

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/319051868>

3D Visual Perception for Self-Driving Cars using a Multi-Camera System: Calibration, Mapping, Localization, and Obstacle Detection

Article in *Image and Vision Computing* · August 2017

DOI: 10.1016/j.imavis.2017.07.003

CITATIONS

233

READS

4,242

7 authors, including:



Christian Häne

University of California, Berkeley

61 PUBLICATIONS 3,423 CITATIONS

[SEE PROFILE](#)



Lionel Heng

DSO National Laboratories

30 PUBLICATIONS 2,970 CITATIONS

[SEE PROFILE](#)



Gim Hee Lee

National University of Singapore

192 PUBLICATIONS 10,022 CITATIONS

[SEE PROFILE](#)



Friedrich Fraundorfer

Graz University of Technology

218 PUBLICATIONS 11,392 CITATIONS

[SEE PROFILE](#)

3D Visual Perception for Self-Driving Cars using a Multi-Camera System: Calibration, Mapping, Localization, and Obstacle Detection

Christian Häne^a, Lionel Heng^c, Gim Hee Lee^d, Friedrich Fraundorfer^e, Paul Furgale^f, Torsten Sattler^b, Marc Pollefeys^{b,g}

^a*Department of Electrical Engineering and Computer Sciences, University of California Berkeley, Berkeley, CA 94720, United States of America*

^b*Department of Computer Science, ETH Zürich, Universitätstrasse 6, 8092 Zürich, Switzerland*

^c*Information Division, DSO National Laboratories, 12 Science Park Drive, Singapore 118225*

^d*Department of Computer Science, National University of Singapore, 13 Computing Drive, Singapore 117417*

^e*Institute for Computer Graphics & Vision, Graz University of Technology, Inffeldgasse 16, A-8010 Graz, Austria*

^f*Department of Mechanical and Process Engineering, ETH Zürich, Leonhardstrasse 21, 8092 Zürich, Switzerland*

^g*Microsoft, One Microsoft Way, Redmond, WA 98052, United States of America*

Abstract

Cameras are a crucial exteroceptive sensor for self-driving cars as they are low-cost and small, provide appearance information about the environment, and work in various weather conditions. They can be used for multiple purposes such as visual navigation and obstacle detection. We can use a surround multi-camera system to cover the full 360-degree field-of-view around the car. In this way, we avoid blind spots which can otherwise lead to accidents. To minimize the number of cameras needed for surround perception, we utilize fisheye cameras. Consequently, standard vision pipelines for 3D mapping, visual localization, obstacle detection, etc. need to be adapted to take full advantage of the availability of multiple cameras rather than treat

Email addresses: `chaene@eecs.berkeley.edu` (Christian Häne),
`lionel_heng@dso.org.sg` (Lionel Heng), `gimhee.lee@nus.edu.sg` (Gim Hee Lee),
`fraundorfer@icg.tugraz.at` (Friedrich Fraundorfer), `paul.furgale@mavt.ethz.ch`
(Paul Furgale), `sattler@inf.ethz.ch` (Torsten Sattler),
`marc.pollefeys@inf.ethz.ch` (Marc Pollefeys)

each camera individually. In addition, processing of fisheye images has to be supported. In this paper, we describe the camera calibration and subsequent processing pipeline for multi-fisheye-camera systems developed as part of the V-Charge project. This project seeks to enable automated valet parking for self-driving cars. Our pipeline is able to precisely calibrate multi-camera systems, build sparse 3D maps for visual navigation, visually localize the car with respect to these maps, generate accurate dense maps, as well as detect obstacles based on real-time depth map extraction.

Keywords: Fisheye Camera, Multi-camera System, Calibration, Mapping, Localization, Obstacle Detection

1. Introduction

Fully autonomous cars hold a lot of potential: they promise to make transport safer by reducing the number of accidents caused by inattentive or distracted drivers. They can help to reduce emissions by enabling the sharing of a car between multiple persons. They can also make commutes more comfortable and automate the search for parking spots. One fundamental problem that needs to be solved to enable full autonomy is the visual perception problem to provide cars with the ability to sense their surroundings. In this paper, we focus on the 3D variant of this problem: estimating the 3D structure of the environment around the car and exploiting it for tasks such as visual localization and obstacle detection.

Cameras are a natural choice as the primary sensor for self-driving cars since lane markings, road signs, traffic lights, and other navigational aids are designed for the human visual system. At the same time, cameras provide data for a wide range of tasks required by self-driving cars including 3D mapping, visual localization, and 3D obstacle detection while working both in indoor and outdoor environments. For full autonomy, it is important that the car is able to perceive objects all around it. This can be achieved by using a multi-camera system that covers the full 360° field-of-view (FOV) around the car. Cameras with a wide FOV, e.g., fisheye cameras, can be used to minimize the number of required cameras, and thus, the overall cost of the system. Interestingly, research in computer vision has mainly focused on monocular or binocular systems. In contrast, limited research is done for multi-camera systems. Obviously, each camera can be treated individually. However, this ignores the geometric constraints between the cameras and can

lead to inconsistencies across cameras.

In this paper, we describe a visual perception pipeline that makes full use of a multi-camera system to obtain precise motion estimates and fully exploits fisheye cameras to cover the 360° around the car with as little as four cameras. More precisely, this paper describes the perception pipeline [12, 16, 17, 25, 23, 27, 24, 26, 29, 28, 31, 13] designed for and used in the V-Charge¹ project [7]. Given the amount of work required to implement such a pipeline, it is clear that the description provided in this paper cannot cover all details. Instead, this paper is intended as an overview over our system that highlights the important design choices we made and the fundamental mathematical concepts used in the approach. As such, we cover the calibration of multi-camera systems [16, 31], including the extrinsic calibration of each camera with respect to the wheel odometry frame of the car, the mathematical models for ego-motion estimation of a multi-camera system [25, 28], as well as Simultaneous Localization and Mapping (SLAM) [23, 27, 24] and visual localization [26, 29] for multi-camera systems. In addition, we discuss depth map estimation from fisheye images [12] and how to obtain dense, accurate 3D models by fusing the depth maps, efficient re-calibration using existing SLAM maps [17], and real-time obstacle detection with fisheye cameras [13]. We provide references to the original publications describing each part in detail.

To the best of our knowledge, ours is the first purely visual 3D perception pipeline based on a multi-camera system and the first pipeline to fully exploit fisheye cameras with little to no visual overlap. Given the advantages of such a pipeline, and its inherent complexity, we believe that such an overview is fundamentally important and interesting for both academia and industry alike. Besides this overview, which is the main contribution of the paper, we also describe our dense height map fusion approach, which has only been published in an earlier version for indoor environments before.

In the following sections, we give a brief overview of our pipeline and review existing perception pipelines for self-driving cars.

1.1. System Overview

The 3D visual perception pipeline described in this paper was developed as part of the V-Charge project, funded by the EU’s Seventh Framework Programme. The goal of the project was to enable fully autonomous valet

¹Autonomous Valet Parking and Charging for e-Mobility, <http://www.v-charge.eu/>

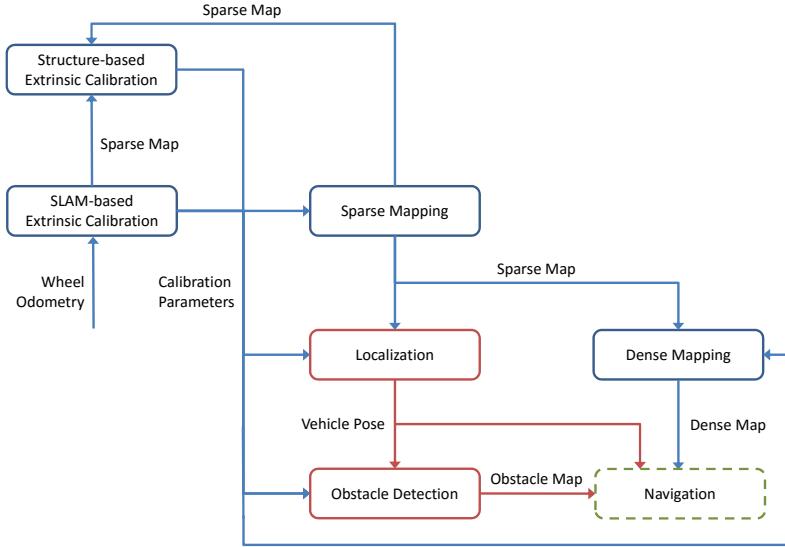


Figure 1: Our 3D visual perception pipeline from calibration to mapping. Each component in the pipeline is marked with a solid outline subscribes to images from the multi-camera system. Components marked with blue outlines run offline while those with red outlines run online. The outputs from our pipeline: the vehicle pose, obstacle map, and dense map, can be used for autonomous navigation.

parking and charging for electric cars. As indoor parking garages were a major target, our 3D perception pipeline does not use any GPS information.

Figure 1 provides an overview of our pipeline. Given sequences of images recorded for each camera during manual driving and the corresponding wheel odometry poses, our SLAM-based calibration approach, described in Section 2, computes the extrinsic parameters of the cameras with respect to the vehicle’s wheel odometry frame. The extrinsic parameters are then used by all the other modules in the pipeline. Our sparse mapping module, described in Section 3, estimates the ego-motion of the car from 2D-2D feature matches between the camera images. The estimated motion is then used to build a sparse 3D map. The sparse 3D map is used by our localization method, described in Section 4, to estimate the position and orientation of the car with respect to the 3D map from 2D-3D matches between features



Figure 2: (left) The two cars from the V-Charge project. (right) The cameras are mounted in the front and back and in the side view mirrors.

in the camera images and 3D points in the map. Given the poses estimated by the sparse mapping module, our dense mapping module, described in Section 5, estimates a dense depth map per camera image and fuses them into an accurate 3D model. Section 6 describes structure-based calibration and obstacle detection methods, both of which leverage our pipeline. Our structure-based calibration approach uses the sparse 3D map for efficient calibration while our obstacle detection uses camera images and the vehicle pose estimates from the localization to build an obstacle map.

The platform used by the V-Charge project is a VW Golf VI car modified for vision-guided autonomous driving. As shown in Figure 2, four fisheye cameras are used to build a multi-camera system. Each camera has a nominal FOV of 185° and outputs 1280×800 images at 12.5 frames per second (fps). The cameras are hardware-synchronized with the wheel odometry of the car.

1.2. Related Work

Many major car manufacturers, technology companies such as Google, and universities are actively working on developing self-driving cars. Google’s self-driving car project [18] relies on a combination of lasers, radars, and cameras in the form of a roof-mounted sensor pod to navigate pre-mapped environments. The cars use a custom-built, expensive and omnidirectional 3D laser to build a laser reflectivity map [30] of an environment, localize against that map, and detect obstacles. In contrast, we use a low-cost surround view multi-camera system. Such multi-camera systems can be found on mass-market cars from well-known automotive manufacturers, including BMW, Infiniti, Land Rover, Lexus, Mercedes-Benz, and Nissan.

The autopilot feature [19] in cars produced by Tesla Motors enables autonomous driving on highways through a combination of a forward-looking camera and radar. Due to the limited field-of-view, these cars cannot execute autonomous driving in cities which require surround perception. In contrast, our multi-camera system is designed to offer a surround view around the car and with no blind spots.

Similarly, the Bertha Benz self-driving car [43] uses vision and radar albeit with many more sensors to achieve a near-surround-view. The car uses front and rear stereo cameras with a 44° FOV to build a sparse map for localization. Additional to the stereo cameras the car is equipped with separate rear and front monocular cameras with a 90° FOV, which are used for localization itself. Notice that this map does not include visual features observed from the sides of the car, and localization may fail if the front and rear cameras are occluded. In addition, the two cameras used for localization provide two independent pose estimates which are fused in a filter-based framework. In contrast, we use all cameras jointly by treating the camera system as a generalized camera and developing novel algorithms for both, sparse mapping and localization.

2. Calibration

Calibration is an essential prerequisite for the use of multi-camera systems in autonomous driving. Projection of a 3D scene point to an image point requires the knowledge of both intrinsic and extrinsic parameters for the multi-camera system in addition to the vehicle pose. In our case, the intrinsic and extrinsic parameters associated with each camera correspond respectively to the parameters of a chosen camera model and that camera's 6-DoF pose with respect to the vehicle's odometry frame.

An imprecise calibration impacts all parts of a visual perception pipeline, leading to inaccurate mapping, localization, and obstacle detection results. Furthermore, environmental changes, wear and tear, and vibrations cause calibration parameters to slowly deviate from their true values over time. Multi-camera-based algorithms are sensitive to such calibration errors, and hence, frequent re-calibration is needed. Thus, we developed automatic calibration methods for the car's multi-camera system, which is capable of estimating accurate calibration parameters.

The vast majority of existing work [21, 22, 31] on calibration of multi-camera systems requires a fiducial target, usually a pattern board. The

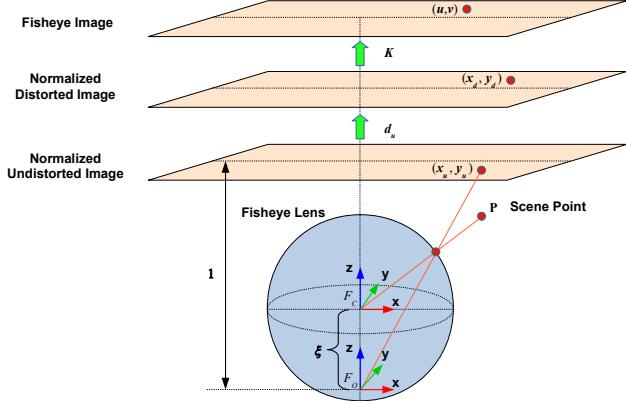


Figure 3: We use the camera model from [33], which comprises both a unified projection model and a radial-tangential distortion model.

fiducial target contains landmarks with known 3D coordinates which allows target-based calibration methods to compute the inter-camera transformations with metric scale. Our SLAM-based calibration [17] removes the need for a fiducial target by relying on features naturally occurring in the environment instead. This calibration jointly estimates the inter-camera transformations and builds a sparse map of the environment. Metric scale is inferred from wheel odometry data. Only one other SLAM-based calibration method [2] for multi-camera systems exists but only estimates the inter-camera transformations up to scale.

In this section, we describe the camera model [33] that is used throughout this paper, and our SLAM-based extrinsic calibration which does not assume overlapping fields of view for any pair of cameras. The code for both the intrinsic and extrinsic calibration is publicly available ².

2.1. Camera Model

Figure 3 illustrates how a scene point is reprojected onto an fisheye image using the camera model from [33], which we explain in the following. For a scene point with coordinates $\mathbf{P}_{F_C} = [X \ Y \ Z]^T$ in the camera's reference frame F_C , its projection onto a unit sphere, transformed into another reference frame F_O via a z -translation with magnitude ξ , is given by

$$\mathbf{P}_{F_O} = \frac{\mathbf{P}_{F_C}}{\|\mathbf{P}_{F_C}\|} + [0 \ 0 \ \xi]^T , \quad (1)$$

²<https://github.com/hengli/camodocal>

where ξ denotes the mirror parameter in the unified projection model.

We reproject \mathbf{P}_{FO} onto the normalized undistorted image plane, and compute the coordinates (x_u, y_u) of the resulting image point as

$$[x_u \ y_u \ 1]^T = \frac{\mathbf{P}_{FO}}{z_{\mathbf{P}_{FO}}} , \quad (2)$$

where $z_{\mathbf{P}_{FO}}$ is the z -component of \mathbf{P}_{FO} .

The camera model from [33] models the radial and tangential distortions caused by an imperfect lens by computing the coordinates (x_d, y_d) of the normalized distorted image point as

$$\begin{aligned} r &= x_u^2 + y_u^2 , \\ \mathbf{d}_u &= \begin{bmatrix} x_u(k_1r + k_2r^2) + 2p_1x_u y_u + p_2(r + 2x_u^2) \\ y_u(k_1r + k_2r^2) + p_1(r + 2y_u^2) + 2p_2x_u y_u \end{bmatrix} , \\ [x_d \ y_d]^T &= [x_u \ y_u]^T + \mathbf{d}_u . \end{aligned} \quad (3)$$

Here, $[k_1, k_2]$ and $[p_1, p_2]$ are the radial and tangential distortion parameters.

Finally, the coordinates (u, v) of the image point in the fisheye image are obtained by applying the projection matrix \mathbf{K} :

$$\begin{aligned} \mathbf{K} &= \begin{bmatrix} \gamma_1 & 0 & u_0 \\ 0 & \gamma_2 & v_0 \\ 0 & 0 & 1 \end{bmatrix} , \\ [u \ v \ 1]^T &= \mathbf{K} [x_d \ y_d \ 1]^T , \end{aligned} \quad (4)$$

where $[\gamma_1, \gamma_2]$ and $[u_0, v_0]$ are the focal lengths and principal point coordinates respectively in the unified projection model.

We use an automatic chessboard-based method for the intrinsic camera calibration which is described in [33]. We typically obtain an average reprojection error of ~ 0.2 pixels for each camera.

2.2. SLAM-based Extrinsic Calibration

Given the intrinsic calibration for each camera, our SLAM-based extrinsic calibration method relies on features naturally occurring in the environment and the motion of the car rather than using a fiducial target. We use OpenCV's SURF [1] implementation to extract features and their descriptors from images. Essentially, our calibration approach finds a solution to

the non-linear least squares (NLLS) problem comprising two sets of inverse-covariance-weighted residuals:

$$\min_{\mathbf{T}_c, \mathbf{V}_i, \mathbf{X}_p} \sum_{c,i,p} \rho(\Delta \mathbf{z}_{c,i,p}^T \mathbf{W}_1 \Delta \mathbf{z}_{c,i,p}) + \sum_i \Delta \mathbf{u}_i^T \mathbf{W}_2 \Delta \mathbf{u}_i , \quad (5)$$

where

$$\begin{aligned} \Delta \mathbf{z}_{c,i,p} &= \pi(\mathbf{T}_c, \mathbf{V}_i, \mathbf{X}_p) - \mathbf{p}_{cip} , \\ \Delta \mathbf{u}_i &= \mathbf{h}(\mathbf{V}_i, \mathbf{V}_{i+1}) - \mathbf{Z}_{i,i+1} . \end{aligned} \quad (6)$$

The first and second sets of residuals correspond to the sum of squared image reprojection errors and the sum of squared relative pose errors. \mathbf{W}_1 and \mathbf{W}_2 denote the inverse of the measurement covariance corresponding to the first and second sets of residuals, respectively. π is a projection function that computes the image coordinates of the scene point \mathbf{X}_p seen in camera c given the vehicle pose \mathbf{V}_i , and the rigid body transformation from the camera frame to the odometry frame \mathbf{T}_c . \mathbf{p}_{cip} is the observed image coordinates of \mathbf{X}_p seen in camera c with the corresponding vehicle pose \mathbf{V}_i . ρ is a robust cost function used for minimizing the influence of outliers. \mathbf{h} is a function that computes the relative pose given two consecutive vehicle poses \mathbf{V}_i and \mathbf{V}_{i+1} , and $\mathbf{Z}_{i,i+1}$ is the observed relative pose between \mathbf{V}_i and \mathbf{V}_{i+1} as measured from wheel odometry data. We visualize the reference frames and variables in Figure 4.

This problem is highly non-convex, and thus, contains many local minimas. Hence, a good initialization is required, which we obtain through a series of steps described below.

2.2.1. Camera-Odometry Calibration

The first step obtains an initial estimate of the extrinsic parameters. In this step, we find a least-squares solution to the planar hand-eye calibration problem [10] that relates the relative motions of the camera and the vehicle, and is not degenerate under planar motion. The relative motions of the camera and the vehicle are estimated via monocular visual odometry and wheel odometry, respectively. The solution yields a 5-DoF transformation between the camera and odometry frames; the height of the camera with respect to the odometry frame is not observable due to planar motion.

2.2.2. Scene Point Reconstruction

The second step refines the extrinsic parameters based on a 3D map of the scene: For each camera, we use the initial estimate of the extrinsic parameters

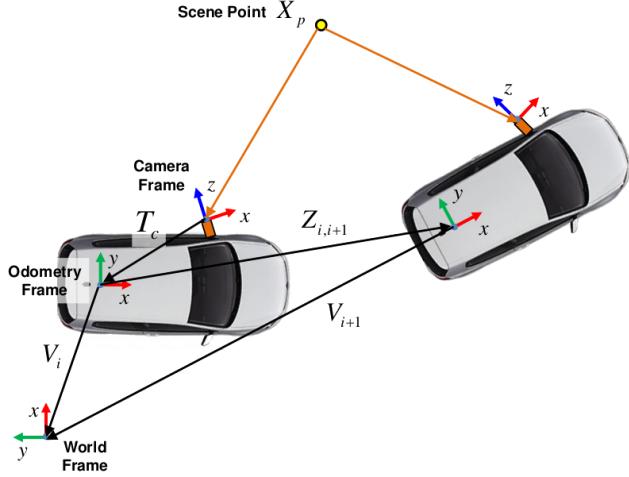


Figure 4: Visualization of the different reference frames and variables described in the NLLS problem which our SLAM-based extrinsic calibration has to solve.

and the vehicle poses, which are set to the odometry poses, to triangulate the 2D-2D feature correspondences found from monocular visual odometry. We then jointly refine the extrinsic parameters and scene point coordinates by minimizing the cost function in Equation 5 while fixing the vehicle poses.

The estimates of the extrinsic parameters, vehicle poses, and scene point coordinates obtained up to this point are still inaccurate due to two issues: The wheel odometry data used to initialize the vehicle poses is inaccurate both locally and globally. The 1 cm resolution of the wheel encoders causes the local inaccuracy while the global inaccuracy arises from the pose drift due to the interoceptive nature of the wheel odometry. The second issue is that there are no feature correspondences between different cameras and thus no constraints that relate the extrinsic parameters of the different cameras in our multi-camera system.

We handle the first issue by finding loop closures and performing pose graph optimization to correct the drift inherent in wheel odometry. These loop closures yield inter-camera feature correspondences, but they are not useful priors for estimating the extrinsic parameters as these feature correspondences often correlate different cameras observing the same scene points far apart in time. In order to accurately estimate the extrinsic parameters, we require inter-camera correspondences that correlate to features found in

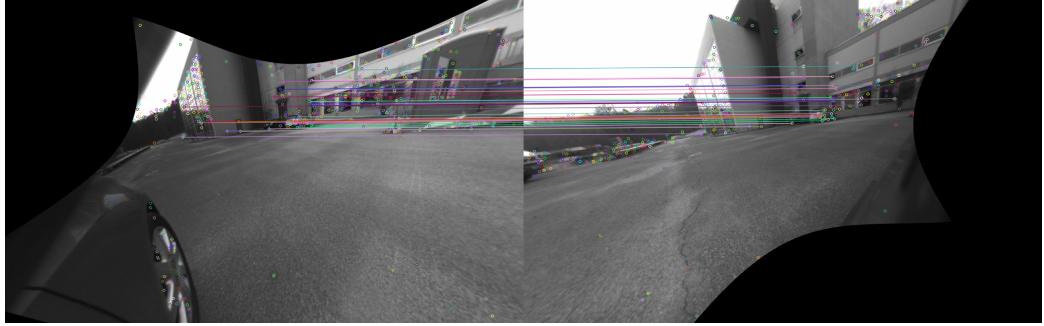


Figure 5: Inlier feature point correspondences between rectified pinhole images in the right and front cameras.

different cameras observing the same scene points close in time. These correspondences provide much stronger constraints.

2.2.3. *Inter-Camera Feature Matching*

In our setup, the cameras are placed to have minimal overlapping fields of view to minimize the number of cameras and thus production costs. For inter-camera feature matching, we maintain a local frame history for each camera and do exhaustive feature-matching between all frames from each camera and all frames stored in the local frame history corresponding to the other cameras. Obviously, longer local frame histories lead to longer matching times.

In our implementation, the local frame history for each camera spans a distance of 3 meters as measured from the vehicle poses.

With significant viewpoint differences between different cameras at different times, feature matching may yield only few correspondences and even less correct matches. The large distortion effects inherent to our fisheye cameras further aggravate this problem. As shown in Figure 5, we thus rectify these two images on a common image plane before feature extraction to find more inlier matches. The plane corresponds to the average rotation between the first camera's pose and the second camera's pose; the camera poses are computed using the current estimates of the extrinsics and the vehicle poses.

2.2.4. *Joint Optimization*

Now that we have a good initial estimate of every parameter, we are able to solve Equation 5 using a NLLS solver. We obtain an accurate estimate of the extrinsic parameters for the multi-camera system with the help of strong

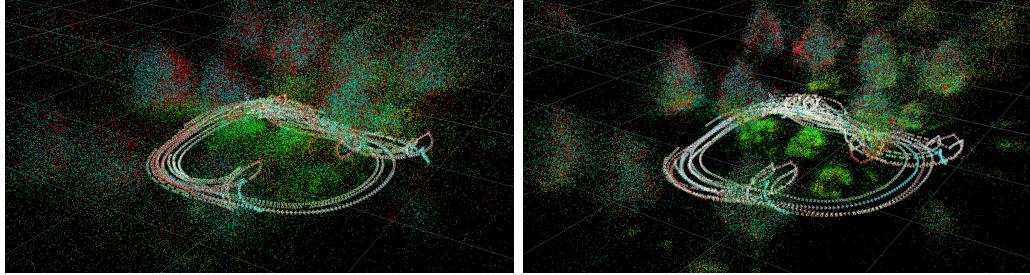


Figure 6: Sparse maps generated with (left) the initial parameters from intrinsic and hand-eye calibration and (right) the parameters obtained after optimization. Points seen by a certain camera are assigned an unique color.

priors provided by inter-camera feature correspondences found in the previous feature-matching step. Figure 6 compares two sparse maps; the map on the left is generated with parameters initially estimated from the steps prior to the joint optimization, while the map on the right is generated with the optimized parameters from the joint optimization. This figure qualitatively shows that the optimized 3D points, and in turn, the extrinsic parameters are significantly more accurate as the corresponding sparse map is much more well-defined.

2.2.5. Summary

More details on the SLAM-based extrinsic calibration and quantitative evaluations of its accuracy can be found in our original publication [17]. Our SLAM-based extrinsic calibration allows us to calibrate a multi-camera system anywhere, in an unsupervised manner, and without the need for initial guesses of the extrinsic parameters and overlapping fields of view. At the same time, our calibration generates a sparse map of the environment which can be used for visual localization.

3. Sparse Mapping

The calibration method described in the last section requires that enough features are found in each camera image. Thus, we usually perform calibration in small, highly textured scenes. In order to be able to also map larger, less textured scenes, we adapted a standard visual SLAM / Structure-from-Motion (SfM) [15] framework to process the images collected from our multi-camera system. Like the calibration described above also our sparse mapping

system uses SURF [1] features. Our approach exploits the known camera calibration and is thus more efficient at building sparse maps offline than the SLAM-based approach introduced in the last section. In the following, we describe how to estimate the ego-motion of the camera system (Section 3.1) and how to detect and handle loop-closures (Section 3.2).

3.1. Ego-Motion Estimation

The first step toward building a sparse map is to estimate the ego-motions of the multi-camera system based on the images collected during driving. A naïve approach for ego-motion estimation would be to treat each camera individually and compute the relative motion between consecutive frames using classical epipolar geometry [15]. However, the resulting motion estimated for each camera is not guaranteed to be consistent with the motions of the other cameras. We thus compute the ego-motion jointly for all cameras by modeling our multi-camera system as a generalized camera [36].

The main difference between a generalized and a perspective camera is that light rays passing through the 2D image features and their respective camera centers do not meet at a single center of projection. Pless *et. al.* [36] suggested that the light rays from a generalized camera can be expressed in a common reference frame with the Plücker line representation.

As illustrated in Figure 7, let V denote the reference frame of the multi-camera system and let (R_C, t_C) be the pose of camera C in V . A Plücker line expressed in V is given by $l = [q^T \ q'^T]^T$, with $q = R_C \hat{x}$ and $q' = t_C \times q$. Here, $\hat{x} = \mathbf{K}^{-1} \mathbf{x}$ is the viewing ray corresponding to a 2D point x in the image of camera C .

Given a pair of image point correspondence expressed as Plücker lines $l \leftrightarrow l'$ over two consecutive frames V and V' , the Generalized Epipolar Constraint (GEC) [36, 32] is given by

$$l'^T \underbrace{\begin{bmatrix} E & R \\ R & 0 \end{bmatrix}}_{E_{GC}} l = 0 , \quad (7)$$

where $E = [t] \times R$ is the conventional Essential matrix [15], and (R, t) is the unknown relative motion between V' and V . E_{GC} is the generalized Essential matrix. Note that the absolute metric scale for t can be obtained from the GEC since the point correspondences expressed as Plücker lines contain metric scale information from the extrinsics.

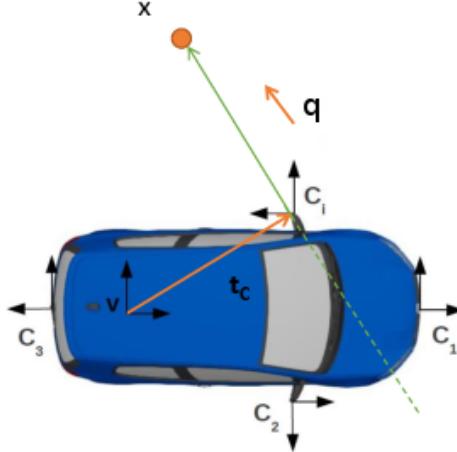


Figure 7: Illustration of a Plücker light ray.

Using Equation 7, (R, t) can be computed from 17 correspondences by solving a linear system [36, 32]. However, this 17-point solver is not practical for real-world applications, where it is necessary to robustly handle wrong correspondences inside a RANSAC loop [6]. In order to find the correct model with a probability of 99%, RANSAC estimate and evaluate $m = \frac{\ln(1-0.99)}{\ln(1-v^n)}$ pose hypotheses from random samples of size n . Here, v is the inlier ratio, i.e., the ratio of correct matches among all correspondences. For $v = 0.5$, the 17-point algorithm ($n = 17$) needs 603606 iterations. The non-linear 6-point algorithm by Stewénius *et al.* [40] only requires 292 iterations, but also generates up to 64 solutions that need to be considered in each iteration. As a result, both solvers are not applicable in practical scenarios where fast RANSAC run-times are required.

In order to reduce both the number of iterations and the number of solutions generated, we exploit the fact that a car typically follows the Ackerman motion constraint [39], i.e., the car undergoes a circular motion on a plane around the Instantaneous Center of Rotation (ICR) (cf. Figure 8). In this case, the relative motion (R, t) between consecutive frames can be parameterized with 2 degrees-of-freedom (DOF) as

$$R = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad t = \rho \begin{bmatrix} \cos \varphi_v \\ \sin \varphi_v \\ 0 \end{bmatrix}, \quad (8)$$

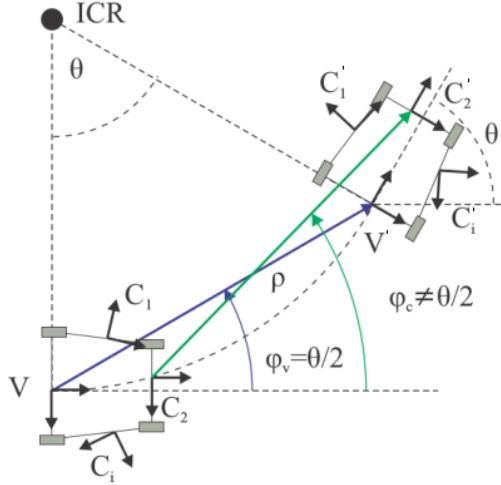


Figure 8: Relation between generalized camera in Ackermann motion [25].

where θ is the relative yaw angle and ρ is the scale of the relative translation. It can be easily seen from Figure 8 that $\varphi_v = \frac{\theta}{2}$. Inserting Equation 8 into Equation 7 and using two correspondences, we obtain a system of two polynomials (see our original publication [25] for details) in the two unknowns ρ and θ . This system can be solved in closed form by finding the roots of a univariate cubic polynomial, resulting in up to 6 solutions. Since only 2 correspondences are required, RANSAC only needs 16 iterations for $v = 0.5$. Thus, applying our solver inside RANSAC is more efficient than using the 17-point and 6-point solvers, leading to an approach that is, in contrast to [32, 36, 40], applicable in practice.

Figure 9 shows a trajectory (red) and 3D map points (blue) obtained by using our solver inside a SfM framework with bundle adjustment and loop-closure detection. As can be seen, the trajectory estimated from our 2-point algorithm is locally very close to the GPS/INS ground truth (green). Globally, the error is below 3m. Some global shape difference is due to the fact that the mapped parking lot is not exactly planar.

3.2. Loop-Closure Detection & Handling

Small errors in the relative motion estimates eventually accumulate and lead to drift, i.e., a significant error in position and orientation when revisiting a place. Consequently, we need to perform loop-closure detection to identify previously visited locations, derive pose constraints between the current and



Figure 9: Top view of the trajectory and 3D map points estimated using our 2-point solver compared with GPS/INS ground truth [25].

previous visits, and optimize the trajectory and map accordingly. To this end, we form a pose-graph [23] $\mathcal{G} \in \{\mathcal{V}, \mathcal{E}\}$. The nodes $\mathcal{V} = v_1, v_2, \dots, v_K$ represent the global poses of the car, computed by concatenating the relative poses estimated by our 2-point algorithm. Among the edges $\mathcal{E} = \{e_{1,2}, e_{2,3}, \dots, e_{i,j}\}$, an edge $e_{i,i+1}$ thus corresponds to the relative motion estimated by the 2-point algorithm between two subsequent time steps. In contrast, an edge $e_{i,j}$, $j \neq i + 1$, corresponds to a pose estimated from non-sequential frames related via a loop-closure event.

Potential loop-closure edges are detected by vocabulary tree-based image retrieval [35], followed by geometric verification [27] to estimate the relative pose between frames. An edge $e_{i,j}$, $i \neq j$, is added to the pose graph if its inlier ratio is above a certain threshold [27]. Given the new edges found during loop-closure detection, the pose graph is optimized as

$$\operatorname{argmin}_{v \in \mathcal{V}} \sum_{e_{i,j} \in \mathcal{E}} \|h(v_i, v_j) - e_{i,j}\|^2 . \quad (9)$$

Here, $h(v_i, v_j)$, $i \neq j$, denotes the relative pose obtained from the two global poses v_i and v_j . Equation 9 thus aims to find a set of global positions that are as consistent as possible to the estimated relative motions. We use the robust approach from our original publication [23] for pose graph optimization.

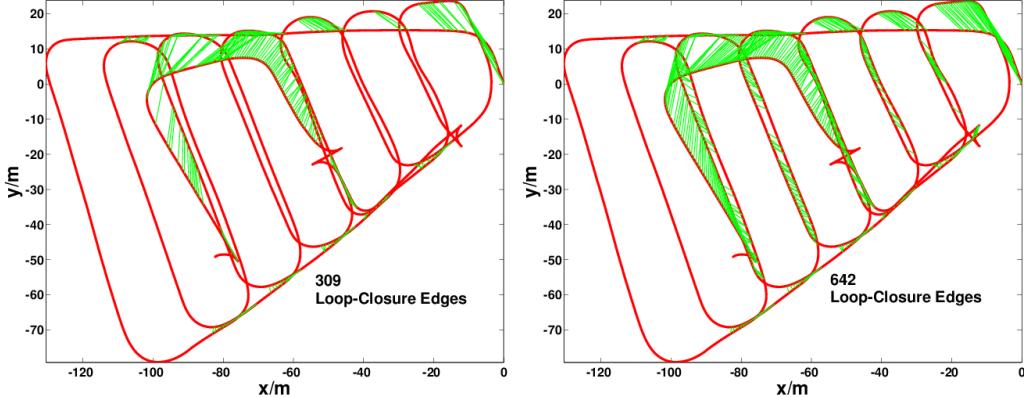


Figure 10: Pose-graph of the Parking Garage dataset from wheel odometry (red) before pose-graph optimization [24]. Comparison on the total loop-closure edges (green) found from (left) a forward looking monocular camera and (right) our multi-camera system.

The Ackermann motion model is only valid for two consecutive frames and is violated if the car revisits a place. Thus, our 2-point solver cannot be used for estimating the relative pose between loop closures. Instead, we only assume planar motion for a loop-closure candidate. Consequently, the relative pose (R, t) has 3 DOF and is given by

$$R = \frac{1}{1+q^2} \begin{bmatrix} 1-q^2 & -2q & 0 \\ 2q & 1-q^2 & 0 \\ 0 & 0 & 1+q^2 \end{bmatrix}, \quad t = \begin{bmatrix} x \\ y \\ 0 \end{bmatrix}, \quad (10)$$

where $q = \tan(\frac{\theta}{2})$. Inserting Equation 10 into Equation 7 and using 3 correspondences, we obtain a system of three polynomials (see our original publication [24] for details) in the three unknowns x , y and q . Solving this system [5] results in up to six real solutions. Since only 3 correspondences are required, we only need 34 RANSAC iterations for $v = 0.5$.

Note that in case the planar motion assumption is violated, e.g., when the car is moving up a ramp, we can make use of the roll and pitch angles from an inertia measurement unit (IMU) to efficiently estimate the motion with a 4-point algorithm as detailed in our previous work [28].

Figure 10 shows a comparison of the total number of loop-closures detected by the monocular front-looking camera and multi-camera system. We can see that the multi-camera system leads to many more loop-closure detections especially in the parts where the car is moving in opposite direction to the previously visited path.

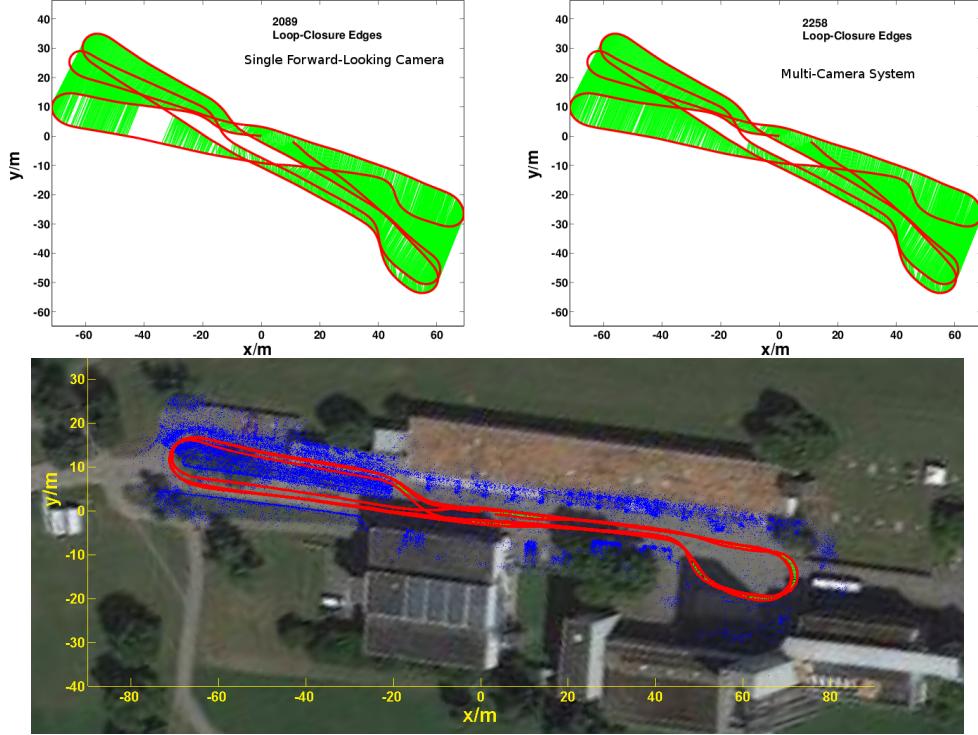


Figure 11: (Top row) Pose-graph of the Campus dataset from visual odometry (red) before pose-graph optimization [24]. Green lines denote loop-closure constraints found with (top left) a single forward-looking camera and (top right) our multi-camera system. (Bottom) Pose-graph (red) and 3D scene points (blue) for Campus dataset after pose-graph optimization overlaid on the satellite image.

Figure 11 shows another example of the loop-closure detection before (top) and after (bottom) pose-graph optimization. We observe almost similar number of loop-closure detections for both the front-looking camera and multi-camera system in this example. This is because most of the loop-closure detection paths are facing the same directions. As can be seen, the reconstructed estimated 3D points after pose-graph optimization align well with the buildings shown in the satellite image.

3.3. Summary

More details of the minimal solvers for ego-motion estimation and loop-closure handling can be found in our original publications [25, 24, 28]. Our minimal solvers forms the backbone of our sparse mapping framework. They

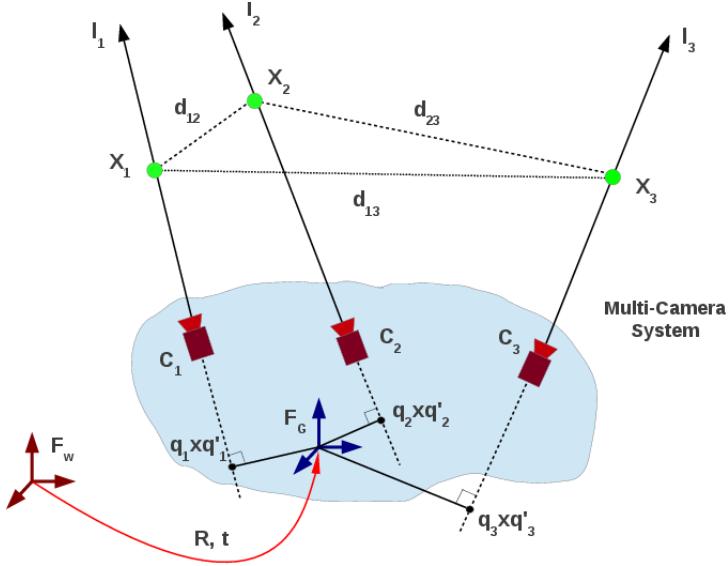


Figure 12: Illustration of the pose estimation problem for a multi-camera system [26, 29].

enable us to map out large areas efficiently by treating multiple cameras that are fixed rigidly onto the car with known extrinsics parameters jointly as one.

4. Localization

Each 3D point in the sparse maps build with the approach from Section 3 has been triangulated from multiple 2D image features, e.g., SURF features [1]. Thus, the maps can also be used for online localization of the car: Given the features extracted from novel images, 2D-3D matches between the image features and the 3D points in the map can be established by comparing the associated feature descriptors [38]. The global pose of the cameras wrt. the map can then be estimated from these 2D-3D correspondences.

Formally, localization refers to the problem of determining the rigid transformation (R, t) between the fixed world frame F_W of the map, and the multi-camera frame F_G , given a set of 2D-3D correspondences (cf. Figure 12). Obviously, the pose could be estimated for each camera individually using standard pose solvers [14, 37, 34]. However, this approach would ignore that we know the geometric relation between the cameras in the multi-camera system. In addition, using a larger FOV usually leads to more accurate pose

estimates. We thus derived an approach for estimating the global pose of a generalized camera [26, 29], which we outline below. As is the case for calibrated perspective cameras [14], our solver requires three 2D-3D correspondences for global pose estimation.

As for the relative pose solvers presented above, we again use Plücker lines to handle the fact that not all light rays meet in a single center of projection. The 3D position of a point X_i^G on a Plücker line defined in F_G is given by

$$X_i^G = q_i \times q'_i + \lambda_i q_i , \quad (11)$$

where $\lambda_i > 0$ is the depth of the point. As for perspective cameras [14], the distances d_{ij} (cf. Figure 12) between the 3D points X_i in F_W and the points X_i^G in F_G need to be the same. Using this constraint and the Plücker line representation, we can derive three equations that can be used to solve for the three unknown depths λ_i . Knowing the depths defines the 3D point positions X_i in F_G and we can compute the global pose (R, t) that aligns the X_i^G and the X_i using absolute orientation [14].

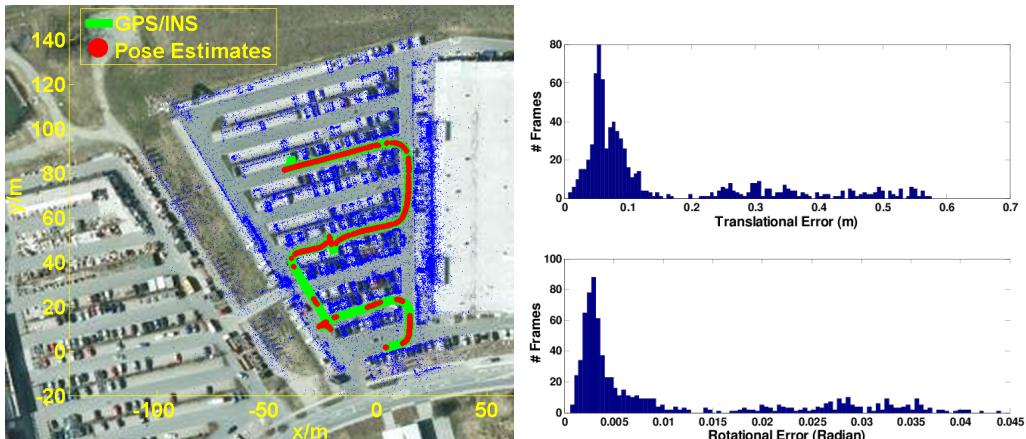


Figure 13: (a) Localization results for the Supermarket dataset. Results from frames with < 20 correspondences are discarded. (b) Plots showing the distribution of the translational and rotational errors against GPS/INS ground truths [26, 29].

Figure 13 shows the results of our visual localization algorithm. We first collected a set of images with GPS/INS ground truth to build the sparse map (blue points). We collect a separate set of images with GPS/INS ground truth (green trajectory) at another time for testing our localization algorithm. The

poses estimated with our algorithm are shown in red. As can be seen, our algorithm produces accurate localization results with only small errors. Note that the missing parts of the trajectory is due to failure in visual recognition because of changes in the scene. Such failures are typically not a problem in practice as these missing parts can be bridged via the car’s wheel odometry. More details can be found in our previous works [26, 29].

5. Dense Mapping

Surfaces such as building facades and the ground are generally not captured well in the sparse maps. This section explains how dense multi-view stereo matching can be done directly on the fisheye images and how the depth maps can be fused into a dense height map representation of the environment. The output of the multi-view stereo matching are depth maps which can be computed in real-time and can be directly used during fully automated driving missions. The dense height map is computed in an off-line manner and is used to map an environment for later automated driving within this space.

5.1. Depth Map Computation

We use plane sweeping stereo matching [4, 42] which has the strength that multiple images can be matched without prior stereo rectification. Meaning, the epipolar lines do not need to be aligned with the image scan lines. Knowledge about predominant surface directions can be incorporated [8] and we extended the algorithm to directly match fish eye images [12].

Images from a similar viewpoint are compared to a reference image by projecting the images onto a plane which serves as proxy geometry. If the plane coincides with the original geometry the projection of the images has high similarity. Instead of projecting to the plane, the reference image is fixed and the other images are warped to reference image via the proxy plane, which is a planar homography for pinhole images. For fish eye images the camera model can be included. The comparison of the images takes place in the reference view. Choosing the plane with lowest dissimilarity for each pixel leads to a dense depth map for the reference image.

We follow our original paper [12] and refer the reader for more details. Given is a set of images $\mathcal{I} = \{I_1, \dots, I_n, \dots, I_N\}$ (tangential and radial distortion removed) and the respective camera parameters $\mathcal{P} = \{P_1, \dots, P_n, \dots, P_N\}$, with $P_n = \{\phi_n, \mathbf{R}_n, \mathbf{C}_n\}$. The camera intrinsics $\phi = \{\xi, \mathbf{K}\}$ are composed of

the parameter ξ of the unified projection model Section 2.1 and the intrinsic camera projection matrix \mathbf{K} . The projection of a 3D point \mathbf{X} to the image plane is described as $[\mathbf{x}^T, 1]^T = \mathbf{K}\bar{h}(\xi, \mathbf{R}\mathbf{X} - \mathbf{R}^T\mathbf{C})$. The inverse operation maps an image pixel \mathbf{x} to a ray from the camera center \mathbf{C} to a point on the ray defined as $\mathbf{X} = \mathbf{R}\bar{h}^{-1}(\xi, \mathbf{K}^{-1}[\mathbf{x}^T, 1]^T) + \mathbf{C}$. Finally, we need a set of planes $\Pi = \{\Pi_1, \dots, \Pi_m, \dots, \Pi_M\}$, which serve as proxy geometry to test the image dissimilarity at different depths. They are defined as $\Pi = [\mathbf{n}^T, d]$, with \mathbf{n} being the unit-length normal direction pointing to the camera and d the distance of the plane to the camera center. The transformation which warps image I_n to the reference image I_{ref} using plane hypothesis Π_m is then given as,

$$\begin{aligned} \mathbf{H}_{n,m}^{\text{ref}} &= \mathbf{R}_n^T \mathbf{R}_{\text{ref}} + \frac{1}{d_m} (\mathbf{R}_n^T \mathbf{C}_n - \mathbf{R}_n^T \mathbf{C}_{\text{ref}}) \mathbf{n}_m^T \\ [\mathbf{x}_n^T, 1]^T &= \mathbf{K}_n \bar{h}(\xi_n, \mathbf{H}_{n,m}^{\text{ref}} \bar{h}^{-1}(\xi_{\text{ref}}, \mathbf{K}_{\text{ref}}^{-1}[\mathbf{x}_{\text{ref}}^T, 1]^T)). \end{aligned} \quad (12)$$

Each image I_n , with $n \neq \text{ref}$ is warped into an image $I_{n,m}^{\text{ref}}$ for each plane Π_m using Equation 12 and a pixel-wise image dissimilarity $\mathbf{D}_{n,m}^{\text{ref}}$ is computed between each of the warped images and the reference image. We use the negative zero mean normalized cross correlation (ZNCC) as image dissimilarity measure. For each pixel and plane we have several image dissimilarity measures originating from all the images I_n . In order to have a single dissimilarity measure for each pixel and plane they are aggregated into a single $\mathbf{D}_m^{\text{ref}}$. We mention two choices, the first one is simply averaging the costs. This is computationally the most efficient variant and is hence used for online applications. The second one is very suitable for images taken in sequence, such as images taken with a single camera while a car is moving. One average is computed over all the images which were taken before the reference image and one average over all of the images which are taken after the reference image. Finally, the one which has a lower dissimilarity is the final aggregated cost. The advantage of this procedure is that it is more robust to occlusions than simple averaging [20] but needs to know images taken in the future with respect to the reference image and is therefore used for offline mapping in our system.

The final depth map is extracted by choosing for each pixel the plane \hat{m} with lowest dissimilarity measure and computing the induced depth as

$$Z_{\hat{m}}(\mathbf{x}) = -\frac{d_{\hat{m}}}{\mathbf{n}_{\hat{m}}^T \bar{h}(\xi_{\text{ref}}, \mathbf{K}_{\text{ref}}^{-1}[\mathbf{x}^T, 1]^T)} . \quad (13)$$

To reduce the artifacts of the depth discretization we use sub-plane interpolation by fitting a parabola [41] into the scores of the best plane and the neighboring planes with the same normal direction. The depth is then chosen at the position where the parabola has its minimum.

The intrinsic camera parameters ϕ are determined during the camera calibration procedure Section 2. The camera poses are either coming directly from the sparse map computed during the extrinsic calibration Section 2.2 or during sparse mapping Section 3. For online mapping the vehicle’s wheel odometry poses can be used (cf. Section 6.2). As input images we select a sequence of a few images captured by a single camera while the vehicle is moving. No matching between different cameras is done. In order to select the planes we first select the normal direction of the planes. The most important direction is fronto-parallel to the image plane. Choosing planes in this direction is very similar to the rectified two-view stereo matching using disparities along scan lines. A near and a far distance to the camera determines the region in which we want to measure the geometry. This is typically a range from 30cm to 50m. Within this range the planes are placed such that the difference of inverse distance $1/d$ to the neighbors is constant, which corresponds to disparities. Note that the sideways facing cameras are pointing towards the ground and therefore a near distance of 30cm to the camera allows us to reconstruct the ground even right next to the car (cf. Figure 14). For a more faithful reconstruction of the ground, planes parallel to the ground can be used in addition. We use this for online obstacle detection Section 6.2. Figure 14 shows example depth maps acquired using our method.

5.1.1. Summary

In this section we presented a system which allows to directly match fisheye images. It has a less than 10% running time increase compared to using images with the same resolution and a pinhole projection model. This allows us to get a full coverage of the field of view of the fisheye camera with very little runtime overhead over first unwarping to a pinhole projection. Further evaluations including run-time performance measurements can be found in [12] and our stereo matching code is publicly available³.

³<http://www.cvg.ethz.ch/research/PlaneSweepLib/>

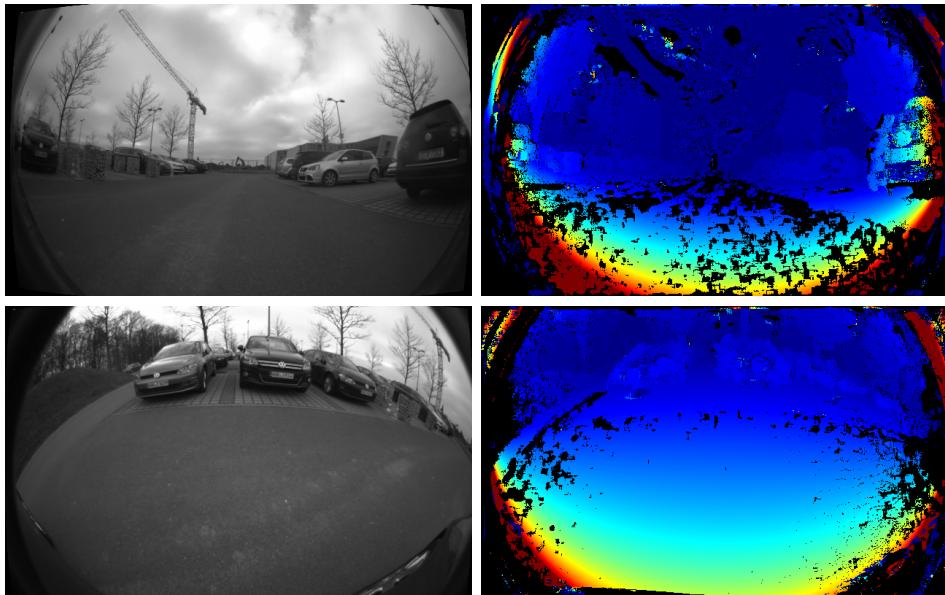


Figure 14: Two example depth maps next to the corresponding reference image. The depth is color coded with dark red corresponding to the selected near distance and dark blue to selected far distance. Note that the ground directly next to the front wheel in the lower image is correctly reconstructed.

5.2. Height Map Fusion

Our height map fusion approach is a modified version of our method [11], which we initially proposed for indoor environments. We give a brief overview of the method focusing on the modified steps and refer the reader to [11] for more details. First, all the depth maps are fused into a voxel grid and afterwards a regularized height map is extracted out of the voxel grid. For each voxel (x, y, z) a signed scalar weight $w_{(x,y,z)}$ is stored which indicates how strongly the data indicates that this voxel lies in the free (negative weight) or occupied space (positive weight). These weights are computed from the input depth maps.

For the next steps it is important to mention that the z -axis of the voxel grid is aligned with the vertical direction. In order to compute the final height map a set of values are extracted for each z -column of the voxel grid on which a 2-pass optimization is executed. The first pass of the optimization divides the domain Ω of the height map into a region which is observed Ω_{obs} and a non-observed region. The second pass determines the height values for locations within Ω_{obs} . Note that once we have the relevant information extracted from each z -column it can be discarded and hence the entire voxel grid never needs to be stored. First, we extract for each point (x, y) an optimal height based on the data of just a single z -column [9] by minimizing the cost function

$$C_{(x,y)}(h) = - \sum_{z < h} w_{(x,y,z)} + \sum_{z \geq h} w_{(x,y,z)} . \quad (14)$$

For the first pass of the optimization we need two values, $C_{(x,y)}^{\min}$ which defines the minimal cost and $C_{(x,y)}^{\text{occ}}$, which defines the cost if we assume that the entire column is in the occupied space. The rationale behind this is that we only want to compute a height value if we are reasonably sure that we did observe free space at a specific location. In order to avoid having a noisy region Ω_{obs} we penalize its boundary length by minimizing the convex energy functional

$$E^{\text{Labeling}}(l) = \int_{\Omega} \|\nabla l(\mathbf{x})\|_2 + \lambda_l(l(\mathbf{x})(C_{(x,y)}^{\min} + \gamma) + (1 - l(\mathbf{x}))C_{(x,y)}^{\text{occ}}) d\mathbf{x} \quad (15)$$

$$\text{subject to } l(\mathbf{x}) \in [0, 1]. \quad (16)$$

The function $l(\mathbf{x})$ indicates if a location is included in Ω_{obs} . The parameter γ is a penalty for including a location into the observed space and is used

to adjust how strongly the free space needs to be observed. The parameter λ_l adjusts the influence of the total variation (TV) smoothness prior which penalizes the boundary length. The functional is minimized using the first order primal-dual algorithm [3].

The second pass of the optimization is again posed as convex optimization. The data is represented as a convex approximation of Equation 14 locally around the per-location minimal cost height $\hat{h}_{(x,y)}$ with a function of the form

$$C_{(x,y)}^{\text{conv}}(h) = \alpha_1[\hat{h}_{(x,y)} - h - \delta]_+ + \alpha_2[h - \delta - \hat{h}_{(x,y)}]_+ , \quad (17)$$

with $[\cdot]_+ = \max\{0, \cdot\}$. The parameters α_1 and α_2 steer how strongly the cost increases while we move away from the locally best position. We propose to use a parameter δ which ensures that the function is flat around the minimum and models the uncertainty due to the discretization and is hence chosen as half the discretization step. In the final optimization pass we propose to regularize the height map with a Huber total variation smoothness prior

$$E^{\text{Height}} = \int_{\Omega_{\text{obs}}} \|\nabla h(\mathbf{x})\|_{\varepsilon} + \lambda_h C_{(x,y)}^{\text{conv}}(h(\mathbf{x})) d\mathbf{x} . \quad (18)$$

$\|\cdot\|_{\varepsilon}$ denotes the Huber norm which is defined as

$$\|\mathbf{x}\|_{\varepsilon} = \begin{cases} \|\mathbf{x}\|_2^2 / (2\varepsilon) & \text{if } \|\mathbf{x}\|_2 \leq \varepsilon \\ \|\mathbf{x}\|_2 - \varepsilon/2 & \text{if } \|\mathbf{x}\|_2 > \varepsilon. \end{cases} \quad (19)$$

Using the Huber norm leads to smooth height map and allows for discontinuities. The first order primal-dual algorithm [3] is used for numerical minimization.

In order to quantitatively asses the performance of our method we mapped an underground parking garage and afterwards measured various horizontal and vertical distances using a tape measure as ground truth. The quantitative results presented in Figure 15 and Table 1 show that the map is accurate enough for autonomous driving with a mean error of 7.06cm over 9 measurements. An example of a larger car park is depicted in Figure 16.

5.2.1. Summary

In this section we presented a method which fuses depth maps into a height map representation. Our quantitative evaluation shows that the error

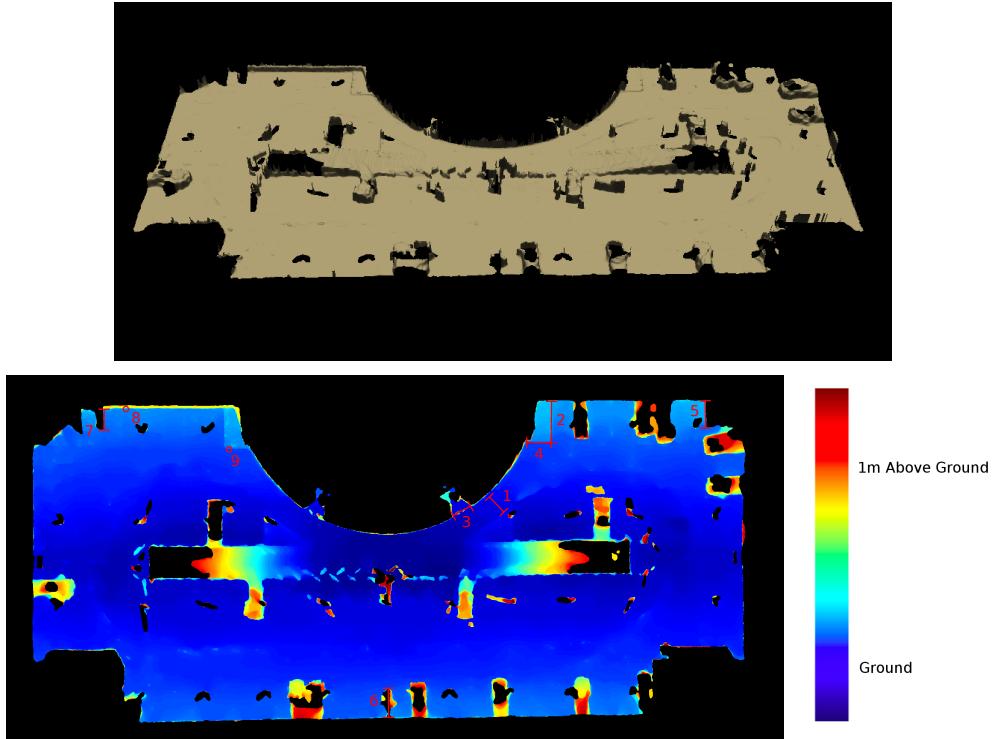


Figure 15: (Top) Height map visualized as mesh. (Bottom) color coded height map (dark blue lowest, dark red highest) with the positions of the 9 measurements indicated in red. Measurements 1 to 7 are length measurements and measurements 8 and 9 measure the height difference that occurs in the indicated circle.

is usually within a few centimeters. Our method allows us to map large parking spaces and efficiently store them as height map. Also refer to an earlier version of our algorithm in [11].

6. Applications

6.1. Structure-based Calibration

Both the SLAM-based extrinsic calibration and sparse mapping pipelines described in Section 2.2 and Section 3 respectively generate a sparse map of the environment. This sparse map is used for visual localization in Section 4. This map can also be used for structure-based calibration [16, 17] which leverages visual localization to calibrate the multi-camera system. Here, we localize each camera individually as the extrinsic parameters are unknown.

Nr.	Ground truth	Map	Error
1	2.725m	2.650m	7.5cm
2	4.895m	4.800m	9.5cm
3	2.570m	2.485m	8.5cm
4	2.740m	2.900m	16.0cm
5	3.140m	3.100m	4.0cm
6	3.450m	3.400m	5.0cm
7	2.505m	2.450m	5.5cm
8	0.850m	0.800m	5.0cm
9	0.125m	0.100m	2.5cm

Table 1: Quantitative evaluation for the 9 indicated measurements in Figure 15. The mean error over all the 9 measurements is 7.06cm.



Figure 16: Height map of a large car park visualized as mesh.

In contrast, in Section 4, we localize a single multi-camera system assuming that the extrinsic parameters are known. By using natural features instead of fiducial targets, we minimize infrastructure setup costs.

With a sparse map of the calibration area, we use visual localization over a frame sequence to obtain 2D-3D feature correspondences for each camera, and estimate the camera poses. In turn, we infer an initial estimate of the extrinsic parameters and vehicle poses, and optimize these variables by minimizing the sum of weighted squared reprojection errors over all 2D-3D feature correspondences. Figure 17 illustrates the visual localization process. This structure-based calibration method is computationally efficient and runs in near real-time compared to the SLAM-based calibration method which takes a few hours to process 500 frames from a four-camera system. Prior knowledge of the 3D landmark coordinates reduces the algorithmic complexity.

Results quantifying the accuracy and repeatability of our structure-based calibration can be found in [16, 17]. We attribute the accuracy and repeatability of our structure-based calibration to two reasons: the sparse map of the calibration area is accurate, and we use many 2D-3D feature correspon-

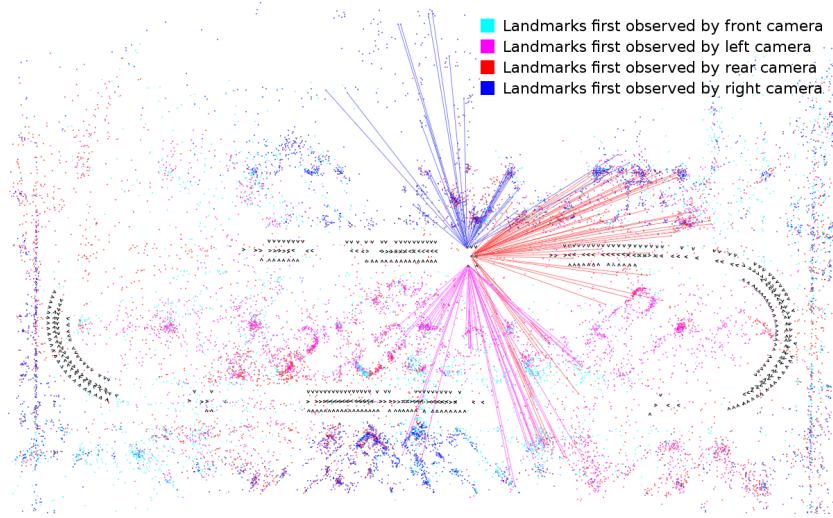


Figure 17: Points represent 3D map landmarks which are colored according to the camera they were first observed in. Small black triangles correspond to estimated camera poses. Lines connect currently observed landmarks to the current camera poses.

dences over many frames for the calibration. Our structure-based calibration allows us to calibrate many multi-camera systems with high accuracy, in a short time, and without the need for infrastructure.

6.2. Obstacle Detection

The depth maps from Section 5 are also used during automated driving in form of an obstacle detection system which facilitates online obstacle extraction [13]. In order to run in real-time we reason about the obstacles in 2D. Everything sticking out of the ground plane, which is given by the car calibration, is considered an obstacle.

Our algorithm extracts obstacles from each depth map independently. First, a 2D occupancy grid is generated from which obstacle hypotheses are extracted along rays. For each ray the closest occupied cell to the car is considered as obstacle. The final position is subsequently refined taking into account all the 3D points which fell into this specific cell.

The obstacle extraction procedure filters a lot of the noise as it combines many 3D points from the depth map into a single obstacle. The remaining noise is removed by fusing obstacle maps over time and from multiple cameras into a consistent 2D obstacle map. Most of the computation is done on GPU. The whole obstacle detection system is running in real-time (12.5Hz

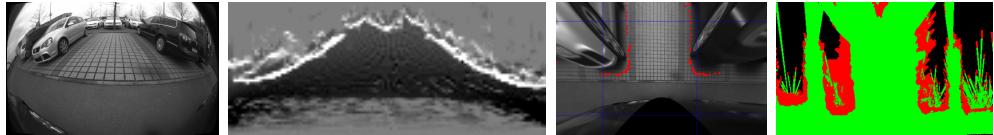


Figure 18: (left to right) reference image, occupancy grid (white occupied, black free), obstacles, fused map (green free, red occupied, black unobserved)

capturing frequency) on the left, right and front facing camera on a single computer with an NVIDIA GeForce GTX 680 GPU. The obstacle map is accurate enough for low speed driving and parking in our experiments we measure errors of less than 10cm, see [13] for the full evaluation.

7. Conclusions

In this paper, we have considered the problem of 3D perception from a multi-camera system in the context of self-driving cars. We have presented a pipeline which facilitates calibration of a multi-camera system composed of four fisheye cameras, computation of metric sparse and dense maps, map-based localization, and online obstacle detection. This system has been used successfully on the self-driving cars of the V-Charge project, demonstrating the practical feasibility of our pipeline. Due to space constraints, this paper has focused on a description of the pipeline as a whole and how the different components interact with each other. For technical details and more thorough experimental results, we refer the interested reader to the original publications referenced in each section.

The chosen camera system contains four fisheye cameras which are mounted on the car such that they have minimal overlap allowing a 360 degree coverage. While more elaborate systems with binocular pairs would be possible, our system only uses a small number of cameras to keep cost and complexity low. Using more cameras would lead to a denser sensing of the environment but would also increase complexity and cost of the system. Naturally, an interesting question is to exploit the trade-off between these factors. Unfortunately, this was not possible due to budget restrictions in the V-Charge project. Instead, a system specifically designed to be (cost) minimal was used. We show that many of the tasks required for self driving car applications at low speeds can be implemented using such a camera setup. We are able to handle the strong distortion of the fisheye lenses by using an appropriate projection model. By working directly on the fisheye images, we were

specifically able to avoid any unwarping to a pinhole projection model and thus were able to avoid a loss in the usable field-of-view.

Future work will address dynamic objects and semantic segmentation. For example, knowledge about the semantic class (e.g. tree, car or lane marking) of specific regions in the sparse or dense map could provide valuable information for localization and obstacle detection.

8. Acknowledgments

This work is supported in parts by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant #269916 (V-Charge) and 4DVideo ERC Starting Grant Nr. 210806.

References

References

- [1] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (surf). In *Computer Vision and Image Understanding (CVIU)*, 2008.
- [2] G. Carrera, A. Angeli, and A. Davison. Slam-based automatic extrinsic calibration of a multi-camera rig. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2652–2659, 2011.
- [3] A. Chambolle and T. Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145, 2011.
- [4] R. T. Collins. A space-sweep approach to true multi-image matching. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 1996.
- [5] D. A. Cox, J. Little, and D. O'Shea. *Ideals, varieties, and algorithms - an introduction to computational algebraic geometry and commutative algebra*. Springer, 1997. ISBN 978-0-387-94680-1.
- [6] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. In *Communications of the ACM*, 1981.

- [7] P. Furgale, U. Schwesinger, M. Rufli, W. Derendarz, H. Grimmett, P. Mühlfellner, S. Wonneberger, J. T. S. Rottmann, B. Li, B. Schmidt, T. N. Nguyen, E. Cardarelli, S. Cattani, S. Brüning, S. Horstmann, M. Stellmacher, H. Mielenz, K. Köser, M. Beermann, C. Häne, L. Heng, G. H. Lee, F. Fraundorfer, R. Iser, R. Triebel, I. Posner, P. Newman, L. Wolf, M. Pollefeys, S. Brosig, J. Effertz, C. Pradalier, and R. Siegwart. Toward Automated Driving in Cities using Close-to-Market Sensors, an Overview of the V-Charge Project. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 809–816, Gold Coast, Australia, 23–26 June 2013.
- [8] D. Gallup, J.-M. Frahm, P. Mordohai, Q. Yang, and M. Pollefeys. Real-time plane-sweeping stereo with multiple sweeping directions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.
- [9] D. Gallup, J.-M. Frahm, and M. Pollefeys. A heightmap model for efficient 3d reconstruction from street-level video. In *International Conference on 3D Data Processing, Visualization and Transmission (3DPVT)*, 2010.
- [10] C. Guo, F. Mirzaei, and S. Roumeliotis. An analytical least-squares solution to the odometer-camera extrinsic calibration problem. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3962–3968, 2012.
- [11] C. Häne, C. Zach, J. Lim, A. Ranganathan, and M. Pollefeys. Stereo depth map fusion for robot navigation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011.
- [12] C. Häne, L. Heng, G. H. Lee, A. Sizov, and M. Pollefeys. Real-time direct dense matching on fisheye images using plane-sweeping stereo. In *International Conference on 3D Vision (3DV)*, 2014.
- [13] C. Häne, T. Sattler, and M. Pollefeys. Obstacle detection for self-driving cars using only monocular cameras and wheel odometry. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015.
- [14] R. Haralick, D. Lee, K. Ottenburg, and M. Nolle. Analysis and solutions of the three point perspective pose estimation problem. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1991.

- [15] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.
- [16] L. Heng, M. Bürki, G. H. Lee, P. Furgale, R. Siegwart, and M. Pollefeys. Infrastructure-based calibration of a multi-camera rig. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.
- [17] L. Heng, P. Furgale, and M. Pollefeys. Leveraging image-based localization for infrastructure-based calibration of a multi-camera rig. *Journal of Field Robotics (JFR)*, 32:775–802, 2015.
- [18] A. Inc. Google self-driving car project. <https://www.google.com/selfdrivingcar>, .
- [19] T. M. Inc. Tesla autopilot. <https://www.teslamotors.com/blog/your-autopilot-has-arrived>, .
- [20] S. B. Kang, R. Szeliski, and J. Chai. Handling occlusions in dense multi-view stereo. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001.
- [21] R. Kumar, A. Ilie, J. Frahm, and M. Pollefeys. Simple calibration of non-overlapping cameras with a mirror. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–7, 2008.
- [22] P. Lebraly, E. Royer, O. Ait-Aider, C. Deymier, and M. Dhome. Fast calibration of embedded non-overlapping cameras. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 221–227, 2011.
- [23] G. H. Lee, F. Fraundorfer, and M. Pollefeys. Robust pose-graph loop-closures with expectation-maximization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013.
- [24] G. H. Lee, F. Fraundorfer, and M. Pollefeys. Structureless pose-graph loop-closure with a multi-camera system on a self-driving car. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013.

- [25] G. H. Lee, F. Fraundorfer, and M. Pollefeys. Motion estimation for a self-driving car with a generalized camera. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.
- [26] G. H. Lee, B. Li, M. Pollefeys, and F. Fraundorfer. Minimal solutions for pose estimation of a multi-camera system. In *International Symposium on Robotics Research (ISRR)*, 2013.
- [27] G. H. Lee, , and M. Pollefeys. Unsupervised learning of threshold for geometric verification in visual-based loop-closure. In *International Conference on Robotics and Automation (ICRA)*, 2014.
- [28] G. H. Lee, M. Pollefeys, and F. Fraundorfer. Relative pose estimation for a multi-camera system with known vertical direction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.
- [29] G. H. Lee, B. Li, M. Pollefeys, and F. Fraundorfer. Minimal solutions for the multi-camera pose estimation problem. In *The International Journal of Robotics Research (IJRR)*, 2015.
- [30] J. Levinson and S. Thrun. Robust vehicle localization in urban environments using probabilistic maps. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2010.
- [31] B. Li, L. Heng, K. Köser, and M. Pollefeys. A multiple-camera system calibration toolbox using a feature descriptor-based calibration pattern. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1301–1307, 2013.
- [32] H. D. Li, R. Hartley, and J. H. Kim. A linear approach to motion estimation using generalized camera models. In *IEEE Conference on Vision and Pattern Recognition (CVPR)*, 2008.
- [33] C. Mei and P. Rives. Single view point omnidirectional camera calibration from planar grids. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2007.
- [34] F. Moreno-Noguer, V. Lepetit, and P. Fua. Accurate non-iterative $O(n)$ solution to the pnp problem. In *International Conference on Computer Vision (ICCV)*, 2007.

- [35] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.
- [36] R. Pless. Using many cameras as one. In *IEEE Conference on Vision and Pattern Recognition (CVPR)*, 2003.
- [37] L. Quan and Z. D. Lan. Linear n-point camera pose determination. In *Pattern Analysis and Machine Intelligence*, 1999.
- [38] T. Sattler, B. Leibe, and L. Kobbelt. Fast image-based localization using direct 2d-to-3d matching. In *IEEE International Conference on Computer Vision (ICCV)*, 2011.
- [39] R. Siegwart, I. Nourbakhsh, and D. Scaramuzza. *Introduction to Autonomous Mobile Robots*. MIT Press, 2nd edition, 2011.
- [40] H. Stewénius, D. Nistér, M. Oskarsson, and K. Åström. Solutions to minimal generalized relative pose problems. In *Workshop on Omnidirectional Vision and Camera Networks (OMNIVIS)*, 2005.
- [41] Q. Tian and M. N. Huhns. Algorithms for subpixel registration. *Computer Vision, Graphics, and Image Processing*, 35(2):220–233, 1986.
- [42] R. Yang and M. Pollefeys. Multi-resolution real-time stereo on commodity graphics hardware. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2003.
- [43] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. G. Keller, E. Kaus, R. G. Hertrich, C. Rabe, D. Pfeiffer, F. Lindner, F. Stein, F. Erbs, M. Enzweiler, C. Knoppel, J. Hipp, M. Haueis, M. Trepte, C. Brenk, A. Tamke, M. Ghanaat, M. Braun, A. Joos, H. Fritz, H. Mock, M. Hein, and E. Zeeb. Making bertha drive - an autonomous journey on a historic route. *IEEE Intelligent Transportation Systems Magazine*, 6(2):8–20, 2014.