

INFORMATION AND COMMUNICATION TECHNOLOGIES
COLLABORATIVE PROJECT

“V-Charge”

Autonomous Valet Parking and Charging for e-Mobility

Grant Agreement Number 269916



V-CHARGE

Deliverable Number	3.2
Deliverable Name	Vehicle Localisation, Local 3D Mapping and Dynamic Object Detection
WP Number	3
Date of Preparation	15.01.2013
Date of Submission	04.02.2013
Editor/Beneficiary	Furgale/ETH Zürich

Contents

1	Introduction	2
1.1	Task Description	2
2	Overview of the contributions	2
2.1	Task 3.1: Pose Estimation using Vision (ETHZ (Lead), VW, TUBS)	2
2.2	Task 3.2: Online robot-centric depth-maps of the surrounding (RBG (lead), ETHZ)	3
2.3	Task 3.3 & 3.4: Detection, tracking and classification of dynamic objects (UNIPR (lead), ETHZ, RBG, VW)	4
3	Conclusion	4
A	Finding the exact rotation between two images independently of the translation	5
B	Using Multi-Camera Systems in Robotics: Efficient Solutions to the NPnP Problem	20
C	Online Localization for an Autonomous Car using Multiple Cameras	29
D	Online Batch Estimation for an Autonomous Car	126
E	A Patch Prior for Dense 3D Reconstruction in Man-Made Environments	171
F	Bosch Stereo Video Sensor for Driver Assistance	180
G	V-Charge Perception Framework	183
H	The V-Charge Project: Obstacles Detection, Tracking and Classification	190

1 Introduction

1.1 Task Description

The overall objective of WP3 is to develop novel approaches for perceiving the local environment from vision (mono & stereo), ultrasonic data and potentially radar, and to reliably classify and track dynamic obstacles (other cars, people). The output of this WP is then used for motion planning and vehicles control in WP4. In particular, the following system modules will be provided in WP3:

- vehicle localization
- detection of dynamic objects
- tracking and prediction of dynamic objects
- object classification (car, pedestrian, etc.)

2 Overview of the contributions

In the first 18 months of the V-Charge project, partners of the institutions involved in WP3 contributed to tasks 3.1, 3.2, 3.3 and 3.4. At the time of this writing, work on tasks 3.5 and 3.6 has just started and no contributions could be integrated in this deliverable. The following gives an overview of the contributions per tasks. All the documents referred to in the following subsections can be found in the appendix.

2.1 Task 3.1: Pose Estimation using Vision (ETHZ (Lead), VW, TUBS)

	Contributions	From	Link
1	Finding the exact rotation between two images independently of the translation, Kneip, Siegwart, Pollefeys, ECCV 2012 Conference (published)	ETHZ	A
2	Using Multi-Camera Systems in Robotics: Efficient Solutions to the NPnP Problem, Kneip, Furgale, Siegwart, ICRA 2013 Conference (submitted)	ETHZ-ASL	B
3	Online Localization for an Autonomous Car using Multiple Cameras, Neunert, Master Thesis ETHZ, 2012	ETHZ-ASL	C
4	Online Batch Estimation for an Autonomous Car, Diem, Master Thesis ETHZ, 2012	ETHZ-ASL	D

Our efforts on task 3.1 have been focused on two aspects of the visual localisation problem: the development of new algorithms to solve the pose estimation problem from image correspondences (contribution 1 and 2) and the implementation of an online localisation system on the vehicles themselves (contribution 3 and 4).

The core idea in contributions 1 and 2 is to devise novel geometric vision algorithms for increased efficiency and robustness in geometric egomotion computation. The work “Finding the exact rotation between two images independently of the translation” presented by Laurent Kneip, Roland Siegwart, and Marc Pollefeys introduces a new epipolar constraint for computing the rotation between two images independently of the translation. Against the common belief in the field of geometric vision that it is not possible to find one independently of the other, we show how this can be achieved by relatively simple two-view constraints. We use the fact that translation and rotation cause fundamentally different flow fields on the unit sphere centered around the camera. This allows to establish independent constraints on translation and rotation, and the

latter is solved using the Gröbner basis method. The rotation computation is completed by a solution to the cheirality problem that depends neither on translation, nor on feature triangulations. Notably, we show for the first time how the constraint on the rotation has the advantage of remaining exact even in the case of translations converging to zero. We use this fact in order to remove the error caused by model selection via a non-linear optimization of rotation hypotheses. We show that our method operates in real-time and compare it to a standard existing approach in terms of both speed and accuracy. While this work is very fundamental, recent work building on top of the novel epipolar constraint aims at a promising generalized relative pose computation algorithm for multi-camera systems.

In the work “Using Multi-Camera Systems in Robotics: Efficient Solutions to the NPnP Problem” proposed by Laurent Kneip, Paul Furgale, and Roland Siegwart, we introduce novel approaches to the generalized-camera absolute pose computation problem. The computation of the absolute pose of a camera with correspondences between known 3D landmarks and image plane measurements is a very traditional problem in computer vision, and has a vast number of potential applications such as 3D modeling, geometric verification in place recognition, camera calibration, camera egomotion tracking, object tracking, computer animation, augmented reality, robotics, and large-scale reconstruction. The proposed minimal and n -point solutions extend the application of this technique to non-central cameras or multi-camera systems, such as the one used in the V-Charge project for estimating the vehicle’s motion. Already existing minimal solutions require exhaustive algebraic derivations. In contrast, our novel minimal solution is solved in a straightforward manner using the Gröbner basis method. Existing n -point solutions are mostly based on iterative optimization schemes. Our n -point solution is non-iterative and—having linear complexity in the number of points—outperforms existing algorithms in terms of computational efficiency. Our results demonstrate the superior noise resilience achieved when using multi-camera configurations, and the efficiency of our algorithms. As a further contribution, we present preliminary visual odometry results on real data with a non-overlapping multi-camera configuration, including a comparison to a loosely coupled alternative.

In contribution 3 and 4, we designed and implemented an online localisation system based on a state-of-the-art multi-camera visual localization and mapping framework. As the system is targeted for production vehicles and should work in indoor and outdoor settings, only cameras and vehicle odometry are used—no expensive inertial sensors, laser sensors, or GPS. While the approach of the project is similar to common SLAM strategies, the hardware setup is unusual and presents special challenges. In contrast to previously developed systems, the V-Charge vehicles are equipped with multiple fisheye cameras which require several alterations to a common visual localization approach. We implemented a pipeline for sparse feature detection, tracking, and triangulation. This pipeline is able to provide long and robust feature tracks across the differently distorted parts of the fisheye images. A novel matching strategy is developed that allows robust keypoint association in strongly distorted images and in scenes with highly self-similar structures. The resulting feature tracks are combined with vehicle odometry in a bundle-adjustment optimization framework that jointly solves for the vehicle motion and the 3D scene structure. The contributions describe the specific development steps and evaluation on real vehicle data compared to a highly accurate DGPS/INS system.

2.2 Task 3.2: Online robot-centric depth-maps of the surrounding (RBG (lead), ETHZ)

	Contributions	From	Link
5	A Patch Prior for Dense 3D Reconstruction in Man-Made Environments, Haene, Zach, Zeisl, Pollefeys, 3DIMPVT 2012 Conference (published)	ETHZ-CVG	E
6	Bosch stereo video sensor for driver assistance Beermann, Technical Report	RBG	F
7	V-Charge Perception Framework Derendarz, Wonneberger, Nguyen, Stellmacher, Technical Report	VW	G

In task 3.2, similarly to task 3.1, our efforts focused on a theoretical contribution on one side (contribution 5) but also on the practical implementation of a local mapping system in contribution 6.

In contribution 5, the core idea behind the local mapping is that in man-made environments we face mostly planar structures. Generic smoothness priors e.g. total variation are often not sufficient to produce convincing results. Consequently, we propose a more powerful prior directly modeling the expected local surface-structure, without the need to utilize expensive methods such as higher-order MRFs. Our approach is inspired by patch-based representations used in image processing. The proposed energy can be optimized by utilizing an efficient first-order primal dual algorithm. We show how to use our prior to model the piece-wise planar 3D structure of man-made environments. This work could potentially help in very challenging environments such as indoor parking garages.

Contribution 5 describes the Bosch stereo camera used for obstacle detection on the project. The data produced by the sensor is represented in a fashion similar to a laser scanner: for each direction horizontally, distances to obstacles - if any - are provided. Beyond the ability of a laser scanner, the stereo camera is robust against changes in the vehicle's pitch angle or a rising road surface. In addition to their distance, obstacles are associated with attributes like motion, height, and visual appearance. Based on this information, static obstacles can be aggregated in a map, dynamic obstacles clustered and tracked, and their visual appearance used for classification.

As it navigates through a parking lot, the V-Charge vehicle must be able to perceive and represent static and dynamic parts of the local environment. For safe navigation it is not only important to capture the spatial form of the obstacles but also to register and even predict their motions. In terms of software architecture it is essential that different motion planning algorithms (i.e. for parking maneuvers or lane following) do not need to deal with individual raw sensor readings but rather can make use of a more abstract and centralized representation of the environment. Contribution 7 describes this software architecture developed for V-Charge to represent robot-centric maps of the environment.

2.3 Task 3.3 & 3.4: Detection, tracking and classification of dynamic objects (UNIPR (lead), ETHZ, RBG, VW)

	Contributions	From	Link
8	The V-Charge Project: Obstacles Detection, Tracking and Classification, Cardarelli, Cattani, IROS 2013 Conference (in preparation)	PARMA	H

Task 3.3 and 3.4 are presented together here as the tools and approaches involved are very closely coupled.

Contribution 8 presents the details of the detection and tracking functionalities implemented: these are designed to provide frontal and rear vehicles recognition, through the application of a classifier in specific region of the input images, determined considering a typical vehicle dimension and different detection distances.

3 Conclusion

As can be seen from the reports attached to this document, Work Package 3 of the V-Charge project is well on track with its scientific, engineering, and integration goals at the time of writing.

A Finding the exact rotation between two images independently of the translation

Authors: Kneip, Siegwart, Pollefeys

Link to [Table of Contents](#).

Finding the exact rotation between two images independently of the translation

Laurent Kneip¹, Roland Siegwart¹, and Marc Pollefeys²

¹ Autonomous Systems Lab, ETH Zurich

² Computer Vision and Geometry Group, ETH Zurich

Abstract. In this paper, we present a new epipolar constraint for computing the rotation between two images independently of the translation. Against the common belief in the field of geometric vision that it is not possible to find one independently of the other, we show how this can be achieved by relatively simple two-view constraints. We use the fact that translation and rotation cause fundamentally different flow fields on the unit sphere centered around the camera. This allows to establish independent constraints on translation and rotation, and the latter is solved using the Gröbner basis method. The rotation computation is completed by a solution to the cheirality problem that depends neither on translation, nor on feature triangulations. Notably, we show for the first time how the constraint on the rotation has the advantage of remaining exact even in the case of translations converging to zero. We use this fact in order to remove the error caused by model selection via a non-linear optimization of rotation hypotheses. We show that our method operates in real-time and compare it to a standard existing approach in terms of both speed and accuracy.

1 Introduction

Numerous works over the past decade proved the feasibility of doing robust structure-from-motion with a single camera only. Some important examples are given by Nistér et al. [1], Davison et al. [2] and Klein and Murray [3]. The biggest problem from a geometric point of view has been identified to be the robust estimation of the rotation between successive camera frames. The effect of certain small translations and rotations on the displacement of the features in the image plane can be very similar and the problem of disambiguation is even amplified if the distribution of the features in the image plane is too inhomogenous.

Klein and Murray [3] find the rotation between two frames by warping and rotating *small blurry images* of the camera frames and then minimizing the sum of squared differences in between them. Another approach has recently been presented by Kneip et al. [4], who solve this problem by taking short-term integrals of gyroscopic signals of an additional inertial measurement unit into account. However, both approaches only deliver an approximate value for the rotation between two frames. In this paper, we present an exact and robust

geometric formulation for an independent computation of the rotation between two camera frames.

The subject of this paper is clearly related to one of the most fundamental and traditional problems in geometric vision, namely the determination of the relative pose between two images. The input information is given in form of point correspondences between the two images, or—in the calibrated case—unit vectors pointing from each viewpoint to the jointly observed 3D points. The goal is to find the translation and rotation between the two frames. Of high interest are solutions that use a minimum number of points and can thus be employed in robust hypothesize-and-test schemes such as the RANSAC approach [5]. The most important minimal solutions are the 5-point solvers by Nistér [6], Stewénius et al. [7], and Kukelova et al. [8] and the famous 8-point solver by Longuet-Higgins [9]. The first solution was presented in 1913 by Kruppa [10]. The main problem with all these approaches—apart from potentially degenerate structure configurations and multiplicity of solutions—is that they mix the parameters for translation and rotation and thus become numerically intractable in the case of vanishing translation magnitudes. This issue notably applies to any parametrization that involves the fundamental or essential matrix between the two images. The way this problem has been handled in the literature consists of applying a model selection criterium that decides whether the displacement has too small translation or not and then solving for pure rotation in case it does. This has for instance been presented by Torr et al. [11]. While this approach achieves robust behavior, it has the downside of not delivering exact rotations in the case of sufficiently small translations. Moreover, it is impossible to reduce this error in a two-view batch optimization. Bundle adjustment depends on feature triangulations, which are undefined if the parallax is neglected and simply set to zero.

In 2009, Kalantari et al. [12] presented a new direct parametrization in terms of translation and rotation. Their method results in a system of 6 equations in 6 unknowns and a Gröbner basis with 40 base monomials. They claim that their approach stays robust in the case of zero translation, however without presenting any quantitative results in this direction. They also solve the Gröbner basis online, which is not the most efficient way regarding the fact that the path of the algorithm actually remains constant for one specific problem. In 2010, Lim et al. [13] present a solution for decoupled translation and rotation computation. However, their method imposes constraints on the distribution of the features on the unit sphere and hence does not work in the general case.

In this paper, we present new general epipolar constraints that allow the independent computation of rotation and translation between two frames based on the individual characteristics of the resulting optical flow on the unit sphere. We obtain a system of equations that allows to compute the rotation between two frames generally and independently of the translation using a minimum number of points. Notably, we also show how the rotation constraint stays numerically robust even in the case of vanishing translation magnitudes and hence allows for an exact computation of the rotation in any case. Section 2 shows the

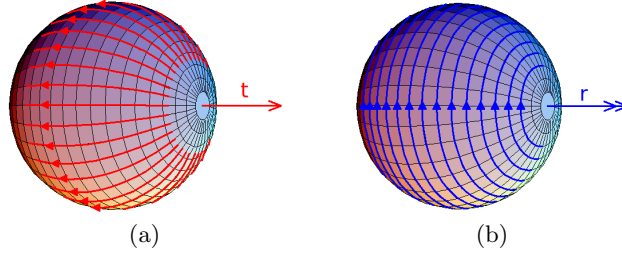


Fig. 1. Optical flow of features caused by camera translation (a) and rotation (b).

derivation of computationally independent constraints on translation and rotation. Section 3 illustrates how the rotation can be solved using the Gröbner basis method. Section 4 presents a non-linear refinement of the solutions and notably how our constraint allows for an exact computation of the rotation for any given translation. Section 5 then presents a concise evaluation and comparison of our algorithm in terms of accuracy, noise resilience and execution time.

2 Translation independent rotation constraint

2.1 Independence of rotation and translation computation

A good illustration of why the rotation and translation of a camera displacement can be separated is given by the optical flow on the unit sphere. As shown in Figure 1, the optical flow caused by a translation consists of the shortest field lines emerging from/ending at the intersection points between the sphere and the translation direction. The rotational optical flow is fundamentally different and consists of field lines contained in parallel planes of which the rotation vector is a normal vector.

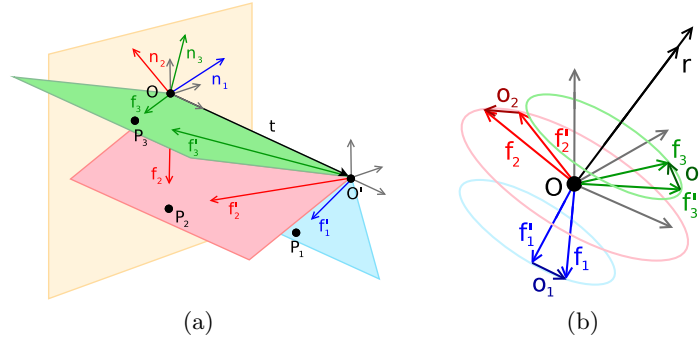


Fig. 2. Geometric relationships of unit feature observation vectors in the case of pure translation (a) and pure rotation (b).

To the end of obtaining implicit constraints on rotation and translation, we introduce the unit feature vectors $\mathbf{f}_i$ and $\mathbf{f}'_i$ describing the feature observations from viewpoint 1 and 2, respectively. Figure 2 shows the influence of a pure translation or a pure rotation on the feature observations. In the case of a pure translation, each correspondence $(\mathbf{f}_i, \mathbf{f}'_i)$ spans a so-called epipolar plane. Moreover, all of the epipolar planes intersect in the line defined by the translation vector. In other words, the normal vectors of the epipolar planes all need to be contained in the normal plane of the translation vector. A normal vector of each epipolar plane is in this special case easily given by $\mathbf{n}_i = \mathbf{f}_i \times \mathbf{f}'_i$. The constraint for a purely translational optical flow thus translates to all cross-products $\mathbf{f}_i \times \mathbf{f}'_i$ being coplanar. In the case of a pure rotation, the situation is even easier. Since the field lines are all contained in parallel planes, it immediately follows that the constraint for a pure rotation translates to all optical flow vectors $\mathbf{o}_i = \mathbf{f}_i - \mathbf{f}'_i$ being coplanar as well.

The idea for independent rotation and translation computation finally reads as follows:

- *rotation* is correctly compensated if and only if the feature correspondences $(\mathbf{f}_i, c_{\mathbf{R}}(\mathbf{f}'_i))$ fulfill the properties of a *purely translational optical flow*, and
- *translation* is correctly compensated if and only if the feature correspondences $(\mathbf{f}_i, c_{\mathbf{t}}(\mathbf{f}'_i))$ fulfill the properties of a *purely rotational optical flow*,

where $c_{\mathbf{R}}(\cdot)$ and $c_{\mathbf{t}}(\cdot)$ are the functions that compensate for rotation and translation, respectively. If $\mathbf{R}$ denotes the rotation from viewpoint 2 to viewpoint 1, the rotation-compensated observations from viewpoint 2 become $c_{\mathbf{R}}(\mathbf{f}'_i) = \mathbf{R}\mathbf{f}'_i$. In this case we need to have a purely translational optical flow and thus all vectors $\mathbf{f}_i \times \mathbf{R}\mathbf{f}'_i$ need to be coplanar. The minimum number of vectors for expressing coplanarity is three and a simple way to encode coplanarity/linear dependency is given by the determinant of these three vectors being zero. A translation independent constraint on the rotation is finally given by

$$|(\mathbf{f}_1 \times \mathbf{R}\mathbf{f}'_1) \ (\mathbf{f}_2 \times \mathbf{R}\mathbf{f}'_2) \ (\mathbf{f}_3 \times \mathbf{R}\mathbf{f}'_3)| = 0. \quad (1)$$

If $\mathbf{t}'$ denotes the translation from viewpoint 2 to viewpoint 1 seen from viewpoint 2, the translation compensated observations from viewpoint 2 become $c_{\mathbf{t}'}(\mathbf{f}'_i) = \frac{\eta'_i \mathbf{f}'_i - \mathbf{t}'}{\|\eta'_i \mathbf{f}'_i - \mathbf{t}'\|}$, with η'_i denoting the depths of the features seen from viewpoint 2. This time we need to have a purely rotational optical flow, meaning that all vectors $\mathbf{f}_i - \frac{\eta'_i \mathbf{f}'_i - \mathbf{t}'}{\|\eta'_i \mathbf{f}'_i - \mathbf{t}'\|}$ need to be coplanar. Following a similar argumentation to the one previously developed, a rotation independent constraint on the translation is given by

$$\left| \left(\mathbf{f}_1 - \frac{\eta'_1 \mathbf{f}'_1 - \mathbf{t}'}{\|\eta'_1 \mathbf{f}'_1 - \mathbf{t}'\|} \right) \left(\mathbf{f}_2 - \frac{\eta'_2 \mathbf{f}'_2 - \mathbf{t}'}{\|\eta'_2 \mathbf{f}'_2 - \mathbf{t}'\|} \right) \left(\mathbf{f}_3 - \frac{\eta'_3 \mathbf{f}'_3 - \mathbf{t}'}{\|\eta'_3 \mathbf{f}'_3 - \mathbf{t}'\|} \right) \right| = 0 \quad (2)$$

It is interesting to see that the depths of the features only appear in the translation constraint. The independent constraint on the rotation (1) appears fairly compact. It is the main subject of the remaining of the paper and we name it the *epipolar plane normal coplanarity constraint*.

2.2 Constraining the rotation

A rotation encodes 3 degrees of freedom. We thus need at least three epipolar plane normal coplanarity constraints in order to fully constrain the rotation. Using two additional features $\mathbf{f}_4$ and $\mathbf{f}_5$, we obtain the following system of equations to calculate the rotation

$$\begin{cases} |(\mathbf{f}_1 \times \mathbf{R}\mathbf{f}'_1) (\mathbf{f}_2 \times \mathbf{R}\mathbf{f}'_2) (\mathbf{f}_3 \times \mathbf{R}\mathbf{f}'_3)| = 0 \\ |(\mathbf{f}_1 \times \mathbf{R}\mathbf{f}'_1) (\mathbf{f}_2 \times \mathbf{R}\mathbf{f}'_2) (\mathbf{f}_4 \times \mathbf{R}\mathbf{f}'_4)| = 0 \\ |(\mathbf{f}_1 \times \mathbf{R}\mathbf{f}'_1) (\mathbf{f}_2 \times \mathbf{R}\mathbf{f}'_2) (\mathbf{f}_5 \times \mathbf{R}\mathbf{f}'_5)| = 0 \end{cases} \quad (3)$$

In addition to these equations, there might be—depending on the employed parametrization—additional constraints enforcing the rotation matrix to actually be a rotation matrix. As illustrated in the following sections, the choice of the parametrization has a huge impact on the complexity of the solution.

2.3 The case of zero translation

The constraints on the rotation (3) turn out to be still valid even if the translation is zero. For a correct rotation compensation, the entities $\mathbf{f}_i \times \mathbf{R}\mathbf{f}'_i$ then ideally become zero, which does however not change the fact that the determinant constraint is zero in this case. Notably, since the mentioned entities deviate from zero if a wrong rotation matrix is chosen—even in the case of pure rotation—the determinant is zero if and only if the rotation is correctly compensated. The only exception is given with degenerate feature configurations. For instance, if all the features are situated along the equator, we would have an optical flow that could be explained by both our rotation-only and translation-only constraints. A similar development than above can be done for the translation constraint.

3 5-point minimal solution

3.1 Finding a Gröbner basis

Despite the compact look, equation system (3) is arduous to solve. It is a multivariate polynomial equation system commonly solved via the Gröbner basis method. A good introduction to the approach can be found in [14]. The method consists of defining a monomial ordering over the polynomial terms and then iteratively generating and reducing new polynomials inside the ideal (the so-called S-polynomials) until a set of polynomials with certain desired criteria w.r.t. solvability is obtained. This method has been applied to numerous minimal problems in geometric vision, among which the most well known approaches certainly are represented by the classical 5-point essential matrix solutions of Nistér [6] and Stewénius [7].³ As a matter of fact, the path followed by the algorithm remains

³ The solution of Nistér actually turns out to be equivalent to computing a lexicographical Gröbner basis instead of a graded-reverse lexicographical Gröbner basis as done by Stewénius. It results in a univariate 10th degree polynomial, which can be solved substantially faster than a 10x10 action matrix.

constant for one specific problem and needs to be resolved only once exactly, which means outside the algebraic field of real numbers. Hence the way such polynomial systems are solved is by choosing random coefficients in a prime field and then doing the computation therein. The path of the solution can be *traced* offline in the prime field and afterwards applied online to the effective feature-dependent coefficients. The final algorithm is fast since we do no longer need to check for all polynomial reductions, but directly generate and reduce only the necessary S-polynomials.

The computation of a Gröbner basis can be extremely long and the complexity depends to a large extent on the initial parametrization of the problem. It is influenced by the following factors: 1) the order of the equations; 2) the number of equations; 3) the number of unknowns; and 4) the chosen monomial ordering. Using our custom-made Gröbner basis computation software (~ 6000 lines of code), we tried out many different rotation matrix parametrizations as for instance Cayley [15], quaternion, and Thompson [16] rotations. However, due to the increasing order of the equations, our conclusion is that this system is best solved using the standard rotation matrix parametrization $\mathbf{R} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}$. Moreover, it is best to increase the number of equations as much as possible and thus take into account as many epipolar plane normal coplanarity constraints as possible. Finally, it is also best to formalize the constraints on the rotation matrix in terms of multiple quadratic constraints and leave out the third order determinant constraint.

$$\left\{ \begin{array}{l} |(\mathbf{f}_1 \times \mathbf{R}\mathbf{f}'_1)(\mathbf{f}_2 \times \mathbf{R}\mathbf{f}'_2)(\mathbf{f}_3 \times \mathbf{R}\mathbf{f}'_3)| = 0 \\ |(\mathbf{f}_1 \times \mathbf{R}\mathbf{f}'_1)(\mathbf{f}_2 \times \mathbf{R}\mathbf{f}'_2)(\mathbf{f}_4 \times \mathbf{R}\mathbf{f}'_4)| = 0 \\ |(\mathbf{f}_1 \times \mathbf{R}\mathbf{f}'_1)(\mathbf{f}_2 \times \mathbf{R}\mathbf{f}'_2)(\mathbf{f}_5 \times \mathbf{R}\mathbf{f}'_5)| = 0 \\ |(\mathbf{f}_1 \times \mathbf{R}\mathbf{f}'_1)(\mathbf{f}_3 \times \mathbf{R}\mathbf{f}'_3)(\mathbf{f}_4 \times \mathbf{R}\mathbf{f}'_4)| = 0 \\ |(\mathbf{f}_1 \times \mathbf{R}\mathbf{f}'_1)(\mathbf{f}_3 \times \mathbf{R}\mathbf{f}'_3)(\mathbf{f}_5 \times \mathbf{R}\mathbf{f}'_5)| = 0 \\ |(\mathbf{f}_1 \times \mathbf{R}\mathbf{f}'_1)(\mathbf{f}_4 \times \mathbf{R}\mathbf{f}'_4)(\mathbf{f}_5 \times \mathbf{R}\mathbf{f}'_5)| = 0 \\ |(\mathbf{f}_2 \times \mathbf{R}\mathbf{f}'_2)(\mathbf{f}_3 \times \mathbf{R}\mathbf{f}'_3)(\mathbf{f}_4 \times \mathbf{R}\mathbf{f}'_4)| = 0 \\ |(\mathbf{f}_2 \times \mathbf{R}\mathbf{f}'_2)(\mathbf{f}_3 \times \mathbf{R}\mathbf{f}'_3)(\mathbf{f}_5 \times \mathbf{R}\mathbf{f}'_5)| = 0 \\ |(\mathbf{f}_2 \times \mathbf{R}\mathbf{f}'_2)(\mathbf{f}_4 \times \mathbf{R}\mathbf{f}'_4)(\mathbf{f}_5 \times \mathbf{R}\mathbf{f}'_5)| = 0 \\ |(\mathbf{f}_3 \times \mathbf{R}\mathbf{f}'_3)(\mathbf{f}_4 \times \mathbf{R}\mathbf{f}'_4)(\mathbf{f}_5 \times \mathbf{R}\mathbf{f}'_5)| = 0 \\ r_{11}^2 + r_{12}^2 + r_{13}^2 = 1 \\ r_{11}r_{21} + r_{12}r_{22} + r_{13}r_{23} = 0 \\ r_{11}r_{31} + r_{12}r_{32} + r_{13}r_{33} = 0 \\ r_{21}^2 + r_{22}^2 + r_{23}^2 = 1 \\ r_{21}r_{31} + r_{22}r_{32} + r_{23}r_{33} = 0 \\ r_{31}^2 + r_{32}^2 + r_{33}^2 = 1 \\ r_{21}r_{32} - r_{22}r_{31} - r_{13} = 0 \\ r_{12}r_{31} - r_{11}r_{32} - r_{23} = 0 \\ r_{11}r_{22} - r_{12}r_{21} - r_{33} = 0 \\ r_{22}r_{33} - r_{23}r_{32} - r_{11} = 0 \\ r_{13}r_{32} - r_{12}r_{33} - r_{21} = 0 \\ r_{12}r_{23} - r_{13}r_{22} - r_{31} = 0 \\ r_{11}^2 + r_{21}^2 + r_{31}^2 = 1 \\ r_{11}r_{12} + r_{21}r_{22} + r_{31}r_{32} = 0 \\ r_{11}r_{13} + r_{21}r_{23} + r_{31}r_{33} = 0 \\ r_{12}^2 + r_{22}^2 + r_{32}^2 = 1 \\ r_{12}r_{13} + r_{22}r_{23} + r_{32}r_{33} = 0 \\ r_{13}r_{21} - r_{11}r_{23} - r_{32} = 0 \\ r_{23}r_{31} - r_{21}r_{33} - r_{12} = 0 \\ r_{11}r_{33} - r_{13}r_{31} - r_{22} = 0 \end{array} \right. \quad (4)$$

The final parametrization is given with equation system (4). It consists of 10 cubic coplanarity constraints and 20 quadratic constraints on the rotation matrix and is solvable—using the grevlex monomial ordering and $r_{33} > r_{32} > r_{31} > r_{23} > r_{22} > r_{21} > r_{13} > r_{12} > r_{11}$ —via 36 S-polynomial reductions only. We finally obtain a Gröbner basis in the 20 base monomials $r_{12}^2, r_{11}r_{33}, r_{11}r_{32}, r_{11}r_{31}, r_{11}r_{23}, r_{11}r_{22}, r_{11}r_{21}, r_{11}r_{13}, r_{11}r_{12}, r_{11}^2, r_{33}, r_{32}, r_{31}, r_{23}, r_{22}, r_{21}, r_{13}, r_{12}, r_{11}, 1$. The SVD of the action matrix then finally leads to 20 direct solutions for the rotation matrix, which is reasonable since each essential matrix actually represents two possible rotation matrices after decomposition. Our Gröbner basis tool basically combines the results of Buchberger [17], Gebauer and Möller [18], and Giovini et al. [19].

The code we extract from the trace (~ 8000 lines) operates on a large matrix in a similar way to Faugère's F4 [20] or the code extracted from Kukulova's framework [21], however substantially faster since operating on a much smaller matrix. This is achieved by the following modifications:

- The number of columns is reduced by only representing the monomials that effectively appear along the computation.
- When creating new S-polynomials, no copies of the pair of generators multiplied by the subsidiary monomials to the LCM are instantiated.
- When creating new S-polynomials, the canceling leading term is left out immediately.
- When reducing S-polynomials, no copies of the reductors multiplied by the quotient term are instantiated.
- We recursively backtrack the S-polynomials that are effectively necessary for the final basis and generate only those.

The operation is supported by vector-functions that access the right generator and reductor coefficients for combining terms without actually multiplying the polynomial by the subsidiary monomials to the LCM (Least Common Multiple). Our matrix is 66×197 and the Gröbner basis is computed in less than 0.3ms.

3.2 Selection of the right solution

The 20 solutions of the action matrix SVD exist in the complex domain and only those with a sufficiently low imaginary part are interesting. Wrong solutions are easily rejected by thresholding the magnitude of the imaginary parts of the singular values from the SVD. The next consistency check then consists in verifying if the determinants of the rotation matrices equal to one, which again allows to purge wrong solutions. Moreover, we back-substitute the remaining rotation matrices into the original constraints and put a threshold on the resulting error. Another way of identifying wrong solutions is given by solving the cheirality ambiguity, as we will explain next.

From the essential matrix decomposition, we know that the obtained rotation matrices may appear in pairs of reflected matrices. The same accounts for the translations. Figure 3 illustrates how this finally leads to 4 possible $(\mathbf{R}, \mathbf{t})$ -pairs for each feature triangulation. The classical way of solving this problem is to triangulate features and then check whether they lie in front or behind the camera. Since we only have rotation matrices, we focus on a constraint that allows us to effectuate the disambiguation without actually deriving the translation or doing feature-triangulation. First of all, it is important to notice that the criterium whether a certain point lies in front or behind the camera plane is not very accurate. For an omnidirectional camera for instance, the front and the back of a camera does not tell anything about the visibility of the feature. We therefore change this criterium to something more meaningful, namely check whether the

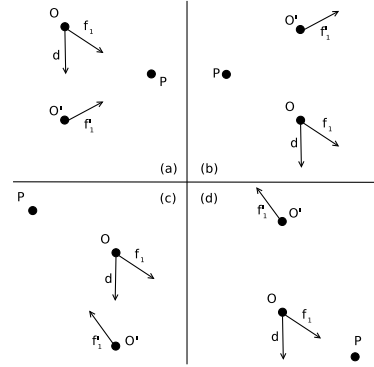


Fig. 3. Ambiguity for translation and rotation known as cheirality.

3D point lying on a line defined by the feature vector also lies on the *beam* defined by the feature vector. Secondly, looking at Figure 3, we notice that (a) and (b) have the same rotation, and (c) and (d) as well. The difference between both cases only consists in the sign of the translation. For (a) and (b), the 3D point is lying either on both or none of the beams. For (c) and (d), the 3D point is lying on one of the beams only. $\mathbf{d}$ represents a direction vector and therefore it is equal to either plus or minus the translation vector. The direction vector can be obtained by any cross-product of two epipolar plane normal vectors and thus we have

$$\mathbf{d} = \pm \mathbf{t} = \mathbf{n}_1 \times \mathbf{n}_2 = (\mathbf{f}_1 \times \mathbf{R}\mathbf{f}'_1) \times (\mathbf{f}_2 \times \mathbf{R}\mathbf{f}'_2). \quad (5)$$

For the correct rotation matrix (cases (a) and (b)), we observe that the cross-product between the direction vector $\mathbf{d}$ and $\mathbf{f}_1$ needs to point to the same direction than the cross-product between the direction vector $\mathbf{d}$ and $\mathbf{R}\mathbf{f}'_1$. In other words, the dot-product of both cross-products must be positive. This translates into the following inequality constraint on each rotation matrix

$$(\mathbf{d} \times \mathbf{f}_1) \cdot (\mathbf{d} \times \mathbf{R}\mathbf{f}'_1) > 0$$

$$\Rightarrow \{(\mathbf{f}_1 \times \mathbf{R}\mathbf{f}'_1) \times (\mathbf{f}_2 \times \mathbf{R}\mathbf{f}'_2) \times \mathbf{f}_1\} \cdot \{(\mathbf{f}_1 \times \mathbf{R}\mathbf{f}'_1) \times (\mathbf{f}_2 \times \mathbf{R}\mathbf{f}'_2) \times \mathbf{R}\mathbf{f}'_1\} > 0 \quad (6)$$

We finally found a constraint that allows the disambiguation of rotation matrices without having to derive the translation or 3D point information. Note that any pair of features can be used for the disambiguation.

3.3 The case of zero translation

Despite the fact that our original rotation constraint is still valid for zero translation, the computation still deteriorates when the translation converges to zero. This phenomenon can be explained as follows. When the translation approaches zero, the 20 solutions of the 5-point algorithm converge to a single solution, which is natural since the rotation in the case of zero translation is uniquely defined by two features already. The side-effect is that—when following the fixed trace of the Gröbner basis computation—we obtain similar polynomials and hence the reductions result in almost zero coefficients in the middle of the computation. The solution parametrization having too high complexity for this specific case, the path finally becomes numerically unstable.

We handle this problem similar to Torr et al. [11] and perform model selection. We try to find a suitable rotation using two features only and accept it in case the unrotated features from view-point 2 turn out to be close enough to the features from view-point 1. Accepting the origin as a third virtual point and following the point-set alignment approach presented in [22], this can easily be done via computing the SVD

$$\mathbf{U} \cdot \mathbf{D} \cdot \mathbf{V}^* = \text{SVD}(\sum_{i=1}^3 (\mathbf{f}'_i - \bar{\mathbf{f}}') \cdot (\mathbf{f}_i - \bar{\mathbf{f}})^t) \Rightarrow \mathbf{R} = \mathbf{U} \cdot \mathbf{V}^t, \quad (7)$$

and then checking the magnitude of the differences $\mathbf{f}_i - \mathbf{R}\mathbf{f}'_i$. The drawback of this approach is that there remains a small error in the computation of the rotation for small translations. Besides, since the 20 local minima are very close to each other, this error can not be corrected via a non-linear optimization with 5 features. Even a two-view batch optimization would be infeasible since depending on feature triangulations, which are undefined if neglecting the parallax and setting it to zero.

4 Non-linear refinement

The final approach to remove the error caused by model selection and find a unique and exact rotation consists of taking more features into account and performing non-linear optimization over all resulting epipolar plane normal coplanarity constraints. In this paper, we will focus on the minimal variant of taking only one additional feature into account, although the accuracy of the optimization can obviously be improved by increasing this number. The minimization can in principle be extended to any number of features. The number of resulting constraints for 6 points is given by the combination $\binom{6}{3} = \frac{6!}{(6-3)!3!} = 20$. The minimization is carried out over the Cayley parameters of a rotation matrix [15]. While this parametrization was unsuited for the Gröbner basis computation, it turns out to provide good properties for an iterative optimization scheme, namely symmetric parametrization in function of only 3 parameters and absence of additional constraints on the rotation matrix.

As illustrated in Figure 4(a), the norm of the errors of the constraints behaves non-linearly. Moreover, as shown in Figure 4(b)—a closeup of the region around the global minimum—the manifold is not free of local minima. However, it equals to zero for the correct rotation values only (in this case $(0\ 0\ 0)^t$). The figure only shows the error over pitch and yaw angle variations, but the behavior for roll angle variations is comparable. The reason for the displayed local minimum is actually given by rotations that attempt to explain the translation as well (in the illustrated example we have a translation along x and the effect

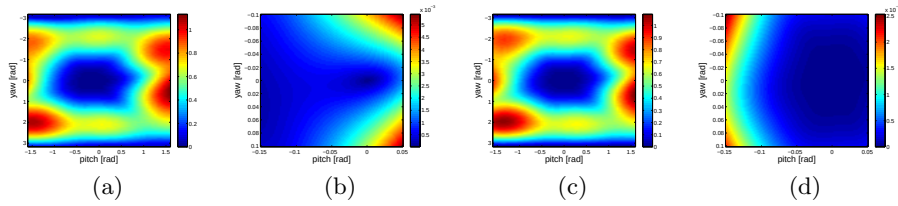


Fig. 4. (a): Norm of the errors of all epipolar plane normal coplanarity constraints for 6 features and pitch and yaw deviations. (b): Closeup of the region around the global minimum. (c): Norm of the error for zero translations. (d): Same closeup of the region around the global minimum for zero translation.

is close to the one of a pitch angle variation (rotation around y)). The main advantage of our constraint is shown in Figure 4(c): the cost-function remains robust for zero translation and thus paves the way for computing exact rotations even in the case of translations down to zero. Moreover, Figure 4(d) shows that in the event of a pure rotation, the mentioned local minimum even disappears. On the other hand, this means that for very small translations, we have local minima very close to the absolute minimum. A good minimization is therefore only possible by taking additional measures not to end up in a local minimum.

Taking the remaining rotations from the 5-point algorithm (or the unique rotation from two features in case of a small translation) as initial points for the optimization, we can finally identify a unique and exact rotation between the two view-points with only 6 points. The entire method can be summarized as follows:

- Select 2 of the 6 features and try to compute the rotation as described in Section 3.3
 - *If* the translation is small enough, retain this unique hypothesis
 - *Else*, compute the rotation hypotheses using the 5-point Gröbner basis method and reject as many as possible as described in Section 3.2
- Finally, optimize all found hypotheses using nonlinear refinement over all 6 points and select the solution with least squares error.

5 Results

The algorithm we present is thoroughly tested by means of synthetic data and compared to Stewénius’ [7] 5-point solution for computing the rotation and translation between two images. This is a clean, accurate and popular approach to epipolar geometry computation. The synthetic data consist of 1’000 3D points that are evenly distributed below the fixed downlooking camera at viewpoint 1, with depths varying uniformly between 4 and 8 in the vertical direction. For each test, we then select a randomly displaced viewpoint 2. The magnitude of the translation is uniformly distributed between 0 and 2 in order to ensure that the position of viewpoint 2 stays within a reasonable distance of the structure. For each experimental run, synthetic 2D-2D correspondences are then created by randomly selecting 6 points from the entire point set and reprojecting them on the unit sphere of our two viewpoints. Depending on the experiment, a different level of white Gaussian noise ranging from 0 to 2 pixels is then added to the measurements by assuming a pinhole camera model and an effective focal length of $f = 800$ pixel.

5.1 Noise resilience

In our first experiment, we examine the influence of noise on the accuracy of the solution. To this end, we add Gaussian noise varying from 0 to 2 pixels to our measurements and execute 1000 runs for each level. Figure 5 indicates the

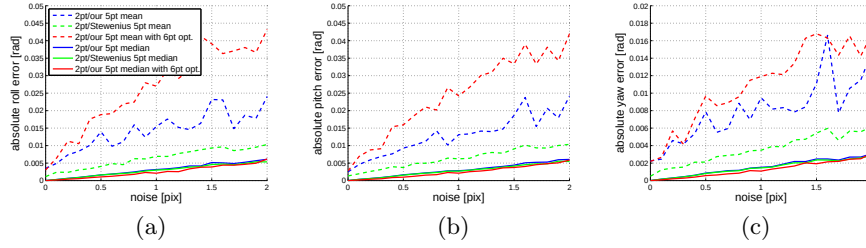


Fig. 5. Mean and median errors of roll (a), pitch (b) and yaw (c) angles for our 5-point, Stewenius’ 5-point and our non-linear 6-point solutions.

resilience of our new 5-point algorithm and the one by Stewenius, both with model selection and 2-point rotation computation in case of small translations. We also display the results after 6-point non-linear refinement of our solutions. We can see that—due to the complexity of the computation—our 5-point algorithm has lower resilience to noise in comparison to Stewenius’ solution. We can also see that the results after our non-linear refinement show highest mean absolute errors in all three rotational dimensions. The reason why the performance is degrading is related to bad rotation hypotheses under noise and hence convergence into wrong local minima. However, having a look at the median values of the error, we note that the result of our 6-point non-linear refinement still performs best. This is underlined by the fact that our 6-point non-linear refinement is able to identify a unique rotation, whereas the correct solution for the 5-point algorithms has been selected each time by hand. We also verified that our non-linear 6-point refinement remains robust in the special case of planar structure.

5.2 Behavior for small translations

We now proceed to analyzing the biggest advantage of our novel epipolar constraint. In order to investigate the influence of small translations on the accuracy of the algorithm, we run specific tests where the magnitude of the translation is varying from 0 to 3% of the average scene depth from viewpoint 1, namely 0.18. As mentioned in Section 3.3, we have to differentiate between cases of small translation and cases where the parallax is sufficiently high for running the 5-point algorithm. The same accounts for Stewenius’ 5-point algorithm and notably every minimal solution that employs the essential matrix parametrization including the 8-point algorithm. Our proposed solution consists in taking the hypothesized rotations as starting points for our non-linear optimization. As shown in Figure 6, this yields very good results and even completely removes the error for small translations committed around the model selection boundary. The figure also displays the errors of pure 2-point and 5-point rotation computation without model selection. It proves that our novel epipolar constraint on the rotation only is indeed able to compute an exact rotation for small or even

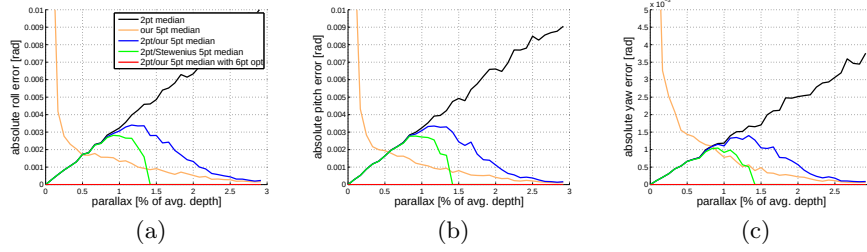


Fig. 6. Mean and median roll(a), pitch (b) and yaw (c) errors for small translation magnitudes.

zero translation magnitudes and does not suffer from numerical instability in this situation.

5.3 Performance in terms of accuracy and execution time

We also analyse the numerical accuracy and execution times of our new 6-point and Stewenius’ 5-point methods. Figure 7 shows that—in the noise-free case—the numerical accuracy of our non-linear least squares solution is superior to Stewenius’ least-squares solution by roughly one order of magnitude.

7(c) then finally shows the drawback of our method. The non-linear global minimization is pretty expensive and consumes 86% of the execution time. The overall execution time required by our method is currently still five times higher than the average time consumed by Stewenius’ algorithm.

6 Conclusion

We have presented in this paper a new epipolar constraint for computing the rotation between two images independently of the translation. While the minimal 5-point computation of the rotation still suffers from the same model selection related inaccuracies as already existing approaches to epipolar geometry computation, the subsequent non-linear refinement of the hypotheses reduces the median error down to zero, even for vanishing translation magnitudes. The entire method thus represents a solution for computing a unique exact rotation and does not suffer from solution multiplicity or inaccuracies caused by the essential matrix parametrization. We also verified that our method does not suffer from planar degeneracies. The minimization is extendable to an arbitrary number of points, and can hence be used in any case where errors around the model selection boundary occur.

Once the rotation problem is solved, the translation between two viewpoints immediately follows as a simple two-point solution. Future work now consists in improving the efficiency of the algorithm and integrating it into a RANSAC scheme for continuous real-time frame-to-frame rotation estimation. The final

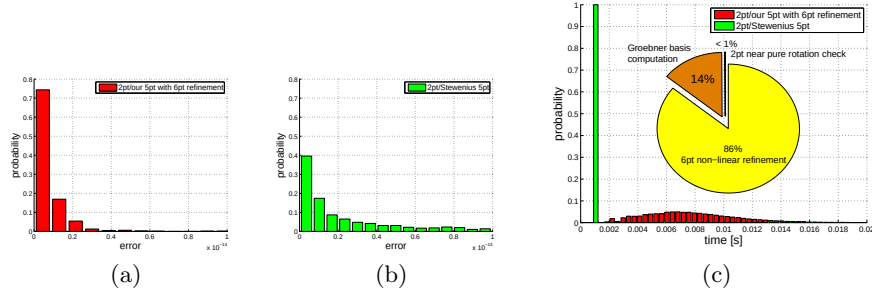


Fig. 7. Probability distribution over numerical accuracy for our proposed (a) and Stewenius’ method (b). (c) shows the probability distribution over execution time for both methods. The pie chart shows the relative time consumption of the different parts of the algorithm.

goal is to check the impact of robust and translation decoupled rotation computation on structure-from-motion algorithms.

References

1. Nistér, D., Naroditsky, O., Bergen, J.: Visual odometry. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Washington, DC, USA (2004) 652–659
2. Davison, A., Reid, D., Molton, D., Stasse, O.: MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* **26**(6) (2007) 1052–1067
3. Klein, G., Murray, D.: Parallel tracking and mapping for small AR workspaces. In: Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR), Nara, Japan (2007)
4. Kneip, L., Chli, M., Siegwart, R.: Robust real-time visual odometry with a single camera and an IMU. In: Proceedings of the British Machine Vision Conference (BMVC), Dundee, Scotland (2011)
5. Fischler, M., Bolles, R.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM* **24**(6) (1981) 381–395
6. Nistér, D.: An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* **26**(6) (2004) 756–777
7. Stewenius, H., Engels, C., Nistér, D.: Recent developments on direct relative orientation. *ISPRS Journal of Photogrammetry and Remote Sensing* **60**(4) (2006) 284–294
8. Kukelova, Z., Bujnak, M., Pajdla, T.: Polynomial Eigenvalue solutions to the 5-pt and 6-pt relative pose problems. In: Proceedings of the British Machine Vision Conference (BMVC), Leeds, UK (2008)
9. Longuet-Higgins, H.: Readings in computer vision: issues, problems, principles, and paradigms. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1987)

10. Kruppa, E.: Zur Ermittlung eines Objektes aus zwei Perspektiven mit innerer Orientierung. Sitzgsber. Akad. Wien, Math. Naturw. Abt., IIa. **122** (1913) 1939–1948
11. Torr, P., Fitzgibbon, A., Zisserman, A.: Maintaining multiple motion model hypotheses over many views to recover matching and structure. In: Proceedings of the International Conference on Computer Vision (ICCV), Bombay, India (1998) 485–491
12. Kalantari, M., Jung, F., Guedon, J.P., Paparoditis, N.: The five points pose problem: A new and accurate solution adapted to any geometric configuration. In: Proceedings of the Pacific Rim Symposium on Advances in Image and Video Technology, Tokyo, Japan (2009) 215–226
13. Lim, J., Barnes, N., Li, H.: Estimating relative camera motion from the antipodal-epipolar constraint. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) **32**(10) (2010) 1907–1914
14. Cox, D.A., Little, J., O’Shea, D.: Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra, 3/e (Undergraduate Texts in Mathematics). Springer-Verlag New York, Inc., Secaucus, NJ, USA (2007)
15. Cayley, A.: About the algebraic structure of the orthogonal group and the other classical groups in a field of characteristic zero or a prime characteristic. *Reine Angewandte Mathematik* **32** (1846)
16. Thompson, E.: A method for the construction of orthogonal matrices. *The Photogrammetric Record* **3**(13) (1958) 55–59
17. Buchberger, B.: Multidimensional Systems Theory - Progress, Directions and Open Problems in Multidimensional Systems. Reidel Publishing Company, Dordrecht - Boston - Lancaster (1985)
18. Gebauer, R., Möller, H.M.: On an installation of Buchberger’s algorithm. *Journal of Symbolic Computation* **6**(2-3) (1988) 275–286
19. Giovini, A., Mora, T., Niesi, G., Robbiano, L., Traverso, C.: “One sugar cube, please” or selection strategies in the Buchberger algorithm. In: Proceedings of the International Symposium on Symbolic and Algebraic Computation, Bonn, West Germany (1991) 49–54
20. Faugère, J.: A new efficient algorithm for computing Gröbner bases (F4). *Journal of Pure and Applied Algebra* **139**(1–3) (1999) 61–88
21. Kukulova, Z., Bujnak, M., Pajdla, T.: Automatic generator of minimal problem solvers. In: Proceedings of the European Conference on Computer Vision (ECCV), Marseille, France (2008) 302–315
22. Arun, K., Huang, T., Blostein, S.: Least-Squares Fitting of Two 3-D Point Sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* **9**(5) (1987) 698–700

Acknowledgments

The research leading to these results has received funding from the European Community’s Seventh Framework Programme (FP7/2007-2013) for the V-Charge project, and from the Swiss National Science Foundation under grant agreement n. 200021 125017/1.

B Using Multi-Camera Systems in Robotics: Efficient Solutions to the NPnP Problem

Authors: Kneip, Furgale, Siegwart

Link to [Table of Contents](#).

Using Multi-Camera Systems in Robotics: Efficient Solutions to the NPnP Problem

Laurent Kneip, Paul Furgale, and Roland Siegwart
Autonomous Systems Lab, ETH Zurich, Switzerland

Abstract—This paper introduces two novel solutions to the generalized-camera exterior orientation problem, which has a vast number of potential applications in robotics: (i) a minimal solution requiring only three point correspondences, and (ii) *gPnP*, an efficient, non-iterative n -point solution with linear complexity in the number of points. Already existing minimal solutions require exhaustive algebraic derivations. In contrast, our novel minimal solution is solved in a straightforward manner using the Gröbner basis method. Existing n -point solutions are mostly based on iterative optimization schemes. Our n -point solution is non-iterative and outperforms existing algorithms in terms of computational efficiency. Our results present an evaluation against state-of-the-art single-camera algorithms, and a comparison of different multi-camera setups. It demonstrates the superior noise resilience achieved when using multi-camera configurations, and the efficiency of our algorithms. As a further contribution, we illustrate a possible robotic use-case of our non-perspective orientation computation algorithms by presenting visual odometry results on real data with a non-overlapping multi-camera configuration, including a comparison to a loosely coupled alternative.

I. INTRODUCTION

The problem of computing the *exterior orientation* of a camera—its position and orientation given a set of correspondences between image observations and known 3D points—is one of the most fundamental in computer vision. It has a large number of potential applications such as camera calibration, augmented reality, object tracking, pose recovery, pose tracking, and visual SLAM, and thus turns out to be a key-component for vision-related tasks in robotics. However, when a camera has a restricted field of view, the solution may be poorly conditioned due to a bad distribution of observed points, or fail entirely due to lack of texture in the current viewing direction. Omnidirectional cameras perform better in this respect but the images they return are highly distorted. These are the main reasons why robotics system engineers are increasingly considering rigidly coupled multi-camera systems for mobile localization tasks, such as the one presented in Figure 1. The availability of integrated algorithms that can jointly process the information from all cameras is however still limited. Such algorithms offer the advantage of handling multiple cameras as one [1], and thus allow for the reuse of standard single-camera computer vision pipelines. The present paper aims at solving part of this problematic, and presents a straight-forward extension of fundamental camera pose computation algorithms from computer vision to the non-perspective or multi-camera case, targeting a wide range of potential robotic applications.

The Perspective- n -Point (PnP) problem for a single cam-

era originates from camera calibration [2], [3], [4], [5] and consists of recovering the camera position and orientation from n known 2D-3D correspondences. The first solutions have been presented outside the field of computer vision more than 150 years ago. Grunert (1846) [6] and Finsterwalder (1903) [7] were the first to derive minimal solutions aiming at solving the problem using three correspondences only. This problem is known in computer vision as the Perspective-*three*-Point (P3P) problem and leads to up to four solutions. Haralick et al. [8] reviewed the major direct solutions up to 1991 including the robust algorithm presented by Fischler and Bolles (1981) [3], who pointed out the importance of minimal solutions for robust hypothesize-and-test schemes when the measurements are affected by outliers (RANSAC). Different solutions to the P3P problem have been later proposed by Quan and Lan (1999) [5] and Gao et al. (2003) [9]. The most efficient solution has been presented in our previous work [10], and involves a parametrization avoiding the intermediate derivation of the point depths in the camera frame, and thus computes the camera pose in a single step.

The P3P problem is the minimal case of the PnP problem. The PnP problem was first solved by photogrammetrists (1963) [11], who also introduced the Direct Linear Transformation algorithm (DLT). The solutions of Fischler and Bolles [3], Horaud et al. [4], Quan and Lan [5], Fiore (2001) [12], and Ansar and Daniilidis (2003) [13] are other notable algorithms able to handle an arbitrary number of points. Lepetit et al. (2009) [14] present *ePnP*, the most efficient solution published to date as it is non-iterative and of linear complexity in the number of points.

The above-mentioned works are all designed for localization of a single perspective camera. In contrast, Chen and Chang (2004) [15] and Nistér and Stévenius (2006) [16] developed minimal 3-point solutions for localization with a generalized camera leading to 8 possible solutions. This is

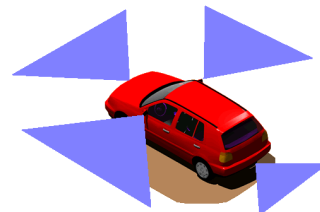


Fig. 1. Four cameras mounted on a car, an example of an n -camera system.

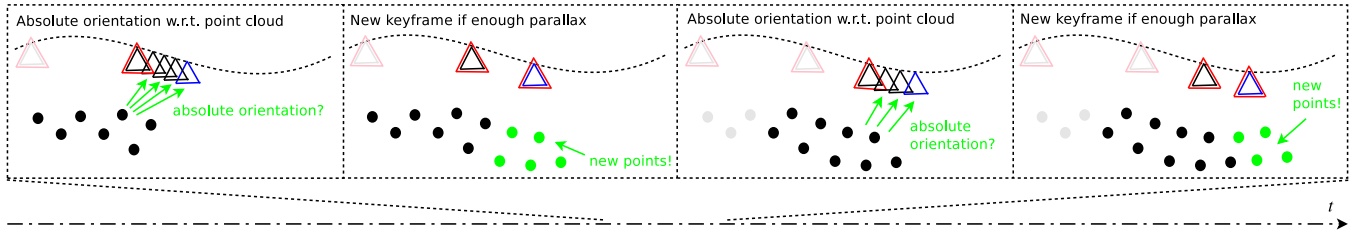


Fig. 2. Role of exterior orientation algorithms in geometric camera egomotion computation. Note that in the multi-camera case, each triangle represents a system of multiple cameras.

known as the Non-Perspective-*three*-Point problem (NP3P), and the solutions are applicable to multi-camera systems. However, in both cases the derivation of the solution is not very intuitive and involves arduous algebraic reasoning and the numerical solution of an 8-th order polynomial. Regarding the Non-Perspective-*n*-Point problem (NPnP), a number of iterative solutions have been presented by Chen and Chang (2004) [15], Schweighofer and Pinz (2008) [17], and Tariq and Dellaert (2004) [18]. These iterative methods, however, are computationally expensive and depend on a critical initialization of the pose—either via perspective or non-perspective geometric solutions in a RANSAC scheme, or temporal prediction in a pose tracking context. The major focus of these iterative methods thus lies on global optimization techniques rather than an improvement of the problem parametrization. A step forward in this direction was achieved by Ess et al. (2007) [19], who presented for the first time a non-iterative linear solution to the NPnP problem. However, the complexity is at least quadratic in the number of points.

The present paper starts off with a novel, intuitive parametrization of the multi-camera absolute pose computation problem, and presents a solution to the minimal NP3P case based on a straightforward application of the Gröbner basis approach. The major contribution then focuses on a novel solution to the NPnP problem with linear computational complexity in the number of points. To the best of our knowledge, this is the first non-iterative solution to the NPnP problem that achieves this level of efficiency. The paper is structured as follows: Section II outlines the synopsis of the problem. Section III presents our generalized minimal 3-point and linear *n*-point algorithms. In Section IV, we focus on a thorough comparison to equivalent state-of-the-art single camera models and an evaluation of different multi-camera setups. As an example, Section V shows for the first time visual odometry results on real data captured with a non-overlapping multi-camera rig, where all cameras are treated as one. Section VI finally concludes the paper.

II. SYNOPSIS OF THE MULTI-CAMERA EXTERIOR ORIENTATION PROBLEM

As an example, we consider in this paper the application of multi-camera exterior orientation computation to visual odometry. As illustrated in Figure 2, exterior orientation algorithms represent a fundamental building block of geometric keyframe based egomotion computation pipelines,

where the camera position is always derived with respect to a local point cloud. Note that a keyframe in the multi-camera sense denotes an entire set of keyframes (one per camera). The relative orientation of cameras required for triangulating new points can be derived from the exterior orientation of consecutive keyframes. This means that—apart from the bootstrapping phase—all geometric computations are achieved through the sole employment of exterior orientation algorithms.

The problem we are looking at is illustrated in Figure 3. It can be abstracted into a generalized form of the P3P algorithm [10], where the origins of the unit feature observation vectors, $\mathbf{f}_i$, are displaced from the rigid body origin by known vectors, $\mathbf{v}_{i0}$. The origins of $\mathbf{f}_i$ are equivalent to camera centers, and each $\mathbf{v}_{i0}$ thus represents the position of a camera inside the body frame. The variables we are interested in are the position, $\mathbf{t}$, of the rigid body in the world frame and the rotation, $\mathbf{R}$, from the body frame to the world frame. The observed points are expressed with $\mathbf{p}_{i0}$. Following the assumption of known extrinsic camera-to-body orientations, the unit feature observation vectors, $\mathbf{f}_i$, and the displacement vectors, $\mathbf{v}_{i0}$, from the body origin are expressed in the body frame. The depth of the features is denoted with n_i . Points expressed in the world and body frame are given the superscripts *w* and *b*, respectively. Note that the use of unit feature bearing vectors $\mathbf{f}_i$ is allowed under the assumption of calibrated cameras. Furthermore, the use of bearing vectors instead of normalized coordinates provides the generality of being applicable to any optical projection system.

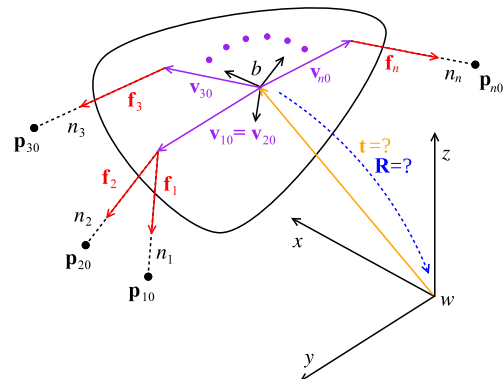


Fig. 3. Synopsis of the NPnP problem.

III. THEORY

The application of absolute orientation algorithms requires two variants. First, a minimal variant that uses only three points and thus can be employed in a hypothesize-and-test scheme. Second, an n -point solution that computes an optimal pose based on the identified inlier subset. This section highlights both approaches.

A. 3-point minimal solution

From Figure 3, we can easily derive the following system of equations

$$\begin{cases} \mathbf{R}(n_1 \mathbf{f}_1 + \mathbf{v}_{10}) + \mathbf{t} = \mathbf{p}_{10}^w \\ \mathbf{R}(n_2 \mathbf{f}_2 + \mathbf{v}_{20}) + \mathbf{t} = \mathbf{p}_{20}^w \\ \mathbf{R}(n_3 \mathbf{f}_3 + \mathbf{v}_{30}) + \mathbf{t} = \mathbf{p}_{30}^w \end{cases} \Rightarrow \begin{cases} n_1 \mathbf{f}_1 - n_2 \mathbf{f}_2 + \mathbf{v}_{12} = \mathbf{R}^T \mathbf{p}_{12}^w \\ n_2 \mathbf{f}_2 - n_3 \mathbf{f}_3 + \mathbf{v}_{23} = \mathbf{R}^T \mathbf{p}_{23}^w \\ n_3 \mathbf{f}_3 - n_1 \mathbf{f}_1 + \mathbf{v}_{31} = \mathbf{R}^T \mathbf{p}_{31}^w \end{cases}, \quad (1)$$

with $\mathbf{v}_{ij} = \mathbf{v}_{i0} - \mathbf{v}_{j0}$ and $\mathbf{p}_{ij} = \mathbf{p}_{i0} - \mathbf{p}_{j0}$. Note that the position of the body center, $\mathbf{t}$, is easily removed by subtracting pairwise equations.

Despite the compact look, this equation system is arduous to solve by hand. It is a multivariate polynomial equation system commonly solved via the Gröbner basis method. A good introduction to the approach can be found in [20]. The method consists of defining a monomial ordering over the polynomial terms and then iteratively generating and reducing new polynomials inside the ideal (the so-called s-polynomials) until a set of polynomials with good criteria for solvability is obtained. The most well-known application of this technique in geometric vision is the 5-point essential matrix solution of Stewénius et al. [21]. Rather than having to apply the Gröbner method for each new set of coefficients, the sequence of reductions performed by the method is often constant for a specific problem. Hence, we may solve the system using randomly chosen coefficients in a prime field and *trace* the solution offline. This trace may then be applied online using coefficients emanating from real data. The final algorithm is fast since we do no longer need to check for all polynomial reductions, but directly generate and reduce only the necessary s-polynomials following a fixed paradigm.

The computation of a Gröbner basis can be extremely long and is typically done by machine-generated code. The complexity depends to a large extent on the initial parametrization of the problem. It is influenced by: (1) the order of the equations; (2) the number of equations; (3) the number of unknowns; and (4) the chosen monomial ordering. In the present case, the system is best solved using the Cayley rotation matrix [22] parametrization $\mathbf{R}(w_1, w_2, w_3)$ and the *grevlex* monomial ordering. This leads to a system of 9 cubic equations in 6 unknowns. The produced Gröbner basis uses 8 base monomials: $w_1^2, n_3, n_2, n_1, w_3, w_2, w_1$, and 1. The generated code (~ 8000 lines) performs only 39 s-polynomial reductions, operating on a 48×85 matrix. The SVD of the corresponding action matrix finally leads to 8 solutions for the rotation matrix and the depths of the features. The position is afterwards derived as $\mathbf{t} = \frac{1}{3} \sum_{i=1}^3 \mathbf{p}_{i0}^w - \mathbf{R}(n_i \mathbf{f}_i + \mathbf{v}_{i0})$. Wrong solutions are easily found by putting a threshold on the imaginary parts of the singular values. In practice, a unique solution is obtained by

considering a fourth point, computing the reprojection error of this point for all remaining valid solutions, and finally selecting the one with the smallest error.

B. gPnP: non-iterative n -point solution with linear complexity in the number of points

As presented already by Ess et al. [19], starting from 6 points and using the standard rotation matrix parametrization $\mathbf{R} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}$, equation system (1) turns into a linear problem and becomes fully constrained with at least 15 linearly independent equations and 15 unknowns. Defining $\mathbf{n} = (n_1 \ n_2 \ n_3 \ n_4 \ n_5 \ n_6)^T$, $\mathbf{i}_1 = (-1 \ 0 \ 0)^T$, $\mathbf{i}_2 = (0 \ -1 \ 0)^T$, and $\mathbf{i}_3 = (0 \ 0 \ -1)^T$, we obtain

$$\begin{pmatrix} \mathbf{f}_1 & -\mathbf{f}_2 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{i}_1 \mathbf{p}_{12}^{wT} & \mathbf{i}_2 \mathbf{p}_{12}^{wT} & \mathbf{i}_3 \mathbf{p}_{12}^{wT} \\ \mathbf{f}_1 & \mathbf{0} & -\mathbf{f}_3 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{i}_1 \mathbf{p}_{13}^{wT} & \mathbf{i}_2 \mathbf{p}_{13}^{wT} & \mathbf{i}_3 \mathbf{p}_{13}^{wT} \\ \mathbf{f}_1 & \mathbf{0} & \mathbf{0} & -\mathbf{f}_4 & \mathbf{0} & \mathbf{0} & \mathbf{i}_1 \mathbf{p}_{14}^{wT} & \mathbf{i}_2 \mathbf{p}_{14}^{wT} & \mathbf{i}_3 \mathbf{p}_{14}^{wT} \\ \mathbf{f}_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{f}_5 & \mathbf{0} & \mathbf{i}_1 \mathbf{p}_{15}^{wT} & \mathbf{i}_2 \mathbf{p}_{15}^{wT} & \mathbf{i}_3 \mathbf{p}_{15}^{wT} \\ \mathbf{f}_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{f}_6 & \mathbf{i}_1 \mathbf{p}_{16}^{wT} & \mathbf{i}_2 \mathbf{p}_{16}^{wT} & \mathbf{i}_3 \mathbf{p}_{16}^{wT} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{n} \\ \mathbf{r}_1^T \\ \mathbf{r}_2^T \\ \mathbf{r}_3^T \end{pmatrix} = \begin{pmatrix} \mathbf{v}_{12} \\ \mathbf{v}_{13} \\ \mathbf{v}_{14} \\ \mathbf{v}_{15} \\ \mathbf{v}_{16} \end{pmatrix}. \quad (2)$$

The complexity of this solution is not easily extendable to an arbitrary number of points since the dimensionality of the solution space is increasing by one each time we add another feature. This leads to an increasingly complicated computation of the pseudo-inverse of the matrix, which is at least of complexity $O(mn^2)$ for $m \times n$ matrices and $m > n$.

Instead, we extend ePnP—the $O(n)$ PnP solution presented in Lepetit et al. [14]—to the NPNP problem. We call our algorithm *gPnP*. The basic idea consists of expressing all n points as a weighted sum of four control points. We define the first control point to be the centroid of the point cloud $\mathbf{c}_0^w = \frac{1}{n} \sum_{i=1}^n \mathbf{p}_{i0}^w$. We then compute the principal components of the point cloud by singular value decomposition $\mathbf{USV}^*$ of the point data matrix. The other control points are then defined as $\mathbf{c}_j^w = \mathbf{c}_0^w + s_j \mathbf{u}_j$, $j \in \{1, 2, 3\}$, where s_j is the j -th singular value and $\mathbf{u}_j$ the j -th column of $\mathbf{U}$. Each point is then defined by a weighted sum of control points $\mathbf{p}_i^w = \sum_{j=0}^3 \alpha_{ij} \mathbf{c}_j^w$. The weighting factors are invariant with respect to the coordinate frame, so we also have $\mathbf{p}_i^b = \sum_{j=0}^3 \alpha_{ij} \mathbf{c}_j^b$. Moreover, we also have $\mathbf{p}_i^b = n_i \mathbf{f}_i + \mathbf{v}_{i0}$. Substitution leads to

$$\sum_{j=0}^3 \alpha_{ij} \begin{pmatrix} c_{xj}^b \\ c_{yj}^b \\ c_{zj}^b \end{pmatrix} = n_i \begin{pmatrix} f_{xi} \\ f_{yi} \\ f_{zi} \end{pmatrix} + \begin{pmatrix} v_{xi0} \\ v_{yi0} \\ v_{zi0} \end{pmatrix}. \quad (3)$$

We now can pick any row (at best in the dimension where the bearing vector coordinates are highest) in order to find an expression for n_i . Let's say this is the third dimension. We obtain $n_i = (\sum_{j=0}^3 \frac{\alpha_{ij}}{f_{zi}} c_{zj}^b) - \frac{v_{zi0}}{f_{zi}}$. Backsubstitution into the two first equations leads to

$$\begin{cases} \sum_{j=0}^3 (\alpha_{ij} f_{zi} c_{xj}^b - \alpha_{ij} f_{xi} c_{zj}^b) = f_{zi} v_{xi0} - f_{xi} v_{zi0} \\ \sum_{j=0}^3 (\alpha_{ij} f_{zi} c_{yj}^b - \alpha_{ij} f_{yi} c_{zj}^b) = f_{zi} v_{yi0} - f_{yi} v_{zi0} \end{cases} \quad (4)$$

For n points, this turns into a linear system as follows

$$\begin{pmatrix} \alpha_{10} (f_{z1} \mathbf{I}_2 & \mathbf{J} \mathbf{f}_1) & \alpha_{11} (f_{z1} \mathbf{I}_2 & \mathbf{J} \mathbf{f}_1) & \alpha_{12} (f_{z1} \mathbf{I}_2 & \mathbf{J} \mathbf{f}_1) & \alpha_{13} (f_{z1} \mathbf{I}_2 & \mathbf{J} \mathbf{f}_1) \\ \alpha_{20} (f_{z2} \mathbf{I}_2 & \mathbf{J} \mathbf{f}_2) & \alpha_{21} (f_{z2} \mathbf{I}_2 & \mathbf{J} \mathbf{f}_2) & \alpha_{22} (f_{z2} \mathbf{I}_2 & \mathbf{J} \mathbf{f}_2) & \alpha_{23} (f_{z2} \mathbf{I}_2 & \mathbf{J} \mathbf{f}_2) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \alpha_{n0} (f_{zn} \mathbf{I}_2 & \mathbf{J} \mathbf{f}_n) & \alpha_{n1} (f_{zn} \mathbf{I}_2 & \mathbf{J} \mathbf{f}_n) & \alpha_{n2} (f_{zn} \mathbf{I}_2 & \mathbf{J} \mathbf{f}_n) & \alpha_{n3} (f_{zn} \mathbf{I}_2 & \mathbf{J} \mathbf{f}_n) \end{pmatrix} \begin{pmatrix} \mathbf{c}_0^b \\ \mathbf{c}_1^b \\ \mathbf{c}_2^b \\ \mathbf{c}_3^b \end{pmatrix} = \begin{pmatrix} \mathbf{E}(\mathbf{f}_1 \times \mathbf{v}_{10}) \\ \mathbf{E}(\mathbf{f}_2 \times \mathbf{v}_{20}) \\ \vdots \\ \mathbf{E}(\mathbf{f}_n \times \mathbf{v}_{n0}) \end{pmatrix}, \quad (5)$$

where $\mathbf{I}_2$ is the 2-by-2 identity matrix, $\mathbf{J} = \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \end{pmatrix}$, and $\mathbf{E} = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \end{pmatrix}$. This problem is of the form $\mathbf{A} \mathbf{s} = \mathbf{b}$, and the least-squares solution is given by $\mathbf{s} = \mathbf{A}^+ \mathbf{b}$, where $\mathbf{A}^+$ denotes the Moore-Penrose pseudo-inverse of matrix $\mathbf{A}$ and $\mathbf{s} = (\mathbf{c}_0^{bT} \ \mathbf{c}_1^{bT} \ \mathbf{c}_2^{bT} \ \mathbf{c}_3^{bT})^T$. Having always a solution space dimensionality of 12, this solution is of linear complexity in the number of points. $\mathbf{A}$ has full rank in the noise-free case. It has been verified that this is guaranteed as long as all points do not originate from the same camera. However, similar to the ePnP algorithm, the situation becomes more complicated under noise. In this case, $\mathbf{A}$ might be rank deficient and the general least-squares solution becomes

$$\hat{\mathbf{s}}_{LS} = \mathbf{A}^+ \mathbf{b} + [\mathbf{I}_n - \mathbf{A}^+ \mathbf{A}] \mathbf{y}, \quad (6)$$

where $\mathbf{y}$ is an arbitrary vector in $\mathbb{R}_n$. In other words, the solution is the sum of $\mathbf{A}^+ \mathbf{b}$ and a varying number of right-most nullspace vectors (the ones corresponding to the smallest singular values) multiplied by some unknown factors. In our experimental section, we consider noise levels up to 10 pixels in standard deviation, and we verify that a good maximum number of right-most nullspace vectors to consider is 5. This leads to a total number of 6 possible cases. $\mathcal{N}_i(\mathbf{A})$ describes the i -th right-most nullspace vector of $\mathbf{A}$.

- case 0: The solution is simply given by $\mathbf{s}_0 = \mathbf{A}^+ \mathbf{b}$. This is the case if no noise is added to the measurements.
- case 1: The solution is given by $\mathbf{s}_1 = \mathbf{A}^+ \mathbf{b} + \lambda_1 \mathcal{N}_1(\mathbf{A})$. The unknown factor λ_1 is found by imposing the constraint that the distances between the control points expressed in the world and body frames need to be preserved. Using $c_{ij} = (\mathbf{c}_i^w - \mathbf{c}_j^w)^T \cdot (\mathbf{c}_i^w - \mathbf{c}_j^w)$ and 3 distance constraints, we obtain

$$\begin{cases} \|\mathbf{I}_3 & -\mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3\| \mathbf{s}_1(\lambda_1) \|^2 = c_{01} \\ \|\mathbf{I}_3 & \mathbf{0}_3 & -\mathbf{I}_3 & \mathbf{0}_3\| \mathbf{s}_1(\lambda_1) \|^2 = c_{02} \\ \|\mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & -\mathbf{I}_3\| \mathbf{s}_1(\lambda_1) \|^2 = c_{03} \end{cases}, \quad (7)$$

which results in 3 quadratic equations in the monomials λ_1^2 , λ_1 , and 1. This is enough equations to find a unique solution. However, this also highlights the main difficulty of the gPnP compared to the ePnP algorithm. We no longer end up with only even powers of the unknown λ_1 . This means that the gPnP solution is substantially more complicated than ePnP, even for a single added nullspace vector. When considering more nullspace vectors, the number of monomials increases drastically, which means that straightforward solution techniques using linearization can not be used. Instead, we propose to consistently use the Gröbner basis method for finding the linear combination factors of nullspace vectors. In the present case, this leads to a 5×3 matrix and one s-polynomial reduction.

- case 2: The solution with 2 nullspace vectors is given by $\mathbf{s}_2 = \mathbf{A}^+ \mathbf{b} + \sum_{g=1}^2 \lambda_g \mathcal{N}_g(\mathbf{A})$. The three distance

constraints from (7) with $\mathbf{s}_1$ substituted by $\mathbf{s}_2(\lambda_1, \lambda_2)$ are still sufficient for finding the unknown linear combination factors λ_1 and λ_2 . The Gröbner matrix in this case is 10×6 and solved via 8 s-polynomial reduction steps.

- case 3: In the case of 3 nullspace vectors the solution is $\mathbf{s}_3 = \mathbf{A}^+ \mathbf{b} + \sum_{g=1}^3 \lambda_g \mathcal{N}_g(\mathbf{A})$. Now we need the additional distance constraint, c_{12} , with $\mathbf{s}_1$ substituted by $\mathbf{s}_3(\lambda_1, \lambda_2, \lambda_3)$, namely

$$\|(\mathbf{0}_3 \ \mathbf{I}_3 \ -\mathbf{I}_3 \ \mathbf{0}_3) \mathbf{s}_3(\lambda_1, \lambda_2, \lambda_3)\|^2 = c_{12}. \quad (8)$$

The resulting system of 4 equations is solved using a Gröbner matrix of 15×18 and 59 s-polynomial reduction steps.

- case 4: $\mathbf{s}_4 = \mathbf{A}^+ \mathbf{b} + \sum_{g=1}^4 \lambda_g \mathcal{N}_g(\mathbf{A})$. The solution requires the consideration of the additional distance constraint, c_{13} ,

$$\|(\mathbf{0}_3 \ \mathbf{I}_3 \ \mathbf{0}_3 \ -\mathbf{I}_3) \mathbf{s}_4(\lambda_1, \dots, \lambda_4)\|^2 = c_{13}. \quad (9)$$

The system of five equations leads to a Gröbner matrix of 25×37 and is solved via 240 s-polynomial reduction steps.

- case 5: $\mathbf{s}_5 = \mathbf{A}^+ \mathbf{b} + \sum_{g=1}^5 \lambda_g \mathcal{N}_g(\mathbf{A})$. The solution requires the consideration of the last available distance constraint c_{23}

$$\|(\mathbf{0}_3 \ \mathbf{0}_3 \ \mathbf{I}_3 \ -\mathbf{I}_3) \mathbf{s}_5(\lambda_1, \dots, \lambda_5)\|^2 = c_{23}. \quad (10)$$

The system of six equations leads to a Gröbner matrix of 44×80 and is solved via 936 s-polynomial reduction steps.

The Gröbner basis computations are considerably exhaustive, especially for an increased number of nullspace vectors. This means that they do not deliver optimal linear combination factors under noise. In order to tackle this problem, cases 1 to 5 are followed by a polishing scheme that consists of a refinement of the respective factors over the six distance-conservation constraints of the original control points. This is not to be confused with a batch optimization of the reprojection error of all points into all cameras; The complexity of this operation is independent of the number of involved points, n , and thus approximately constant. $\mathbf{R}$ is finally derived by control point alignment: $(\mathbf{U}, \mathbf{D}, \mathbf{V}) = \text{SVD}(\sum_{j=0}^3 (\mathbf{c}_j^b - \bar{\mathbf{c}}^b)(\mathbf{c}_j^w - \bar{\mathbf{c}}^w)^T) \Rightarrow \mathbf{R} = \mathbf{V} \mathbf{U}^T$. The position of the body center is then given by $\mathbf{t} = \frac{1}{4} \{(\sum_{j=0}^3 \mathbf{c}_j^w) - \mathbf{R}(\sum_{j=0}^3 \mathbf{c}_j^b)\}$. Knowing the pose of the body, we can now properly select the best solution based on the smallest reprojection error of the control points. The reprojection errors are evaluated as a function of the dot-products between unit bearing vectors from the body frame. Finally, the depth of each point can be retrieved by reusing the weight factors in order to recompute the points in the body frame, and then transforming them into the corresponding camera frame: $n_i = \|(\sum_{j=0}^3 \alpha_{ij} \mathbf{c}_j^b) - \mathbf{v}_{i0}\|$.

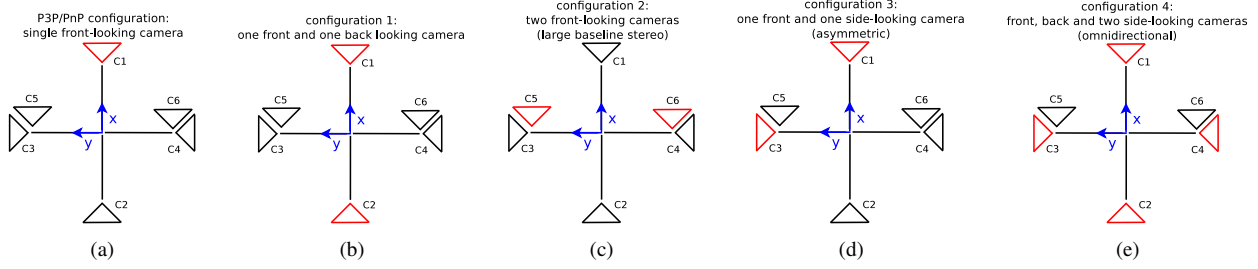


Fig. 4. The five configurations that are explored in the simulation experiments.

IV. SIMULATION RESULTS

This section presents experimental results on the algorithms presented in Section III. After introducing the experiment outline, we present analyses of noise resilience, numerical accuracy, and computational efficiency. A comparison of both the minimal and the n -point solutions to their most efficient, state-of-the-art single camera equivalents is included.

A. Experiment outline

Besides being applicable to non-central cameras that do not have a single effective viewpoint, the non-perspective pose algorithms offer the advantage of being applicable to a rigidly coupled system of multiple central cameras. In this way, the camera system can be treated as one single camera, allowing the reuse of a single-camera structure-from-motion pipeline. With this concept in mind, we evaluate different multi-camera setups and compare the results to perspective localization with a single camera using state-of-the-art methods. As a reference minimal solution for a single camera, we use the novel P3P algorithm presented in [10], which shows superior behavior in accuracy and efficiency compared to alternative solutions. The reference for n -point perspective localization is given by the efficient non-iterative linear-complexity ePnP algorithm [14] from which we also derived the basic idea for our gPnP approach. The disambiguation of the multiple solutions returned by the minimal algorithms is done each time by considering a fourth point and picking the solution that leads to its smallest reprojection error.

The different setups are illustrated in Figure 4. All virtual cameras have a distance of 1m to the fictive body center and camera calibration parameters taken from a real camera. The focal length equals to 400, the resolution to 640×480 , the principal point to $(320, 240)$, and the field of view to $77^\circ \times 62^\circ$. For each experimental iteration, we generate 50 random visible points in each camera with a uniformly varying depth between 10m and 20m. The single camera algorithms are evaluated by considering only the points from camera C1. The generalized multi-camera algorithms are evaluated using four different configurations: 1) C1 and C2: two cameras facing opposite directions, 2) C5 and C6: two cameras facing the same direction (classical stereo configuration with large baseline), 3) C1 and C3: asymmetric configuration with two cameras facing orthogonal directions, and 4) C1, C2, C3, and

C4: four cameras facing in all four directions. The ground-truth pose for each experimental run is simply kept at $\mathbf{t} = \mathbf{0}$ and $\mathbf{R} = \mathbf{I}_3$. All experiments have been performed on a standard 2.8GHz Intel Core 2 Duo CPU.

B. Noise resilience

In order to evaluate the resilience to noise, we add Gaussian noise with zero mean and standard deviations reaching from 0 to 10 pixels to our measurements in the image plane. We execute 10,000 iterations for each combination of noise level, algorithm, and camera-configuration. The resulting plots show the mean and median norm of translation and rotation error vectors. For a rotation error, we use the rotation angle, in radians, between the solution and the ground truth. Because the true orientation is at identity, we can compute this as the angle component of the axis/angle decomposition of $\mathbf{R}$.

Figure 5 shows the results for the minimal algorithms. It can be seen that configuration 2 (two front-looking cameras) behaves worst and very similar to the single front-looking camera. Configuration 1 (symmetric) provides lower median error than configuration 3 (asymmetric). However, configuration 1 shows a slightly elevated mean translation error, which

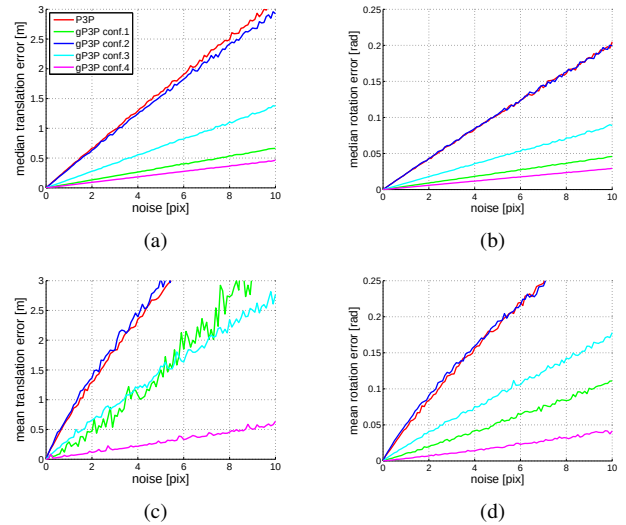


Fig. 5. Mean and median error of translation (a, c) and rotation (b, d) for the non-linear minimal algorithms. As indicated in Figure 4, the reference P3P algorithm (P3P, red) [10] uses only one camera, whereas our new NP3P algorithm (gP3P) is tested with four different multi-camera configurations.

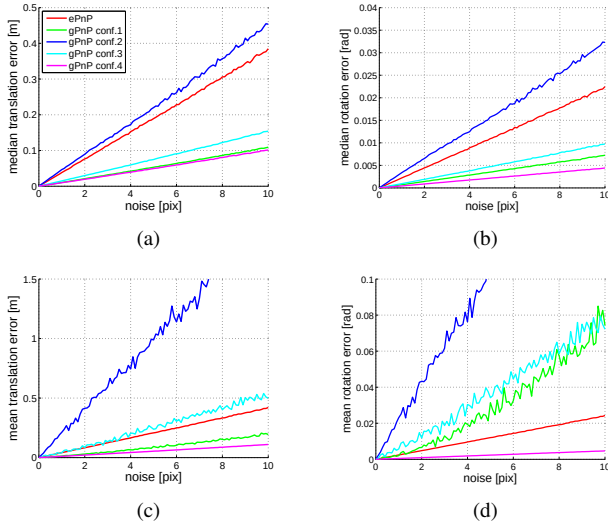


Fig. 6. Mean and median error of translation (a, c) and rotation (b, d) for the linear n -point algorithms. As indicated in Figure 4, the reference PnP algorithm (ePnP, red) [14] uses only one camera, whereas our new NnP algorithm (gPnP) is tested with four different multi-camera configurations.

leads to the conclusion that the translation computation is less robust in this case. Configuration 4 with four cameras pointing in all directions shows best behavior.

As indicated in Figure 6, the behavior for the n -point solutions is not much different. We can see that, again, configuration 2 behaves worst, this time even slightly worse than the ePnP algorithm. The median error of configurations 1, 3 and 4 are much smaller with configuration 4 again being the best configuration. Looking at the mean errors, we can see that configurations 1, 2 and 3 are deficient in terms of robustness, and mostly lead to even higher errors than the ePnP, especially in the rotational degrees of freedom. However, configuration 4 again behaves best for the mean error too.

C. Efficiency and numerical precision

Figure 7(a) shows the execution time of the minimal solutions and the n -point algorithms with 100 points. The generalized algorithms have higher complexity and thus also higher execution times. They, however, still remain real-time capable. Looking at Figure 7(b), we observe the major benefit of the ePnP and the gPnP algorithms, namely that their execution time stays linear as a function of the number of used points. The higher slope indicates that the per-point time consumption is higher for the gPnP algorithm. This is however not problem, since we have to bear in mind that the algorithm also solves a different, more complicated problem. Moreover, the low execution time of 12 ms for 5000 points proves that the gPnP algorithm is perfectly suited for real-time applications. Table I shows that the numerical accuracy (median error for zero noise) of the gP3P algorithm is worse than P3P. This is clearly related to the difference in computational complexity. The numerical accuracy of the gPnP algorithm, however, remains competitive with the ePnP algorithm. The reason is that the linear-combination-factor-

polishing compensates for errors originating from the more-complicated Gröbner basis solutions.

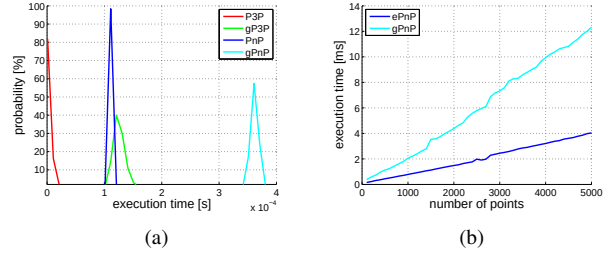


Fig. 7. Comparison of the execution time of the different algorithms (a) and execution time of the ePnP and gPnP algorithms in function of the number of points (b).

TABLE I
NUMERICAL ACCURACY OF THE DIFFERENT ALGORITHMS

	P3P	gP3P
trans. median error [m]	1.7392e-014	1.1829e-011
rot. median error	8.1712e-016	2.2107e-013
	ePnP	gPnP
trans. median error [m]	2.6246e-014	7.1394e-014
rot. median error	1.1096e-015	6.4384e-016

V. CASE STUDY: VISUAL ODOMETRY

In order to underline the potential of the presented algorithms in robotics, we integrated them into a structure-from-motion pipeline as outlined in Section II. We extended the framework presented in our earlier work [23], which operates on two non-overlapping cameras facing opposite directions, as indicated in Figure 8. The performance is evaluated on the same dataset as in [23], which has been collected by moving the camera rig in a room equipped with a Vicon motion capture system offering ground truth data. The algorithm works as follows:

- Bootstrapping is done using the exact pipeline presented in our previous work, which executes single camera structure-from-motion in each of the two cameras separately, and then fuses the individual relative displacements in order to determine the visual scale factors and recover results in metric scale.
- Once the scale is converged, we switch to our novel algorithms treating all cameras as one. As illustrated

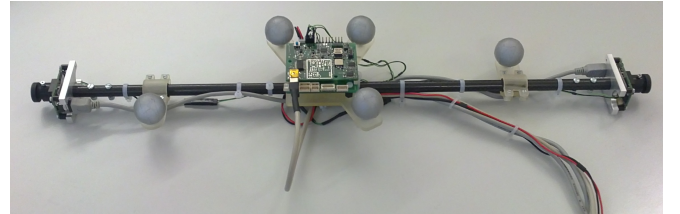


Fig. 8. Stereo-camera rig with two cameras facing opposite directions used in our experiments. The cameras are synchronized and capture WVGA images at 10Hz.

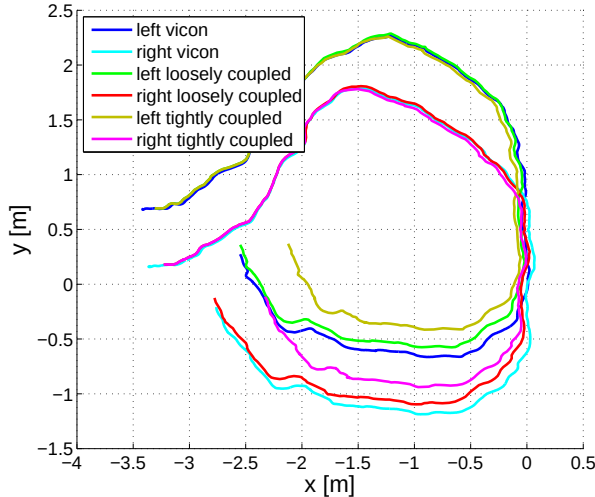


Fig. 9. Top view on visual odometry results obtained on a real indoor dataset. The plot shows a comparison between ground truth, results obtained using the IMU-supported framework in [23], and results obtained using a minimal single-camera like vision-only pipeline based on the proposed algorithms.

in Figure 9, we achieve stable egomotion computation with our novel approach.

- The point clouds from both sides are unified into a single one, and we use the non-perspective 3-point and n -point algorithms presented in this paper for robust exterior orientation computation.
- Each time the cameras experience enough median disparity between feature correspondences, a new point cloud is triangulated. This procedure is followed by bundle adjustment over two keyframes only, meaning 2 pairs of views including the rigid-coupling constraint in between.

Note that the higher drift is not a sign of bad performance, but rather the natural consequence of our minimal approach. The original loosely coupled framework presented in [23] employs the visual odometry algorithm presented in [24] extended by windowed bundle adjustment over 10 keyframes. Moreover, it employs IMU information in order to retrieve relative rotation between successive camera frames, and performs explicit scale propagation and scale estimation over multiple keyframes in order to achieve low drift and stable metric scale recovery. In contrast, the presented tightly coupled pipeline is very minimalistic and operates with point clouds optimized over two keyframes only. It also does not use any IMU information. Metric scale is implicitly recovered by including the rigid coupling constraint into our minimal non-linear optimization. However, as explained in [23], the observability of metric scale in the non-overlapping case is affected by motion singularities. This also explains the drift in the results: The metric scale is at times badly observable, which leads to a drifting translation magnitude during non-linear optimization. Similar to [23], the observability of the metric scale can easily be improved in a future

implementation that performs joint windowed bundle adjustment over both cameras and multiple keyframes, including the rigid coupling constraint.

In conclusion, the fact that we can robustly compute the egomotion of the rig in a minimal, “single-camera like” fashion disposing of a number of elements such as scale estimation, scale propagation, windowed bundle adjustment, and IMU information certainly proves the potential given by our novel algorithms, and leads to significantly lower implementational and computational complexity. To the best of our knowledge, this marks the first visual odometry results on real data treating cameras with non-overlapping fields of view as one. Note that—even though the present experiments employ information from two cameras only—the presented approach can be used for an arbitrary number of cameras due to the generality of the employed geometric algorithms.

Note to the reviewers: The presented geometric algorithms along with the perspective relative and absolute camera pose computation algorithms presented in our earlier work are planned to be made publicly available in an open-source ROS package toward the end of the year. The same is planned for a second ROS-package that will make all mentioned single- and multi-camera visual odometry pipelines available.

VI. CONCLUSION

In this paper, we presented two novel solutions to the non-perspective camera pose computation problem. The first solution is minimal and solved in a straightforward manner using the original parametrization and the Gröbner basis method. The second solution is a linear-complexity and non-iterative n -point algorithm for generalized cameras outperforming existing solutions in terms of efficiency. Our simulation results show that using these algorithms in conjunction with rigidly coupled multi-camera systems easily outperforms well-established single camera solutions. The best results are obtained with camera-systems pointing into all viewing directions. The algorithms are robust and real-time compliant. Finally—as one of numerous potential applications in robotics—we show a minimal single-camera-like visual odometry pipeline for multi-camera systems based on the presented algorithms. This leads for the first time to real results implementing the generalized concept of “treating multiple cameras as one” on a non-overlapping multi-camera rig.

ACKNOWLEDGMENT

The research leading to these results has received funding from the EU project V-Charge (FP7-269916), and from the Swiss National Science Foundation under grant agreement n. 200021 125017/1.

REFERENCES

- [1] R. Pless. Using many cameras as one. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 587–593, Madison, WI, USA, 2003.

- [2] M.A. Abidi and T. Chandra. A new efficient and direct solution for pose estimation using quadrangular targets: Algorithm and evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 17(5):534–538, 1995.
- [3] M.A. Fischler and R.C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.
- [4] R. Horaud, B. Conio, and O. Le Boulleux. An analytic solution for the perspective 4-point problem. *Computer Vision, Graphics, and Image Processing*, 47(1):33–44, 1989.
- [5] L. Quan and Z. Lan. Linear n-point camera pose determination. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 21(8):774–780, 1999.
- [6] J.A. Grunert. Das pothenotische Problem in erweiterter Gestalt nebst über seine Anwendungen in Geodäsie. In *Grunerts Archiv für Mathematik und Physik*, 1841.
- [7] S. Finsterwalder and W. Scheufele. *Das Rückwärtseinschneiden im Raum*. Verlag Herbert Wichmann, Berlin, Germany, 1937.
- [8] R.M. Haralick, C. Lee, K. Ottenberg, and M. Nolle. Analysis and solutions of the three point perspective pose estimation problem. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Maui, USA, 1991.
- [9] X.S. Gao, X.R. Hou, J. Tang, and H.F. Cheng. Complete solution classification for the perspective-three-point problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 25(8):930–943, 2003.
- [10] L. Kneip, D. Scaramuzza, and R. Siegwart. A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Colorado Springs, USA, 2011.
- [11] I. Sutherland. Sketchpad: A man machine graphical communications system, 1963. Technical Report 296, MIT Lincoln Laboratories.
- [12] P.D. Fiore. Efficient linear solution of exterior orientation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 23(2):140–148, 2001.
- [13] A. Ansar and K. Daniilidis. Linear pose estimation from points or lines. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 25(5):578–589, 2003.
- [14] V. Lepetit, F. Moreno-Noguer, and P. Fua. Epnp: An accurate $O(n)$ solution to the pnp problem. *International Journal of Computer Vision (IJCV)*, 81(2):578–589, 2009.
- [15] Chu-Song Chen and Wen-Yan Chang. On pose recovery for generalized visual sensors. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(7):848–861, 2004.
- [16] D. Nistér and H. Stewénius. A minimal solution to the generalized 3-point pose problem. *Journal of Mathematical Imaging and Vision (JMIV)*, 27(1):67–79, 2006.
- [17] G. Schweighofer and A. Pinz. Globally optimal $O(n)$ solution to the pnp problem for general camera models. In *Proceedings of the British Machine Vision Conference (BMVC)*, Leeds, UK, 2008.
- [18] Sarah Tariq and Frank Dellaert. A multi-camera 6-DOF pose tracker. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 296–297, Arlington, VA, USA, 2004.
- [19] A. Ess, A. Neubeck, and L. Van Gool. Generalised linear pose estimation. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 22.1–22.10, Warwick, UK, 2007.
- [20] David A. Cox, John Little, and Donal O’Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*, 3/e (Undergraduate Texts in Mathematics). Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.
- [21] H. Stewénius, C. Engels, and D. Nistér. Recent developments on direct relative orientation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 60(4):284–294, 2006.
- [22] Arthur Cayley. About the algebraic structure of the orthogonal group and the other classical groups in a field of characteristic zero or a prime characteristic. *Reine Angewandte Mathematik*, 32, 1846.
- [23] T. Kazik, L. Kneip, J. Nikolic, M. Pollefeys, and R. Siegwart. Real-Time 6D Stereo Visual Odometry with Non-Overlapping Fields of View. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Providence, USA, 2012.
- [24] L. Kneip, M. Chli, and R. Siegwart. Robust real-time visual odometry with a single camera and an IMU. In *Proceedings of the British Machine Vision Conference (BMVC)*, Dundee, Scotland, 2011.

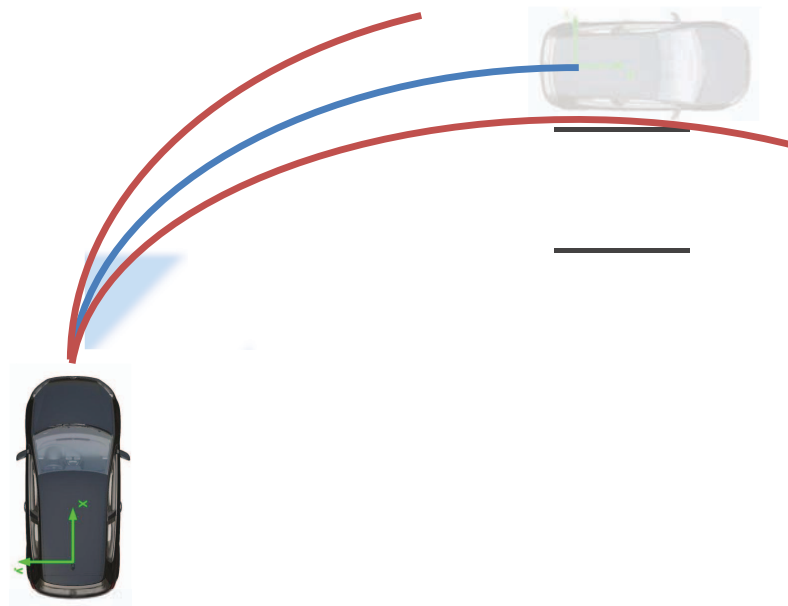
C Online Localization for an Autonomous Car using Multiple Cameras

Author: Neunert

Link to [Table of Contents](#).

Online Localization for an Autonomous Car Using Multiple Cameras

Master Thesis



Supervised by

Dr. Paul Furgale
Dr. Cédric Pradalier

Author

Michael Neunert

Abstract

The presented thesis is embedded in an EU funded research project on future personal transport called V-Charge. The goal of the V-Charge project is the development of an autonomous drop-off and retrieval system (autonomous valet parking system) for electric cars in public spaces. Therefore, these cars have to be equipped with on board localization capabilities that allow the systems to estimate their current position with respect to a map describing the essential structures of the environment. The development of this localization system is the aim of the presented work.

As the system is targeted for regular cars, the sensors used for localization should be consumer affordable and integrate into the structure of the car. Additionally, the system is supposed to work outdoors (parking lots) as well indoors (parking garages). Thus, this system is purely based on cameras and wheel odometry.

While the approach of the project is similar to common SLAM strategies, the hardware setup is unusual and presents special challenges. In contrast to previously developed systems, the car used in this project is equipped with multiple fisheye cameras which require several alterations to a common visual localization approach. Also, the localization system is designed for unaltered in- and outdoor parking spaces and thus cannot not rely on GPS or offboard sensor data.

In a first part, a robust front-end (feature extraction and matching, landmark triangulation) is developed which is able to provide long and robust feature tracks across the differently distorted parts of the fisheye image. A novelty matching strategy is developed that allows robust keypoint association in strongly distorted images and in scenes with highly self-similar structures.

An optimization back-end solves the problems of pose estimation, outlier rejection and landmark refinement efficiently within a single integral step.

Tested within the car's software environment on real world data recordings, the capabilities of the system are assessed. Furthermore, the core elements of the approach taken are compared to state-of-the-art implementations and performance improvements over these are validated.

The results demonstrate the high accuracy of the localization system and also prove that the achievable processing times allow realtime operation.

Acknowledgement

First of all, I would like to thank both of my supervisors Dr. Paul Furgale and Dr. Cédric Pradalier. It has been a unique experience to be supervised by two Postdocs and learn from their expertise in autonomous and vision based systems. But also on the software development side they provided a huge asset of tips and hints which helped me to dive deeper into programming.

Special thanks to Paul for the inspiring on- and off-topic discussions. It has been an honour to work together with such a highly talented researcher and his clear, well-structured approach to research is simply amazing. But also with his great knowledge of human nature he kept motivation on a very high level throughout the project.

Many thanks also to Cédric for all the great support during but also before the master thesis. Without him, I would not have been given the chance to work on such a great project. But apart from his huge high level effort, he still was there whenever needed and was able to come up with solutions instantly when problems occurred.

My highest appreciations go to Fabio Diem who has been a great fellow on our common quest for a master's degree. With his unswerving persistency he did not only create a great working atmosphere but also helped to identify major bugs in my part of program. At the same time, he was always patient with me, no matter how bad the delivered visual odometry data was that his thesis was highly depending on.

I would also like to acknowledge the work of the other members of the V-Charge project, specifically: the VW group for providing the cars, but also especially Peter Mühlfellner for the joint development of core elements, the ADTF interface and data handling classes as well as Wojciech Derendarz for the great support during the project and the organization of the integration week; CVG especially Lionel Heng and Dr. Friedrich Fraundorfer for calibrating the intrinsics, but also Professor Marc Pollefeys, Gim Hee Lee, Bo Li and Christian Häne for the fruitful meetings; and last but not least Bosch for providing the extrinsic calibration.

Zurich, October 2012

Michael Neunert

Contents

1	Introduction	1
1.1	Project V-Charge	1
1.1.1	About the Project	1
1.1.2	Hardware Setup	1
1.1.3	The Two Cars	3
1.1.4	Software Environment	3
1.2	Visual Localization for V-Charge	4
1.2.1	Goal of the Project	4
1.2.2	Starting Point	4
1.2.3	State-of-the-Art Localization for Multi Camera Systems	5
1.2.4	Localization Pipeline	6
1.3	Notation and Coordinate Systems	7
1.3.1	Notation	7
1.3.2	Coordinate Frames	8
1.4	Structure of the Report	9
2	Camera Model	11
2.1	Distorted Fisheye Camera Model	11
2.1.1	Brief Description	11
2.1.2	Projection Process	12
2.2	Perfect Spherical Camera	13
2.3	Epipolar Constraint	14
2.3.1	About the Epipolar Constraint	14
2.3.2	Generalized Epipolar Constraint	16
2.3.3	Epipolar Constraint for Spherical Cameras	17
2.4	Verification of the Camera Model	18
2.5	Camera Calibration	18
2.5.1	Calibration Process	18
2.5.2	Calibration Results	19
3	Front-End	23
3.1	Feature Extraction	23
3.1.1	State-of-the-art Feature Detectors and Descriptors	23
3.1.2	Selected Feature Extractors	25

3.2	Map Projection	25
3.3	Feature Matching	26
3.3.1	Matching Strategies	26
3.3.2	Uncertainty Cone Based Matching Approach	27
3.4	Triangulation	35
3.4.1	State of the Art	35
3.4.2	Implemented Approach	36
3.5	Observation Model and Descriptor Propagation	36
4	Back-End	39
4.1	State of the Art	39
4.1.1	Pose Estimation	39
4.1.2	Outlier detection and removal	40
4.1.3	Landmark Refinement	42
4.2	Back-End Approach	43
4.2.1	Problem Formulation	43
4.2.2	Problem Definition	45
4.2.3	Properties of the problem	46
4.3	Optimization Frameworks	47
4.3.1	Available Optimization Frameworks	47
4.3.2	Optimization Framework Choice	48
5	Implementation	49
5.1	Build System and Software Environment	49
5.1.1	Build System	49
5.1.2	ADTF Middleware	49
5.2	Software Structure	51
5.2.1	Software Elements	51
5.2.2	Structure of the Localization Pipeline	51
5.2.3	Modules	52
5.2.4	Filters	55
5.2.5	Executables	56
5.3	Configuration	56
5.3.1	Pre-build Configuration	57
5.3.2	Configuration File	57
5.4	Running, Debugging, Logging and Analyzing the Localization Pipeline	57
5.4.1	Running the Localization Pipeline	57
5.4.2	Debugging the Localization Pipeline	58
5.4.3	Logging and Analyzing the Localization Pipeline	58

6	Results	59
6.1	Front-End	59
6.1.1	Comparison of Feature Extractors	59
6.1.2	Evaluation of Matching Strategies	60
6.2	Back-end	62
6.2.1	Outlier Rejection and Landmark Refinement	62
6.2.2	Evaluation of Pose Prior	63
6.3	Visual Odometry	64
6.3.1	Filtering of Wheel Odometry	64
6.3.2	Pose Estimation Accuracy	66
6.3.3	Long Distance Tests	69
6.3.4	Timings	70
7	Summary	75
7.1	Conclusion	75
7.1.1	Setup and Integration	75
7.1.2	Visual Odometry Pipeline	75
7.2	Future Work	76
7.2.1	Integration	76
7.2.2	Visual Odometry Pipeline	76
8	Account	77
	List of Figures	78
	Bibliography	81

List of Acronyms

ADTF	Automotive Data and Time-Triggered Framework
ASL	Autonomous Systems Lab
BRISK	Binary Robust Invariant Scalable Keypoints
CVG	Computer Vision and Geometry Group
ETH	Eidgenössische Technische Hochschule Zürich
GPS	Global Positioning System
GPU	Graphics Processing Unit
GUI	Graphical User Interface
LRF	Local Reference Frame
MLE	Maximum Likelihood Estimation
OFOV	Overlapping field-of-view
RAM	Random-Access Memory
RANSAC	RANdom SAmple Consensus
SFM	Structure from Motion
SLAM	Synchronous Localization and Mapping
SQL	Structured Query Language
SURF	Speeded Up Robust Features
VO	Visual Odometry
VRF	Vehicle Reference Frame
VW	Volkswagen AG
WRF	World Reference Frame

1 Introduction

1.1 Project V-Charge

1.1.1 About the Project

The presented master thesis is carried out at the Autonomous Systems Lab at ETH Zurich and is an integral part of the project V-Charge. V-Charge itself is a EU funded research project, focused on an automated car drop-off and retrieval in public areas. The goal is to develop a system that enables a set of electrically driven cars to autonomously navigate in parking lots or garages. Additionally, these cars have to be coordinated to share charging stations to fill up their batteries. This system should only be based on widely available and consumer affordable technology. Hence, it is mostly based on vision and simple motion sensors.

The project is carried out by a consortium of European universities and companies. Current partners are ETH Zurich (specifically the Autonomous Systems Lab and the Computer Vision and Geometry Group), Volkswagen AG, University of Oxford, Robert Bosch GmbH, Università degli Studi di Parma and Technische Universität Braunschweig. Both ETH laboratories are mainly involved in work package 3 which covers the localization and mapping system of the cars.

1.1.2 Hardware Setup

Volkswagen AG together with Bosch provides the V-Charge project with two VW Golf V cars, which are equipped with a rich set of sensors described below.

Cameras Both cars feature four cameras with fisheye lenses. The front and back cameras are located in the car's VW logos on the front and the back respectively. The side cameras are mounted inside the side mirrors. The camera locations are shown in green in figure 1.1. While the front and back facing cameras are on a near horizontal level, the side cameras are facing a little downwards. Still, all neighbouring cameras share some overlapping field of view. The cameras used are Allied Vision Manta G-125 cameras with a resolution of 1292 by 964 pixels paired with Sunex DSL215 miniature fisheye lenses with a horizontal field of view of 185°. The cameras are set up to capture images at a frequency of 12.5 Hz.

There is also a front facing stereo camera in the rear-view mirror (indicated by the left blue coordinate frame in figure 1.1). However, due to intellectual property reasons, the images of this camera cannot be used within this project.

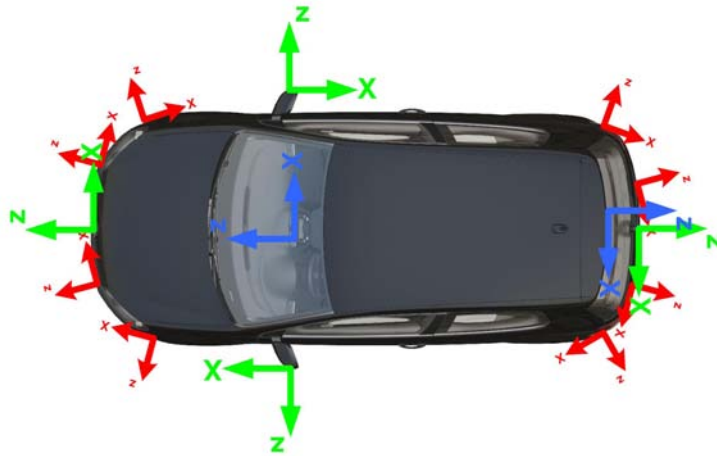


Figure 1.1: Sensor locations on the V-Charge vehicles [28]

Wheel odometry VW provides two wheel odometries.

The first is raw, unprocessed wheel odometry data containing two wheel tick counters (rear left and rear right) as well as a steering wheel angle. Additionally, a conversion formula from steering wheel angle to steering angle is provided. Due to the wheel tick counter and a high sensing rate of 50 Hz, this data is highly quantized.

The second wheel odometry is preprocessed using an unknown car model. Thus, the processed wheel odometry consists of a translational position on a horizontal plane as well as a yaw angle. Both measurements are with respect to the location of initialization. Even though this data is preprocessed, it is still highly quantized. Additionally, it is not synced with other sensors and in particular not with the camera frames. Therefore, an extra preprocessing step described in section 6.3.1 is necessary.

Gyroscope and Accelerometer In addition to the cameras, there is also an one axis gyroscope (measuring rotation around the vertical axis) and a two axis accelerometer (measuring accelerations in the horizontal plane), which deliver additional information on the pose changes of the car. While these sensors are not used in the visual odometry pipeline itself, they will be used to improve the localization results (see [7]).

iTrace One car also features an iTrace sensor which fuses GPS and IMU data. Thus, it is delivering highly precise, ground-truth state information. However, these kind of sensors are quite expensive. As described in subsection 1.1.1 the used sensors should be affordable and also work inside parking garages. Hence, the iTrace sensor is only used for verification and comparison of the pipeline but never as input data.

Velodyne The same car that has the iTrace sensor on it, also features a Velodyne, a highly precise 3D laser scanner. It is able to provide range information on the surrounding of the car. As this sensor is also quite expensive, it is also only used as a verification device and not used for visual odometry or state estimation itself.

Others Additionally, other car sensors like damping sensors are made available. However, due to the lack of documentation these sensor data cannot be used at the current stage.

Additionally to the sensors, the V-Charge cars also feature several computers for onboard data processing. As four camera images at a frame rate of 12.5 Hz need to be processed, the computer used for localization and mapping features state-of-the-art hardware. A reference PC which mirrors the specification of the onboard PC is used for testing.

Reference PC The reference PC features a quad-core (8 threads) Intel I7-2600 processor running at 3.4 GHz. The processor is supported by a total of 12 GB RAM. For computationally complex processes a NVIDIA Geforce GTX 560 Ti with 1 GB RAM and 384 Cuda cores is added.

1.1.3 The Two Cars

For the V-Charge project VW is providing two cars with identical base (VW Golf V). For easier referencing the vehicles were named after the Sesame Street characters "Grobi" (German for "Gromit") and "Kermit". Accordingly, Grobi is coloured in blue while Kermit is green.

While the base and the onboard computers are the same for both cars, their sensor layout differs. Kermit features a basic sensor layout with the four fisheye cameras, the stereo camera and all onboard sensors (odometry, damper sensors etc.). Grobi is equipped with an additional iTrace GPS unit plus a Velodyne laser scanner, both used for verification. During the early stage of the project both cars were developed and maintained on the VW site. A few months after the integration week Kermit was delivered to ETH Zurich and can be used for testing and evaluation.

1.1.4 Software Environment

As many soft- and hardware components need to be integrated in the project, a middleware is required. Due to the fact that most existing hard- and software components are already developed for VW's proprietary middleware ADTF, it will be used as the major integration point for all software. Hence, the software developed within the V-Charge project has to integrate or at least interface with ADTF.

Due to its high speed and the great availability of well-tested, widely-used libraries, C++ is used as a development language. While this allows seamless integration with ADTF,

the software developed within the presented thesis is written in cross-platform, ADTF independent code and only features a tiny interface to ADTF itself. Thus, the software or parts thereof are easily portable onto other systems.

The localization pipeline is based on and contributes to the new ASLam toolbox developed at ASL. This toolbox provides a versatile software environment which can be used to rapidly develop SLAM like systems for all kinds of sensor configurations.

For mid-level image processing OpenCV is used. Matrix and vector operations are handled using the Eigen Library. Furthermore, boost, SuiteSparse and the Schweizer-Messer libraries are used.

1.2 Visual Localization for V-Charge

1.2.1 Goal of the Project

The goal of the presented thesis is to provide a highly accurate and robust online localization for the vehicle using only the cameras and integrated wheel odometry. The algorithm should be able to process the sensor data in realtime and provide a precise pose estimation with low drift over time.

At a later stage, the localization algorithm should integrate with a preprocessed map. However, for early testing and to increase the robustness, also an online map should be generated, i.e. new landmarks should be triangulated and tracked online. Thus, even though it targets visual odometry, the developed software is a full SLAM framework only lacking loop closure detection.

1.2.2 Starting Point

As visual odometry is one of the early deliverables of the ASL in the V-Charge project, the thesis is started from scratch.

At the beginning of the project, the reference PC with a standard Windows installation is provided. On this PC there is no software development environment set up nor is the proprietary ADTF software or any other library installed.

Additionally, for the first two months of the thesis¹, only sample images but no datasets are available. Also, there is still a lack of documentation and therefore, it is often not clear what the interfaces and data structures look like. Furthermore, also for the other V-Charge member groups, this is a special project resulting in experimental hard- and software. Thus, as a later analysis will show, the data provided is partly defective and suffers from other effects (quantization).

As mentioned above the ASLam library will be used in the project. While the development is targeted at all platforms, it has never been run on a Windows before. This is especially critical as it depends on libraries like SuiteSparse which do not official support

¹This is equal to one third of the total thesis.

a native Windows build system. Additionally, ASLam is still under development and thus some desired features are not yet implemented.

Due to this difficult starting point a lot of work is invested into integration work and setting up a good development environment. These important steps will not only enable to develop a localization pipeline but will also provide a solid base for future projects.

1.2.3 State-of-the-Art Localization for Multi Camera Systems

In recent years, vision is increasingly used for odometry, localization, state estimation and navigation in robotics. This is mainly enabled through more sophisticated cameras as well as sensors (e.g. Microsoft's Kinect) and high computational power becoming available at a low price. Through their images, cameras provide a rich and dense measurement of the environment. The content of information is so high that only a small part of it can currently be used in today's implementations. Thus, there is still a huge unexploited potential for vision based systems.

While the Kinect sensor boosted the use of stereo cameras in robotics, there is a demand for a higher coverage of the environment. Therefore, multi camera systems and/or wide angle (e.g. fisheye and catadioptric) lenses are becoming increasingly popular. The systems used vary from custom made camera rigs to fully integrated omnidirectional camera systems like Point Grey's LadyBug cameras.

Multi Camera Systems

The approaches for designing a SLAM system for multiple cameras can be quite different. One possible way is to use monocular SLAM to create an individual map for each camera and then fuse all created maps. Even though used for calibration, the system developed by G. Carrera et al. ([5]) falls into this category. In their approach, the authors use a very classical approach with SURF features, an extended Kalman filter for state estimation and bundle adjustment for map refinement. Another approach where cameras are handled separately for a big part of the system is proposed by Kazik et al. ([18]). In their work, they first generate motion estimates for each camera individually before the estimates are fused together.

An approach where all cameras are fused together prior the mapping stage is presented by J. Kim et al. ([19]). Their approach is again a very common approach (RANSAC, bundle adjustment), however limited to a spherical camera configuration, where the optical centers of the cameras intersect in a single point.

While the approaches mentioned above did not have or use overlapping field of view between the cameras, an interesting approach is presented by J. Tardif et al. ([26]). In their paper, the authors show a SLAM approach where translation estimation is decoupled from rotation estimation. Translation is estimated by matching against the

map whereas the overlapping field of view is used to estimate rotation. The authors also emphasize the importance of overlapping field of view for multi camera systems. Another interesting multi camera system SLAM approach is proposed by M. Keass and F. Dellaert ([15]). In their system, a rig of perspective cameras are used with multiple optical centers (as opposed to [19]) but without overlapping field of view. The authors use Harris corners as features and an expectation maximization for finding correspondences as well as estimating the pose.

Multi Camera Systems on Cars

Camera systems also attract more attention in the car industry. Especially omnidirectional, catadioptric cameras are used (e.g. [10]). The purpose is often to generate overhead images for monitoring the close surroundings of the car, e.g. for parking. An overview of such systems is presented in [25].

With increasing demand for fully autonomous cars, vision systems are increasingly often used for pose estimation and obstacle detection. Many sophisticated systems like the autonomous car developed by Google or projects presented within the Darpa Grand Challenges depend on additional sensors like laser scanners or radar. But also lower cost, vision-only systems start to become more popular. Especially pose estimation is often handled by vision only (e.g. see [24] and [26]).

Summary

Even though many approaches exist, there does not seem to be a solution for multi camera systems that use fisheye lenses. Subproblems of the presented task have been tackled in other projects, however an integral solution for efficiently handling highly distorted images with overlapping field are missing. Additionally, many systems are designed for small ground robots instead of cars and thus are designed for different scenarios.

1.2.4 Localization Pipeline

The approach taken in this project consists of a feature-based, SLAM-like visual odometry front-end using a windowed bundle adjustment back-end. Additionally, this visual odometry is paired with a continuous time batch estimator developed within a separate master thesis (see [7]). While developed for four cameras, a generalist approach is taken which makes the system suitable for any multi camera system (with and without overlapping field of view) independent on the number of cameras, the lens type or the camera configuration.

An overview of the structure of the localization pipeline is shown in figure 1.2. In a first step, images and odometry are synchronized, i.e. odometry is extrapolated to fit

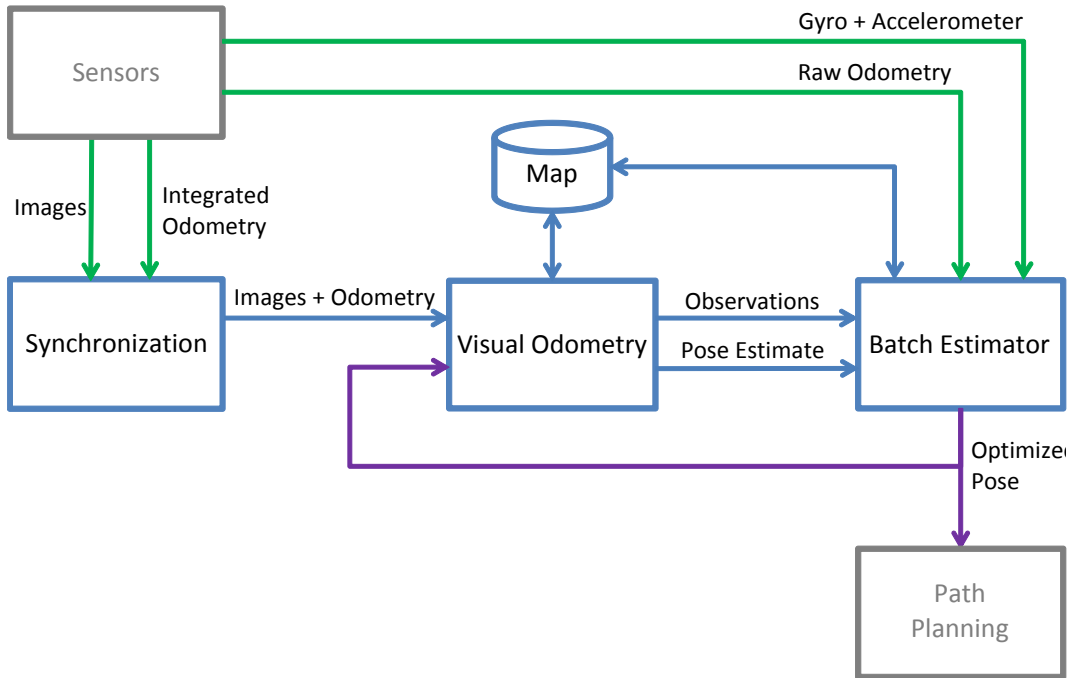


Figure 1.2: Outline of the Localization Pipeline

the timestamp of the image (see section 6.3.1). Then the visual odometry pipeline processes the incoming data stream to update and localize against the map. The found observations as well as an estimated pose is then fed to the batch estimator. The batch estimator then refines the estimated pose using additional sensor data and applies a large window bundle adjustment on the map. Finally, the optimized pose estimate is fed to the following pipeline steps, e.g. path planning.

The focus of the presented thesis lies on the Visual Odometry (VO) part of the localization pipeline. A more detailed overview of the VO sub-pipeline is shown in figure 1.3. As optimization and outlier removal are closely linked to the map, they are referred to as the back-end (see chapter 4) in the following chapters. The remaining elements deal with images and features and are thus called front-end (see chapter 3).

1.3 Notation and Coordinate Systems

1.3.1 Notation

Font example	Variable Type
a	scalars
$\vec{a}$	3D vectors
A	Matrices
$P_c\{x, y, z\}$	Point in 3D space expressed in coordinate frame c

Table 1.1: Notation used in this document.

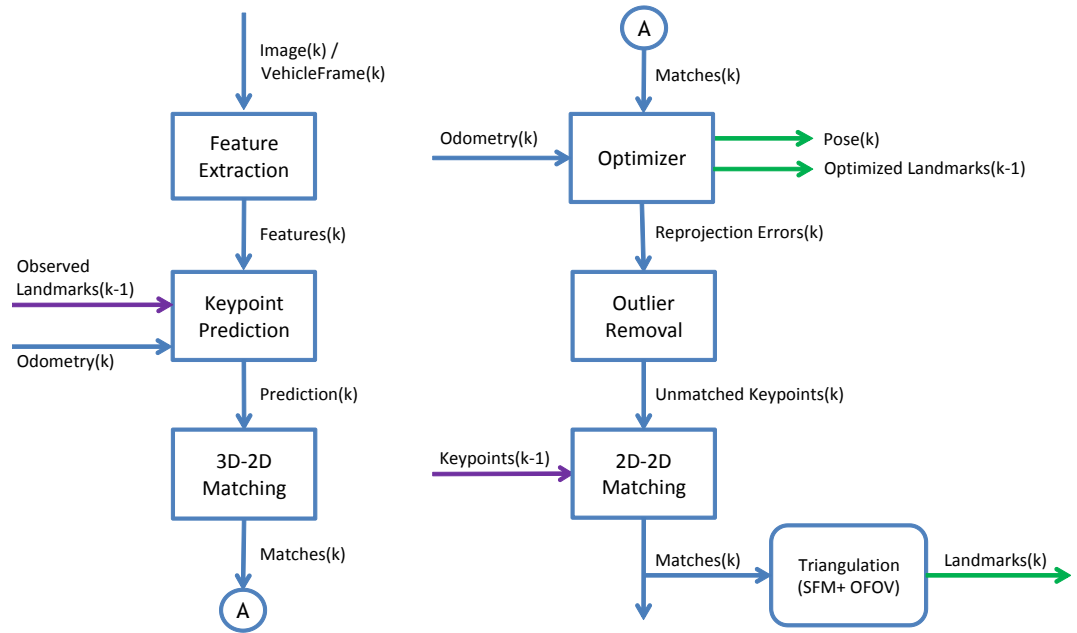


Figure 1.3: Outline of the Visual Odometry Pipeline

In the presented thesis, common text book notation is used. Table 1.1 shows a brief overview of the notation used for the different variable types.

1.3.2 Coordinate Frames

Due to the different types of sensors used in the project, several coordinate frames have to be defined.

World Reference Frame (WRF)

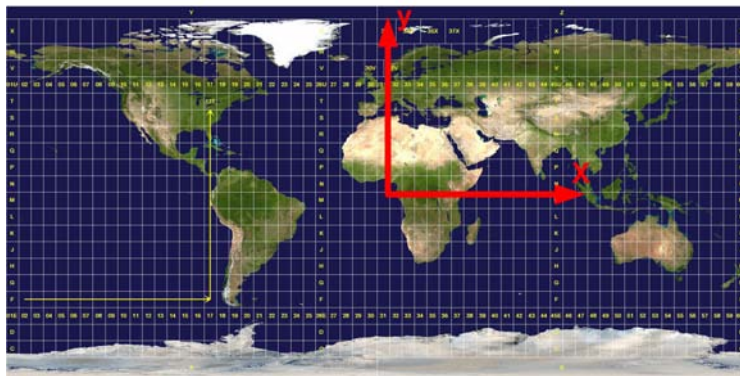


Figure 1.4: Globally defined World Reference Frame [28]

The World Reference Frame is globally defined and attached to the world (see figure 1.4). Due to its large scale, it is used seldom in this project. However, the iTrace GPS sensor delivers data defined in this frame.

Local Reference Frame (LRF)



Figure 1.5: Local Reference Frame attached to the scene [28]

The local reference is a coordinate frame that is attached to the map (or local scene) as illustrated in figure 1.5. Thus it is similar to a global (or world) coordinate frame with the exception that it is initialized by placing its origin to a selected point in the scene. Thus, there is a constant offset of this reference frame to the world reference frame. If no offline map is available the LRF is initialized at the vehicles starting position.

Vehicle Reference Frame (VRF)

The Vehicle Reference Frame is important for the sensors located on the vehicle. The VRF is located on the rear axle of the car (see figure 1.6). The positions of the sensors are defined with respect to the VRF and thus this coordinate system serves as a common reference for wheel odometry and visual odometry.

1.4 Structure of the Report

The presented report is split into six chapters. After this introduction, chapter 2 explains the camera model used in the project and gives a brief insight into the calibration process and results. Following up is chapter 3 about the front-end covering the process of feature extraction, feature matching and triangulation. In chapter 4, the back-end approach including optimization, pose estimation and outlier rejection is described. Chapter 5 covers the implementation of the chosen approach. Performance evaluation and tests are presented in chapter 6. Chapter 7 concludes the thesis and proposes future work.



Figure 1.6: Vehicle Reference Frame attached to the vehicle [28]

2 Camera Model

2.1 Distorted Fisheye Camera Model

In order to get accurate projection results, an appropriate projection model as well as a distortion model is selected. As fisheye and omnidirectional cameras become increasingly popular in robotics, the selection of available camera models also increased.

To come up with a reasonable calibration quickly, a camera model with an included calibration is chosen. The camera model used in this project is the one proposed by C. Mei [21] which itself is an adaptation of the model developed by C. Geyer et al. ([11]) and J. Barreto et al. ([1]).

2.1.1 Brief Description

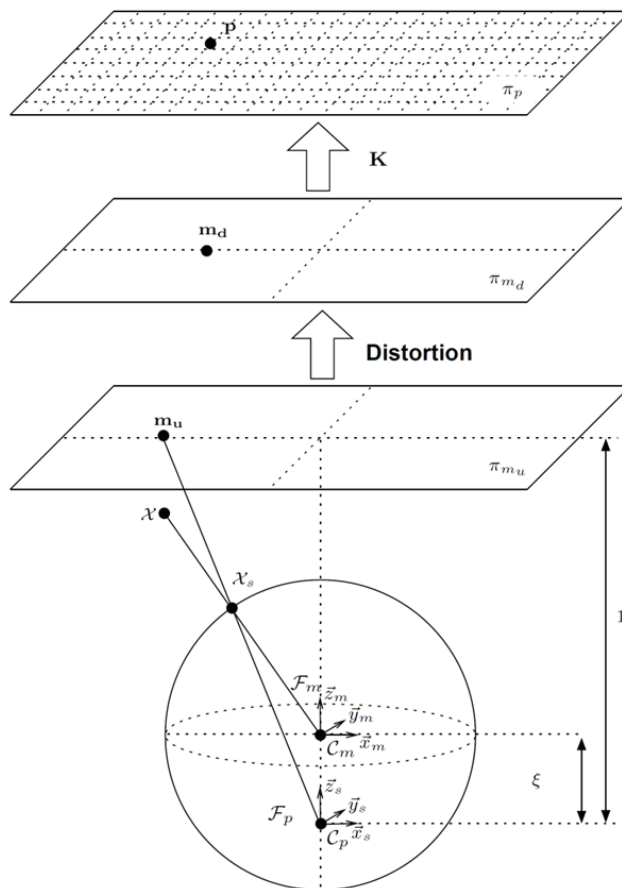


Figure 2.1: Spherical camera model as proposed by C. Mei [21]

The selected model features a spherical projection with a shifted principal point (denoted by ξ in figure 2.1) and adapted focal length. In this way, various camera types including omnidirectional, fisheye, planar and perspective cameras can be modelled. Depending on the camera type, the shifted principal point does not necessarily coincident with the center of the projection sphere. The benefit of this projection model however is, that the distortion function which is added at a later stage is quite simple and usually does not warp the projection very heavily.

2.1.2 Projection Process

The required steps for projecting a point defined in the world reference frame onto the image (or pixel) are the following

1. Projection of the point in space onto unit sphere
2. Change of coordinate frame to shifted principal point
3. Projection onto the normalized image plane
4. Adding of distortion
5. Projection into pixel space

Let $P_w = \{X, Y, Z\}$ denote a point in space. In the first step, this point is projected onto the unit sphere located at the origin. Hence, we can simply compute the projection by

$$P_s = \frac{P_w}{\|P_w\|} = \frac{1}{\sqrt{x^2 + y^2 + z^2}} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} := \begin{pmatrix} x_s \\ y_s \\ z_s \end{pmatrix} \quad (2.1)$$

In the second step, the projected coordinates are then transformed to the shifted principal point which is translated from the center point of the sphere by ξ in negative z-direction. This translation can be applied by simply adding ξ to the z-coordinate of our projected point

$$P_t = P_s + \begin{pmatrix} 0 \\ 0 \\ \xi \end{pmatrix} = \begin{pmatrix} x_s \\ y_s \\ z_s + \xi \end{pmatrix} := \begin{pmatrix} x_t \\ y_t \\ z_t \end{pmatrix} \quad (2.2)$$

Please note that we are using a slightly modified notation than the one introduced by C. Mei.

Following the translation is step #3 where the point in space is projected onto the normalized image plane. This image plane is located at $z = 1$ ¹. The projection onto the normalized image plane can be seen as a regular camera projection (e.g. for perspective camera) with a focal length of 1. Hence, this image plane is called the normalized image plane.

$$m_u = \begin{pmatrix} x_t/z_t \\ y_t/z_t \\ 1 \end{pmatrix} := \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} \quad (2.3)$$

On the normalized image plane, distortion is then added in step #4. The distortion function used is the same as the one implemented in the camera calibration in OpenCV. It consists of two radial and two tangential distortion coefficients² denoted as k_1, k_2 and p_1, p_2 respectively. The distortion function is non-linear in both parameters and image coordinates and is defined as follows

$$m_d = \begin{pmatrix} x'(1 + k_1 \cdot r^2 + k_2 \cdot r^4) + 2 \cdot p_1 \cdot x' \cdot y' + p_2(r^2 + 2x'^2) \\ y'(1 + k_1 \cdot r^2 + k_2 \cdot r^4) + 2 \cdot p_2 \cdot x' \cdot y' + p_1(r^2 + 2y'^2) \\ 1 \end{pmatrix} \quad (2.4)$$

where

$$r^2 = x'^2 + y'^2 \quad (2.5)$$

Finally, the distorted points are projected into pixel space by a simple (perspective) projection using the adapted camera matrix.

$$p = K \cdot m_d = \begin{pmatrix} \gamma & 0 & u0 \\ 0 & \gamma & v0 \\ 0 & 0 & 1 \end{pmatrix} \cdot m_d \quad (2.6)$$

Instead of the regular camera intrinsics, the camera matrix K contains the transformed focal length γ which is depending on the camera type.

2.2 Perfect Spherical Camera

As described above, the omnidirectional camera model is a linear model with the only exception being the sphere projection and the distortion function. In contrast to the sphere projection, there is no simple inversion of the distortion function. This is es-

¹The z-coordinate is propagated through the next steps to use a simple matrix-vector multiplication for the final projection.

²OpenCV 2.4 provides additional distortion coefficients that are resolution/scene dependent, that are not used in this project.

pecially a problem when projecting a point from the image plane into space (e.g. for triangulation). To invert the distortion, one can use iterative algorithms and numerical approximations. However, these are computationally expensive.

To avoid this issue, it is possible to undistort the keypoint (or the whole image if necessary). This undistortion simply removes the distortion as described in section 2.1.2 but still uses the camera model with the shifted principle point. Hence, the undistorted images do not look like images taken by a perspective camera as they still contain the deflection of the sphere. The undistortion itself is a pixel to pixel mapping between the distorted and the undistorted image (or image section) and can be precomputed for efficiency.

An example for undistorted images is given in figure 2.2. Figure 2.2b shows an undistorted image using a smaller value of ξ and different focal length but similar undistortion parameters. This nicely illustrates that the overall distortion is not only depending on the distortion function parameters but also on focal length and shifted principal point.

2.3 Epipolar Constraint

2.3.1 About the Epipolar Constraint

The epipolar constraint is often known from stereo cameras but is applicable to all cases where two views of a scene are available. The only requirements are one or more calibrated cameras and a known transformation between them. This transformation can be either fix, e.g. in a camera configuration with overlapping field of view, or dynamic, e.g. a motion between two camera poses.

The idea behind the epipolar constraint is that a point on the image plane limits the possible location for the underlying landmark to a line ("line of sight"), which is defined by the principal point of the camera (O_L in figure 2.3) and the viewing direction given by the feature (denoted by x_L) on the image plane. In other words, from a 2D observation the landmark location (denoted by X) can be estimated up to depth.

Now if this line is projected onto a second image, the search for a corresponding keypoint which emerged from the same landmark as the keypoint in the first image can be limited to this so called epipolar line (plus an uncertainty area around it). Thus, keypoint matching accuracy and robustness can be increased when using the epipolar constraint. All projected epipolar lines intersect in one point: the principal point of the other camera (in the case of overlapping field of view) or the principle point of the same camera at its old pose (in case of structure from motion). These locations of the principal points in image space are often referred to as the epipoles.

Depending on the camera type the projected epipolar line can have different shapes. On undistorted perspective cameras the epipolar lines are straight. Also it is possible to

(a) $\xi = 2.23$ (b) $\xi = 0.87$

Figure 2.2: Comparison of perfect (undistorted) spherical camera images (right camera) for different values of ξ

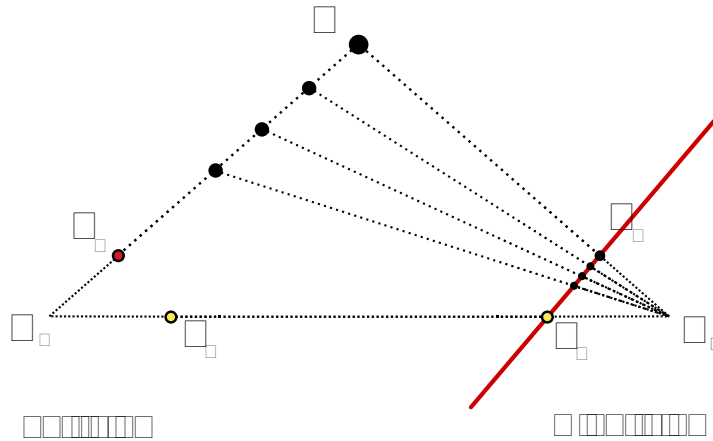


Figure 2.3: Illustration of the epipolar constraint [29]

rectify the image which results in epipolar lines which are horizontal on the image plane and perfectly straight. This makes searching along the epipolar lines easier.

However, on distorted cameras and non-perspective cameras the epipolar line becomes an epipolar curve which can have different shapes. For more details please refer to section 2.3.3.

2.3.2 Generalized Epipolar Constraint

The generalized epipolar constraint uses a so called fundamental matrix (or essential matrix for normalized image coordinates) to describe the epipolar constraint. For a full derivation of the fundamental matrix, please refer to [13]. Below, a short overview of the resulting generalized epipolar constraint is given.

The fundamental matrix (denoted by F) contains the projection matrices of the cameras as well as the relative transformation between the camera poses. Therefore, it allows to calculate the epipolar line in a first image ($\vec{l}_1$) for an image point in a second image (x_2)

$$\vec{l}_1 = F \cdot x_2 \quad (2.7)$$

Equivalently, the transpose of the fundamental matrix also maps points to epipolar lines in the other direction

$$\vec{l}_2 = F^T \cdot x_1 \quad (2.8)$$

Furthermore, to test if a point lies on the epipolar line defined by another point, the relation

$$x_1^T \vec{l}_1 = 0 \quad (2.9)$$

must be satisfied. When substituting $\vec{l}_1$ with equation 2.7 the generalized epipolar constraint

$$x_1^T F x_2 = 0 \quad (2.10)$$

is derived.

2.3.3 Epipolar Constraint for Spherical Cameras

The generalized epipolar constraint explained above only works for simple projection types where projection matrices are used. However, the spherical model includes additional projection steps. One possibility to apply the generalized epipolar constraint to spherical cameras is to use the spherical model up to the last projection step which then normally use the projection matrix. At this point, the generalized epipolar constraint can be applied.

As shortly mentioned in section 2.3.1 the epipolar line turns into an epipolar curve for the spherical camera model. The reason for that is the spherical projection which transforms epipolar line into a curve. Additionally, the off-centric principal point further warps the curve.

In order to find an analytical equation for the epipolar line, first the epipolar line ("line of sight") in 3D space is defined

$$\vec{l} = \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix} + \begin{pmatrix} u \\ v \\ w \end{pmatrix} t \quad (2.11)$$

This line is a simple straight line in space, where the origin ($P_w = \{x_0, y_0, z_0\}$) is defined as the principal point of one camera and the direction ($\vec{v} = \{u, v, w\}$) is given by the inverse projected keypoint. The parameter of the straight line is denoted by t . By limiting t to $0 \leq t \leq \infty$ only points in front of the camera are described by the resulting half line. Thus, invalid points behind the camera are excluded.

Now, if $\vec{l}$ is imagined as an infinite set of points in space, these can be projected using the equations presented in 2.1.2. The resulting equation for the epipolar curve is

$$m = \frac{f}{w + \xi \cdot t \cdot (z_0 + \sqrt{(\frac{u}{t} + x_0)^2 + (\frac{v}{t} + y_0)^2 + (\frac{w}{t} + z_0)^2})} \begin{pmatrix} u + x_0 \cdot t \\ v + y_0 \cdot t \end{pmatrix} \quad (2.12)$$

As equation 2.12 shows, the epipolar curve is nonlinear in the line parameter t . Hence, the epipolar curve does not resemble a straight line.

Furthermore, the non-linearity makes it difficult to come up with an equation for calculating the distance of a point to this curve. However, this distance calculation is essential for epipolar matching. Of course it is also possible to find the closest distance with iterative solutions. However, these are computationally expensive. A novel solution that still enables efficient epipolar matching with spherical cameras is described in section 3.3.1.

2.4 Verification of the Camera Model

Before using the camera model, a proper verification of the camera model needs to be done. The spherical model has been widely used for omnidirectional cameras. Courbon et al. recently also verified the model for fisheye cameras ([6]). Furthermore, the authors show that it also works well with radial tangential distortion which is also used in this project.

Additionally, the Computer Vision and Geometry Group at ETH Zurich verified the theory behind the model for the V-Charge project (currently unpublished).

2.5 Camera Calibration

The calibration requires finding the correct extrinsic and intrinsic parameters of the camera which reflect the projection as accurately as possible. The extrinsics define pose and orientation of the camera while the intrinsic parameters define parameters related to projection model, i.e. focus length and image center.

2.5.1 Calibration Process

The camera calibration itself is not part of this project. Thus, calibration from external sources is used. However, for an accurate SLAM-like system a good calibration is required. Thus, the used calibration process as well as the incoming calibration data needs to be checked.

Extrinsic Calibration

The extrinsic calibration is done by Bosch. The calibration data is delivered as is with little documentation about the calibration process and environment. Therefore, the provided data has to be accepted as is. If necessary, extrinsic calibration will be repeated by the ASL.

Intrinsic Calibration

In the early stage of the project, the initial calibration was done using checkerboard images and the Christopher Mei Toolbox. However, this toolbox is implemented in Matlab and thus suffers from speed issues. Additionally, the automated corner detector often fails to detect the checkerboard corners in highly distorted areas of the image. Hence, CVG has developed a C++ based tools using the Google Ceres optimizer and the OpenCV checkerboard corner finder.

During calibration an optimization over all calibration images is run. One issue is that the spherical camera model has two local minima for the calibration of the V-Charge cameras. The first minima is located at a shifted principal point value of $\xi \approx 2.0$. The second minima corresponds to a value of $\xi \approx 0.9$. The offset in the location of the principal point is compensated by the focal length. The distortion parameters remain on the same level. However, due to the increase of the focal length, the distortion has a bigger influence which can be seen in figure 2.2 which shows images undistorted to mimic perfectly spherical cameras. At the current stage, it is hard to tell which calibration minimum of the two is better. According to CVG, both are able to reach low reprojection errors during calibration and thus should work.

2.5.2 Calibration Results

To verify the calibration quality, an epipolar curve drawing tool is developed. This tool allows to click on any point in a first image and will draw the epipolar curve for this point in a second image. Of course, this only works if the clicked point lies in the overlapping field of view of both cameras. As the epipolar line is depending on intrinsic and extrinsic calibration of both cameras, the tool is a integral verification for the overall calibration. However, the extrinsics are only verified between cameras and not between the cameras and the vehicle. The latter is also important for the project but they cannot be checked using images only.

Calibration for Grobi

As with both cars, first the extrinsics have been calibrated by Bosch while the intrinsic calibration is done by CVG.

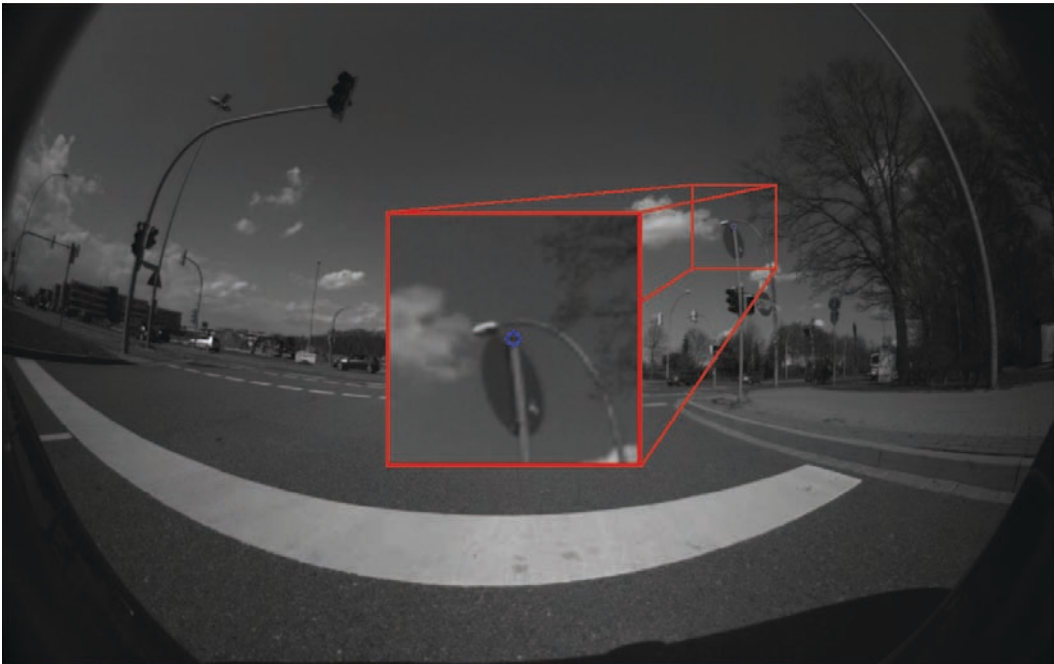
As Grobi is present at VW only datasets from the integration week are available. The calibration has been done with the same configuration of camera and no alterations have been made between calibration and dataset. Hence, the calibration result is good and even in outer regions of the image, the epipolar line almost perfectly intersects with the selected reference point (see figure 2.4b).

Calibration for Kermit

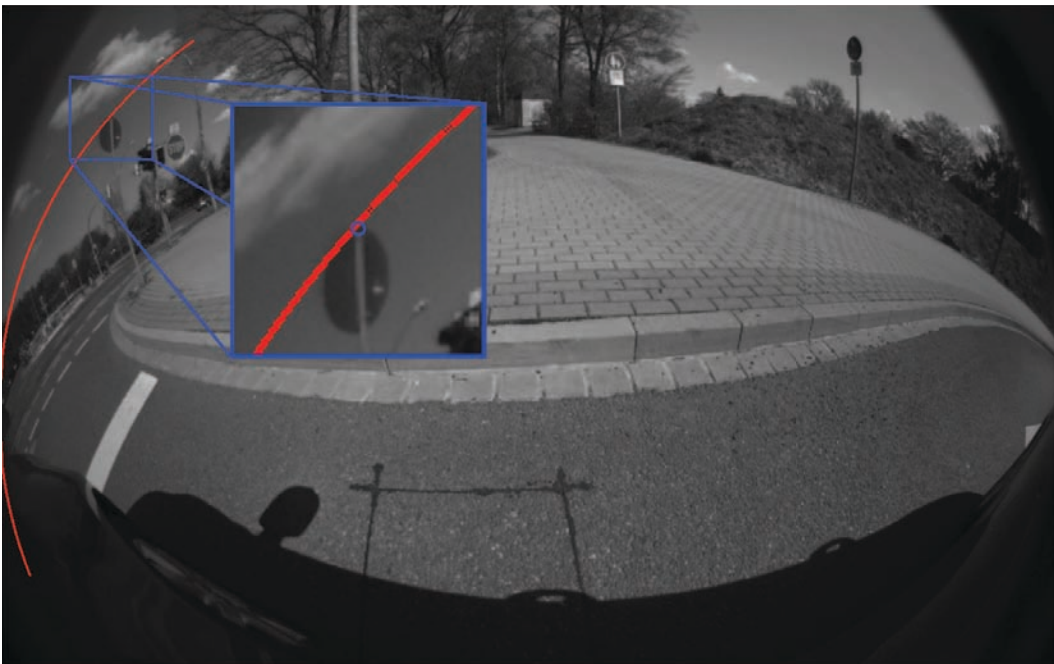
The extrinsic calibration for Kermit was done before the integration week. However, before the car was shipped to ETH Zurich, two cameras (front and left) were exchanged and maintenance work was done on the others. Hence, the provided extrinsic calibration is not valid any more.

While the intrinsics are recalibrated the overall calibration result is not very convincing. Figure 2.5 shows the output of the calibration verification tool. A point (blue circle) is selected in the front image (figure 2.5a) while the epipolar line (red line) gets drawn in the right image (figure 2.5b). As it can be seen, the epipolar line is quite far away from its corresponding point (blue circle, hand-picked).

While calibration is better for points closer to the center, calibration is still not accurate. This poses several problems for the visual odometry pipeline. Especially the cone matcher described in section 3.3.2 relies on a good calibration. Of course it is still possible to run the pipeline with a bad calibration, but it will heavily influence the pose estimation quality. Thus, a new calibration for Kermit is needed.

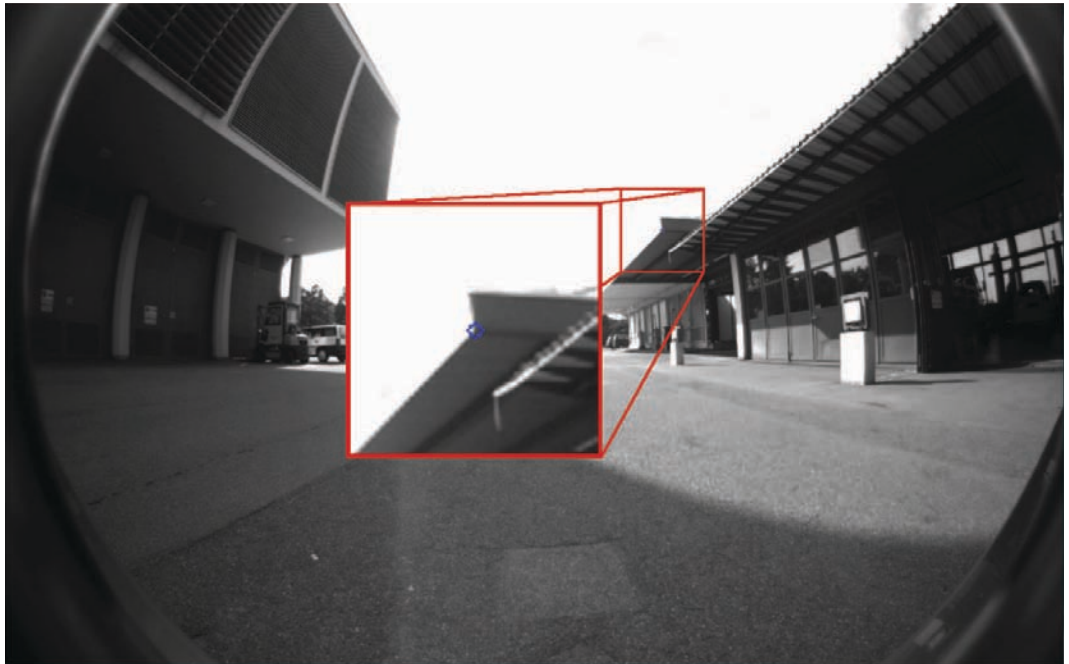


(a) Front camera image with selected point (blue) in overlapping field of view



(b) Right camera image with epipolar line (red) and hand-picked reference point (blue)

Figure 2.4: Output of the epipolar line based calibration verification tool for Grobi



(a) Front camera image with selected point (blue) in overlapping field of view



(b) Right camera image with epipolar line (red) and hand-picked reference point (blue)

Figure 2.5: Output of the epipolar line based calibration verification tool for Kermit

3 Front-End

The term "front-end" describes the pipeline steps between incoming images and triangulated landmarks. The main elements within the front-end are feature extraction, feature matching and landmark triangulation (see figure 3.1).

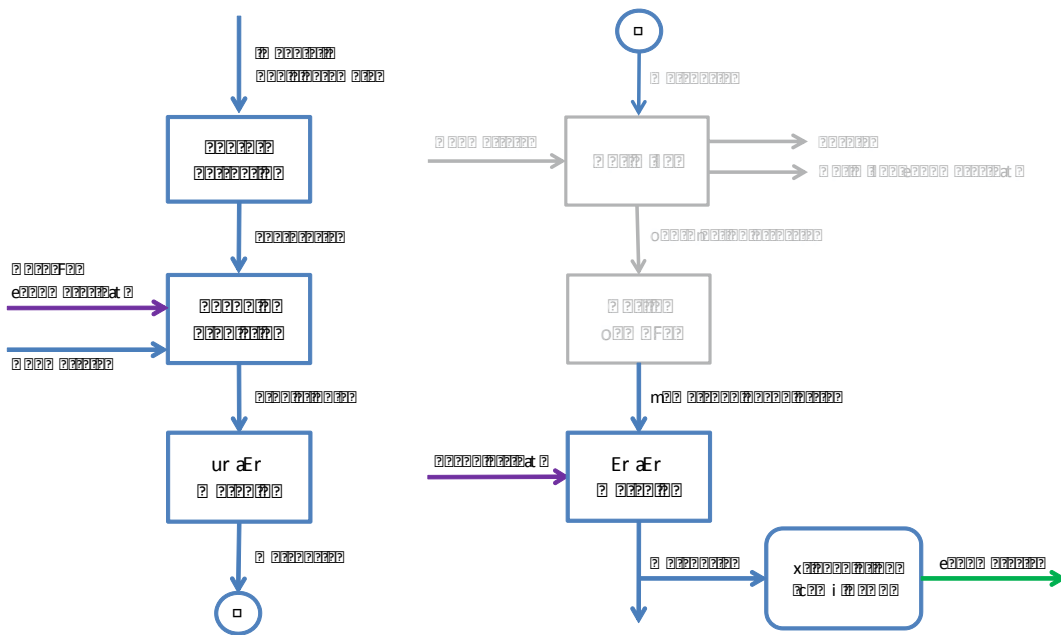


Figure 3.1: Overview of the localization pipeline highlighting the front-end

3.1 Feature Extraction

Feature extraction is a definition for finding characteristic points of an image and calculating a distinct, image independent representation (the descriptor) for these points. The goal is to create correspondences between images by matching features with each other. Hence, feature extractors should be robust to noise but also distinct in their descriptors.

3.1.1 State-of-the-art Feature Detectors and Descriptors

In today's computer vision, there is a great variety of feature detectors and descriptors available. In general, most feature detect significant image patches (e.g. SURF [2]) or a significant shapes like corners (e.g. Harris corner detector [12]) or edges (Canny edge detector [4]). Most of these detectors work by shifting a detection window over

the image, computing a score and finally using global or local maxima of these score functions. Additionally some detectors work on the regular image as well as down sampled versions of it (so called "octaves").

Mostly, feature detection and description is bound together and hence the combined algorithm is called feature extraction. However, detection and description can also happen in separate steps. Hence, it is also possible to combine different feature extractors, e.g. FAST corner detection with SURF descriptors.

As created for different purposes, feature extractors and their properties can be very different. Depending on the application, the optimal feature extractor choice can vary. To narrow down the possible choices and finally select the most appropriate extractor, a list of requirements is created:

- Scale invariance
- Rotation invariance
- Available implementation
- Computational inexpensive extraction

Scale and rotation invariance are very important in the presented project. Due to the large field of view and the special geometry of the fisheye lenses, an object in the outmost region of the image appears smaller than the same object in the center. Hence, also the scale of the feature is different. In order to be still able to match such kind of a feature pair, scale invariance is needed. Also the orientation of features varies over different image areas. The orientation of a feature that moves clock-wise on the outer edge of the visible field of view will also rotate clock-wise. Hence, rotation invariance is also required.

Additionally, the localization pipeline integrates several different components. In order to keep the overall complexity of the project at a reasonable level, only feature extractors with robust implementations that can be used out of the box are taken into account.

As the pipeline should run in real time and works with quite high resolution images, the extractor should additionally be computationally inexpensive. That means that its computational complexity must be reasonable and its implementation must be optimized for speed.

One source for great implementations of feature extractors is the OpenCV library. But there are also other feature extractors outside of OpenCV that show great potential (e.g. BRISK [20]).

3.1.2 Selected Feature Extractors

Based on the criteria above, the choice is narrowed down to SURF and BRISK. Both extractors fulfil the criteria listed above. They are scale and rotation invariant and well tested implementations exist.

SURF provides a good trade-off between descriptor strength and speed. Additionally, a GPU version of SURF is included in OpenCV which promises good performance.

BRISK uses binary descriptor which makes detection, description and matching computationally cheap. Thus, even CPU implementations for each of these steps run with great speed. However, due to BRISK's simpler descriptor structure, it might not be as precise as the SURF counterpart.

One big difference between both is that SURF extracts the globally strongest keypoints while BRISK uses a local non-maximum suppression to find local maxima. Both extractors have alternative implementations which allow global or local maxima detection. In general, both approaches can be helpful. On the one hand, globally strongest keypoints are more significant and thus possibly more stable. On the other hand an even keypoint distribution can make pose estimation more robust as it leads to landmarks with different distance to the camera and in different positions.

Therefore, it is not possible to take a final specification based decision but integration tests with the whole pipeline are necessary. Please see section 6.1.1 for a thorough comparison of both extractors.

3.2 Map Projection

For now, it is assumed that there is already a map with 3D landmarks. To match newly detected 2D keypoints to these 3D landmarks, the landmarks are projected onto the image creating keypoint predictions. Afterwards, a simple dense matcher (see section 3.3.1) can be used to match predicted and detected keypoints.

In order to predict the keypoint location, the landmark must be projected using the camera model. For this projection, the landmark location must first be transformed to the camera frame. For this transformation the current vehicle pose must be known. As the pose estimation already requires landmark to keypoint associations, the estimated pose is not yet available during this step. Therefore, wheel odometry is used for the keypoint prediction. The offset between projected and detected keypoint is therefore highly dependent on the wheel odometry which underlines the requirement for good wheel odometry.

As predicted and detected keypoints have to be matched, the keypoints also require a descriptor. As described in section 3.5 the descriptor from the last observation is used.

3.3 Feature Matching

In order to triangulate or to track a landmark over different poses or cameras, observations (i.e. keypoints) of this landmark from different images have to be matched. In general there are two types of matching. 2D to 2D matching and 3D to 2D matching. The first case is needed when matching keypoints in overlapping field of view or from different vehicle frames. In those cases, keypoints do not yet have an underlying landmark associated with them and hence the depth (distance to the camera) of the underlying landmark is unknown. Thus, when observing a feature in one image, its position in the other image cannot be predicted. However, the possible locations of the second observation can be limited to a so called epipolar curve, which is a curve (or line in the planar case) in image space. The epipolar line can be calculated based on camera geometry, vehicle movement and keypoint location (see section 2.3.1 for more details on the epipolar constraint).

In the second case, when landmarks are available (in an online or offline generated map), features (2D) need to be matched to landmarks (3D). As the landmark positions and the pose of the vehicle are known in these cases, it is possible to predict the location of a landmark in image space. Thus matching is quite simple as the search area for possible matches can be limited to a small search area according to the uncertainty of the landmark measurement and the estimated pose around the predicted landmark location.

In both cases often multiple matching candidates are possible. Therefore, the selection of the best match is based on the descriptor distance, i.e. calculating the difference (and thus the dissimilarities) between the descriptors of two keypoints. Finally, the match with the lowest descriptor distance is selected as the match.

3.3.1 Matching Strategies

Feature matchers can work in different ways and use different properties of a matching scenario to speed up matching.

The biggest difference in feature matching is the correspondence type. Most features create one to one matches, meaning that each keypoint in one image can only have one corresponding match in the other image. However, there are also matchers that allow one to many matches meaning that one keypoint can have several matching keypoints. One to one matchers provide easier to handle correspondences while one to many matchers offer a fall back alternative in case a match is detected as an outlier.

But also in the matching strategy, differences can be found. The very basic approach is a matcher that compares all keypoints from one image with all keypoints from a second image. This is often also referred to as brute force matching. However, the complexity of such matchers is quite high as the descriptor distance needs to be calculated for the

maximum number of combinations. Depending on the descriptor type this can require a significant amount of computing time.

Therefore, a common technique is to restrict the number of matching partners by adding additional conditions that need to be satisfied before the descriptor is compared.

One type of such a matcher is the dense matcher. It uses a distance threshold in image space and only features with a distance less than that threshold are checked for a possible match. This is a great concept when keypoint predictions are available as the number of possible matches can be strictly limited. However, when no prediction is available, such a matcher suffers from a major issue. One way to restrict the distance is to use the distance to the epipolar line. But there is one draw back there. Landmarks close to the vehicle can move significantly farther in image space between two subsequent images than far away landmarks. Thus, a small deviation in the measurement of a close by landmark can lead to a significant larger distance to the epipolar line than the same deviation of a far away landmark. This makes it very difficult to set a good threshold for the dense matching. Either the threshold suppresses nearby features or it allows too many matching candidates. Additionally, as described in section 2.3.3, due to the special geometry of the epipolar curve there is no quick and analytical equation for finding the distance of a feature location to this epipolar curve. Hence, calculating the distance requires an iterative solver which is computationally complex.

Summarized, a dense matcher promises great performance when a 1D (point) feature location prediction is available. This is the case for the 3D to 2D matching. However, when no prediction or only a non-linear 2D (line) prediction is available, a dense matcher will perform worse. Thus, a better solution has to be found for the 2D to 2D matching.

3.3.2 Uncertainty Cone Based Matching Approach

As described above, there is a clear need for a better matching strategy or at least for a better preprocessing step. Especially when using SURF or other expensive to compare descriptors, reducing the number of possible matches can improve speed. When using simpler descriptors, preprocessing can help to reduce the number of self-similar cases and thus reduce mismatches.

The following criteria define the properties of the new required matching strategy:

- Highly reduce number of possible matches
- Use available motion estimate
- Properly incorporate measurement uncertainty
- Computationally cheap

The criteria put difficult constraints on the new matching approach. On the one hand, the number of matching candidates should be effectively reduced while respecting measurement uncertainty. On the other hand, the preprocessing should be very efficient. Based on the criteria above, a new matching strategy called Uncertainty Cone Based Matching is developed. The main idea behind this strategy is that two keypoints can only emerge from the same landmark (and thus are only a valid match) if their inverse projections intersect in a single point in space. As estimating the depth of a landmark from a single 2D measurement is not possible, the inverse projection is defined by a semi infinite ray (the epipolar line) in space. This ray starts at the principal point of the camera and intersects with the keypoint on the image sensor plane.

Now, because there is uncertainty (image measurement uncertainty as well as a transformation uncertainty) between two frames, two inverse projective rays will almost never intersect perfectly in a single point (the landmark). So, the question arises how this issue can be handled correctly.

At first, calculating the shortest distance between both projective rays and putting a threshold on that, seems attractive. However, small measurements deviations for far away keypoints can lead to huge distances between the projective rays in space. So, the same problem as using the epipolar line occurs.

Uncertainty Cones

Therefore, some kind of adaptive distance threshold must be used. As the distance of both projective rays emerges from the measurement noise, the uncertainty of that measurement should be used. The uncertainty of a keypoint measurement is normally defined by an ellipse around the center of the keypoint. As Y. Kanazawa and K. Kanatani show in [17] feature extractors create keypoints which can be safely approximated with a circular uncertainty (a diagonal covariance matrix). In the presented case, the uncertainty is approximated as a circle with the diameter being the largest entry of the covariance matrix. This works for a circular as well as elliptic uncertainty. In the case of elliptic uncertainty, the uncertainty is simplified but also artificially increased. As it can be seen later, this might lead to an increased number of possible matches. But as the keypoint uncertainty is almost circular, the increase in possible matches will also be small. The result of the inverse projection of this uncertainty circle is an uncertainty cone, where the tip of the cone intersects with the principal point of the camera. Such an uncertainty cone is illustrated on the left of figure 3.3.

Configuration of Two Uncertainty Cones

Thus, when looking at two features it is possible to compare their uncertainty cones. When comparing two uncertainty cones and the minimal distance between their central axes, four different configurations are possible (as also illustrated in table 3.1).

minimal distance	cones	valid match	average # of cases
in front	intersecting	yes	1.0%
in front	skew	no	17.8%
behind	converging	yes	0.4%
behind	diverging	no	80.8%

Table 3.1: Four different cases for two uncertainty cones in space

The simplest case is when the closest point of the central axes is in front of the camera and both cones intersect. Obviously, this defines a valid match. In contrast, if the closest points are in front of the camera but the cones do not intersect, than the pair cannot be a valid match.

When the closest point is behind the camera, the situation is a little special. First, both cones can be diverging in front of the camera, i.e. the distance between the cones increases with increasing distance to the camera. Therefore, the cones will not intersect in front of the camera and thus this is not a possible match. However, even if the closest points are behind the camera it is still possible that the cones intersect. This is the case if the central axis are slower diverging than the cones are "growing". This situation occurs when the angle between the central axes is smaller than the sum of the half cone angles. It might be counterintuitive to still define this as a match. However, especially with landmarks close or at infinity and slightly noisy measurements this can still occur also for valid cases.

As described, two keypoints are only a valid match if the two cones intersect in front of the camera. Hence, of the four possible configurations of two uncertainty cones in space, only two (case #1 and case #3) describe a valid match. The other cone combinations can be sorted out before comparing the descriptors of the underlying features. The benefit of this step becomes clear when looking at how often each case occurs. The cases which lead to a valid match occur quite seldom, together only in 1.4% of all cases. Hence, in 98.6% of all cases, no descriptor distance need to be calculated.

Comparing two uncertainty cones

Comparing two uncertainty cones in space first sounds like a computationally expensive task, especially compared to simply comparing descriptors. However, the underlying problem consist of simple calculations.

For the description of the algorithm, it is assumed that there are two cones, each described by a central axis (the epipolar line)

$$\vec{l}_w = \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix} + \begin{pmatrix} u \\ v \\ w \end{pmatrix} t = P_w + \vec{v} \cdot t \quad (3.1)$$

and the half opening angle of the cone (also referred to as half cone angle). Using the half cone angle instead of the full opening angle has computational advantages as shown later (see step #3).

In this special case, the location vector of the line in space corresponds to the tip of the cone which itself is the principal point of the camera.

Step #1: Shortest distance between two lines in space

In the first step of the comparison, the distance between the two central axes is computed. There are two ways of finding the shortest distance between two rays in space. As the line with shortest distance between both rays is perpendicular to them, a vector which satisfies this constraint has to be found. This is done by taking the cross product of the direction vectors of the two rays. Then any line connecting the two lines is projected onto the perpendicular vector. The simplest choice for the connecting line is the baseline between the two camera poses, which is simply the difference of the camera locations. As a last step the projected distance needs to be scaled by the perpendicular vector. Expressed in a formula, this can be described as

$$d = \frac{P_1 P_2 \cdot (v_1 \times v_2)}{\|v_1 \times v_2\|} \quad (3.2)$$

The downside of this formula is that it delivers the distance but not the location of the closest points on both rays. However, the latter are needed as the algorithm needs to distinguish whether the closest points are in front or behind the camera.

The second method of finding the shortest distance between two lines in space again uses the constraint of the connecting line with shortest distance being perpendicular to both vectors. This line in its most general form can be described as

$$\vec{d}_{1,2} = \vec{l}_1 - \vec{l}_2 \quad (3.3)$$

This line again has to be perpendicular two both lines:

$$\vec{d}_{1,2} = \vec{l}_1 - \vec{l}_2 = d \cdot \frac{\vec{v}_1 \times \vec{v}_2}{\|\vec{v}_1 \times \vec{v}_2\|} \quad (3.4)$$

Now substituting $\vec{l}_1$ and $\vec{l}_2$ with equation 3.1 leads to

$$\vec{d}_{1,2} = P_2 \vec{P}_1 + \vec{v}_1 \cdot t_1 - \vec{v}_2 \cdot t_2 = d \cdot \frac{\vec{v}_1 \times \vec{v}_2}{\|\vec{v}_1 \times \vec{v}_2\|} \quad (3.5)$$

$P_1 \vec{P}_2$ once again represents the baseline between the cameras. If the direction vectors are normalized beforehand, the equation can be transformed into a linear system of

equations

$$\begin{pmatrix} \vec{v}_1 & \vec{v}_2 & \vec{v}_1 \times \vec{v}_2 \end{pmatrix} \begin{pmatrix} t_1 \\ t_2 \\ d \end{pmatrix} = -P_2 \vec{P}_1 = P_1 \vec{P}_2 \quad (3.6)$$

This 3x3 linear system is easily solvable using the inverse of the 3x3 matrix

$$\begin{pmatrix} t_1 \\ t_2 \\ d \end{pmatrix} = \begin{pmatrix} \vec{v}_1 & \vec{v}_2 & \vec{v}_1 \times \vec{v}_2 \end{pmatrix}^{-1} \cdot P_1 \vec{P}_2 \quad (3.7)$$

This does not only provide the distance d , but also the parameters t_1 and t_2 for both lines indicating the location of the points with closest distance.

Step #2: Location of the points with closest distance

For distinguishing the different cases, it is required to know the points on both lines with closest distance and to check whether they are in front or behind the camera.

A simple way of checking if the closest points are in front of the camera is by defining $\vec{v}$ to be pointing from the camera's principal point towards the main viewing axis of the camera. If taking a look at how $\vec{v}$ is built, this can be easily achieved. $\vec{v}$ is calculated by first inverse projecting the keypoint with arbitrary but positive depth. From that $\vec{v}$ can easily be calculated

$$\vec{v} = L_w - C_w \quad (3.8)$$

where L_w is the inverse projected keypoint and C_w is the camera location. Once defined like this, the line parameter t determines the line direction. Hence, if t is positive the closest point is in front of the camera. This condition must be satisfied for both lines ($t_1 > 0$ and $t_2 > 0$).

Step #3a: Check if cones intersect (closest points in front of cameras)

If the closest points are in front of the cameras, the cones must intersect to form a valid match. If not parallel, the lines are converging up to the points with closest distance and then they are diverging again. Hence, the points with closest distance are great candidates for checking if the cones intersect. Virtually cutting the cone at this point reduces the problem to checking if two infinite thin disks (or circles) intersect (see figure 3.2). Also only the direction of the line connecting the points with closest distance needs to be checked as with growing cones this will be the first point of intersect. Hence, the problem can be decomposed into two 2D problems. Each problem lies in the plane of

one line and the perpendicular distance line. The result is a triangle as if the cone is sliced by the plane.

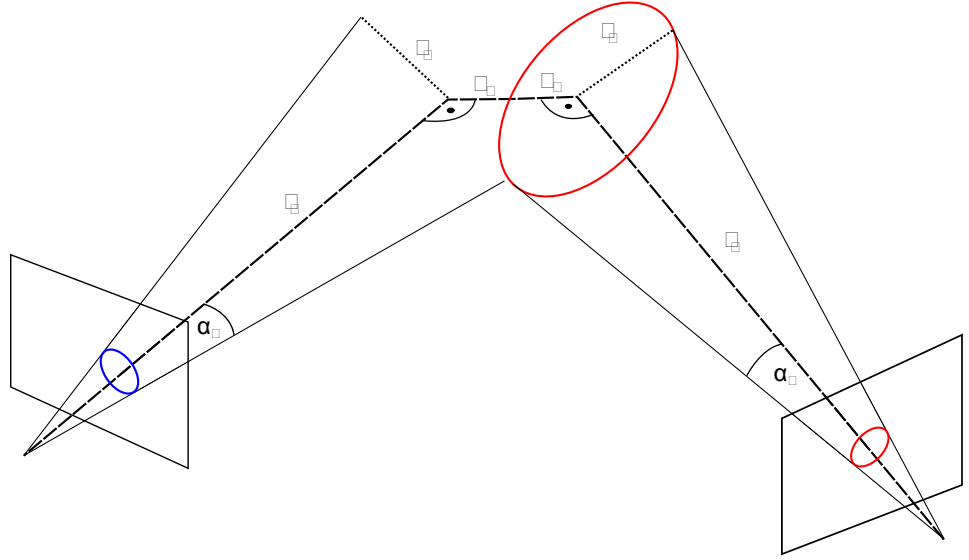


Figure 3.2: Illustration of intersecting Uncertainty Cones

In order to determine if the cones intersect, the sum of the radii (r_1 and r_2) of both circular cone slices at the points with closest distance must be larger or equal to the distance ($d = d_1 + d_2$) itself

$$d \leq r_1 + r_2 \quad (3.9)$$

The radii itself can be calculated by using trigonometric properties of the triangle resulting from the slicing

$$r = \tan(\alpha) \cdot t \quad (3.10)$$

t denotes the distance to the principal point and if $\vec{v}$ is normalized, this is the line parameter t . α denotes the half cone angle and thus can be precomputed for each keypoint and does not need to be recomputed for each pair.

Summarizing, if equation 3.9 is satisfied, the two keypoints are a valid match. If not, both keypoints cannot be a valid pair.

Step #3a: Check if cones intersect (closest points not in front of cameras)

If one or both points with closest distance are not in front of the cameras, the central axes of the cones are diverging in front of the cameras. However, the underlying keypoints

can still be a match¹. This is the case if the sum of both half cone angle is larger than the angle between both lines

$$\beta \leq \alpha_1 + \alpha_2 \quad (3.11)$$

The angle between both lines can be calculated as

$$\cos \beta = \frac{\vec{v}_1 \cdot \vec{v}_2}{\|\vec{v}_1\| \cdot \|\vec{v}_2\|} \quad (3.12)$$

Again, as $\vec{v}_1$ and $\vec{v}_2$ are normalized, the equation 3.12 can be simplified to

$$\cos \beta = \vec{v}_1 \cdot \vec{v}_2 \quad (3.13)$$

As uncertainty is limited $0 < \alpha_1 + \alpha_2 \leq \pi$ and $\beta < \pi$ and because $\cos \gamma$ is strictly monotonically decreasing on the interval $0 < \gamma < \pi$ equation 3.11 leads to

$$\beta \leq \alpha_1 + \alpha_2 \rightarrow \cos \beta \geq \cos(\alpha_1 + \alpha_2) \quad (3.14)$$

Hence, using this condition on equation 3.13 leads to

$$\cos(\alpha_1 + \alpha_2) \leq \cos \beta = \vec{v}_1 \cdot \vec{v}_2 \quad (3.15)$$

So in order for the cones to converging faster than the central axes are diverging and thus for the cones to intersect, this inequality equation needs to be satisfied.

One might ask why the assumption in equation 3.14 is necessary and if it would not have been easier to directly compare angles. The reason is from an implementational point of view: In order to calculate β it would be necessary to calculate $\beta = \arccos(\vec{v}_1 \cdot \vec{v}_2)$. However, the standard C++ implementation of the arccosine is much slower than the cosine implementation. Additionally, when uncertainty values are quantized, equation 3.15 allows using a look up table as the combinations of α_1 and α_2 are limited. This is not possible when directly comparing angles as it would require to create a look-up table for arccosine in equation 3.12. In that equation however, any combination between any two three dimensional unit vectors is possible and hence there is an infinite number of states making it impossible to create an accurate look-up table without wasting memory.

Uncertainty Cones and Transformation Uncertainty

In the current implementation, the uncertainty cones only incorporate the uncertainty of the keypoint measurement. However, there is also an uncertainty in the calibration

¹Accepting diverging cones seems counter intuitive at first. However, imagine a point at infinity. If they underlying keypoint moves due to measurement by only one pixel to either side the central axes are diverging, even though keypoints correspond to the same point.

and the pose estimation. The first influences the matching results in overlapping field of view while the latter influences matching in structure from motion. Unfortunately, the uncertainty of the calibration is unknown and also the uncertainty of the motion is not calculated. Still, a brief insight is given below how to incorporate these uncertainties into the Uncertainty Cone Based Matching approach.

The major issue of not taking these uncertainty into account is that due to small noise in the transformation positive pairs can be detected as false ones and hence never reach the matching stage. One simple way of avoiding this is to artificially increase the opening angle of the uncertainty cone. However, calibration and pose estimation uncertainty do not influence the keypoint measurement but the camera pose. Thus, increasing the keypoint measurement uncertainty is not the best solution.

In order to keep the simple structure of the cone based approach, the idea is to find a cone that approximates the uncertainty of keypoint measurement and pose estimation as closely as possible. Previously, the cone was defined by the opening angle resulting from the keypoint measurement uncertainty and the cone tip intersecting with the principal point of the camera. In order to now incorporate the uncertainty of the principal point, the cone can be defined with a shifted tip point. The location of the cone tip and the opening angle must then be chosen that it fits the uncertainty of the principal point and secondly the uncertainty of the keypoint measurement. As the keypoint uncertainty is defined with respect to the camera frame, it is enlarged when transformed to the local reference frame.

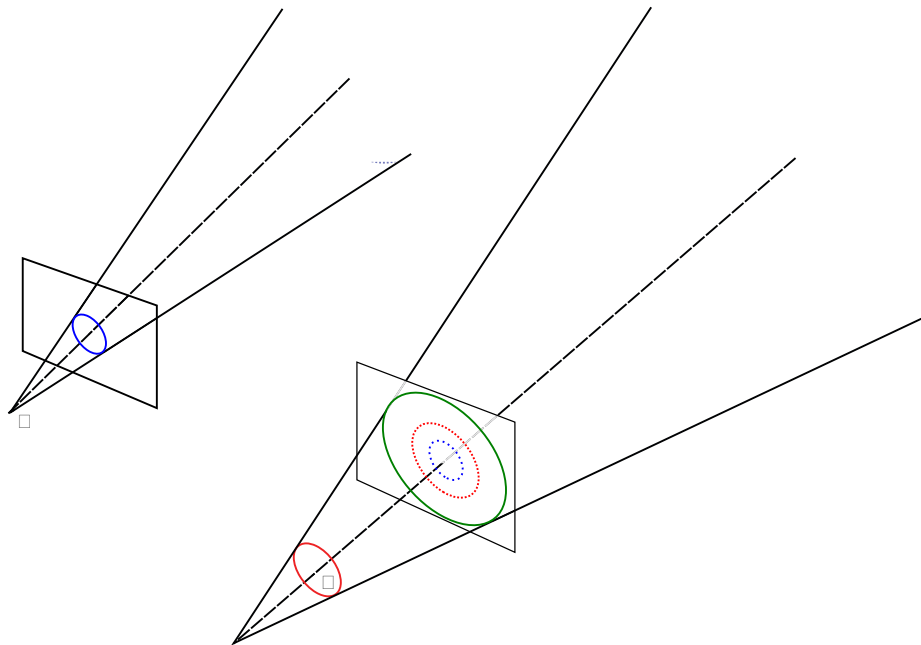


Figure 3.3: Alteration of the Uncertainty Cone to take transformation uncertainty into account

Figure 3.3 illustrates this methods. The left hand side of the illustration shows the regular uncertainty cone with a principal point (red) with no uncertainty and a keypoint measurement with a known uncertainty (blue). The right hand side of the image shows the altered cone with an uncertain principal point resulting from uncertain transformations (calibration and pose estimation). This transformation uncertainty also makes the image plane location uncertain (dashed red). Thus this uncertainty must be added to the keypoint uncertainty (blue) forming the overall uncertainty (green) of the keypoint location. This uncertainty together with the transformation uncertainty define the new cone which has a shifted tip that does not intersect with the principal point (P) anymore.

Relation of Uncertainty Cones to the Generalized Epipolar Constraint

The Uncertainty Cones Based Matching might seem similar to the generalized epipolar constraint described in section 2.3.2. The underlying concept of using the epipolar line to check for correspondences between points is in fact very similar. However, the major difference is that the uncertainty cones, as the name suggests, takes the uncertainty of the keypoint into account. In general, it would also be possible to include the uncertainty into the generalized epipolar constraint. But there are several advantages of the Uncertainty Cone Based Matching. First, by using inverse projective rays it is fully decoupled from any projection model and thus will work for any camera model without altering the algorithm. Hence, it allows to quickly evaluate different camera models. Additionally, the Uncertainty Cone Based Matching is computationally not much more expensive, but triangulation is already included in the calculation.

3.4 Triangulation

3.4.1 State of the Art

After two keypoints from different views have been matched, the 3D position of the underlying landmark can be estimated. This is done by forming a triangle between the keypoints on the image planes and the 3D location of the underlying landmark. As described in section 2.3.1 each keypoint defines an inverse projective ray on which the underlying landmark lies (see also figure 2.3). Together with the baseline between the camera poses, they form the sides of the triangle. However, as shown in the same section, these projective rays will not perfectly intersect in one point due to measurement uncertainty. In order to be still able to triangulate a landmark with a single location in the scene different techniques were developed which are briefly described below.

Midpoint Approach

As P. Beardsley et al. show in [3] landmarks can be triangulated by selecting the midpoint on the line with shortest distance between the two projective rays. From a statistical

point of view, this does not lead to the optimal or most likely position of the landmark as the resulting angular deviation for both keypoints can be different.

Relative Distance Approach

R. Hartley and P. Sturm propose altering this method by not using the midpoint but rather selecting a point on the line of shortest distance according to the ratio of the distances of the landmark to each camera pose ([14]). However, the authors still claim that this is not the optimal solution and thus proposing a new method.

Minimization of the Generalized Epipolar Constraint

In [14] R. Hartley and P. Sturm show that the optimal solution for triangulation is to minimize the generalized epipolar constraint (2.3.2). Amongst iterative solutions to this minimization problem, the authors propose an explicit algorithm to find the optimal landmark location. In [16] Kanatani et al. show an alternative approach called "Optimal Correction" for explicitly finding the landmark's location. However, the underlying mathematical problem remains the same.

3.4.2 Implemented Approach

While minimization of the generalized epipolar constraint promises great results, both proposed methods are rather complex compared to the midpoint approach. This is even more pronounced when comparing it to the Uncertainty Cone Based Matching, where the geometric problem of finding the midpoint is already solved.

Therefore, as a quick starting point the midpoint approach is implemented first. However, the required parameters for the relative distance approach are also already calculated in the Uncertainty Cone Based Matching. Hence, if necessary using the relative distance would be easy.

As landmarks are refined in the optimization step if they are observed again, the triangulated position will be corrected during that step. Thus, the optimal position will be iteratively found making the selected approach similar to a two step approach combining the techniques described above.

3.5 Observation Model and Descriptor Propagation

In order to propagate matches across images an observation model is used. Each landmark stores its observations which are added once the landmark is associated with a keypoint (and not detected as an outlier). In that way an implicit track of a landmark is stored.

This also enabled descriptor propagation. When projecting a landmark onto an image, the descriptor from the very last observation is used to calculate the descriptor distance

to newly detected keypoints. In this way, landmarks can be tracked across large areas of the image where the descriptor changes significantly. The only constraint is that the descriptor does not change significantly between single frames.

Another benefit of the observation based landmarks is that it does not require any special treatment for the different landmark source (structure from motion and overlapping field of view) making it a universally usable.

The biggest benefit of the Kalman filter is that it is a proven principle and that it fuses multiple sensor data in a statistically optimal way¹. Additionally, it is a model based approach and thus can be very accurate. The downside of the Kalman filter is that it can become computationally complex, especially with increasing state vector. Additionally, depending on the problem, it can be very difficult to set up correctly.

Another common filtering approach that can also handle non Gaussian noise is particle filtering. It was first introduced to mapping by K. Murphy in 1999 [23]. A famous implementation of it is FastSLAM by M. Montemerlo et al. [22] where the authors also show a potential speed gain over EKF based approaches.

Optimization

An alternative to filtering is to an iterative optimization. The key idea behind optimization is to find a motion hypothesis that minimizes a designed cost function. This does not require a motion model but in order to show great results, a high number of measurements needs to be present. Additionally, as often gradient based approaches are used, a first estimate is required. The benefit of optimization is that these minimization problems are well numerically solvable also for larger problems. Normally, in navigation the cost function is mainly composed of the reprojection errors of observations.

4.1.2 Outlier detection and removal

To distinguish between outliers and inliers many implementations use a motion model and check for compliance of each match to this motion model. Most motion models result from some sort of system modelling or on board motion estimate like wheel odometry. But also more simplistic approaches exists where the motion to the current time step is approximated by the motion from the previous time step (constant velocity approach).

RANSAC

One common approach if no motion hypotheses are present is to generate them from a subset of the matches and then check for compliance with the remainder of the full set. The more features support a motion hypothesis the higher its score will be. This scoring algorithm makes it robust to outliers which is the major benefit of e.g. a maximum likelihood estimate or a least squares fit. This approach is called RANSAC (RANdom SAmple Consensus) and was first introduced in 1981 by M. Fischler and R. Bolles [8].

M-Estimator

Another approach is using an iteratively reweighted least squares optimization (or other MLE). In general, minimization problems are prone to become unstable in presence of

¹This is only the case for linear systems.

outliers. Especially in least squares solution this can be problematic. In a least squares problem each outlier has a quadratic influence on the total cost and thus "pulling" the estimation away from the optimal solution. One approach to circumvent these issues and make the problem robust to outliers are M-Estimators. These M-Estimators are dynamic, error dependent weighting factors for each error term (or measurement) in the problem. These weighting functions decrease with increasing error and often the decrease is faster than the squared error increase. Thus, the influence of outliers decreases and they might even be "pushed away" if they cannot be optimized to fit the common hypothesis.

Over the years, many different M-Estimators (or different weighting functions) have been developed. The most widely known M-Estimator is the Huber Estimator. Its weighting function behaves like the regular least-squares weight (i.e. is equal to 1, see figure 4.2a) up to a tuning parameter k . It then decreases linearly with the (non-squared) error (see figure 4.2b). The weighting function is given by

$$w_H(z) = \begin{cases} 1 & |z| \leq k \\ \frac{k}{|z|} & |z| > k \end{cases} \quad (4.1)$$

Thus, the Huber Estimator is not very "aggressive" and error terms larger than k are invariant to small changes.

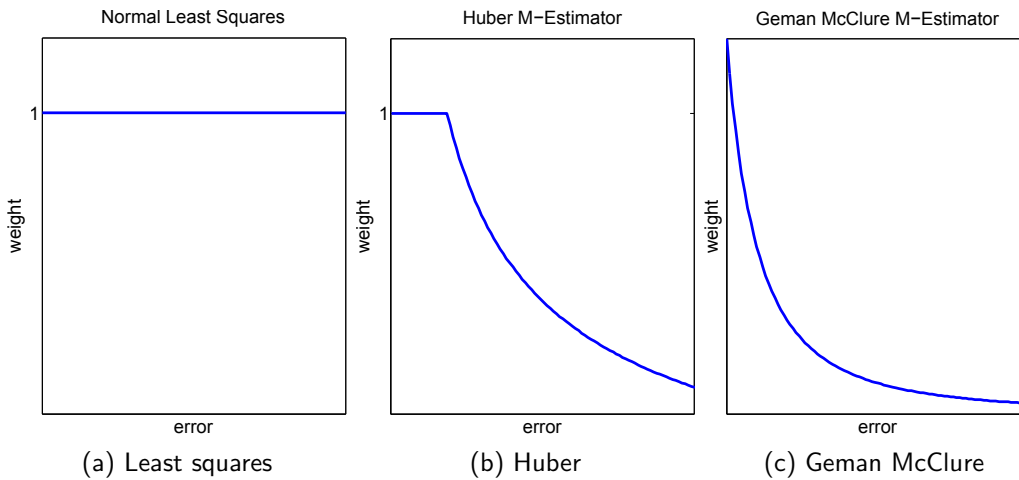


Figure 4.2: Comparison of different weighting factors for estimation problems

Another M-Estimator is the Geman McClure M-Estimator. Its weighting function, shown in figure 4.2c, shows a steeper decline than the Huber M-Estimator and thus is more "strict". One downside of Geman McClure is that the weight for zero error is depending on the configuration parameter and thus often not equal to one. In general, this does not influence the solution. However, when building an optimization problem where also other error terms with different weighting functions are included, than the relative influence can change when reconfiguring the Geman McClure estimator. The weighting function

is defined as

$$w_G(z) = \frac{\sigma}{(\sigma + x)^2} \quad (4.2)$$

For more details on the presented M-Estimators or robust estimation, there is a great discussion of these topics ([30]) written by Z. Zhang.

4.1.3 Landmark Refinement

In online generated maps landmarks created by vision based systems are normally triangulated and hence either associated to one (stereo / overlapping field of view) or two (structure from motion) poses. However, such poses are prone to drift and so are the landmarks attached to the poses. Additionally, the accuracy is prone to noise and has limited accuracy which is also highly dependent on the distance from the camera(s) and the baseline between the poses or camera locations.

Bundle Adjustment

However, as landmarks are often observed over several poses after triangulation, the landmarks' positions can be refined to match these observations accurately. This is done by minimizing the reprojection errors of each landmark. At the same time, the poses associated with the observations can be refined. As landmarks normally represent static objects in a given scene, the drift of the poses can be reduced or even eliminated. As often a larger set of poses and landmarks are refined together, this approach is called bundle adjustment.

As reprojection errors are represented as squared errors, bundle adjustment is a non-linear least squares problem. There are many solvers for least squares problems but most are based on Levenberg-Marquardt and try to exploit the properties (e.g. sparseness) of the problem to provide a computational fast and quickly converging solution.

In general, bundle adjustment is characterized on its scope (often called "window"). Most offline SLAM systems use global bundle adjustment which uses all available poses and observations. This promises the best results but the optimization problem can be huge. Thus, especially online SLAM systems which are constraint in computational power use sliding window bundle adjustment. As the name suggests, there is a limited window size which only contains a few poses and landmarks. This window then moves over the map or recent poses and runs bundle adjustment on the poses included at the current window position. As subsequent window positions overlap on a few frames the sliding window bundle adjustment can still lead to great results.

A thorough introduction and description of bundle adjustment as well as the underlying mathematical problem can be found in [27].

4.2 Back-End Approach

The online localization should be able to run on the vehicle at real time. This highly constraints the complexity of the approach. Additionally, the required accuracy is very high. Thus the selected strategy is to combine all three steps (pose estimation, outlier rejection and landmark refinement) in one single step. As the short overview in the previous section shows, pose estimation and landmark refinement can be achieved by solving an optimization problem. Furthermore, section 4.1.2 shows that estimation problems can be robust to outliers when using M-Estimators. Therefore, a separate outlier removal step is not necessary.

Summarizing, an M-Estimator based optimization problem will be used to cover the back-end. This optimization problem is similar to a robust bundle adjustment. In order to keep complexity at a reasonable level, the optimization will only include two subsequent poses. As there will be a separate batch estimating using a larger window size and fusing additional sensors (see [7]), the achievable accuracy of the selected approach should be sufficient.

4.2.1 Problem Formulation

An optimization problem consists of two types of elements: Design variables which are the variables that will be estimated in the problem and error terms that define the cost for each state of the variables. Each error term is attached to at least one design variable and vice versa. Of course, design variables can be attached to many error terms and the other way around as well.

Design Variables

In the presented project the following design variables are present

Poses (current and previous) Pose design variables actually consist of two design variables: a translation vector and a rotation quaternion which together form a transformation matrix. Poses are defined with respect to the local reference frame. The current pose will be estimated while the previous pose is considered fixed

Camera extrinsics The extrinsics are also represented by transformation matrices and thus consist of rotation and translation. However, the extrinsics are hard to estimate over a very small window and thus are set to be fixed. However, they still appear in the problem as they are required for linking landmark positions to reprojection errors (see below).

Landmark positions Landmarks are simple homogeneous points which are estimated. In order for a landmark to be correctly constraint, there must be at least two measurements (two reprojection errors) per landmark. Otherwise the landmark's

position would be under-constraint and could drift away. Also, estimating and therefore refining the landmarks' positions is not an absolute necessity. Refinement can improve tracking, however also the complexity of the optimization problem drastically increases.

Error Terms

Corresponding to the design variables, the following error terms are included in the problem

Pose prior Optimization problems work great if there is a high number of measurements (here: landmark observations). However, if the number of measurements is low, the problem might become unstable. In vision applications a minimum number of tracked landmarks cannot be guaranteed. Also if the outlier ratio is too high, the problem might as well become unstable. Therefore, the optimization problem has to be made robust against both effects

As a counter measure a pose prior is added. This pose prior is an error term based on the deviation, both rotational and translational, of the estimated pose with respect to a prior. In the presented case, wheel odometry is chosen as a pose prior.

The pose prior uses a constant weight (in contrast to M-Estimators) as it will never be a guaranteed inlier (up to a certain accuracy). A benefit of that is also that the relative weight of the pose prior decreases with an increasing number of reprojection errors. Each error term adds to the total costs and thus the total costs increase with every error term while the error of the pose prior stays the same. In that way, when more measurements are available and the optimization result becomes more reliable, then the solution is under less influence of the pose prior.

Reprojection errors Each reprojection is tied to all three types of design variables as reprojection is dependent on the current position of the landmark as well as orientation of the camera (extrinsics) and the pose of the vehicle. Furthermore, to be able to compute the reprojection, it is linked to the intrinsic calibration of the camera as well as the keypoint it has been observed from.

The error itself is computed by first projecting the landmark into the current camera frame, reprojecting the landmark and finally calculating the distance between reprojection and keypoint. As a distance function the Mahalanobis distance is used. The Mahalanobis distance takes into account the uncertainty which makes it scale-invariant. The Mahalanobis distance is defined as

$$d(x, y) = \sqrt{(x - y)^T S^{-1} (x - y)} \quad (4.3)$$

where x and y represent projected point and keypoint and S^{-1} represents the inverse of the covariance matrix of the keypoint.

4.2.2 Problem Definition

The goal of the optimization problem estimating the pose is to reduce the overall error function. This function consists of the reprojection errors as well as a pose prior. Both are scalar, squared error terms. By using a good initial estimate (i.e. wheel odometry) the problem can be solved using an iterative, gradient based approach. The key idea is to locally approximate the non-linear optimization problem.

As shown in [27] a second order Taylor series can be used for this approximation

$$f(x + \delta x) \approx f(x) + g^T \delta x + \frac{1}{2} \delta x^T H \delta x \quad (4.4)$$

where g is the gradient vector

$$g \equiv \frac{df}{dx}(x) \quad (4.5)$$

and H is the Hessian matrix

$$H \equiv \frac{d^2 f}{dx^2}(x) \quad (4.6)$$

In order to find the minimum, the standard algebraic condition for a minimum (derivative must be equal to zero) is applied to the second order approximation

$$\delta x = -H^{-1}g \quad (4.7)$$

Equation 4.7 is also called the Newton step. The issue here is that solving the Newton step equation can be computationally very complex, especially with a large number of states. Thus exploiting the sparseness (see section 4.2.3) of the problem can speed up the implementation.

According to Triggs et al. the Newton step is prone to failure in two cases. First, it might run the problem into areas of saddle points where optimization will come to an end. Secondly, the second order model might not represent a good approximation of the underlying problem and thus some Newton steps can even lead to higher costs. Therefore, the step size must be controlled. For this case the Damped Newton regularization method is applied

$$(H + \lambda W)\delta x = -g \quad (4.8)$$

where λ is a dynamic weighting factor and W a weight matrix to decrease the step size. For the choice of λ several approaches exist. Most widely known is Levenberg-Marquardt.

The presented derivation is a very general description of a gradient based optimization. In [9], P. Furgale derives the gradient based optimization problem for scalar, squared error terms. Furthermore, the author shows how measurement noise can be included in the problem and how to solve it efficiently. As P. Furgale's work nicely illustrates these optimization problems for computer vision applications, please refer to it for a more details.

4.2.3 Properties of the problem

Sparseness

With an increasing number of design variables and error terms, the optimization problem becomes more complex to solve. This is partly due to the increasing size of the Hessian matrix. However, as the relations between error terms and design variables is sparse, the Hessian matrix will also be sparse. Thus, by using a sparse matrix representation and solver, the speed of optimization can be significantly increased.

Schur Complement Trick

One way of benefit from the sparseness of the problem is to apply the Schur Complement. As B. Triggs et al. show in [27] a linear system of equations

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \quad (4.9)$$

can be effectively reduced by right multiplying the system matrix by

$$L = \begin{pmatrix} 1 & 0 \\ -CA^{-1} & 1 \end{pmatrix} \quad (4.10)$$

which finally leads to a reduced system

$$\bar{D}x_2 = \bar{b}_2 \quad (4.11)$$

with

$$\bar{D} \equiv D - CA^{-1}B \quad (4.12)$$

and

$$\bar{b}_2 \equiv b_2 - CA^{-1}b_1 \quad (4.13)$$

As B. Triggs et al. show this can be efficiently applied to the bundle adjustment problem when ordering the Hessian matrix into structure and camera blocks. The Schur comple-

ment promises a great performance improvement especially when structural parameters largely outnumber camera parameters which is the case in the presented two window bundle adjustment.

4.3 Optimization Frameworks

As these optimization problems are very common in robotics and navigation, a few optimization frameworks exist. These frameworks offer an interface to create and finally solve the problem. While features slightly differ, most are able to exploit the sparseness of the problem and optimize the solving process.

4.3.1 Available Optimization Frameworks

For quite a while the g2o framework was the de facto standard optimization framework until Google released a new optimization framework called Ceres in May 2012. Early 2012 an optimizer for the newly designed ASLam framework was developed at the ASL. All these frameworks support gradient based solving based on Levenberg-Marquardt and use the SuiteSparse library for handling sparse matrices.

g2o

g2o has been developed by R. Kümmerle et al. at the University of Freiburg. The target of g2o are SLAM applications but it can also be used for other optimization problems. While many researches successfully use g2o, the interface and lack of extensive documentation makes it a little difficult to use and the framework is maintained by the researches and a small community only.

Google Ceres

Google also uses its Ceres optimization framework for 3D reconstruction of buildings for Google Maps and thus has to handle a quite high amount of data. Additionally, the software is written by professional computer scientists. For these two reasons, the optimization framework promises great speed and stability. Furthermore, it provides an automatic differentiation algorithm for computing the jacobian matrices (which are used to build the Hessian matrix) from the problem formulation.

ASLam Optimizer

The ASLam Optimizer (and Optimizer 2) was designed to provide an easy to use interface to solve SLAM related optimization problems. However, also speed was not neglected and solving is reasonable fast. While it still lacks a large user base it is sufficiently stable and covers all required features.

4.3.2 Optimization Framework Choice

As many great and well proven solution exist, using an available optimization framework is much simpler than designing an optimizer from scratch. While Google Ceres probably promises the best performance, it was not ready during the beginning of the problem and the difficult usability of the g2o framework did not make it a very attractive candidate. In fact, this was one of the reasons that led to the development of the ASLam Optimizer. While it is not widely used, the ASLam Optimizer promised a quick start towards an early prototype implementation. The biggest benefits of the ASLam Optimizer for this particular project are the in-house development, the easy to use interface as well as the seamless integration into the ASLam framework. Thus the ASLam Optimizer is used in this project

5 Implementation

This chapter will give a short overview of the implementation. It is more hands-on than the other chapters and is partly written similar to a manual. Readers solely interested in the scientific part of this theses can skip this chapter without hesitation.

Continuing work that someone else has started is always a difficult task. And especially because this project is the basis for the remaining part of the V-Charge project, the chapter still has a right to exist and will hopefully facilitate further development.

5.1 Build System and Software Environment

5.1.1 Build System

CMake

The build system of the localization pipeline is based on CMake. Apart from platform independent build systems, CMake also allows to easily integrate existing libraries into a project. Furthermore, it is possible to use CMake as a configuration tool for non-runtime parameters.

Due to the lack of a package manager under Windows, most software libraries have to be compiled manually. For easy installation, these software packages are thus integrated into the build system. This includes SuiteSparse, ASLam, Schweizer Messer and SQLite. The only libraries that are not included are Boost and OpenCV. Both require a long time to compile and therefore are better build separately.

Microsoft Visual Studio

As a development environment Microsoft Visual Studio is used. Visual Studio does not only provide the compiler and linker but also debugging and profiling tools. The Microsoft Visual Studio project is configured by CMake and all major project settings are set in the CMake configuration.

5.1.2 ADTF Middleware

ADTF is a middleware that is developed by a subsidiary company of VW. It is a proprietary software and runs under Windows and Linux. As the V-Charge PC uses Windows, the Windows version of ADTF will be used in the project. Figure 5.1 shows the graphical user interface of ADTF

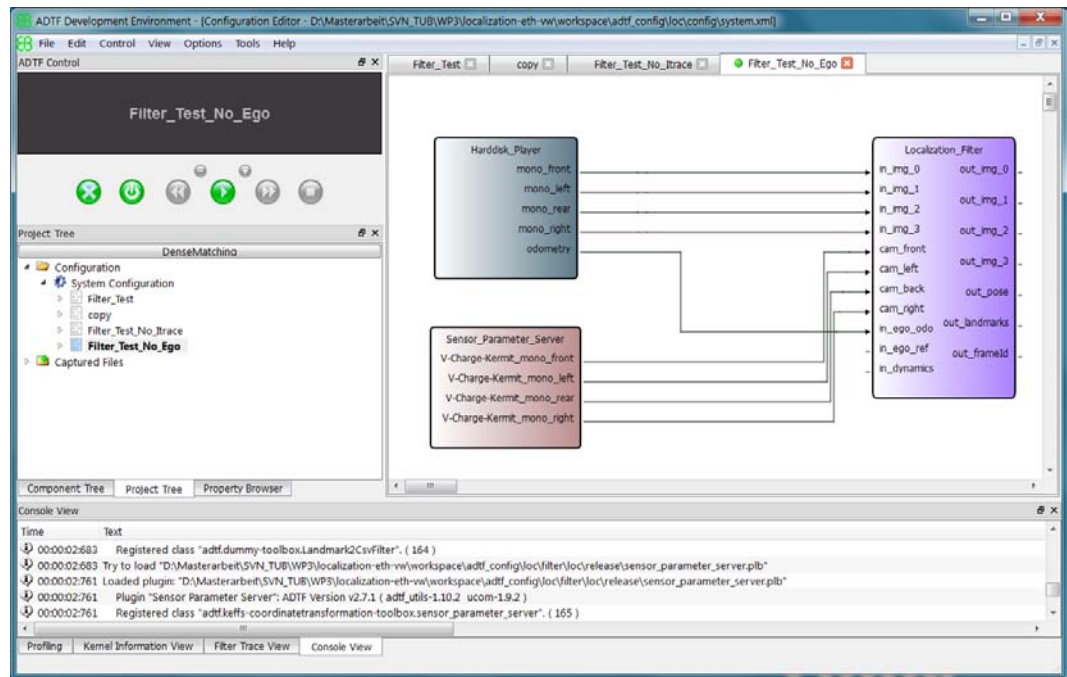


Figure 5.1: Graphical user interface of ADTF

General Layout

ADTF features a configuration editor, which is a whiteboard where modules (filter blocks) can be dynamically attached to each other and therefore, different configurations do not require recompiling. Furthermore, ADTF features a time scroll bar with media player like buttons (e.g. start, stop and play).

Filter Concept

ADTF organizes the different modules running on the system in so called filters (colored boxes in figure 5.1). These filters are C++ libraries that get called by ADTF. Each filter can have static (hardcoded) or dynamic (created at runtime) in- and output pins. Through these pins, the different modules can communicate. ADTF handles the transportation of the messages send through these pins and calls the pin event function of the receiving modules. In contrast to other middleware (e.g. Robot Operating System) all modules run in a single thread. This means that if any module crashes, all other modules and also ADTF will stop working.

Data Files

The V-Charge recordings are stored in a binary format (.dat files).

ADTF supplies some basic tools for playing, cutting and exporting data from these files. The "Harddisk Player" filter allows to play a recorded dataset and thus mimics

the behaviour of the filter configuration during recording. Matched with the "Harddisk Recorder" filter it is possible to cut segments from the data files. There are also some additional tools in the "Tools" menu of ADTF. Most useful is probably the "DAT Exporter" that allows to export the image streams as individual image files.

Apart from the ADTF tools there is also an ADTF data analysis and export tool written by the ASL. The tool runs in python and in contrast to the ADTF DAT Exporter, it allows to access all data streams included in the recorded file.

5.2 Software Structure

5.2.1 Software Elements

There are three different types of software elements in the project

Modules Modules are external libraries as well as libraries developed in this project. Each module provides a set of general classes and routines. However, these modules are not standalone executables but are included in ADTF filters and executables.

Filters Filters are libraries build with the ADTF Software Development Kit and thus provide interfaces to the ADTF middleware. While filters are still libraries that get called by ADTF, they have a similar structure to executables. Throughout the project, the content of the filters is minimized to write as little non-portable, ADTF specific software as possible. Thus, filters are mainly an interface between ADTF and the modules. They read out ADTF messages and call the appropriate functions.

Executables Executables are the counterpart to ADTF filters. They are standalone executables, which also mainly contain modules. Executables are mostly small tools (e.g. the epipolar line checking tool) that use a few selected modules. Also, they are often used for small tests or debugging purposes. In general, it would also be possible to create an executable that mirrors the entire pipeline.

The main elements from these three categories are explained in detail later in this section.

5.2.2 Structure of the Localization Pipeline

Figure 5.2 shows an overview of the localization pipeline. The core pipeline and its modules are embedded in a single ADTF filter (the Localization Filter). Some general modules like the map, visualization and the parameters live inside the localization filter but outside of the core pipeline. The other modules only accessed from the core pipeline live within the core pipeline.

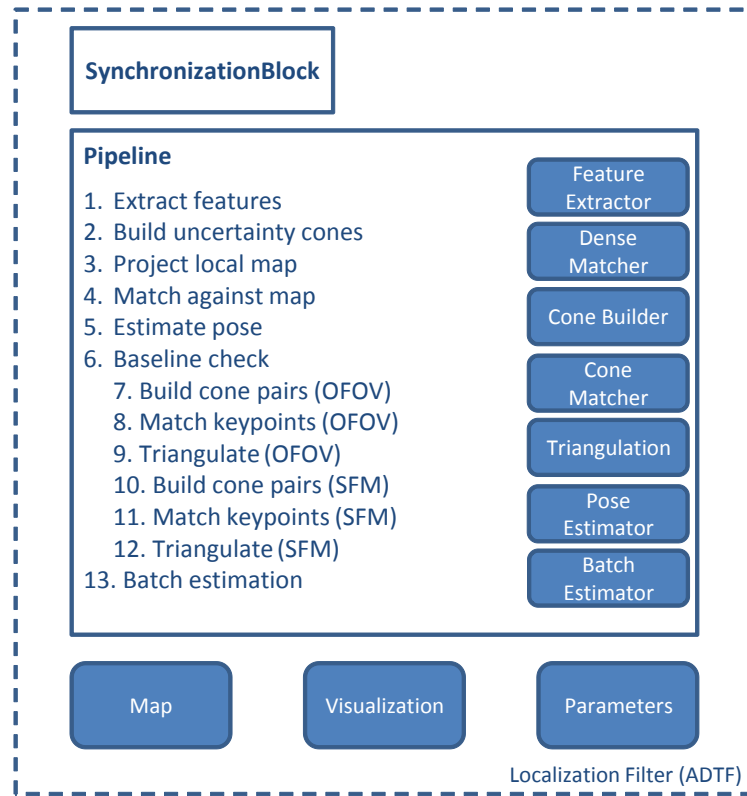


Figure 5.2: Structure of the Localization Pipeline

The process of localization is (apart from signal processing steps) a thirteen step process, briefly shown in figure 5.2.

The indent shows optional steps. As a key frame approach is used, new structures (step #7 to step #12) are only generated when a certain baseline is passed. Pose and batch estimation however are always executed to deliver the most up-to-date pose estimation. The modules shown in this diagram are explained in more detail in the next section.

5.2.3 Modules

ASLam

The ASLam module is one of the external libraries. It provides basic structures (key-points, landmarks, cameras etc.) for a SLAM system. Furthermore, it also contains several implementations of vision and robotic algorithms including feature extractors, matchers and optimization frameworks.

However, there is also a V-Charge specific part within ASLam. The ASLam V-Charge front-end wraps different feature extractors including BRISK, OpenCV SURF (CPU), OpenCV SURF (GPU) and ASRL SURF (GPU). Due to the Uncertainty Cone Based Matching strategy, special descriptors are necessary (VchargeSurfDescriptor and VchargeBriskDescriptor). These descriptors are derived from the regular ASLam descrip-

tors but additionally contain a inverse projection direction, a half cone angle and the precomputed tangent of the half cone angle. Moreover, there is the `VehicleFrame` class which ties together individual frames from the four cameras (front, left, rear and right). Last but not least, there is the `VehicleCameraSystem` class. This class can handle n cameras with selective overlapping field of view. It can calculate the overlapping field of view and also creates undistorted versions of the cameras. Additionally, it stores the extrinsics of the cameras as well as the baseline between them.

Common

The common module is very small. It contains the V-Charge type definitions as well as a simple singleton logging class.

Estimator

The estimator module contains both estimators used in the localization pipeline: The batch estimator as well as the pose estimator. Both estimators are based on the ASLam optimizer and are therefore wrappers that read the V-Charge structures and fill in the ASLam optimizer structure.

Features

The features module, as the name suggests, handles features after they have been extracted. The two main elements are the Uncertainty Cone Based Matcher as well as the triangulation.

The `CorrespondanceCandidates` submodule contains data structures for storing the result of the uncertainty cone pair building (prior to matching). A `CorrespondanceCandidates` object maps n frames to n frames and thus works for overlapping field of view and structure from motion. Each of these objects contain n `CandidateSets` which store the cone combinations of m times p `KeypointPairs`. Each `KeypointPair` holds the two line parameters for the points on the inverse projective rays with closest distance as well as the distance between them. Additionally, two flags store whether the `KeypointPair` is valid and whether the underlying landmark is at infinity.

The `KeypointCone` submodule is responsible for calculating the Uncertainty Cones by inverse projecting the keypoint and calculating the cone half angles. As the Uncertainty Cone Based Matching does not yet take transformation uncertainty into account, the half cone angle is currently fixed to 1.5° . The `KeypointCone` submodule also fills the `KeypointPairs` (uncertainty cone pairs) described above.

For matching keypoint cones, the `Dense3DMatcher` is developed. It is based on the ASLam `DenseMatcher` and works the exact same way. However, instead of the distance check, it reads out the validity flag of the `KeypointPair` prior calculating the descriptor

distance. This implementation is not very efficient, as it still cycles through all keypoint combinations. Instead, it would be simpler to only keep valid cone pairs (and replace the `CorrespondanceCandidates`). The reason for the current implementation is that it is good for debugging and testing purposes.

Finally, there is the triangulation submodule which simply performs a midpoint triangulation and fills in the localization pipeline data structures.

Map

The map module is provided by VW and contains data structures for storing vehicle frames, poses and landmarks. An underlying database module allows map files to be written to harddisk.

Pipeline

The pipeline module is the core of the localization pipeline. It contains the `SynchronizationBlock` which collects four frames, one from each camera, and calculates the corresponding synchronized wheel odometry by calculating the relative wheel odometry transformations and extrapolating between these. Once such a combination of vehicle frame and pose is filled, it is passed onto the pipeline. The pipeline itself defines the main structure of the localization pipeline and defines the sequence in which the individual parts are called. Within the pipeline, the timings are recorded.

Schweizer Messer

The Schweizer Messer is a robotics library developed by P. Furgale. It defines basic mathematical tools and structures like homogeneous points, transformations and a quaternion algebra library. Furthermore, it enhanced Boost Property Trees which are used for propagating configurations through the pipeline. Last but not least it provides a timing class to assess the performance of function calls and operations.

The Schweizer Messer is a growing library and there are additional tools available not used in this project.

Visualization

The visualization module provides simple tools for visualizing different parts of the pipeline. It can visualize keypoints, matches and feature tracks. Additionally, it provides a few functions to export the visualization output as individual images or a video stream.

Others (external)

As explained above, many modules are external libraries included into the build system. The following external libraries are included as modules

Eigen The Eigen library is a linear algebra toolbox and is used in this project for handling matrices and vectors.

ASRL SURF This SURF implementation has been developed at the Autonomous Space Robotics Lab at the University of Toronto. Compared to OpenCV's SURF implementation it offers several enhancements like gridded extraction or delivering a fixed number of keypoints. However, it is still in an experimental stage.

gTest Google's unit testing library is used to develop unit tests verifying the functionality of the developed modules.

KD-Tree The KD-Tree is the underlying storage class used by the map to store vehicle frames, landmarks and poses.

SQLite The SQLite library is a file based SQL implementation which is used by the map to write the map data to file to free RAM and store the map.

SuiteSparse The Suitesparse library is a linear algebra library specialized for sparse matrices and solving equations with sparse matrices. SuiteSparse does not provide a build system that runs under Windows. Hence, a CMake based custom build is created.

5.2.4 Filters

Localization Filter

The localization filter is the main ADTF filter that wraps the entire pipeline. As described in section 5.1.2 the goal is to reduce the amount of ADTF specific code. Thus the localization filter simply receives the camera images and wheel odometry measurements and passes them on to the synchronization block of the pipeline.

Checkerboard Finder

The checkerboard finder filter runs the OpenCV checkerboard corner finder on a provided datasets and writes out all images where the OpenCV checkerboard finder succeeds to find the checkerboard corners. This is a useful tool for sorting out bad images when calibrating cameras.

Others (external)

There are several small logging filter that write different data streams (e.g. wheel odometry) to file for debugging or visualization with external programs (e.g. Matlab).

5.2.5 Executables

ShowEpipolarConstraint

The epipolar line drawing tool allows the user to click a point in one image. If this point lies in the overlapping field with any other camera, the epipolar line is drawn in the other image. Thus, it can be used for verification of the calibration. There are two versions of this tool, one which uses undistorted images (as if they have been taken with a perfect spherical camera) and another which works with fully distorted images. With both it is possible to verify the intrinsic and extrinsic calibration of the cameras.

LocalizationExecutable

The localization executable was used in the beginning of the project, mainly for debugging purposes. It is supposed to mimic the behaviour of the localization filter. As it is not up to date with latest changes in the pipeline, it is not ready to be used.

5.3 Configuration

The localization pipeline is a complex combination of different modules. Each module has different properties that can be configured. But also the interplay of modules can be influenced or is automatically influenced by these parameters. Additionally, for some pipeline steps several modules are available which the user can chose from. Thus, there is a large amount of configuration parameters and settings.

Wherever possible, the parameters and options are included in a configuration file that is read at runtime. Thus, parameters included in this configuration file can be changed without rebuilding the software. Even though not used yet, the pipeline is prepared to allow parameter changes during execution.

However, especially with interchangeable modules, some settings must be chosen prior building the localization pipeline. These settings are included in the CMake configuration and can be set e.g. using the CMake GUI.

There is one exception where parameters are hardcoded in the source: the camera calibration. The idea is to also include those in the configuration file. However, the SensorParameterServer, an ADTF filter used by VW, requires the calibration data before the configuration file is loaded. As the SensorParameterServer is closed source an integration of the calibration data into the configuration file is not possible¹ until VW submits a patch for its filter.

Please find below a descriptions of the available options.

¹At least it would not be a lasting solution.

5.3.1 Pre-build Configuration

While there are many variables in CMake, the following ones present options of the localization pipeline:

ADTF_ENABLED Enables/disables building of ADTF filters. This can increase build speed and allows building on machines without ADTF.

BUILD_CPP Enables/disables building of the executables.

BRISK_ENABLED Enables/disables BRISK as a feature extractor. This flag overrides all SURF flags

GPU_ENABLED Enables/disables OpenCV's GPU version of SURF. This requires an NVidia GPU and Cuda to be installed. If not set, OpenCV's CPU implementation is used.

GPU_SURF_ASRL Enables/disables the ASRL SURF extractor. This flag overrides the GPU_ENABLED flag.

LOG_LEVEL The log level can be set to any value between 0 and 3 and defines how much data is logged while running the pipeline. While 3 is a good setting for debugging, 1 is recommended for regular execution. 0 almost suppresses logging entirely.

TIMING_ENABLED Enables/Disables timing of the different pipeline steps. If this option is set, timings will be recorded, independent of what the LOG_LEVEL option is set to.

5.3.2 Configuration File

The configuration file is named "vcharge.info" and is read at runtime by the Boost Property Tree which is used to pass the parameters through the pipeline. All available parameters are described in the configuration file. Worth mentioning here is that different feature extractors require different parameters. In fact, the values of those are very different such that running the pipeline with bad parameters will lead to no landmarks or pose estimates.

5.4 Running, Debugging, Logging and Analyzing the Localization Pipeline

5.4.1 Running the Localization Pipeline

Prior running the pipeline, it must be configured properly using the pre-build and post-build configuration (see section 5.3 for more details). Even though some of the configuration parameters appear in the ADTF GUI, they can only be safely changed in the

configuration file. Previous logs located written to the "files" directory in the ADTF working directory should be moved or deleted to avoid mixed logs.

Due to the out of source build, the pipeline can be run by simply starting ADTF using the batch file located in the ADTF working directory. There are two versions, "start.bat" for release builds and "start_debug.bat" for debug builds. As VW does not always provide debug build binaries for their proprietary filters, some modules might only be available in release mode.

5.4.2 Debugging the Localization Pipeline

In both build modes (debug and release) debug symbols are generated during the build process. However, due to optimizations not all features are available during debugging in release mode. Additionally, some asserts are only active in debugging mode. Often, these asserts can help finding errors more easily.

Before debugging, multithreading should be disabled in the configuration file. Otherwise, not all variables and execution paths can be accessed. Afterwards, ADTF can be launched normally (preferably in debug mode) and then the Visual Studio debugger can be attached to the process.

Additionally, the logging level should be increased as it can provide further insight into the processes of the pipeline.

5.4.3 Logging and Analyzing the Localization Pipeline

The simplest way to log major parts of the pipeline is to increase the log level. However, the logging class provides macros for logging additional variables. These macros can be used anywhere across the pipeline.

For analysing the logged data several Matlab tools exists. These tools were also used to generate the plots contained in the presented thesis.

There is also a Matlab tool for visualizing timings. However, when speed optimizing the pipeline the Visual Studio profiler should be used. This profiler is able to show the "hot paths" (functions/operations that consume the highest amount of time) as well as execution time of individual operations. The profiler can be used similar to the debugger by attaching it to a running ADTF process.

6 Results

6.1 Front-End

6.1.1 Comparison of Feature Extractors

Test Scenario

As described in section 3.1.2 the choice of feature detectors is narrowed down to BRISK and SURF. BRISK and its binary descriptor promise fast extraction but potential higher self-similarity compared to SURF. To see how big the difference of these feature extractors are, they are both run over the same dataset of 750 frames. Furthermore, the number of extracted keypoints is tried to be kept constant. Of course, this comparison cannot eliminate difference in the configuration styles. However, settings are chosen to mimic an equal behaviour. The major difference of both extractors is that BRISK provides an even distribution (local maxima) while SURF provides to globally strongest keypoints (global maxima).

Test Results and Discussion

	BRISK	SURF
# inlier matches		
- average	311.4	419
- min	4	52
- max	885	1203
- standard deviation	115.8	197.5
inlier/outlier ratio		
- average	66.3%	83.5%
- min	17.4%	32.8%
- max	99.2%	100%
- standard deviation	14.7%	14.0%
feature track length		
- average	4.1	5.7
- max	46	118
- standard deviation	1.94	4.6
landmarks generated	643'634	277'535

Table 6.1: Comparison of feature extractors (BRISK vs. SURF)

Table 6.1 shows the results from the comparison. As expected, SURF shows better performance. The average number of inliers as well as the inlier to outlier ratio is significantly higher. As a result, the average feature track length is larger and less landmarks are generated.

However, the difference between both extractors is larger than expected. While the performance of BRISK is still good, it does not live up to the expectations prior the experiment. One reason for the performance difference is that BRISK provides an even distribution of keypoints. While in many cases this can be a benefit, it poses a problem in the presented project. Especially the side cameras which are tilted slightly downwards cover big segments of the ground. Hence, BRISK provides also many features on the ground. However, in the absence of road markings, the road has very little texture and is therefore very self-similar. Therefore, mismatches are much more likely to happen. Additionally, BRISK is based on the Harris Corner Detector and therefore is dependent on rectangular corners. However, the fisheye cameras warp/distort corners heavily in the outer image areas. Hence, extraction is less reliable.

Concluding, with the current implementation SURF performs better in terms of feature quality. However, tests with globally strongest BRISK features (or gridded SURF) and undistorted descriptors may show a very different results.

For a timing comparison of both extractors please refer to section 6.3.4.

6.1.2 Evaluation of Matching Strategies

As described in section 3.3.2 the new Uncertainty Cone Based Matching strategy is used in this project. One important test is the comparison to a purely descriptor based (dense) matcher.

Test Scenario

For this test, the same dataset as in section 6.1.1 is used. Furthermore, the feature extractor is SURF. The regular matcher does not have a distance threshold as this would foreclose matching of landmarks close to the camera¹. Both matchers use the exact same implementation, with the alternation for a validity check of a potential match in the Cone Based Matching.

Test Results and Discussion

From the results listed in table 6.2 the increase in robustness of the Uncertainty Cone Based Matcher becomes evident. The total number of inliers increases significantly and also does not decrease as much in difficult cases. The same applies to the inlier to outlier ratio. One criterion where the dense matcher shows better results is the minimal inlier

¹See section 3.3.1 for a detailed explanation of this effect.

	Dense Matcher	Cone Matcher
# inlier matches		
- average	241.9	419
- min	34	52
- max	696	1203
- standard deviation	120.3	197.5
inlier/outlier ratio		
- average	79.7%	83.5%
- min	35.7%	32.8%
- max	100%	100%
- standard deviation	15.6%	14.0%
timing per frame set		
- total	0.032 s	0.165 s
- cone pair calculation	0.000 s	0.138 s
- matching	0.032 s	0.027 s

Table 6.2: Comparison of feature matchers (Dense Matcher vs. Cone Based Matcher)

to outlier ratio. The reason for that is that the Uncertainty Cone Based Matcher in its current implementation expects perfect intrinsics and poses. However, as section 6.3.1 explains especially the pose is noisy. Therefore, if the Uncertainty Cone Based Matching is altered as proposed in section 3.3.2 even better results can be expected.

When analyzing the timings the situation changes. As expected, the Cone Matcher is faster during the matching step. The difference here is rather small. The major reason is that the same matcher is used, i.e. it still cycles to all keypoint combinations. A faster approach would be to add possible combinations in the cone building stage to a list of possible matches and finally only loop through that list. This is especially promising as over 98% of combinations are excluded by the matcher.

But also the cone pair calculation requires a lot of time. While it has been slightly optimized, there is still a lot of room for improvement. Currently, the result of the cone pair calculation is stored for all cone pairs and not only for possible matches. Additionally, the computational expensive distance calculation (solving a 3x3 linear system of equations) is not optimized. Here some performance optimized library (like Intel Performance Primitives) or loop unrolling would probably help a lot. Also, multithreading is not exploited in the cone pair building.

Summarizing, the Uncertainty Cone Based Matcher already shows great results while there is still room for further improvements. If computational power is not super critical, the use of the Cone Based Matcher can improve matching significantly.

6.2 Back-end

6.2.1 Outlier Rejection and Landmark Refinement

As described in sections 4.1.2 and 4.1.3, outlier rejection and landmark refinement are an integral step solving an optimization problem with M-Estimator weighted reprojection errors. As explained in those sections, the M-Estimators should reduce the reprojection errors of inliers while the influence of outliers is reduced.

Test Scenario

In order to verify this, a test dataset of around 100 frames is run through the pipeline. After pose optimization but before outlier rejection, the reprojection errors of all error terms in the problem are recorded and sorted by their unweighted reprojection error.

Test Results and Discussion

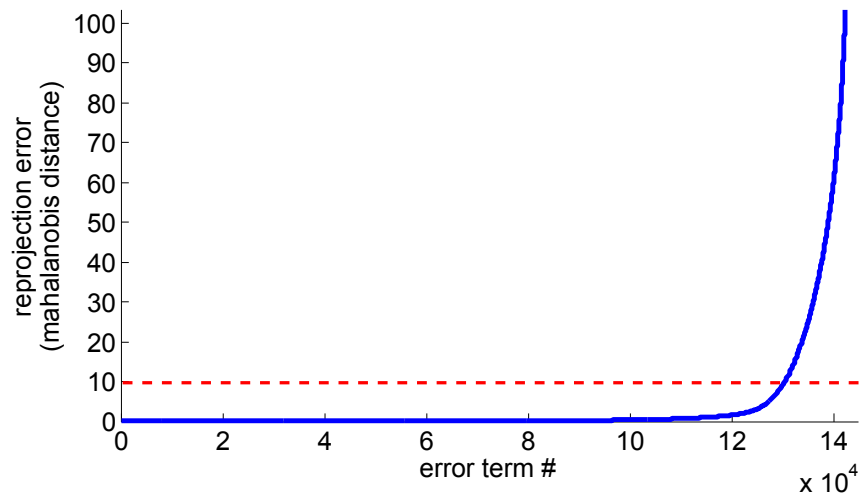


Figure 6.1: Reprojection errors after optimization sorted by error

Figure 6.1 shows the results of the test. For scaling reasons, the plot is cut off at the right hand side and thus it is not showing error terms with a magnitude of above 80 pixels².

The reprojection error plot shows the expected behaviour of the M-Estimators. Most of the reprojection errors are reduced to low value and their mahalanobis distance is well below 5 pixels. Shortly before error term #13'000 a very pronounced increase of the error slope is observable after which the reprojection error drastically increases (up to about $2.0e^{-6}$ pixels). This "knee" (up to some uncertainty) separates inliers on the left hand side and outliers on the ride hand side.

²Totally about 15% of all error terms are not shown.

To have a sharp criterion to distinguish between in- and outliers, a reprojection error threshold is set. Due to imperfect pose estimates and calibration results, some inliers may not be perfectly optimizable. Hence, setting the threshold too low will also exclude inliers. Thus, a reprojection error of around 10 pixels is reasonable.

Landmark positions where all associated error terms are below this threshold are updated. If an error term is above the threshold the associated observation is deleted and the landmark position remains untouched as it was before the optimization. If a landmark has less than two observations, it is completely removed from the map.

6.2.2 Evaluation of Pose Prior

As described in section 4.2.1 a pose prior is added to the optimization problem to make it robust in case of a low number of inliers.

Test Scenario

To verify if the pose prior works as expected, a test data recorded with Kermit is run. As described in section 2.5.2, the calibration is not very good. Together with a bad odometry measurement, the Uncertainty Cone Based Matching will possibly fail in a few cases. This is the point where the pose prior should fill in and prevent the estimated pose from becoming unstable (and thus becoming an outlier).

Test Results and Discussion

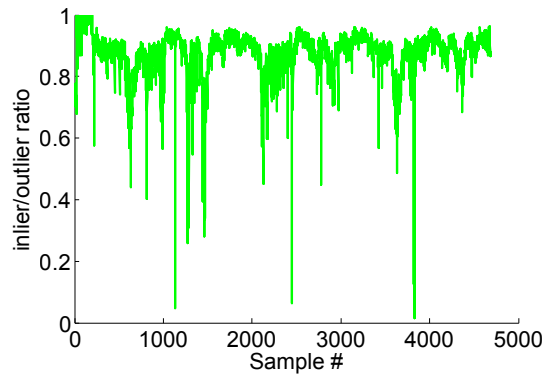


Figure 6.2: Inlier/outlier ratio of landmark observations

Figure 6.2 shows that as expected matching fails in a few cases and the inlier to outlier ratio drops even below 0.1 for a few sample frames. Now, in these cases the optimization problem should be unstable and thus, an outlier in the pose estimate should be visible. However, when looking at the overhead pose in 6.3 no outlier is visible but the estimated poses form a smooth trajectory.

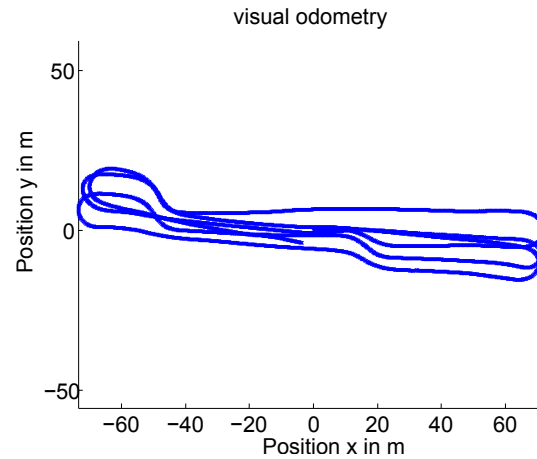


Figure 6.3: Pose estimates generated by the visual odometry

While the results show the benefits of a pose prior, one must be especially careful when setting the weight of the pose prior. If the weight is too low, the pose prior will not have enough influence on the overall optimization problem to keep the pose from drifting away in case of a high outlier ratio. However, when setting the pose prior too high, the influence of landmarks is reduced and thus the visual odometry might be less accurate. Thus, finding a good weight for the pose prior is very difficult, especially since the rotation error and translation error can be weighted separately.

6.3 Visual Odometry

6.3.1 Filtering of Wheel Odometry

Issues of the Wheel Odometry

As explained in section 3.3.2 the cone matcher requires a very good estimate of the motion. But also a better pose prior (see section 4.2.1) will lead to a better pose estimation result. In general, wheel odometry provides a very accurate estimate of the motion. However, there are two issues.

First, wheel odometry is based on wheel ticks, which are quantized. Together with a high sensing rate of 50 Hz this leads to quantized wheel odometry. Also the preprocessing done by VW does not seem to entirely eliminate this effect which can be seen in figure 6.4.

Second, wheel odometry is not synchronized with image frames. While a pure time offset would be less than $t_o \leq 1/50 = 20$ ms and therefore acceptable, another issue occurs. As the image stream frequency (12.5 Hz) is one fourth of the wheel odometry frequency (50 Hz), normally four odometry measurements are received per image measurement. However, due to very slightly varying sensing frequency it is possible that not four but three or even five measurements are received in between. As odometry is passed to the

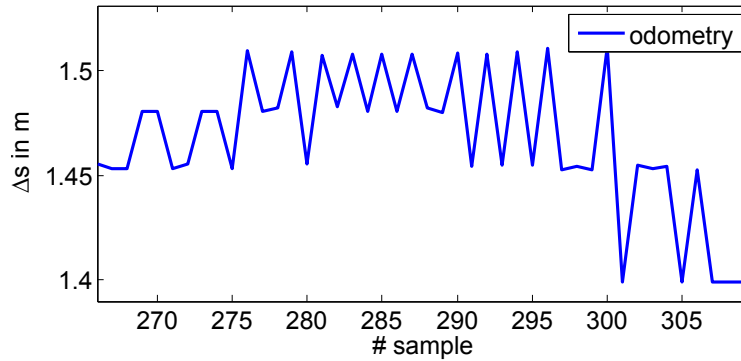


Figure 6.4: Plot of preprocessed wheel odometry visualizing the quantization effect

pipeline as a relative transformation to the last odometry send, this can lead to quite small (three measurements) or large (five measurements) steps in wheel odometry.

Synchronization and Filtering of Wheel Odometry

In order to correct for the issues of wheel odometry on short scales, it is synchronized with the image stream and filtered in an integral step.

To synchronize wheel odometry with the image stream two approaches are possible. First, extrapolate wheel odometry based on previously received measurements up to the time the image is received. Secondly, one could wait with passing the image to the pipeline until the next wheel odometry measurement is received. Then interpolation would be possible. The choice between interpolation and extrapolation might not be critical. However, extrapolation is chosen here as it does not impose additional delay to the pipeline³,

While odometry is of higher sensing rate than the images and odometry is integrated, subsampling to image rate already behaves like a moving average (FIR lowpass) filter with varying window size (depending on odometry samples received between two images). However, it must also be ensured that extrapolation is also based on a filtered signal and not only the last two measurements. Otherwise, the quantization effect would be amplified. Thus, extrapolation is based on the whole set of measurements received between two images. Extrapolation itself is done linearly.

Results and Discussion

While the presented solution is rather a quick solution than a perfectly engineered signal processing approach, it shows good results.

Figure 6.5a shows the transversal correction. While the unfiltered wheel odometry at image time is unevenly distributed, the filtered signal is much smoother. Figure 6.5b

³The delay of interpolation could be compensated for if the waiting time for the next odometry measurement to arrive is used for pipeline steps that do not require a pose (e.g. feature extraction)

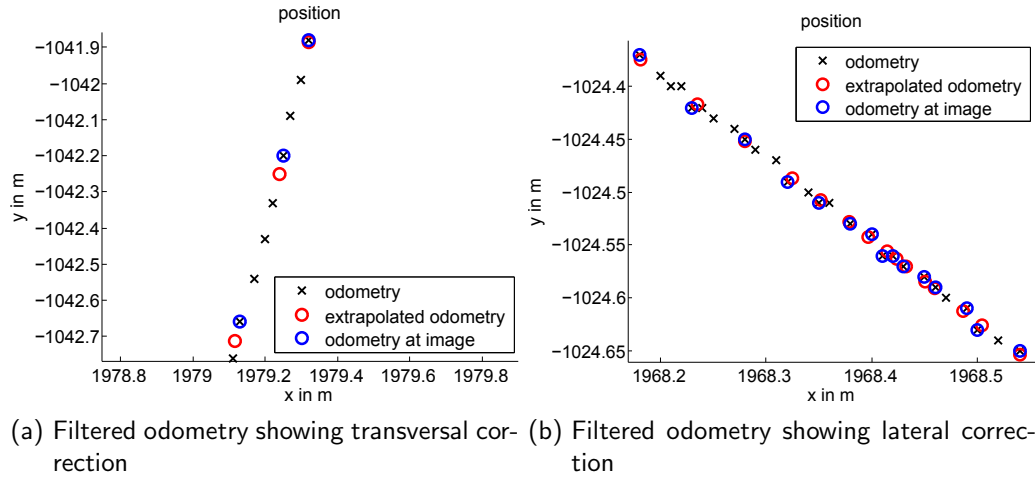


Figure 6.5: Comparison of filtered to unfiltered wheel odometry

shows the second benefit of the filtering strategy. The unfiltered wheel odometry has large deviations from the imagined straight motion. The filtered signal performs better and lies closer to the line of motion.

In the real pipeline the filtered odometry makes a big difference. With unfiltered odometry, there are areas where the feature count heavily drops as the incoming pose does not reflect the real pose properly. With the filtered odometry these feature drops happen very seldom.

6.3.2 Pose Estimation Accuracy

One major requirement for the visual odometry pipeline is that the estimated poses are accurate. While the pose estimate will be refined in the batch estimator, a better initial estimate will help to improve the results.

Test Scenario

To test the accuracy a test sequence of about 7 km is used. This sequence contains different scenarios including drives through urban areas as well as fast drives and standing phases in and around Wolfsburg. This sequence is run through the pipeline in a single run. Afterwards, pose estimates, wheel odometry and GPS are cut into segments with fixed length. First 200 m long segments are used, then the segment length is increased to 400 m. After cutting, a linear interpolation is run on the data to obtain synchronous data sets.

The first pose of each segment is then aligned with the origin to eliminate drift from previous segments. Afterwards, the first 10% of each segment are used to align the rotation of wheel and visual odometry to the GPS data. This is done using a nonlinear

minimization that minimizes translational errors between wheel odometry and GPS or visual odometry and GPS respectively.

Finally, the magnitude of the deviation from the ground truth data (GPS) is then measured in longitudinal and lateral direction (with respect to GPS). Also, an average over these errors is calculated.

Test Results and Discussion

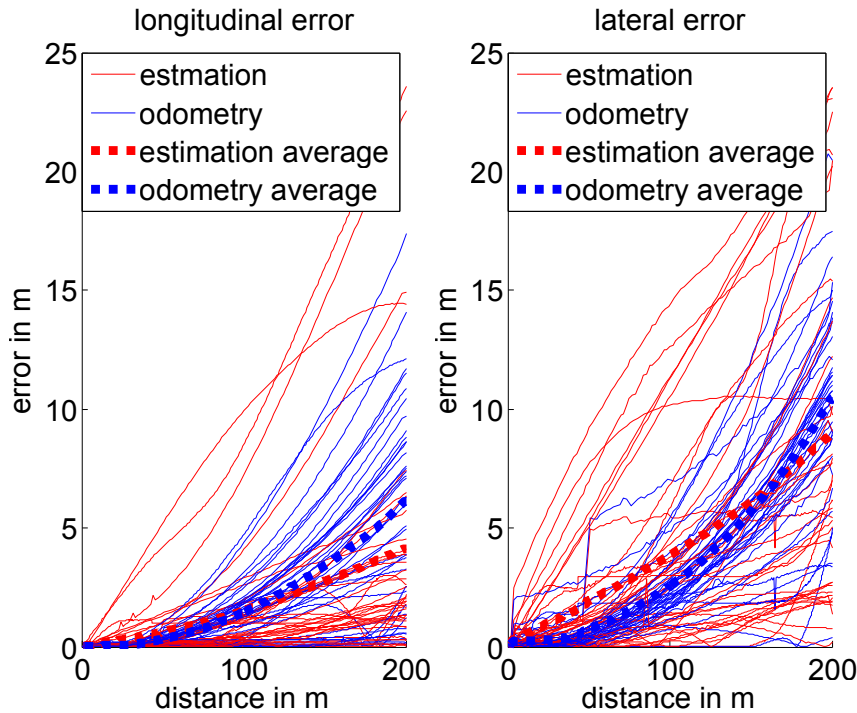


Figure 6.6: Longitudinal and lateral error of wheel and visual odometry (segment length of 200 m)

Figures 6.6 shows that odometry shows good accuracy on short distances. This finding supports the use of odometry as an initial pose estimate and as a pose prior in the optimization. The visual odometry performs slightly worse on short scale. However, accuracy is still on a good level. On average, visual odometry stays within a bound of around 5% error with many good estimates and a few outliers.

An interesting fact is that the errors of the wheel odometry both lateral and longitudinal shows more of an exponential growth while the errors visual odometry grow almost linearly. This is also visible in figure 6.7 where 400 m segments are used. Therefore, the initial advantage of wheel odometry in terms of accuracy is soon lost and visual odometry becomes much more accurate. Again, wheel odometry stays within a bound of 5% error which supports the hypothesis of linear growth of the errors.

Interestingly, wheel odometry also performs worse in longitudinal direction. This is a little surprising as one would expect wheel odometry to work very well in driving

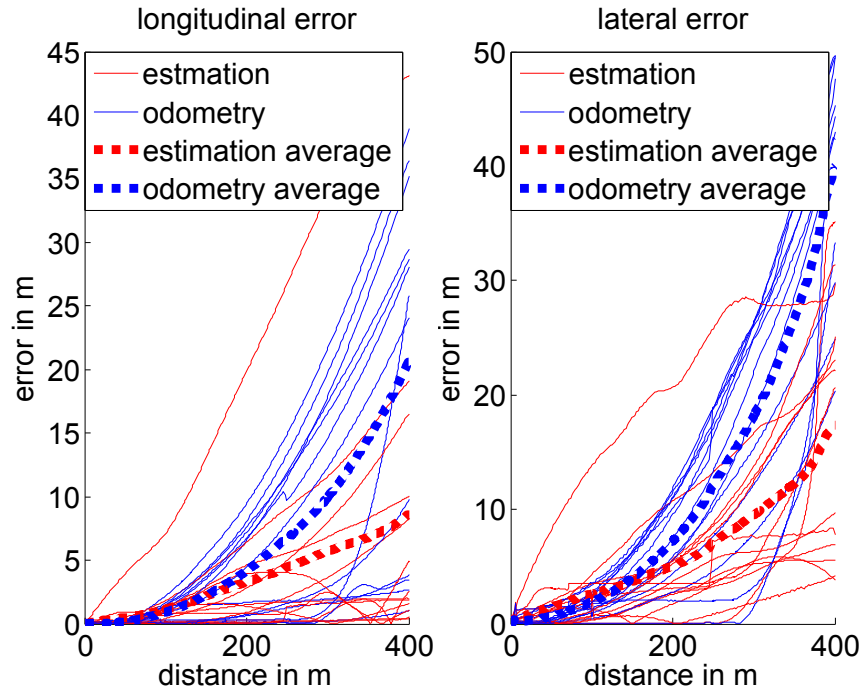


Figure 6.7: Longitudinal and lateral error of wheel and visual odometry (segment length of 400 m)

direction. However, when looking at the topview pose estimates in figure 6.8 one can see a very likely cause of this. While visual odometry shows quite a good performance, the rotation of the wheel odometry measurements drifts heavily. This could also explain the exponential growth of the wheel odometry errors and why the errors of wheel odometry are lower on a short scale.

While visual odometry also depends on wheel odometry, figure 6.8 shows that the visual odometry can very well compensate for the rotational drift of wheel odometry. The translational drift is within reasonable limits. One explanation why rotational correction works very well while translation is good but not outstanding lies in the camera configuration. Through its omnidirectional view, landmarks in all directions can be triangulated. The overlapping field of view possibly helps here as well. Due to the SURF approach with globally strongest keypoints, there are more keypoints on the structural rich background (trees, buildings etc.) than the low textured road in the foreground. Thus, translation suffers from a small amount of drift in the short scale, but works great on the long scale.

While visual odometry still slightly drifts, this test proves the high quality of the visual estimate. This is even more pronounced when looking at the scale of the test. After a driven distance of more than 7 km at various speeds and tens of thousands of processed frames, the overall drift in rotation is only slightly more than 90° . And these results are achieved with a simple two window bundle adjustment without loop closure detection or any additional local optimization.

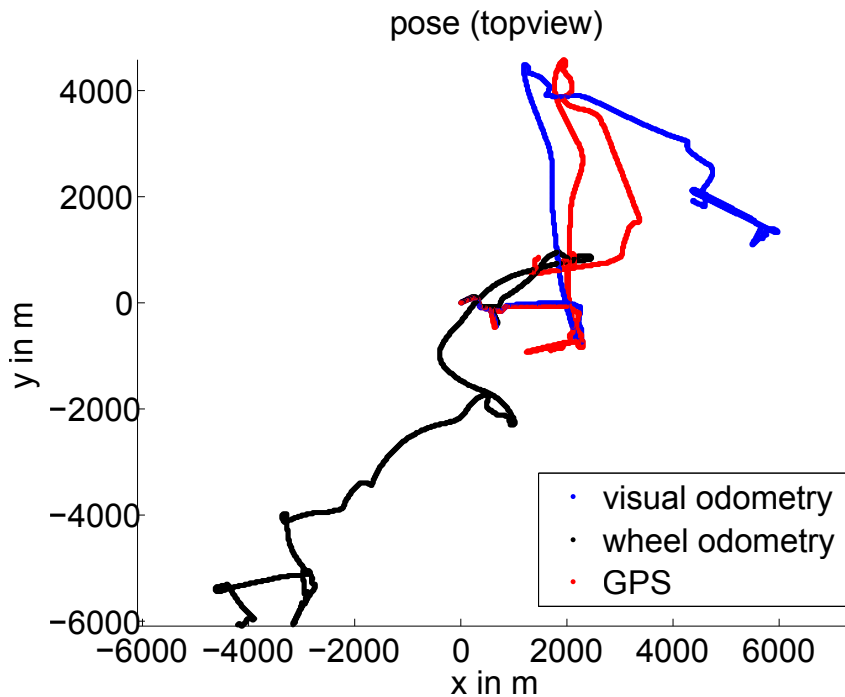


Figure 6.8: Topview plot of the different pose data for a 7 km dataset

6.3.3 Long Distance Tests

While section 6.3.2 already shows the results for a quite long dataset, the performance of the pipeline is further verified to analyze the robustness and accuracy for even longer drives. This does not only test the quality of the pipeline but also the stability of the implementation.

Test Scenario

This time, a 44 minute long dataset with more than 20 km distance driven is taken. The recording is taken from a drive through and around the area of Wolfsburg. It contains regular city traffic, driving and standing phases, parking lot situations as well as a short drive on the nearby highway.

Test Results and Discussion

Figure 6.9 shows a comparison of the overhead poses. In red the iTrace GPS data is shown which can be taken as an almost perfect reference on large scale. The blue visual odometry is able to replicate the route quite well. The low amount of drift can be seen in the square similar shape that is driven twice. While the iTrace perfectly overlaps, the visual odometry shows some slight drift. In contrast, the wheel odometry is not able to accurately replicate the shape even once and also shows a large drift over the whole

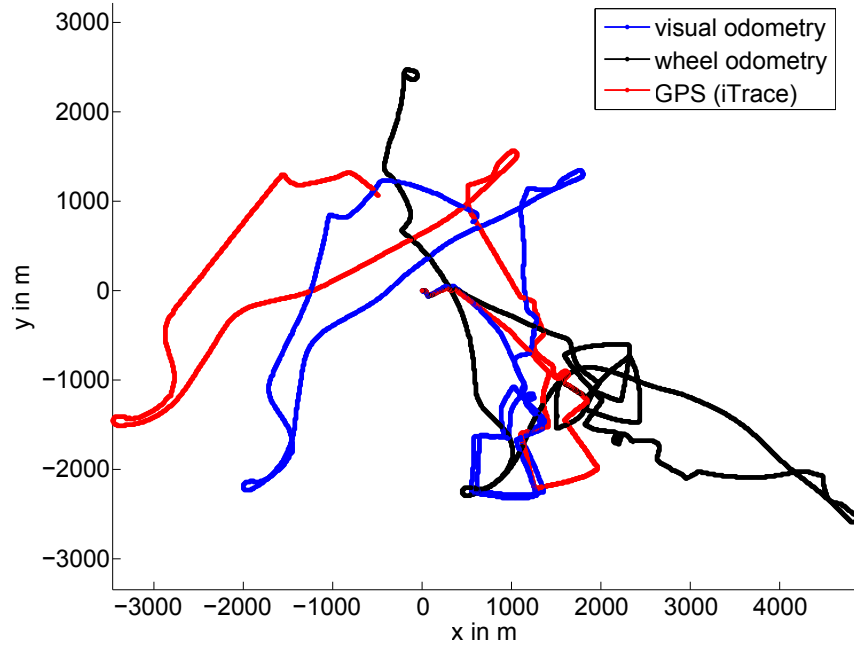


Figure 6.9: Topview plot of the different pose data for a long distance test (20 km, 0:44 h)

dataset. As a result, the overall shape of the driven path is hard to recognize when looking at wheel odometry.

Summarizing, the visual odometry shows good performance both while driving slow as well as while driving fast. Additionally, the drift is quite low which is even more convincing when looking at the rather small window for the bundle adjustment of only two frames.

Furthermore, the stability of the implementation is proven. It is able to process more than 100 GB of data and about 45 minutes driving distance without crashing or other instabilities.

6.3.4 Timings

Per definition the localization pipeline should run in realtime. At best, all incoming frames are processed and none have to be skipped. Due to the image frame rate of 12.5 Hz the maximum execution time may not exceed $t_{e,max} = \frac{1}{12.5} = 0.08$ s. Also, if this limit is exceeded it is helpful to see which component consumes the largest part of the required execution time.

Test Scenario

For the scenario a test sequence of 100 frames is used. Also, all available features of the pipeline, including cone based matching, are activated⁴. There is a total execution

⁴The batch estimator is not activated as it is not speed optimized at the current stage

timer and additionally each pipeline step is timed individually. While this includes most of the overhead, ADTF overhead is not included. As feature extraction is the slowest part of the pipeline, the tests are run with SURF and BRISK both on full size and on down sampled images (half size) to evaluate the influence of extractor type and image size.

Test Results and Discussion

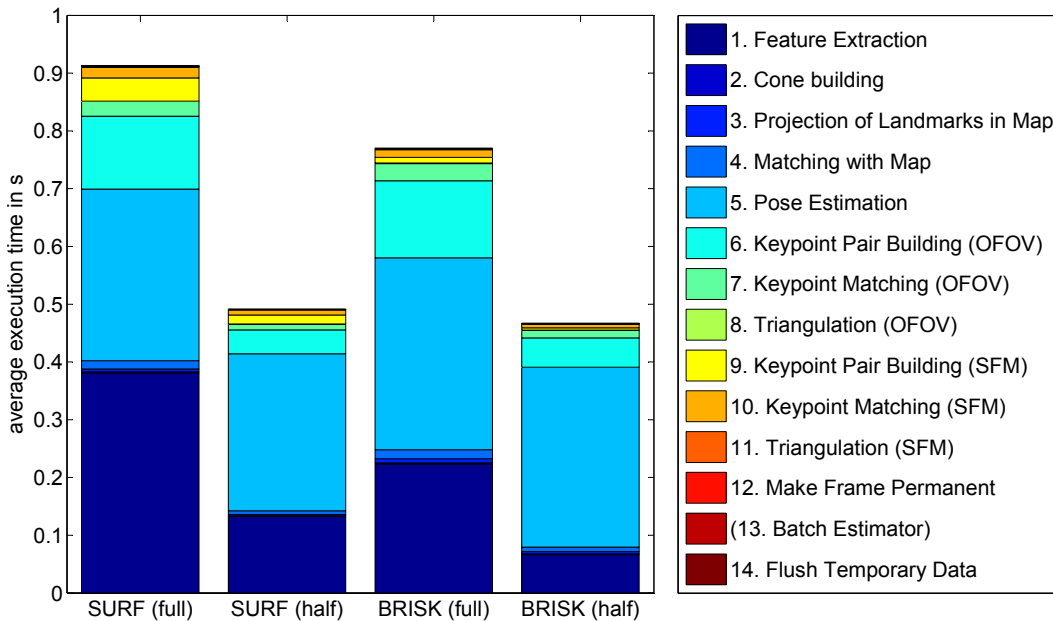


Figure 6.10: Visual odometry pipeline timings for different feature extractor and image size combinations

The recordings in figure 6.10 are not recorded on the reference PC but a Laptop. The general architecture (8 core Intel 64bit processor and NVidia GPU) are the same, but performance-wise the laptop cannot live up to the PC. While the absolute values are not accurate, the relative timings are representative.

Figure 6.10 shows that the dominant parts of the pipeline timings are the feature extraction and pose estimation. But also the cone matching consumes a significant amount of time. As expected, BRISK feature extraction is significantly faster than SURF. Also the when downsampling the images to half of the original size, the extraction timings are far less than half of the ones recorded for full size images. This is true for both extractors. For both feature extractors, the number of detected keypoints is about the same which can be seen when looking at the keypoint pair building which consumes about the same time. As expected with a similar number of keypoints, matching for BRISK is faster than SURF. However, compared to the other timings, this only has a small influence on the overall execution time.

Pose estimation is slow for a few reasons. First, the purpose of the ASLam optimizer is quick prototyping and while it is speed optimized, there is probably still room for improvement. Additionally, building the optimization problem is expensive. As implemented at the current stage, the optimization problem and all involved error terms are rebuilt every frame. However, as the two window bundle adjustment acts on two subsequent frames, the error terms of the newer frame could be stored for usage during the next iteration.

Realtime Test

One question that immediately arises is how close the visual odometry pipeline is to running realtime. To investigate the speed capabilities, the fastest configuration from above (BRISK on half size images⁵) is chosen. Furthermore, landmark estimation and matching/triangulation in overlapping field of view are switched off. While this might introduce additional drift it will also simplify the optimization problem. To record good absolute data, the realtime test is run on the V-Charge reference PC.

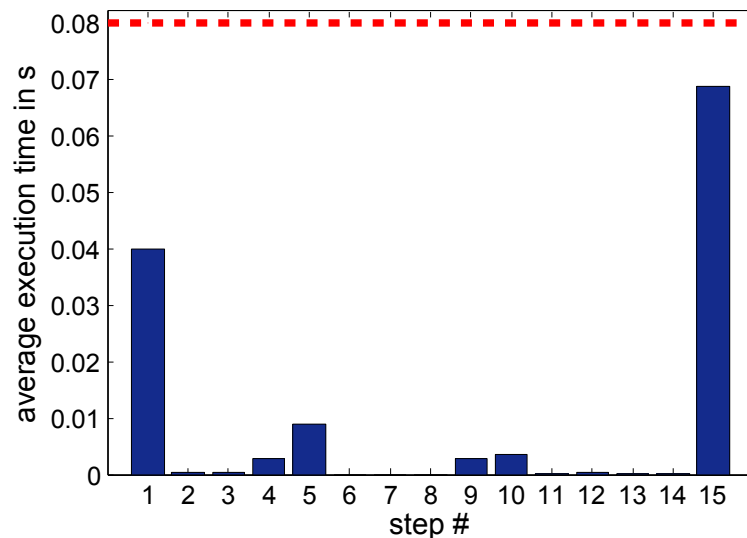


Figure 6.11: Realtime test of the visual odometry pipeline (legend see figure 6.10)

Figure 6.11 shows some interesting results. First, extraction (step #1) is about as fast as during the run on the different machine (6.10) proving that the timings are not too different on both machines. Furthermore, a significant speed increase for pose estimation (step #5) is observable. The reason for that is that the landmark positions are not refined in this test. This also gives an indication where time is consumed in the pose estimation step. As building of the optimization problem and the error terms is the same for both tests, the optimization itself plus updating the landmark positions are possibly slow.

⁵This still means a resolution of 640x480, which is a very commonly used image size for visual odometry.

The most interesting observation however is that the total execution time (indicated as step #15) is well below the critical point of 0.08 s per image frame (indicated as a red dashed line). This shows that the visual odometry pipeline in its current implementation is already realtime capable. Of course, this is achieved by a simpler configuration. However, by carefully selecting a good simplified configuration, visual odometry can still be accurate.

Besides the measures applied in this test, there are many other potential simplifications to increase speed. Amongst others, possible simplifications are:

- frame rate reduction (only process every n-th frame)
- ignoring certain cameras (e.g. front and back)
- disable Uncertainty Cone Based Matching
- decrease the number of extracted features
- use an offline generated map (and disable online mapping)

It is impossible to tell which measures should be preferred and will also depend on a case to case decision. However, the wide range of measures allows to configure the localization pipeline to run in realtime while still providing good results.

7 Summary

7.1 Conclusion

7.1.1 Setup and Integration

The presented thesis together with the batch estimation thesis ([7]) and the efforts by the V-Charge group at VW set the basis for visual localization for the V-Charge project. Starting from a bare Windows installation a lot of integration work has been done in the initial phase of the project. Especially, the fusion of proprietary VW software, ASL software and open source libraries presented a challenge. The result is a pipeline that integrates well into the V-Charge environment and is ready to be run on the car. Also the implementation provides a solid base for future development. Additionally, it is also portable, able to be run without VW's ADTF as well as on platforms other than Windows.

Furthermore, an essential part of the integration work was gathering of data from various sources and analyzing it. As shown in the case of odometry (see section 6.3.1), input data quality varies a lot throughout the project. Also, an initial lack of documentation impeded but also underlined the importance of this analysis. As a result a more thorough understanding of the input data is now present at the ETH group. Also a rich set of preprocessing and analysis tools is made available for future use and the pipeline is able to handle different quality input data.

7.1.2 Visual Odometry Pipeline

For research purposes, there is not one single implementation for the visual odometry pipeline but a choice of different modules is offered. These modules can be selected/exchanged based on the task and available computational power. An additional rich set of configuration parameters allows further customization. This flexibility paired with the high flexibility of the ASLam library opens up a vast set of other camera based applications.

The presented visual odometry pipeline is the first of its kind, using highly distorted, high resolution fisheye images. The innovative matching strategy helps to efficiently match features in these distorted images and is robust to descriptor changes. Paired with the optimizer back-end, the pipeline produces long and robust feature tracks which form a basis for the pose estimation but also the batch estimator.

The back-end efficiently combines pose estimation, outlier detection and landmark refinement in one single optimization problem. The resulting estimated poses already

shows a strong improvement over wheel odometry, especially on large scales. Due to the pose prior the estimated pose is smooth and does not fail in low feature regions. While the pipeline represents a sophisticated SLAM approach, it can be easily configured to run in real time. Additionally, the structure enables seamless integration with an offline generated map. While the pipeline still suffers from a few smaller issue, it tackles the major challenges of the presented system and is ready for future work to be integrated.

7.2 Future Work

7.2.1 Integration

The major focus of future work should lie on the integration on the car and with the offline map.

The first necessary step is to improve the calibration for Kermit as this will be the car used for the upcoming V-Charge review presentation. Additionally, more speed improvements are required. While the visual odometry pipeline can be run in realtime using a simplified configuration, its full potential cannot yet be exploited. Also, the upcoming batch estimator will consume additional computational power. However, the reference PC's capabilities provide enough room for speed improvements.

Once the above issues are solved, the next test should be localizing against an offline map. Also tests in the future application environment (dark parking garages, parking lots with changing scenes) will have to be conducted.

7.2.2 Visual Odometry Pipeline

While the global framework proves to be working nicely, a few smaller but potentially very useful alterations are possible.

On the front-end side the cone matching approach already shows great results but still possesses an even higher potential. Improvements especially in terms of robustness against pose estimation and calibration noise as well as in terms of speed should be pursued. But matching can probably also be improved when undistorting or at least rotationally aligning feature descriptors.

Also, further filtering of the incoming odometry data will not only help with this issue but will provide the estimator with a better initial guess and pose prior. Here, also developing a vehicle model and using raw odometry is thinkable.

On the back-end, also speed improvements are required to make landmark refinement realtime capable. The best solution would possible be to closely integrate pose estimation and batch estimation

8 Account

List of Figures

1.1	Sensor locations on the V-Charge vehicles [28]	2
1.2	Outline of the Localization Pipeline	7
1.3	Outline of the Visual Odometry Pipeline	8
1.4	Globally defined World Reference Frame [28]	8
1.5	Local Reference Frame attached to the scene [28]	9
1.6	Vehicle Reference Frame attached to the vehicle [28]	10
2.1	Spherical camera model as proposed by C. Mei [21]	11
2.2	Comparison of perfect (undistorted) spherical camera images (right camera) for different values of ξ	15
2.3	Illustration of the epipolar constraint [29]	16
2.4	Output of the epipolar line based calibration verification tool for Grobi .	21
2.5	Output of the epipolar line based calibration verification tool for Kermit	22
3.1	Overview of the localization pipeline highlighting the front-end	23
3.2	Illustration of intersecting Uncertainty Cones	32
3.3	Alteration of the Uncertainty Cone to take transformation uncertainty into account	34
4.1	Overview of the localization pipeline highlighting the back-end	39
4.2	Comparison of different weighting factors for estimation problems . . .	41
5.1	Graphical user interface of ADTF	50
5.2	Structure of the Localization Pipeline	52
6.1	Reprojection errors after optimization sorted by error	62
6.2	Inlier/outlier ratio of landmark observations	63
6.3	Pose estimates generated by the visual odometry	64
6.4	Plot of preprocessed wheel odometry visualizing the quantization effect	65
6.5	Comparison of filtered to unfiltered wheel odometry	66
6.6	Longitudinal and lateral error of wheel and visual odometry (segment length of 200 m)	67
6.7	Longitudinal and lateral error of wheel and visual odometry (segment length of 400 m)	68
6.8	Topview plot of the different pose data for a 7 km dataset	69

6.9	Topview plot of the different pose data for a long distance test (20 km, 0:44 h)	70
6.10	Visual odometry pipeline timings for different feature extractor and image size combinations	71
6.11	Realtime test of the visual odometry pipeline (legend see figure 6.10) .	72

Bibliography

- [1] JP Barreto and H Araujo. Issues on the geometry of central catadioptric image formation. In *Computer Vision and Pattern Recognition, 2001*, volume 2, pages 422–427, 2001.
- [2] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (SURF). *Computer vision and image . . .*, 110(03):346–359, 2008.
- [3] P Beardsley, A Zisserman, and D Murray. Navigation using affine structure from motion. *Computer Vision—ECCV’94*, pages 85–96, 1994.
- [4] J Canny. A computational approach to edge detection. *IEEE transactions on pattern analysis and machine intelligence*, 8(6):679–98, June 1986.
- [5] Gerardo Carrera, Adrien Angeli, and Andrew J. Davison. SLAM-based automatic extrinsic calibration of a multi-camera rig. *2011 IEEE International Conference on Robotics and Automation*, pages 2652–2659, May 2011.
- [6] Jonathan Courbon, Youcef Mezouar, and P Martinet. Evaluation of the Unified Model of the Sphere for Fisheye Cameras in Robotic Applications. *Advanced Robotics*, 26(September 2012):37–41, 2012.
- [7] Fabio Diem. Online Batch Estimation for an Autonomous Car, 2012.
- [8] MA Fischler and RC Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6), 1981.
- [9] P Furgale. *Extensions to the Visual Odometry Pipeline for the Exploration of Planetary Surfaces*. PhD thesis, University of Toronto, 2012.
- [10] T Gandhi and M Trivedi. Parametric ego-motion estimation for vehicle surround analysis using an omnidirectional camera. *Machine Vision and Applications*, 2005.
- [11] Christopher Geyer and Kostas Daniilidis. A unifying theory for central panoramic systems and practical implications. *Computer Vision—ECCV 2000*, pages 445–461, 2000.
- [12] Chris Harris and Mike Stephens. A combined corner and edge detector. *Alvey vision conference*, pages 147–152, 1988.

- [13] R Hartley and A Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.
- [14] Richard I. Hartley and Peter Sturm. Triangulation. *Computer Vision and Image Understanding*, 68(2):146–157, November 1997.
- [15] Michael Kaess and Frank Dellaert. Probabilistic structure matching for visual SLAM with a multi-camera rig. *Computer Vision and Image Understanding*, 114(2):286–296, February 2010.
- [16] K. Kanatani, Y. Sugaya, and H. Niitsuma. Triangulation from Two Views Revisited: Hartley-Sturm vs. Optimal Correction. *Proceedings of the British Machine Vision Conference 2008*, pages 18.1–18.10, 2008.
- [17] Y. Kanazawa and K. Kanatani. Do we really have to consider covariance matrices for image features? *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, 2:301–306, 2001.
- [18] T. Kazik, L. Kneip, J. Nikolic, M. Pollefeys, and R. Siegwart. Real-time 6D stereo Visual Odometry with non-overlapping fields of view. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1529–1536, June 2012.
- [19] Jun-sik Kim, Myung Hwangbo, and Takeo Kanade. Motion estimation using multiple non-overlapping cameras for small unmanned aerial vehicles. *2008 IEEE International Conference on Robotics and Automation*, pages 3076–3081, May 2008.
- [20] Stefan Leutenegger, Margarita Chli, and Roland Y. Siegwart. BRISK: Binary Robust invariant scalable keypoints. *2011 International Conference on Computer Vision*, pages 2548–2555, November 2011.
- [21] Christopher Mei. Projection model, 2006.
- [22] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. Fast-SLAM: A factored solution to the simultaneous localization and mapping problem. *Proceedings of the National . . .*, 2002.
- [23] K Murphy. Bayesian Map Learning in Dynamic Environments. In *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*, pages 1015–1021, Denver, CO, USA, 1999.
- [24] M Schönbein, B Kitt, and M Lauer. Environmental Perception for Intelligent Vehicles Using Catadioptric Stereo Vision Systems. *aass.oru.se*, pages 1–6, 2011.
- [25] A Stenkula. *Vehicle vicinity from above: a study of all-around environment displaying system for heavy vehicles*. PhD thesis, Royal Institute of Technology, 2009.

- [26] Jean-Philippe Tardif, Yanis Pavlidis, and Kostas Daniilidis. Monocular visual odometry in urban environments using an omnidirectional camera. *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2531–2538, September 2008.
- [27] Bill Triggs and P McLauchlan. Bundle Adjustment—A Modern Synthesis. *International Workshop on Vision Algorithms*, 34099:298—372, 2000.
- [28] V-Charge. V-Charge Wiki, 2012.
- [29] Wikipedia. Epipolar Constraint, 2012.
- [30] Zhengyou Zhang. Parameter estimation techniques: A tutorial with application to conic fitting. *Image and vision Computing*, 1997.

D Online Batch Estimation for an Autonomous Car

Authors: Diem

Link to [Table of Contents](#).

Online Batch Estimation for an Autonomous Car

Master Thesis

October 2012

Supervised by:

Dr. Paul Furgale

Dr. Cédric Pradalier

Author:

Fabio Diem

Abstract

This master thesis presents a first implementation of a batch estimator. This estimator is used within a localization system built on a VW Golf. The final batch estimator has to be real time capable and highly flexible regarding the used set of sensors. The set of sensors includes four fisheye cameras and wheel odometry. The estimated vehicle states are represented in continuous time using B-splines. Other estimated variables include the camera extrinsics and the positions of detected landmarks. The results show that the presented estimator concept works, but as it is a first implementation, not all final goals have been met.

Contents

Abstract	iii
1 Introduction	1
1.1 Project V-Charge	1
1.2 Sensor System	1
1.3 Localization Pipeline	4
1.4 Aim of this Master Thesis	5
2 Batch Estimator	7
2.1 Continuous Time State Representation	7
2.1.1 B-Splines	7
2.2 Optimization Problem	9
2.2.1 Batch nonlinear least squares optimization	9
2.2.2 Sliding window bundle adjustment	10
2.2.3 Coordinate Frames and Vehicle States	11
2.2.4 Error Formulation	12
3 Implementation	17
3.1 ASLam Backend	17
3.1.1 Design Variables	17
3.1.2 Error Terms	17
3.1.3 Optimization Problem	18
3.2 Batch Estimator	18
3.2.1 Observations	18
3.2.2 Estimation	19
3.3 Issues	20
4 Results	23
4.1 Parameter Settings	23
4.2 Batch Estimated Visual Odometry Pipeline	24
4.2.1 Overall Results	24
4.2.2 Short Segment Results	26
4.3 Integrated Odometry, Gyroscope and Landmarks	28
4.3.1 Overall Results	28
4.3.2 Short Segment Results	30
4.4 Issues	33
4.4.1 Quality of Estimation Results	33
4.4.2 Processing Speed	33
4.4.3 Integration within Localization Pipeline	33
5 Conclusion	35

Chapter 1

Introduction

This chapter gives an overview of the context in which this thesis has been written. First, project V-Charge is introduced. Then the sensor system used within this project is presented. Finally, it is shown how this thesis integrates into project V-Charge.

1.1 Project V-Charge

The work presented in this master thesis is part of project V-Charge [1]. The objective of project V-Charge is to develop a low-cost car system which is able to autonomously navigate and drive within a designated area. A typical scenario for such a car might be a park and ride system at a train station, where the car autonomously navigates between the drop off area, the parking space and the recharge station.

V-Charge is a research project funded by the European Union. The project consortium consists of four Universities and two Industrial Partners. Among other things the Autonomous Systems Lab (ASL) at ETH Zürich [2] is responsible of delivering a real-time capable, vision based localization pipeline using the low-cost sensor system given in the project definition.

1.2 Sensor System

Most autonomous car systems presented so far use extremely expensive sensor equipment.¹ While those systems often produce good results, the high additional costs make their implementation in consumer vehicles highly unlikely. Therefore one of the key points within the V-Charge project definition is the exclusive use of low-cost sensors.

The car used for this project is a VW Golf. The sensor system consists of equipment built into a normal VW Golf and sensors included exclusively for this project. For this thesis, the following sensors have been used:

¹ E.g. the driverless car developed by Google [3] is equipped with a \$70,000 lidar (light detection and ranging) system.

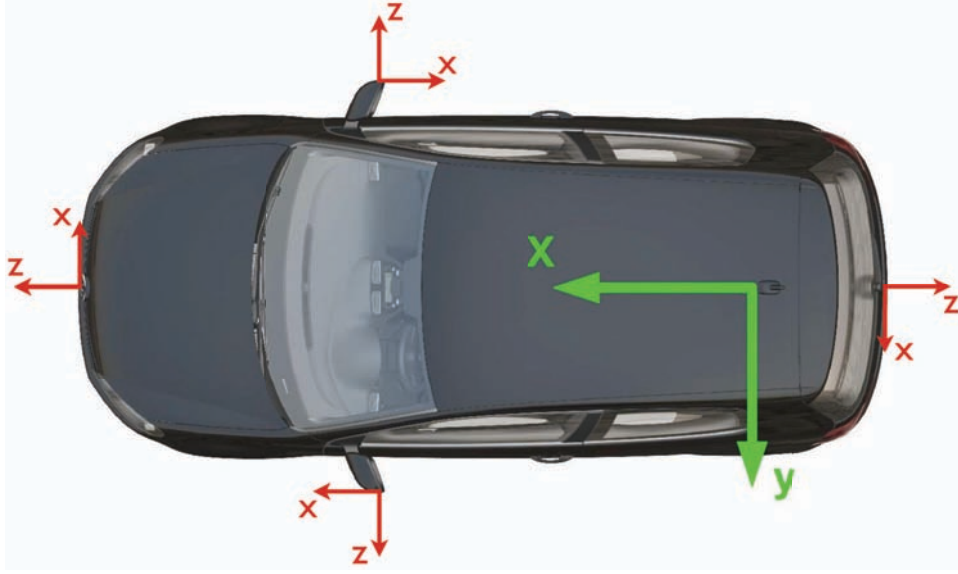


Figure 1.1: Fisheye camera reference frames

Fisheye Cameras The main sensors for this vision based localization system are four fisheye cameras. They are mounted in the front and back VW sign and the two side mirrors (red coordinate systems in figure 1.1). The images have a resolution of 1280×800 px and are captured with a frequency of 12.5 Hz. The fisheye lenses used have a maximal horizontal angle of 185° . In figure 1.2 the overlapping fields of view in the corners of the car and the distortion from the fisheye lenses are visible.



Figure 1.2: Images from the four fisheye cameras (front, right, left, rear)

Integrated Odometry VW combines raw wheelticks and the steering wheel angle to estimate basic vehicle odometry. Because the raw measured wheelticks are heavily quantized², and no filtering is applied, quantization effects can be found in the integrated odometry data as well (see figure 1.3).

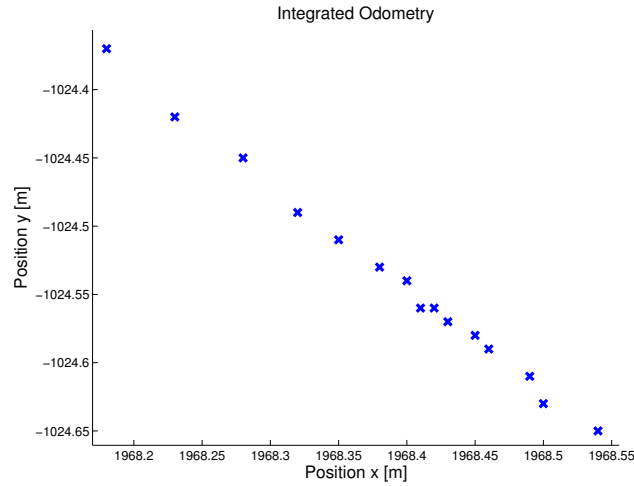


Figure 1.3: Position from integrated odometry

Accelerometer Acceleration in both horizontal directions is measured at a frequency of 50 Hz. Figure 1.4 shows that the acceleration measurements are quantized. It also shows that the sensor resolution is different depending on its axis.

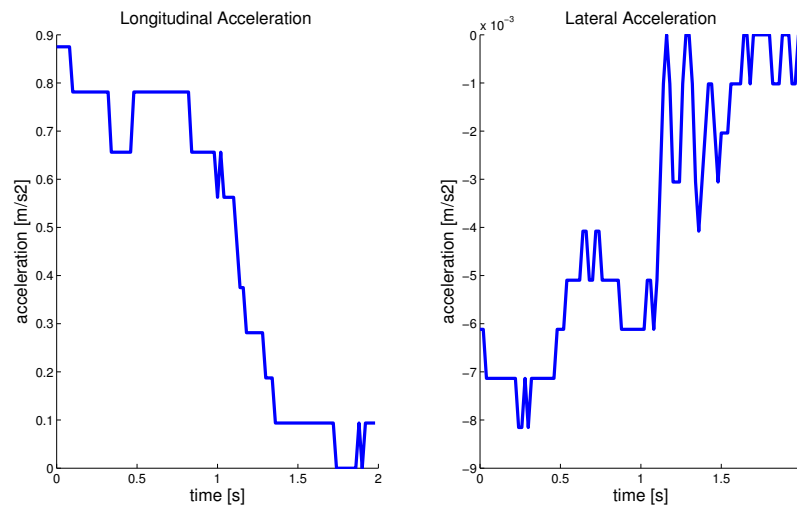


Figure 1.4: Longitudinal and lateral accelerations

² The rotary encoders used have a resolution of about 2.3 cm at 50 Hz.

Gyroscope A one axis gyroscope measures the yaw angular rate with a frequency of 50 Hz. A sample can be seen in 1.5.

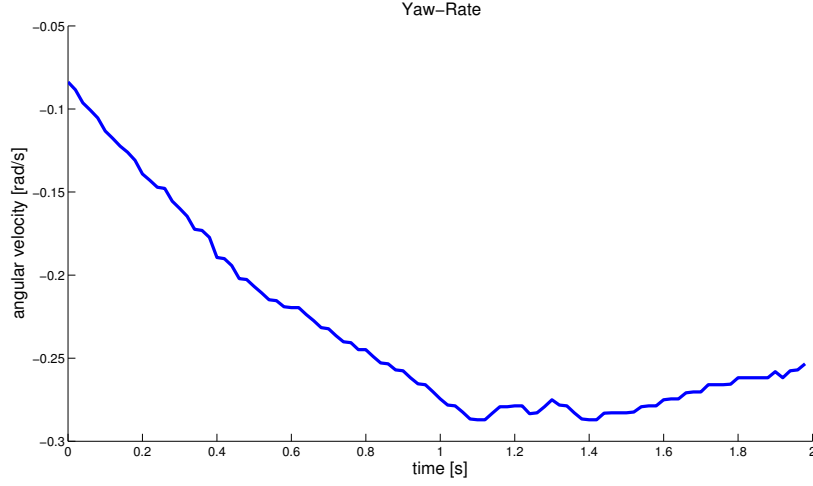


Figure 1.5: Yaw rate

iTrace The iTrace is a combined GPS and IMU unit. Its GPS data is only used as ground truth reference to compare results achieved by the localization system. GPS measurements are provided at a frequency of 100 Hz, but with signal gap of more than 0.1 sec recurring each second (see figure 1.6).

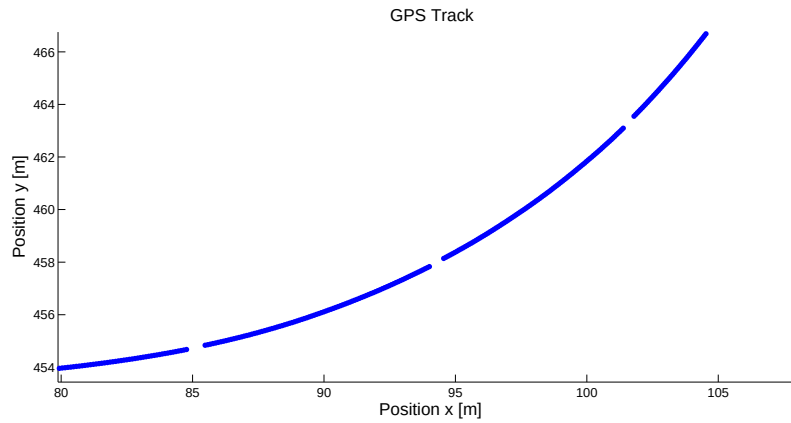


Figure 1.6: Position from GPS

1.3 Localization Pipeline

The localization pipeline has the following structure:

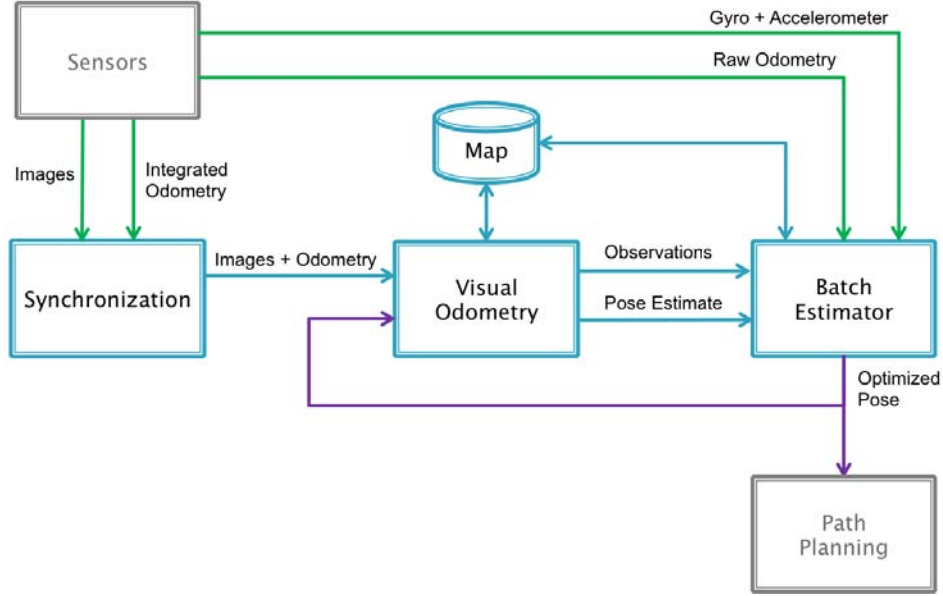


Figure 1.7: Localization Pipeline Structure

Synchronization In a first step the images from the four fisheye cameras and the integrated odometry are collected. Whenever a full set of images is available, the collected data is sent to the visual odometry pipeline.

Visual Odometry The visual odometry pipeline uses data from an existing map and the new data from synchronization to extract features from the images. In a second step, the vehicle pose and the position of landmarks associated to the features is estimated and saved in the map. The visual odometry pipeline is presented in [4].

Batch Estimator The batch estimator combines the preliminary results from the visual odometry pipeline with measurements from other sensors. Then all relevant states (vehicle pose, landmark positions, camera extrinsics etc.) are estimated over a given timespan. The results of this localization pipeline are then used in its next iteration and for other elements within project V-Charge like path planning and navigation.

1.4 Aim of this Master Thesis

The goal of the batch estimator is to improve the final vehicle pose estimate compared to the visual odometry pipeline, and to allow estimation of other states and inclusion of other sensors. This thesis presents a first implementation of the batch estimator.

Chapter 2

Batch Estimator

This chapter presents an overview of the theoretical background of the batch estimator. One part describes and explains the chosen state representation. The other part outlines the optimization problem formulation, including the integration of the various sensors.

2.1 Continuous Time State Representation

In robotics, state estimation problems are often formulated in discrete time. While this approach has the advantage of being well researched and documented, there are some major drawbacks when high frequency sensors have to be integrated. Formulating the state in discrete time means that, for each point in time where the state has to be estimated, an additional state vector is needed. Integrating e.g. an IMU¹ leads to either state vectors of unpractical size if all measurements are considered, or it requires some sort of downsampling of the original signal to keep the state vector and therefore the estimation problem reasonable.

The batch estimator used for project V-Charge has to integrate various sensors. Therefore it needs to work with measurements having not only a big variety of frequencies, but also sporadic signal drops. Furthermore, the estimation problem has to be easily adjustable to a random selection of sensors. When comparing these requirements with the known obstacles in discrete time, a continuous time approach seems to be preferable.

2.1.1 B-Splines

In Furgale et al. [5] a continuous time approach using B-splines to approximate the representation of the states is proposed. A detailed insight into B-splines in general can be found in [6]. [7] has more details on adapting B-splines for parametrizing states in robotics estimation problems.

¹ Inertial Measurement Unit, often measuring angular and linear velocities and accelerations with more than 100 Hz.

Based on [7], a cubic B-spline can be outlined as a "weighted sum of a finite set of known temporal basis functions". A cubic temporal basis function (figure 2.1) is defined as a piecewise cubic polynomial over four segments. The *knots*, which are the segment transition points, are marked with vertical lines. A cubic polynomial known as the *segment basis polynomial* defines the curve between two knots. To guarantee a smooth transition at the knots, the values as well as the first and second derivatives of two connecting cubic polynomials are identical.

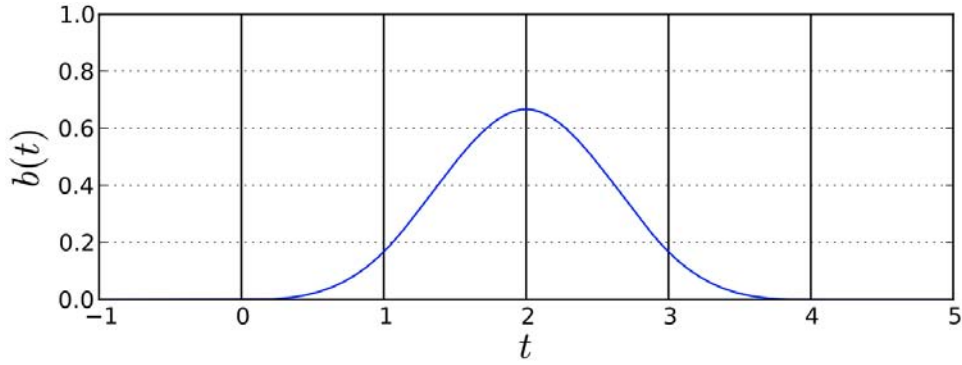


Figure 2.1: Cubic temporal basis function

By aligning temporal basis functions with a non-decreasing knot sequence, the base of a B-spline is defined. A cubic B-spline is defined to be, at each time, the weighted sum of four temporal basis functions. Therefore the only time segment in figure 2.2, where a B-spline can be defined, is $t = [3, 4]$.

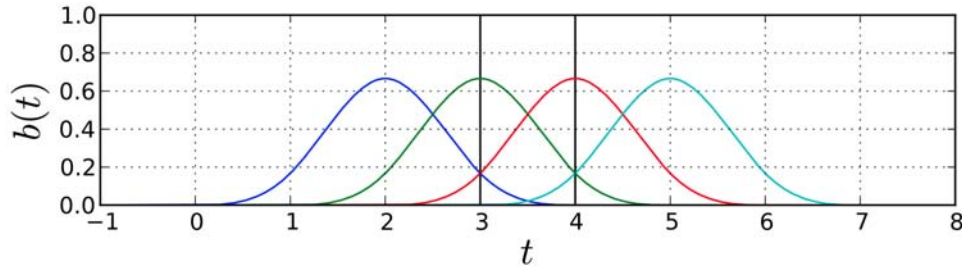


Figure 2.2: Aligned cubic temporal basis functions

Adding a weighting factor for each temporal basis function and summing them up gives the B-spline function (figure 2.3). Thus, to define the shape of a cubic B-spline over a given time interval, the following elements can be used:

- Knot distribution
- Weighting factors

For the use of B-splines in optimization problems this means that, if the knot spacing is defined, the number of optimization variables is independent of the number of measurements

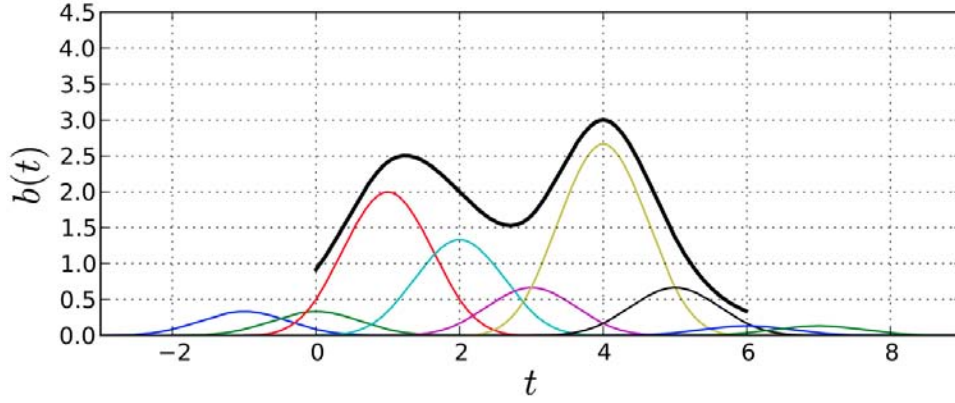


Figure 2.3: B-spline (black) as the sum of its weighted basis functions (colored)

in between two knots. Using the B-spline from figure 2.3 would mean that, independent from the number of measurements with $t = [3, 4]$, there are only four variables needed to define the B-spline within this interval.

It is obvious that, compared to the discrete time approach, using B-splines for state representation in optimization problems with high rate sensors can result in a far lower number of optimizable variables.

There is one other factor which is not that obvious, but also helps to keep the number of variables low: Using identical, analytically well defined temporal basis functions means that their respective derivatives and integrals are defined as well. Therefore the B-spline itself has analytical derivatives and integrals. This is especially helpful when sensors measuring velocities and accelerations are included in the optimization problem.

2.2 Optimization Problem

There are many ways of building an optimization problem to estimate states based on sensor measurements. For this batch estimator, a nonlinear least squares optimization using a sliding window approach is implemented. Sensors are then included as errors between the estimated state and the corresponding measurement.

2.2.1 Batch nonlinear least squares optimization

This short introduction on batch nonlinear least squares optimization is based on chapter 3.1 in [8].

The optimization problem can be formulated as a scalar sum J of squared errors $\mathbf{e}$. The optimal solution then corresponds to the choice of state parameters $\mathbf{x}$ which minimize J .

$$J(\mathbf{x}) := \frac{1}{2} \sum_i \mathbf{e}_i(\mathbf{x})^T \mathbf{W}_i \mathbf{e}_i(\mathbf{x}) \quad (2.1)$$

Each error term $\mathbf{e}_i$ is weighted by the matrix $\mathbf{W}_i$. All error terms presented in chapter 2.2.4 are based on observation models of the form

$$\mathbf{z}_i = \mathbf{h}_i(\mathbf{x}) + \mathbf{n}_i \quad (2.2)$$

where $\mathbf{z}_i$ is an observation and $\mathbf{h}_i$ is the function which describes the modeled connection between the states $\mathbf{x}$ and the observation. Each observation is assumed to be disturbed by independent, zero mean Gaussian noise $\mathbf{n}_i$:

$$\mathbf{n}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_i), \quad E[\mathbf{n}_i \mathbf{n}_i^T] = \mathbf{R}_i, \quad E[\mathbf{n}_i \mathbf{n}_k^T] = \mathbf{0} \quad (2.3)$$

The error $\mathbf{e}_i$ is defined as the difference between observation and modeled observation:

$$\mathbf{e}_i(\mathbf{x}) := \mathbf{z}_i - \mathbf{h}_i(\mathbf{x}) \quad (2.4)$$

Defining the weighting matrix $\mathbf{W}_i$ as

$$\mathbf{W}_i := E[\mathbf{e}_i \mathbf{e}_i^T]^{-1} = \mathbf{R}_i^{-1} \quad (2.5)$$

makes

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} (J(\mathbf{x})) \quad (2.6)$$

equivalent to a maximum-likelihood estimate of the state given the observations [9]. To find $\mathbf{x}^*$, the first derivative of J is set to zero:

$$\frac{\partial J(\mathbf{x})}{\partial \mathbf{x}} \stackrel{!}{=} \mathbf{0} \quad (2.7)$$

Because the observation models $\mathbf{h}$ are not necessarily linear in $\mathbf{x}$, equation 2.7 has to be solved iteratively.

2.2.2 Sliding window bundle adjustment

In general, bundle adjustment includes all available observations. While this usually results in good state estimates for all of the covered time frame, the optimization problem grows over time. This is typically not an issue for offline optimization. However in the case of online estimation with large numbers of observations and states to estimate, full bundle adjustment becomes unreasonable.

The sliding window approach minimizes this issue by only considering data from a shorter window. This window can e.g. be defined as the last few seconds. This means that only the observations of the last few seconds are included in each optimization. While this usually leads to slightly worse long term estimation results compared to full bundle adjustment, it helps to keep the optimization problem size reasonable.

More details on bundle adjustment in general can be found e.g. in [10]. Sliding window bundle adjustment is presented in [11].

2.2.3 Coordinate Frames and Vehicle States

Coordinate Frames The following three coordinate frames are defined: An inertial frame (*Local Reference Frame*) $\vec{\mathcal{F}}_l$, a frame fixed to the vehicle (*Vehicle Reference Frame*) $\vec{\mathcal{F}}_v$ and a frame fixed to the sensor (*Sensor Reference Frame*) $\vec{\mathcal{F}}_s$.

Vehicle Pose The vehicle pose is described by six B-splines. The first three splines describe the vehicle position $\mathbf{r}_l^{v,l}(t)$ with respect to and expressed in the local reference frame.² The vehicle orientation is encoded in the three remaining splines $\phi(t)$ using an axis-angle representation.³ With this full vehicle pose $\mathbf{x}(t)$, the transformation $\mathbf{T}_{l,v}(t)$ taking vectors from $\vec{\mathcal{F}}_v$ to $\vec{\mathcal{F}}_l$ is:

$$\mathbf{T}_{l,v}(t) = \begin{bmatrix} \mathbf{C}_{l,v}(t) & \mathbf{r}_l^{v,l}(t) \\ \mathbf{0} & 1 \end{bmatrix} \quad \text{where} \quad \mathbf{C}_{l,v}(t) = \mathbf{C}(\phi(t)) \quad \text{and} \quad \mathbf{x}(t) = \begin{bmatrix} \mathbf{r}_l^{v,l}(t) \\ \phi(t) \end{bmatrix} \quad (2.8)$$

$\mathbf{C}_{l,v}$ transforms vectors from $\vec{\mathcal{F}}_v$ to $\vec{\mathcal{F}}_l$. $\mathbf{C}(\phi(t))$ is the function which builds a rotation matrix from the rotation parameters $\phi(t)$. The exact form of $\mathbf{C}(\phi(t))$ can be found on page 30 of [12].

Similarly, the time independent transformation matrix $\mathbf{T}_{v,s}$ describes the transformation from sensor to vehicle frame. It contains a rotation matrix $\mathbf{C}_{v,s}$ and translation vector $\mathbf{r}_v^{s,v}$.

Linear and Angular Vehicle Velocity and Acceleration in the Local Reference Frame

As $\vec{\mathcal{F}}_l$ is an inertial frame, velocity and acceleration in $\vec{\mathcal{F}}_l$ are:

$$\mathbf{v}_l^{v,l} = \dot{\mathbf{r}}_l^{v,l} \quad (2.9)$$

$$\mathbf{a}_l^{v,l} = \dot{\mathbf{v}}_l^{v,l} = \ddot{\mathbf{r}}_l^{v,l} \quad (2.10)$$

$$\boldsymbol{\omega}_l^{v,l} = \mathbf{S}(\phi(t)) \dot{\phi}(t) \quad (2.11)$$

$$\boldsymbol{\alpha}_l^{v,l} = \dot{\boldsymbol{\omega}}_l^{v,l} \quad (2.12)$$

$$(2.13)$$

The exact form of matrix $\mathbf{S}(\phi(t))$, which relates rotation parameters to angular velocities, can be found on page 31 of [12].

² $\mathbf{r}_a^{b,c}$ is a vector r from point c to b and expressed in $\vec{\mathcal{F}}_a$.

³ The detailed information on this and other common rotation representations can be found in [12].

Linear and Angular Vehicle Velocity and Acceleration in the Vehicle Reference Frame

$$\mathbf{v}_v^{v,l} = \mathbf{C}_{l,v}^T \mathbf{v}_l^{v,l} = \mathbf{C}_{l,v}^T \dot{\mathbf{r}}_l^{v,l} \quad (2.14)$$

$$\boldsymbol{\omega}_v^{v,l} = \mathbf{C}_{l,v}^T \boldsymbol{\omega}_l^{v,l} \quad (2.15)$$

$$\mathbf{a}_v^{v,l} = \mathbf{C}_{l,v}^T \mathbf{a}_l^{v,l} = \mathbf{C}_{l,v}^T \ddot{\mathbf{r}}_l^{v,l} \quad (2.16)$$

$$\boldsymbol{\alpha}_v^{v,l} = \mathbf{C}_{l,v}^T \boldsymbol{\alpha}_l^{v,l} = \mathbf{C}_{l,v}^T \dot{\boldsymbol{\omega}}_l^{v,l} \quad (2.17)$$

$$(2.18)$$

Linear and Angular Vehicle Velocity and Acceleration in the Sensor Reference Frame

$$\mathbf{v}_s^{s,l} = \mathbf{C}_{v,s}^T \mathbf{v}_v^{s,l} = \mathbf{C}_{v,s}^T \left[\mathbf{v}_v^{v,l} + \boldsymbol{\omega}_v^{v,l \times} \mathbf{r}_v^{s,v} \right] \quad (2.19)$$

$$\mathbf{a}_s^{s,l} = \mathbf{C}_{v,s}^T \mathbf{a}_v^{s,l} = \mathbf{C}_{v,s}^T \left[\mathbf{a}_v^{v,l} + \boldsymbol{\alpha}_v^{v,l \times} \mathbf{r}_v^{s,v} + \boldsymbol{\omega}_v^{v,l \times} (\boldsymbol{\omega}_v^{v,l \times} \mathbf{r}_v^{s,v}) \right] \quad (2.20)$$

$$\boldsymbol{\omega}_s^{s,l} = \boldsymbol{\omega}_s^{v,l} = \mathbf{C}_{v,s}^T \boldsymbol{\omega}_v^{v,l} \quad (2.21)$$

$$\boldsymbol{\alpha}_s^{s,l} = \boldsymbol{\alpha}_s^{v,l} = \mathbf{C}_{v,s}^T \boldsymbol{\alpha}_v^{v,l} \quad (2.22)$$

2.2.4 Error Formulation

Based on chapter 2.2.1, the sensors introduced in chapter 1.2 are included into the optimization problem by deriving error terms connecting actual with predicted observations. Below, an error term for each observation type is presented.

Accelerometer

The accelerometer measures the linear acceleration of the Vehicle as seen from $\mathcal{F}_l$ and expressed in $\mathcal{F}_s$ at time t_k . Included in this measurement $\mathbf{z}_{acc,k}$ is gravity $\mathbf{g}_l$. Each measurement is corrupted by a time-varying accelerometer bias $\mathbf{b}_a(t_k)$ and zero-mean Gaussian noise $\mathbf{n}_{ak}$.

$$\begin{aligned} \mathbf{z}_{acc,k} &= \mathbf{a}_s^{s,l}(t_k) - \mathbf{C}_{s,l}(t_k) \mathbf{g}_l + \mathbf{b}_a(t_k) + \mathbf{n}_{ak} = \mathbf{C}_{v,s}^T \mathbf{a}_v^{s,l}(t_k) - \mathbf{C}_{s,l}(t_k) \mathbf{g}_l + \mathbf{b}_a(t_k) + \mathbf{n}_{ak} \\ &= \mathbf{C}_{v,s}^T \left[\mathbf{C}_{l,v}^T(t_k) (\mathbf{a}_l^{v,l}(t_k) - \mathbf{g}_l) + \boldsymbol{\alpha}_v^{v,l}(t_k) \times \mathbf{r}_v^{s,v} + \boldsymbol{\omega}_v^{v,l}(t_k) \times (\boldsymbol{\omega}_v^{v,l}(t_k) \times \mathbf{r}_v^{s,v}) \right] \\ &\quad + \mathbf{b}_a(t_k) + \mathbf{n}_{ak} \\ \Rightarrow \mathbf{z}_{acc,k} &= h_{acc}(\ddot{\mathbf{r}}(t_k), \dot{\boldsymbol{\phi}}(t_k), \dot{\boldsymbol{\phi}}(t_k), \ddot{\boldsymbol{\phi}}(t_k), \mathbf{b}_a(t_k)) + \mathbf{n}_{ak} \\ &\quad \text{with } \dot{\mathbf{b}}_a(t) = \mathbf{w}_{acc}(t) \quad \mathbf{w}_{acc}(t) \sim \mathcal{GP}(\mathbf{0}, \mathbf{Q}_{acc} \delta(t - t')) \end{aligned}$$

Gyroscope

The gyroscope measures the angular velocity of $\mathcal{F}_s$ as seen from $\mathcal{F}_l$ and expressed in $\mathcal{F}_s$ at time t_k . Each measurement $\mathbf{z}_{gyro,k}$ is corrupted by a time-varying gyroscope bias $\mathbf{b}_g(t_k)$ and zero-mean Gaussian noise $\mathbf{n}_{gk}$.

$$\begin{aligned}
z_{gyro,k} &= \omega_s^{s,l}(t_k) + \mathbf{b}_g(t_k) + \mathbf{n}_{gk} = \mathbf{C}_{v,s}^T(t_k) \omega_v^{v,l}(t_k) + \mathbf{b}_g(t_k) + \mathbf{n}_{gk} \\
\Rightarrow z_{gyro,k} &= h_{gyro}(\phi(t_k), \dot{\phi}(t_k), \mathbf{b}_g(t_k)) + \mathbf{n}_{gk} \\
&\text{with } \dot{\mathbf{b}}_g(t) = \mathbf{w}_{gyro}(t) \quad \mathbf{w}_{gyro}(t) \sim \mathcal{GP}(\mathbf{0}, \mathbf{Q}_{gyro} \delta(t - t'))
\end{aligned}$$

Landmark Reprojection

The features, which get detected and matched in the visual odometry pipeline, are included as landmark reprojection errors. For this, the landmark is projected into the corresponding camera frame and on the image plane. The error is then built from the Mahalanobis distance between this projected landmark position on the image plane and the keypoint it was matched to. More details on the used landmark reprojection error can be found in [4].

Steering Wheel Angle

The steering transfer function⁴, which defines the actual steering angle γ of an imaginary front middle wheel, based on the steering wheel angle γ_{sw} , is:

$$\gamma(t_k) = -0.00021789 + 0.06051162 \cdot \gamma_{sw,k} + 2.0145 \cdot 10^{-5} \cdot \gamma_{sw,k}^2 + 8.3272 \cdot 10^{-5} \cdot \gamma_{sw,k}^3$$

To avoid inverting this cubic transfer function, γ_{sw} can be defined as additional state. This makes predicting, measuring and including the steering wheel angle in error terms easier than if γ would be used as state.

Each steering wheel angle measurement $z_{sw,k}$ is corrupted by zero-mean Gaussian noise $n_{\gamma k}$. The sensor model for measuring the true steering wheel angle γ_{sw} is:

$$\begin{aligned}
z_{sw,k} &= \gamma_{sw}(t_k) + n_{\gamma k} \\
\Rightarrow z_{sw,k} &= h_{sw}(\gamma_{sw}(t_k)) + n_{\gamma k}
\end{aligned}$$

Transformation

A six dimensional error vector is useful to define the difference between two almost identical transformation matrices. Based on the expressions derived in chapter 3.3 of [8], the difference between a transformation matrix $\mathbf{T}$ and its prior $\hat{\mathbf{T}}$ is:

$$\mathbf{T}\hat{\mathbf{T}}^{-1} \approx (\mathbf{1} - \mathbf{e}(\mathbf{T})^{\boxplus})$$

By reverting the $\boxplus$ operator, the six elements of the error vector $\mathbf{e}$ can be calculated.

⁴ provided by VW

Wheelticks

Assuming no slip, the linear velocity v of a wheel is:

$$v = \frac{c \cdot \Delta n}{\Delta t \cdot n_{circ}}$$

where c is the wheel circumference, Δn is the number of wheelticks in time Δt , and n_{circ} is the number of wheelticks in one wheel rotation.

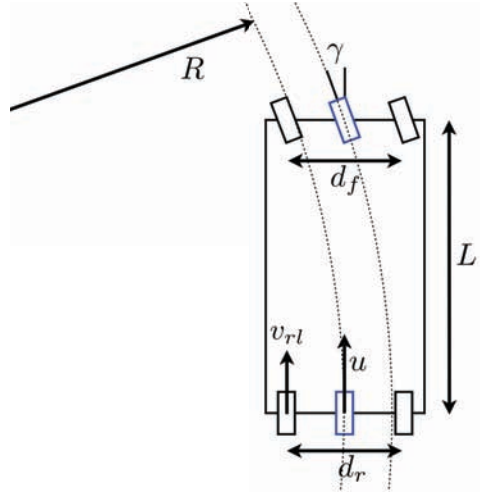


Figure 2.4: Bicycle and Ackermann vehicle model

A bicycle model is used to define the main car motion. This means that an imaginary rear middle wheel defines the car longitudinal velocity u , and an imaginary front middle wheel defines the steering angle γ (see figure 2.4). By combining this bicycle model with an Ackermann vehicle model,⁵ the four individual wheel velocities can be calculated. At this place the calculation for the rear left wheel velocity is presented. The velocities of all other wheels can be calculated similarly.

From figure 2.4 it is obvious that the radius R of the circle on which the rear middle wheel is moving is

$$R = \frac{L}{\tan(\gamma)} \quad (2.23)$$

Based on the Ackermann vehicle model, the rotation velocities of all wheels along their respective circles are identical:

$$\frac{v_{rl}}{R_{rl}} = \frac{u}{R} \quad (2.24)$$

Combining equations 2.23 and 2.2.4, the velocity of the rear left wheel is:

⁵ The Ackermann vehicle model assumes that all wheels are moving tangentially on concentric circles without wheel slip.

$$v_{rl} = R_{rl} \frac{u}{R} = u \left(1 - \frac{d_r}{2R} \right) = u \left(1 - \frac{d_r \tan(\gamma)}{2L} \right)$$

where d_r is the distance between the two rear wheels. Similarly, the remaining wheel velocities are:

$$\begin{aligned} v_{rr} &= u \left(1 + \frac{d_r \tan(\gamma)}{2L} \right) \\ v_{fl} &= u \sqrt{\tan^2(\gamma) + \left(1 - \frac{d_f \tan(\gamma)}{2L} \right)^2} \\ v_{fr} &= u \sqrt{\tan^2(\gamma) + \left(1 + \frac{d_f \tan(\gamma)}{2L} \right)^2} \end{aligned}$$

Each rear left wheeltick measurement $z_{wt,rl,k}$ is corrupted by zero-mean Gaussian noise $n_{wt,rl,k}$.

$$\begin{aligned} z_{wt,rl,k} &= \frac{\Delta t_k \cdot n_{circ}}{c_{rl}} \cdot v_{rl}(t_k) + n_{wt,rl,k} \\ &= \frac{\Delta t_k \cdot n_{circ}}{c_{rl}} \cdot u(t_k) \left(1 - \frac{d_r \tan(\gamma(t_k))}{2L} \right) + n_{wt,rl,k} \\ \Rightarrow z_{wt,rl,k} &= h_{wt,rl}(\mathbf{r}(t_k), \boldsymbol{\phi}(t_k), \gamma(t_k), \Delta t_k) + n_{wt,rl,k} \end{aligned}$$

Measurement models for the other wheels can be set up similarly.

Chapter 3

Implementation

3.1 ASLam Backend

The batch estimator is implemented using the ASLam backend. The ASLam backend is a nonlinear optimization framework currently under development at the ASL [2]. To implement an optimization problem with the ASLam backend, three elements are required: design variables, error terms and an optimization problem.

3.1.1 Design Variables

Design variables are the parameters which can be estimated. For this batch estimator the following design variables are defined:

- Vehicle pose (in this case six B-splines)
- Landmark positions
- Camera extrinsics from the four fisheye cameras
- Accelerometer and gyroscope bias

Currently the steering wheel angle is not implemented as a design variable. The reason for this is given in chapter 3.2.

3.1.2 Error Terms

Error terms are basically the implementation of the modeled observation errors presented in chapter 2.2.4. Additionally, the ASLam backend requires the jacobian of the errors with respect to all design variables to be implemented analytically.

Defining analytical jacobian matrices is straight forward for the gyroscope, the landmark reprojection, the steering wheel angle and the wheelticks errors. To do so with the accelerometer and the transformation errors the following problems have to be solved.

To derive the analytical jacobian of the accelerometer observation model, the jacobian of the rotation matrix $\mathbf{C}(\phi(t))$ has to be calculated. Depending on the chosen rotation parametrization, this can be done at least element by element. However, finding an analytical jacobian for the chosen axis-angle representation was not possible in reasonable time. As including the accelerometer sensor had only a low priority, no further time was invested to solve this issue.

The analytical jacobian of the given transformation error with respect to the vehicle pose $\mathbf{x}$ is not trivial, but it can, at least for the axis-angle representation, be written as

$$\frac{\partial \mathbf{e}_{transf}}{\partial \mathbf{x}} = \begin{bmatrix} \mathbf{1} & \hat{\mathbf{r}}^\times \\ \mathbf{0} & \mathbf{S}(\hat{\phi})^{-1} \end{bmatrix} \quad (3.1)$$

with the prior transformation

$$\hat{\mathbf{T}} = \begin{bmatrix} \mathbf{C}(\hat{\phi}) & \hat{\mathbf{r}} \\ \mathbf{0} & 1 \end{bmatrix} \quad (3.2)$$

More details on this jacobian can be found in chapter 3.3 of [8]. The exact form of $\mathbf{S}(\hat{\phi})^{-1}$ is given on page 31 of [12].

3.1.3 Optimization Problem

The optimization problem is the container holding all design variables and error terms used for one batch estimation. In the current implementation, the optimization problem is completely rebuilt for each estimation.

3.2 Batch Estimator

The batch estimator is where an estimation is set up and run. It constantly collects observations. When a new estimation is started, it builds design variables, then error terms based on these observations and subsequently starts the estimation with a new optimization problem.

3.2.1 Observations

The following list shows the observations and their respective error terms.

Camera Extrinsic Prior If the camera extrinsics are estimated, for each camera a camera extrinsics prior is built by using its default extrinsics as prior transformation in the transformation error.

Gyroscope The yaw rate observations from the gyroscope are added with its corresponding error.

Integrated Odometry The integrated odometry observations are provided by VW. They combine raw odometry (wheelticks and steering wheel angle) with a car model to calculate a two dimensional relative vehicle position and a relative vehicle yaw angle. An integrated odometry measurement is included by building a transformation error which uses the difference transformation between two consecutive measurements as prior.

During slow vehicle motion, quantization effects from the wheelticks are visible in the integrated odometry (see figure 1.3). Depending on the set of error terms included in the estimation, this can lead to erroneous estimation results.¹ To avoid this, only observations with a translational difference above a certain threshold to the previous measurement are added.

This first batch estimator implementation uses the integrated odometry instead of the raw odometry observations to simplify the task. Furthermore, at this moment it is not clear how the estimation results are influenced by the usage of the raw odometry instead of the integrated odometry.

Landmark Reprojections All vehicle frames defined as key frames at the end of the visual odometry pipeline (see [4]) are added to the batch estimator. The landmarks and their belonging features in the images are then included in the optimization problem with landmark reprojection errors.

Pose Prior From the previous batch estimation, the estimated vehicle pose at the time which is the current beginning of the estimation window is extracted. This pose is then included as a prior transformation using the transformation error.

3.2.2 Estimation

Trigger The primary trigger for a new estimation is whenever a new key frame is added. As vehicle frames are built from fisheye camera images, which have an update frequency of 12.5 Hz, this gives an upper bound to the number of batch estimations which are run per second. However, the definition of a key frame asks for a minimal vehicle motion compared to the last key frame. Thus, if the car e.g. stands at a red light, no key frames are added and no batch estimation is started.

A lower bound to the number of batch estimations per second is set by defining a time threshold between estimations. This secondary trigger is checked after each run of the visual odometry pipeline, and results in a valid batch estimation even when standing still.

Sliding Window The previously mentioned sliding window approach is implemented by only considering observations from within n seconds before the current estimation was trig-

¹ When, during motion, two following integrated odometry observations have the same position, the estimated pose tries to match this phase with no motion.

gered. All observations before this window are deleted. This means that, while the amount of observations by type stays rather constant for most sensors, the fact that not all vehicle frames are added to the estimation problem leads to big changes in the number of error terms from landmark reprojections.²

Error Weighting The weighting matrices $\mathbf{W}$ of the error terms (see section 2.2.1) can be used as tuning parameters.

In transformation errors, the rotation weighting elements are about a factor 1000 smaller than the translation weights in order to give a good relation between the weight on a 1 m translational error compared to a 1 rad rotational error.

In general, all error weights are constant. The only implemented exception is the weighting of the landmark reprojection errors, where a *Huber M-Estimator* is applied to minimize the influence of outliers on the estimation results. More details on the effect of M-Estimators on landmark reprojection errors can be found in [4].

3.3 Issues

Several issues have come up during the implementation of the batch estimator. Here are the most important unresolved ones:

Analytical Jacobians Analytical jacobians are needed to define error terms within the ASLam backend. Deriving these jacobians in a clean way, if possible independent from the chosen rotation parametrization, can be difficult. The analytical jacobians for the vehicle rotation matrix $\mathbf{C}_{l,v}$ and the one for the body angular acceleration in the vehicle frame $\alpha_v^{v,l}$ are not yet implemented. Therefore, the acceleration observations are currently not used in the batch estimator.

Optimizer Boundaries In a preliminary implementation, the raw odometry instead of the integrated odometry was used. To prevent introducing an unnecessary state, the steering wheel angle was directly combined with the wheelticks. This led to the problem that, when extracting the steering wheel angle from the wheelticks,³ the optimizer accidentally pushed the estimate into invalid regions.⁴ At that point, there was no possibility to prevent this from happening. Thus, the integrated odometry is used instead of the raw odometry.

Processing Speed In the current implementation, the whole optimization problem, including all error terms and design variables, is completely rebuilt for every estimation.

² All sensors have a fairly constant update frequency, which gives a constant stream of measurements.

³ This was done to build an optimizable error term with the raw steering wheel angle observations.

⁴ E.g. the optimizer assumed that the steering angle was $> 90^\circ$.

Especially building design variables is computationally expensive. In fact it is so expensive that the whole batch estimator is far away from being real-time capable.⁵

However, a huge increase in speed can be achieved by keeping the optimization problem from the previous estimation, including all design variables and error terms. Then, by deleting only the part of the optimization problem which is older than the current estimation window, and by adding only the new, not yet included observations, the processing cost can be minimized.

⁵ Currently, a decrease in computation time of at least a factor 100 is needed.

Chapter 4

Results

The following pages present results from estimations with various parameter settings. All results were obtained from one at least three minute long part of a longer dataset recorded in Wolfsburg, Germany, in April 2012. The distance covered in these examples is at least one kilometer.

4.1 Parameter Settings

Some of the batch estimator parameters are identical for all presented estimations:

- Batch window length is one second
- Only one B-spline segment for the one second window
- The B-spline is of fourth order
- Landmark positions are estimated if the landmark is seen in at least two images.
- The heaviest weighted error term is the pose prior error

Varying parameters are the set of included error terms and therefore the used set of sensors, and the weighting factors of the error terms.

4.2 Batch Estimated Visual Odometry Pipeline

For the first results, the batch estimator uses only the landmark reprojections and the integrated odometry observations. Therefore, comparing these results with the ones from the visual odometry pipeline alone mainly shows the advantage of using longer batch estimation windows.¹

4.2.1 Overall Results

Figure 4.1 shows the estimated positions over the full example dataset. The GPS track is used as ground truth. Both the results of the visual odometry pipeline and the batch estimator are aligned with the GPS track over the first 200 meters.

Comparing both estimated tracks with the one from GPS shows that the batch estimator improves the pure visual odometry result. The influence of the integrated odometry, which gives a much better approximation when driving along a straight rather than a bent course, is visible. However, when estimating over a longer time window, the landmark tracks, which give a good estimate of the rotation, minimize the effects of the bad yaw rate estimate caused by the integrated odometry. Figure 4.2 shows the same effect, with the longitudinal errors being much smaller than the lateral errors.

¹ In this case a one second window compared to a 0.08 second window in the visual odometry pipeline.

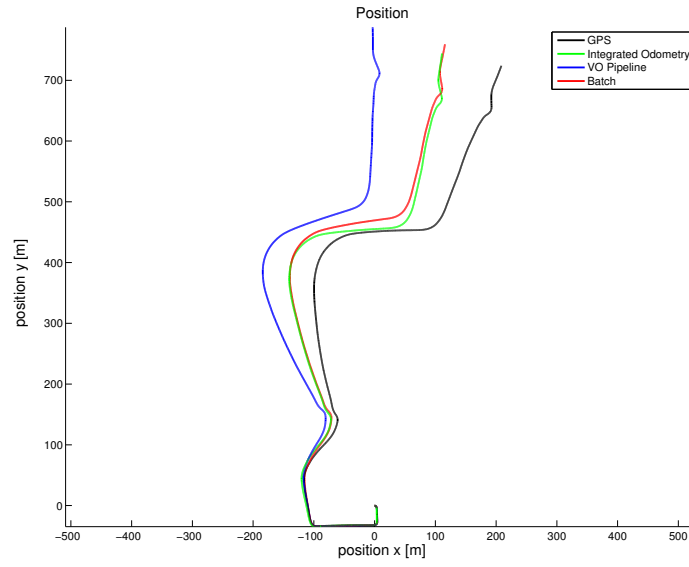


Figure 4.1: Position tracks from GPS, integrated odometry, visual odometry and batch estimator (with integrated odometry and landmark observations)

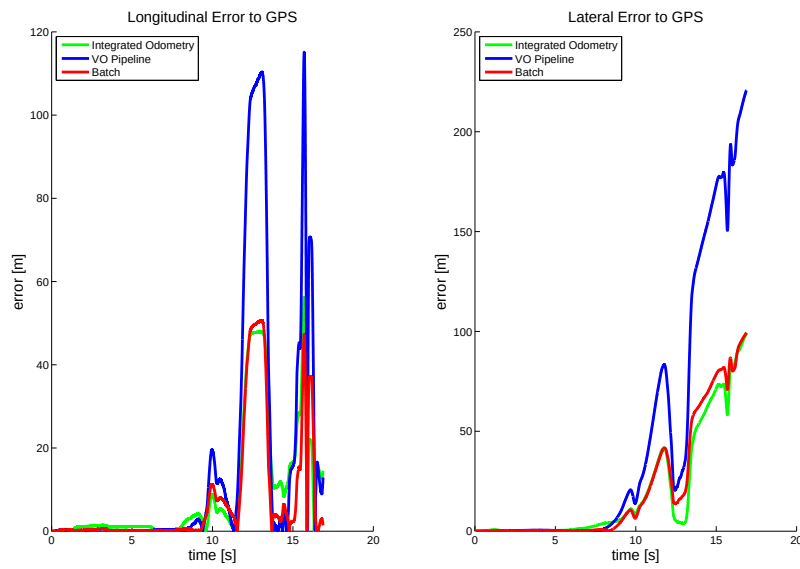


Figure 4.2: Absolute position errors from tracks in figure 4.1 with respect to GPS

4.2.2 Short Segment Results

For this segment covering about 10% of the full example, the first 20 m were used for alignment. Figure 4.3 shows that, in this segment, the batch estimator improves the result. Figure 4.4 shows that, because both estimates use the same observations, the effects from the bad integrated odometry are visible in both results.

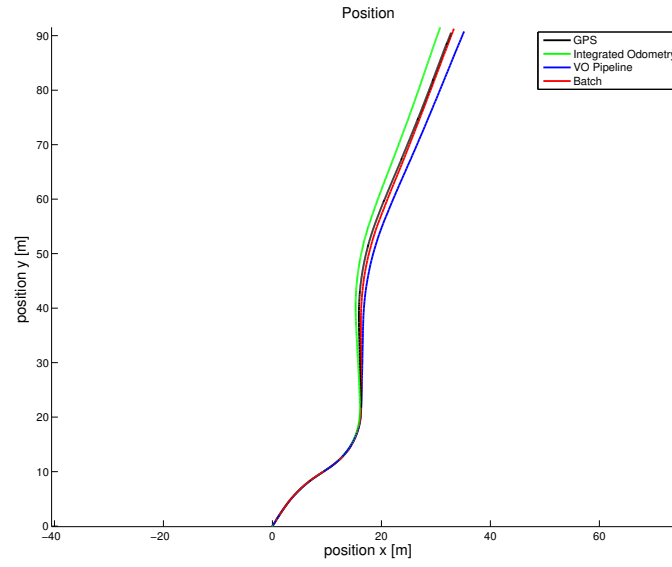


Figure 4.3: Segment of 100 m length from data presented in figure 4.1

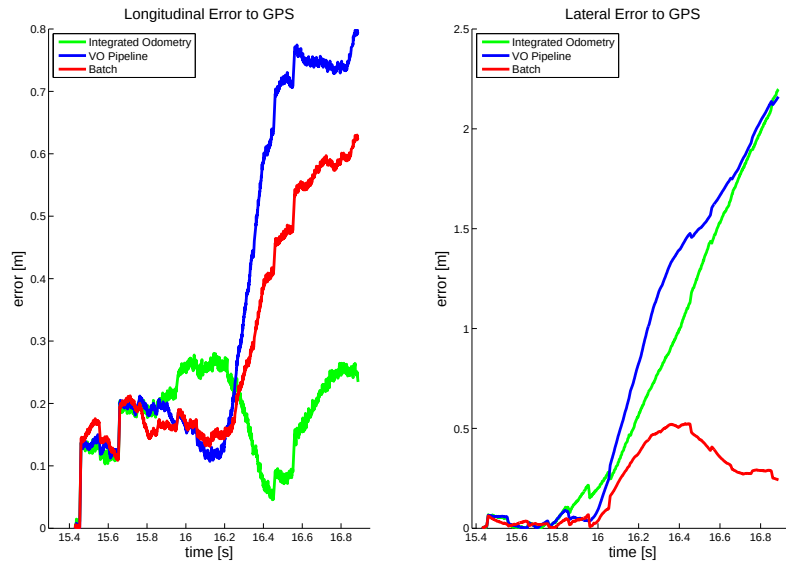


Figure 4.4: Absolute position errors from tracks in figure 4.3 with respect to GPS

4.3 Integrated Odometry, Gyroscope and Landmarks

The second batch estimator results show the estimates when including the yaw rate measurements from the gyroscope.

4.3.1 Overall Results

The tracks from the full example dataset (figures 4.5 and 4.6) show that the overall results from the batch estimator are better than the ones from the visual odometry pipeline. The dataset used for this example is the same as the one used in chapter 4.2, but starting after the second corner in figure 4.1 (at coordinates $(-100, -30)$). Again, errors from integrated odometry are visible in both the visual odometry and the batch estimator tracks.

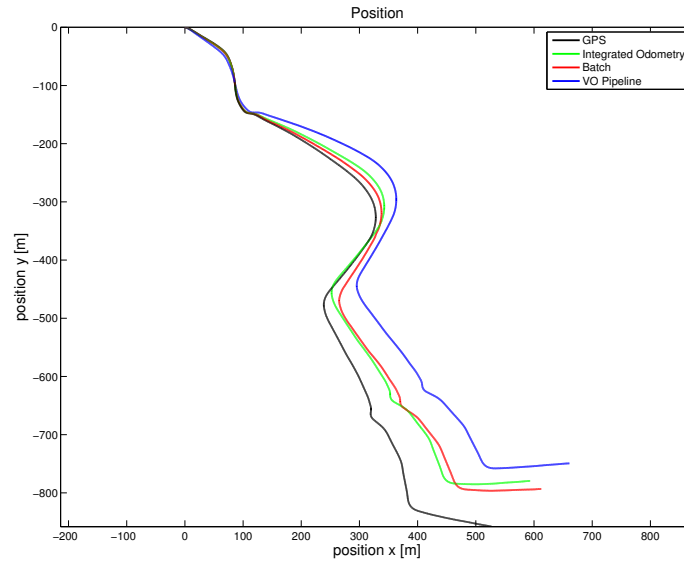


Figure 4.5: Position tracks from GPS, integrated odometry, visual odometry and batch estimator (with integrated odometry, gyroscope and landmark observations)

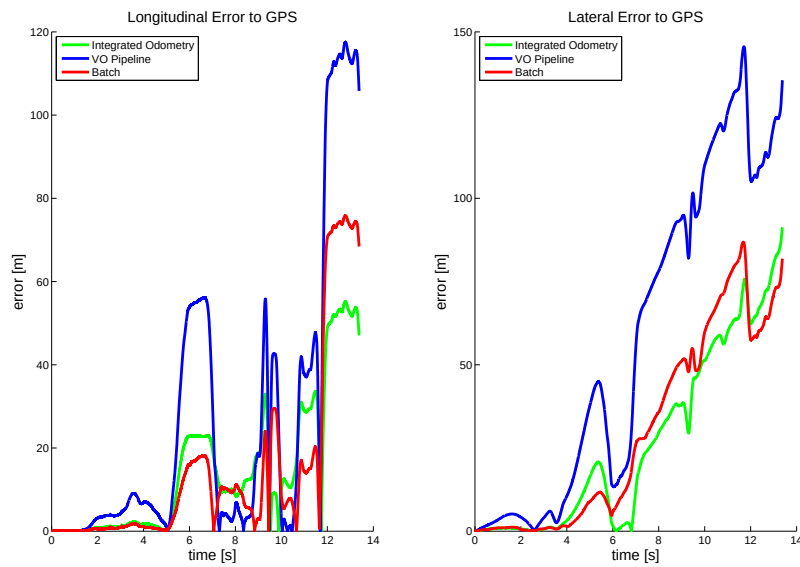


Figure 4.6: Absolute position errors from tracks in figure 4.5 with respect to GPS

4.3.2 Short Segment Results

A closer look at short segments from the same dataset shows that in some segments the batch estimator has a better performance than the visual odometry pipeline (figures 4.7 and 4.8), while in other segments performance is worse (figures 4.9 and 4.10). Most likely both the overall and the segment results can be improved by further tuning the weighting parameters of the used error terms. However, for the reasons given in the next section, this is not yet done.

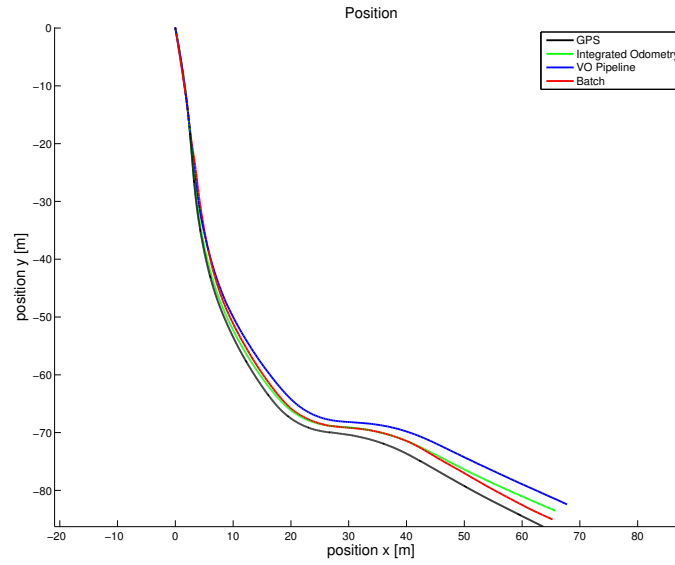


Figure 4.7: Segment from data presented in figure 4.5, in which the batch estimator improves the results

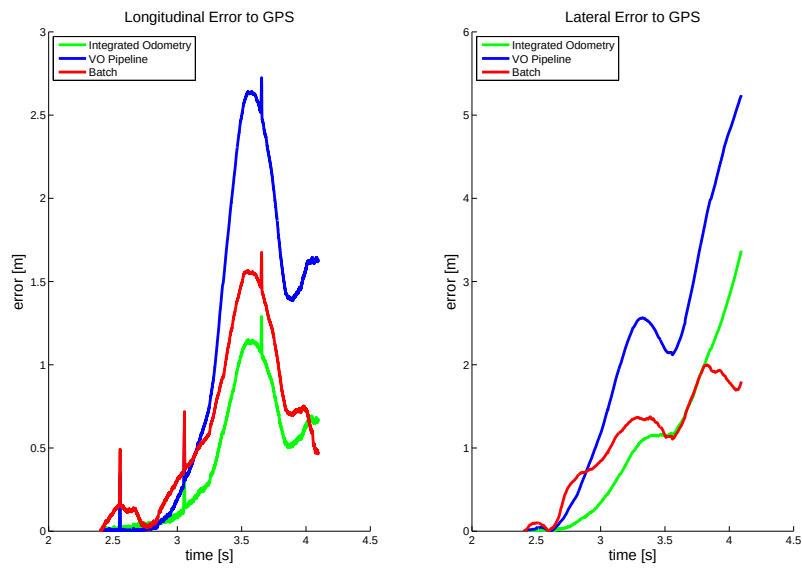


Figure 4.8: Absolute position errors from tracks in figure 4.7 with respect to GPS

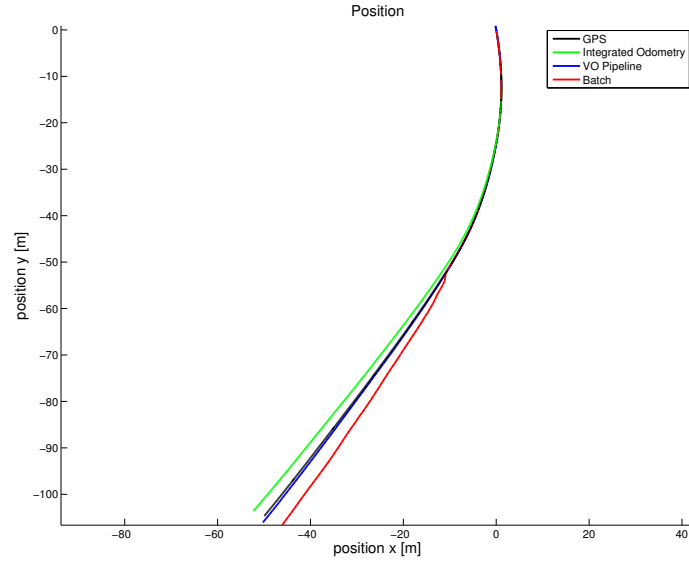


Figure 4.9: Segment from data presented in figure 4.5, in which the batch estimator results are worse than all other available position estimates

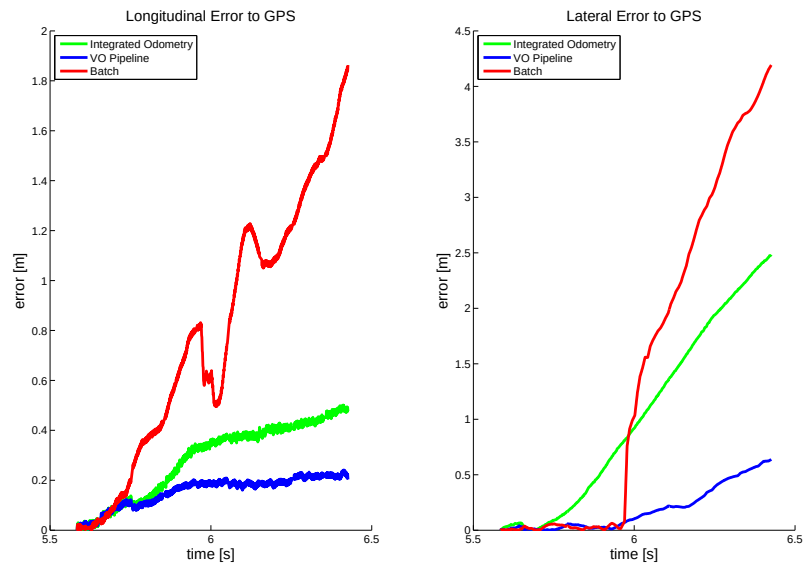


Figure 4.10: Absolute position errors from tracks in figure 4.9 with respect to GPS

4.4 Issues

Currently these are the only valid batch estimator results. The three main reasons for this limited performance are:

4.4.1 Quality of Estimation Results

The reason why the dataset in chapter 4.3 starts after the second corner of the example in chapter 4.2 is that sometimes the estimated results are extremely unlikely. In this corner, the batch estimator (with the same parameters as in chapter 4.3) estimates for each estimation window the positions shown in figure 4.11. It is obvious that, coming from the right side, at some point the optimizer starts to estimate a series of unlikely positions. Given the fact that, besides the offset, after a few estimations the results are again much better, it has to be assumed that this effect is caused by very few observations.

In an analysis of all observations influencing this situation, no measurement was conspicuous. Running the estimator only on this short segment with varying sets of the sensors used in chapter 4.3 shows that this effect is only reproducible with the exact same configuration, but not when using any subset of sensors. Thus this problem seems to base on the combination of multiple sensors and their error weights.

To prevent these unlikely estimates from being handled as true, a cross-validation of the estimated pose with the estimate from the visual odometry pipeline is done after each estimation. If this distance based cross-validation fails, the batch estimate is discarded and replaced with the visual odometry pipeline estimate. This error recovery helps in some critical situations, but not always.

4.4.2 Processing Speed

One of the issues mentioned in chapter 3.3 is the extremely slow processing speed. Analyzing a dataset of a three minutes drive takes up to four hours. This makes analyzing the estimator to understand critical situations and improving the results by tuning parameters an extremely time consuming task.

4.4.3 Integration within Localization Pipeline

The localization pipeline, and especially the visual odometry pipeline, are still under development. Adjustments on the algorithms used to detect features and to triangulate landmarks often change the quality and the average number of features included in the reprojection errors within the batch estimator. At the current stage, the batch estimator is highly sensible to such changes. Thus, changes made somewhere in the localization pipeline often have a big influence on the batch estimator results. Combined with the processing speed and the general estimation result quality issues, this means that improving the estimation results is very time consuming, and often the origin of an observed effect is not detectable at first.

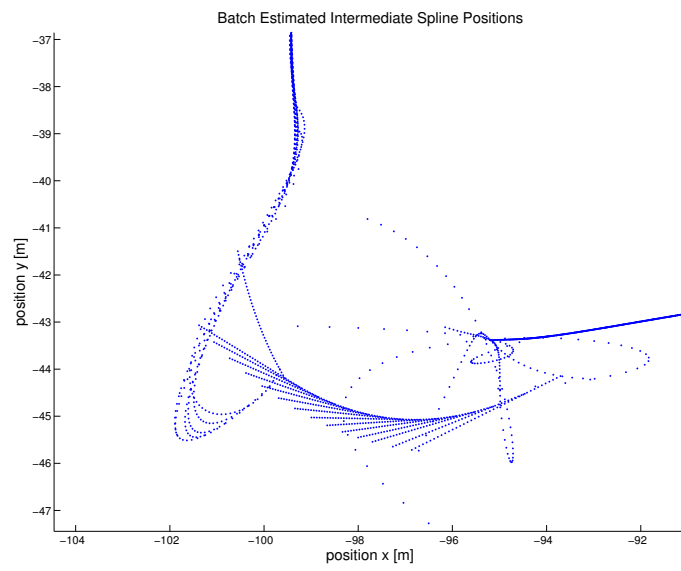


Figure 4.11: Estimated intermediate positions within batch windows

Chapter 5

Conclusion

The first version of a batch estimator presented in this thesis is a working proof of concept. Estimating the vehicle state in continuous time using B-splines is a promising approach. The goal of providing a basis which is flexible regarding the sensors used is met. However, the currently computational expensive setting takes the whole localization system far away from being real-time capable. This is also the main reason for the low number of currently presentable results.

An earlier implementation of the batch estimator had a partial solution to the speed problem. That batch estimator was able to delete only parts of an optimization problem, which means that only the newest observations had to be added. While this heavily decreased processing time, at that time this batch estimator was discarded because of unexplainable side effects. Later it was discovered that these effects had no connection to the way the optimization problem was handled. Thus, decreasing the processing time to get closer to a real time capable localization system is doable.

The next tasks after increasing the processing speed include a detailed analysis of the unlikely estimation results, the implementation of further sensors and parameter tuning on long datasets. The first version of the batch estimator presented in this thesis is a good basis for all these tasks.

Bibliography

- [1] <http://www.v-charge.eu>, October 2012.
- [2] <http://www.asl.ethz.ch>, October 2012.
- [3] http://en.wikipedia.org/wiki/Google_driverless_car, October 2012.
- [4] M Neunert. Online Localization for an Autonomous Car Using Multiple Cameras. Master's thesis, ASL, ETH Zürich, 2012.
- [5] P T Furgale, T D Barfoot, and G Sibley. Continuous-time batch estimation using temporal basis functions. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2088–2095, St. Paul, MN, 14-18 May 2012.
- [6] R H Bartels, J C Beatty, and B A Barsky. *An Introduction to Splines for Use in Computer Graphics And Geometric Modeling*. Morgan Kaufmann, 1987.
- [7] P T Furgale. State estimation for robotics using b-splines. Unpublished note.
- [8] P T Furgale. *Extensions to the Visual Odometry Pipeline for the Exploration of Planetary Surfaces*. PhD thesis, University of Toronto Institute for Aerospace Studies, 2011.
- [9] A H Jazwinski. *Stochastic Processes and Filtering Theory*. Academic Press, Inc, New York, USA, 1970.
- [10] B Triggs, P McLauchlan, R Hartley, and A Fitzgibbon. *Bundle Adjustment - A Modern Synthesis*. 2000.
- [11] K Konolige, M Agrawal, and J Solà. *Large Scale Visual Odometry for Rough Terrain*. 2007.
- [12] P C Hughes. *Spacecraft Attitude Dynamics*. John Wiley & Sons, 1986.

E A Patch Prior for Dense 3D Reconstruction in Man-Made Environments

Authors: Haene, Zach, Zeisl, Pollefeys

Link to [Table of Contents](#).

A Patch Prior for Dense 3D Reconstruction in Man-Made Environments

Christian Häne¹ Christopher Zach² Bernhard Zeisl¹ Marc Pollefeys¹

ETH Zürich¹
Switzerland

{chaene, zeislb, pomarc}@inf.ethz.ch

Microsoft Research²
Cambridge, UK

chzach@microsoft.com

Abstract—Dense 3D reconstruction in man-made environments has to contend with weak and ambiguous observations due to texture-less surfaces which are predominant in such environments. This challenging task calls for strong, domain-specific priors. These are usually modeled via regularization or smoothness assumptions. Generic smoothness priors, e.g. total variation are often not sufficient to produce convincing results. Consequently, we propose a more powerful prior directly modeling the expected local surface-structure, without the need to utilize expensive methods such as higher-order MRFs. Our approach is inspired by patch-based representations used in image processing. In contrast to the over-complete dictionaries used e.g. for sparse representations our patch dictionary is much smaller. The proposed energy can be optimized by utilizing an efficient first-order primal dual algorithm. Our formulation is in particular very natural to model priors on the 3D structure of man-made environments. We demonstrate the applicability of our prior on synthetic data and on real data, where we recover dense, piece-wise planar 3D models using stereo and fusion of multiple depth images.

I. INTRODUCTION

Dense 3D modeling from images often suffers from a lack of strong matching costs, especially in man-made environments. Such environments usually exhibit texture-less surfaces and also non-Lambertian ones violating underlying, e.g. brightness constancy assumptions. The presence of weak data terms must be compensated by strong model priors in order to obtain plausible 3D reconstructions.

In this work we propose to utilize a spatial regularization prior for depth maps inspired by sparse representations. While our energy clearly resembles the dictionary-based energy functionals employed in image processing formulations (e.g. [1]), there are important differences due to the different characteristics of image (intensity) data and depth maps. Most prominently, depth maps representing man-made environments are typically comprised of very few structural elements, but the specific sparsity assumption used for processing intensity images is not necessarily appropriate for depth maps. Thus, the over-complete dictionary used for sparse representations can be substituted by a small one, replacing the sparseness assumption on dictionary coefficients by different priors. One benefit of this reduced model is the significant reduction in the number of unknowns, therefore increasing the efficiency of numerical optimization.

Although not being the focus of this work, we anticipate the possibility of extracting the dictionary elements and the statistical prior on the coefficients from training data.

The smoothness prior proposed in this work shares the main motivation with other methods aiming on regularization terms beyond binary potentials. All of these methods are global optimization methods, which are required in order to properly incorporate the spatial smoothness assumptions on the desired solution (i.e. depth map). The most generic (and also the computationally most demanding) approach to handle arbitrary smoothness priors is the formulation using higher-order Markov random fields. [2] proposes to explicitly use second-order regularization priors for stereo in order to obtain smooth surfaces without the typical staircasing effects. Ishikawa and Geiger [3] argue that the human vision system is contrary from current smoothness models; they use second-order priors to penalize large second derivatives of depth and thereby better reflect the behavior of human vision.

Segmentation based stereo approximates pixel-wise higher-order priors by enforcing the constraint that segmented regions of the image should describe smooth 3D surfaces. Often not a general smoothness prior, but a hard constraint is applied, preventing curved surfaces; e.g. [4] uses graph cuts to optimize in the segment domain. Contrary to that is the explicit minimization of a second order prior via alternating, iterative optimization over surface shape and segmentation: Birchfield and Tomasi [5] utilize an affine model to account for slanted surfaces, whereas Lin and Tomasi [6] introduce a spline model to cover more general curved and smooth surfaces. Finally [7] groups pixels of similar appearance onto the same smooth 3D surface (planes or B-splines) and regularizes the number of surfaces, inherently employing soft-segmentation. A parametric formulation for optical flow explicitly aiming on favoring piecewise planar results (instead of generally smooth, or piecewise constant flow fields) is presented in [8]. A recently proposed generalized notion of total variation [9] is also applied to merge several depth maps [10]. This model can be seen as a special case of our formulation.

The contribution of our work is based on the introduction of *patch-based* priors, modeling higher-order priors by

means of a dictionary-based approach. The motivation for this patch based regularization is twofold. First, as dictionaries can be rather small, inference is fast; and second, computational complexity does not increase considerably with the usage of larger patches, allowing for regularization beyond triple cliques and second-order priors. In the following we present the corresponding energy formulation and concentrate on explaining the different choices for the data fidelity term, the dictionary and coefficient priors (Section II). An efficient inference framework is described in Section III to solve for depth maps and dictionary coefficients. Fusion of several noisy depth maps using the patch-based prior is detailed in Section IV, and a computational stereo approach is presented in Section V. Finally, Section VI concludes with a summarizing and prospective discussion of our approach.

II. PROPOSED FORMULATION

This section describes the basic energy functional used in our approach and discusses the relation to existing methods. Let Ω be the image domain (usually a 2D rectangular grid), and $\phi_{\mathbf{p}}(\cdot)$ is a family of functions for $\mathbf{p} \in \Omega$ modeling the data fidelity at pixel $\mathbf{p}$. $\phi_{\mathbf{p}}$ is assumed to be convex. Further, let $R_{\mathbf{p}} : \mathbb{R}^{\Omega} \rightarrow \mathbb{R}^N$, be a function extracting an image patch centered at $\mathbf{p}$. Since we allow different shapes for the extracted patches (and therefore several different functions $R_{\mathbf{p}}$), we will use another index k to indicate the shape geometry, i.e. $R_{\mathbf{p}}^k$. By allowing different shape geometries for different dictionary sets, the size of the dictionary can be reduced substantially by not using a e.g. square patch uniformly. We utilize the following energy model:

$$E(u, \alpha) = \int \phi_{\mathbf{p}}(u_{\mathbf{p}}) + \eta \sum_k \|R_{\mathbf{p}}^k u - \mathcal{D}^k \alpha_{\mathbf{p}}^k\| + \psi(\nabla \alpha_{\mathbf{p}}) d\mathbf{p}, \quad (1)$$

where u is the desired depth (respectively disparity) map, $\alpha^k : \Omega \rightarrow \mathbb{R}^{|\mathcal{D}^k|}$ are the coefficients for the dictionary $\mathcal{D}^k$, η and μ are positive constants, and $\psi(\cdot)$ is a convex function. The individual terms have the following interpretation:

- 1) The first term, $\phi_{\mathbf{p}}(u_{\mathbf{p}})$, is the data fidelity term at pixel $\mathbf{p}$, and measures the agreement of u with the observed data.
- 2) The second term, $\sum_k \|R_{\mathbf{p}}^k u - \mathcal{D}^k \alpha_{\mathbf{p}}^k\|$, penalizes deviations of u from a pure dictionary generated patch in a region $R_{\mathbf{p}}^k$ containing $\mathbf{p}$. As distance measure we use either the L_1 norm $\|\cdot\|_1$, or the L_2 (Euclidean) norm, $\|\cdot\|_2$. The choice of $\|\cdot\|_1$ allows u to locally deviate from the predicted patch $\mathcal{D}^k \alpha_{\mathbf{p}}^k$ at a sparse set of pixels, whereas selecting $\|\cdot\|_2$ resembles group Lasso [11] leading to a sparse set of patches in disagreement with $\mathcal{D}^k \alpha_{\mathbf{p}}^k$.
- 3) The last term, $\psi(\nabla \alpha_{\mathbf{p}})$, adds spatial regularization on the coefficients α . Choosing $\psi(\nabla \alpha_{\mathbf{p}}) = \mu \|\nabla \alpha_{\mathbf{p}}\|$ (for some positive weight μ) leads to piece-wise constant solutions for α .

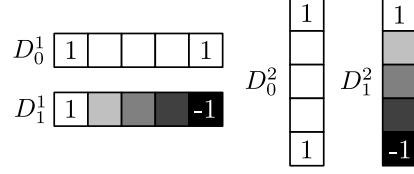


Figure 1. Piecewise planar dictionary elements of length 5.

A. Choice for $\mathcal{D}^k$

The dictionary $\mathcal{D}^k$ determines how the solution should look like locally. It could be acquired by learning it from given training data. By including a sparsity term on the coefficients α^k it would even be possible to use over-complete dictionaries. As this work focuses on the reconstruction of man-made environments we define the dictionary as follows. Man-made environments, and in particular indoor ones, are dominated by locally planar surfaces. Hence it is natural to strongly favor piece-wise planar depth (or disparity) maps on the absence of strong image observations. For the pinhole camera model, it is easy to see that planar surfaces in 3D correspond to locally linear disparity maps (i.e. linear in $1/\text{depth}$ but not in the depth itself) Consequently, although we use the terms depth map and disparity map equivalently, the smoothness prior is always applied on the disparity representation. For piecewise planar disparity maps the utilized dictionary is very compact and contains only four elements (see Fig. 1),

$$\begin{aligned} D_0^1 &= \mathbf{1}^T & D_1^1 &= \left(\frac{2i}{P_{\text{length}} - 1} - 1 \right)_{i=0}^{P_{\text{length}}-1} \\ D_0^2 &= \mathbf{1} & D_1^2 &= (D_1^1)^T, \end{aligned} \quad (2)$$

i.e. D_1^1 and D_1^2 are the horizontal and vertical linear gradients, respectively. The coefficients corresponding to the constant elements D_0^1 and D_0^2 , α_0^1 and α_0^2 , essentially cover the absolute disparity, whereas α_1^1 and α_1^2 represent the local slope in horizontal and vertical image direction. P_{length} denotes the length of the patch.

B. The Choice $\psi(\cdot)$

In order to favor piecewise planar (but not piecewise constant) disparity maps piece-wise constant solutions for the coefficients of the linear gradient elements should be preferred. As we do not want to penalize the actual disparity there is no regularization on the coefficients belonging to the constant dictionary elements. Piece-wise constant solutions can be favored by using the total variation as a penalization:

$$\psi(\nabla \alpha_{\mathbf{p}}^k) = \mu \|\nabla (\alpha_{\mathbf{p}}^k)_2\| \quad (3)$$

We either use the isotropic L_2 total variation or the anisotropic L_1 total variation.

C. Choices for $\phi(\cdot)$

The family of functions $\phi_{\mathbf{p}}$ represents the fidelity of the solution u to the observed data at pixel $\mathbf{p}$. We discuss two important choices for $\phi_{\mathbf{p}}$.

In the application of merging multiple depth maps, a varying number of depth measurements (including none) is given for each pixel. Hence, the choice of $\phi_{\mathbf{p}}$ is reflecting the underlying noise model in the input depth maps and penalizes the distance to the given depth measurements $u_{\mathbf{p}}^l$. Depth estimates originating from visual information are often subject to quantization, therefore we use a “capped” L_1 -norm to allow deviations within the quantization level. To combine multiple measurements the individual distances are summed up:

$$\phi_{\mathbf{p}}(u_{\mathbf{p}}) = \sum_l \lambda_{\mathbf{p}}^l \max\{0, |u_{\mathbf{p}} - u_{\mathbf{p}}^l| - \delta\}, \quad (4)$$

where $\lambda_{\mathbf{p}}^l$ is a weight that can be specified for each depth measurement. This enables down-weighting of measurements with high matching costs or non-unique matches. This particular choice of $\phi_{\mathbf{p}}$ implicitly fills in missing depth values by setting all weights of missing values to zero.

Another choice for $\phi_{\mathbf{p}}$ addresses the task of computing depth/disparity maps between images. $\phi_{\mathbf{p}}$ may directly measure the similarity of pixel intensities in the reference and matching images, I_0 and I_1 , respectively:

$$\phi_{\mathbf{p}}(u_{\mathbf{p}}) = \lambda |I_1(u_{\mathbf{p}}) - I_0|, \quad (5)$$

where λ is the weight on the data term. Since with this choice $\phi_{\mathbf{p}}$ is not convex, the definition above is usually replaced by its first order approximation with respect to the linearization point u^0 ,

$$\phi_{\mathbf{p}}(u_{\mathbf{p}}) = \lambda |I_1(u_{\mathbf{p}}^0) + (u_{\mathbf{p}} - u_{\mathbf{p}}^0) \nabla_u I_1 - I_0|. \quad (6)$$

More generally, any image similarity function can be evaluated locally in the neighborhood of u^0 and its convex surrogate utilized for $\phi_{\mathbf{p}}$ (e.g. the second order approximation of matching costs proposed in [12]). By using only a local approximation of the matching score function, the numerical procedure to find a minimizer u needs to be embedded into a coarse-to-fine framework (or alternatively, an annealing method like [13]).

III. DETERMINING u AND α

This section addresses the determination of the unknown disparity map u and corresponding coefficients α for a given dictionary and a coefficient prior. Although the functional in Eq. 1 is convex with respect to the unknowns u and α , it requires minimizing a non-smooth energy. There is a large set of methods for optimizing non-smooth convex

problems with additive structure, and many of those contain the proximity operator $\text{prox}_{\sigma f}$,

$$\text{prox}_{\sigma f}(\hat{x}) = \arg \min_x \frac{1}{2\sigma} \|x - \hat{x}\|^2 + f(x) \quad (7)$$

for a convex function f , as a central element. Due to its algorithmic simplicity we chose the primal-dual method proposed in [14], which can be intuitively described as combined gradient descent (in the primal domain) and ascent (in the dual domain) followed by suitable proximity operations. Eq. 1 can be written as convex-concave saddle-point problem (recall that the energy is minimized with respect to u and α),

$$\begin{aligned} E(u, \alpha) &= \int \phi_{\mathbf{p}}(u_{\mathbf{p}}) + \eta \sum_k \|R_{\mathbf{p}}^k u - \mathcal{D}^k \alpha_{\mathbf{p}}^k\|_1 \\ &\quad + \mu \|\nabla \alpha_{\mathbf{p}}\|_2 d\mathbf{p} \\ &= \max_{\mathbf{q}, \mathbf{r}} \int \phi_{\mathbf{p}}(u_{\mathbf{p}}) + \sum_k \mathbf{q}_k^T (R_{\mathbf{p}}^k u - \mathcal{D}^k \alpha_{\mathbf{p}}^k) \\ &\quad + \mathbf{r}^T \nabla \alpha_{\mathbf{p}} d\mathbf{p} \\ &\text{subject to } \|\mathbf{q}\|_{\infty} \leq \eta, \quad \|\mathbf{r}\|_2 \leq \mu. \end{aligned} \quad (8)$$

Here the constraints on $\mathbf{q}$ and $\mathbf{r}$ are formulated for the L_1 norm on the reconstruction error in the primal, and for the isotropic total variation regularizer for α . By choosing different norms the constraints change accordingly. The primal-dual method requires the application of respective proximity operations for the constraints on the dual variables, $\|\mathbf{q}\|_{\infty} \leq \eta$ and $\|\mathbf{r}\|_2 \leq \mu$, which are projection steps into the corresponding domain (i.e. element-wise clamping to $[-\eta, \eta]$ and length normalization whenever $\|\mathbf{r}\|_2 > \mu$, respectively).

This defines the general framework for using the primal-dual method, and the implications of the different choices of ϕ on the optimization method are discussed in the following section.

IV. PIECEWISE PLANAR DEPTH MAP FUSION

Our depth map fusion takes multiple nearby depth maps and combines them in a robust and regularized way to achieve higher quality. One of the input depth maps acts as a reference view, and all other depth images are warped to the reference view-point (using OpenGL-based mesh rendering). Thus, the task is to recover a single depth map $u_{\mathbf{p}}$ from multiple depth measurements $u_{\mathbf{p}}^l$ per pixel. As mentioned in Section II-C we utilize a “capped” L_1 distance accounting for the discrete/quantized nature of depth values. We introduce additional dual variables $\mathbf{s}$, and rewrite the

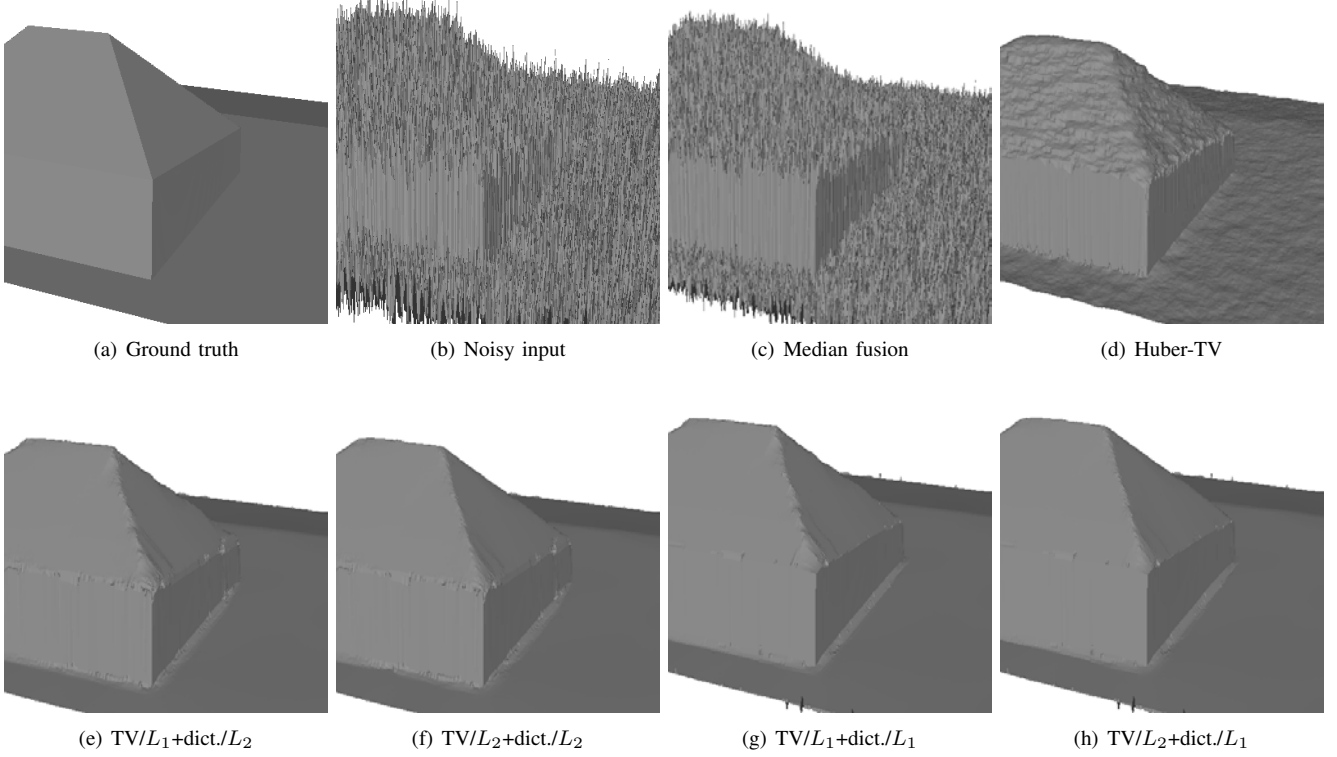


Figure 2. Top row: synthetic ground truth, one out of 5 noisy inputs, median fusion, Huber-TV fusion. Bottom row: results with the proposed piecewise planar prior with different norms for the dictionary and TV-term. From left to right: TV/ L_1 + dictionary/ L_2 , TV/ L_2 + dictionary/ L_2 , TV/ L_1 + dictionary/ L_1 , TV/ L_2 + dictionary/ L_1 .

primal energy into a saddle-point one and obtain

$$\begin{aligned}
 E(u, \alpha) &= \int \sum_l \lambda_{\mathbf{p}}^l \max\{0, |u_{\mathbf{p}} - u_{\mathbf{p}}^l| - \delta\} \\
 &\quad + \eta \sum_k \|R_{\mathbf{p}}^k u - \mathcal{D}^k \alpha_{\mathbf{p}}^k\|_1 + \mu \|\nabla \alpha_{\mathbf{p}}\|_2 d\mathbf{p} \\
 &= \max_{\mathbf{q}, \mathbf{r}, \mathbf{s}} \int \sum_l s_{\mathbf{p}}^l (u_{\mathbf{p}} - u_{\mathbf{p}}^l) + |s_{\mathbf{p}}^l| \delta \\
 &\quad + \sum_k \mathbf{q}_k^T (R_{\mathbf{p}}^k u - \mathcal{D}^k \alpha_{\mathbf{p}}^k) + \mathbf{r}^T \nabla \alpha_{\mathbf{p}} d\mathbf{p} \quad (9) \\
 \text{s. t. } &\|\mathbf{q}\|_{\infty} \leq \eta, \quad \|\mathbf{r}\|_2 \leq \mu, \quad |s_{\mathbf{p}}^l| \leq \lambda_{\mathbf{p}}^l.
 \end{aligned}$$

The proximity operator for the additional constraint is again a projection to the feasible set for $s_{\mathbf{p}}^l$. It is possible to compute the proximity operator for the above data term ϕ directly (without introducing additional dual variables), but this is relatively expensive (see e.g. [15]) and hinders data-parallel implementations.

A. Synthetic Data

We evaluate the behavior of the proposed fusion method for different choices of L_1/L_2 -norms in the reconstruction and the TV terms by using a synthetic height-map of a man-made scene (Fig. 2(a)). The input to the fusion is generated by adding zero-mean Gaussian noise with $\sigma = 0.8$

to the ground-truth height-map, which is in the range $[0, 10]$ (see Fig. 2(b) for one of the noisy input depths). Simply taking the pixel-wise median is clearly not sufficient to return a convincing result (Fig. 2(c)), hence enforcing spatial smoothness is required. Adding a smoothness prior via Huber-TV [10] (i.e. enforcing homogeneous regularization for small depth variations and a total variation prior at large depth discontinuities) still results in staircasing artefacts (Fig. 2(d)).

Figs. 2(e-h) depict the results for our proposed method using different combinations of L_1 and L_2 penalizers for the reconstruction error, $\|R_{\mathbf{p}}^k u - \mathcal{D}^k \alpha_{\mathbf{p}}^k\|$, and for the TV regularizer, $\|\nabla \alpha_{\mathbf{p}}\|$. The parameters were chosen as follows: the patch width was fixed to 5, $\lambda_{\mathbf{p}}^l = 1.5$ and $\mu = 10$, to adapt to the different penalization of the reconstruction error, we set $\eta = 1$ for L_1 penalization and $\eta = 1.5$ for L_2 penalization. For the datacost we used $\delta = 0$, which means L_1 distance penalization. In general, all results look rather similar. There are some artifacts at the hip of the roof when using a TV- L_1 penalization for the dictionary coefficients $\alpha_{\mathbf{p}}$. An L_2 -norm penalization in the dictionary term slightly cuts-off the edges at the eaves of the roof. The combination of an L_1 -norm in the dictionary term and an isotropic L_2 -norm total variation penalization of the dictionary coefficients visually gives the best solutions. In

the remainder of the document we only use this way of penalization.

B. Real-World Data

For our experiments with real-world data we took datasets with 25 images each¹. To obtain the camera poses we run a publicly available SfM software [16]. Input depth maps are obtained by running plane sweep stereo on 5 input images using a 3×3 ZNCC matching score and best half-sequence selection for occlusion handling. We use semi-global matching [17] to extract depth maps from the ZNCC cost volume, thereby obtaining five depth maps used for subsequent fusion. The depth maps are warped to the reference view by meshing and rendering the depth maps. Implausible triangles corresponding to huge depth discontinuities (i.e. most likely occlusion boundaries) are culled. Further, warped depth values with a corresponding ZNCC matching score smaller than 0.4 are also discarded. For the fusion parameters we used the same settings for all datasets. The patch width was set to 3, $\eta = 1$, $\mu = 5$, $\lambda_p^l = 0.3$ and $\delta = 0.015$. For the Huber-TV fusion we used the same “capped”- L_1 datacost also with $\delta = 0.015$ and the Huber parameter was set to 0.015 as well. In Figs. 3 and 4 we show results for piece-wise planar fusion and Huber-TV fusion on the same input data. Although the Huber-TV fusion aims on reducing the staircasing effect of the standard TV, there are still visually distracting artifacts in the rendered 3D-Model. When utilizing the proposed piece-wise planar structure prior the rendered 3D-Model is visually much more appealing especially when rendered with texture.

Additional results and a video showing the 3D models can be found in the supplementary material.

V. PIECE-WISE PLANAR DEPTH FROM STEREO

We can directly incorporate an image matching function (respectively a convex surrogate) for the data fidelity term ϕ_p in Eq. 1. We use the L^1 distance between Sobel-filtered image patches as basic matching costs (as suggested in [18], but we utilize only a 3-by-3 window instead of the suggested 9-by-9 one to avoid over-smoothing), and convexify the sampled matching costs using a quadratic approximation as proposed in [12]. The necessary proximal step for this choice of ϕ is given by

$$\text{prox}_\phi(u) = \frac{u + \lambda \ddot{c} u^0 - \lambda \dot{c}}{1 + \lambda \ddot{c}}, \quad (10)$$

where u^0 is the current linearization point used to sample the matching cost c , and $\dot{c}$ and $\ddot{c}$ are the first- and second-order derivatives of c with respect to disparity changes, computed via finite differences. Since the true matching cost approximation is only valid in a neighborhood of u^0 , we explicitly add the box constraint $|u - u^0| \leq 1$ to limit the range of disparity updates to 1 pixel. Adding this constraint

to ϕ means that the proximal operator above is followed by a clamping step to force u to be in $[u^0 - 1, u^0 + 1]$.

Further, the numerical procedure has to be embedded into a coarse-to-fine framework, with optional multiple cost sampling (i.e. image warping) steps per pyramid level. Without a suitable initialization this comes at the risk of missing small structures in the final disparity map (which is an intrinsic problem of all multi-scale methods).

Figs. 5 and 6 illustrate the difference between stereo with a total variation smoothness prior and the piecewise planar prior using the proposed formulation. The weighting between the data fidelity and the smoothness term in the TV model is selected, such that the results of both formulations are visually similar. As expected, using the TV regularizer leads to visible staircasing, in particular in textureless regions, which can be overcome by using the patch-based prior. Using 8 CPU cores the stereo approach generates usable depth maps for 384×288 images at 5 Hz, and more than 7 Hz can be achieved for 512×384 images using a GPU implementation (measured on an NVidia GTX 295). Consequently, it is conceivable e.g. to enable live and dense reconstruction for challenging indoor environments in the spirit of [19], [20].

VI. CONCLUSION AND FUTURE WORK

In this work we present an energy formulation for depth map recovery utilizing a patch-based prior, and apply the proposed framework to depth map fusion and computational stereo. We describe an efficient method to infer depth from given image data. One major difference of our approach to second order regularization like [8], [2] is, that our formulation is able to consider much larger neighborhoods, without the computational drawbacks of higher-order MRFs with large clique sizes.

The focus of this work was on the model formulation and on the inference step to obtain depth maps, with the assumption that the patch dictionary and the priors on dictionary coefficients are known. Learning patch elements and coefficient priors from training data is subject of future work. Beside replacing the manual design of dictionaries by an automated training phase, linking dictionary elements with appearance based classifiers for category detection is an interesting future topic. Joint optimization for depth and semantic segmentation is recently addressed in [21], where absolute depth and object categories are directly combined. Linking the local depth structure (i.e. the dictionary coefficients) with object segmentation seems to be a more powerful approach.

ACKNOWLEDGMENT

This work was supported in parts by the European Community’s Seventh Framework Programme (FP7/2007-2013) under grant n.269916 (V-Charge).

¹Datasets are available at <http://people.inf.ethz.ch/chaene/>

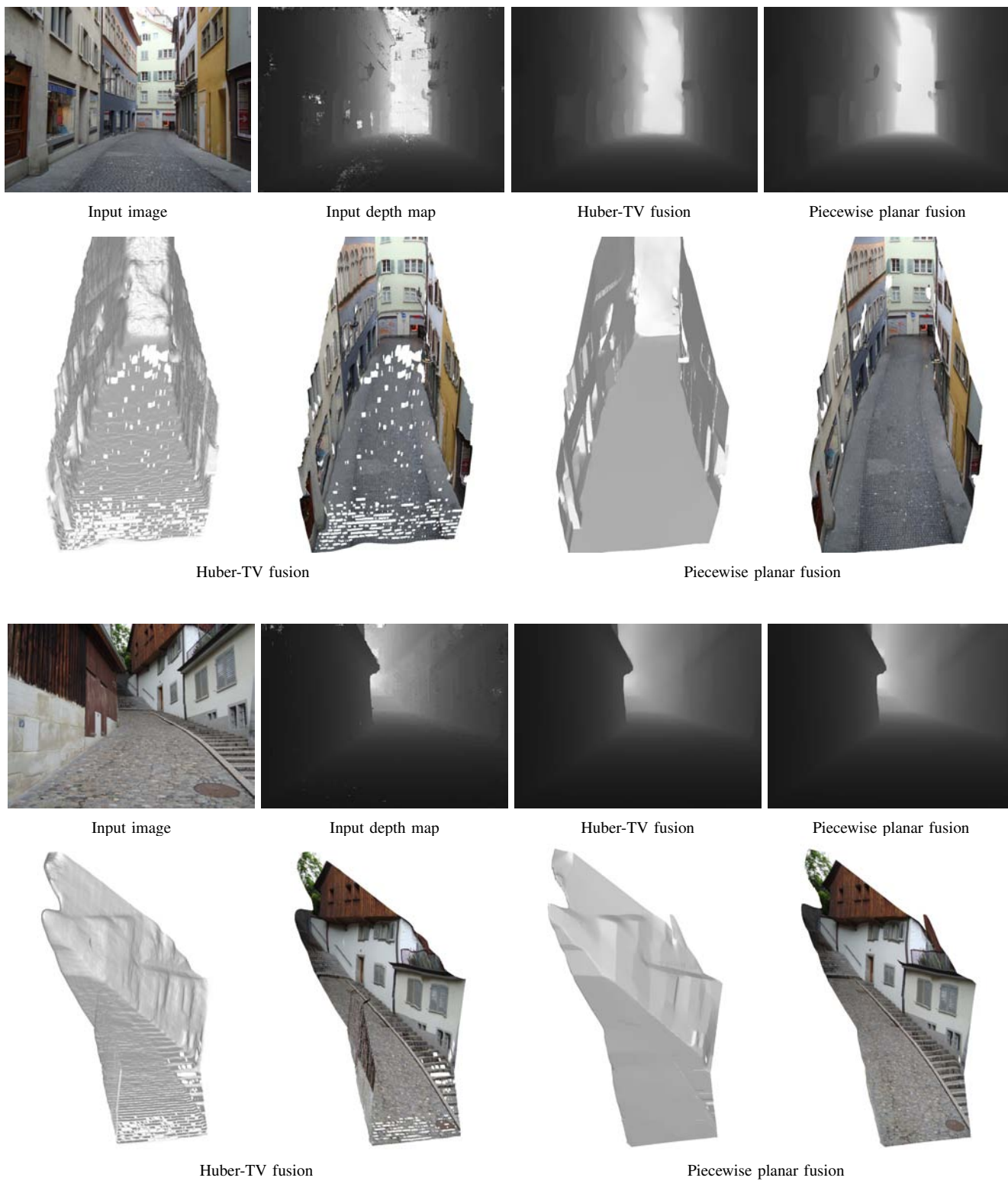


Figure 3. Outdoor depth map fusion results. Odd rows: One out of 25 input images, one out of 5 generated input depth maps, depth map from Huber-TV fusion, depth map with proposed piece-wise planar prior. Even rows: Untextured and textured 3D-Model. Left from Huber-TV fusion and right with piecewise planar prior.

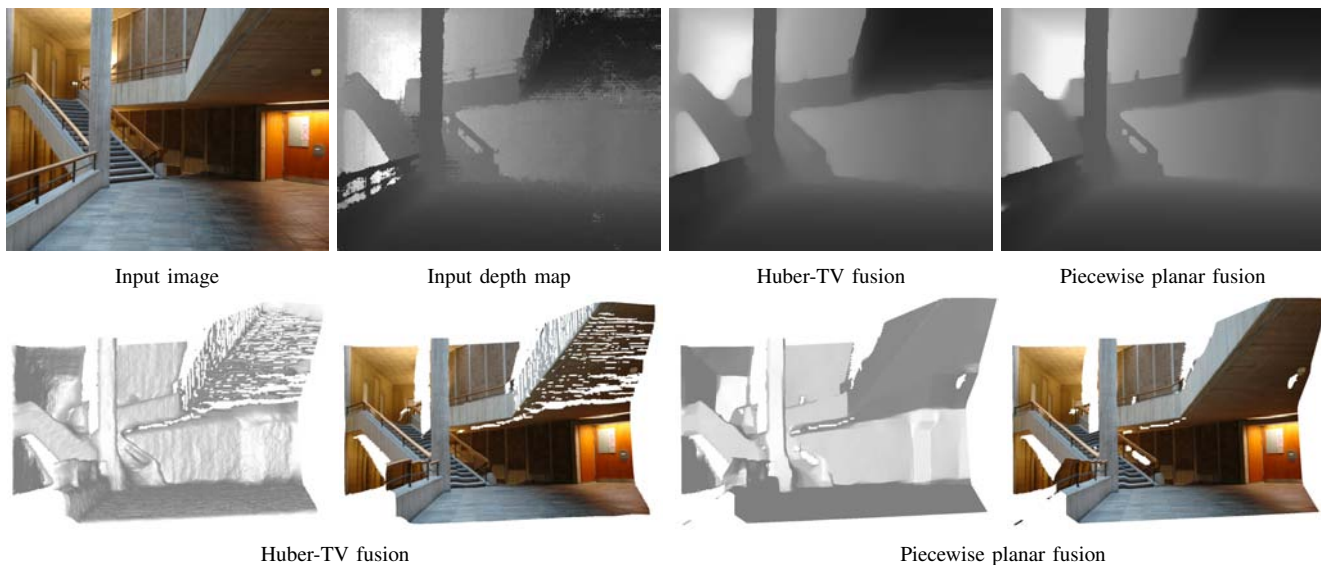


Figure 4. Indoor depth map fusion results. First row: One out of 25 input images, one out of 5 generated input depth maps, depth map from Huber-TV fusion, depth map with proposed piece-wise planar prior. Second row: Untextured and textured 3D-Model. Left from Huber-TV fusion and right with piecewise planar prior.

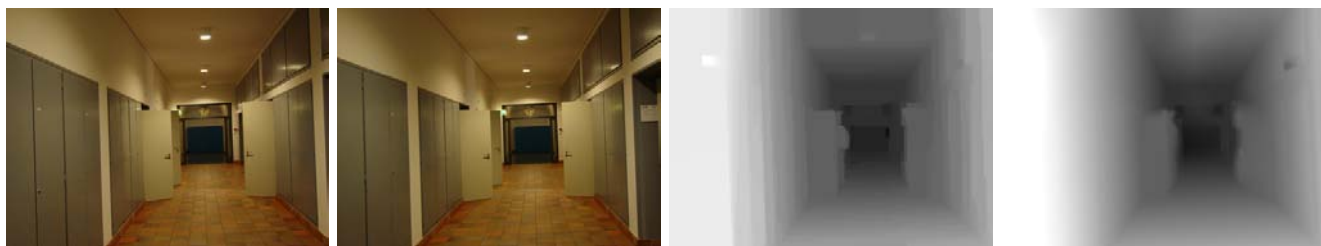


Figure 5. From left to right: left and right input image, stereo result with TV prior, stereo result with piece-wise planar prior

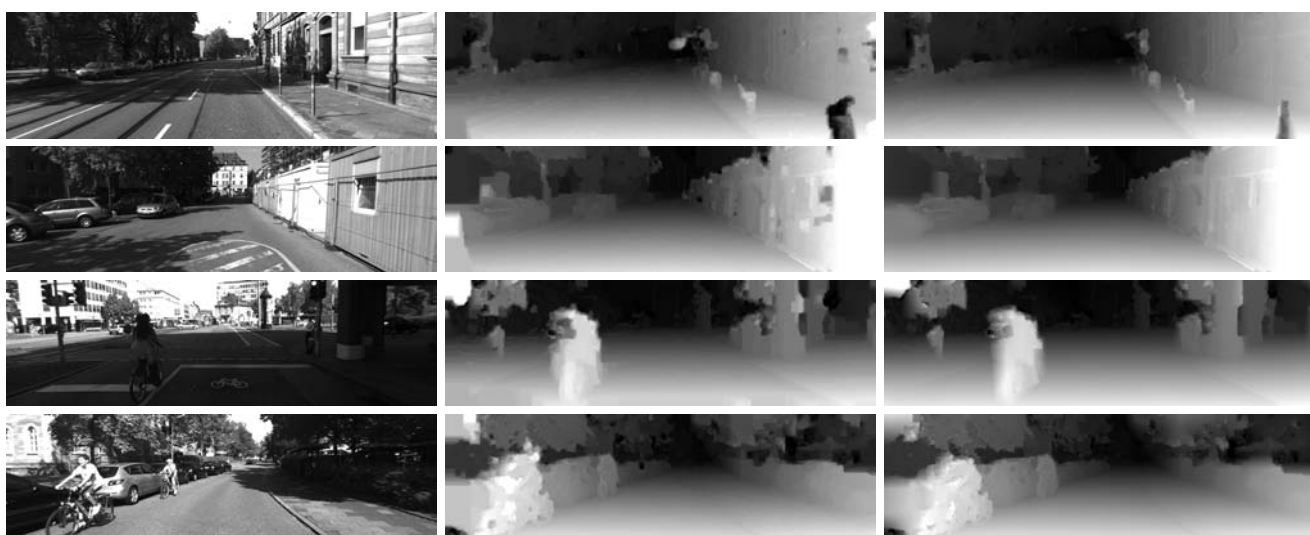


Figure 6. Results for the four urban data sets (available from <http://rainsoft.de/software/libelas.html>): left input image (672x196), depth from stereo using the TV prior, depth from stereo using the piecewise planar prior.

REFERENCES

- [1] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Transactions on Image Processing*, vol. 15, no. 12, pp. 3736–3745, 2006.
- [2] O. Woodford, P. Torr, I. Reid, and A. Fitzgibbon, "Global stereo reconstruction under second-order smoothness priors," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 31, pp. 2115–2128, 2009.
- [3] H. Ishikawa and D. Geiger, "Rethinking the prior model for stereo," in *European Conference on Computer Vision (ECCV)*, 2006, pp. 526–537.
- [4] L. Hong and G. Chen, "Segment-based stereo matching using graph cuts," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 2004.
- [5] S. Birchfield and C. Tomasi, "Multiway cut for stereo and motion with slanted surfaces," in *IEEE International Conference on Computer Vision (ICCV)*, 1999, p. 489.
- [6] M. Lin and C. Tomasi, "Surfaces with occlusions from layered stereo," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, pp. 1073–1078, 2004.
- [7] M. Bleyer, C. Rother, and P. Kohli, "Surface stereo with soft segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.
- [8] W. Trobin, T. Pock, D. Cremers, and H. Bischof, "An unbiased second-order prior for high-accuracy motion estimation," in *Proc. DAGM*, 2008.
- [9] K. Bredies, K. Kunisch, and T. Pock, "Total generalized variation," *SIAM J. Imaging Sci.*, vol. 3, no. 3, pp. 492–526, 2010.
- [10] T. Pock, L. Zebadin, and H. Bischof, "Rainbow of computer science," 2011, ch. TGV-Fusion, pp. 245–258.
- [11] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *Journal of The Royal Statistical Society Series B*, vol. 68, no. 1, pp. 49–67, 2006.
- [12] M. Werlberger, T. Pock, and H. Bischof, "Motion estimation with non-local total variation regularization," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.
- [13] F. Steinbrücker, T. Pock, and D. Cremers, "Large displacement optical flow computation without warping," in *IEEE International Conference on Computer Vision (ICCV)*, 2009.
- [14] A. Chambolle and T. Pock, "A First-Order Primal-Dual Algorithm for Convex Problems with Applications to Imaging," *Journal of Mathematical Imaging and Vision*, pp. 1–26, 2010.
- [15] C. Zach, T. Pock, and H. Bischof, "A globally optimal algorithm for robust TV- L^1 range image integration," in *IEEE International Conference on Computer Vision (ICCV)*, 2007.
- [16] C. Zach, M. Klopschitz, and M. Pollefeys, "Disambiguating visual relations using loop constraints," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010, pp. 1426–1433.
- [17] H. Hirschmüller, "Accurate and efficient stereo processing by semi-global matching and mutual information," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005, pp. 807–814.
- [18] A. Geiger, M. Roser, and R. Urtasun, "Efficient large-scale stereo matching," in *Asian Conference on Computer Vision*, 2010.
- [19] R. A. Newcombe and A. J. Davison, "Live dense reconstruction with a single moving camera," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.
- [20] J. Stuehmer, S. Gumhold, and D. Cremers, "Real-time dense geometry from a handheld camera," in *Pattern Recognition (Proc. DAGM)*, 2010, pp. 11–20.
- [21] L. Ladický, P. Sturgess, C. Russell, S. Sengupta, Y. Bastanlar, W. Clocksin, and P. Torr, "Joint optimization for object class segmentation and dense stereo reconstruction," *Int. Journal of Computer Vision*, pp. 1–12, 2011.

F Bosch Stereo Video Sensor for Driver Assistance

Authors: Beerman

Link to [Table of Contents](#).

Bosch stereo video sensor for driver assistance

Markus Beerman

December 17, 2012

1 Introduction



Figure 1: **Bosch stereo video sensor for driver assistance** The new Bosch stereo video camera detects objects' height and distance as well as capturing standard video images. This makes it possible to implement multiple assistance and safety functions with just one sensor.

We need two eyes to see the world in binocular vision, the capacity that enables us to calculate object sizes and distances and perceive longitudinal motion. Bosch is now using a new stereo video camera to give driver assistance systems the same capability. Thanks to its binocular vision, driver assistance systems can measure distances to other objects using the video signal alone. On the basis of this, Bosch is developing functions that can independently take evasive action or steer the vehicle through a construction site. And of course

these will work in conjunction with existing assistance functions such as ACC adaptive cruise control and automatic emergency braking.

The new Bosch video camera supplies data for many different tasks. The information it provides can significantly reduce both the risk and the consequences of collisions with vehicles, pedestrians, and cyclists at speeds of up to 80 kilometers per hour. It thus offers the ideal basis for improving safety in urban traffic. A mono video camera requires extensive training before it can distinguish between different types of objects, such as pedestrians and cars, in its image field. In contrast, a stereo video camera can measure and detect all the obstacles based only on their movement and distance.

With a 12-centimeter baseline distance, the Bosch stereo camera may well be the smallest system of its kind currently available in the field of automotive solutions. As a result, it is particularly simple for automobile manufacturers to integrate into their vehicles. Each of the two CMOS image sensors has a resolution of 1.2 megapixels. Thanks to its high-quality lens system, the camera is able to capture an angle of view of 25 degrees vertically and 45 degrees horizontally, and offers a 3D measurement range in excess of 50 meters. The highly light-sensitive image sensors are capable of processing very high contrasts and cover the full spectrum of light visible to the human eye.

For V-Charge, internal distance data from the Bosch stereo camera system is made available to the environment perception module in the project. The data is represented in a fashion similar to a laser scanner: for each direction horizontally, distances to obstacles - if any - are provided. Beyond the ability of a laser scanner, the stereo camera is robust against changes in the vehicle's pitch angle or a rising road surface. In addition to their distance, obstacles are associated with attributes like motion, height, and visual appearance. Based on this information, static obstacles can be aggregated in a map, dynamic obstacles clustered and tracked, and their visual appearance used for classification.

G V-Charge Perception Framework

Authors: Derendarz, Wonneberger, Nguyen, Stellmacher

Link to [Table of Contents](#).

V-Charge Perception Framework

Wojciech Derendarz, Stefan Wonneberger,
Nghia Thien Nguyen, Martin Stellmacher

February 2, 2013

1 Central environment model

As it navigates through a parking lot, the V-Charge vehicle must be able to perceive and represent its environment. For safe navigation it is not only important to capture the spatial form of the obstacles but also to register and even predict their motions. In terms of software architecture it is essential that different motion planning algorithms (i.e. for parking maneuvers or lane following) do not need to deal with individual raw sensor readings but rather can make use of a more abstract and centralized representation of the environment. In our case this centralized environment representation - also called the environment model - consists of three distinct parts that cover distinct elements thereof:

1. **Road Graph** is used to describe the geometry of the long-term static elements of the environment - like lane structure, walls and other construction items - as well as the high level road network topology (i.e. which lane is drivable in which direction). It is created in the offline survey phase and changes only negligibly little during a single online run of a V-Charge vehicle (one example of a dynamic change would be that a given part of the road network is blocked by traffic jam). Being essentially an offline component *Road Graph* will not be further discussed in this document.
2. **Online Gridmap** is used to represent both the long-term static obstacles in the environment as well as short-term static obstacles like parked cars, motorbikes or construction site barriers. It is limited to the direct local neighborhood of the vehicle. The advantages of grid representation originate in the fact that it is a model-free approach and is thus apt at describing objects of varying shapes and sizes or even groups of objects. On the other hand, grid maps cannot natively handle movements of objects in a consistent way.
3. **Object List** is used to represent moving obstacles such like vehicles, pedestrians and alike. In this representation we assume that all moving objects can be well modeled by 3D cuboids, of which at different times only some parts - for example L-shapes - are visible. We call the process of modeling obstacles as objects and tracking them by the term *object fusion*. Because of the model-based approach object fusion is well capable of capturing the object motion. This in turn allows to perform also short-term prediction of object motion.

As can be seen in Figure 1 some elements of the environment such as walls, k-rails and pillars will be represented both within Road Graph and Online Gridmap. This overlap is natural and could be even exploited as an additional cue in the absolute localization problem. It is also intuitive that there is no

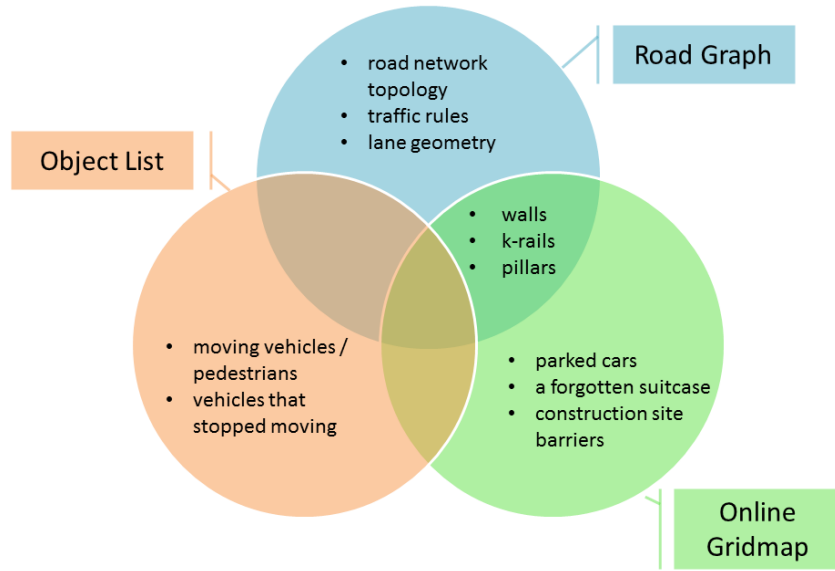


Figure 1: Some examples of environment elements mapped to the 3 components of the central environment model: Road Graph, Online Gridmap, Object List

overlap between Road Graph and Object List. Conversely, the lack of overlap between Online Gridmap and Object List is in general the desired behaviour.¹

2 Sensor setup and data sources

As shown in Figure 2 currently V-Charge vehicles are equipped with 3 different sensor systems: *stereo camera*, *mono camera system* and *ultrasonic sensor array*.² Both the stereo system and ultrasonic systems provide distance readings.³ The mono system provides fish-eye imagery covering the 360° surroundings of the vehicle. In course of the project a multitude of perception algorithms will be employed to extract higher level data like distance or obstacle information out of this raw imagery. These mono video perception algorithms include:

- **object detection via classification** - targeted at detecting vehicles and pedestrians,
- **motion stereo** - used to detect static obstacles of any form,
- **object detection via flow field segmentation** - targeted at detecting moving objects,
- **object detection via background subtraction** - targeted at detecting moving objects whilst the ego vehicle is at standstill.

¹In some borderline situations that goal can be difficult to achieve - consider for instance a very slowly moving vehicle.

²In the course of the project another rear facing stereo camera will be integrated with the vehicle.

³Of course the spatial resolution and range of the stereo system are much better than that of the ultrasonic system.

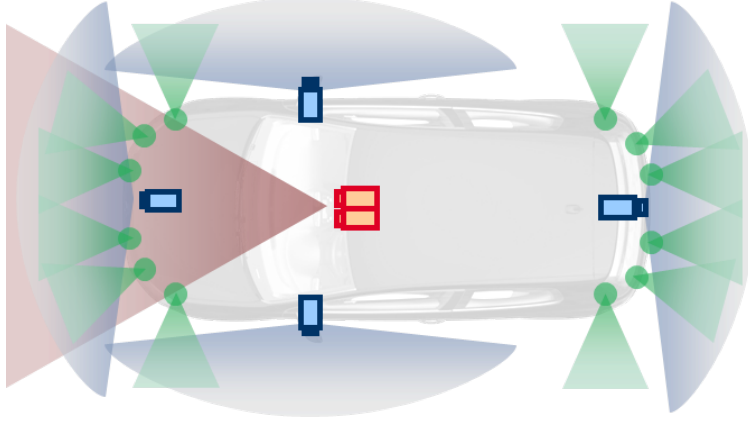


Figure 2: The sensor setup of the V-Charge test vehicles including a schematic representation of the field of view of the individual sensors (green is used for sonar sensors, red for stereo-camera system, and blue for mono fish-eye cameras).

Currently only distance readings from the stereo camera and the sonar system as well as results of object detection via classification (see appendix G) are available in the vehicle.

3 Data processing

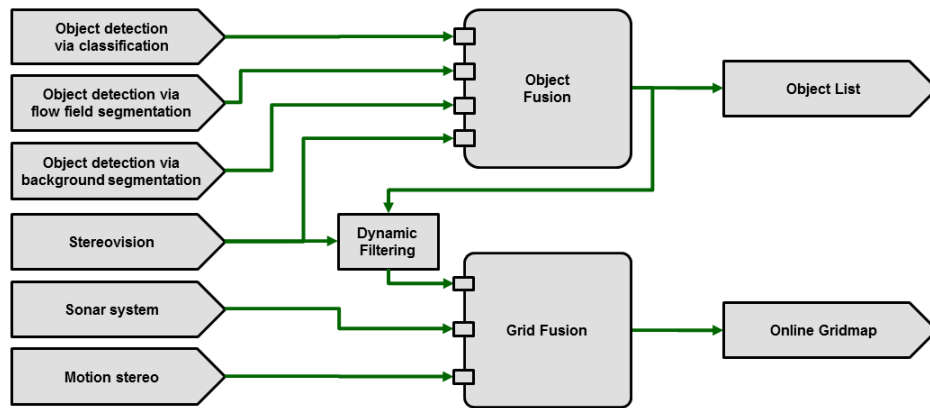


Figure 3: High-level architecture of the perception framework

Figure 3 shows how data from different sources will be integrated into the central environment model. It can be seen that data provided by some modules - like ultrasonic and motion stereo - will only be processed within the online gridmap. On the other hand data provided by modules such as object detection via classification, object detection via flow field segmentation or object detection via background subtraction will be processed by the object fusion. Obstacle data from the stereo system will be first processed by object fusion. The obstacle data that was not associated to any of the existing object hypothesis and that did not lead to initialization of a new object hypothesis will be then processed in the gridmap.

Please note that in our current implementation data corresponding to both static and dynamic obstacles is being processed in the occupancy gridmap.

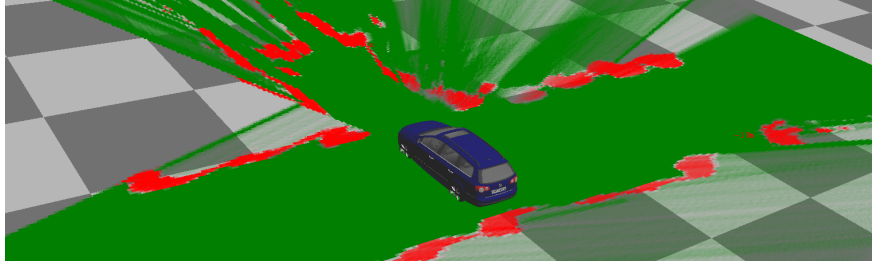


Figure 4: Occupancy grid for a typical parking lot scene: vehicle has passed several occupied parking spaces and positioned itself in front of an empty one.

3.1 Multiple Sensor Static Grid Mapping and Fusion

In the gridmap representation of the environment the direct neighborhood of the vehicle is sampled into individual quadratic cells.⁴ We use binary logic, which means that each cell may be either *free* or *occupied* (i.e. by an obstacle). The data processing in gridmap amounts to finding a probability that a given cell is occupied. The occupancy probability varies between 0 (cell is free) and 1 (cell is occupied). All cells are initialized with the value of 0.5 meaning that the occupancy status is unknown.

The online gridmap within V-Charge is world-centric in the sense that each cell has unique *world* coordinates, that do not change over time. Of course the online gridmap needs to represent the local neighborhood of the vehicle. As the vehicle moves and explores new areas the *active* section of the grid map (i.e. 50x50m large) is shifted such that it is centered around the vehicle. Please note that no rotation is performed. For efficiency reasons, the shift of the gridmap is performed by using a double-ring-buffer (x and y direction). The position of the ego-vehicle in the gridmap is updated based on vehicle odometry, which is jump-free and locally consistent.

In the current implementation, each sensor output is first processed in an individual layer so that sensor specific details can be handled accordingly. Each layer might use different cell and grid size, which - for computational efficiency reasons - are defined as multiples of the values of an abstract base layer. The Bayesian logic is used with the logit representation with saturation, which reduces the computational complexity of a cell update significantly.

Each sensor is modeled by a specific sensor model which determines the individual cells that are to be updated during each sensor measurement cycle. The stereo system provides obstacle information represented as object segments (compare appendix F), each of which corresponds to a group of columns in the image. In the stereo sensor model, for each potential object segment⁵ a corresponding circular section is calculated. Please note that all circular sections sum up to the original horizontal field of view of the camera. Based on that circular section one can easily derive the individual cells of the gridmap that are to be updated. For simplicity and performance reasons a discretized polygon traverse is used. For the stereo system the area of the circular arc corresponding to the detected obstacle is updated by a positive value. In this process the uncertainty of the obstacle position is modeled by a Gaussian distribution. The area corresponding to the circular section up to the detected obstacle is updated by a negative value.⁶ The area behind the obstacle is left unchanged. Circular sections for which no object segment was delivered in a given measurement cycle are updated with

⁴Such approaches go back to the early works of e.g. [1].

⁵Each measurement cycle provides object segments for all groups of columns. All of them are likely to be set to valid if vehicle is positioned in front of a wall. Conversely if no obstacles are in the field of view of the camera, all object segments are marked as invalid.

⁶Please note that we use the logit representation in which the occupancy probability is expressed by a float value ranging from negative (free) over zero (unknown) to positive (occupied).

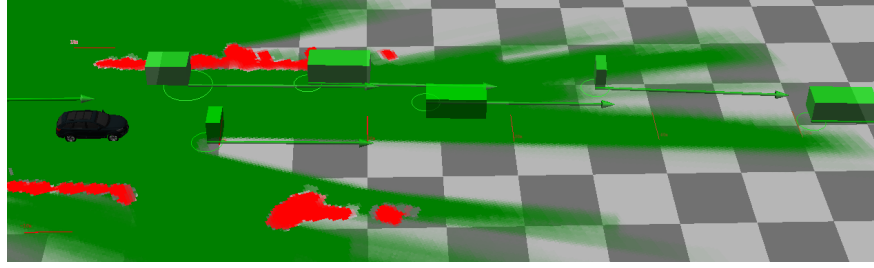


Figure 5: Visualization of dynamic objects with determined velocity and static obstacles. Note the consistency of the environment model: the dynamic obstacles are only represented by 3D cuboids and are not present in the gridmap.

a negative value only and maximal range. Such update strategy results from the assumption that the stereo system does not oversee obstacles. Fig. 4 shows a map of a parking scene, created by integrating the stereo data. Red areas correspond to occupied cells and green areas to free spaces.

For the ultrasonic system an analogous model is applied with the difference that a single circular arc corresponds to a single sensor. For now only the primary echoes are taken into account. The secondary and cross echoes are ignored. Currently the ultrasonic sensor model is parameterized for collision avoidance: a worst-case positive update is performed in the gridmap. Specifically, that means that for each echo the occupancy probability of all grid cells corresponding to the respective circular arc is increased. As the field of view of the ultrasonic sensors is large (about 120°) this often leads to over-pessimistic results. In our future work we will investigate ways to overcome those shortcomings.

In the final step, the two grid layers - corresponding to sonar sensors and stereo camera - are binarized and fused. The binarization is performed by means of empirically estimated thresholds. In the fusion step a single or-connection is applied with a prioritization on occupied state in case of contrasting states.

3.2 Object fusion

The integration of the object fusion is underway. A preliminary result obtained using a different sensor-setup can be seen in Fig. 5. The details of the object fusion framework, as well as more results will be given in Deliverable 3.4.

4 Conclusion and Outlook

In the first part of this technical explains the general concepts behind V-Charge perception framework. Encouraged by the first results, we firmly believe that the presented concepts form a good basis for safe navigation in dynamic environments.

Second part of this document provides details about the current implementation, that is available within the vehicle. During last months the online gridmap was used as a basis for a series of successful automated obstacle avoidance maneuvers. On the other hand, the experiments suggest that for successful parking maneuvers more elaborate processing of ultrasonic data is necessary.

In our future work we will focus on the following aspects:

- improvements to the existing modules, especially the ultrasonic data processing,
- integration of the object fusion into the vehicle
- integration of all available and upcoming data sources, especially: object detection via classification and motion stereo.

References

- [1] H.P. Moravec. Sensor fusion in certainty grids for mobile robots. *AI magazine*, 9(2):61, 1988.

H The V-Charge Project: Obstacles Detection, Tracking and Classification

Authors: Cardarelli, Cattani

Link to [Table of Contents](#).

The V-Charge Project: Obstacles Detection, Tracking and Classification

Elena Cardarelli, Stefano Cattani

15th December 2012

The activities performed by University of Parma has been focused on obstacles recognition, providing two separate schemes for, respectively, mono and stereo images processing. For the mono processing case the work done has involved three main aspects to handle: detection, tracking and classification of static and dynamic obstacles. Concerning the stereo processing the main focus has been only the classification of the information associated to the 3D object clusters provided by stereo preprocessing phase. The mono and stereo processing pipelines are currently designed to allow frontal and rear vehicles recognition, through the application of a classifier in specific region of the input images: future developments will be dedicated to the integration of the pedestrian detection and classification modules. Moreover the detection algorithm will be improved to allow the vehicles recognition from all faces.

Mono processing

The preprocessing stage for vehicles detection has involved, firstly, the perspective images processing in order to correct the distortion introduced by fish-eye camera lens: this allowed to obtain undistorted images representing the scene of the V-Charge vehicle surrounding area.

Vehicles Detection

The vehicle detector implemented is based on the application of an AdaBoost classifier [1] with a Viola-Jones design [2] in specific areas of the

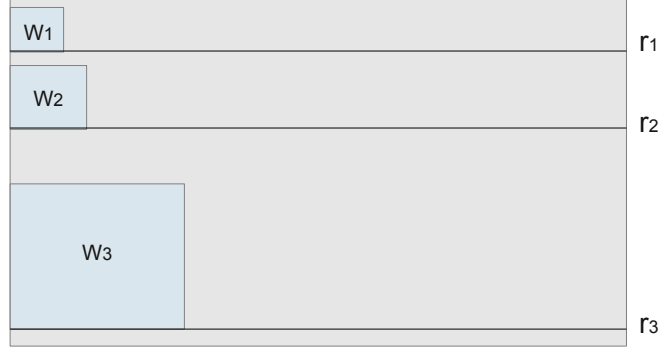


Figure 1: Determination of the region of interest through the world to image coordinates transform. r_1 , r_2 and r_3 are images rows that represent possible vehicles detection distances; while w_1 , w_2 and w_3 are the corresponding investigated windows, defined according to the area occupied by a vehicle, with a typical dimension, at the specific distance represented by that image row.

perspective images previously processed. To determine the region of interest on where execute the classifier a typical vehicle dimension (i.e. a vehicle width and height aspect ratio) is considered: thus, each row of the input image is investigated reconstructing, through the exploiting of the calibration parameters and the relative world to image coordinates transform, a set of candidate windows, each corresponding to the area that a vehicle, with the typical size defined, may occupy at the specific distance represented by the processed row (figure 1).

The set of region reconstructed are then processed using an AdaBoost classifier trained with frontal and rear vehicle samples: this allows to determine if, in the area delimited by each preprocessed window, there is a vehicle or not. A multiscale detector scheme is designed to improve, in presence of a vehicle, the accuracy in the determination of its covered area.

AdaBoost training

Particular attention has been dedicated to the training of the AdaBoost classifier: to allow the vehicles recognition in real situation of the road environment the patterns used to train the classifier have been extracted from

the perspective images preprocessed. Thus, a set of 14087 samples of 32×32 pixels, representing vehicles from frontal and rear faces, have been obtained and separated in training and validation set. As shown in table 1 the training set consists of 10507 samples, which 3430 are positive samples and 7077 are false positive ones. The validation set, used to test the detector performance, consists of 3580 patterns representing only positive samples: to measure the capability of correctly identifying false positives the vehicle detector has been tested runtime, on images not integrated in the training set, counting in the input frames, respectively, the average and the maximum number of false positives generated. The test demonstrated that the false positives number is on average lower than 1, while in the input images the maximum number of false positives detected is 3.

	Positive Samples		False Positives	Total
	<i>Front</i>	<i>Rear</i>	<i>Front and Rear</i>	<i>Front and Rear</i>
Training Set	1382	2048	7077	10507
Validation Set	1015	2565	/	3580
Total	2397	4613	7077	14087

Table 1: Design of the training and validation set.

For the recognition step a single classifier, trained either with frontal and rear vehicle views has been designed to optimize the computational overhead: with the acquired data it has been experimentally demonstrated that to obtain the correct and complete training set recognition, two distinct classifiers need 400 iterations (106 for the frontal detection and 264 for the rear one), while with a single classifier the 100% of the training set correct recognition is reached in less time, with 356 iterations. The classifier used for the vehicles detection is currently an AdaBoost with softcascade scheme, trained for 200 stages. This choice allows to provide a compromise between the accuracy of the multidetection scheme and the computational time required.

Vehicles Tracking

To cope with the presence of blinking detection a tracking filter is applied on the region candidates to contain a vehicle: this allows to improve the performance making more stable and robust the results. The designed tracking procedure is based on a Kalman filter: assuming that all the objects in the scene follow an uniform linear motion, the information tracked corresponds to the position of the obstacle ground point expressed in world coordinates.

The tracking filter allows to attenuate the variation of an object movement among subsequent frames, limiting the motion to a specific range, determined according to the object velocity and position measured, processed and estimated by the tracking filter. Moreover to reduce the variability of the results, coping with blinking detection, an object is considered valid only if it has been detected at least 4 times in subsequent frames and, in case of a miss detection, the object position reconstruction is provided for at most the next 3 frames.

The uniform linear motion assumption introduces noise in the object position estimation when the V-Charge vehicle faces on curves: in future, the integration of the odometry and vehicle status information will allow to cope with this problem, improving the accuracy of the results.

Performance

The vehicles detector and tracker developed has been tested on the acquired images, considering for the evaluation of the classification performance either the traditional AdaBoost scheme and its cascade version. The two classifiers have been tested on the defined validation set, measuring the correct detection rate. It has been demonstrated that, from a detection performance point of view, the two classifiers provide equivalent results: the correct detection rate for AdaBoost is 80%, while for the cascade scheme is 76%. Concerning the computation analysis the cascade version has provided better results than the traditional AdaBoost. Table 2 shows the times required by the two schemes varying the number of scales for octaves considered in the designed multidetection scheme; increasing the number of the scales, a more accurate detection result is ensured, at the expense of a greater computational cost.

Number of scales	AdaBoost	Cascade
1	53 ms	14 ms
2	98 ms	26 ms
4	184 ms	52 ms

Table 2: Computational analysis.

To provide a compromise between computational performance and accuracy the multidetector scheme designed is a softcascade with 4 scales for octave. This detection scheme and the pattern dimension defined for the training set (32×32) allow to recognize vehicles in a range between 2 and 25 meters.

Figure 2 show some results relative to frontal and rear vehicles detection at different distances in images acquired either with frontal and rear camera.

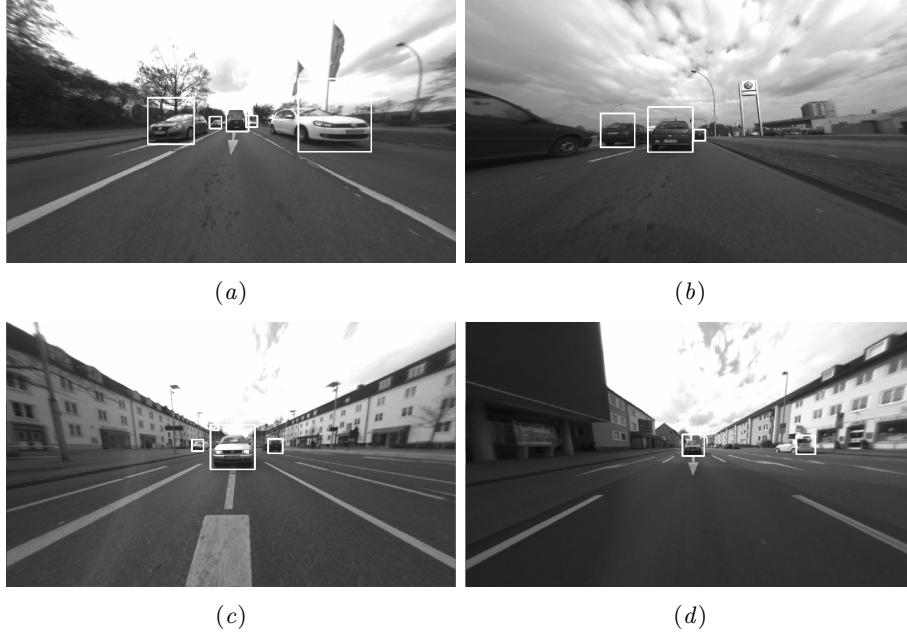


Figure 2: (a) Frontal vehicles detection at different distances, in images acquired from the rear camera. (b) Rear vehicles detection at different distances, in images acquired with the frontal camera. (c)-(d) Frontal and rear vehicles detection with rear camera.

Stereo processing

The stereo preprocessing phase is based on the analysis of the stereo segments obtained by the disparity processing: the main focus is the determination of the 3D clusters compatible with the presence of an obstacle.

Vehicle classification

The vehicles classification phase relative to the stereo pipeline is based on an approach similar to the mono processing one and it involves the using of an AdaBoost classifier with multitetection scheme. The main difference

is that the classifier is not executed in areas determined considering typical object sizes and different detection distances, but the specific windows for vehicle presence investigation depend on the area delimited in the image by the 3D clusters preprocessed.

Thus, the classification processing involves firstly the world to image coordinates transform of the 3D clusters, in order to determine the areas they occupy in the image. Then, the same classifier used for the mono pipeline is applied on these region to determine the possible presence of a vehicle.

References

- [1] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the Second European Conference on Computational Learning Theory*, EuroCOLT '95, pages 23–37, London, UK, UK, 1995. Springer-Verlag.
- [2] P. Viola and M. Jones. Rapid Object Detection using a Boosted Cascade of Simple Features. In *Intl. Conf. on Computer Vision & Pattern Recognition*, volume 1, pages 511–518, Dec. 2001.