed as $d = \frac{f \cdot b}{Z}$.

In practice, it is virtually impossible to achieve a perfect setup with exactly aligned cameras. Even small rotations or translations will introduce a significant error with respect to 3D reconstruction. Offsets in vertical direction will deteriorate stereo matching because corresponding pixels have different y coordinates. Offsets in horizontal directions will lead to wrong disparity measurements affecting the accuracy of distance measurements.
However, the images can be rectified to what they would look like if they were captured by perfectly aligned cameras. The rectification transform is a homography computed from the inter-camera rotation and translation. It maps both images to a common image plane by means of rotation translation and scaling. The rectification transform is chosen in a way such that the overlapping area is maximized and the amount of perspective distortion is minimized.

4.1.3 The Relation of Disparity and Depth

Let us consider a standard camera setup as shown in Fig. 6.2. Since corresponding points have the same y-coordinates only the disparity, the horizontal offset, is relevant. A disparity map D, which is the output of our embedded stereo camera, stores the disparity for every pixel of the reference image. For example, $D(x, y) = d$ means that the point (x, y) in the left image corresponds to the point $(x - d, y)$ in the right image.
As it turns out there is a very simple relationship between the disparity and the depth Z of a projected 3D point. As shown in Fig. 6.2 the common coordinate system is centered at the projection center of the left camera. Then, the transformation matrix for the right camera is the homogeneous translation

$$\begin{pmatrix} 1 & 0 & 0 & -b \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Both cameras observe a point $\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$ which the left camera projects to

$$\begin{pmatrix} x_1 \\ y \end{pmatrix} = \begin{pmatrix} f\frac{X}{Z} \\ f\frac{Y}{Z} \end{pmatrix}$$

and the right camera projects to

$$\begin{pmatrix} x_2 \\ y \end{pmatrix} = \begin{pmatrix} f\frac{X-b}{Z} \\ f\frac{Y}{Z} \end{pmatrix}.$$

The disparity d is the difference of image x-coordinates

$$\begin{aligned} d =& x_1 - x_2 \\ d =& \frac{f}{Z}(X - (X - b)) = \frac{f \cdot b}{Z}. \end{aligned}$$

The equation $d = \frac{f \cdot b}{Z}$ is the fundamental to the standard stereo setup. It gives insight into the properties of stereoscopic depth measurements. As can be seen in Fig. 4.3, the depth resolution is not equidistant. At close range the resolution is very fine-grained and becomes very coarse at a larger distance. For our camera with focal length $f = 651.6$ and baseline $b = 25\,\text{cm}$ the disparity at $Z = 10\,\text{m}$ is $d = 16.29$. When the distance doubles the disparity halves. At $Z = 50\,\text{m}$ it will be at $d = 3.258$. The smaller the disparity the larger is the distance gap $\frac{f \cdot b}{d-1} - \frac{f \cdot b}{d}$. That is the main disadvantage compared to LIDAR scanners which measure distances at a constant resolution.
Another drawback is the small number of possible disparity values. The block-matching algorithm on the stereo camera has a disparity range of [0..127]. Its matching algorithm compares every pixel location in the left

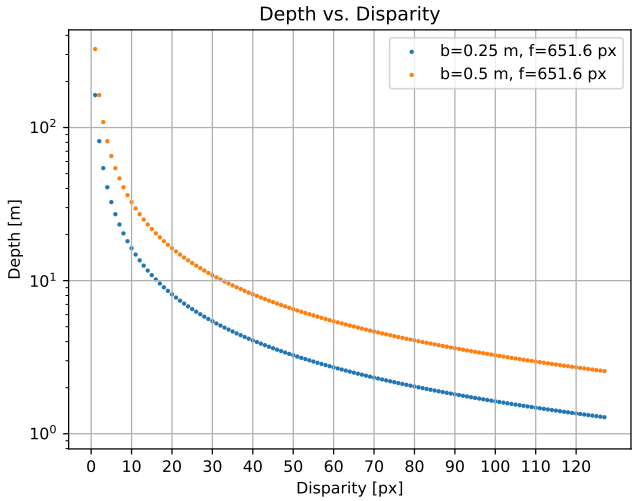


Figure 4.3: The relation of disparity d and depth z is governed by $z = \frac{f \cdot b}{d}$. Our stereo camera has a base line of $b = 25\,\text{cm}$ and a focal length of $f = 651.6$ px. The disparity resolution is very dense at close range and coarse at large range. Distances beyond 10 m have a disparity ≤ 16.29.

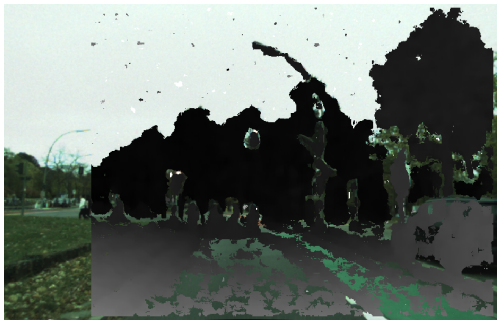
image to 128 pixel locations in the right image. Using a window around the locations it computes the Hamming distance of Census-transformed pixels. In a winner-takes-it-all fashion the disparity with the lowest cost is chosen. Then, it uses the matching cost function values $C(d-1)$, $C(d)$ and $C(d+1)$ to interpolate a disparity with fractional part. The sub-pixel interpolation is a heuristic which very much depends on the smoothness of the cost function. Other options to increase the depth resolution are a wider baseline and a larger focal length, i.e. a narrower field of view.

4.2 3D Image Segmentation

The disparity image provided by the stereo camera holds disparity values for pixels of the left camera's image. The re-projection matrix Q is defined as follows:

$$Q = \begin{pmatrix} K_{11}^{-1} & 0 & 0 & K_{13}^{-1} \\ 0 & K_{22}^{-1} & 0 & K_{23}^{-1} \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{1}{t_x \cdot f} & 0 \end{pmatrix}$$

(a) The left (rectified) view of the stereo images. The other camera is positioned 25 cm to the right.



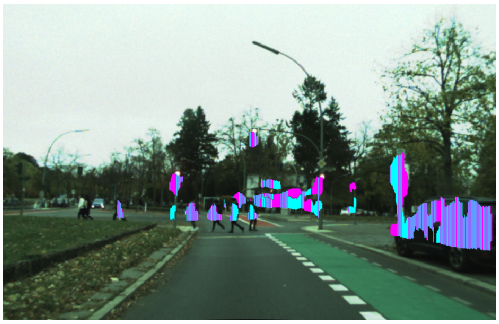
(b) The left (rectified) view with disparity overlay. The brightness corresponds to the disparity values $\in [0..127]$. The camera computes disparity values starting at column 128.

Figure 4.4: The embedded stereo camera outputs color images and disparity images.

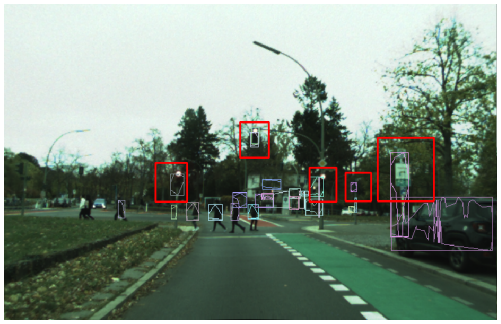
where K is the intrinsic matrix of stereo camera's reference view, $f = K_{11}$ is the focal length and t_x the lateral distance between the two cameras (the baseline b). Q maps a vector $\begin{pmatrix} u \\ v \\ d \\ 1 \end{pmatrix}$ containing image point $\binom{u}{v}$ and disparity d to the 3D point $\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \in \mathbb{P}_3$. The mapped points reside in the camera's coordinate system. Using the camera-to-car transformation $T_{4\times4}$ we can directly transform them to vehicle coordinates using the modified re-projection matrix $Q' = T_{4\times4} \cdot Q$.
My obstacle segmentation algorithms works on the 3D image created by re-projection of the disparity image with Q'. Presuming perfect extrinsic calibration and stereo depth measurements, the ground plane points will be at $z = 0$ and planar faces perpendicular to the ground will have constant Z values along the columns of the 3D image.
The 3D image is processed in several steps:

1. Scan for vertical constant-depth stripes (v-stripes).
2. Merge neighboring v-stripes to objects.
3. Fill the occupancy grid according to the obstacles' boundaries.
4. Extract free space from the occupancy grid.

Each of these steps will be discussed in detail in the following subsections.



(a) Vertical 1-column wide stripes alternating in cyan and magenta. Colors follows the scan order, which is first top-down, then left-to-right.



(b) Bounding boxes and 2D contours of detected objects.

Figure 4.5: The segmentation works on the reconstructed 3D image. It is scanned top-down column-wise to find stripes of nearly constant distance. The vertical stripes are merged according to proximity criteria.

4.2.1 Detection of Vertical Constant-Depth Stripes (V-Stripes)

This step of the algorithm searches the 3D image for one column wide vertical stripes of roughly constant longitudinal distance with respect to the local car coordinate system. The outline of the algorithm can be found in the Appendix (Alg. 7). The search space of the algorithm is limited to a volume of interest, e.g. the road area with a height range of 0.3 m–2 m. In a double loop the 3D image is traversed column-wise from top to bottom. A vertical stripe is started if a point is within the volume of interest and if it is at a depth edge, i.e. the above point is much further away. The stripe continues as long as the longitudinal distance of a succeeding pixel does not deviate too much from the minimum found so far. Small gaps due to missing values in the disparity image are ignored. Once a second depth edge has been reached the vertical patch is complete, and another patch might be found in the in the remainder of the column.

4.2.2 Horizontal Merge of V-Stripes

When all columns have been processed the vertical stripes are merged to clusters, if they are close in pixel distance as well as in 3D distance. The merge criteria have been determined heuristically to work for a high level of disparity noise. Connected vertical stripes must be

- less than 10 image columns apart,
- less 1 meter apart in lateral distance,
- and less then 0.5 meter apart in height.

A graphed-based connected component algorithm merges stripes according to these criteria as outlined in Alg. 8 (see Appendix). The result are clusters of stripes with a 2D contour defined by outer contour of the clustered stripes and a 3D contour defined by the 3D points belonging to the stripes.

4.2.3 Filling the Occupancy Grid

In my implementation I have used the ROS Grid Map library[59], an efficient implementation of occupancy grids. The grid size was set to 100 m length and 16 m width with square cells of 0.25 m width. The library offers a convenient way to iterate over cells within a polygon. This feature allows to consider only cells within the stereoscopic field of view.
The 3D points of the segmented objects are filled into the occupancy grid according to the depth uncertainty of the associated disparity value. The lower the disparity the more cells in longitudinal direction must be updated. In my implementation a disparity delta of 0.5 was used to compute the range of grid cells to be updated.

4.3 Experimental Evaluation

4.3.1 Comparison of Free Space Derived from Occupancy Grids

For this evaluation the occupancy grid cells are scanned radially with the camera position at the center. The horizontal field of view is separated into 40 equal angular segments. For each segment the distance to the closest occupied cell is determined. A cell is regarded as occupied if the maximum height exceeds 0.3m. The same is done with an occupancy grid computed from measurements of the Velodyne HDL-64 laser scanner. The Velodyne has 64 lasers covering approximately 26.4° vertical field of view, which is an average vertical resolution of 0.4°. The rotation frequency of 10 Hz yields a horizontal resolution of 0.16°. It is not as dense as the stereo point cloud, but it is sufficient for comparison. Fig. 4.6 depicts an overview of the test scenario. The evaluation is limited to the camera's stereoscopic field of view, i.e. where depth measurements are available.

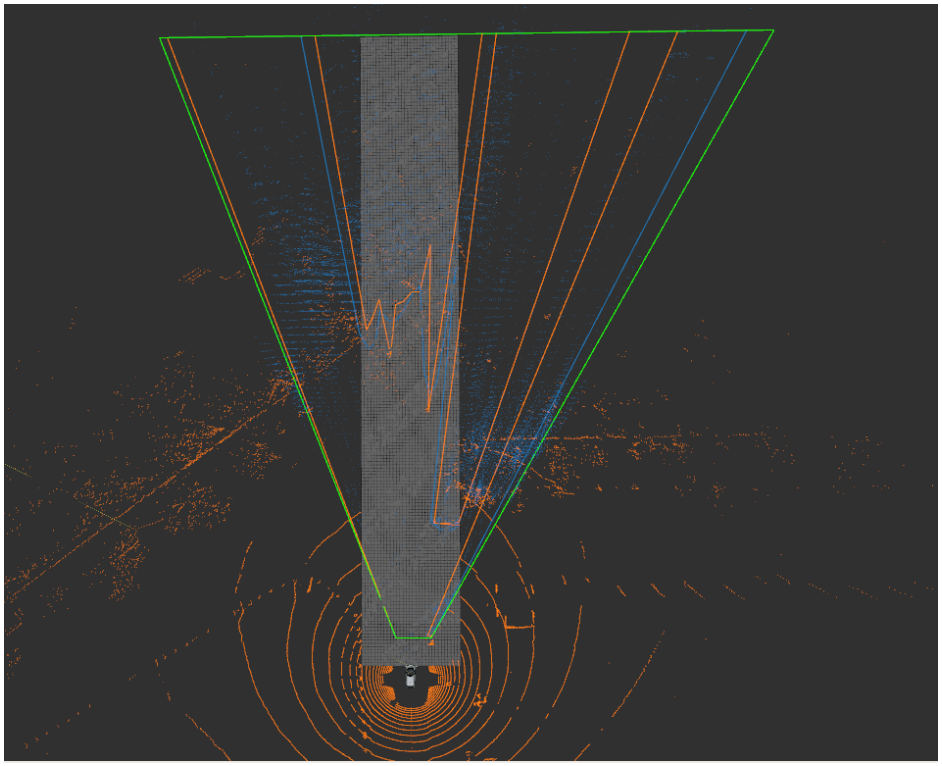


Figure 4.6: This is a top view of the evaluation scenario. The occupancy grid is displayed in gray, the stereo point cloud in blue, and the velodyne point cloud in orange. The green polygon is the stereoscopic field of view. The area of interest for the evaluation is the intersection of the grid and the green polygon. The orange polygon shows the radial distance computed from stereoscopic occupancy grid, while the blue polygon was computed from the velodyne occupancy grid.

On a test sequence of 12365 stereo images the free space polygons for the stereo camera and for the Velodyne have been computed and compared by the Intersection-over-Union metric (IoU). It measures the overlapping area divided by the union of both polygons. The average IoU over the whole video sequence was 0.83 with a standard deviation of 0.11. As can be seen in Table 4.1 the free space congruence increases if the distance is limited to shorter ranges.

4.3.2 Comparison of Stereo Obstacles and Velodyne Measurements

In this experiment the position accuracy of detected stereo obstacles is evaluated. The contour of each obstacle was intersected with the midpoint of

Distance Range [m]	<10 m	<20 m	<30 m	<40 m	<50 m	<60 m	<70 m	<80 m	<90 m	<100 m
Free Space IoU average	0.96	0.94	0.92	0.90	0.88	0.86	0.85	0.84	0.83	0.82
Free Space IoU standard deviation	0.07	0.09	0.09	0.10	0.10	0.10	0.10	0.11	0.11	0.11

Table 4.1: Comparison of Stereo vs. Velodyne free space by the Intersection over Union metric (IoU).

Obstacle Distance [m]	N	TP rate	FP rate	FP distance average [m]	FP distance standard deviation [m]
<10 m	24095	0.89	0.11	0.34	0.64
10 m–20 m	49783	0.84	0.16	0.43	1.10
20 m–30 m	31994	0.71	0.29	0.78	1.05
30 m–40 m	20736	0.64	0.36	1.37	1.81
40 m–50 m	16484	0.50	0.50	1.77	1.94
50 m–60 m	10290	0.44	0.56	3.10	3.51
60 m–70 m	13796	0.41	0.59	4.19	5.23
70 m–80 m	6994	0.29	0.71	5.89	8.13
80 m–90 m	8286	0.16	0.84	9.07	10.65
90 m–100 m	3475	0.08	0.92	13.31	12.54
any range	185933	0.65	0.35	3.47	6.56

Table 4.2: True positives (TP) are obstacles supported by Velodyne measurements. False positive (FP) have zero support. FP distance refers to the distance between a FP and the closest Velodyne measurement.

occupied cells from the Velodyne occupancy grid. The grid has a length of 100 m, a width of 16 m and grid cell size of 25 cm. Table 4.2 shows the results for a video sequence of 12365 stereo images. The true positive detections (TP) are all obstacle whose contour contour contains at least one occupied cell of the Velodyne occupancy grid. Otherwise it is counted as false positive detection (FP) and the distance to the closest occupied cell has been conputed (FP distance). The TP rate is highest at close range (0.89 at less than 10 m distance) and falls off with increasing distances. Conversely, the distances of false positives to Velodyne measurement increase with the distance limit.

4.4 Conclusions

The presented stereo obstacle detection system can compute the available free space in front of the car with reasonable accuracy. The camera's depth measurement noise is considerably higher than that of a Velodyne laser scanner. Object detections on a frame-by-frame basis have large position uncertainties increasing with the object distance. The object boundaries in the image are more robust than the distance measurements, because depth edges provide

a strong prior for segmentation. Hence, image-based tracking of obstacles is more stable than 3D tracking.
The localization of static objects can be improved by filtering distance estimates over time. In Chapter 5 we will see an application of the obstacle detection system to the problem of mapping positions of traffic light poles.

Chapter 5

Traffic Light Detection

This chapter presents a traffic light detection system for urban driving scenarios with focus on robustness and real-time capability. It has been successfully tested with our intelligent car *MadeInGermany*. For the purpose of traffic light detection our car is equipped with two monocular cameras covering a wide field of view in order to enable close-range detection. Additionally, the system relies on a map which contains the global positions of all traffic lights and their association to lanes. Thus, when the ego-position and the planned route is known, the position of the next relevant traffic lights can be extracted from the map and, using the known extrinsic calibration, the position of the traffic light in the images can be predicted. Effectively, only small areas of the images must be processed, leading to a higher computational efficiency and better detection performance.

5.1 Prior Knowledge about Traffic Lights

Traffic lights have a well defined appearance. For example in German, they are specified by official regulations which define measurements, light frequencies, brightness etc. Different stages of the presented traffic light detection system use these information to accept or reject intermediate detection results. The most relevant priors used by my implementation are:

1. The traffic signal light is monochromatic red (613-631 nm), yellow (585-597 nm), and green (489-508 nm). The emitted light is directed. It has maximal luminance towards its principal direction.

2. Circular traffic light signals have a diameter of either 20 or 30 cm.

3. Traffic lights signals are mounted with a housing at certain heights,

Figure 5.1: Two AV Guppy F-036C cameras were mounted with suction cups behind the windshield. The camera are pointed slightly upwards and sideways to maximize the covered field of view for close-range traffic light detection.

either on poles in ca. 2 m height or hanging above lanes in circa 4.5 m height.

4. The traffic light housing is usually of rectangular shape and has a dark green, gray or black color.

5. The different signal elements are ordered vertically with the red light at the top.

6. There are four valid active states (in Germany): Red light, red and yellow light, yellow light, and green light.

These rules are represented by different parameters in the detection modules, which can be adapted if some of the rules change, e.g. in other countries.

5.2 System Overview

The overall system is shown in Fig. 5.2. It consists of four components:

1. ROI prediction: Given the planned trajectory and the ego-position, find the relevant traffic lights in the map and project their positions to regions of interest (ROIs).

2. Detection: Search each ROI for traffic light candidates.

3. Tracking and Selection: Track all traffic light detections over time, and choose for each ROI the most likely traffic light.

4. Traffic Light State Selection: Decide most likely state from all detections.

The traffic light detection system processes the image stream of two color cameras of type Guppy F-036C, which are fully calibrated with respect to intrinsic and extrinsic parameters. They are mounted behind the windshield pointed slightly upwards and to the side in order to cover a maximum field of view for close range detection. Detection is triggered only when traffic lights are expected to be visible. This mechanism relies on information from other modules:

- The vehicle *MadeInGermany* has a very precise localization system, the Applanix POS LV 200. Additionally, differential GPS and a lane detection system increase the accuracy. In combination with a map of global traffic light locations, including the height, it is possible to project the expected position of the traffic lights. The projection accuracy directly depends on the quality of the map and of the localization. Depending on the estimated accuracy a volume around the 3D position is projected and represented as a bounding box in the image. This box serves as Region of Interest (RoI) and thus simplifies the detection task. Small RoIs reduce the number of false positives significantly.

- Applanix' navigation solution delivers exact roll and pitch measurement for the car, allowing the correction of the pitch and roll rotations induced by the uneveness of the road. These high-frequency rotations move the horizon and the correction is necessary to compute a correct projection of traffic light positions to the images.

- The measurement of the ego-velocity enables the exact prediction of tracked traffic lights.

- The path planning module creates a trajectory along road lanes, to which the associated traffic lights are stored in the map. Thus, only the traffic lights relevant to the planned trajectory must be considered for the RoI projection.

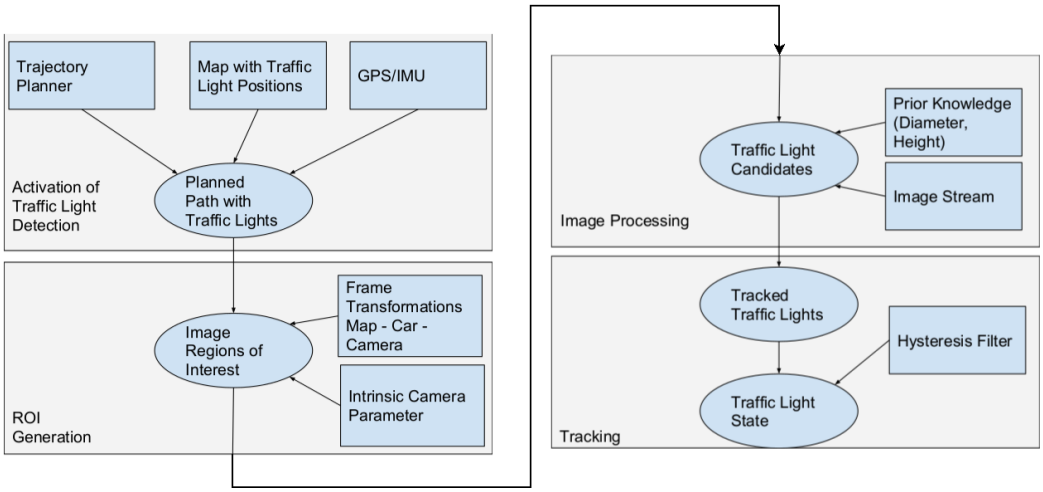


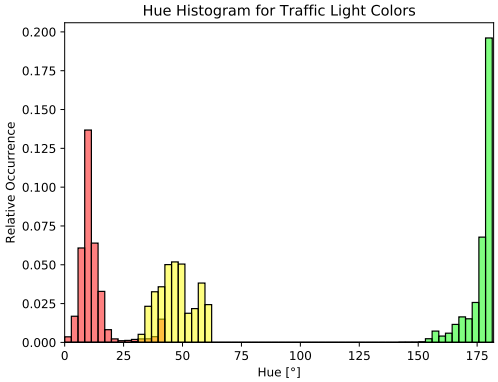
Figure 5.2: System overview: Map-assisted traffic light detection system for MadeInGermany

5.3 Computation of Regions of Interest (ROI)

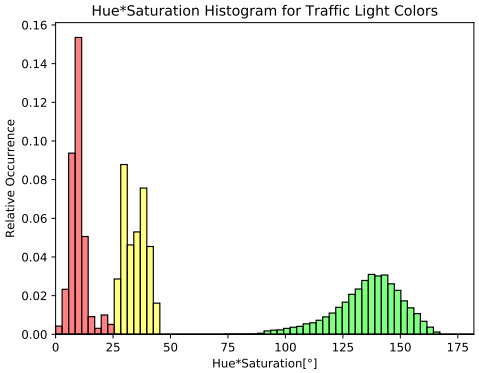
While driving autonomously *MadeInGermany*'s planning module decides which route to take and which lanes to follow. The route planning takes place on a graph deduced from the annotated digital map. There is a N-to-M mapping between lanes and traffic lights. A lane can be associated to several traffic lights and a traffic light can govern several lanes. Knowing the planned trajectory, the ROI generation module retrieves all relevant traffic light positions and maps them to image coordinates. If there is no traffic light within the planning horizon, ROIs will not be generated and the detection will not be triggered.
The projection uses the global pose of the car (R_w, t_w), the pre-calibrated extrinsic matrix of the camera with respect to the car (R_c, T_c), and the intrinsic calibration of the camera K. R_w contains the car's heading direction (yaw angle) and the roll and pitch angles which determine the car's attitude with respect to the ground plane. The homogeneous projection matrix, which maps global a point P to an image point p is constructed as follows:

$$p_{3\times1} = \begin{bmatrix} K & 0 \end{bmatrix}_{3\times4} \begin{bmatrix} R_c & t_c \\ 0 & 1 \end{bmatrix}_{4\times4} \begin{bmatrix} R_w & t_w \\ 0 & 1 \end{bmatrix}_{4\times4} P_{4\times1}$$

The actual ROI is the projection of the traffic light's known 3D extent plus a margin of 1 meter to reflect errors of the mapped positions or of the vehicle's localization. The procedure is repeated for the second camera's image as well and the ROIs are then forwarded to the image processing modules.



(a) A histogram of hue values from detected traffic light pixels. Red and yellow hue ranges overlap.



(b) The product of hue and saturation yields a better separation of red and yellow pixels.

5.4 Detection of Traffic Lights within a ROI

This part of the pipeline constructs and verifies the traffic light candidates, which consist of a bounding box for each of the three color signals and their state (on or off).

HSV Color Classification

The first step converts the images from RGB to HSV (hue, saturation, value) color space. The conversion is defined as:

$$
\begin{aligned}
&R, G, B \in [0..1] \\
&M := \max(R, G, B) \\
&m := \min(R, G, B) \\
&H := \begin{cases} 0 & \text{if } R = G = B \\ 60° \cdot (0 + \frac{G-B}{M-m}) & \text{if } M = R \\ 60° \cdot (2 + \frac{B-R}{M-m}) & \text{if } M = G \\ 60° \cdot (4 + \frac{R-G}{M-m}) & \text{if } M = B \end{cases} \\
&H := H + 360° \text{ if } H < 0 \\
&S := \begin{cases} 0 & \text{if } R = G = B = 0 \\ \frac{M-m}{M} & \text{else} \end{cases} \\
&V := M
\end{aligned}
$$

With respect to traffic light colors, HSV has more meaningful components than RGB. Traffic light signals emit monochromatic bright light, which di-

	Hue [°]	Saturation ∈ [0..1]	Value ∈ [0..1]	Hue · Saturation [°]
Red	0°-43°	0.3-1.0	0.3-1.0	0-27.7
Yellow	19°-84°	0.3-1.0	0.3-1.0	27.7-44.3
Green	144°-181°	0.5-1.0	0.3-1.0	55.4-166.1

Table 5.1: Thresholds used for color classification. Values may vary depending on the camera settings and hardware specifics.

rectly translates to a small hue range with high value and high saturation. However, cameras measure all incoming light which includes also the ambient light from the sky and other light sources. Figure 5.3a shows a typical distribution of hue values collected from samples of detected traffic lights. The hue ranges of red and yellow pixels overlap significantly leading to potential confusion of red and yellow lights. The influence of ambient (white) light can be measured by the saturation of traffic light pixels. Adding white light to monochromatic light increases all RGB values and therefore reduces the saturation. Depending on the white balance settings of the camera, white light can increase one RGB component more than others. In case of our cameras, there is a slight bias towards green. Ambient white light shifts the hue angle towards green (hue=180°). Therefore, in Fig. 5.3a the red pixels with larger hue values tend to have a lower saturation. The overlap of hue ranges can be resolved by evaluating the product of hue and saturation. As shown in Fig. 5.3b the product creates a good separation of red and yellow pixels.
The classification performs a color thresholding on values of hue, saturation, value, and the product of hue and saturation. Thresholds are sensor dependent and have been derived from collected data. A typical parameter set for our cameras is shown in Table 5.1. The pixels within the ROIs are tested against thresholds and collected for further processing.
For performance reasons the number of points per color class has been limited to $N = 2000$. The exact number depends on available processing time and the resolution of the image. It guarantees that the system is not bogged down by circumstances which create a lot bright colored points, e.g. direct sun-light into the camera. When there too many points, the ones with the least saturation are discarded.

Color Point Clustering

This phase of the algorithm finds connected components in the pixels of each of the three color classes. The clustering method uses OpenCV's implementation of the *Connected-Component* algorithm from Chapter 21 in *Introduction*

to Algorithms[60]. It is a graphed-based method working with Union-Find data-structures. The adjacency relation of the graph can be freely defined which offers more flexibility than a simple flood-fill algorithm. For example, one can define an inter-pixel distance larger than 1 to merge across gaps. The draw-back is the $O(N^2)$ asymptotic run-time of the algorithm. Therefore, I have implemented a two-step variant of the connected component analysis. The points of a color class are ordered by their image position from top left to bottom right. The list of points is split into parts spanning at most two rows. Then, the connected component algorithm is run on each of the subset which generates a list horizontal stripes of color pixels. A second iteration of the connected component algorithm merges the horizontal stripes to larger cluster spanning more than two image rows. A pseudo-code implementation of this clustering method can be found in the Appendix (Alg. 9).
The resulting clusters are then filtered by a range of perspective and geometric filters:

1. The width-to-height ratio must be close to 1.
2. The ratio of number of pixels and the bounding box area must larger than a density threshold.
3. The size must be consistent with the expected (projected) size derived from the known 3D position and known light diameter.

Construction of Traffic Lights from Detected Light Blobs

In this phase the green, yellow, and red light blobs from the previous stage are analyzed to generate hypotheses of 3-segment traffic lights.
The algorithm loops over all colored clusters and computes the theoretical positions of the other (unlit) lights. Depending on the color these may be above or below of the given light blob. It requires to know the projection of the 3D up direction for an image position. Since this is a recurring operation the up vectors are computed beforehand and stored in a lookup table. The up vector is computed by re-projecting an image point to any point along the direction of a pixel. Then the height of this point is increased and the modified point is projected back to the image, yielding the up direction in the image.
The areas of unlit light segments are tested for low saturation ($S < 0.15$) and moderately low value ($V < 0.5$). If the density, the number of these pixels divided by the area, is larger than a threshold, the three segments are accepted and stored as a traffic light candidate.
A special case is the state in which the red and yellow light are switched on

simultaneously. To detect these, all pairs of red and yellow light blobs are tested if they are close to each other and of similar size. If the region at the position of the green light passes the test the candidate is accepted.
Pseudo-code of the algorithm can be found in the Appendix (Alg. 10).

Tracking of Traffic Lights and State Determination

The tracking of detected traffic lights is very effective to remove spurious false positives. The tracker enforces:

- temporal consistency (remove spurious detection),
- spatial consistency (feasible trajectory), and
- size consistency (diameter).

When a new set of measurements arrives, the already tracked traffic lights are predicted to the time stamp of the new measurements, using the known ego-motion. Then, the new detected traffic lights are matched to tracked ones by comparing their size and the distance between measured and predicted position. Tracking enforces consistency because it removes traffic light candidates not updated for some time.
For each input region of interest a tracked traffic light will be selected if its last observation happened within a small time range and it has been tracked for at least 3 frames. If several candidates are available the one with the most plausible light blob diameter is chosen.
Finally, the joint traffic light state must be decided. If the detected states from all ROIs have are contradictory the majority wins. In case of a split vote, the more prohibitive signal wins. The joint state is then passed through a hysteresis filter. A change is published only if the new state has been detected three times in a row.

5.5 Experimental Evaluation

The testing track is a 13km long loop on the Straße des 17. Juni, in the inner city of Berlin as shown in Fig. 5.4. It comprises of four-lane roads with two large roundabouts and intersections with groups of 2 to 4 traffic lights. The evaluation has been run on image data from different days with different lighting conditions. Altogether there were 130 traffic lights belonging to 44 groups where a group is defined by having a common state. Table 5.2 shows statistics for the detection of single traffic lights. The first column refers to the distance at which the traffic light detection was trying to detect it for the

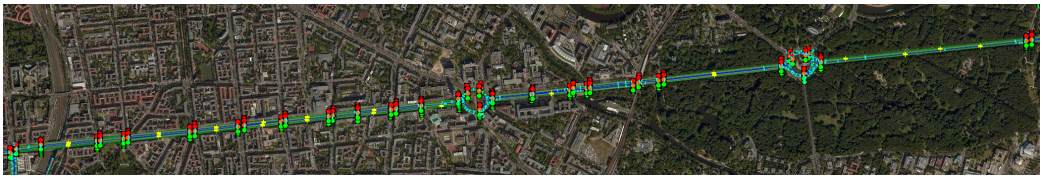


Figure 5.4: Straße des 17. Juni, a main avenue in Berlin with two large roundabouts and more than 130 traffic lights.

First Sight [m]	N	Average First Detection [m]	Standard Deviation [m]	Not detected
<40 m	20	15.6	11.4	6
40 m–60 m	19	38.4	6.5	0
60 m–80 m	55	40.1	22.8	4
80 m–100 m	477	68.4	27.5	32
any Range	571	62.8	29.4	42

Table 5.2: Detection results for single traffic lights. First sight denotes at which range the detection system was triggered the first time for a traffic light.

first time. A successful detection is assumed when it has been detected for at least ten times. Some of the traffic lights appear at rather short distances. This usually happens in the roundabouts where there are many intersections close to each other. Both roundabouts have five connecting roads with up to four lanes each. The average detection range was 62.8 m with a rather high standard deviation of 29.4 m. There are two main reasons for missed or late detections. First, there is dense traffic on that street, large vehicles may obstruct line of sight to a traffic light. And second, there were two types of light sources, LED lights and conventional light bulbs focused by a mirror behind them. The LEDs emit bright light in almost any direction, while the conventional lights have a narrow cone of high luminance. This was especially problematic at the exit of the roundabout where the car approaches the traffic light at a steep angle.
On many intersections there are multiple traffic lights showing the same state. Often it is enough to detect only one of them correctly. Table 5.3 shows results for this case. Here, the average first detection range was at 90.2 m. There are some outliers, where the state could only be determined very late which happens for the reasons mentioned before, especially for groups of one or two traffic lights.

First Sight [m]	N	Average First Detection [m]	Standard Deviation [m]	Worst First Detection [m]
<40 m	9	27.3	7.1	13.6
40 m–60 m	10	48.5	2.4	43.9
60 m–80 m	22	74.8	3.6	69.7
80 m–100 m	153	98.9	3.9	80.5
any Range	194	90.2	19.4	13.6

Table 5.3: Detection results for traffic light groups. Traffic lights with a shared state constitute a group. On average three traffic lights belong to a group of which at least one must be detected. First sight denotes at which range the detection system was triggered the first time for a group.

5.6 Stereoscopic Traffic Light Detection and Mapping

Traffic light detection can employ the depth measurements to discard single pixels at very early stage[42] because traffic lights appear either on poles next to the road or hang above the road. However, because of the inverse relationship of disparity and depth (see Section 4.1.3) this constraint cannot be enforced very rigorously.
Robuster than a pixel-wise depth evaluation is the detection of the poles using the obstacle detection system from Chapter 4. It provides 3D locations of potential traffic lights which the traffic light detection system maps to regions of interest. The 3D positions of detected and tracked traffic lights are collected and used to create a filtered estimate of the global 3D traffic light position. This solves the problem of annotating maps with positions and heights of yet unknown traffic lights.

5.6.1 Detection of Potential Traffic Light Locations

For the purpose of this task the obstacle detection system from Chapter 4 is parametrized in two ways. Firstly, the 3D regions of interest define a "tunnel" around the road. Objects on the road are not of interest. Instead the volume of interest covers the area next to the road up to a height of 3 m and the height range of 3 m to 6 m above the road.
Secondly, the found objects are filtered by their 3D shape. Traffic lights next to the road are mounted on poles usually at a height of 2 m and the whole housing is at least 1 m tall. Hence, the algorithms scans for thin (width < 0.5 m) and tall (height > 2.5 m) objects. Traffic lights hanging above the road have a height of at last 1 m and the height-width ratio of the housing is circa 3 to 1. Also, it must be surrounded by clear depth edges as there should be no other object nearby.
All objects meeting these criteria are passed as potential locations of interest to the traffic light detection system. The resulting regions of interest usually capture potential traffic lights, but also appear on pole-like structures like trees, lamp posts or sign posts.

Fig. 5.5 shows an example scene which illustrates the criteria for obstacles to be selected as potential traffic light. The pedestrians on the road are not selected because they are outside the volume of interest and they are not tall enough. The car parked on the right side is too wide to be a traffic light pole, and the same is true for the obstacles far in the back.

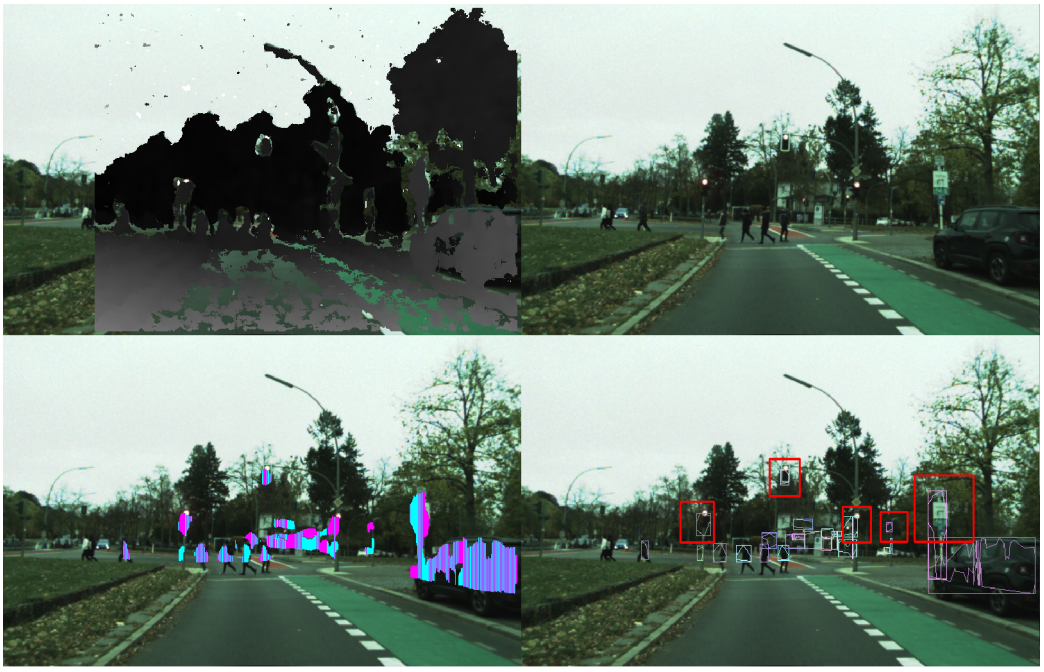


Figure 5.5: From top left to bottom right: 1. Image with overlaid disparities 2. Left image view 3. Vertical obstacles stripes 4. 2D obstacles contours and bounding boxes. Proposed traffic light RoIs are in red.

5.6.2 Automatic Mapping of 3D Traffic Light Positions

The 3D objects will be mapped by the traffic light detection to image regions of interest and from that point the detection pipeline proceeds as described in the previous sections. It will detect traffic lights, and track them including their (global) 3D positions. A tracked traffic light will expire once it has not been seen for some defined duration threshold. The discarded tracks will be analyzed if they are long enough and if the last position has been close to the car, i.e. if the car has just passed it. If these criteria are met the positions are combined using the measurement update steps of a Kalman Filter.
The Kalman Filter[61] models the global (x, y) coordinates. The height is derived from the stereo obstacle's height and set to either 2.1 m for pole-mounted traffic lights, or 4.5 m for suspended traffic lights. The measurement and transition matrix are set to the identity matrix. The measurement noise matrix is directly derived from the covariance matrix of the points belonging to the 3D object received from the stereo obstacle detection. The process noise is assumed to be very small, it is basically the localization error of the GNSS system (Applanix POS LV 550).
The traffic light's facing direction is stored as the opposite of the car's heading

Number of ROIs Average	Number of ROIs Standard Deviation	Traffic Light Detection Range Average[m]	Traffic Light Detection Range Standard Deviation [m]
5.1	2.1	28.3	5.3

Table 5.4: The stereo obstacle detection generates on average 5–6 regions of interest. The first detection of an actual traffic light pole happened on average at a distance of 28 m.

direction, when it was closest to the traffic light. The final filtered position will be published and stored in a database. The mapping is repeated multiple times to compensate for localization and spurious detection errors. Given enough repetitions, repeatedly mapped traffic light will cluster around the their true position.

5.6.3 Experimental Evaluation

Traffic Light Proposal

The first evaluation metric compares regions of interest generated from disparity maps with those generated from pre-mapped GPS positions. There are two metrics: the distance at which the correct RoIs are generated and the number of proposed 3D objects per frame. The experiment was conducted on video data where the cars drives through Berlin and occasionally encounters intersections with up to three traffic lights. The stereo obstacle detection system searches every frame for potential traffic light objects. Pole-like structures like trees, lamp post, traffic signs appear frequently such that on average the system proposes 5–6 potential traffic light objects per frame. As can bee seen in Table 5.4 the actual traffic light poles were detected at distances of up to 28 m. At larger distances thin structures like poles could not be reliably segmented due to a lack of depth resolution.

Traffic Light Position Mapping

The second experiment compares the mapped traffic light positions with manually mapped positions at the Thielallee intersection close to the Freie Universität Berlin. For each direction there was one suspended and two pole-mounted traffic lights. The traffic light detection module will search all proposed regions of interest and create a track of detections including the estimated map positions. Table 5.5 shows the numerical results for the intersection depicted in Fig. 5.6. Traffic lights detections are clustered around the ground truth mapped positions. The longitudinal position standard de-

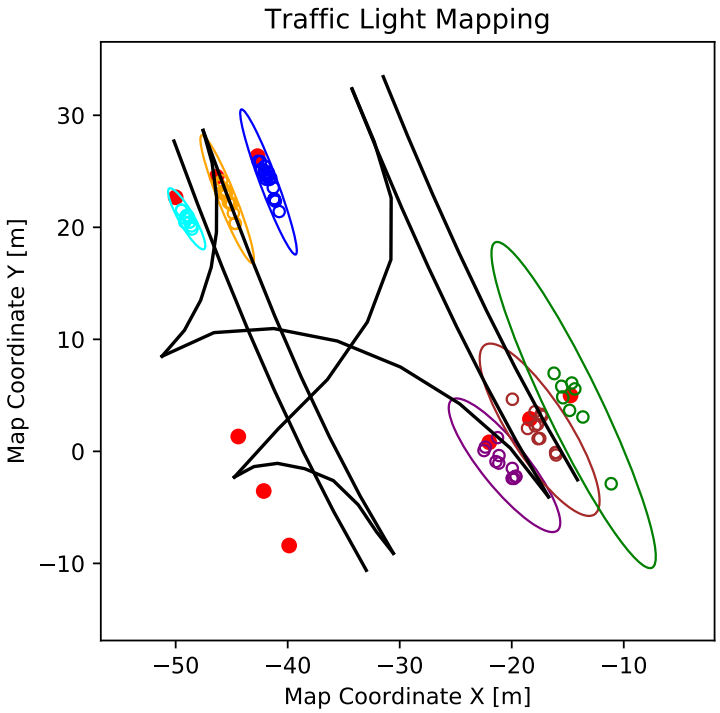


Figure 5.6: The plot shows the outline of a small 3-way junction near the Freie Universität Berlin. The manually mapped positions of traffic lights are marked red, three at each stop line. Each colored circle corresponds to the final state of a Kalman filter which has processed a whole sequence from the tracker. The ellipses visualize the covariance of a cluster.

viation is on average 1.6 m, while the lateral deviation is on average 0.3 m. The mapping was evaluated on image data from five different days. Repeated mapping is necessary to filter out false positive detections which do not create such dense clusters of positions.

Number of Detection Tracks	Ground Truth X [m]	Ground Truth Y [m]	Mapped X [m]	Mapped Y [m]	σ_1 [m]	σ_2 [m]
22	-42.7	26.4	-41.7	24.0	0.2	1.4
15	-46.3	24.5	-45.4	22.5	0.2	1.2
9	-50.0	22.7	-49.0	20.8	0.1	0.6
13	-22.0	0.8	-20.6	-1.2	0.4	1.5
12	-18.4	2.9	-17.5	1.9	0.5	1.8
8	-14.8	5.0	-14.5	4.1	0.5	3.2

Table 5.5: Detection results for the six traffic lights depicted in Fig. 5.6. Data was collected on five different days. A detection track is a whole sequence of detections filtered to one position estimate. The standard deviations refer to the scatter of the mapped positions.

5.7 Conclusions

The presented traffic light detection system using two monocular cameras can robustly detect traffic light states at intersections. The distance of first detection was on average 60 m. The main challenges were occlusion by other vehicles and the narrow cone of high luminance for some non-LED traffic lights. The detection was guided by a map with annotated traffic light locations and their association to lanes.

The second part of this chapter showed how a combination of my stereoscopic obstacle detection and the traffic light detection maps the locations of yet unknown traffic lights. The former system proposes potential traffic light objects, while the latter verifies and filters the 3D positions. The position's longitudinal standard deviation was on average 1.6 m and the more important lateral standard deviation was around 0.3 m.

Chapter 6

Visual Ego-Motion Estimation

This chapter presents an ego-motion estimation method that processes disparity maps provided by our stereo smart camera and sparse optical flow in order to measure the car's velocity and angular velocity. First, we will quickly recapture the concepts of motion field and optical flow. Then, the algorithm will be presented followed by results from experiments, where visual ego-motion is compared to measurements of our very precise navigation solution (Applanix POS LV 220).

6.1 The Motion Field of a Moving Camera

The motion field is a vector field that describes the movement of 3D points relative to the camera. Besides illumination changes, the motion flow is the principal cause for changes in image intensities, also known as optical flow. In static scenes optical flow is exclusively caused by the camera's movement. The presence of independently moving objects causes additional optical flow. With monocular cameras these two components cannot be easily separated which makes visual ego-motion estimation from optical flow challenging.

The Projection of Motion

As can be seen in Fig 6.1, 3D coordinates refer to the camera's coordinate frame at a given time t. The observed motion vector on the image plane is a projection of the object's movement. The velocity of a 3D point P is the first derivative by time t: $\mathbf{V} = \frac{d\mathbf{P}}{dt}$. In case of discrete time steps, the velocity is the change between two consecutive poses. The transformation between two poses is defined by a rotation matrix $\mathbf{R}$ and a translation vector $\mathbf{T}$. The

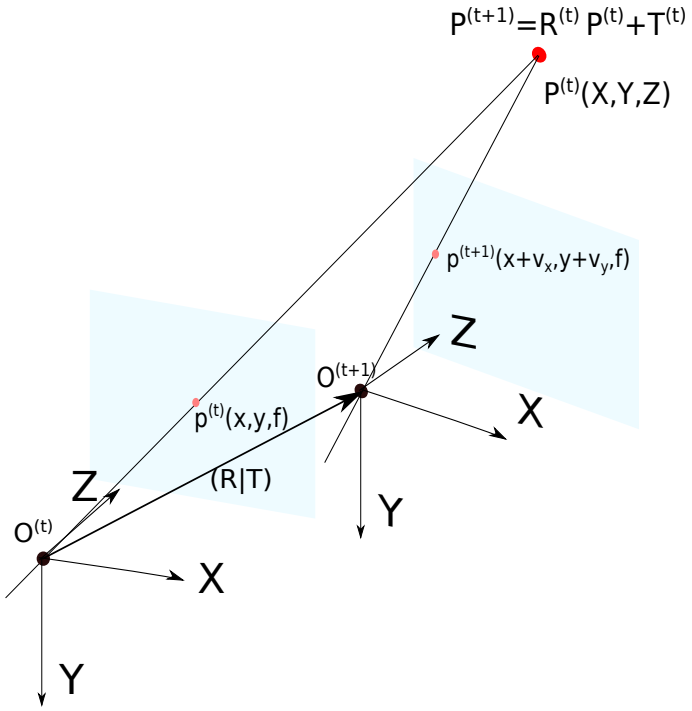


Figure 6.1: The camera's movement is described by a rotation vector ω and a translation vector $\mathbf{T}$. It moves the location of a point $\mathbf{P}$ in the camera's coordinate frame and its projection on the image plane. The vector $\mathbf{v}$ between the two image points is determined according to equations 6.2 and 6.3.

velocity is computed as the difference of consecutive camera positions:

$$\mathbf{V} = \mathbf{P}^{(t+1)} - \mathbf{P}^{(t)} = \mathbf{R}^{(t)} \times \mathbf{P}^{(t)} + \mathbf{T}^{(t)} - \mathbf{P}^{(t)} \tag{6.1}$$

An arbitrary rotation can be decomposed into three consecutive rotations around the coordinate system axes. Here, we will work with the Tait-Bryan angles, where the rotation matrix is the composition of rotations around z-, y-, and x-axis, in that order. This sequence is often used in aviation and also used by our navigation solution system (Applanix POS LV 220), which will serve as ground truth in the experiments. The multiplication of three rotation matrices $\mathbf{R} = \mathbf{R_x} \cdot \mathbf{R_y} \cdot \mathbf{R_z}$ yields:

$$\mathbf{R} = \begin{bmatrix} c\omega_y c\omega_z & c\omega_y s\omega_z & -s\omega_y \\ s\omega_x s\omega_y c\omega_z - c\omega_x s\omega_z & s\omega_x s\omega_y s\omega_z + c\omega_x c\omega_z & c\omega_y s\omega_x \\ c\omega_x s\omega_y c\omega_z + s\omega_x s\omega_z & c\omega_x s\omega_y s\omega_z - s\omega_x c\omega_z & c\omega_y c\omega_x \end{bmatrix}$$

With the small angle approximation ($cos\alpha \approx 1$, $sin\alpha \approx \alpha$, $sin\alpha \sin\beta \approx 0$)

we get a linear function

$$L\{\mathbf{R}(\omega_x, \omega_y, \omega_z)\} = \begin{bmatrix} 1 & \omega_z & -\omega_y \\ -\omega_z & 1 & \omega_x \\ \omega_y & -\omega_x & 1 \end{bmatrix} = \begin{bmatrix} 0 & \omega_z & -\omega_y \\ -\omega_z & 0 & \omega_x \\ \omega_y & -\omega_x & 0 \end{bmatrix} + \mathbf{R} = \mathbf{S}_\omega + \mathbf{I}$$

which can be written as the sum of the skew matrix $\mathbf{S}_\omega$ and the Identity $\mathbf{I}$. Replacing $\mathbf{R}$ with $\mathbf{S}_\omega + \mathbf{I}$ equation 6.1 becomes:

$$\begin{aligned} \mathbf{V} &= (\mathbf{S}_\omega + \mathbf{I}) \cdot \mathbf{P} + \mathbf{T} - \mathbf{P} \\ &= \mathbf{S}_\omega \cdot \mathbf{P} + \mathbf{T} \\ &= \mathbf{S}_\omega \times \mathbf{P} + \mathbf{T} \end{aligned}$$

Now we want to define the pixel velocity $\mathbf{v} = \left(\begin{smallmatrix} v_x \\ v_y \end{smallmatrix}\right)$ of a pixel $\mathbf{p} = \left(\begin{smallmatrix} x \\ y \end{smallmatrix}\right)$, which is a projection of the 3D point $\mathbf{P} = \left(\begin{smallmatrix} X \\ Y \\ Z \end{smallmatrix}\right)$. Under central projection they are related by the equation

$$\mathbf{p} = \frac{f \cdot \mathbf{P}}{Z}$$

The pixel velocity $\mathbf{v}$ is defined as first time derivative of $\mathbf{p}$:

$$\mathbf{v} = \frac{d\mathbf{p}}{dt} = \frac{d\frac{f \cdot \mathbf{P}}{Z}}{dt}$$

Using the quotient rule for derivatives and $\frac{d\mathbf{P}}{dt} = \mathbf{V}$ we get:

$$\mathbf{v} = f \cdot \frac{\mathbf{V} \cdot Z - \mathbf{P} \cdot V_z}{Z^2} = f \cdot \frac{\mathbf{V}}{Z} - \frac{\mathbf{P} \cdot V_z}{Z}$$

The components of $\mathbf{v}$ are

$$v_x = f \cdot \frac{V_x}{Z} - x \cdot \frac{V_z}{Z}$$

$$v_y = f \cdot \frac{V_y}{Z} - y \cdot \frac{V_z}{Z}$$

where the components of $\mathbf{V}$ can be substituted by

$$V_X = T_x - \omega_y \cdot Z + \omega_z \cdot Y$$

$$V_Y = T_y + \omega_x \cdot Z - \omega_z \cdot X$$

$$V_Z = T_z - \omega_x \cdot Y + \omega_y \cdot X$$

which yields

$$v_x = \frac{T_x \cdot f - T_z \cdot x}{Z} - \omega_y \cdot f + \omega_z \cdot y + \frac{\omega_x \cdot x \cdot y}{f} - \frac{\omega_y \cdot x^2}{f} \tag{6.2}$$

and

$$v_y = \frac{T_y \cdot f - T_z \cdot y}{Z} + \omega_x \cdot f - \omega_z \cdot x - \frac{\omega_y \cdot x \cdot y}{f} + \frac{\omega_x \cdot y^2}{f} \tag{6.3}$$

The image velocity vector $\mathbf{v}$ consists of translation terms, which are inversely proportional to depth Z, and of rotational terms, which are independent of depth Z. Under pure rotation all image points will move with the same speed. In contrast, the projection of a translation depends on an object's distance. Close objects appear to move faster than far away objects.

6.2 Motion Field Reconstruction from Optical Flow and Disparity Maps

6.2.1 Motion Perception Through Optical Flow

The motion field's projection onto the image is measurable by the optical flow, which is defined as the intensity change of pixels over time. Optical flow relies on the brightness-constancy assumption

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t),$$

which postulates that the projected intensity of moving objects is the same on consecutive frames. This a reasonable simplifications for small time-intervals because lighting conditions do not change much in short time.

As described in Chapter 3 the optical flow implementation from OpenCV has been used to compute sparse optical flow on key-point descriptors. The optical flow is augmented by the stereo depth measurements, reconstructing a sparse version of the 3D motion field.

6.2.2 Depth Perception with Disparity Maps

The stereo camera's FPGA computes disparity maps on rectified images. The rectification transformation establishes the standard stereo setup as described in Chapter 4 where epipolar lines are horizontally aligned, such that corresponding points have the same y coordinate. This camera setup, as shown in Fig. 6.2, projects an observed point P to image coordinates which only differ by a horizontal offset, the disparity d. The difference between the left camera's projection

$$x = \frac{fX}{Z}$$

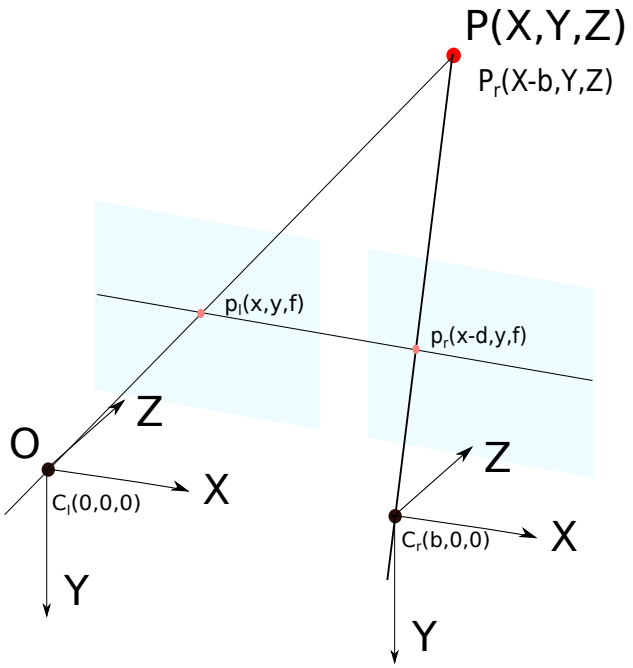


Figure 6.2: The standard stereo camera setup relates a pixel disparity d in the image space to a baseline disparity b of the stereo rig. This canonical setup yields the simple equation: $Z = \frac{fb}{d}$

and the right camera's projection

$$x - d = \frac{f(X - b)}{Z}$$

yield a simple relationship between d and Z:

$$d = \frac{f \cdot b}{Z}.$$

The stereo depth measurements serve two purposes in the ego-motion algorithm. They are used to detect flow vectors that belong to distant points, whose projection generate pure rotational flow in the image. And secondly, motion flow of close-range points is used to estimated the vehicle's velocity.

6.3 The Ego-Motion Algorithm

Fig. 6.3 shows an overview of the proposed algorithm. It uses stereo depth measurements to identify flow vectors that were caused only by rotation of

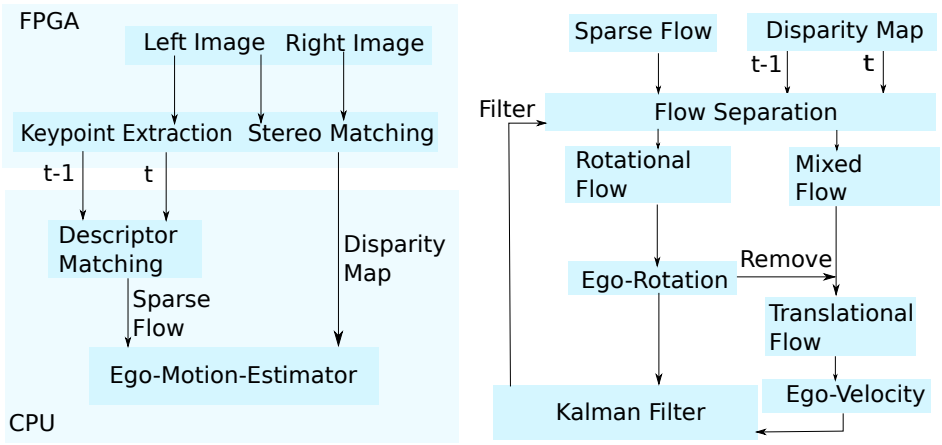


Figure 6.3: Left: Overview of the Algorithm. Right: Detailed View of the Ego-Motion Estimator.

the camera. If we recall the equations for $\mathbf{v}$ (Eq. 6.2 and 6.3), we see that for large distances the translation term becomes very small. Hence, we can assume that optical flow of distant points is only caused by rotation of the camera. Once the ego-rotation has been computed, it can be subtracted from the 3D flow vector field, leaving only the 3D translation component. On this vector field the ego-translation is estimated as the solution which minimizes the error between projected 3D flow end points and the optical flow end points.

The following subsections provides details of the crucial steps of the algorithm.

6.3.1 Separation of Rotational Flow Components

As stated in equations 6.2 and 6.3, optical flow induced by rotation is independent of the distance to an observed object, while the translation component is inversely proportional to depth. With the help of disparity maps, we have the distance for all pixels available. Any points further away than 60 m are considered as bearing only rotational flow and used for the next processing step. For robustness reasons, the disparity of both vector end points is checked to have similar depth.
Simplifying the equations 6.2 and 6.3 to the pure rotational case yields:

$$v_x = -\omega_y \cdot f + \omega_z \cdot y + \frac{\omega_x \cdot x \cdot y}{f} - \frac{\omega_y \cdot x^2}{f}$$

and

$$v_y = \omega_x \cdot f - \omega_z \cdot x - \frac{\omega_y \cdot x \cdot y}{f} + \frac{\omega_x \cdot y^2}{f}$$

which generates two linear equations per flow vector. The equations for all available flow vectors are stacked to an over-determined equation system and solved for $[\omega_x, \omega_y, \omega_z,]$ in a least squares manner.

6.3.2 Velocity Computation

Once the rotation vector ω is known, all 3D flow vector start points are rotated accordingly. As a result, only the translation part of the 3D flow remains. Ideally, all vectors are now identical but due to noise in the optical flow, an inaccurate estimate of the rotation matrix, or independently moving objects the vectors are usually very different. Therefore, all available 3D translation vectors are evaluated as candidates. First, the start points of the 3D flow is rotated and translated by the transformation to be evaluated. The resulting 3D points are then projected to the camera image and compared to the end point of the optical flow vectors. The sum of distances serves as error metric and the translation with the lowest overall error is finally chosen as best estimate of the vehicles inter-frame movement.

6.3.3 Velocity Smoothing with a Kalman Filter

The frame-by-frame velocity estimates are then smoothed with a Kalman filter. Due to inertia the velocity cannot change abruptly and the ego-motion state must be updated accordingly. The Kalman filter's transition matrices are set to the identity matrix making no assumptions about the intentions of the driver. The state vector consists of three linear and three angular velocity components. The measurement noise matrix is set to an identity matrix scaled according to the relative end point error. It is defined as the ratio of the average end point error (computed by the algorithm) and the average length of optical flow vectors.

6.4 Experimental Evaluation

Several experiments have been conducted which compare the estimated linear and angular velocities generated by the proposed algorithm against the ground truth provided by the Applanix POS LV 220 system installed on our vehicle. The test drive was a round track where the car drove up to 50 km/h

(circa 17 m/s) and took several U-turns. Table 6.1 presents the mean standard deviations and percentile distribution of the absolute tracking error for the estimated linear and angular velocities. Figures figs. 6.4 to 6.8 show the results for one minute of video compared against the ground truth from the Applanix POS LV 220.
The Pearson correlation coefficient serves as similarity measure to compare the sequence of estimates with the sequence of ground truth values. The Pearson coefficient P for two discrete functions A and B is defined as:

$$P(A,B) = \frac{covar(A,B)}{\sqrt{var(A)} \cdot \sqrt{var(B)}}$$

where

$$\begin{aligned} var(X) =& \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu(X))^2 \\ \mu(X) =& \frac{1}{N}\sum_{i=1}^{N} x_i \\ covar(X,Y) =& \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu(X)) \cdot (y_i - \mu(Y)) \end{aligned}$$

For a car driving on a almost-planar world, the three most important numbers are the x- and y-component of the velocity and the yaw rate. According to the Pearson correlation, the presented algorithm can track the true values of yaw rate and forward speed with very good accuracy. The lateral velocity component was estimated with reasonable accuracy, while the correlation coefficents for roll and pitch are fairly low. Most likely this is caused by high frequent changes of roll and pitch due to the vehicle's shock absorbers. It is fairly reasonable to assume that the estimation will improve with a higher frame rate of the camera. The variance of the estimated velocity increases significantly with the speed of the vehicle. The main reason for that is that the stereo camera estimation have the highest accuracy at close range, but points at close distance may disappear on consecutive frames and thus no optical flow can be reliably computed. Again, a higher frame rate of the camera could mitigate this problem.

Video Duration [s]	long. speed	lat. speed	yaw rate	roll	pitch
460	0.998	0.896	0.993	0.547	0.664
325	0.994	0.878	0.987	0.225	0.318
667	0.998	0.896	0.992	0.384	0.385

Table 6.1: Pearson correlation coefficients for longitudinal and lateral speed, yaw rate, roll and pitch on three different test drives.

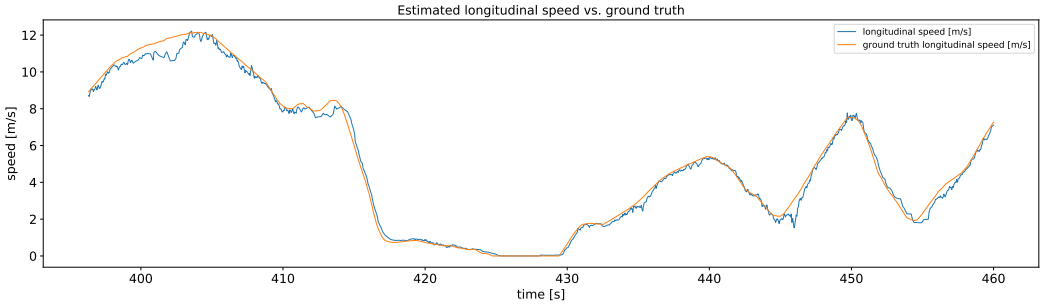


Figure 6.4: The estimated forward component of the vehicle's velocity (blue) compared to the ground truth from the Applanix POS LV 200 (orange).

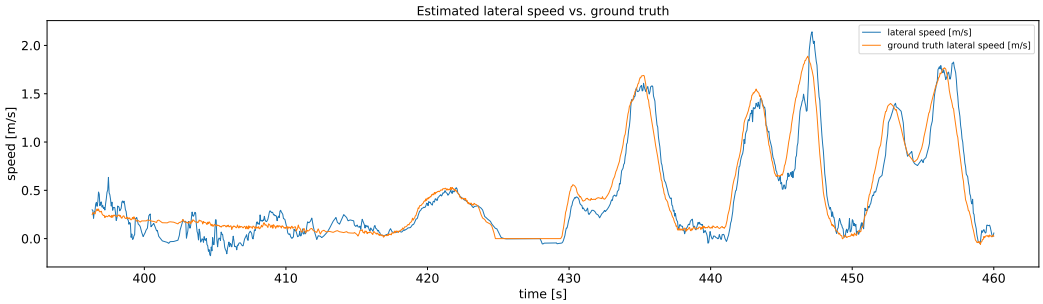


Figure 6.5: The estimated lateral component of the vehicle's velocity (blue) compared to the ground truth from the Applanix POS LV 200 (orange).

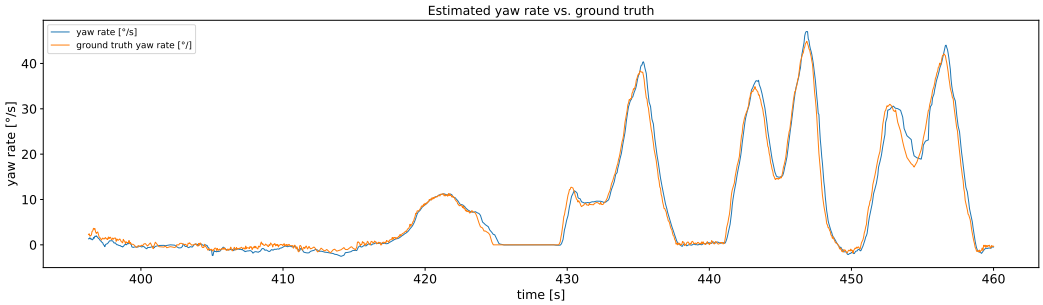


Figure 6.6: The estimated vehicle's yaw rate (blue) compared to the ground truth from the Applanix POS LV 200 (orange).

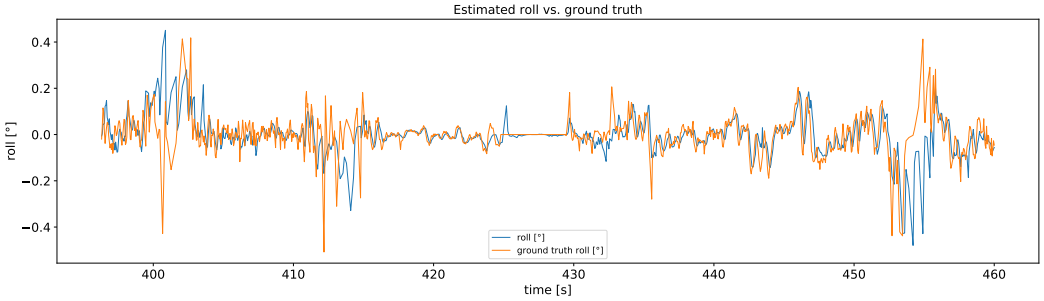


Figure 6.7: The estimated vehicle's roll rate (blue) compared to the ground truth from the Applanix POS LV 200 (orange). High frequency oscillation are not accurately estimated.

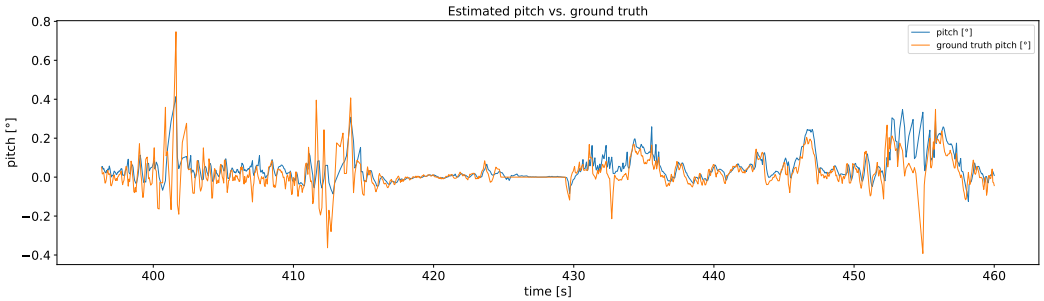


Figure 6.8: The estimated vehicle's pitch rate (blue) compared to the ground truth from the Applanix POS LV 200 (orange). High frequency oscillation are not accurately estimated.

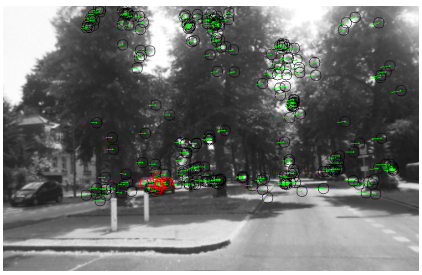
6.5 Applications

A useful application of my algorithm is fault detection. The camera might become de-calibrated after an accident, or by material exhaustion. The presented algorithm provides very good estimates of the speed and the yaw rate which can be compared to vehicle's on-board sensors, e.g. wheel odometry. Measurement disagreement can indicate that either of the sensors is faulty.

A second use-case could be the fusion of the visual estimates with an IMU and/or an odometer. The ground truth odometry is provided by the expensive Applanix POS LV 220 system which fuses the information from GPS, wheel odometry and an IMU. Combining the results of my algorithm with a cheaper (and less accurate) wheel odometry or IMU should yield good results.

And finally, the estimated ego-motion can be applied to the obstacle detection task. It can help to detect regions in the image, where the optical flow cannot be explained by the ego-motion flow. The detection of independently moving objects is useful to detect other vehicles approaching the lane on which our car plans to drive.

Fig. 6.9 shows two examples. On the left image the segmentation indicates an approaching vehicle which appears only shortly between a row of trees. The right image shows how the flow segmentation distinguishes the moving cars on the intersection from the waiting car on the left.



(a) The approaching vehicle perturbs the ego-motion flow.

(b) Optical flow separates stopped cars from moving cars.

Figure 6.9: Optical flow vectors are green. Optical flow which cannot be explained by the estimated ego-motion is marked with red circles.

6.6 Conclusions

The presented algorithm uses stereo depth measurements to identify optical flow belonging to far away 3D points. According to the equations of central projection this type of optical flow is almost entirely caused by rotational movement of the camera. This separation of optical flow is used to separately estimate the rotation and the translation part of the camera's movement between consecutive frames. The experimental results show accurate estimates for longitudinal speed and yaw rate, and reasonably good estimates for the lateral speed. Pitch and roll estimates were very noisy due to high frequency oscillations.

Chapter 7

Conclusions

7.1 Summary of Contributions

In this thesis I have presented:

- A real-time online calibration method to determine the rotation matrix of cameras mounted on a vehicle.
- A 3D obstacle detection method based on an embedded stereo camera developed by our research group.
- A traffic light detection system using two separate monocular cameras and a digital map with annotated global traffic light positions.
- An application of my obstacle detection and traffic-light detection to the task of traffic light mapping.
- A method to estimate the linear and angular velocity of the ego-vehicle, using stereoscopic vision and optical flow.

My calibration method is able to calibrate from scratch or iteratively refine the rotation parameters of a vehicle-mounted camera. Different from standard methods it does not rely on a dedicated calibration target. Instead, it uses the optical flow computed on images during a test drive. The method exploits the fact that a car must obey movement restrictions which results in distinctive optical flow patterns. The flow converges in epipolar points whose distribution is used to recover the camera's rotation matrix with respect to the car frame. The experimental evaluation on several differently mounted camera's showed robust calibration results with diminishing variance and good convergence properties.

My obstacle detection framework evaluates the depth measurements from an embedded stereo camera and generates a representation of obstacles and free space. It performs a hybrid 2D/3D segmentation, which enforces heuristic constraints defined in image space as well as in 3D space. The evaluation against a LIDAR-based occupancy grid showed high congruence of the drivable space border, but also identified limitations caused by the stereo camera's functional principle. The experimental results showed that accuracy drops of significantly beyond 30 meters distance, indicating that the stereo camera cannot function as a drop-in replacement of LIDAR sensors.
My traffic light detection system relies on a two-camera setup covering the left and right hemisphere in front of the car to detect oncoming traffic lights in real-time. It relies on the self-localization and planned trajectories of the car to retrieve the global positions of relevant traffic lights. The global positions are mapped to regions in the image which benefits execution time as well as the detection accuracy. The detection system was successfully demonstrated to the public (e.g. "Mission Brandenburg Gate"[1]) or in competitions (1st place Dubai World Challenge for Self-Driving Transport 2019[2]). The experimental evaluation showed high true-positive rates and detection ranges of 60 m on average which is decent given the resolution of the cameras.
Combining the obstacle detection with the traffic light detection, I have used detected objects resembling the shape of traffic lights as an input to the traffic light detection system. It projects these objects to regions of interest to be searched for traffic lights. The experimental evaluation showed that the effective detection range was around 25 meters. Traffic light poles were hard to detect at larger distances because depth boundaries of far away objects are not distinct enough for a robust segmentation. However, this range proved enough to verify the existence of a traffic light (by detection) and to robustly estimate the global position.
My last contribution was the estimation of ego-motion purely from visual input provided by the embedded stereo camera. Combining disparity maps and optical flow image interest points were tracked in time and space in order to estimate the vehicle's movement. The main challenge was the separation of static and dynamic environment. The latter causes motion flow inconsistent to the ego-motion flow. Here, I implemented my novel idea to separate the optical flow of points by their 3D distance. Far away points barely contribute to the optical flow. They provide a reliable way to estimate the inter-frame rotation of the camera. Removing the rotational component from the flow field drastically simplifies the estimation of the linear velocity. This two-step

[1] `https://autonomos.inf.fu-berlin.de/media/videos/`
[2] `https://sdcongress.com/finalists`

approach yielded linear velocity and yaw rate comparable to the measurements of the Applanix reference localization system, albeit variance and noise were higher. However, the strong correlation provides a good tool to determine a faulty camera calibration, or if it is deemed correct a faulty calibration of the reference sensor.

Chapter 8

Appendix

8.1 Pseudo Code for Extrinsic Camera Calibration

Algorithm 1 Compute homogeneous line for two points

Require: $p \in \mathbb{R}^2, q \in \mathbb{R}^2$
Ensure: $l \in \mathbb{R}^3 \wedge l_x^2 + l_y^2 = 1$
$l \leftarrow \begin{pmatrix} p_x \\ p_y \\ 1 \end{pmatrix} \times \begin{pmatrix} q_x \\ q_y \\ 1 \end{pmatrix}$
$l \leftarrow \frac{l}{\sqrt{{l_x}^2 + {l_y}^2}}$ // make the normal vector of the line unit length

Algorithm 2 Compute Epipole e from Optical Flow with RANSAC

Require: $P_1 \in \mathbb{R}^{N\times 2}, P_2 \in \mathbb{R}^{N\times 2}, n \in \mathbb{N}_+, K \in \mathbb{N}_+$ (RANSAC iterations)
Ensure: $e \in \mathbb{R}^2 \wedge |I| \geq n$ (inlier set I)

```
U ← ∅ // Used flow vector pairs
e = (∞ ∞)
I = ∅ // inliers for e
for i = 1 to K do
  repeat
    a, b ← random indices ∈ [1...N]
    v1 ← P2[a] − P1[a]
    v2 ← P2[b] − P1[b]
  until a ≠ b ∧ (a, b) ∉ U ∧ ∠(v1, v2) > 0.5° // non-parallel lines
  U ← U ∪ {(a, b), (b, a)}
  l1 ← constructLine(P1[a], P2[a])
  l2 ← constructLine(P1[b], P2[b])
  x ← l1 × l2 // homogeneous intersection point of l1 and l2
  x ← x / ||xz||
  J ← ∅ // inliers for x
  for j = 1 to N do
    l ← constructLine(P1[j], P2[j])
    m ← constructLine(P1[j], x)
    if |lx · mx + ly · my| > cos(0.5°) then
      // less than 0.5° deviation
      J ← J ∪ {j}
    end if
  end for
  if |J| ≥ max(n, |I|) then
    e ← x
    I ← J
  end if
end for
e ← refit using all inliers
```

Algorithm 3 Compute Force f and Angular Momentum m for a Point

Require: $p \in \mathbb{R}^2, k \in \mathbb{R}, D \in \mathbb{R}$
Ensure: $f = force(p, k)$
  if $||p|| > D$ **then**
    $f \leftarrow \binom{0}{0}$ // ignore outliers
  else
    $s = \exp(\frac{-|p_x|}{k}) + \exp(\frac{-|p_y|}{k})$
    $w_1 \leftarrow \frac{\exp(\frac{-|p_x|}{k})}{s}$
    $w_2 \leftarrow \frac{\exp(\frac{-|p_y|}{k})}{s}$ // $w_1 + w_2 = 1$
    $f \leftarrow \begin{pmatrix} -w_1 \cdot p_x \\ -w_2 \cdot p_y \end{pmatrix}$
    $r = \sqrt{||p - c||}$ // vector from center of point cloud to p
    $M = r_x \cdot f_y - r_y \cdot f_x$
    // shorten length to square root
    $f \leftarrow \frac{f}{\sqrt{||f||}}$
    $M \leftarrow \frac{M}{\sqrt{|M|}}$
  end if

Algorithm 4 Compute Forward Epipole e and Horizon Angle θ

Require: $P \in \mathbb{R}^{N\times 2}, N \in \mathbb{R}, k \in \mathbb{R}, D \in \mathbb{R}$
Ensure: $e \in \mathbb{R}^2, \theta \in \mathbb{R}$
  $c \leftarrow \text{median}(P)$ // or init T with solution from previous iteration
  $T \leftarrow \begin{pmatrix} 1 & 0 & -c_x \\ 0 & 1 & -c_y \\ 0 & 0 & 1 \end{pmatrix}$
  $P \leftarrow T \cdot P$ // subtract median
  $c \leftarrow \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ // keeps track of current center of P
  for $i = 1$ to N **do**
    $f \leftarrow \begin{pmatrix} 0 \\ 0 \end{pmatrix}$
    $m \leftarrow 0$
    for $p \in P$ **do**
      $f \leftarrow f + force(p, ...)$
      $m \leftarrow m + momentum(p, ...)$
    end for
    $t \leftarrow \text{computeTranslation}(f, ...)$ // proportional to the average force
    $\phi \leftarrow \text{computeAngle}(m, ...)$ // proportional to the average momentum
    $T_i \leftarrow \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & c_x \\ 0 & 1 & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -c_x \\ 0 & 1 & -c_y \\ 0 & 0 & 1 \end{pmatrix}$
    // rotate around c, then translate by t
    $P \leftarrow T_i \cdot P$
    $c \leftarrow T \cdot c$
  end for
  $U \leftarrow T^{-1}$
  $e \leftarrow \begin{pmatrix} U_{02} \\ U_{12} \end{pmatrix}$
  $\theta \leftarrow atan2(U_{10}, U_{00})$

Algorithm 5 Compute the Camera's Extrinsic Rotation Matrix R

Require: $e \in \mathbb{R}^2, \theta \in \mathbb{R}, P \in \mathbb{R}^{3\times4}$(Camera's Projection Matrix)
Ensure: $R \in \mathbb{R}^{3\times3}$

$d_x = P^{-1} \cdot \begin{pmatrix} e_x \\ e_y \\ 1 \end{pmatrix}$ // reproject the epipole to the 3D direction vector
$d_x = \frac{d_x}{||d_x||}$ // car's forward direction (x) in camera coordinates
$h = \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix}$
$d' = P^{-1} \cdot \begin{pmatrix} e_x - h_x \\ e_y - h_y \\ 1 \end{pmatrix}$ // $\begin{pmatrix} e_x - h_x \\ e_y - h_y \end{pmatrix}$ is a point on the horizon left to e
$d' = \frac{d'}{||d'||}$
$d_z = d_x \times d'$
$d_z = \frac{d_z}{||d_z||}$ // car's up direction (z) in camera coordinates
$d_y = d_z \times d_x$ // car's sideway direction (y) in camera coordinates
$R = \begin{pmatrix} {d_x}^T \\ {d_y}^T \\ {d_z}^T \end{pmatrix}$ // R rotates from the camera's to car's coordinate frame

8.2 Pseudo Code for Obstacle Detection on Disparity Maps

Algorithm 6 Segment Vertical Stripes on 3D Image

Require: $D : \mathbb{N}^2 \to \mathbb{R}^3, V \in \mathbb{R}^3$ // 3D image D, Volume of Interest V
Require: $g \in \mathbb{N}, d \in \mathbb{R}$ // maximal pixel gap g, maximal depth gap d
Ensure: S

```
S ← [ ] // list of stripes
M, N ← dim(D) // (rows, columns) of 3D image D
for u = 1 to N do
  s ← [ ] // list of stripe points (u, v, x, y, z)
  for v = 1 to M do
    (x, y, z) ← D(u, v)
    if D(u, v) ∈ V then
      if |s| = 0 then
        // start a new stripe
        s ← s ⊕ (u, v, x, y, z)
      else
        if v − max_v(s) > g ∨ |D(u, v) − min_x(s)| > d ∨ v = M then
          // finish stripe
          S ← S ⊕ [s]
          s ← [ ]
        else
          // continue stripe
          s ← s ⊕ (u, v, x, y, z)
        end if
      end if
    end if
  end for
end for
```

Algorithm 7 Adjacency relation for vertical stripes

Require: $a, b \in \mathbb{S}^2, d \in \mathbb{R}^3, h \in \mathbb{R}, w \in \mathbb{N}$ // two vertical stripes a, b, max. 3D deviation d, max. 3D height gap h, max. image column gap w
Ensure: $c =$ **true** if a and b are adjacent, else $c =$ **false**
$\Delta \leftarrow |mean_{xyz}(a) - mean_{xyz}(b)|$
$z_1 \leftarrow \max_z(a)$
$z_2 \leftarrow \max_z(b)$
$c \leftarrow |a_u - b_u| < w \wedge \Delta_x < d_x \wedge \Delta_y < d_y \wedge \Delta_y < d_y \wedge |z_1 - z_2| < h$

Algorithm 8 Merge neighbored vertical stripes

Require: $S, t \in \mathbb{N}$ // Vertical stripes S
Ensure: O // obstacles O
$V \leftarrow S$ // Vertices: vertical stripes
$E \leftarrow \{(a, b) \in V^2 \,|\, \text{adjacent}(a, b, ...)\}$ // Edges: defined by Alg. 7
$G \leftarrow (V, E)$ // a graph
$O \leftarrow$ Connected-Components(G) // see Chapter 21 of [60]

8.3 Pseudo Code for Traffic Light Detection

Algorithm 9 Cluster color-thresholded points to light blobs

Require: $P \in \mathbb{N}_{n \times 2}$ // Pixels of on color (red/yellow/green)
Ensure: B // list of circular shaped point sets
  $T \leftarrow$ splitPoints(P, 100) // split points P in small subsets (at least 100 points and two image rows)
  $U \leftarrow [\,]$ // clusters created from subsets
  for t in T **do**
    $V \leftarrow P$ // Vertices: points
    $E \leftarrow \{(a, b) \in V^2 \mid |a_x - b_x| \leq m \wedge |\,|a_y - b_y| \leq m\}$ // two points adjacent if within L1 range m
    $G \leftarrow (V, E)$ // a graph
    $u \leftarrow$ Connected-Components(G) // see Chapter 21 of [60]
    $U \leftarrow U \oplus [u]$
  end for
  $V \leftarrow U$ // Vertices: subset clusters
  $E \leftarrow \{(a, b) \in V^2 \mid$ boundingBox(a) touches boundingBox(b)$\}$
  $G \leftarrow (V, E)$ // a graph
  $B \leftarrow$ Connected-Components(G) // see Chapter 21 of [60]
  $B \leftarrow$ apply heuristic filters on B // removes clusters with wrong shape/position (see Subsection 5.4)

Algorithm 10 Construct traffic lights from colored pixel clusters

Require: R, Y, G // red, yellow, green pixel clusters
Ensure: T // list of traffic light candidates
  for c in $R \cup Y \cup G$ **do**
    $g \leftarrow$ area of unlit signals
    if isDark(g) **then**
      $T \leftarrow c \oplus g$ // add traffic light: colored pixel cluster + dark area
    end if
  end for
  $S \leftarrow \{(a, b) | a \in R, b \in Y,$ correct relative position$\}$ // set of red/yellow cluster which could make one traffic light
  for $c_r, c_y in S$ **do**
    $g \leftarrow$ area of unlit green signal
    if isDark(g) **then**
      $T \leftarrow c_r \oplus c_y \oplus g$ // add red yellow traffic light: red cluster + yellow clusters + dark area
    end if
  end for

8.4 Pseudo Code for Ego-Motion Estimation

Algorithm 11 Compute rigid motion from stereo-flow

Require: O, P // optical flow O, 3D flow P
Ensure: T // rigid 4×4 transformation T
  $O_f, P_f \leftarrow$ 2d/3D flow of points at least 60 meters away
  $A, b \leftarrow$ motion field equations with points from O_f, P_f // the equations are derived in Subsection. 6.3.1
  $\omega \leftarrow SVD(A, b)$
  $P_t \leftarrow P \ominus \omega$ // remove rotational component from 3D flow
  $e \leftarrow \infty$
  for t in P_t **do**
    $T_i \leftarrow \omega \oplus t$
    $p_2 \leftarrow P \cdot (T_i \cdot P_1)$ // transform 3D points from frame 1, project them to camera frame 2
    $e_i \leftarrow$ endpoint error of input optical flow vs. p_2
    if $e_i < e$ **then**
      $e \leftarrow e_i$
      $T \leftarrow T_i$
    end if
  end for

Bibliography

[1] "Technical documentation for the applanix pos lv 220." https://www.applanix.com/products/poslv.htm. Accessed: 2019-12-09.

[2] "Schematical figure of madeingermany." http://autonomos.inf.fu-berlin.de/wp-content/uploads/mig-bezeichnungen.jpg. Accessed: 2019-12-09.

[3] "Technical documentation for the ibeo lux laser scanner." https://www.ibeo-as.com/en/produkte/sensoren/IbeoLUX. Accessed: 2019-12-09.

[4] "Technical documentation for the velodyne hdl-64e s2 laser scanner." https://www.velodynelidar.com/hdl-64e.html. Accessed: 2019-12-09.

[5] "Technical documentation for the allied vision guppy f036-c camera." https://www.alliedvision.com/en/products/cameras/detail/Guppy/F-036.html. Accessed: 2019-12-09.

[6] R. Spangenberg, T. Langner, and R. Rojas, "Weighted semi-global matching and center-symmetric census transform for robust driver assistance," in *International Conference on Computer Analysis of Images and Patterns*, pp. 34–41, Springer, 2013.

[7] B. Fischer, *Embedded Vision-based Environment Perception and Control for Mobile Robotics*. Der Andere Verlag, 2016.

[8] R. Cipolla, T. Drummond, and D. P. Robertson, "Camera calibration from vanishing points in image of architectural scenes.," in *BMVC*, vol. 99, pp. 382–391, 1999.

[9] Z. Zhang, M. Li, K. Huang, and T. Tan, "Practical camera auto-calibration based on object appearance and motion for traffic scene visual surveillance," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1–8, IEEE, 2008.

[10] S. Hold, S. Gormer, A. Kummert, M. Meuter, and S. Muller-Schneiders, "A novel approach for the online initial calibration of extrinsic parameters for a car-mounted camera," in *Intelligent Transportation Systems, 2009. ITSC'09. 12th International IEEE Conference on*, pp. 1–6, IEEE, 2009.

[11] M. B. de Paula, C. R. Jung, and L. da Silveira Jr, "Automatic on-the-fly extrinsic camera calibration of onboard vehicular cameras," *Expert Systems with Applications*, vol. 41, no. 4, pp. 1997–2007, 2014.

[12] M. Hwangbo, J.-S. Kim, and T. Kanade, "Gyro-aided feature tracking for a moving camera: fusion, auto-calibration and GPU implementation," *The International Journal of Robotics Research*, p. 0278364911416391, 2011.

[13] M. Miksch, B. Yang, and K. Zimmermann, "Homography-based extrinsic self-calibration for cameras in automotive applications," in *Workshop on Intelligent Transportation*, pp. 17–22, 2010.

[14] T. Ruland, H. Loose, T. Pajdla, and L. Krüger, "Hand-eye autocalibration of camera positions on vehicles," in *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pp. 367–372, IEEE, 2010.

[15] E. Tapia and R. Rojas, "A note on calibration of video cameras for autonomous vehicles with optical flow," 2013.

[16] L. Heng, B. Li, and M. Pollefeys, "Camodocal: Automatic intrinsic and extrinsic calibration of a rig with multiple generic cameras and odometry," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pp. 1793–1800, IEEE, 2013.

[17] R. Spangenberg, T. Langner, and R. Rojas, "On-line stereo self-calibration through minimization of matching costs," in *Scandinavian Conference on Image Analysis*, pp. 545–554, Springer, 2013.

[18] E. Rehder, C. Kinzig, P. Bender, and M. Lauer, "Online stereo camera calibration from scratch," in *Intelligent Vehicles Symposium (IV), 2017 IEEE*, pp. 1694–1699, IEEE, 2017.

[19] M. Perrollaz, A. Spalanzani, and D. Aubert, "Probabilistic representation of the uncertainty of stereo-vision and application to obstacle detection," in *Intelligent Vehicles Symposium (IV), 2010 IEEE*, pp. 313–318, IEEE, 2010.

[20] A. Wedel, H. Badino, C. Rabe, H. Loose, U. Franke, and D. Cremers, "B-spline modeling of road surfaces with an application to free-space estimation," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 10, no. 4, pp. 572–583, 2009.

[21] F. Oniga and S. Nedevschi, "Processing dense stereo data using elevation maps: Road surface, traffic isle, and obstacle detection," *Vehicular Technology, IEEE Transactions on*, vol. 59, no. 3, pp. 1172–1182, 2010.

[22] K. Schauwecker and A. Zell, "Robust and efficient volumetric occupancy mapping with an application to stereo vision," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 6102–6107, IEEE, 2014.

[23] Y. Li and Y. Ruichek, "Occupancy grid mapping in urban environments from a moving on-board stereo-vision system," *Sensors*, vol. 14, no. 6, pp. 10454–10478, 2014.

[24] P. Marın-Plaza, J. Beltrán, A. Hussein, B. Musleh, D. Martın, A. de la Escalera, and J. M. Armingol, "Stereo vision-based local occupancy grid map for autonomous navigation in ros," in *11th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications VISIGRAPP*, vol. 2016, 2016.

[25] C. Yu, V. Cherfaoui, and P. Bonnifait, "Evidential occupancy grid mapping with stereo-vision," in *Intelligent Vehicles Symposium (IV), 2015 IEEE*, pp. 712–717, IEEE, 2015.

[26] M. Cordts, T. Rehfeld, L. Schneider, D. Pfeiffer, M. Enzweiler, S. Roth, M. Pollefeys, and U. Franke, "The stixel world: A medium-level representation of traffic scenes," *Image and Vision Computing*, vol. 68, pp. 40–52, 2017.

[27] H. Harms, E. Rehder, and M. Lauer, "Grid map based free-space estimation using stereovision," in *Proc. Workshop Environment Perception Automated On-road Vehicles, IEEE Intelligent Vehicles Symposium*, 2015.

[28] H. Badino, R. Mester, T. Vaudrey, U. Franke, and A. Daimler, "Stereo-based free space computation in complex traffic scenarios," in *Image Analysis and Interpretation, 2008. SSIAI 2008. IEEE Southwest Symposium on*, pp. 189–192, IEEE, 2008.

[29] M. Diaz-Cabrera, P. Cerri, and P. Medici, "Robust real-time traffic light detection and distance estimation using a single camera," *Expert Systems with Applications*, vol. 42, no. 8, pp. 3911–3923, 2015.

[30] M. Omachi and S. Omachi, "Detection of traffic light using structural information," in *Signal Processing (ICSP), 2010 IEEE 10th International Conference on*, pp. 809–812, IEEE, 2010.

[31] H. Tae-Hyun, J. In-Hak, and C. Seong-Ik, "Detection of traffic lights for vision-based car navigation system," in *Pacific-Rim Symposium on Image and Video Technology*, pp. 682–691, Springer, 2006.

[32] Z. Cai, M. Gu, and Y. Li, "Real-time arrow traffic light recognition system for intelligent vehicle," in *Proceedings of the International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV)*, p. 1, The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2012.

[33] S. Sooksatra and T. Kondo, "Red traffic light detection using fast radial symmetry transform," in *electrical engineering/electronics, computer, telecommunications and information technology (ECTI-CON), 2014 11th international conference on*, pp. 1–6, IEEE, 2014.

[34] S. Hosseinyalamdary and A. Yilmaz, "A bayesian approach to traffic light detection and mapping," *ISPRS journal of photogrammetry and remote sensing*, vol. 125, pp. 184–192, 2017.

[35] G. Siogkas, E. Skodras, and E. Dermatas, "Traffic lights detection in adverse conditions using color, symmetry and spatiotemporal information.," in *VISAPP (1)*, pp. 620–627, 2012.

[36] S. Omachi and M. Omachi, "Traffic light detection with color and edge information," in *Computer Science and Information Technology, International Conference on(ICCSIT)*, vol. 00, pp. 284–287, 08 2009.

[37] R. De Charette and F. Nashashibi, "Real time visual traffic lights recognition based on spot light detection and adaptive traffic lights templates," in *Intelligent Vehicles Symposium, 2009 IEEE*, pp. 358–363, IEEE, 2009.

[38] Y. Jie, C. Xiaomin, G. Pengfei, and X. Zhonglong, "A new traffic light detection and recognition algorithm for electronic travel aid," in *intelligent control and information processing (ICICIP), 2013 fourth international conference on*, pp. 644–648, IEEE, 2013.

[39] M. Diaz-Cabrera, P. Cerri, and J. Sanchez-Medina, "Suspended traffic lights detection and distance estimation using color features," in *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*, pp. 1315–1320, IEEE, 2012.

[40] J. Levinson, J. Askeland, J. Dolson, and S. Thrun, "Traffic light mapping, localization, and state detection for autonomous vehicles," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 5784–5791, IEEE, 2011.

[41] N. Fairfield and C. Urmson, "Traffic light mapping and detection," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 5421–5426, IEEE, 2011.

[42] T. Langner, D. Seifert, B. Fischer, D. Goehring, T. Ganjineh, and R. Rojas, "Traffic awareness driver assistance based on stereovision, eye-tracking, and head-up display," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pp. 3167–3173, IEEE, 2016.

[43] A. Fregin, J. Müller, and K. Dietmayer, "Three ways of using stereo vision for traffic light recognition," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 430–436, IEEE, 2017.

[44] H.-K. Kim, Y.-N. Shin, S.-g. Kuk, J. H. Park, and H.-Y. Jung, "Nighttime traffic light detection based on svm with geometric moment features," *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. 7, no. 4, pp. 472–475, 2013.

[45] Z. Shi, Z. Zou, and C. Zhang, "Real-time traffic light detection with adaptive background suppression filter," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 3, pp. 690–700, 2016.

[46] V. John, K. Yoneda, B. Qi, Z. Liu, and S. Mita, "Traffic light recognition in varying illumination using deep learning and saliency map," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pp. 2286–2291, IEEE, 2014.

[47] K. Behrendt, L. Novak, and R. Botros, "A deep learning approach to traffic lights: Detection, tracking, and classification," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1370–1377, IEEE, 2017.

[48] M. Weber, P. Wolf, and J. M. Zöllner, "Deeptlr: A single deep convolutional network for detection and classification of traffic lights.," in *Intelligent Vehicles Symposium*, pp. 342–348, 2016.

[49] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry for ground vehicle applications," *Journal of Field Robotics*, vol. 23, no. 1, pp. 3–20, 2006.

[50] M. Agrawal and K. Konolige, "Real-time localization in outdoor environments using stereo vision and inexpensive gps," in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, vol. 3, pp. 1063–1068, IEEE, 2006.

[51] A. Howard, "Real-time stereo visual odometry for autonomous ground vehicles," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pp. 3946–3952, IEEE, 2008.

[52] A. Talukder and L. Matthies, "Real-time detection of moving objects from moving vehicles using dense stereo and optical flow," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 4, pp. 3718–3725, IEEE, 2004.

[53] H. Badino, "A robust approach for ego-motion estimation using a mobile stereo platform," in *Complex Motion*, pp. 198–208, Springer, 2007.

[54] "Tikz source code for drawing the pinhole camera model." `https://tex.stackexchange.com/questions/96074`. Accessed: 2019-11-09.

[55] C. B. Duane, "Close-range camera calibration," *Photogramm. Eng*, vol. 37, no. 8, pp. 855–866, 1971.

[56] A. Criminisi, I. Reid, and A. Zisserman, "Single view metrology," *International Journal of Computer Vision*, vol. 40, no. 2, pp. 123–148, 2000.

[57] F. L. Markley, Y. Cheng, J. L. Crassidis, and Y. Oshman, "Averaging quaternions," *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 4, pp. 1193–1197, 2007.

[58] F. Von Hundelshausen, M. Schreiber, F. Wiesel, A. Liers, and R. Rojas, *MATRIX: A force field pattern matching method for mobile robots*. Freie Univ., Fachbereich Mathematik und Informatik, 2003.

[59] P. Fankhauser and M. Hutter, "A universal grid map library: Implementation and use case for rough terrain navigation," in *Robot Operating System (ROS)*, pp. 99–120, Springer, 2016.

[60] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Introduction to algorithms second edition," *The Knuth-Morris-Pratt Algorithm, year*, 2001.

[61] G. Welch, G. Bishop, *et al.*, "An introduction to the kalman filter," 1995.

Selbstständigkeitserklärung

Ich versichere hiermit, dass diese Arbeit von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel wie Berichte, Bücher, Internetseiten oder ähnliches sind im Literaturverzeichnis angegeben, Zitate aus fremden Arbeiten sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

Berlin, 3. Februar 2020

Tobias Langner