Rheinische
Friedrich-Wilhelms-
Universität Bonn

Institute for Computer Science
Department VI
Autonomous Intelligent Systems

Rheinische
Friedrich-Wilhelms-Universität Bonn

Master thesis

Combining Feature-based and Direct Methods for Semi-dense Real-time Visual SLAM from Stereo Cameras

Author:
Nicola Krombach

First Examiner:
Prof. Dr. Sven Behnke

Second Examiner:
Prof. Dr. Cyrill Stachniss

Advisor:
M.Sc. David Dröschel

Submitted: February 11, 2016

Declaration of Authorship

I declare that the work presented here is original and the result of my own investigations. Formulations and ideas taken from other sources are cited as such. It has not been submitted, either in part or whole, for a degree at this or any other university.

Location, Date

Signature

Abstract

For autonomous navigation of Micro Aerial Vehicles (MAVs), a reliable and efficient state estimation is of utmost importance. Due to their low weight and size, cameras are often used as sensors on MAVs and can cover a wide field of applications, such as simultaneous localization and mapping (SLAM), obstacle detection or 3D reconstruction. Specific applications for cameras on MAVs are for example position control, obstacle avoidance, and autonomous navigation and mapping, showing the flexible possibilities a camera as a sensor offers. Multiple different tasks can be performed using a single sensor.

For state estimation with visual SLAM, our MAV is equipped with six fish eye cameras, covering an omnidirectional field of view.

In this thesis, a novel approach for estimating the egomotion of MAVs using visual SLAM from stereo cameras is presented, which combines feature-based matching with semi-dense direct image alignment.

Recent visual SLAM methods can be classified into either feature-based methods or direct methods. While feature-based methods rely on sparse image features only, direct methods use the entire image for tracking. As feature-based methods keep a sparse representation of the image, they are inherently faster than direct methods. Direct methods, however, estimate the motion based on all information of the image and are thus more accurate. By combining both paradigms we are able to combine their strengths. We are not only able to efficiently estimate the pose of our MAV with a high frame rate, but also reconstruct a semi-dense map of the environment online, using direct tracking.

The major challenges are the strong radial distortions of the employed fish eye lenses and the real-time requirements for reliable navigation on MAVs.

We evaluate our method on the popular KITTI dataset, on the EuRoC dataset and on our MAV as well. In experiments we show, that our approach accurately estimates the camera motion and reconstructs a globally consistent semi-dense map of the environment. Quantitative results show, that our approach outperforms other state-of-the-art methods on the KITTI and on the EuRoC dataset. Due to the strong radial distortion of the fish eye lenses and a non-rigid stereo setup, our approach shows less accurate results on our MAV dataset, but still achieves an average absolute trajectory error (ATE) of 0.8 m.

Contents

List of Figures

List of Tables

1 Introduction

Recently, unmanned micro aerial vehicles (MAVs) have become increasingly popular in ongoing research towards autonomous mapping and exploration, surveillance and inspection, or search and rescue missions. Due to their low weight and size, micro aerial vehicles can be used in restricted or cluttered environments without getting stuck as likely as ground vehicles might. Especially quadrotors and hexarotors are very popular for such tasks, because they can easily hover at a position, are reliable and compact and need only low maintenance. However, autonomous navigation necessitates a reliable sensor setup and fast on-board methods for state estimation and control.

On ground vehicles, laser scanners are heavily used as main sensors and are successfully deployed for laser-based simultaneous localization and mapping (SLAM) and obstacle avoidance. However, laser scanners are usually very heavy and large, restricting their use on MAVs by the limited payload and battery life. Thus, a popular approach is to equip the MAV with one or more lightweight cameras and use computer vision techniques to estimate the state of the MAV.

While other MAVs are often equipped with one monocular camera or one stereo pair facing forward, the presented setup consists of an omnidirectional mounting of three stereo camera pairs, allowing for horizontal obstacle perception in all directions. Moreover, they achieve a high frame rate of up to 45 Hz allowing for dynamic maneuvers and fast reactive behavior. For state estimation and localization, methods for visual odometry or even visual SLAM can be used with one or more cameras. Similar to laser-based SLAM, visual SLAM simultaneously tries to localize the current pose while incrementally building a map. Recent visual SLAM methods can be classified into either feature-based methods or direct methods. While feature-based methods only use a sparse representation of the given image by reducing the scene to the set of observed feature points, dense direct methods use all pixels of the image to perform direct image alignment. As efficiency and real-time constraints are of utmost importance for autonomous navigation, dense methods are often unfeasible.

Recently, LSD-SLAM (Engel, Schöps, et al., 2014) has been proposed as a semi-dense monocular method, which locally tracks the camera motion using direct image alignment and continuously builds a semi-dense depth map. When using monocular methods for autonomous navigation, one has to deal with the inherent scale ambiguity by either using additional sensors to estimate the scale or by initialization with absolute values. Monocular methods suffer from the problem that the absolute scale of the scene cannot be observed, so that further sensors are needed for absolute scale estimation. Moreover, the absence of absolute scale leads to scale drift over time, yielding inconsistent maps.

The goal of this thesis is to extend LSD-SLAM to a stereo SLAM system. By adding a second camera, the absolute scale becomes observable, greatly reducing scale drift and the necessity for additional sensor measurements. Moreover, instant absolute depth measurements can now be used for the direct tracking of the images, as well as for the mapping of the environment. A drawback of direct methods for visual SLAM, which use the entire image for tracking, is that they are more computationally demanding than feature-based methods, because—even with VGA resolution—they deal with several hundred thousand constraints in contrast to a sparse set of 200 to 1000 features. Moreover, they assume very small image displacements for tracking, and thereby tend to fail at large inter-frame motions and rotations. In addition to the stereo extension of LSD-SLAM, we present a novel approach, which combines the direct image alignment with feature-based matching to keep up with the high frame rate and the dynamic motions of our MAV. We employ the direct tracking for key frames only, while all other frames are tracked using a feature-based method.

Thus, we are not only able to track the MAV in real-time, but are also able to reconstruct a semi-dense 3D map of the environment, as shown in Figure 1.1.

1.1 Contributions

In this thesis we present a novel approach for visual SLAM with stereo cameras, which combines semi-dense direct SLAM with a feature-based method for visual odometry. We build upon an open-source semi-dense direct method which uses a monocular camera. The drawback of monocular visual SLAM methods is that the absolute scale of the scene is not observable. This leads to scale drift over time or to the need of additional sensor fusion from, for example, measurements of an inertial measurement unit (IMU). As our MAV is equipped with three stereo camera pairs, stereo vision is possible.

Thus, the first contribution of this thesis is to extend the existing monocular approach to stereo cameras. By using stereo cameras, the absolute scale becomes observable, leading to less scale drift. With stereo cameras, instant depth measurements become available, while in monocular methods the depth can only be estimated up to a scale.

As additional contribution we introduce a novel approach of a combined visual SLAM scheme: while the direct tracking is done on key frames only, all frames in between are tracked using sparse visual odometry. By reducing the direct tracking to key frames we can keep up with high frame rates and can use the estimate of the sparse visual odometry as a decent initialization for the tracking of new key frames. Therefore, we are not only able to track the camera motion in real-time,

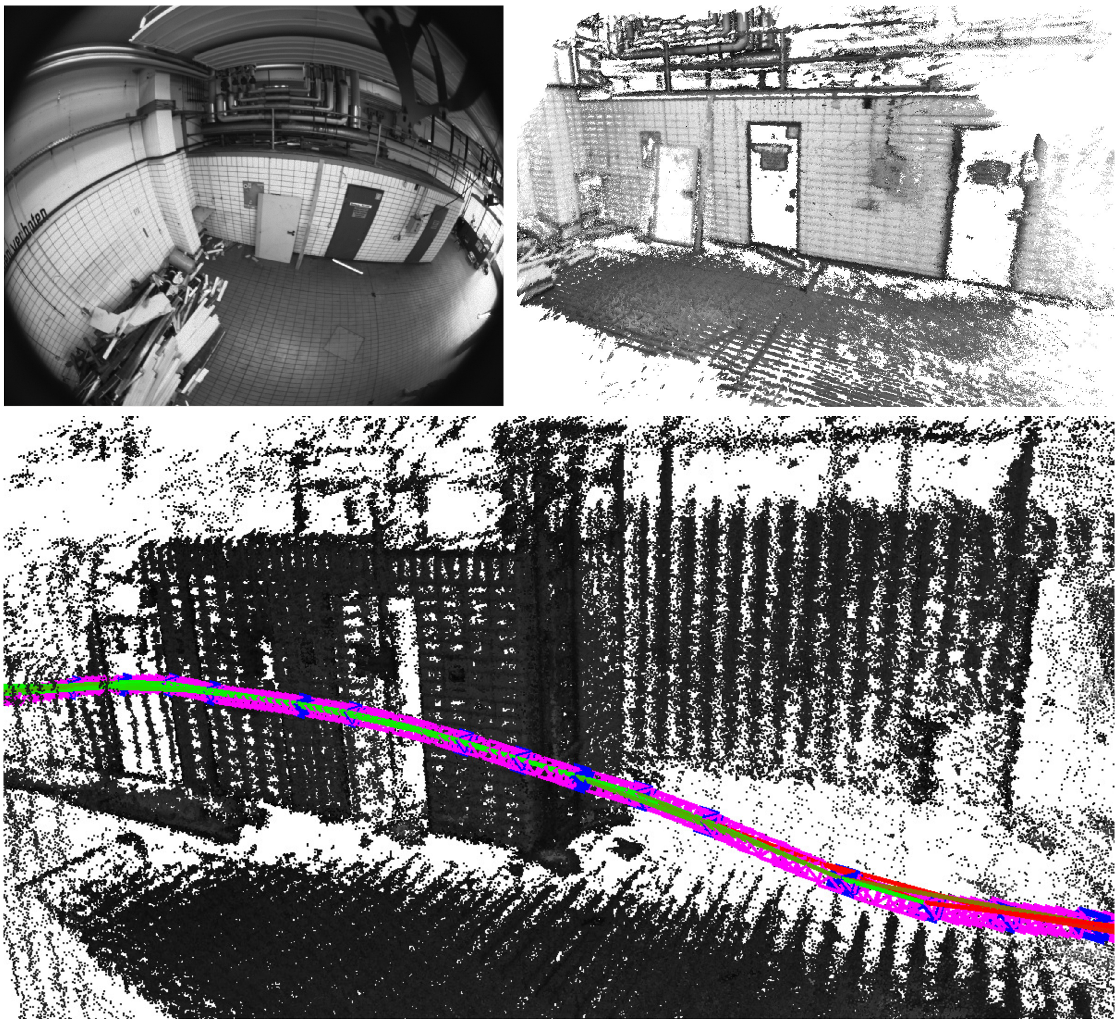


Figure 1.1: 3D reconstruction of a scene captured by our MAV during flight: while estimating the camera motion using our semi-direct approach, a semi-dense 3D map of the environment is constructed. The raw fish eye image and the corresponding semi-dense depth map are shown on top. The trajectory is visualized as pose-graph containing key frames (blue), feature-based tracked frames (pink) and edge constraints (green).

but also can do a dense reconstruction of the environment by semi-dense direct tracking on stereo frames.

1.2 Outline

In Chapter 2 the MAV—used for the experimental part of this thesis—is described. Furthermore the general theoretical background needed for retrieving the camera pose is introduced. Then, an overview of recent related work about sparse and dense visual SLAM methods follows in Chapter 3. Chapter 4 deals with the calibration of our camera system and presents different calibration methods. Afterwards, Chapter 5 presents the approach developed in this thesis. We start with explaining the general concepts of LSD-SLAM as our choice for a direct method and LIBVISO2, as feature-based method, and present the algorithmic advancements developed in this thesis. Moreover, we then evaluate the newly proposed method quantitatively and qualitatively on different datasets, for example, the well-known KITTI dataset, in Chapter 6. Finally, the contributions and results are summarized in Chapter 7.

2 Background

In this chapter, we introduce the utilized MAV and its main sensors as well as the theoretic background that forms the base of this thesis.

2.1 Sensor Setup

Our MAV is built as high-performance platform with a multimodal omnidirectional sensor setup (Beul et al., 2015). As MAVs have very limited payload, we use only lightweight components and are capable of navigating indoor and outdoor. Especially for (fully) autonomous navigation in unknown and dynamic environments, a multimodal and omnidirectional sensor setup is of great advantage. The strengths of the different sensors can be combined and their measurements can be fused in an occupancy grid map.

The MAV is built as hexarotor with six 14″ propellers each connected to a MK3644/24 motor. For better stability and collision protection the MAV is surrounded with a non-rigid milled frame that does not only protect the rotors, but also serves as mount for various sensors. For on-board computation in real-time, the MAV is equipped with a mini-ITX board, namely a Gigabyte GB-BXi7-4770R with an Intel Core i7-4770R quad-core CPU, 16 GB DDR-3 memory and a 480 GB SSD to process all sensor outputs.

We employ a multimodal sensor setup consisting of IMU, laser scanners and cameras. The MAV is equipped with a Pixhawk Autopilot flight control unit (FCU) (Meier et al., 2012) for low-level velocity and altitude control. The Pixhawk FCU comes with accelerometers, a barometer, gyroscopes and a compass. The included firmware has been adapted to fulfill our needs by additional integration of visual and laser odometry estimates. Moreover, our system is equipped with two laser scanners and six cameras for high-level autonomous operation and navigation.

In particular, we use two rotating Hokuyo UST-20LX laser scanners, each with a scan range of 20 m and 270° apex angle. Together they can perform a full 3D scan of the environment with 4 Hz. They are used for obstacle perception and SLAM-based 6DOF localization (Nieuwenhuisen et al., 2015).

For visual obstacle detection and visual SLAM, the MAV is equipped with an

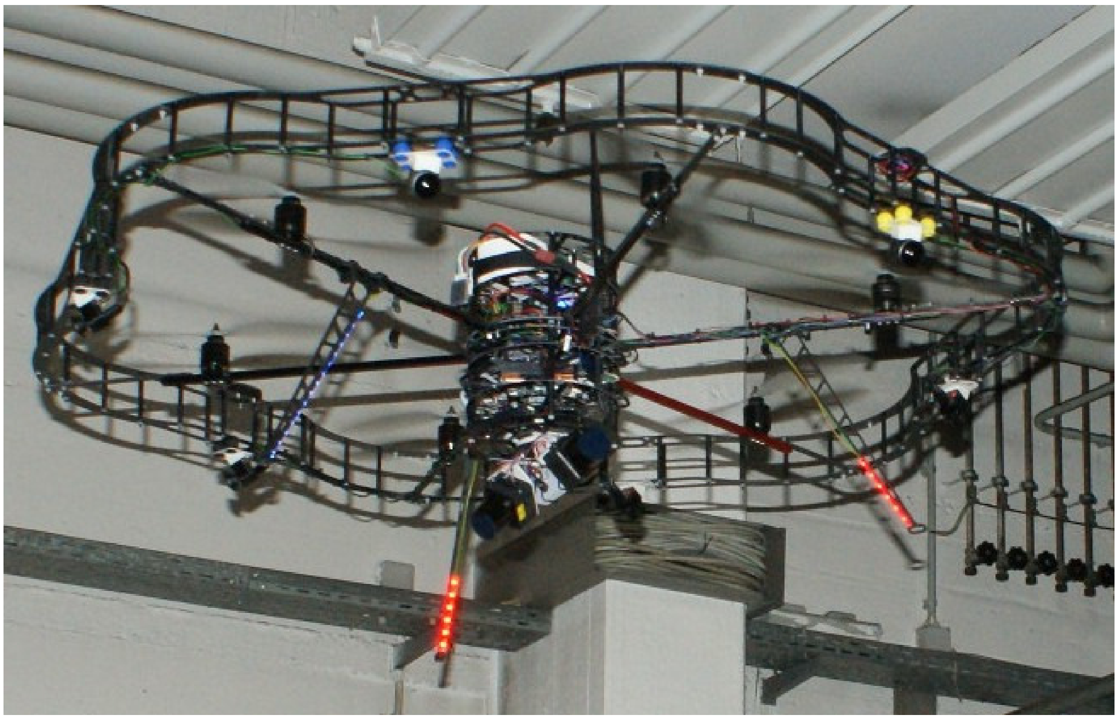


Figure 2.1: High performance MAV during flight. The omnidirectional sensor setup includes three fish eye stereo pairs covering a wide field of view for autonomous navigation.

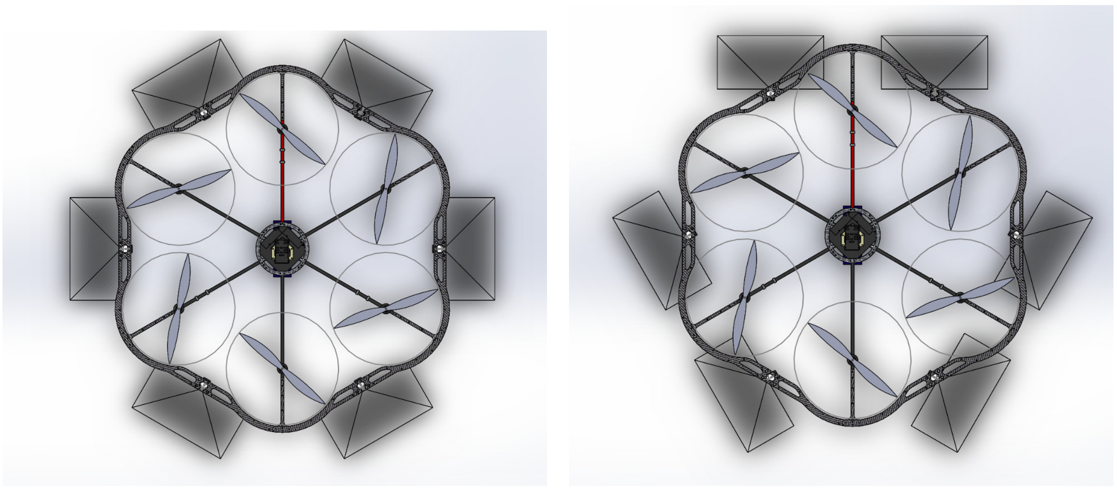


Figure 2.2: Top down view of possible camera configurations: the left image shows a fully omnidirectional setup with independent optical axis, while on the right a stereo setup consisting of three independent stereo pairs is shown.

omnidirectional camera setup. The cameras are mounted to the non-rigid body frame using dampers to filter out vibrations induced by the six propellers. The camera mounting can easily be switched from a fully omnidirectional setup with independent optical axes to a stereo setup with three stereo camera pairs, as can be seen in Figure 2.2. The multi-camera setup allows omnidirectional perception of the environment and allows robust state estimation due to redundant information sources, i.e., even if one stereo pair faces a homogeneous wall with no texture the other two pairs still allow for robust localization. We use XIMEA MQ013MG-E2 global-shutter monochrome USB 3.0 cameras with 1.3 MP resolution in combination with Lensagon BF2M2020S23 fish-eye lenses for a wide field of view. By making use of the available independent USB controllers of the on-board system, we distribute the USB traffic and thus can achieve high camera frame rates at full resolution. Each stereo pair is connected to a USB 3.0 HUB, which is connected to a dedicated on-board USB 3.0 port that offers full USB 3.0 speed. Through this setup we ensure that for each camera enough bandwidth is available. Assuming that each HUB offers 2400 Mbit/s (300 MB/s), each camera may use up to 1200 Mbit/s (150 MB/s). Theoretically, each camera can achieve the best possible frame rate of 60 Hz in 8-bit mode and 57 Hz in 16-bit mode.

However, the real data rate is limited by additional system and protocol overhead when reading and writing from the connected devices.

Under real lighting conditions and depending on exposure times we achieve up to 50 Hz for each camera in 16-bit mode. Our camera driver not only ensures that the images are published synchronously, but also offers advanced functionality like downsampling, gamma correction or rectification.

2.2 Camera Model

2.2.1 Perspective Camera Model

The Perspective Camera Model or Pinhole Camera Model, as shown in Figure 2.3, is a simple and widely used model. It describes the mathematical projection from 3D world coordinates to a 2D image plane. This perspective transformation contains the projection from 3D world coordinates to 3D camera coordinates, which is a mapping from $\mathbb{R}^3$ to $\mathbb{R}^3$, and the projection from 3D camera coordinates to 2D image coordinates, which by contrast is a mapping from $\mathbb{R}^3$ to $\mathbb{R}^2$.

The origin of the camera coordinate system is called center of projection COP. The line from the COP perpendicular to the image plane is called principal axis, and its intersection with the image plane is called principal point or optical center, denoted with $c = (c_x, c_y)^T$, whereas the distance of the COP to the image plane

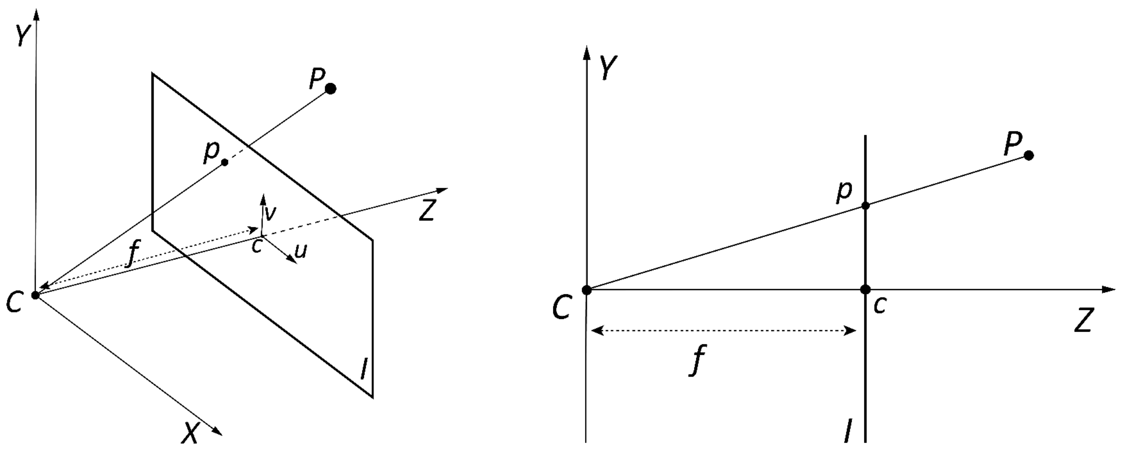


Figure 2.3: Perspective camera model: a 3D Point P is mapped to a point p on image plane I by the ray connecting P with the center of projection C.

is called focal length f. A 3D point in camera coordinates $P_{cam} = (X, Y, Z)^T$ is mapped to the image plane I at the intersection of the ray connecting the 3D point with the COP, as visualized in Figure 2.3. Using similar triangles the 2D projection (u, v) of a 3D point $P_{cam} = (X, Y, Z)^T$ can be calculated by:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \frac{f}{Z} \begin{pmatrix} X \\ Y \end{pmatrix} . \tag{2.1}$$

Equation (2.1) assumes that the origin of the image plane is at its principal point c, while in practice the image plane's origin is located—depending on definition—at the lower or upper left corner of the image. Thus, the mapping π from camera coordinates to image coordinates becomes:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \pi(X, Y, Z) := \frac{f}{Z} \begin{pmatrix} X \\ Y \end{pmatrix} + \begin{pmatrix} c_x \\ c_y \end{pmatrix} . \tag{2.2}$$

For the inverse mapping from image coordinates to camera coordinates, the depth Z has to be known, as it is lost in the projection from $\pi : \mathbb{R}^3 \to \mathbb{R}^2$:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \pi^{-1}(u, v, Z) := Z \begin{pmatrix} f^{-1}u - c_x \\ f^{-1}v - c_y \\ 1 \end{pmatrix} . \tag{2.3}$$

The projection can be expressed as matrix multiplication when using homogeneous coordinates:

$$Z \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \underbrace{\begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{K} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{(I|0)} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}. \tag{2.4}$$

The matrix K is also called camera calibration matrix and contains the so-called intrinsic camera parameters f and c. Using Equation (2.4), 3D points given in camera coordinates can now be mapped to 2D image coordinates. To describe the mapping from world coordinates to camera coordinates we introduce an additional transformation between coordinate frames. The coordinate frame of the camera must not have the same origin as the world, and instead may be rotated and translated in the world coordinate frame. A mapping from a point in world coordinates to a point in camera coordinates presents itself as:

$$P_{cam} = RP_{world} + t. \tag{2.5}$$

The 3D rotation matrix $R \in \mathbb{R}^{3\times3}$ and 3D translation $t \in \mathbb{R}^3$ form the extrinsic parameters and can also be expressed as matrix:

$$M = \left[R|t\right]. \tag{2.6}$$

By combining the transformation from world to camera coordinates with the intrinsic camera matrix, which describes the mapping from camera coordinates to the image plane, we can describe the projection from world coordinates to image coordinates by the projection matrix $G = KM$:

$$w \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = GP_{world} = KMP_{world} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \left[R|t\right] \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}. \tag{2.7}$$

2.2.2 Modeling the Lens Distortion

The pinhole camera model describes a very simple camera with a very small pinhole as aperture and no additional lens. The smaller this pinhole is, the sharper the projected image, because ideally each 3D point is projected by a single ray to the image plane. However, in reality this leads to a very dark picture or the need for very long aperture times. Contrarily, making the pinhole larger, to allow for more light, results in a blurred image, due to an increasing center of confusion as more light rays project the same 3D point to a broader area on the image plane. Therefore, in modern cameras, lenses are used, to admit more light into the camera, but also to refract the light rays to get a sharp projection.

In practice, real lenses are not perfect and come with certain radial and tangential distortion. While radial distortions result from the lens itself, and are usually stronger on fish eye lenses, tangential distortions are caused by imperfect manufacturing and alignment of the camera, causing the lens and image plane to be not exactly coplanar.

The distortions can be modeled by further nonlinear transformations from distorted image coordinates (u', v') to undistorted coordinates (u, v). The distortion at the center of distortion c_d is usually zero, but gets higher with increasing radius $r = \sqrt{u'^2 + v'^2}$. The radial distortion $(\Delta u_r, \Delta v_r)$ as correcting factor for (u', v') can me modeled using an even polynomial

$$\begin{aligned} \Delta u_r &= (1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + ...) \\ \Delta v_r &= (1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + ...). \end{aligned} \tag{2.8}$$

Additionally, the tangential distortion $(\Delta u_t, \Delta v_t)$ as correcting offset can be modeled using a slightly different polynomial with tangential distortion coefficients p_i

$$\begin{aligned} \Delta u_t &= (p_1(r^2 + 2u'^2) + 2p_2 u'v')(1 + p_3 r^2 + ...) \\ \Delta v_t &= ((2p_1 u'v') + p_2(r^2 + 2v'^2))(1 + p_3 r^2 + ...). \end{aligned} \tag{2.9}$$

The undistortion of image points (u', v') can now be expressed as:

$$\begin{aligned} u &= u' \cdot \Delta u_r + \Delta u_t \\ v &= v' \cdot \Delta v_r + \Delta v_t. \end{aligned} \tag{2.10}$$

Depending on the distortion of the lens, usually the five distortion coefficients k_1, k_2, k_3, p_1, p_2 are used.

2.2.3 Spherical Camera Model

Contrarily to normal lenses, fish eye lenses have a very wide field of view (FOV) and high radial distortions, so that the perspective model often is not sufficient to model the projection. Instead of mapping 3D points to a 2D image plane, the points are mapped to a spherical projection surface. For simplicity, a sphere

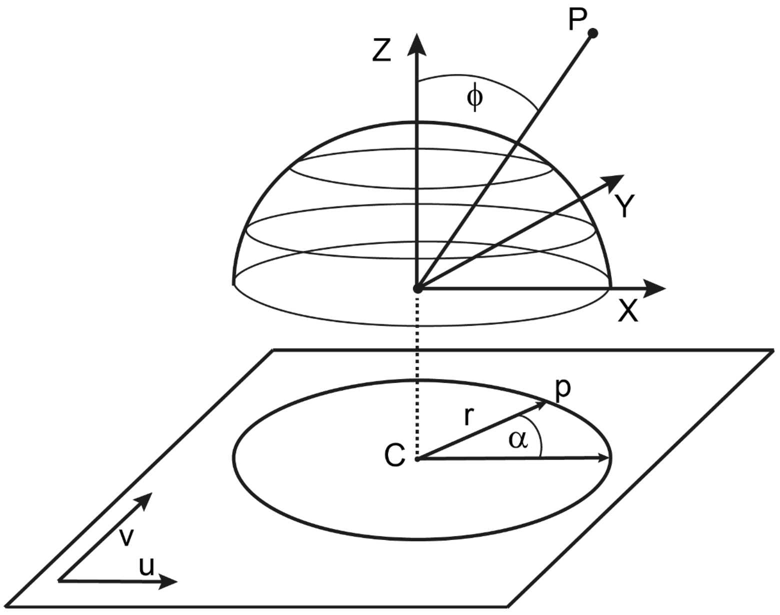


Figure 2.4: Fish eye camera model: a 3D point $P = (X, Y, Z)$ is projected to normalized image coordinates $p = (u, v)$ depending on radius r and angles α, ϕ.

with radius one is often used: the unit sphere $\mathbb{S}^2 = \{P \in \mathbb{R}^3 | \|P\| = 1\}$. The transformation from 3D world coordinates to 3D camera coordinates does not change and is equal to Equation (2.5). The main difference to the perspective transformation is that, in order to project a point P on the unit sphere, we now divide by the distance from the COP instead of division by the Z-coordinate as in Equation (2.1):

$$\begin{pmatrix} u \\ v \end{pmatrix} = \frac{f}{\|X\|} \begin{pmatrix} X \\ Y \end{pmatrix} = \frac{f}{\sqrt{X^2 + Y^2 + Z^2}} \begin{pmatrix} X \\ Y \end{pmatrix}. \tag{2.11}$$

There exists a variety of different projection functions, that describe the mapping from the unit sphere to an image plane. An overview of various projection functions and their mappings from camera coordinates to normalized image coordinates and backwards is shown in Table 2.1. For example, one of the models of Abraham and Foerstner (2005) models image coordinates (u, v) dependent on angles α and ϕ, and on the center of projection (c_x, c_y). Image coordinates are then calculated by

$$\begin{aligned} u &= c_x \cos[\alpha] r'[\phi] + c_x + \Delta u \\ v &= c_y \sin[\alpha] r'[\phi] + c_y + \Delta v \end{aligned} \tag{2.12}$$

with r' as radial projection function (see Table 2.1) and $(\Delta u, \Delta v)$ as correction offset depending on the chosen distortion model.

projection model	from camera coordinates to normalized image coordinates $(X,Y,Z) \to (x',y')$	from normalized image coordinates to camera coordinates $(x',y') \to (X,Y,Z)$
perspective $r' = f\tan(\phi)$	$x' = \frac{X}{Z}$ $y' = \frac{Y}{Z}$	$X = x'$ $Y = y'$ $Z = 1$
stereo-graphic $r' = f\tan(\phi/2)$	$x' = \frac{X}{\sqrt{X^2+Y^2+Z^2}+Z}$ $y' = \frac{Y}{\sqrt{X^2+Y^2+Z^2}+Z}$	$X = \frac{2x'}{1+x'^2+y'^2}$ $Y = \frac{2y'}{1+x'^2+y'^2}$ $Z = \frac{1-(x'^2+y'^2)}{1+x'^2+y'^2}$
equi-distant $r' = f\phi$	$x' = \frac{X}{\sqrt{X^2+Y^2}} \arctan\left(\frac{\sqrt{X^2+Y^2}}{Z}\right)$ $y' = \frac{Y}{\sqrt{X^2+Y^2}} \arctan\left(\frac{\sqrt{X^2+X^2}}{Z}\right)$	$X = \frac{x'}{\sqrt{x'^2+y'^2}} \sin\left(\sqrt{x'^2+y'^2}\right)$ $Y = \frac{y'}{\sqrt{x'^2+y'^2}} \sin\left(\sqrt{x'^2+y'^2}\right)$ $Z = \cos\left(\sqrt{x'^2+y'^2}\right)$
orthogonal $r' = f\sin(\phi)$	$x' = \frac{X}{\sqrt{X^2+Y^2+Z^2}}$ $y' = \frac{Y}{\sqrt{X^2+Y^2+Z^2}}$ $(Z > 0)$	$X = x'$ $Y = y'$ $Z = \sqrt{1-(x'^2+y'^2)}$
equi-solid-angle $r' = f\sin(\phi/2)$	$x' = \frac{X}{\sqrt{2(X^2+Y^2)}}\sqrt{1-\frac{Z}{\sqrt{X^2+Y^2+Z^2}}}$ $y' = \frac{Y}{\sqrt{2(X^2+Y^2)}}\sqrt{1-\frac{Z}{\sqrt{X^2+Y^2+Z^2}}}$	$X = 2x'\sqrt{1-(x'^2+y'^2)}$ $Y = 2y'\sqrt{1-(x'^2+y'^2)}$ $Z = \sqrt{1-(x'^2+y'^2)}$

Table 2.1: Projection models for fish eye lenses in terms of a radial projection function r'.

2.3 Rigid Body Motion

A rigid body motion is a Euclidean transformation, consisting of a rotation R and translation t, that preserves distances and angles of the transformed object. Rigid body motions form the so-called special Euclidean transformations and form the so-called special Euclidean group $SE(3)$. In the three-dimensional space we define a rigid body motion as a mapping g:

$$g : \mathbb{R}^3 \to \mathbb{R}^3; \tag{2.13}$$

$$g(x) = Rx + t. \tag{2.14}$$

In homogeneous coordinates we can express the rotation and translation together in one transformation matrix $T \in \mathbb{R}^{4\times4}$:

$$g\begin{pmatrix} x \\ 1 \end{pmatrix} = Tx = \begin{pmatrix} R & t \\ 0^T & 1 \end{pmatrix}\begin{pmatrix} x \\ 1 \end{pmatrix} \tag{2.15}$$

and its inverse T^{-1} as

$$g^{-1}\begin{pmatrix} x \\ 1 \end{pmatrix} = T^{-1}x = \begin{pmatrix} R^T & -R^T t \\ 0^T & 1 \end{pmatrix}\begin{pmatrix} x \\ 1 \end{pmatrix} \tag{2.16}$$

By using this matrix representation, we can concatenate multiple rigid body motions by left multiplication of consecutive transformations:

$$T_{i+1} = T_i T_{i-1} \cdots T_0 \tag{2.17}$$

There is a variety of representations for three-dimensional rotations, including rotation matrices, quaternions, Euler-angles or the axis-angle representation. The representation of a three-dimensional rotation as 3×3 matrix is very common and has the advantage that rotations can be concatenated by matrix multiplication. A major drawback is that with nine parameters and only three degrees of freedom, rotation matrices are over-parametrized. For minimization problems using numerical optimization it is useful to greatly reduce the effort and use a minimal representation of rotation and translation in a rigid body motion. We achieve this by using the associated Lie Algebra $se(3)$ of the Lie Group $SE(3)$. Every transformation $T \in SE(3)$ has a corresponding parameter vector $\xi = (\nu, \omega)^T \in se(3)$, consisting of the translational velocity $\nu = (\nu_1, \nu_2, \nu_3)^T$ and the angular velocity $\omega = (\omega_1, \omega_2, \omega_3)^T$. These so called twist coordinates have 6 parameters for 6 degrees of freedom, and thus, form a minimal representation of a rigid body motion.

The mapping from twist coordinates ξ to the corresponding Lie Group is calculated using the exponential map:

$$\begin{aligned} exp: se(3) &\rightarrow SE(3) \\ G(\xi) = \exp^{\hat{\xi}} &\mapsto T \end{aligned} \tag{2.18}$$

where $\hat{\xi}$ is the following 4×4 matrix, also known as twist:

$$\hat{\xi} = \begin{bmatrix} [\omega]_\times & \nu \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 & \nu_1 \\ \omega_3 & 0 & -\omega_1 & \nu_2 \\ -\omega_2 & \omega_1 & 0 & \nu_3 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{2.19}$$

The matrix exponential of twist $\hat{\xi}$ is then computed as

$$\exp^{\hat{\xi}} = \begin{bmatrix} \exp^{[\omega]_\times} & V\nu \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \tag{2.20}$$

with

$$\begin{aligned} \exp^{\hat{\xi}} &= I + \frac{\sin(\|\omega\|)}{\|\omega\|}[\omega]_\times + \frac{1-\cos(\|\omega\|)}{\|\omega\|^2}[\omega]_\times^2 \\ V &= I + \frac{1-\cos(\|\omega\|)}{\|\omega\|^2}[\omega]_\times + \frac{\|\omega\| - \sin(\|\omega\|)}{\|\omega\|^3}[\omega]_\times^2 . \end{aligned} \tag{2.21}$$

Elements from the Lie Group are mapped to elements of their Lie Algebra by the logarithmic map:

$$\begin{aligned} log: SE(3) &\rightarrow se(3) \\ \hat{\xi} &= \log(T) \end{aligned} \tag{2.22}$$

with

$$log\left(\begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}\right) = \begin{bmatrix} [\omega]_\times & V^{-1}t \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} [\omega]_\times & \nu \\ 0 & 0 \end{bmatrix} \tag{2.23}$$

where

$$[\omega]_\times = \frac{\phi}{2\sin\phi}\left(R - R^T\right) \tag{2.24}$$

with ϕ satisfying $1 + 2\cos\phi = Tr(R)$ and

$$V^{-1} = I - \frac{1}{2}[\omega]_\times + \frac{2\sin\|\omega\| - \|\omega\|(1 + \cos\|\omega\|)}{2\|\omega\|^2 \sin\|\omega\|}[\omega]_\times^2 . \tag{2.25}$$

2.4 Visual SLAM

In visual SLAM systems, localization and mapping is performed using inputs from one or more cameras only. Visual SLAM methods are often built upon visual odometries. In visual odometry problems we seek the camera motion between subsequent frames given an input image stream. More formally, visual odometry is "the process of estimating the egomotion of an agent [..] using only the input of a single or multiple cameras attached to it" (Scaramuzza and Fraundorfer, 2011). Historical, visual odometry evolved from structure from motion problems. Structure from motion deals with simultaneously estimating the relative camera motion and 3D structure of the environment from a set of camera images, which do not necessary need to be ordered. Visual odometry can be described as sub-case of structure from motion, which only deals with estimating the relative camera poses. The term visual odometry has been first introduced by the work of Nister et al. (2004), relating it to wheel odometry, that similarly estimates the egomotion incrementally using, for example, rotary encoders. While earlier structure from motion problems focused on an accurate offline optimization, this work proposed the first real-time camera pose estimation over a long trajectory. They reduce the computational complexity by using, for example, feature matching instead of tracking, 3D-to-2D motion estimation instead of 3D-to-3D and RANSAC for outlier detection. In contrast to wheel odometry, visual odometry does not suffer from wheel slip on difficult terrains, but needs sufficient illumination, texture and scene overlap to work robustly. When these assumptions are fulfilled, visual odometry has been shown to be more accurate than wheel odometry (Scaramuzza and Fraundorfer, 2011) and is especially well-suited in GPS-denied environments as for underwater or aerial vehicles.

Equipped with a monocular or stereo camera, the agent moves through the environment and captures images at subsequent discrete time-steps i. We define the camera motion as rigid body transformation consisting of a rotational and translational part. Therefore, the camera motion between two (mono- or stereo-) frames can be described by the rigid body transformation $T_{i,i-1} \in SE(3)$:

$$T_{i,i-1} = \begin{bmatrix} R_{i,i-1} & t_{i,i-1} \\ 0 & 1 \end{bmatrix} \tag{2.26}$$

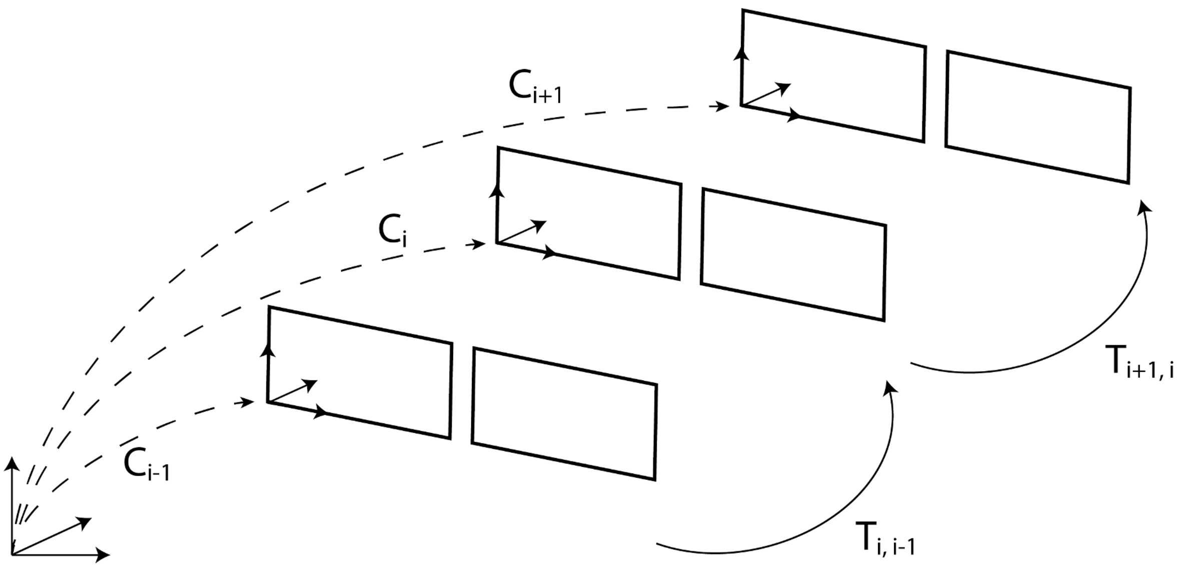


Figure 2.5: Visual odometry incrementally updates the camera pose estimate. Absolute camera poses C_i with regards to a global coordinate system, can be obtained by concatenating all previous estimated relative transformations $T_{1,0} \cdots T_{i,i-1}$ starting from an initial pose C_0.

Given the transformation between subsequent frames, the camera pose C_i at time-step i can be computed incrementally by concatenating all previous transformations:

$$C_i = T_{1,0} \cdots T_{i,i-1} = C_{i-1} T_{i,i-1}. \tag{2.27}$$

The initial camera pose C_0 can be set arbitrarily by the user, and is usually chosen as the identity matrix. A visualization of this transformation chain is given in Figure 2.5.

As the pose is updated incrementally, the error on previous pose estimates is accumulated over time. Additionally, uncertainties in the motion estimates further increase the uncertainty of new poses. Those uncertainties lead to drift over time, causing the estimated trajectory to differ from the real path. It is therefore necessary to keep the uncertainties of the camera transformations small, to decelerate the drift.

In contrast, visual SLAM systems use optimization techniques as global pose-graph optimization or bundle adjustment to counteract drift. Both methods are commonly used in visual SLAM systems to obtain a globally consistent trajectory. Bundle adjustment is often used locally to optimize over a window of the last m images, to ease the computational effort. It tries to optimize the camera motion, while optimizing the position of 3D landmark detections at the same time. Moreover, it only considers a local window of n images, instead of optimizing over all camera poses. Thereby, it is possible to reduce the computational amount by choosing a smaller window, which makes real-time bundle adjustment tractable. The optimization of the camera poses and landmark parameters is done by minimizing the reprojection error of known 3D landmarks X^i:

$$\underset{X^i, C_k}{\arg\min} \sum_{i,k} \left\| p_k^i - g(X^i, C_k) \right\|^2 , \tag{2.28}$$

where p_i^k is the point corresponding to landmark X^i in image k and $g(x^i, C_k)$ is the reprojection of the same landmark according to the current camera position C_k. In bundle adjustment the world is represented through sparse feature points.

When using pose-graph optimization the world is represented as a graph consisting of the camera poses as nodes that are connected with edges, representing the rigid body camera motion. If additional transformations between images are found, they can be added as further edge constraints to the graph. So-called loop closures, as visualized in Figure 2.7, occur when the agent reobserves places that he already visited before. This allows to draw inferences about accumulated drift, that can then be corrected by defining a cost function over all edge constraints e_{ij}:

$$\sum_{e_{ij}} \left\| C_i - T_{e_{ij}} C_j \right\|^2 , \tag{2.29}$$

which can be minimized by using, for example, nonlinear optimization algorithms to find a global consistent trajectory.

Besides, another possibility to reduce the drift is to incorporate measurements from other sensors, for example, from an IMU, GPS or laser scanner, and filter the result. Especially, for monocular versions additionally sensor measurements are necessary to recover the absolute scale of the scene.

While the main goal of visual odometry is to compute the relative camera motion $T_{i,i-1}$ incrementally and to be locally consistent, visual SLAM aims not only at tracking the camera motion, but also at building a globally consistent map. As visual odometry does not keep a representation of the environment, or at least builds a local map over the last m frames, it is not possible to recognize landmarks, that have been visited earlier. These so-called loop closures are of great importance for reducing drift. A great advantage of SLAM systems is the ability to detect loop closures and to maintain a globally consistent map. In general, visual odometry methods can be extended to SLAM by adding loop-closure detection and a global optimization scheme. Loop-closures are usually detected by doing a constraint search over a window of the nearest frames.

Methods for visual SLAM can be classified into feature-based and direct meth-

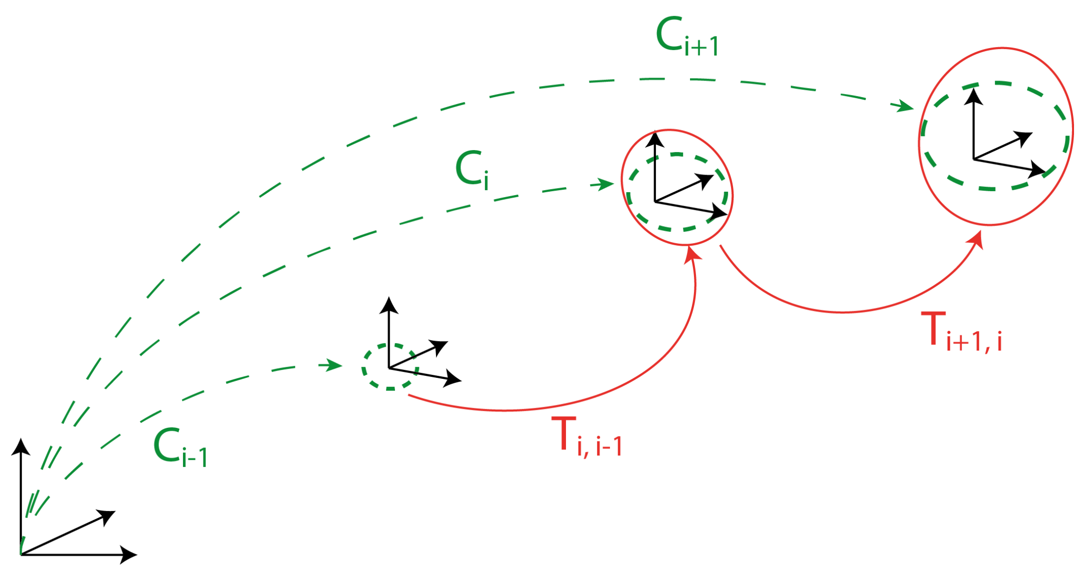


Figure 2.6: Visual odometry methods suffer from pose-drift due to noise and uncertainties in estimates of the previous pose (green dashed ellipses) and of the last transformation (red solid ellipses).

ods. Both approaches will be discussed in the following sections.

2.4.1 Feature-based Methods

Feature-based methods estimate the camera motion by using sparse interest points. The features are either tracked or matched over subsequent frames. The general scheme of these methods can be generalized as follows:

1. Image Acquisition from single or multiple cameras
2. Image Correction (Distortion, Rectification)
3. Feature Detection using Interest Point Operator
4. Feature Tracking or Matching
5. Motion Estimation (3D-3D, 3D-2D, 2D-2D)
6. Mapping
7. Optimization

In the first two steps the images of a mono or stereo camera are captured, undistorted and—in the case of stereo cameras—rectified to speed up the correspondence search. Afterwards, the image is searched for salient points, so-called feature or interest points. There is a variety of different feature point detectors that detect corners (Förstner, 1986; Harris and Pike, 1988; Rosten and Drummond, 2006) or blobs (Bay et al., 2008; Calonder et al., 2012; Lowe, 2004) in the image (Scaramuzza and Fraundorfer, 2011). A good feature detector is characterized by repeatability, localization accuracy, robustness, distinctiveness, invariance to photometric and geometric changes and computational efficiency. The choice of the feature detector heavily depends on the application. While corner detectors are fast to compute, blob detectors are often slower. On the other hand blob detectors are more distinctive than corner detectors.

Therefore, the choice of the right detector is often a trade-off between fast or more distinct matching.

In the fourth step the detected feature points are either tracked or matched in the next frame. Tracking describes the local search for the same detection in the next images, while matching individually detects features in both images and then tries to find the best matches using a similarity measure (SSD, NCG). Both approaches have their pros and cons: Feature tracking is best used when the motion between frames is small, to keep the search range feasible (small-scale environments). On the other hand feature matching is well-suited for wide-baseline approaches and large motions (large-scale).

The given correspondences are then used in a fifth step to compute the relative camera motion. In general, there are three different approaches, depending on the feature representation:

- **3D-to-3D:** features are represented as 3D world coordinates. The camera motion can be determined by finding the transformation that best aligns both point sets, for example, by minimizing over the L_2 distance (ICP).
- **3D-to-2D:** features of the first frame are specified as 3D world coordinates, while features of the second frame are represented as 2D image coordinates. The optimal camera motion is found by minimizing the reprojection error.
- **2D-to-2D:** features are represented as 2D image coordinates. The camera motion is computed by estimating the essential matrix and afterwards extracting R and t from it.

In the next step these feature correspondences are maintained in a map, so that newly tracked frames can be localized with regards to previous feature observations.

The final optimization step addresses the problem of accumulating drift over time. In addition to the constraints found by localizing to feature points in the current map, it is often possible to match features between frame k and the last

n frames. These constraints help to minimize the accumulated drift and can be either inserted in a global pose-graph optimization, or in a bundle adjustment to simultaneously the landmarks positions in the map.

2.4.2 Direct Methods

In contrast to feature-based methods that compute the camera motion based on a sparse abstraction of feature observations, direct methods recover the camera motion directly from the intensities of the pixels in the image. They do not need to extract feature points and track or match them between frames, which both can be computationally demanding. Instead of using a sparse set of feature points as representation of the image, direct methods use the brightness information from all pixels in the image. The underlying assumption is the so-called photo-consistency constraint (or brightness constancy equation), which states that, given a world point $P_W = (X, Y, Z)^T$, the projections of this point in two frames I_k and I_{k+1} have the same intensity value in both images:

$$I_k(p) = I_{k+1}(p'), \tag{2.30}$$

where p and p' are the projections of P_W in the first and in the second frame respectively.

Assuming that the photo-consistency holds for all pixels of the image, this leads to much more constraints than in the feature-based methods: For example, even an image with VGA resolution (640×480) provides about 300.000 constraints, compared to about 100 to 200 feature observations.

Ideally, Equation (2.30) always holds, but in reality one has to deal with different lighting conditions, sensor noise, pose-errors, non-Lambertian surfaces and dynamic objects, so that the photo-consistency assumption is often violated. In other words, this means that the residual

$$r = \|I_k(p) - I_{k+1}(p')\| \tag{2.31}$$

will be non-zero.

Therefore, the idea is to find a camera motion ξ, that minimizes the residual, also called the photometric error. Each pixel contributes to the minimization by adding one constraint. The constraints from all pixels are combined to minimize the photometric error:

$$E = \sum_{p \in \Omega} \|I_k(p) - I_{k+1}(\pi(\xi, P))\|, \tag{2.32}$$

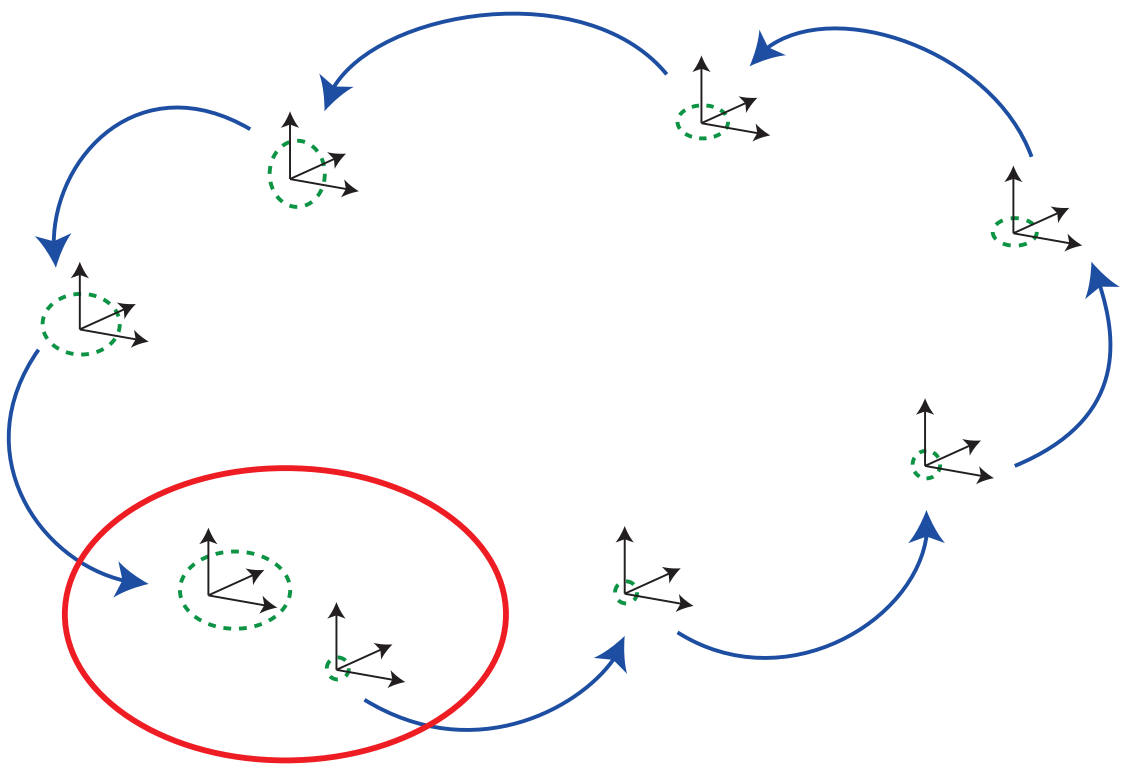


Figure 2.7: Loop constraints occur when a already visited scene is reobserved as highlighted by the red ellipse. When loop closures are detected, the resulting edges of the estimated transformation between involved frames are added as further constraints to the pose-graph optimization.

where $\pi(\xi, P)$ is the projection function, which warps a 3D Point P transformed by the camera motion ξ to image coordinates, as in Equation (2.2). The error function can then be solved numerically, using, for example, the Gauss-Newton algorithm. For using a numerical optimization algorithm to estimate the camera motion parameters it is efficient, if the number of parameters is kept minimal. Thus, the camera motion $\xi = (v_1, v_2, v_3, \omega_1, \omega_2, \omega_3,)^T$ is represented as 6D vector containing the twist parameters. As the camera motion is nonlinear in the rotation, the error function E is linearized using a second-order approximation.

One great advantage of direct methods is the sub-pixel accuracy that comes from the over-parametrized equation system: given constraints from all pixels and only six parameters of ξ to estimate, these parameters can be estimated with very high precision. This means that usually the misalignment error is less than 0.1 pixel (Irani and Anandan, 2000). By using confidence weighted local constraints, for example, by weighting pixel differently depending on their gradient magnitude, image regions with homogeneous texture contribute less to the minimization than regions with sufficient texture. Direct methods work best for small motions of less than one pixel and sufficient image overlap. However, it has been shown that using a coarse-to-fine refinement, starting with a high level of an multi-resolutional image pyramid and refining on lower levels, greatly increases the success on large motions, making it possible to track motions up to 10-15 percent of the image size (Irani and Anandan, 2000). This range can be further increased by handing over good initial estimates to the minimization step. Moreover, when using a coarse-to-fine approach, direct methods lock to the dominant motion present on the coarse level and are thereby robust to outliers or multiple motions in the images.

On the other hand, when compared to feature-based approaches, direct methods are computationally very demanding as they deal with thousands of more constraints. Additionally, wide-baseline comparisons are very challenging for direct methods and often an initial estimate is needed. Changes in image brightness can affect the results, as the brightness constancy assumption is violated and tracking may fail.

3 Related Work

Visual SLAM continuously estimates the motion of a vehicle or robot using image information alone, while—in parallel—building a map of the environment. There exists a wide variety of different visual SLAM methods starting with monocular methods, that use only a single camera, to stereo or multi-camera methods, that use two or more cameras. The visual odometry tutorials by Scaramuzza and Fraundorfer (2011) give a brief overview of the research conducted in the last decades. As in the early stages only limited processing power was available, most methods were executed offline. With increasing processing power real-time computation became possible and the motion of a robot could now be tracked online.

Visual SLAM is particularly important for autonomous navigation of MAVs, since cameras are the preferred sensor due to their size and weight.

Ross et al. (2013) use a monocular camera for obstacle avoidance with a MAV in cluttered forest environment. As the detection of frontal obstacles with a single camera is challenging, Mori and Scherer (2013) employ a relative size detector for obstacle avoidance. Since monocular methods suffer from scale ambiguity, visual-inertial methods have become popular for the use on MAVs (Weiss, Achtelik, et al., 2012). Recent methods for visual-inertial odometry also include stereo cameras (Schmid et al., 2014) or multiple cameras (Schauwecker and Zell, 2014). The existing approaches can often be split into sparse feature-based and dense direct visual SLAM methods, as described in Section 2.4:

3.1 Feature-based Methods

Feature-based methods abstract the images to a sparse set of feature correspondences. The general pipeline for feature-based methods consists of feature detection and extraction, feature matching between subsequent frames, egomotion estimation based on the estimated feature correspondences as well as mapping and optimization. At the beginning, features are detected in the incoming images and are then either matched or tracked over time. Based on the feature correspondences between subsequent frames the relative motion between these frames is computed by minimizing the reprojection error. Stereo methods can retrieve

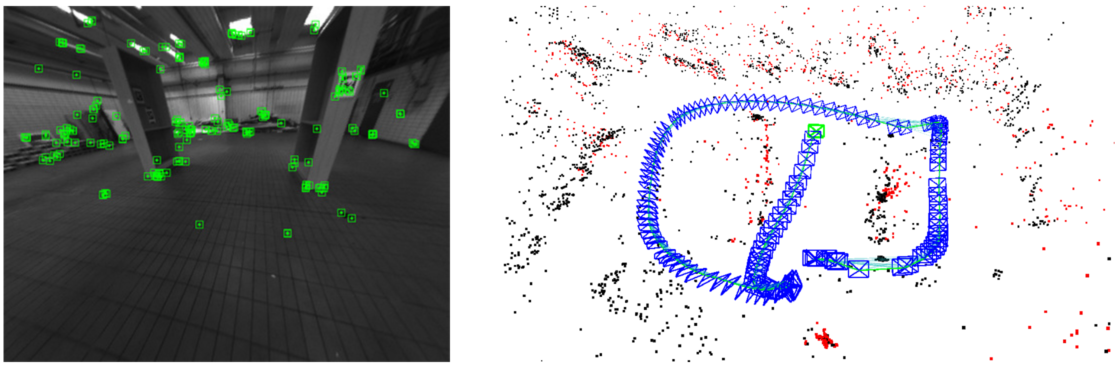


Figure 3.1: Feature-based SLAM methods abstract the image to a sparse set of feature detections. Left: ORB features (green) tracked by ORB-SLAM during a flight with our MAV. Right: corresponding sparse map of the environment built by ORB-SLAM including the retrieved key frame graph. Key frame positions are shown in blue, edge constraints in green and the landmarks of the map are drawn red (for active) and black (for currently inactive).

3D features and thus some methods also use ICP to align the 3D measurements of two subsequent stereo frames. In some methods the recent extracted features are also tracked to the existing map for further refinement and loop detection. Without loop closure Visual SLAM degrades to visual odometry, as the camera motion is updated incrementally from frame to frame, which tends to drift over time. Hence, the error of the motion estimates resulting from noise in the measurements are propagated to the next frame. This again leads to an increased uncertainty of the new camera pose, so that the error is accumulated over time.

Popular monocular feature-based methods are, for example, MonoSLAM (Davison et al., 2007) or Parallel Tracking and Mapping PTAM (Klein and Murray, 2007). PTAM is a widely used feature-based monocular SLAM method, which allows robust state estimation in real-time. It was first developed for AR-applications, but has been successfully used on MAVs with a monocular camera (Achtelik et al., 2011; Weiss, Scaramuzza, et al., 2011). PTAM was one of the first visual slam systems that introduced distinct threads for tracking and mapping, that run independently. While the tracking part continuously estimates the camera motion by matching FAST features (Rosten and Drummond, 2006) and compares them to the map, the mapping part updates the map with new key frames and optimizes the camera poses and point features by minimizing the reprojection error.

However, as a monocular method, PTAM requires sufficient camera translation parallel to the image plane as initialization pattern. The scale is then often initialized, so that the average distance between the camera position and feature points is one. Weiss, Scaramuzza, et al. (2011) employ PTAM on a MAV with a down-looking camera to achieve sufficient scene overlap. To allow for real-time performance, they greatly reduce the number of tracked feature points. As PTAM does not recover the absolute scale of the scene they integrate measurements from an IMU by fusing them with the visual measurements in an Extended Kalman Filter (EKF).

Recently, ORB-SLAM (Mur-Artal et al., 2015) has been proposed as a monocular visual SLAM system that tracks sparse ORB features (Rublee et al., 2011). Similarly to PTAM, it distributes the work along different threads: tracking, mapping and loop closing is done separately to enable real-time processing.

When using monocular methods, additional sensors are needed to observe the absolute scale of a scene. In contrast stereo methods for visual SLAM do not suffer from scale ambiguity. Stereo methods have been broadly used on ground vehicles, as, for example, for planetary rovers (Moravec, 1980) or autonomous driving cars (Nister et al., 2004). They have the great advantage that no additional sensory input is needed to observe the absolute scale of the scene. The egomotion estimation can be done by minimizing the distances of 2D image correspondences using the reprojection error or by aligning 3D point features using ICP.

In our work we rely on an efficient feature-based library for visual stereo odometry (Geiger, Ziegler, et al., 2011), that shows a good trade-off between accuracy and runtime.

3.2 Direct Methods

In contrast to feature-based methods, which abstract the images into a sparse set of feature points, direct methods use the whole image for tracking by minimizing the photometric error. The underlying brightness constancy equation is visualized in Figure 3.2: Given the projection p of a 3D Point P in the first image, the point can be warped into the second image using the relative camera motion between both images ξ. As the projections in both images represent the same 3D point P, ideally the intensity values of both points should be equal.

As direct methods minimize the photometric error over all pixels in the images, they are computationally very intense and thus much slower than feature-based methods. Usually, direct tracking can only recover small motions and needs sufficient image overlap. Therefore, often coarse-to-fine approaches are applied. Instead of tracking all pixels of the original image, a multi-resolution image pyramid is built. The tracking starts on a coarse resolution with small motion and is iteratively refined in finer levels of the pyramid. Thereby, even large motions can

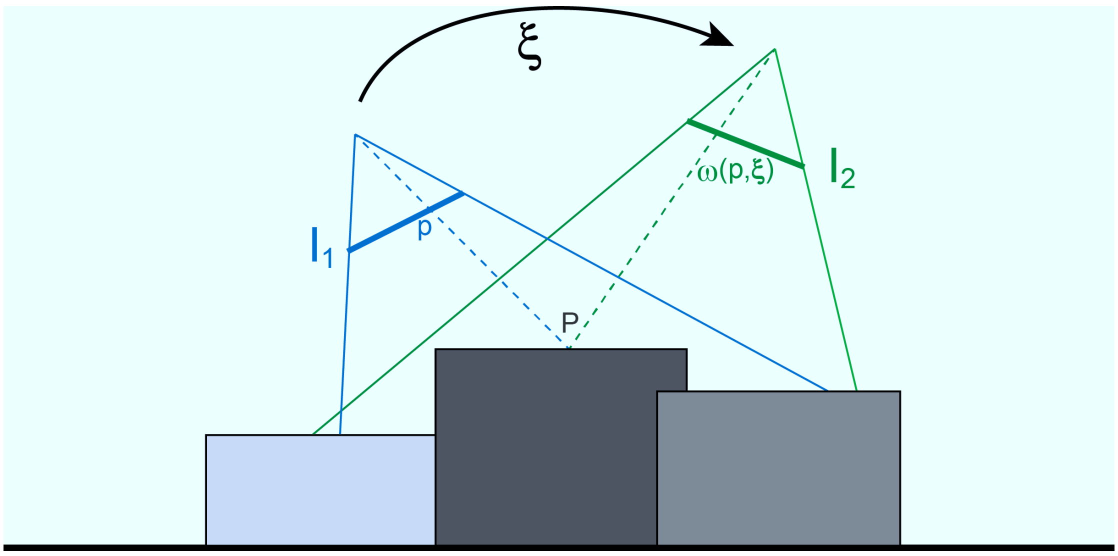


Figure 3.2: Direct image alignment assume that, given two projections, p and $\omega(p,\xi)$ of the same 3D Point P have identical intensity values (visualized as boxes with different color). Therefore, direct methods seek the camera motion ξ that minimizes the photometric error.

be tracked using direct alignment (Irani and Anandan, 2000). Moreover, with the coarse-to-fine approach direct methods are robust to dynamic motions in the image: even if multiple motions are present, direct methods lock to the most dominant motion. A great advantage of direct methods is, that every pixel contributes to the minimization of the error function. Thereby, the camera motion can be retrieved up to a very high precision. The estimation of the minimal camera pose $\xi \in se(3)$ is done using thousands of constraints. However, outliers can disturb the minimization and are often handled by using weighted residuals in the least square estimation.

Direct approaches go back to the work of Lucas and Kanade (1981) on 2D image alignment using global optimization techniques. Comport et al. (2007) extend their approach to 3D by defining a quadrifocal warping function, that warps points from a stereo reference pair to the current stereo pair. Hereby, the stereo reference pair has to be initialized with dense correspondences. In particular, the quadrifocal warping is composed of two trifocal tensors that depend on the intrinsic and extrinsic camera parameters and on the current relative motion. Both pairs are then aligned by minimizing the photometric error between the warped reference pair and the current pair using efficient second order approximation. They claim that their approach runs at 1.5 Hz on a resolution of 759×280, but would be real-time capable when dense correspondences for the reference pair are found online.

As the approach is based on warping 3D points into the next view, it has been adapted to RGB-D cameras (Kerl et al., 2013; Newcombe et al., 2011), as RGB-D cameras already come with depth measurements for each image. Hence, the costly stereo matching step can be skipped, as dense depth maps are already provided by the sensor. As the minimization over all pixels is very costly, recent approaches, that perform in real-time, often make use of heavy GPU parallelization.

A different way of reducing the workload, is to track only image points with sufficient information. These so-called semi-dense methods estimate depth only for pixels with sufficient gradient magnitude and have been proposed for monocular (Engel, Schöps, et al., 2014) and stereo cameras (Engel, Stueckler, et al., 2015). They directly track images and estimate a semi-dense representation of the environment in real-time using a CPU only.

3.3 Semi-direct Methods

There is little work regarding the combination of direct methods and feature-based methods. The existing approaches often use both methods separately by computing the camera motion of the whole trajectory using a feature-based approach, and afterwards performing an offline dense reconstruction of the scene using direct approaches. A recent semi-direct method uses a combination of both—direct and feature-based—methods for fast visual odometry on a MAV with a downward-looking camera (Forster et al., 2014). They distribute tracking and mapping to two separate threads as proposed by Klein and Murray (2007) and achieve frame rates up to 55 Hz. With the initialization of new key frames FAST features are detected at different image pyramid levels. New frames are then tracked towards the key frame using sparse image alignment on the extracted feature set. The computational complexity of the direct alignment is reduced by minimizing the photometric error on coarse pyramid levels only, and only for a subset of all pixels. Afterwards, the estimated motion is further refined by minimizing the reprojection error of the observed features, as this greatly reduces drift. The mapping thread continuously updates the depth estimates of the key frames' features with every new reobservation in the newly tracked frames. However, as the approach was designed for a downward-looking camera, it struggles with forward motion and pure rotations.

In contrast to this, we use the motion estimates of feature-based tracking as initialization for direct tracking of key frames. We thereby combine feature-based and direct tracking over time, taking advantage of the fast tracking from the feature-based method and the accurate alignment of a direct method.

4 Camera Calibration

In order to calibrate and rectify the fish eye stereo camera system, we evaluate different calibration methods. The goal of the calibration is to estimate the intrinsic and extrinsic parameters of the camera system and as well to rectify the images onto a plane, so that corresponding points lie on horizontal scan-lines. In particular we employed four different methods, which were publicly available and capable to deal with fish eye distortion. We start with the methods that use a 2D checkerboard as calibration target, then discuss the method that uses a 2D aprilboard and finally discuss the method that uses a 3D calibration target. The different calibration targets are shown in Figure 4.1. As a quality measure we compare the average reprojection error.

4.1 OpenCV

With the release of the newest version of OpenCV (3.0.0) support for images acquired with a fish eye camera has been added allowing for a wide-angle lens model. The model is based upon the pinhole projection model but uses a different distortion model. As described in Equation (2.5) 3D world coordinates P_w are projected into 3D camera coordinates P_c by a rigid body motion. The points are

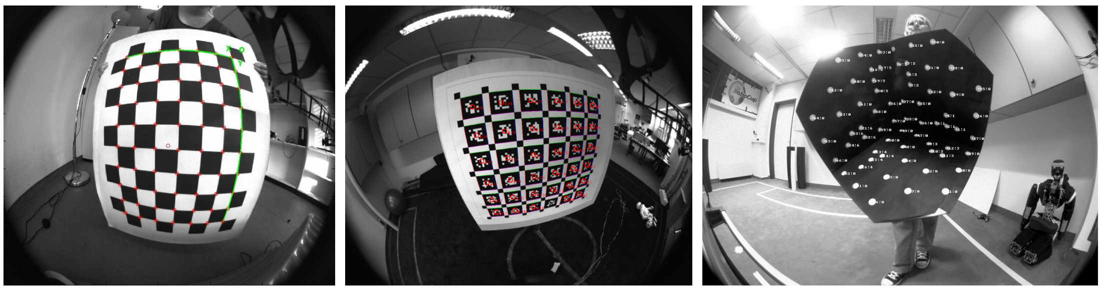

(a) Checkerboard **(b)** Aprilboard **(c)** 3D Point Target

Figure 4.1: Calibration targets for camera calibration. We evaluate fish eye calibration methods with 2D and 3D targets.

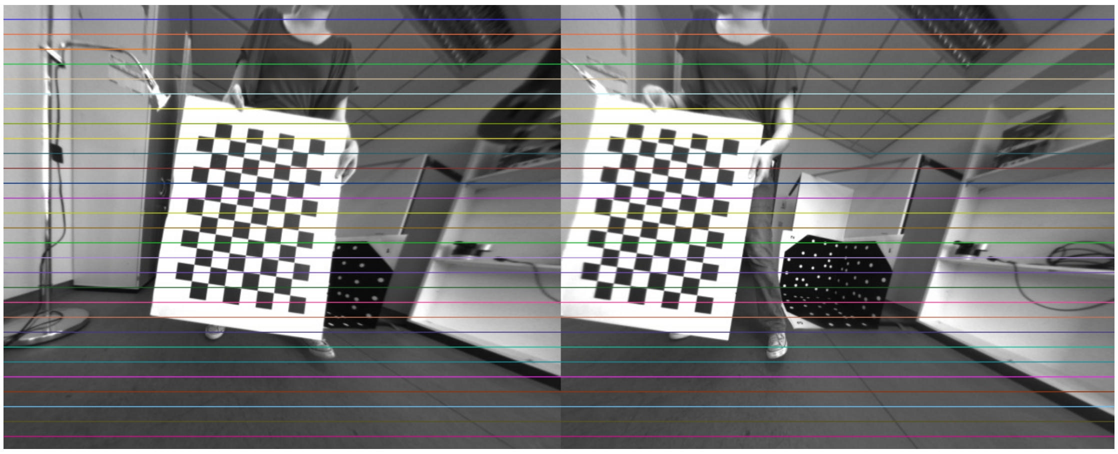


Figure 4.2: Left and right rectified images after fish eye stereo rectification with OpenCV. As can be seen, corresponding feature points lie on the same horizontal scan-line.

then projected on a planar surface, so that:

$$x' = \frac{X}{Z} \quad y' = \frac{Y}{Z}. \tag{4.1}$$

Furthermore, radius r and angle θ are defined as

$$\begin{aligned} r^2 &= x'^2 + y'^2 \\ \theta &= \operatorname{atan}(r). \end{aligned} \tag{4.2}$$

The fish eye distortion is modeled as higher-order polynomial of θ with distortion coefficients (k_1, k_2, k_3, k_4)

$$\theta_d = \theta(1 + k_1\theta^2 + k_2\theta^4 + k_3\theta^6 + k_4\theta^8). \tag{4.3}$$

The image coordinates of the distorted points (u', v') can be retrieved by

$$u' = \left(\frac{\theta_d}{r}\right) X_c \qquad v' = \left(\frac{\theta_d}{r}\right) Y_c \tag{4.4}$$

leading to undistorted coordinates (u, v)

$$u = f_x(u' + \alpha v') + c_x \qquad v = f_y v' + c_y \tag{4.5}$$

For a good calibration the calibration pattern has been presented to the camera in different positions and viewing angles. After stereo rectification, corresponding

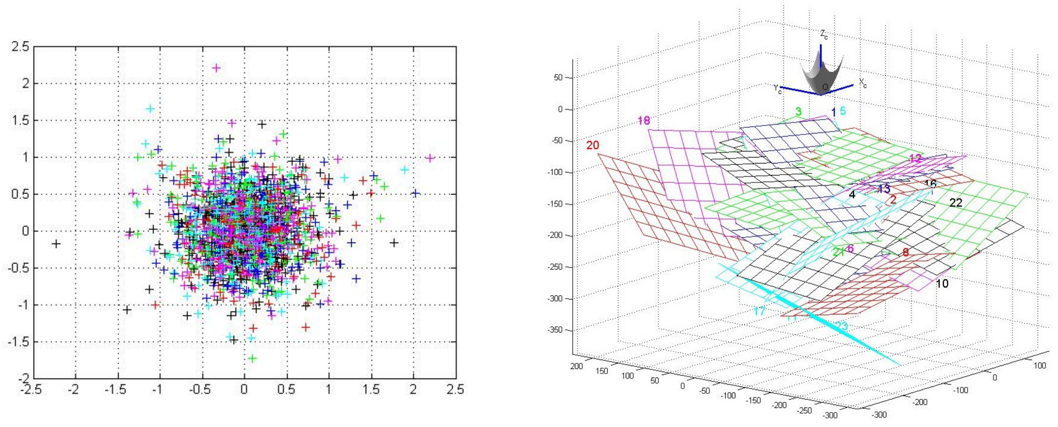


Figure 4.3: (a) Reprojection Error of the Omnidirectional Camera Calibration Toolbox. (b) Estimated extrinsics of the captured checkerboard planes used for calibration.

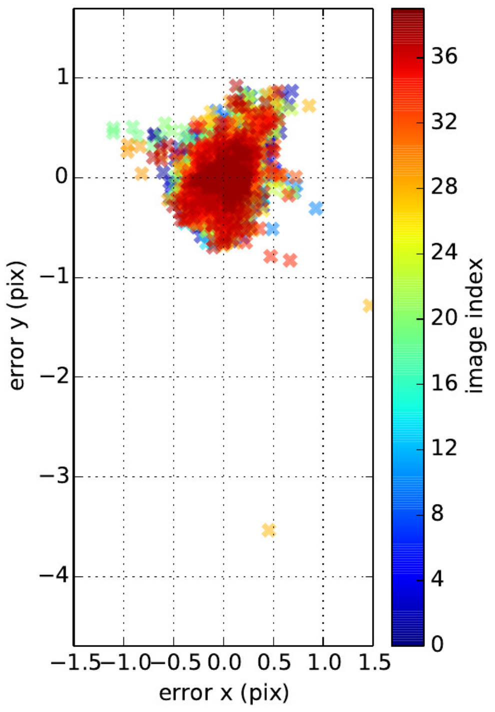
features can be found along horizontal scan-lines, as highlighted in Figure 4.2.

The average reprojection error with this calibration is 0.994 pixel and the retrieved baseline is 53.129 cm.

4.2 OCamCalib

As a second calibration method we evaluate the Omnidirectional Camera Calibration Toolbox for Matlab (OCamCalib) by Scaramuzza, Martinelli, et al. (2006). It can be used for the intrinsic calibration of catadioptric and fish eye cameras. Similarly to the OpenCV calibration, OCamCalib uses a 2D checkerboard as calibration target to estimate the lens parameters. The toolbox offers an automatic corner extraction, that—in our case—had problems in detecting all corners of the checkerboard. As a solid corner extraction has a severe influence on the quality of the calibration, we opted for the manual extraction of corners. When using manual corner detection, the user has to select the corners for each image manually. The toolbox offers assistance by using a corner detector for finding the best corner point around the selected point. Even though the manual extraction for every corner on every image is very time consuming, it yields the best result.

In the underlying omnidirectional model a 3D vector P is calculated from image



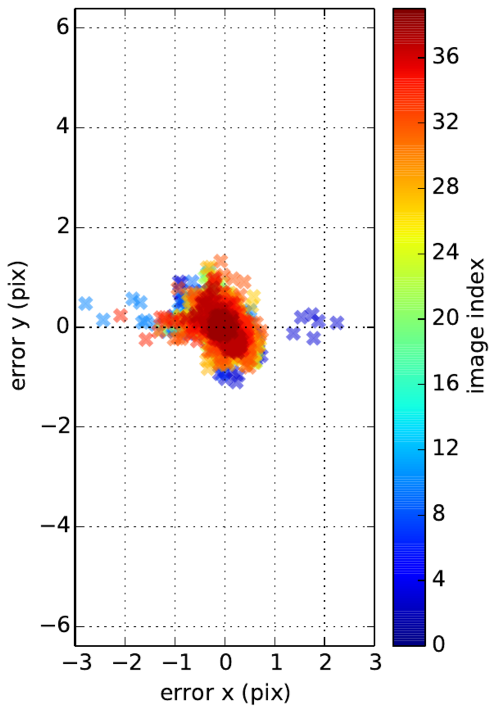


Figure 4.4: Reprojection error of Kalibr for left and right camera of a stereo system.

coordinates (u, v) as

$$P = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} u \\ v \\ f(r) \end{pmatrix}, \tag{4.6}$$

where the z coordinate is a function of the distance from the image point (u, v) to the image center—the radius $r = \sqrt{u^2 + v^2}$. The function $f(r)$, that backprojects the extracted grid corners from 2D to 3D, is defined as a higher-order polynomial, which coefficients $a_0, a_1, \ldots$ are estimated in the calibration procedure:

$$f(r) = a_0 + a_1 r + a_2 r^2 + a_3 r^3 + \ldots \tag{4.7}$$

We followed the authors advice to use a 4th order polynomial for calibration. The average reprojection error we got is 0.743 pixels. Reprojection errors and extrinsics are shown in Figure 4.3.

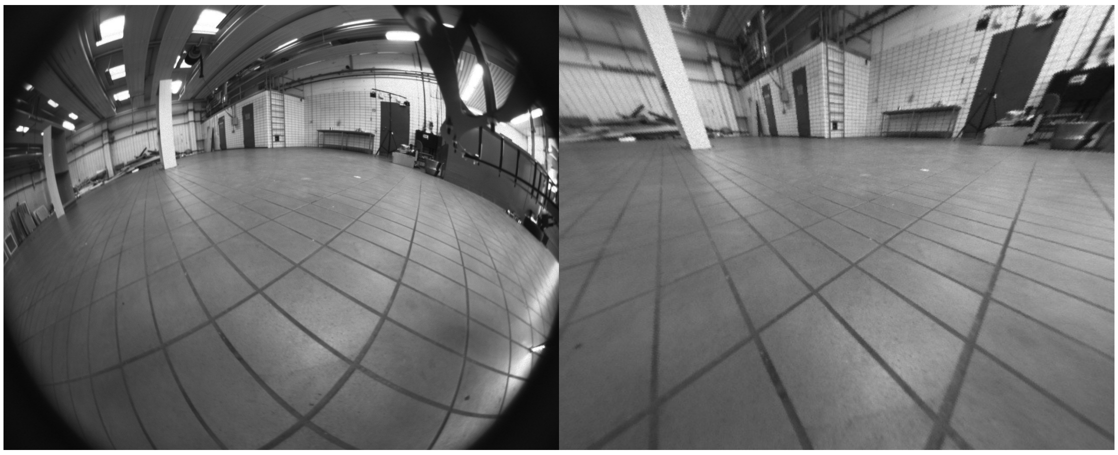


Figure 4.5: Stereo Rectification with TCC. Raw fish eye images (a) are rectified onto a plane with half the resolution (b).

4.3 Kalibr

The third calibration method we employ is Kalibr (Maye et al., 2013), a calibration toolbox for multi-camera calibration and camera to IMU calibration. Additionally to a 2D checkerboard, Kalibr also supports 2D aprilboards, consisting of multiple different apriltags, as calibration target. This bears the advantage that different apriltags—in contrast to the general checkerboard corners—can be distinguished individually, so that even only partially visible targets can be used. Moreover, the targets pose can be fully resolved, because there are no ambiguous poses as with a symmetric checkerboard. Therefore, the calibration procedure is more robust to images where the calibration target is only partly visible. Kalibr offers support for the standard pinhole model, as well as a generic omnidirectional camera model for wide-angle and fish eye lenses, described in Kannala and Brandt (2006). For data acquisition the MAV is fixed and the aprilgrid is moved in front of the stereo camera system with varying distances and angles. Unfortunately, the calibration did not converge for the pinhole camera model. When using the omnidirectional model the average reprojection error is 0.26 pixel, as shown in Figure 4.4, and the baseline was recovered to 53.364 cm.

4.4 TCC

As a fourth calibration method, we use the calibration toolbox TCC (**T**est field based self-**C**alibration of multi-**C**amera-systems) by Abraham (2004), that was especially developed for fish eye camera calibration. The software supports different projection models for standard and fish eye lenses. Contrarily to the other calibration methods, it uses a 3D calibration target with various points on three orthogonal planes (see Figure 4.1c), that is observed from different distances and angles. The approximate positions of the 3D targets have to be known. Afterwards, the intrinsic and extrinsic calibration parameters are estimated together with the 3D coordinates of the calibration target by bundle adjustment, formulated as least squares problem. Once the bundle adjustment is done, the user has the choice between different projection and distortion models. For modeling our cameras, we selected the epipolar equi-distant model for fish eye cameras, that describes the projection of a spherical image onto a plane as shown in Figure 2.4.

The lens distortion is described using a Cheby-chev polynomial as described in Equation (2.12). Afterwards, TCC generates look-up tables for the rectification onto a plane with horizontal epipolar lines. Overall, we get a reprojection error of 0.75 pixels and a baseline of 53.362 cm.

5 Approach

5.1 Overview

The approach presented in this thesis is based on LSD-SLAM (Engel, Schöps, et al., 2014) and LIBVISO2 (Geiger, Ziegler, et al., 2011). LSD-SLAM is a monocular direct SLAM method and thus not capable of retrieving the absolute scale of the scene. The contributions of this thesis are the extension of LSD-SLAM to stereo, which allows to estimate the absolute scale of the scene and greatly reduces scale drift. In addition, we combine the direct approach with a feature-based approach to be more robust to large inter-frame motions and strong rotations. We choose LIBVISO2 as feature-based method for visual odometry, because it shows a good trade-off between accuracy and runtime. In the following, we will first introduce the underlying methods LSD-SLAM and LIBVISO2. Afterwards, the stereo extension and semi-direct tracking will be described.

5.2 LSD-SLAM

The general processing pipeline of LSD-SLAM consists of three main components: Tracking, Depth Map Estimation and Global Map Optimization. A key frame at timestep i is represented as $KF_i = (I_i, D_i, V_i)$, consisting of the intensity image $I_i : \Omega \to \mathbb{R}$, the depth map $D_i : \Omega \to \mathbb{R}^+$ and the variance of the depth map $V_i : \Omega \to \mathbb{R}^+$, where $\Omega \in \mathbb{R}^2$ is the image space that maps pixel coordinates to their brightness value.

Tracking is based on maximizing photo-consistency and thus minimizing the photometric error between two frames using Gauss-Newton:

$$E(\xi) := I_{KF}(x) - I(\pi(p')) \tag{5.1}$$

where p' is the warped point from I to I_{KF} by ξ. New frames are tracked towards a key frame and the rigid body motion of the camera $\xi \in se(3)$ is estimated.

In the Depth Map Estimation tracked frames are then used to refine the existing depth map of the key frame by many small baseline stereo comparisons: with each

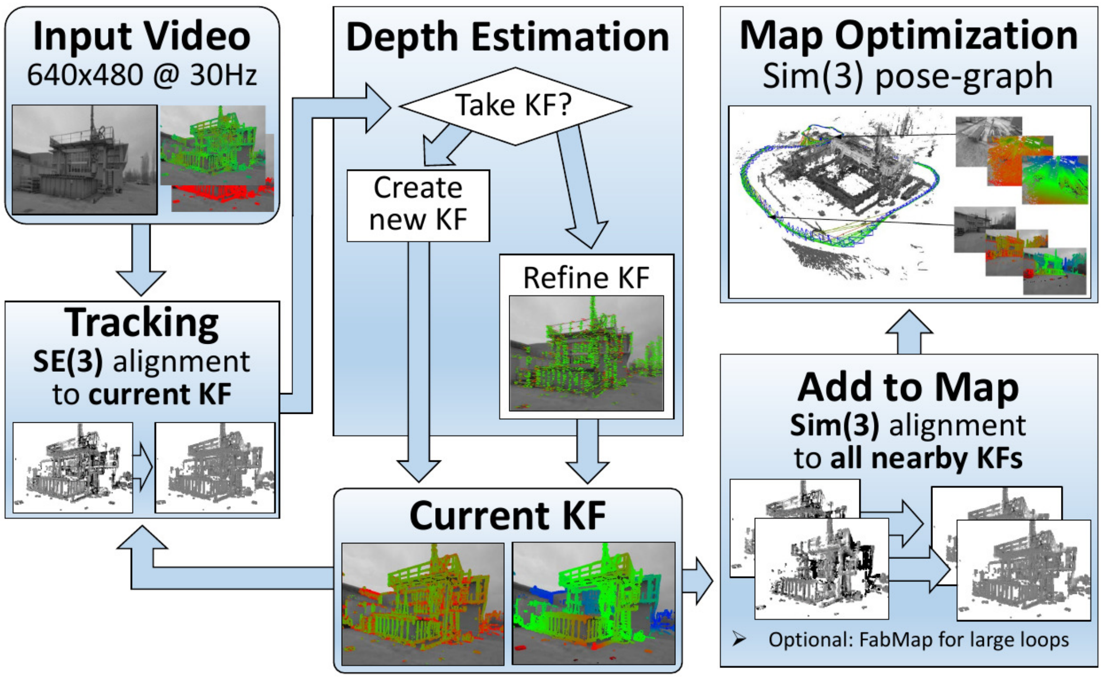


Figure 5.1: Overview LSD-SLAM (Engel, Schöps, et al., 2014): In LSD-SLAM incoming frames are always tracked towards the current key frame. If tracking succeeds, they either refine or replace the current key frame. If a key frame got replaced, and hence, will not be further refined, it is added to the pose graph. A constraint search on nearby key frames is performed to add further edge constraints for to the graph for optimization and drift compensation.

new tracked frame the depth map of the key frame is refined, by either creating new depth hypotheses or improving existing ones. New key frames are created when the distance exceeds a certain threshold and are initialized by propagating depth of the previous key frame towards the new candidate. Once a key frame is replaced it is added to the pose-graph for further refinement and loop closing. As for monocular methods no instant depth information is available, the first frame is initialized with random depth and large variance. Afterwards, the frame is added to the pose-graph as initial key frame. The following frames are tracked using the random initialization and with sufficient translation the algorithm locks to a certain camera movement and the depth measurements are refined.

5.2.1 Tracking

New frames are always tracked with regards to the current key frame and its depth map. Given the current key frame $KF_i = (I_i, D_i, V_i)$ the relative pose $\xi \in se(3)$ is computed by minimizing the photometric error $E(\xi)$. This is done by warping the image points from the key frame to the tracked frame using their existing depth hypothesis z:

$$w(p, z, \xi) := \exp(\xi) \begin{pmatrix} \frac{u - c_x}{f_x} \\ \frac{v - c_y}{f_y} \\ z \end{pmatrix} \tag{5.2}$$

Given the projection in the first and second frame their intensity difference form the residual r_p:

$$r_p(p, \xi_{ji}) = I_{KF_i}(p) - I_j(\pi(w(p, D_i(p), \xi_{ji}))) \tag{5.3}$$

The uncertainty of the depth estimates is incorporated by normalizing the photometric error $E(\xi_{ji})$ with variance $\sigma^2_{r_p(p,\xi_{ji})}$:

$$E_p(\xi_{ji}) = \sum_{p \in \Omega_{D_i}} \left\| \frac{r_p^2(p, \xi_{ji})}{\sigma^2_{r_p(p,\xi_{ji})}} \right\|_\delta \tag{5.4}$$

$$\text{with } \sigma^2_{r_p(p,\xi_{ji})} := 2\sigma_I^2 + \left(\frac{\partial r_p(p, \xi_{ji})}{\partial D_i(p)} \right)^2 V_i(p), \tag{5.5}$$

where σ_I^2 is defined as Gaussian image intensity noise and $\| \cdot \|_\delta$ is the Huber norm, used to down-weight strong outlier:

$$\|r^2\|_\delta := \begin{cases} \frac{r^2}{2\delta} & \text{if } |r| \leq \delta \\ |r| - \frac{\delta}{2} & \text{otherwise.} \end{cases} \tag{5.6}$$

The photometric error is minimized with regards to the relative camera motion $\xi_{ji} \in se(3)$ using weighted Gauss-Newton optimization. The optimal camera pose is the camera pose ξ^* which minimizes the photometric error:

$$\xi^* = \arg\min_\xi E_p(\xi). \tag{5.7}$$

The camera motion of the previously tracked frame serves as initialization for the optimization. The tracking is done using a coarse-to-fine scheme to cope with large motions. Starting on a high pyramid level corresponding to a coarse resolution, the result is used as initial estimate on the following pyramid level.

Afterwards, the new tracked frame is either chosen to become a new key frame or to refine the depth map of the current key frame.

5.2.2 Depth Map Estimation

In the depth map estimation the current depth map D_i is continuously refined with stereo measurements to newly tracked frames. The new tracked frames are either used to refine the depth map of the current key frame or chosen to replace the current key frame.

The refinement of the current key frame is done by using adaptive baseline techniques. With increasing camera motion the baseline from the key frame to the following frames grows, allowing for stereo correspondence search along different baselines. Given the transformation between a tracked frame and the key frame, that has been estimated prior in the tracking, the epipolar lines are calculated. Afterwards, for each pixel with sufficient gradient its depth hypothesis is updated with stereo measurements. The depth is calculated by finding the best matching point along the epipolar line, that is the point which minimizes the SAD error measured over five equidistant points along the epipolar line (Engel, Sturm, et al., 2013). Given the disparity d depth can be retrieved using the known transformation between both images. The computed depth is then integrated into the depth map by either creating a new depth hypothesis for the pixel or refining an existing one. If the pixel does not have a depth hypothesis yet, it is simply initialized with the estimated one. Otherwise, the new depth z_{new} is fused with the existing hypothesis z_{old} to depth z similar to the update step in a Kalman filter:

$$z = (1 - w) * z_{old} + w * z_{new}, \tag{5.8}$$

where w stands for the respective variance.

If the camera moves too far away from the key frame, which is measured using an adaptive threshold on the relative distance to the key frame

$$dist(\xi_{ji}) := \xi_{ji}^T W \xi_{ji}, \tag{5.9}$$

the current key frame is replaced with the latest tracked frame. As new frames come without depth information, the depth map is initialized by propagating the replaced depth maps to the new one. This is done by projecting all points in the old depth map in the coordinate system of the new frame using the estimated relative transformation ξ_{ji} between the key frame and the tracked frame. Afterwards, the depth map is regularized and outliers are removed.

Once a key frame is replaced, its depth map will not be refined anymore and the key frame is added to the pose-graph in the map optimization part.

5.2.3 Global Map Optimization

In the global map optimization key frames are added as vertices to the graph-based SLAM back-end g^{2}o (Kümmerle et al., 2011). The edges between key frames are similarity constraints $\xi \in sim(3)$ between nearby key frames: before a key frame is added to the graph, the similarity transformation to the previous key frame is estimated in a constraint search using direct tracking on $sim(3)$ minimizing following error function:

$$E(\xi_{ji}) := \sum_{p \in \Omega_{D_i}} \left\| \frac{r_p^2(p, \xi_{ji})}{\sigma^2_{r_p(p,\xi_{ji})}} + \frac{r_d^2(p, \xi_{ji})}{\sigma^2_{r_d(p,\xi_{ji})}} \right\|_\delta \tag{5.10}$$

$$\text{with } \sigma^2_{r_d(p,\xi_{ji})} := V_j([p']_{1,2}) \left(\frac{\partial b}{\partial a} \right)^2 + V_i(p)\,(x)^2, \tag{5.11}$$

$$\text{and } r_d(p, \xi_{ji}) := [p']_3 - D_j([p']_{1,2}). \tag{5.12}$$

$$\tag{5.13}$$

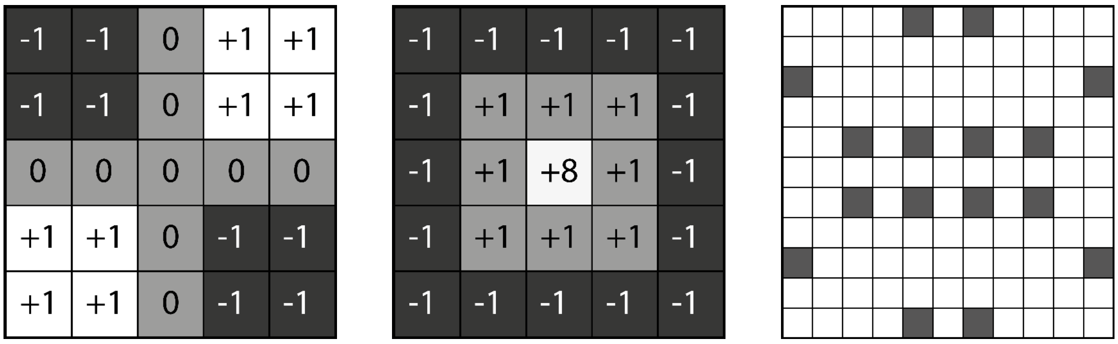
The minimization itself is similar to the direct tracking on $se(3)$ using the weighted Gauss-Newton method.

Once, the key frame is added to the map, further constraints for loop closure detection are searched. Therefore, the key frame is compared to the n closest key frames to find further similarity edges, that can be added as additional constraints to the graph optimization. To find robust constraints a reciprocal tracking check is introduced, which independently tracks the transformation ξ_{ji} and $\xi_{ij} \in sim(3)$. Only if both estimates are similar the constraint is added to the pose graph.

5.3 LIBVISO2

LIBVISO2 is a very fast feature-based visual odometry library for mono and stereo cameras. While the monocular version is still very experimental and expects the camera to be in a fixed and known height above the ground, the stereo version is more robust to outliers and, significantly faster. Furthermore, the approach is very general and does not require a certain motion model. The only prerequisite is, that the input images have to be rectified and the calibration parameters have to be known.

Similar to other feature-based methods, LIBVISO2 extracts and matches features over subsequent stereo frames and estimates the egomotion by minimizing the reprojection error. To be robust to outliers RANSAC is used for initialization



(a) Blob detector **(b)** Corner detector **(c)** Feature descriptor

Figure 5.2: LIBVISO2 filter masks and feature descriptor: A corner (a) and a blob filter (b) are used to extract and group salient features into four classes (corner min, corner max, blob min, blob max). Features are matched within their classes by comparing the SAD of horizontal and vertical sobel responses from the highlighted locations (c).

of the minimization step.

5.3.1 Feature Matching

In the feature matching step, image features are matched between the current and the previous stereo pair. For feature extraction each image is filtered with a 5×5 corner (Figure 5.2b) and blob mask (Figure 5.2a) independently, resulting in two filtered images each. Afterwards, salient features are found by performing non-maximum- and non-minimum-suppression on both filtered images. The resulting features are then grouped into the four classes *corner min*, *corner max*, *blob min* and *blob max*. To reduce the computational cost of the following matching stage, features are only matched within their classes. Two feature points are matched, by comparing their responses to 11×11 horizontal and vertical sobel filters. The sum of absolute distances is used as error metric. Instead of computing the SAD over all 121 points of the window, a sparse set of 16 locations is chosen as representatives, which again greatly reduces computational costs. The descriptor is shown in Figure 5.2c.

Given the stereo pairs of the current and previous view, features are matched in a circle between the left and right images as well as over time. Starting from all feature candidates of the current left images, for each feature the best match within a $m \times m$ search window is found in the previous left image. This feature is then again matched with its best match along the epipolar line of the previous right image.

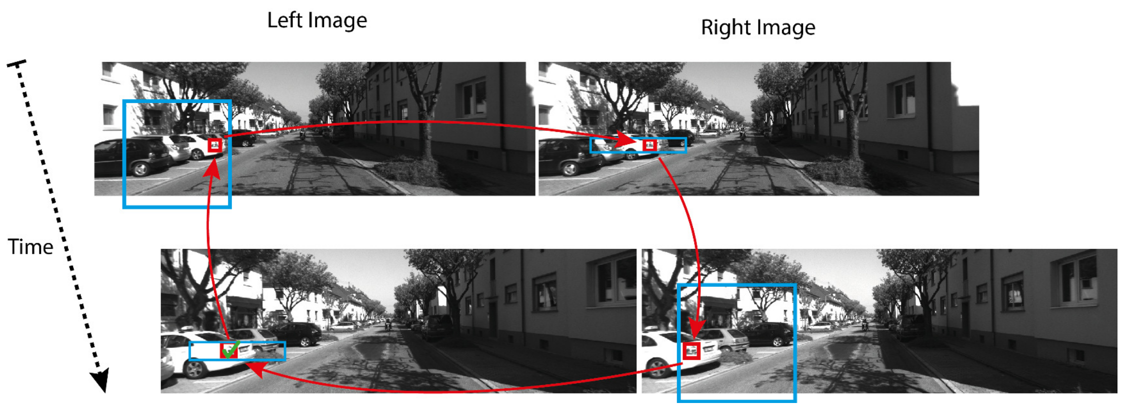


Figure 5.3: Circular matching of feature points by LIBVISO2: starting from the current left image (lower left) a windowed correspondence search (visualized as blue box) is performed on the previous left image (upper left). If a match has been found it is matched along the stereo baseline to the previous right image and from there on to the current right image. As a last step the best match for this point is searched in the current left window. Only if this feature location coincides with the starting location the match is accepted.

Afterwards, the best candidate is searched within a window in the current right image and this candidate is then matched back to the current left image, which closes the circle. The circle matching is visualized in Figure 5.3. If the last match found in the current left image coincides with the starting point, the *circle match* gets accepted and will be used for egomotion estimation.

Even though the matching costs are already reduced by in-class matching and a sparse descriptor, the computation over all feature points in four images takes several seconds. By splitting the computation into two stages the feature extraction process becomes real-time capable. In a first pass only a subset of all features is used for matching. The subset is estimated by using a more restrictive non-maxima-suppression than before, which leads to a much smaller set of features, that can be matched very efficiently. In a second pass the found correspondences of this subset serve as prior for all remaining features. For this the image is divided into equally distributed bins and for each bin an individual search space is estimated using the prior information. This greatly narrows down the search space for the remaining feature matches. To further increase the efficiency the first matching pass can be estimated on half of the resolution and can be refined in the second pass on full resolution.

Based on all found circle matches the egomotion is then estimated by minimizing the reprojection error using Gauss-Newton in combination with RANSAC for outlier removal.

5.3.2 Egomotion Estimation

Given all the circular correspondences from the feature matching stage the egomotion is computed by minimizing the sum of reprojection errors. In a first step the image is divided into buckets of size $k \times k$ and the number of features is reduced to n features per bucket. Usually, a set of 200 to 500 features is used for the minimization. The bucketing allows to keep the real-time performance and additionally enforces a uniform feature distribution over the image. Afterwards, feature points found in the previous stereo pair are projected into 3D using the calibration parameters of the camera and the inverse projection function π^{-1}:

$$X = \frac{(u - c_x) * baseline}{d} \tag{5.14}$$

$$Y = \frac{(v - c_y) * baseline}{d} \tag{5.15}$$

$$Z = \frac{f * baseline}{d}; \tag{5.16}$$

where d is the disparity of the corresponding image match in pixel.

The 3D Point is then transformed with a rotation R and translation t and back projected into the current stereo image using the projection function π given in Equation (2.2). Comparing this reprojection with the feature coordinates estimated in the matching stage, gives the reprojection error, we seek to minimize. The rotation and translation represent the camera motion from the previous to the current stereo pair and are initialized with the identity.

Then, the reprojection error is iteratively refined with regards to R and t using Gauss-Newton optimization:

$$E(R,t) = \sum_{i=1}^{n} \left\| x_i^L - \pi^L(X_i, R, t) \right\|^2 + \left\| x_i^R - \pi^R(X_i, R, t) \right\|^2 \tag{5.17}$$

RANSAC is used for outlier detection and to determine a good initialization. In a first step three randomly chosen point correspondences are used for estimating the camera motion using Gauss-Newton. This step is repeated n-times and the estimate with the highest inlier count is chosen for the final optimization. In the final optimization step all inliers are used for the iterative refinement.

5.4 Stereo Extension

In this section, we present our stereo extension of LSD-SLAM. The main motivation behind this is that absolute scale becomes observable when using stereo

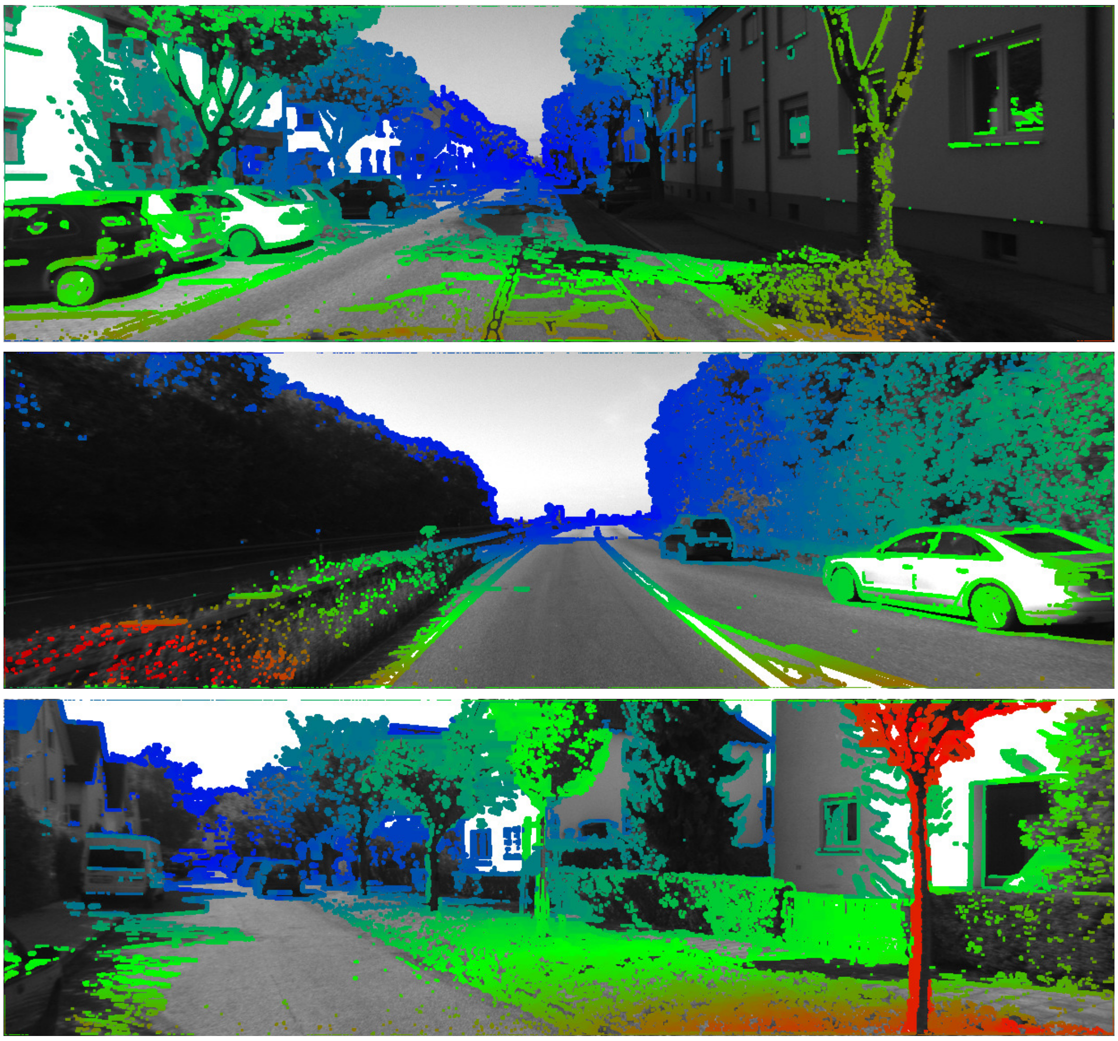


Figure 5.4: Computed semi-dense depth maps for KITTI datasets (sequences 00, 01 and 08). Color encodes depth: while close objects are drawn red, distant objects are drawn blue.

cameras. As we plan to navigate our MAV with visual SLAM, absolute values are of utmost importance for obstacle avoidance and navigation in the constructed 3D map. In contrast to the monocular version of LSD-SLAM, in the stereo version the absolute scale becomes observable. In addition, the stereo extension greatly reduces scale drift, as absolute depth values can be observed for each contributing pixel at every timestep. Moreover, stereo is much more robust to rotational motion, as the depth of new points can be calculated instantly and does not need the propagation over several images. With one camera it often occurs that more than half of the feature correspondences are lost at rotations due to missing image overlap between subsequent frames. In the stereo variant these lost features can be retrieved immediately using a correspondence search along epipolar lines. In the following sections we propose our stereo contributions to LSD-SLAM as well as a semi-direct approach, that employs LIBVISO2 as feature-based method for tracking the motion between key frames. We define a key frame at time i as $KF_i = \{I_i^L, I_i^R, D_i, V_i\}$, where I_i^L and I_i^R are the left and right intensity image, D_i and V_i are the depth map and corresponding variance map.

5.4.1 Stereo Tracking

In stereo tracking new frames are always tracked with regards to the current key frame. The first key frame is initialized with corresponding left and right intensity images. In contrast to the monocular version, we do not initialize the depth map with random values and large variance, but, as instant stereo is available, initialize the first key frame with ELAS (Geiger, Roser, et al., 2010) for a reliable depth estimate. We belief that the initialization of the depth map with a reliable and accurate stereo method is preferable to bootstrapping from random values. However, with an average runtime of approximately 90 ms ELAS is very expensive, which is why we only use it for a decent initialization. Afterwards, the following frames are tracked towards the key frame by minimizing the photometric error as well as the depth error. While in the monocular case absolute depth is not observable, with stereo cameras absolute depth is observable for every incoming stereo pair. This allows us to minimize the depth error in addition to the photometric error. Hence, for direct tracking with stereo we extend the minimization of the photometric residual r_p to take the depth residual r_d into account:

$$E_{STEREO} = \|r_p + r_d\| \tag{5.18}$$

with

$$\begin{aligned} r_p(p, \xi_{ji}) &= I_{KF_i}(p) - I_j(\pi(w(p, D_i(p), \xi_{ji}))) \\ r_d(p, \xi_{ji}) &= D_{KF_i}(p) - D_{STEREO_j}(p'_j, \xi_{ji}) \end{aligned} \tag{5.19}$$

where $D_{STEREO_j}(p'_j, \xi_{ji})$ is the depth of the warped point p'. The minimization is performed analogue to Section 5.2.1 by using the weighted least square formulation and solving it with the Gauss-Newton method. The residual is formulated as stacked residual

$$r = \begin{pmatrix} r_p \\ r_d \end{pmatrix} \tag{5.20}$$

and is weighted with a 2×2 weight matrix

$$W = \begin{pmatrix} w_p & 0 \\ 0 & w_d \end{pmatrix}, \tag{5.21}$$

where both residuals are weighted with the Huber norm as described in Equation (5.6).

For fast convergence tracking is started on the coarsest level of a gaussian image pyramid with four levels. If tracking did not converge, the estimated transformation is further refined at finer levels of the pyramid. Once, the relative pose to the current key frame is estimated, the tracked frame, together with its relative pose to the key frame, is fed into the depth mapping thread, where it either refines or replaces the current key frame.

5.4.2 Depth Map Update

By extending LSD-SLAM to stereo, we combine the existing depth map computation over time with instant stereo of the current image pair. The depth map of each key frame is updated with instant stereo measurements as well as with propagated depth from the previous key frame. Once a frame is chosen to become a new key frame, we immediately initialize its depth map with instant stereo depth from the left and right intensity images. An efficient and simple way for stereo estimation is to compute the sum of squared distances (SAD) along epipolar lines and to choose the best match, that minimizes the SAD error. As our images are already rectified, epipolar lines lie in the same image rows, and thus the correspondence search reduces to a one-dimensional search along image rows with the same row index. We compute the SAD error over a 15×15 pixel wide search window to be more robust to outliers. The depth of a pixel is computed using the disparity of the best match and the focal length f and baseline b of the camera system:

$$Z = \frac{bf}{\text{disparity}}. \tag{5.22}$$

The variance of the new depth hypothesis is computed using the SAD error and gradient magnitude: low SAD response and high gradient magnitude correspond to a low variance, while a high error and low magnitude correspond to a large variance.

In addition, we add a different weighting scheme for fish eye lenses. As fish eye lenses suffer from distortion at the image borders, we further increase the variance of a pixels depth hypothesis depending on the distance $r(u, v)$ to the optical center

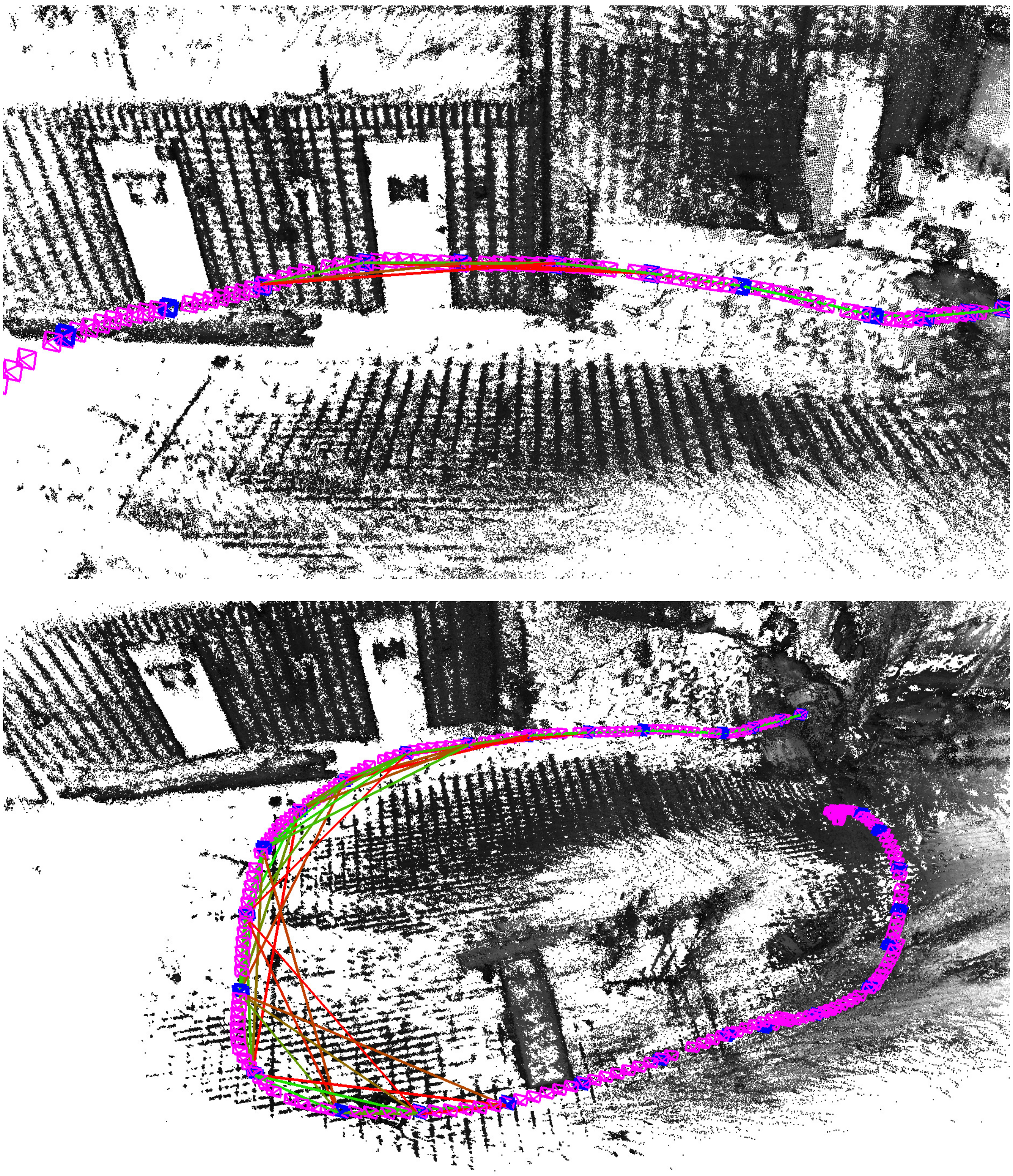


Figure 5.5: In the global map optimization edges between key frames (shown in blue) are added to the pose-graph. Nearby key frames usually match better than distant key frames (visualized as line color). The top picture shows the graph before optimization with many red edges. Below: after graph optimization the key frame poses have been refined, and the red edges turned green.

(c_x, c_y):

$$r(u,v) = \sqrt{(u-c_x)^2 + (v-c_y)^2}. \tag{5.23}$$

After the initialization of the depth map with stereo measurements, the depth estimates are refined by propagating depth hypotheses of the old key frames' depth map to the new candidate:

$$P_{new}(P) = R_{C,KF}P + t_{C,KF}, \tag{5.24}$$

where P is the 3D point in the old key frame. The rotation $R_{C,KF}$ and translation $t_{C,KF}$ describe the coordinate transformation from the key frames coordinate system KF to the candidates coordinate system C. Depth values are extracted from the Z coordinate of the propagated 3D points and from available instant depth measurements. If the depth residual between the propagated and instant depth is high, the depth value with smaller variance is chosen. Otherwise both estimates are fused as a variance-weighted sum:

$$d_{new} = (1-\omega)d_{STEREO} + \omega d_{PROP}. \tag{5.25}$$

Different examples for resulting semi-dense depth maps can be seen in Figure 5.4. The depth map estimation performs well on urban scenarios and even on highways.

Similarly to the monocular method, we are also able to refine the depth map by updating the map with measurements over time. This increases robustness, because the stereo search is performed on variable baselines, not only horizontally, but also on baselines along the camera trajectory.

For the success of the instant stereo computation, the accuracy of the extrinsic parameters of the camera are of utmost importance. As our cameras are attached to a non-rigid body frame, the baseline of the stereo pairs may change during flight. To overcome this, we estimate the camera extrinsics for every n-th key frame by directly tracking the right frame to the left frame. If the new estimate differs from the old, the baseline is updated with the new value. However, as the KITTI and EuRoC dataset use rigid cameras, this is only done for the special case on our MAV. For the public datasets we use the available offline calibration.

5.4.3 Global Mapping

So far, we presented an approach performing incremental visual odometry by directly tracking incoming stereo images in combination with semi-dense depth reconstruction.

The new pose estimate is updated with regards to the pose estimate of a previous key frame and the estimate of the motion between both frames. Due to noise in both estimates the trajectory drifts over time. With each new update the error of previous estimates is propagated over times, as shown in Figure 2.5 of Section 2.4.

To compensate for drift we use g^{2}o (Kümmerle et al., 2011) for global pose-graph optimization. The pose-graph contains the positions of the key frames as vertices and their relative transformations as edges. Instead of optimizing $sim(3)$ constraints as in the monocular SLAM, we estimate constraints between key frames as their $SE(3)$ rigid-body motion. Once a key frame is replaced with a new candidate, its pose is added to the key frame graph as node. Afterwards, we search the existing nodes in the graph for additional constraints, that can refine the pose graph. For this the closest n key frames, that have sufficient scene overlap, are reciprocally tracked towards the key frame: we estimate the transformation between both frames in both directions, by tracking the constraint candidate to the key frame and vice versa. Only if the tracking succeeds for both directions and the resulting transformations are similar, the resulting transformation is added as additional constraint to the graph. All edge constraints e_{ji} define a cost function, that is optimized using g^{2}o:

$$E = \sum_{e\ ji} \left\| C_i - T_{e_{ji}} C_{ji} \right\|^2 . \tag{5.26}$$

Figure 5.5 shows how the map for an exemplary scene is built and more and more SLAM constraints are added to the pose-graph. While key frames are colored blue, in-between frames, that are tracked using LIBVISO2, are colored pink. The constraints between key frames are visualized as lines connecting them. Edge constraints with high error are colored red, while constraint with a lower error are colored green. It can be seen, that constraints that connect distant key frames generally show a higher error than those connecting nearby frames.

5.5 Semi-direct Tracking

Our idea is inspired by combining feature-based methods with direct methods for taking advantage of the different strengths of both approaches. For efficient and reliable state estimation we combine fast feature matching with precise image alignment. The general pipeline of our approach is visualized in Figure 5.6.

At the beginning, we initialize the first key frame with the first pair of images and a dense depth map computed by ELAS. Following frames are then tracked towards the key frame using feature-based LIBVISO2. The relative pose of the tracked frames are concatenated and form the relative pose of the camera to the key frame

$$\xi_{feat} = \xi_{in} \circ \xi_{in-1} \circ \cdots \circ \xi_{i0}. \tag{5.27}$$

The current absolute pose of the camera at step j and key frame i can be retrieved by:

$$\xi_{ij} = \xi_{KFi} \circ \xi_{ij-1} \tag{5.28}$$

We perform feature-based odometry as long as the motion is sufficient small. As soon as the motion exceeds the motion threshold, we perform direct tracking again. The motion threshold is computed using the estimated optical flow of LIBVISO2:

$$\epsilon_{MOTION} = \frac{1}{n} \sum_{i=1}^{n} \sqrt{(u_c - u_p)^2 + (v_c - v_p)^2}, \tag{5.29}$$

where n is the number of matched feature points and (u_c, v_c) and (u_p, v_p) are pixel coordinates of corresponding matches between the current and previous image.

As soon as large or fast motions occur, the feature-based estimate ξ_{feat} of the camera motion is passed as initial estimate to the direct tracking of a new key frame:

$$\xi_{KFi+1} = \xi_{KFi} \circ \xi_{feat} \tag{5.30}$$

This allows us to track larger motions faster and more robust. Once a new key frame is tracked, we start feature-based matching again.

The depth map of a new key frame is initialized by instant stereo correspondences and then fused with the previous depth map by propagation, as described in Section 5.4.2.

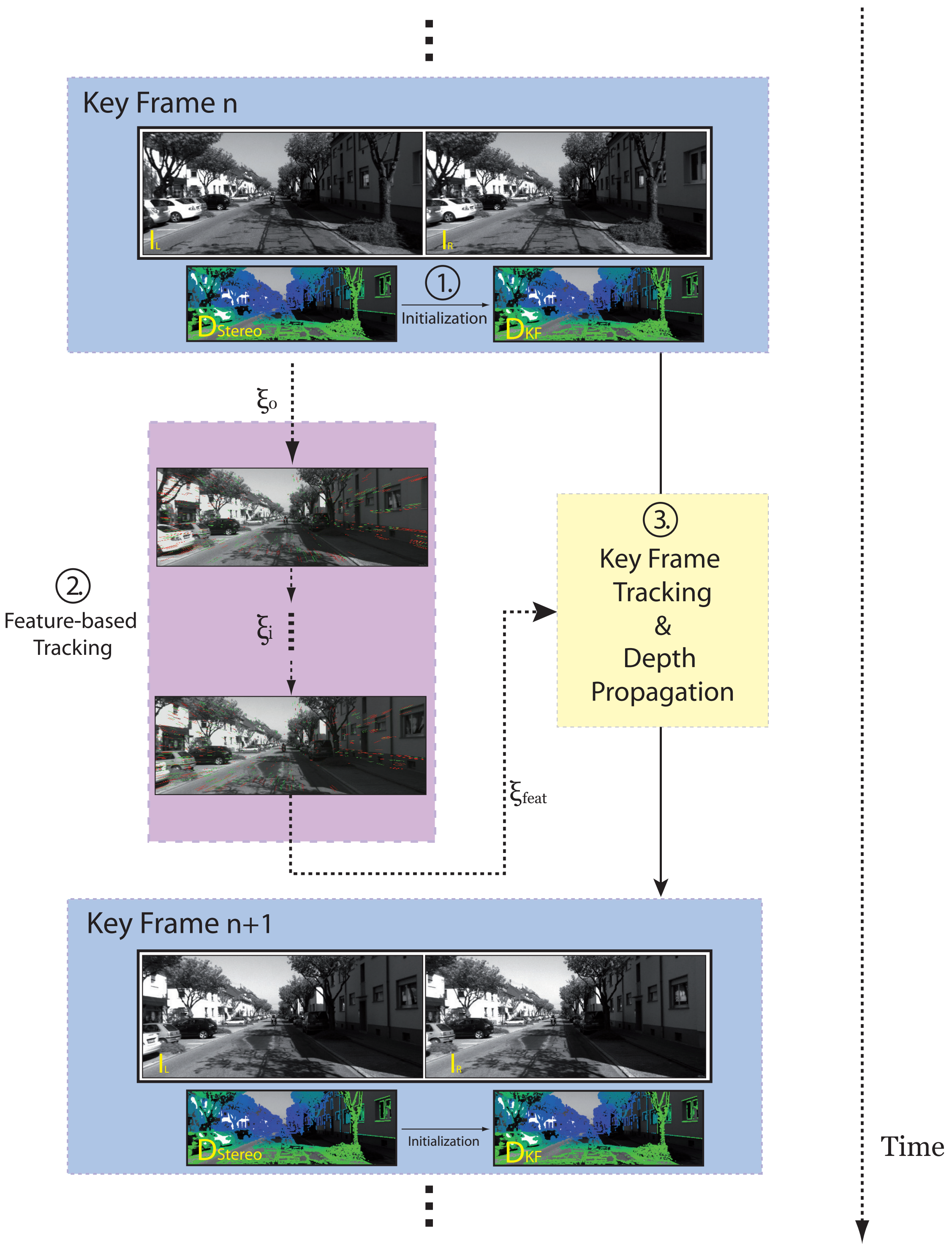


Figure 5.6: Overview of our combined semi-direct approach. While direct tracking is only performed on key frames, feature-based tracking is performed for frames in between. The output of the feature-based odometry serves as prior for direct tracking of key frames.

6 Evaluation

For the evaluation of our semi-direct approach we perform experiments on three challenging stereo datasets: the well-known KITTI-dataset (Geiger, Lenz, et al., 2012), the EuRoC dataset[1] and a dataset recorded with our high-performance MAV presented in Section 2.1. The datasets differ in terms of frame rate, apparent motion and stereo baseline. All experiments have been conducted on an Intel Core i7-4702MQ running at 2.2 GHz with 8 GB RAM.

We compare the quality of our combined approach in terms of accuracy and runtime to LSD-SLAM (Engel, Schöps, et al., 2014) and LIBVISO2 (Geiger, Ziegler, et al., 2011), as well as to state-of-the-art methods like S-PTAM (Pire et al., 2015) and ORB-SLAM (Mur-Artal et al., 2015). The execution of the referred methods has been obtained using the provided default parameters.

As ground truth for all sequences is available, we employ the evaluation metrics by Sturm et al. (2012) and measure the absolute trajectory error (ATE) by computing the root mean squared error (RMSE) over the whole trajectory. In addition, we also provide the median error for better insight, because single outliers can greatly affect the final result. The ATE is a popular measure for the evaluation of visual SLAM systems, as it measures the Euclidean distance between ground truth poses and estimated poses at corresponding timestamps, and thereby allows to evaluate the global consistency of SLAM systems. In a first step the trajectories are aligned, because they come from different coordinate systems. Moreover, a similarity alignment is performed for the monocular systems to estimate the absolute scale of the estimated trajectory. For an intuitively accessible visualization, trajectories are always shown in bird's eye perspective. As we use the camera coordinate system, this means that the height axis Y is omitted in the plots. In the following sections we first present detailed results for each dataset, individually. Moreover, we evaluate the performance of visual SLAM compared to pure visual odometry and provide quantitative result. Afterwards, we shortly summarize the obtained average results for accuracy and runtime and conclude with qualitative results of our 3D reconstruction.

[1]http://projects.asl.ethz.ch/datasets/doku.php?id=kmavvisualinertialdatasets

KITTI Sequence	Absolute Trajectory Error RMSE (Median) in m Ours	LIBVISO2	ORB-SLAM	S-PTAM
00	**5.79 (4.54)**	29.71 (18.49)	8.30 (6.04)	7.83 (6.30)
01	**61.55 (54.57)**	66.54 (60.46)	335.52 (303.79)	204.65 (157.10)
02	18.99 (14.38)	34.26 (27.36)	**18.66 (15.03)**	20.78 (17.28)
03	**0.63 (0.52)**	1.67 (1.54)	11.91 (9.19)	10.53 (10.41)
04	**0.67 (0.46)**	0.80 (0.66)	2.15 (1.73)	0.98 (0.88)
05	5.47 (4.14)	22.14 (19.07)	4.93 (4.73)	**2.80 (2.24)**
06	**2.06 (1.80)**	11.54 (10.26)	16.01 (15.56)	4.00 (4.01)
07	2.34 (1.67)	4.41 (4.37)	4.30 (3.65)	**1.80 (1.53)**
08	8.42 (7.04)	47.67 (34.84)	38.80 (18.12)	**5.13 (4.26)**
09	**5.46 (3.33)**	89.83 (77.57)	7.46 (6.91)	7.27 (4.61)
10	**1.68 (1.37)**	49.35 (36.00)	8.35 (7.55)	2.08 (1.70)
mean	**10.28 (8.53)**	32.54 (26.42)	41.49 (35.66)	25.74 (20.26)
mean w/o S 01	**5.15 (3.93)**	29.14 (23.02)	12.09 (8.85)	7.85 (6.57)

Table 6.1: ATE Results on KITTI Dataset.

6.1 KITTI

The KITTI dataset (Geiger, Lenz, et al., 2012) is a very popular dataset for the evaluation of visual and laser-based odometry or SLAM methods. It contains 22 stereo sequences accompanied by laser scans, and ground truth from a localization unit consisting of a GPS and an IMU. The stereo camera rig and the laser scanner are mounted on top of a standard station wagon—the autonomous driving platform Annieway (Kammel et al., 2008). The stereo rig has a baseline of approximately 54 cm.

Rectified images are provided with 10 Hz and with a resolution of 1240 × 376 pixels. The sequences are recorded in real-world driving situations along urban, residual and countryside roads. The distance traveled ranges from a few 100 meters up to 5 kilometers with driving speeds up to 80 km/h.

The dataset is very challenging, because the low frame rate in combination with fast driving speed leads to large inter-frame motions up to 2.8 m per frame. This greatly limits the amount of possible feature correspondences. Moreover, dynamic motions from passing vehicles, bicycles or pedestrians, that have great impact on the performance of visual odometry systems, are included frequently.

We compare the performance of our semi-direct method with four state-of-the-art methods for visual odometry and SLAM.

We selected LIBVISO2 and LSD-SLAM for reference, as our method is built upon them. Moreover, we chose two recent feature-based SLAM algorithms, that presented promising results: ORB-SLAM as a monocular and S-PTAM as a stereo method. All processing is done on the original image resolution of the rectified

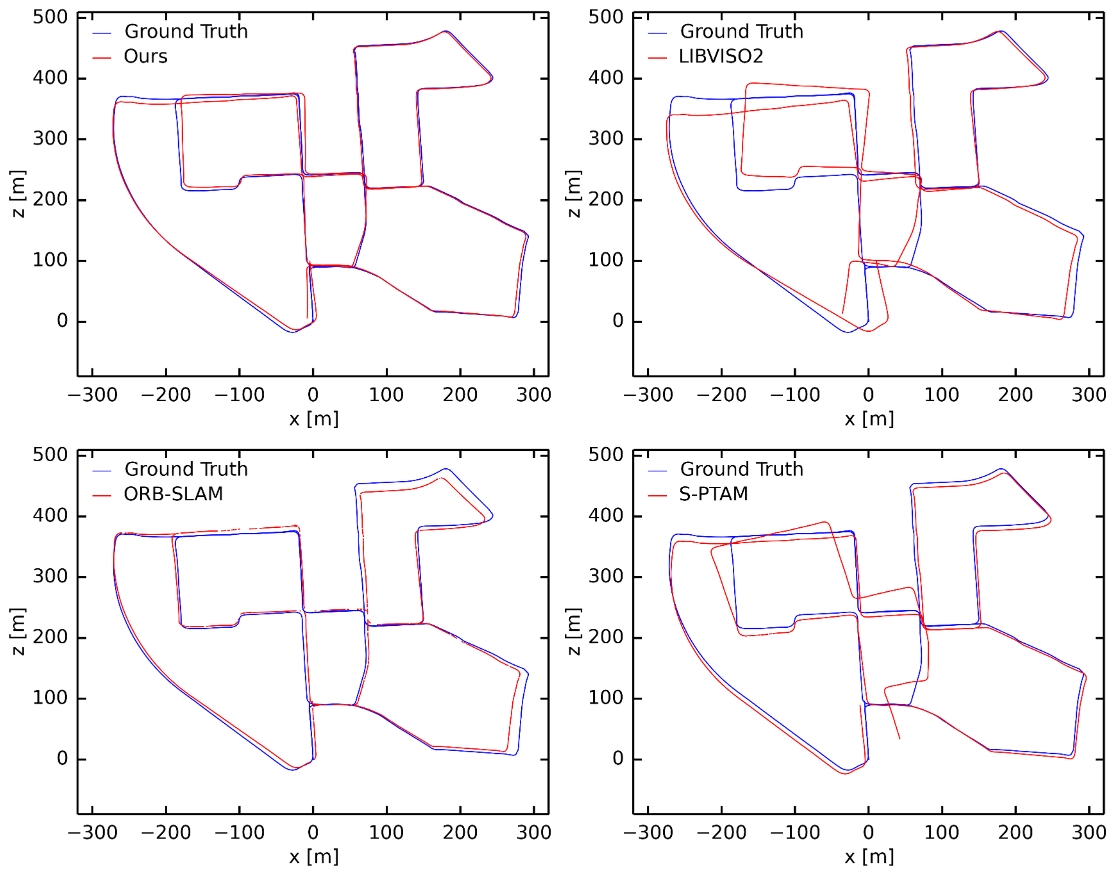


Figure 6.1: Results for KITTI Sequence 00. Comparison of our method to LIBVISO2 (top row), ORB-SLAM and S-PTAM (bottom row). Our methods achieves the lowest ATE (5.79 m

images of 1240 × 376.

Both error measures—RMSE and Median—for the training sequences 00 to 10 of the KITTI dataset are listed in Table 6.1.

Unfortunately, LSD-SLAM fails on all sequences of the KITTI dataset. This is probably caused by too large inter-frame motion for a pure monocular direct method, as sufficient scene overlap is important for successful tracking. Moreover, it can be seen, that all SLAM methods lack performance on sequence 01, resulting in a a very high ATE. Sequence 01 contains images from driving on a highway, thus it is hard to find re-occurring feature points in subsequent frames.

Overall our method is equally good and in seven of eleven cases even better than state-of-the-art methods. Especially sequences 03 and 04 show very accurate results below 1 m. In three of the cases S-PTAM and in one case (sequence 02) ORB-SLAM performs better. As LIBVISO2 is a pure odometry method, it

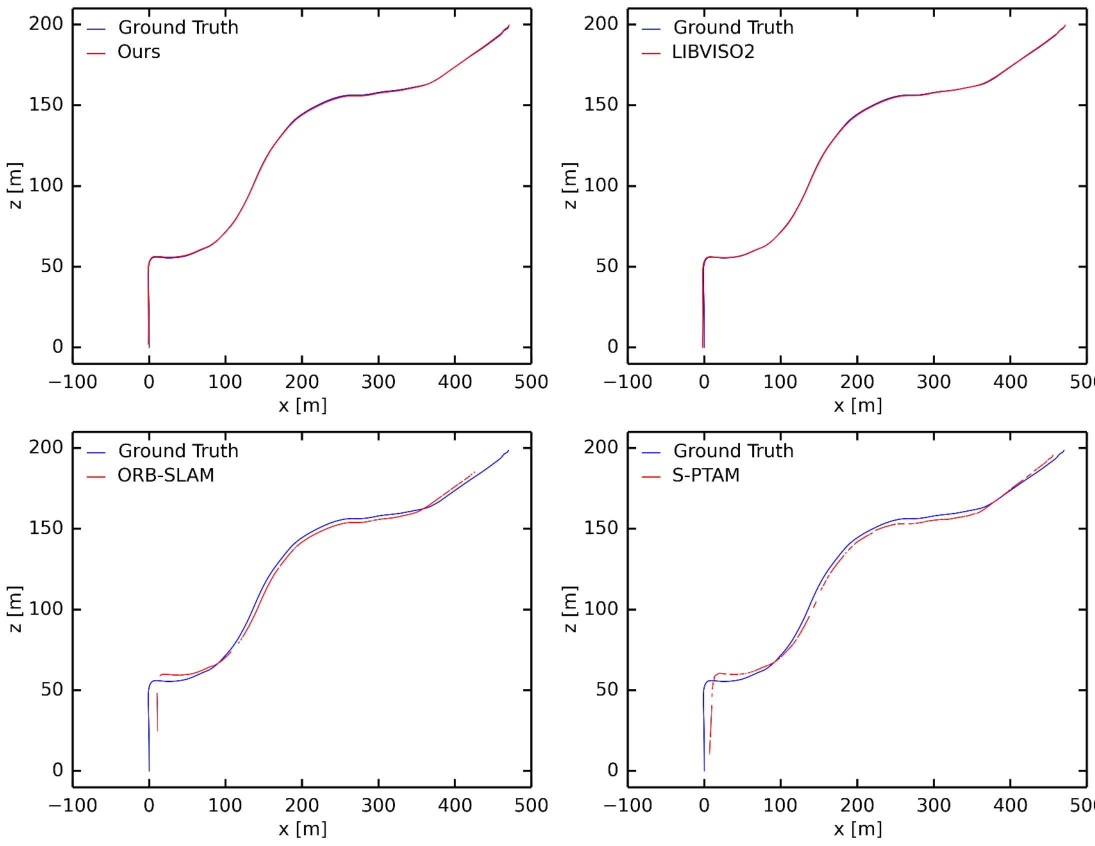


Figure 6.2: Results for KITTI Sequence 03. Comparison of our method to LIBVISO2 (top row), ORB-SLAM and S-PTAM (bottom row). Our method and LIBVISO2 show accurate trajectories.

performs significantly worse than the SLAM methods on all datasets.

When averaging over the eleven training sequences our method ranks first, followed by S-PTAM, ORB-SLAM and LIBVISO2. However, the bad results from sequence 01 greatly affect the final average computation, as all methods bring in very high ATEs from sequence 01. One could argue, that such high ATE values count as outlier or failure. Therefore, we also show resulting means when omitting sequence 01 for all methods. It follows, that these results show distinct lower ATEs. When omitting sequence 01 our method achieves a mean (median) ATE of 5.15 m (3.99 m) compared to the S-PTAM result of 7.85 m (6.57 m).

For a better visualization exemplary trajectories are shown in birds-eye perspective for the sequences 00, 03, 09 and 10.

Sequence 00 is shown in Figure 6.1. It can be seen, that our approach performs best, followed by ORB-SLAM, S-PTAM and LIBVISO2. Moreover, limitations of the approaches become visible: as LIBVISO2 is a pure odometry method, it

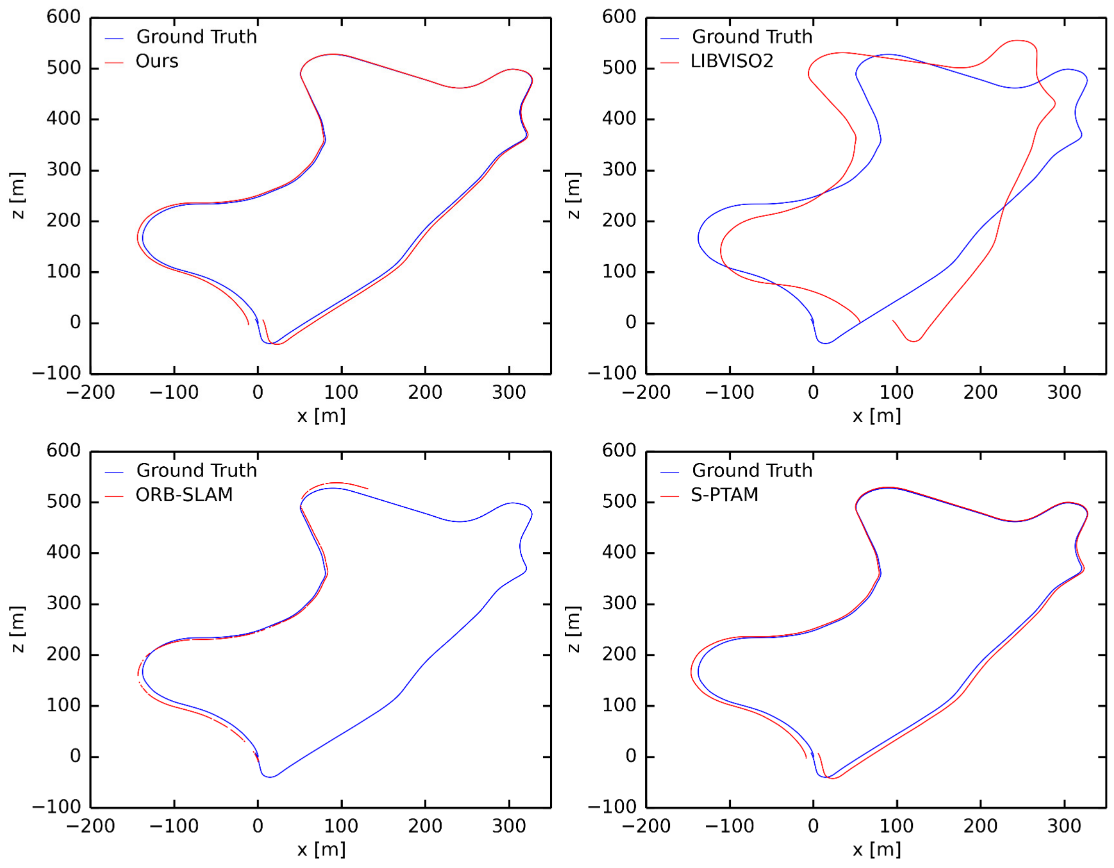


Figure 6.3: Results for KITTI Sequence 09. Comparison of our method to LIBVISO2 (top row), ORB-SLAM and S-PTAM (bottom row). While ORB-SLAM looses track and LIBVISO2 accumulates drift, S-PTAM and our method stay close to the ground truth.

accumulates more drift over time, and as ORB-SLAM is a monocular method, scale is not always estimated correctly. Rotations are challenging for all methods. In this sequence S-PTAM fails to track rotations frequently and exhibits drift for the last part of the trajectory.

Figure 6.2 shows the results for sequence 03, a trajectory without full loop-closure. We choose this sequence to compare the drift over time, when no full loop can be closed. All estimated trajectories are close to the ground truth. However, our method is—with 0.63 m ATE—distinctively more accurate than ORB-SLAM (11.91 m) and S-PTAM (10.53 m). Additionally, LIBVISO2 also shows accurate results with an ATE of 1.67 m and does not accumulate much drift for this trajectory.

In sequence 09 a full loop closure appears at the very end of the trajectory, that is not always detected from the SLAM methods before the sequence ends. This

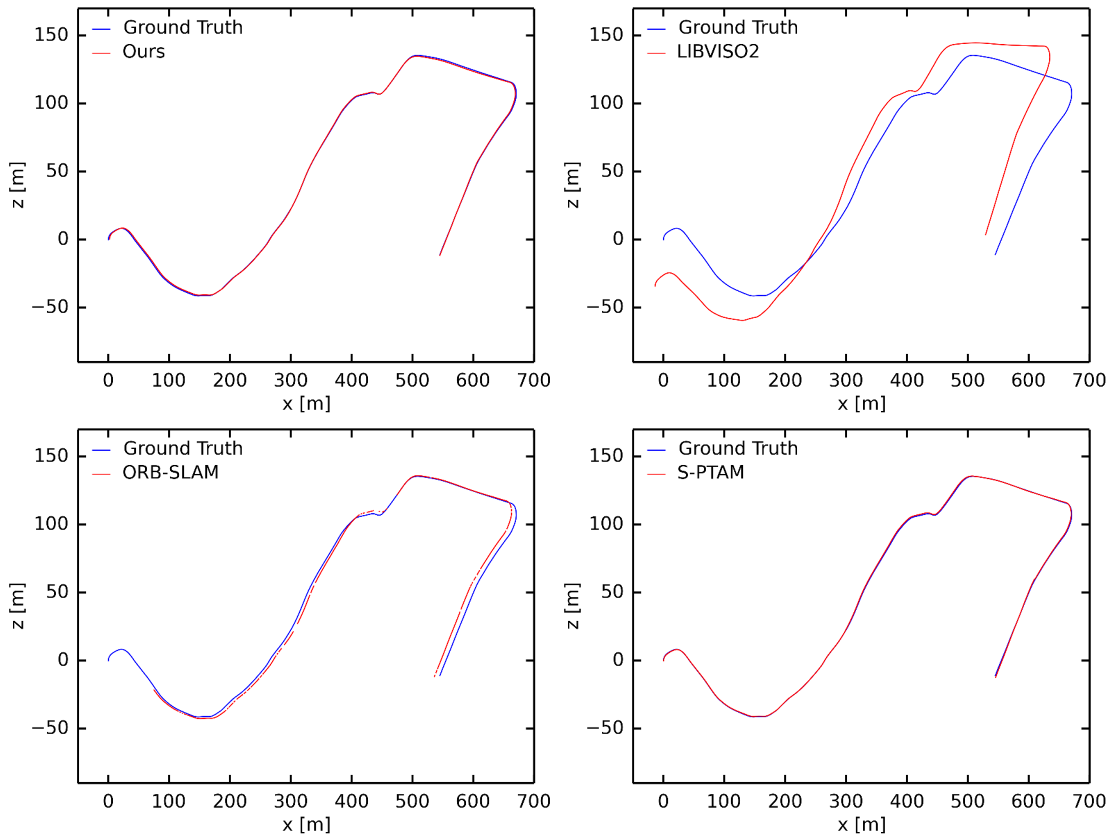


Figure 6.4: Results for KITTI Sequence 10. Comparison of our method to LIBVISO2 (top row), ORB-SLAM and S-PTAM (bottom row) on a longer trajectory without loop closures.

behavior is shown in Figure 6.3. Again, LIBVISO2 suffers from drift over time, while the results from our Semi-Direct SLAM (5.46 m) are more accurate than the results from S-PTAM (7.27 m). Moreover, it can be seen, that ORB-SLAM lost track at some point and failed to relocalize. Thus, more than a half of the trajectory remains uncovered. This is not visible in the error measure, because the ATE is only computed over existing measurements.

Sequence 10 is similar to sequence 03, as it contains no full loop, but it covers a longer path and performs more rotations. Results for this sequence are visualized in Figure 6.4. They show, that our method performs well, even if the path of LIBVISO2 drifts over time. ORB-SLAM fails to initialize right from the beginning, but later on retrieves a trajectory consistent with the ground truth, but with little offset. S-PTAM again shows similar results to Semi-Direct SLAM, though Semi-Direct SLAM performs slightly better (1.68 m to 2.08 m respectively).

Unfortunately, to our knowledge there is no other publicly available direct method other than LSD-SLAM to compare with. However, as LSD-SLAM fails on the KITTI sequences, we compare our semi-direct approach to its fully direct version without feature-based initial estimates. In particular, we compare the combined semi-direct approach to its building blocks—LIBVISO2 and direct stereo tracking—separately. As LIBVISO2 is a pure odometry method, we evaluate it against results from our semi-direct odometry without closing loops.

In Figure 6.5 the results from three different datasets $(00, 02$ and $06)$ are shown in birds-eye perspective. The left column shows the resulting path LIBVISO2 computed and the right column the path from the semi-direct odometry. It can be seen, that LIBVISO2 drifts more over time than the semi-direct approach, while the semi-direct approach stays closer to the ground truth trajectory. However, both methods tend to drift over time as they are pure odometry methods, but the semi-direct approach shows much less drift.

When comparing our semi-direct approach to its fully direct version without feature-based odometry as initial estimate, we noticed that a fully direct version has problems with strong turns in the dataset. Moreover, the dataset is very challenging to a fully direct method as it contains large inter-frame motions and difficult lighting changes. Large inter-frame motions are challenging to direct methods, because direct methods assume small pixel displacements (Irani and Anandan, 2000). Without a good initial estimate they often fail to retrieve large displacements. Difficult lighting changes, induced by auto-exposure and changing sunlight, are challenging, as they violate the brightness constancy assumption. Thereby, it can be seen in Figure 6.6 that the fully direct odometry accumulates more drift over time than our semi-direct version. Again, while the semi-direct approach is shown in the right column, the fully direct approach is visualized in the left column for dataset 00 and 06. Fully direct tracking tends to fail especially at strong turns and at street crossings where lighting changes increase, because the car leaves shadowed street canyons. In contrast, our approach is more robust to strong rotations and lighting changes.

The semi-direct method performs better than its isolated building blocks. The direct tracking is in principle more accurate, but has problems with large motions. However, when a good initial estimate is available, as in our case from LIBVISO2, direct tracking succeeds even at large motions and with a low frame rate.

Generally speaking, a combined semi-direct odometry performs better than both—feature-based and direct—odometries alone. Overall, our approach shows promising results on the KITTI dataset when compared to other state-of-the-art methods.

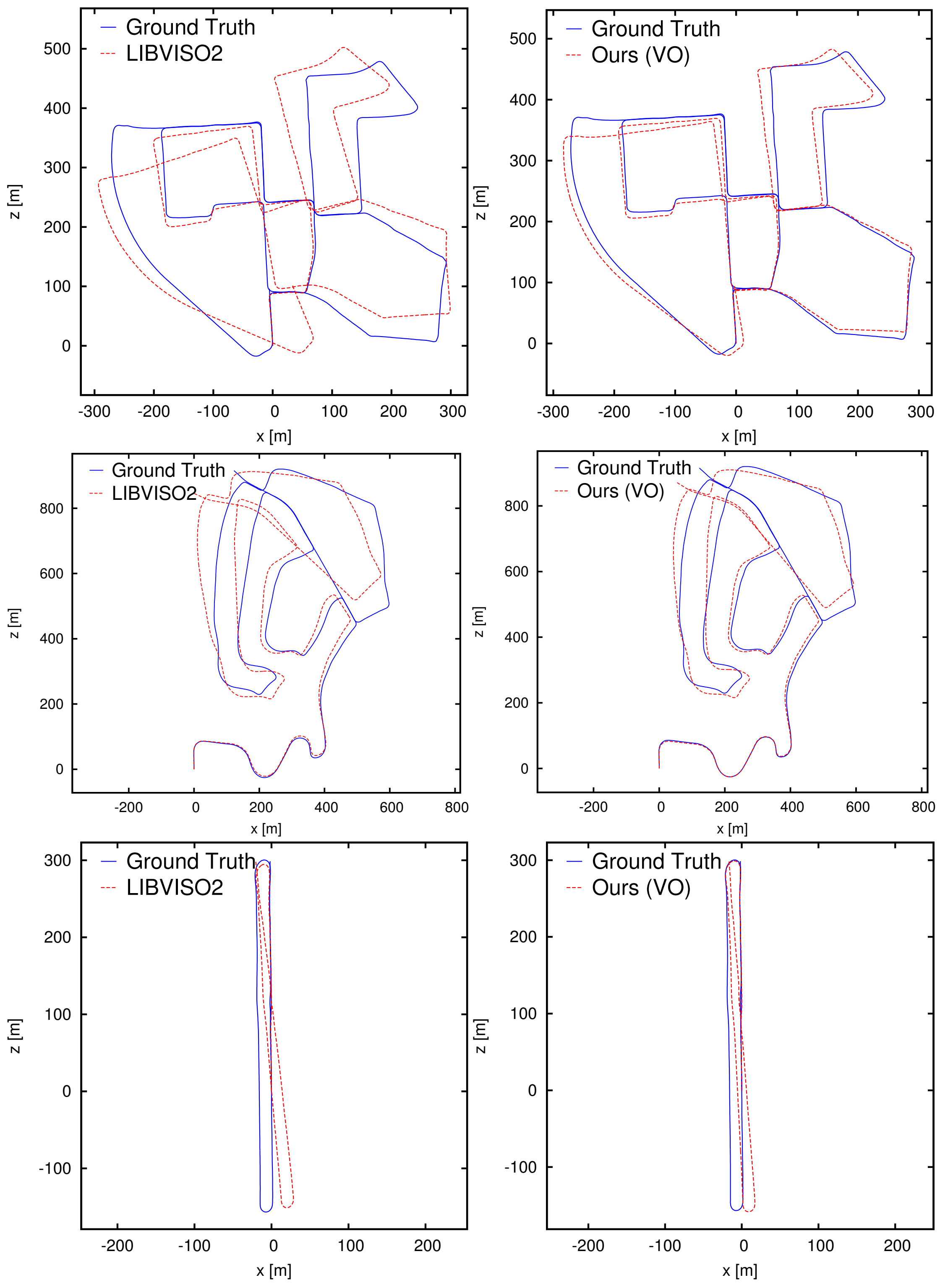


Figure 6.5: Comparison of the results from LIBVISO2 (left) to our semi-direct odometry (right). Top Row: KITTI Sequence 00, Middle Row: KITTI Sequence 02, Bottom Row: KITTI Sequence 06. In direct comparison to LIBVISO2 our method accumulates less drift.

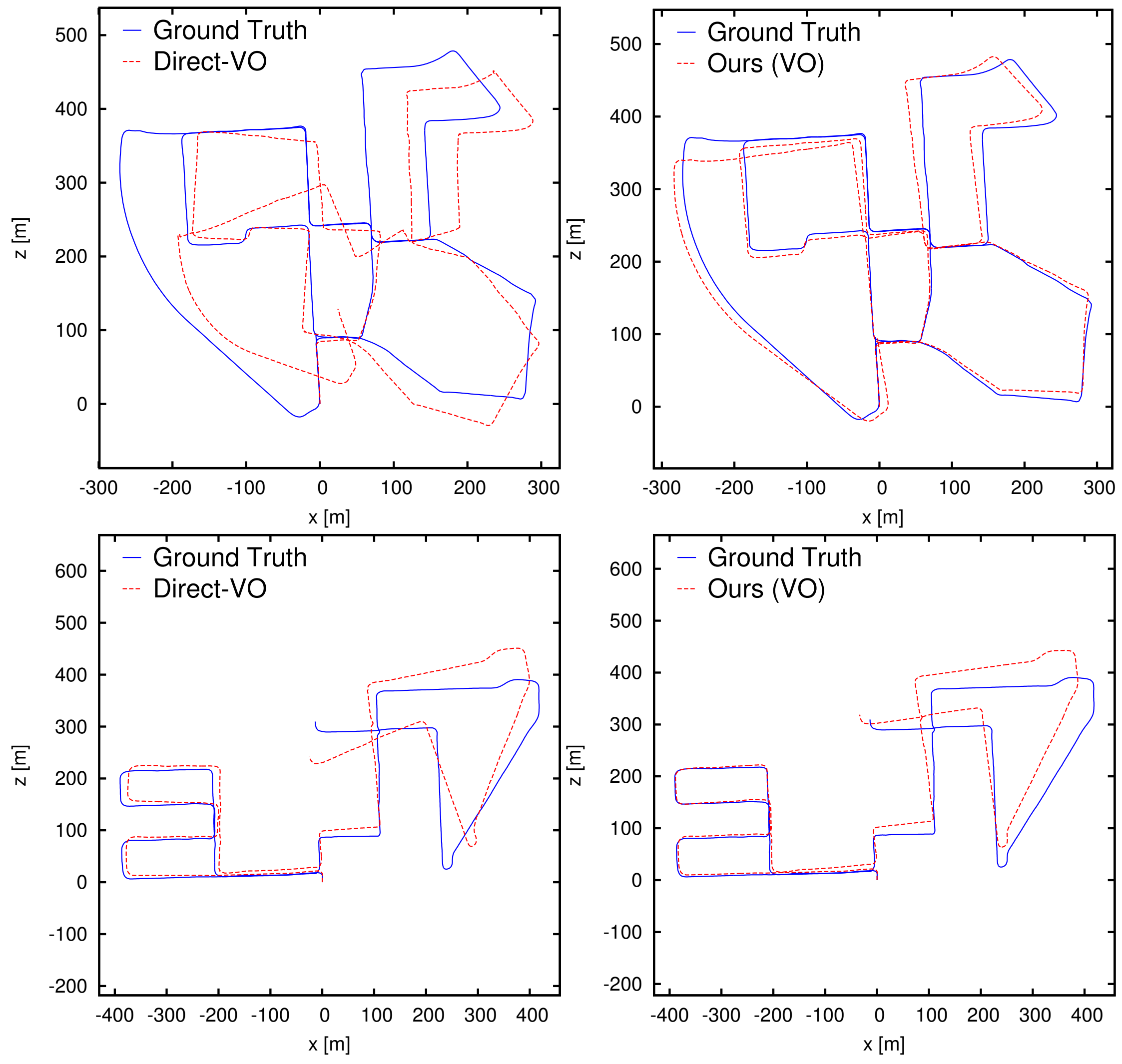


Figure 6.6: Comparison of the results from direct odometry (left) to our semi-direct odometry (right). Top Row: KITTI Sequence 00, Bottom Row: KITTI Sequence 08. In direct comparison to the direct visual odometry our method is clearly more robust to fast rotations and to large motions.

6.2 EuRoC

In addition to the evaluation on the KITTI dataset, we perform further experiments on the recently presented visual-inertial EuRoC MAV dataset, that contains stereo images and synchronized IMU reading from the on-board computer of an Asctec Firefly hex-rotor helicopter. We choose six trajectories with different difficulties from the two Vicon datasets V0 and V1. The data has been collected from flights in a room, that is stocked with a Vicon motion capture system, offering 6D ground truth poses.

The MAV is equipped with a visual-inertial sensor (Nikolic et al., 2014), that captures stereo images of WVGA resolution with 20 Hz and synchronized IMU measurements with 200 Hz.

Each dataset contains three trajectories with increasing difficulty named as: easy (_01), medium (_02) and difficult (_03). The easy trajectories have good illumination, are feature rich and show no motion blur and only low optical flow and low varying scene depth. They capture a static scene. The difficulty increases in the medium trajectories by adding challenging lighting conditions, high optical flow and medium varying scene depth. However, they still show a static scene and a feature rich environment without motion blur. In contrast, the difficult scene contains areas with only few visual features and more repetitive structures. Moreover, they add motion blur and challenging lighting conditions. The MAV performs very aggressive flight maneuvers resulting in high optical flow and highly varying scene depth in a non-static scene.

The dataset is known to have different issues, that make a reliable state-estimation more challenging: for example, the stereo images were captured using an automatic exposure control that is independent for both cameras. Therefore, shutter times are different, which results in different image brightnesses, making stereo match-

EuRoC	**Absolute Trajectory Error RMSE (Median) in m**				
Dataset	Ours	Libviso2	LSD-SLAM	ORB-SLAM	S-PTAM
V1_01	**0.12 (0.11)**	0.31 (0.31)	0.19 (0.10)	0.79 (0.62)	0.28 (0.19)
V1_02	**0.11 (0.10)**	0.29 (0.27)	0.98 (0.92)	0.98 (0.87)	0.50 (0.35)
V1_03	**0.75 (0.45)**	0.87 (0.64)	X	2.12 (1.38)	1.36 (1.09)
V2_01	**0.18 (0.12)**	0.40 (0.31)	0.45 (0.41)	0.50 (0.42)	2.38 (1.78)
V2_02	**0.27 (0.22)**	1.29 (1.08)	0.51 (0.48)	1.76 (1.39)	4.58 (4.18)
V2_03	**0.87 (0.66)**	1.99 (1.66)	X	X	X
mean	**0.38 (0.28)**	0.85 (0.71)	0.53 (0.48)	1.23 (0.94)	1.82 (1.52)

Table 6.2: ATE Results on EuRoC Dataset.

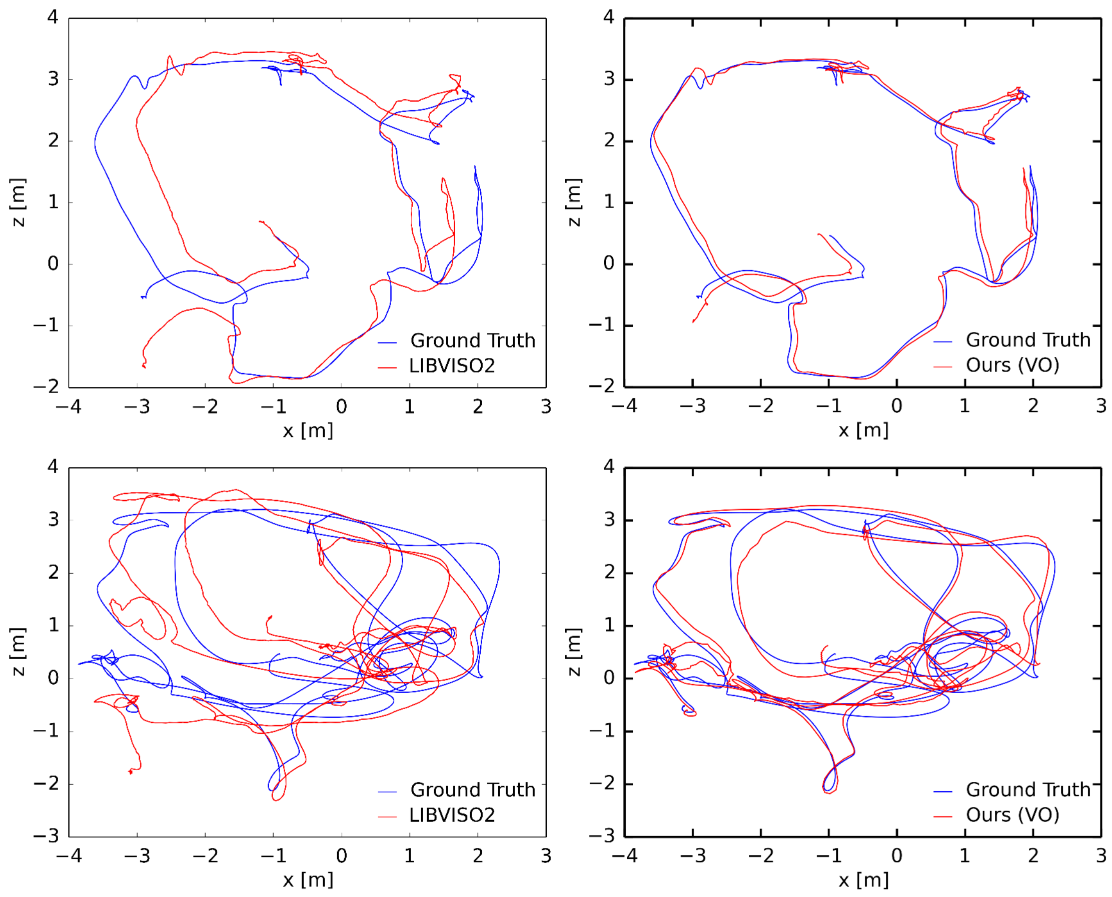


Figure 6.7: Comparison of the results from LIBVISO2 (left) to our semi-direct odometry (right) on datasets V2_01 and V2_02 with ground truth from a Vicon motion capture system. Again our method is much closer to the ground truth even without SLAM.

ing and feature tracking more challenging. This is especially important, as direct methods minimize the photometric error.

Moreover, as the ground truth is recorded from a different physical device than the images, the accuracy depends on the synchronization scheme used (Lab, 2015).

The resulting ATE values are listed in Table 6.2. As the difficult datasets V1_03 and V2_03 contain very dynamic movements and fast rotations with an MAV, LSD-SLAM often loses track after a few seconds and is then unable to re-localize for the rest of the trajectory. In Table 6.2 this is denoted as failure (X). Similarly, S-PTAM and ORB-SLAM lose track for the difficult trajectory V2_03. This dataset shows very challenging conditions with strong motion blur and fast aggressive maneuvers. Moreover, the absence of sufficient visual features makes it hard for the feature-based method to succeed.

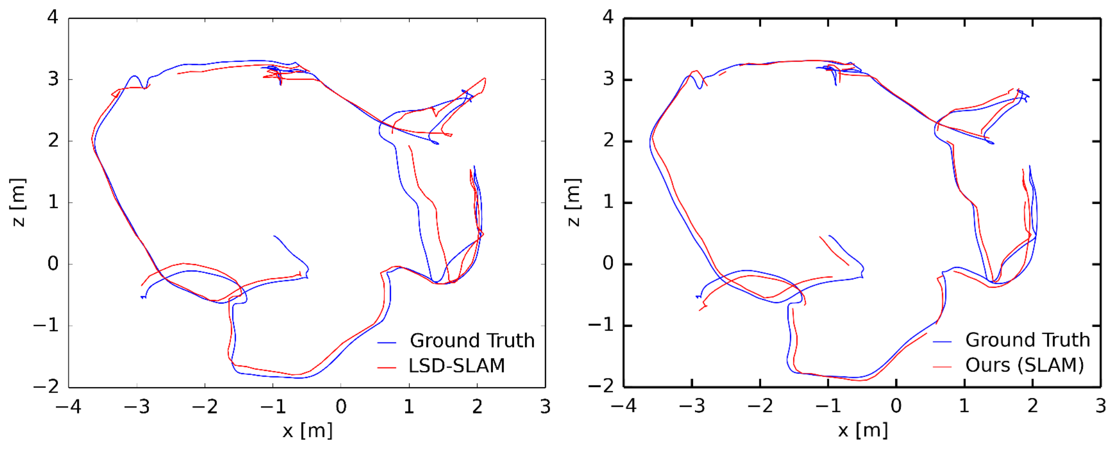


Figure 6.8: Comparison of the results from LSD-SLAM (left) to our semi-direct SLAM (right) on dataset V2_01 with ground truth from a Vicon motion capture system. While LSD-SLAM shows an ATE of 0.45 m our methods performs better with an ATE of 0.18 m

Table 6.2 also shows, that our approach outperforms the other methods and reliably recovers the motion for all test sequences. Additionally, it can be seen, that the results of LIBVISO2 are improved on every trajectory.

On average semi-direct SLAM achieves a higher accuracy, with 0.38 m ATE, than LSD-SLAM, with 0.53 m, and ORB-SLAM, with 1.23 m ATE, and S-PTAM with 1.82 m ATE. LSD-SLAM, ORB-SLAM and S-PTAM often suffer at fast motions in combinations with rotations, and then tend to lose track temporarily.

Additionally, we again directly compare results from Semi-Direct Visual Odometry to LIBVISO2 and to Direct Odometry from LSD-SLAM. Figure 6.7 shows the resulting trajectories for datasets V2_01 and V2_02 of LIBVISO2 and Semi-Direct Visual Odometry. Both methods were performed without loop-closures, and thus drift over time by accumulating small errors in the estimates. It can clearly be seen, that the Semi-Direct Odometry is closer to the ground truth from the Vicon system than LIBVISO2. Even though the datasets contain fast rotations, our method stays close to the ground truth path.

In comparison to LSD-SLAM our approach is more robust to fast rotations in the trajectory, as can be seen in Figure 6.8. While LSD-SLAM computes wrong estimates at strong turns, our method follows the path more precisely.

In addition to the official datasets we performed one manual flight in the Vicon room—named Vicon_m—where we evaluated the mapping abilities of our approach. As an example sequence for our mapping abilities Figure 6.9 shows a sequence captured on the manual flight: the MAV captures a corner of the Vicon room and is able to reconstruct a semi-dense 3D representation of the recorded

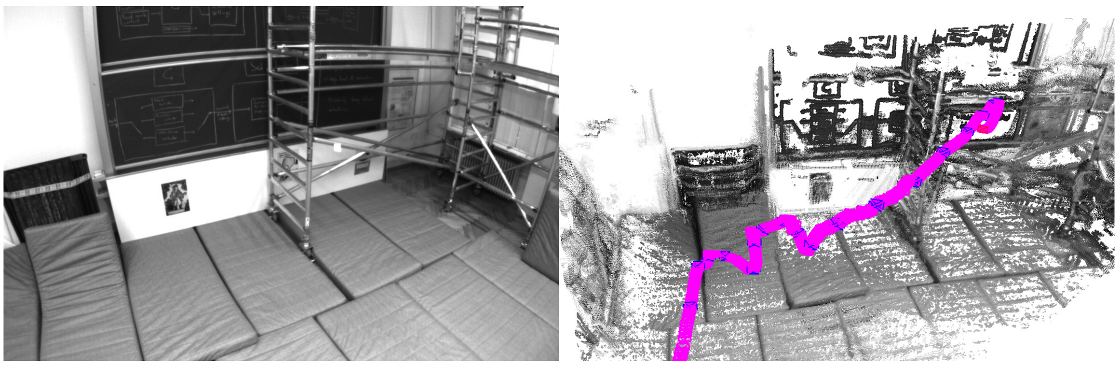


Figure 6.9: Exemplary results of flight Vicon_m: While the left image shows a capture of the recorded scene, the right image shows the retrieved camera trajectory and reconstructed semi-dense depth. Key frames are shown in blue, while feature-based tracked frames are shown in pink.

scene. As can be seen, details of the scaffold are retrieved as well as the ground plane and drawings on the blackboard. The recovered camera trajectory is shown, too. Key frames are colored in blue, while frames that were tracked feature-based with LIBVISO2 are shown in pink.

In total, we showed that our method is more robust to dynamic motions than the other evaluated methods and achieves a lower ATE on all evaluated datasets.

6.3 MAV

In the previous chapters we outlined, that our semi-direct approach is capable of accurate pose estimation with standard stereo cameras. As our MAV is equipped with fish eye lenses and a wide baseline, we assess our method using different datasets, that have been acquired with our MAV. We use laser-based SLAM (Droeschel et al., 2015) as ground truth and again compare the results to state-of-the-art SLAM methods. In total we captured four flights in a decommissioned car service station with challenging lighting conditions. While on the first two flights, named rect1 and rect2, the MAV covers a rectangular path without many loop closures, the other two flights, loop1 and loop2, include three to four full loops.

A general prerequisite for stereo computation is to rectify the images. As described in Chapter 4 there exist different methods for camera calibration and rectification. To allow different models for calibration we build a general rectification nodelet in ROS, that rectifies the images given respective look-up tables as input. The look-up tables can be either calculated offline beforehand or on-

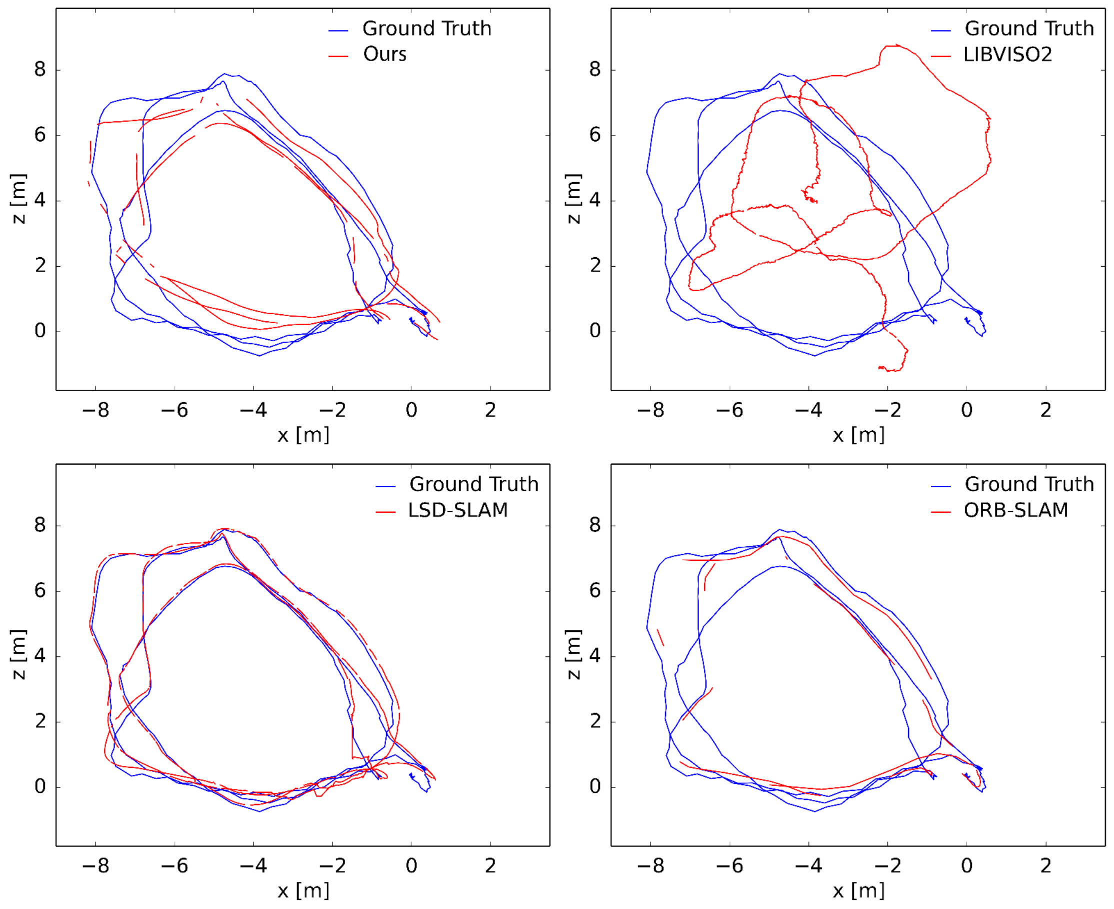


Figure 6.10: Results for MAV loop 1. Comparison of our method to LIBVISO2 (top row), LSD-SLAM and ORB-SLAM (bottom row) on a challenging dataset that contains large loop closures. As can be seen the monocular methods perform best, while LIBVISO2 heavily accumulates drift. Still, our method is able to reconstruct the trajectory with an ATE of 0.63 m, while other S-PTAM fails completely.

line using, e.g., , the computer vision library OpenCV. The rectification nodelet publishes rectified images together with camera info messages, that contain the necessary calibration parameters from intrinsic and extrinsic calibration. Moreover, we added functionality to down-sample the rectified images by a factor c for further run time enhancement. The images are captured with full resolution of 1280×1024 in 16 bit-encoding and are down-sampled to half the resolution and 8 bit in the rectification step.

The rectification of the images runs in parallel for all six cameras and takes 1 ms for a single image, when down-sampling to half the original resolution. For an even smaller resolution of 320×256 the rectification takes 0.7 ms and for the full

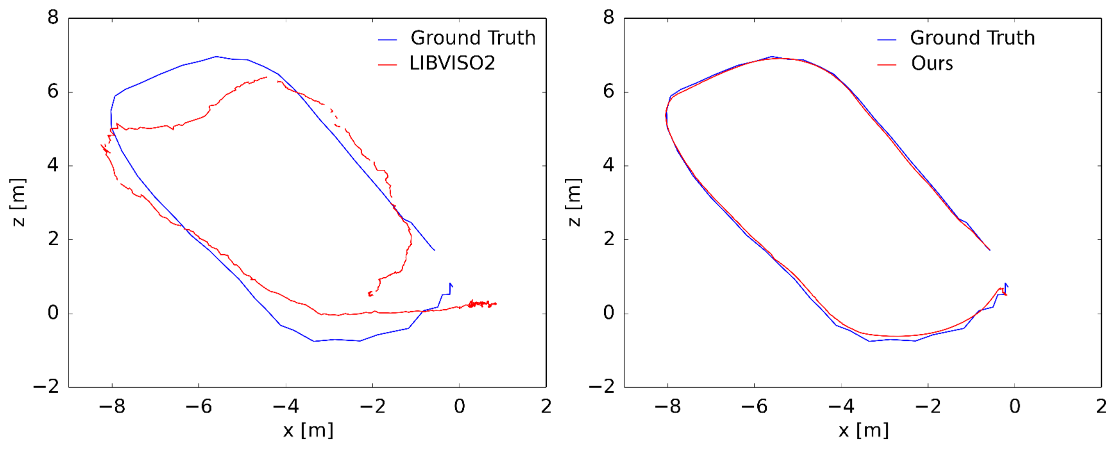


Figure 6.11: Comparison of LIBVISO2 (left) and our method (right) on the MAV dataset rect1. Even though, the dataset contains no loop closure, our method shows accurate results.

resolution 4 ms.

Experiments on the four flights show, that retrieving the correct camera motion is more challenging than on the previous datasets. Especially, the stereo methods suffer at these datasets. S-PTAM fails to initialize correspondences on all datasets and thus can not be taken into account for comparison.

Exemplary results are shown for a trajectory with repeating loop closures in Figure 6.10. It can be seen, that the stereo methods Semi-Direct SLAM and LIBVISO2 show a higher offset to the ground truth trajectories than the monocular methods. Especially LIBVISO2 accumulates high errors at this circular trajectory and the result is not as accurate as before, thereby limiting the output of the semi-direct approach. As LIBVISO2 performs no loop closure detection, errors in the absolute trajectory can not be resolved, which leads to a globally inconsistent trajectory. Semi-Direct SLAM uses only the relative motion estimates of LIBVISO with regards to the current key frame. Thereby, Semi-Direct SLAM is still able to reconstruct a path close to the ground truth with an ATE of 0.63 m. Contrarily, LSD-SLAM and ORB-SLAM achieve an ATE below 0.31 m.

The fact, that monocular methods seem to perform better than stereo methods, suggests that the underlying projection model for the stereo calibration might be inaccurate. Additionally, the non-rigid attachment of the stereo cameras introduces difficult conditions for stereo correspondence search along horizontal lines. We assume, that the wide non-rigid baseline of 53.37 cm in combination with the perspective rectification onto a plane, raise difficulties at the stereo correspondence search. It would generally be more appropriate to model the fish eye lenses as a rectification onto a sphere. As described in Section 5.4.2, we use an additional

MAV Dataset	Absolute Trajectory Error RMSE (Median) in m			
	Ours	Libviso2	LSD-SLAM	ORB-SLAM
rect1	**0.13 (0.11)**	1.24 (0.49)	0.30 (0.29)	0.98 (0.24)
rect2	0.84 (0.81)	1.61 (1.59)	**0.38 (0.37)**	0.59 (0.25)
loop1	0.63 (0.57)	1.66 (0.99)	0.31 (0.28)	**0.25 (0.21)**
loop2	1.58 (0.71)	2.61 (1.90)	**0.54 (0.42)**	1.19 (0.78)
mean	0.80 (0.55)	1.78 (1.24)	**0.38 (0.34)**	0.75 (0.37)

Table 6.3: ATE Results on MAV Dataset.

weighting scheme, that down-weights the influence of inaccurate depth measurements close to the image borders, to cope with strong distortions. Moreover, we repeatedly estimate the extrinsic transformation of the cameras online to cope with the non-rigidity. Therefore, we are able to retrieve stereo correspondences and estimate the trajectory on this challenging dataset, in contrast to S-PTAM, which fails to initialize any correspondences.

Figure 6.11 shows trajectories for the sequence rect1 computed by LIBVISO2 and Semi-Direct SLAM. The output of LIBVISO2 shows very noisy estimates and leads to a comparable high ATE of 1.24 m. In contrast Semi-Direct SLAM produces a smoother trajectory with an ATE of 0.13 m. However, in general we achieve a higher ATE than the monocular methods. Table 6.3 summarizes the resulting ATE on all datasets.

In terms of accuracy the monocular methods perform better than all stereo methods. This time S-PTAM is unable to track features on all datasets and fails in recovering any motion. It is remarkable, that monocular methods perform better than stereo methods on these datasets, which leads to the assumption that the rectification of the fish eye images onto a plane in combination with non-rigid stereo cameras is very challenging for stereo computations. Moreover, the wide baseline is demanding as the image overlap between both stereo images is reduced.

On average, we achieve an ATE of 0.8 m, while LSD-SLAM achieves an average ATE of 0.38 m.

6.4 Odometry versus SLAM

In this section, we will compare the quantitative results of visual odometry to visual SLAM. As visual odometry tends to drift over time, global optimization methods such as bundle adjustment or pose-graph optimization help to reduce the drift.

EuRoC	**Absolute Trajectory Error RMSE (Median) in m and Improvement in %**		
Dataset	Our Semi-Direct VO (m)	Our Semi-Direct SLAM (m)	Improvement (%)
V1_01	0.26 (0.18)	0.12 (0.11)	53.85 (38.89)
V1_02	0.59 (0.59)	0.11 (0.10)	81.36 (83.05)
V1_03	0.81 (0.76)	0.75 (0.44)	7.41 (42.11)
V2_01	0.22 (0.13)	0.18 (0.12)	18.18 (7.69)
V2_02	0.31 (0.25)	0.27 (0.22)	12.90 (12.00)
V2_03	1.13 (0.97)	0.87 (0.66)	23.01 (31.96)
mean	0.55 (0.48)	0.38 (0.28)	30.72 (42.71)

Table 6.4: Odometry compared to SLAM on EuRoC.

In Semi-Direct SLAM loop-closures are detected between key frames and are added as additional constraints to the global pose-graph (see Section 5.4.3).

The trajectories of the EuRoC dataset contain many possible loop-closures. Therefore, we show comparative results between visual odometry and SLAM exemplary on this dataset. Qualitative results are listed in Table 6.4. In addition to the ATE as error measure, we also state the percentage improvement gained by SLAM. We measure the improvement as

$$Improvement = \frac{VO - SLAM}{VO}. \tag{6.1}$$

The average improvement for all seven trajectories lies at 30.72%, denoting an absolute improvement of 0.17 m on average. It can clearly be seen, that for each trajectory the odometry result is further improved by SLAM. The improvements range from 7.41% up to 81.36%. The maximum improvement reached an absolute enhancement of 0.48 m. As the trajectories V1_01 and V1_02 show distinct improvements of 53.85% and 81.36% respectively, both results are visualized in Figure 6.12. The advantages of SLAM are visible in both examples. In comparison to the pure odometry, SLAM retrieves trajectories closer to the ground truth. The bottom row of Figure 6.12 highlights the improvement of 81.36% on dataset V1_02. This dataset is of medium difficulty and contains very dynamic translational and rotational movements. It can be seen, that the odometry might be locally accurate, but exhibits accumulated drift. In the global graph SLAM the drift is corrected by loop closures, resulting in a better aligned trajectory.

In contrast to the EuRoC dataset the KITTI dataset shows notably less loop closure possibilities. However, when loop-closures are found the global consistency of the map is re-established. Sequence 06 contains a distinct loop. While visual odometry produces an ATE of 4.37 m on Sequence 06, the result is corrected after closing the loop and the ATE decreases to 2.06 m, showing an improvement of

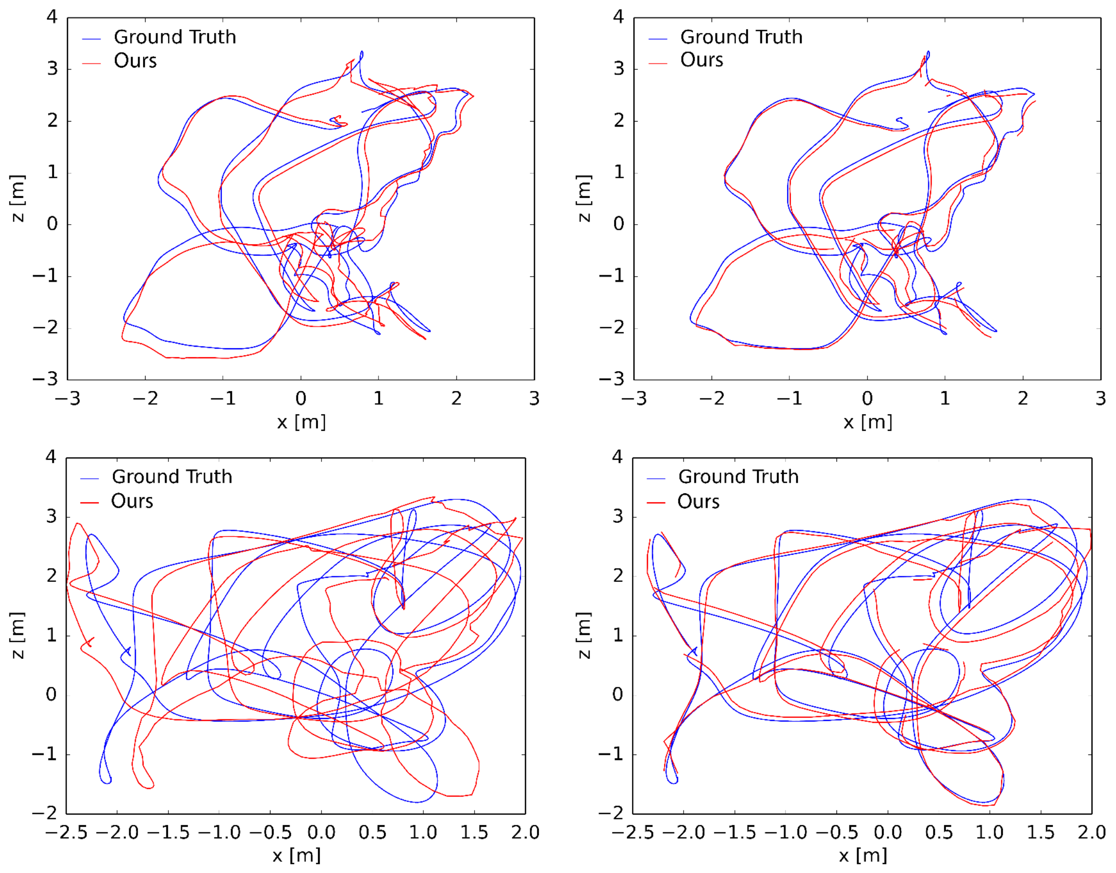


Figure 6.12: Comparison of Semi-Direct Odometry (left) and Semi-Direct SLAM (right) on the EuRoC dataset V1_01 and V1_02. With SLAM loop closures are found and accumulated drift is corrected, yielding percentage improvements of 53.85% and 81.36% respectively

52.9%. Figure 6.13 visualizes this behavior: while the odometry drifts over time and does not retrieve the circular path, the SLAM extension closes the loop and continues the trajectory on the previous driven path.

Additionally, we also evaluate the performance of SLAM in comparison to pure odometry on our MAV. Similarly to above results, loop closures greatly help to reduce the drift on the datasets loop1 and loop2. While on dataset loop1 the odometry yields an estimate with 1.1 m ATE, the visual SLAM recovers the camera motion with 0.63 m. On dataset loop2 the odometry result improves from 2.16 m to 1.58 m, when performing SLAM. The percentage improvements on these datasets are 42.7% and 26.9% respectively.

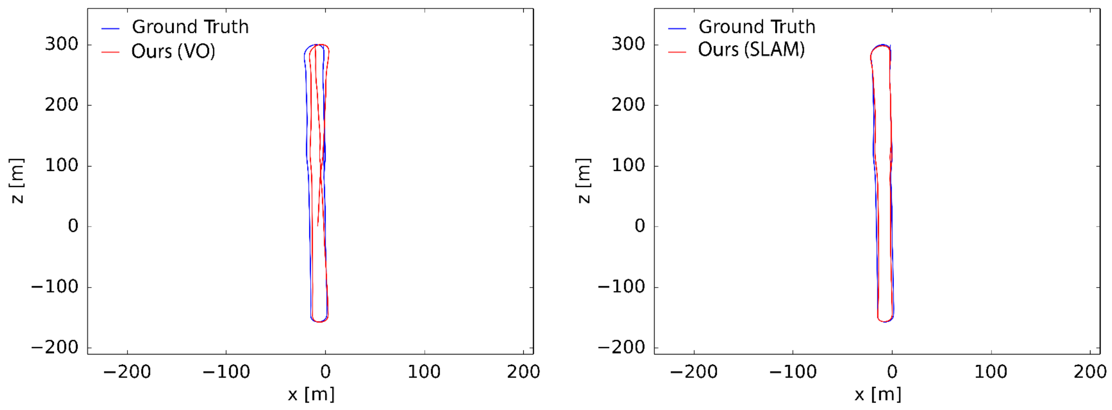


Figure 6.13: Comparison of Semi-Direct Odometry (left) and Semi-Direct SLAM (right) on KITTI Sequence 06. Example of a full loop closure found by our SLAM method, while pure odometry drifts. SLAM achieves an improvement of 52.9%.

	Absolute Trajectory Error RMSE (Median) in m				
Dataset	Ours	Libviso2	LSD-SLAM	ORB-SLAM	S-PTAM
KITTI	**10.28 (8.53)**	32.54 (26.42)	X	41.49 (35.66)	25.74 (20.26)
EuRoC	**0.38 (0.28)**	0.85 (0.71)	0.53 (0.48)	1.23 (0.94)	1.82 (1.52)
MAV	0.80 (0.55)	1.78 (1.24)	**0.38 (0.34)**	0.75 (0.37)	X

Table 6.5: Average ATE Results on the different evaluated datasets.

6.5 Accuracy

We have shown on different challenging datasets that in terms of accuracy we achieve similar results as current state-of-the-art stereo methods. The mean results for all datasets are summarized in Table 6.5. As can be seen in the table, our method achieves a lower ATE than the other evaluated methods on the KITTI and EuRoC datasets. On our MAV monocular methods outperform the stereo methods. However, in comparison to the other stereo methods, our approach performs better and more robust.

Moreover, we measure relative pose errors as proposed by Geiger, Lenz, et al.

RPE	Ours (VO)	Libviso2	Direct VO
Translation Error (%)	**0.8061**	0.8449	0.8168
Rotation Error (deg /m)	**0.0051**	0.0052	0.0053

Table 6.6: Relative pose errors of the odometry methods. Translational drift is measured in percentage and rotational drift in deg /m.

Dataset	Method	Tracking	Mapping	Constraint Search	Optimization	Total (VO)	Total (SLAM)
KITTI	Ours	26.5 ms	36.6 m	253.5 m	564.6 m	63.1 m	881.2 m
	LSD-SLAM	-	-	-	-	-	-
	ORB-SLAM	30.7 ms	254.0 ms	7.8 ms	1315.6 ms	284.6 ms	1608.0 ms
	S-PTAM	71.1 ms	5.7 ms	-	2036.9 ms	77.4 ms	2114.3 ms
	LIBVISO2	33.8 ms	-	-	-	33.8 ms	-
EuRoC	Ours	22.6 ms	39.6 m	153.5 m	684.2 m	62.2 m	899.9 m
	LSD-SLAM	27.6 ms	85.6 ms	158.1 ms	207.3 ms	113.2 ms	478.5 ms
	ORB-SLAM	17.9 ms	159.2 ms	3.7 ms	535.6 ms	177.1 ms	716.4 ms
	S-PTAM	47.3 ms	1.5 ms	-	976.9 ms	48.8 ms	1025.7 ms
	LIBVISO2	24.8 ms	-	-	-	24.8 ms	-
MAV	Ours	17.5 ms	25.8 ms	140.0 ms	130.7 ms	43.3 ms	313.3 ms
	LSD-SLAM	28.7 ms	67.3 ms	314.0 ms	637.3 ms	79.0 ms	951.3 ms
	ORB-SLAM	24.3 ms	221.2 ms	11.0 ms	353.8 ms	245.5 ms	610.3 ms
	S-PTAM	-	-	-	-	-	-
	LIBVISO2	25.3 ms	-	-	-	25.3 ms	-

Table 6.7: Average runtimes of all evaluated methods.

(2012) to measure the performance and drift of pure odometry over large-scale sequences as in the KITTI dataset. Table 6.6 summarizes the results of our Semi-Direct Odometry in comparison to LIBVISO2 and Direct Odometry. Translational and rotational errors are measured separately. Results show, that our method shows less translational and rotational drift over time. Moreover, as already seen above in the exemplary trajectory plots, the fully direct odometry has a higher rotational error than the other methods, as at large rotations direct alignment of frames becomes harder.

In summary, our semi-direct approach shows accurate results for all datasets. Even on challenging fish eye stereo the whole trajectory can be retrieved and loop closures are found, while S-PTAM fails to find any correspondences.

6.6 Runtime

For state-estimation with visual odometry or SLAM real-time capabilities build an important factor. We thereby measure the efficiency of our method in terms of average runtime in ms.

We measure the average runtime as well as the runtime of the different blocks, because often it is sufficient if tracking can be done with high frequency, as global optimization usually does not run in real-time. The runtimes are broken down to the individual blocks: tracking, mapping, constraint-search and pose-graph optimization. Timings for all datasets are listed in Table 6.7. Missing values are denoted with '-', e.g., S-PTAM does not perform a constraint search as the other methods, and LIBVISO only does tracking. The table clearly highlights, that the SLAM parts, consisting of the constraint search and pose-graph optimization are the bottleneck for all systems.

In general, it can be seen that our approach is able to track incoming frames with 30 Hz. The mapping thread runs in parallel to tracking at approximately 30 Hz, too. However, global optimization is still very costly for all methods. Especially in large-scale sequences the runtime rises.

6.7 Qualitative Results

A major advantage of our semi-direct approach is, that 3D point clouds are estimated at runtime, yielding an accurate semi-dense reconstruction of the environment. Thus, we are not only able to estimate the current pose of the camera, but also maintain a 3D map of the environment, which can be used for additional tasks, like obstacle avoidance.

Qualitative results are shown for sequence 00 of the KITTI dataset exemplary. As can be seen in Figure 6.14 an accurate and consistent 3D reconstruction is achieved by Semi-Direct SLAM. For better comparison to feature-based SLAM methods, the resulting sparse map built by ORB-SLAM is shown in Figure 6.15. While the reconstruction of ORB-SLAM only contains sparse points, our reconstruction allows detailed inference to existing objects in the scene. Most objects, that are visible in the camera image, can be recovered in our semi-dense map. For example, one can clearly distinguish between individual trees and cars. Contrarily, in the sparse map of ORB-SLAM one can vaguely guess, where the street runs.

Figure 6.16 shows the estimated pose-graph of the camera trajectory and reconstructed map of the medium difficult EuRoC dataset V1_02. The images prove, that our estimated 3D reconstruction is globally consistent. The objects shown in the exemplary given camera image can easily be retrieved in the reconstructed map.

In conclusion, we state that our method builds globally consistent semi-dense 3D maps of the environment. It is well suited for large-scale sequences as in the KITTI dataset, as well as for smaller indoor sequences as in the EuRoC dataset. We believe, that the semi-dense 3D reconstruction serves a great advantage for autonomous visual navigation.

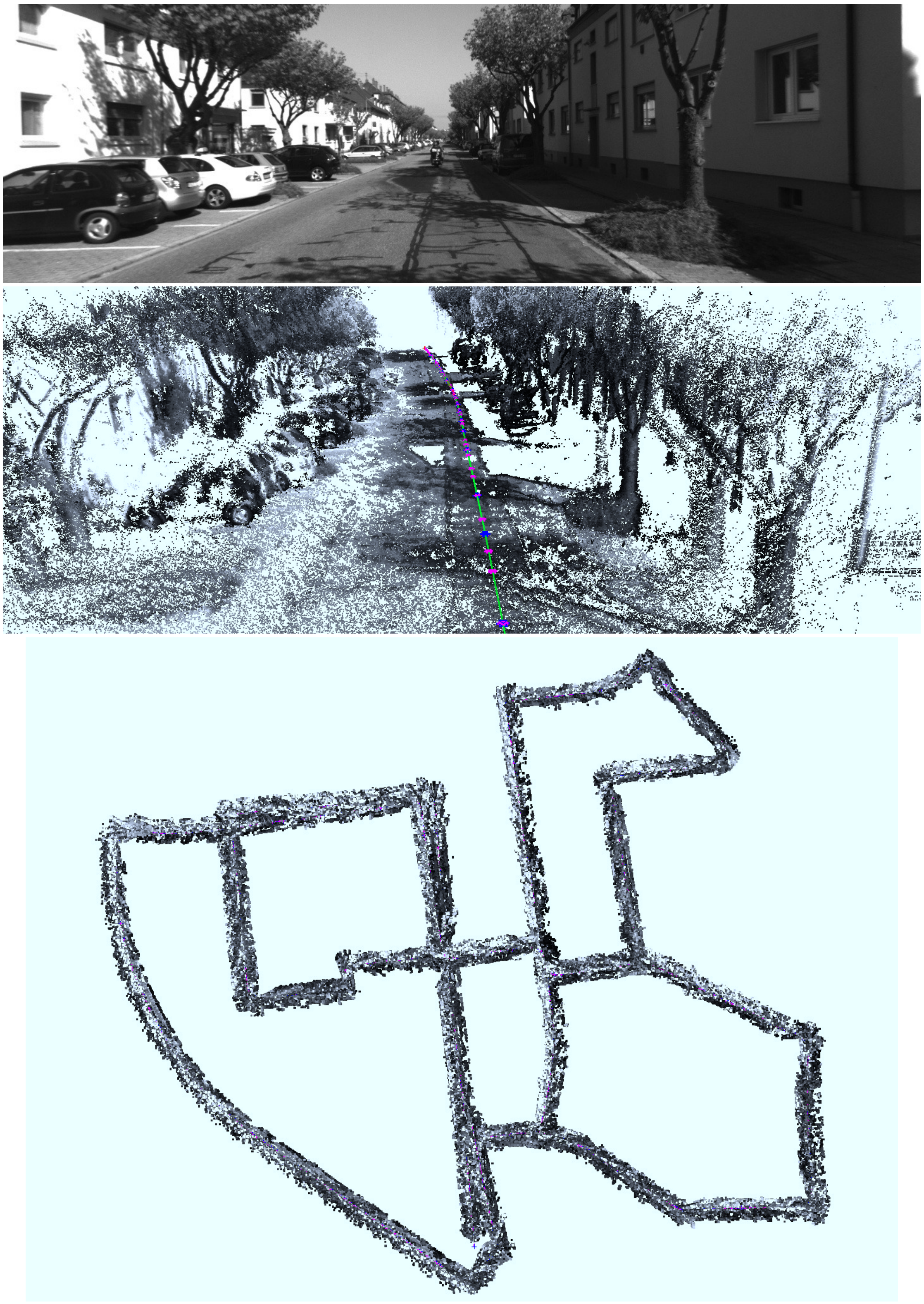


Figure 6.14: Semi-dense 3D Reconstruction of KITTI 00: The top image shows the reconstructed scene as captured by the camera. Below the semi-dense 3D reconstruction of this scene and the complete reconstruction of this dataset is shown.

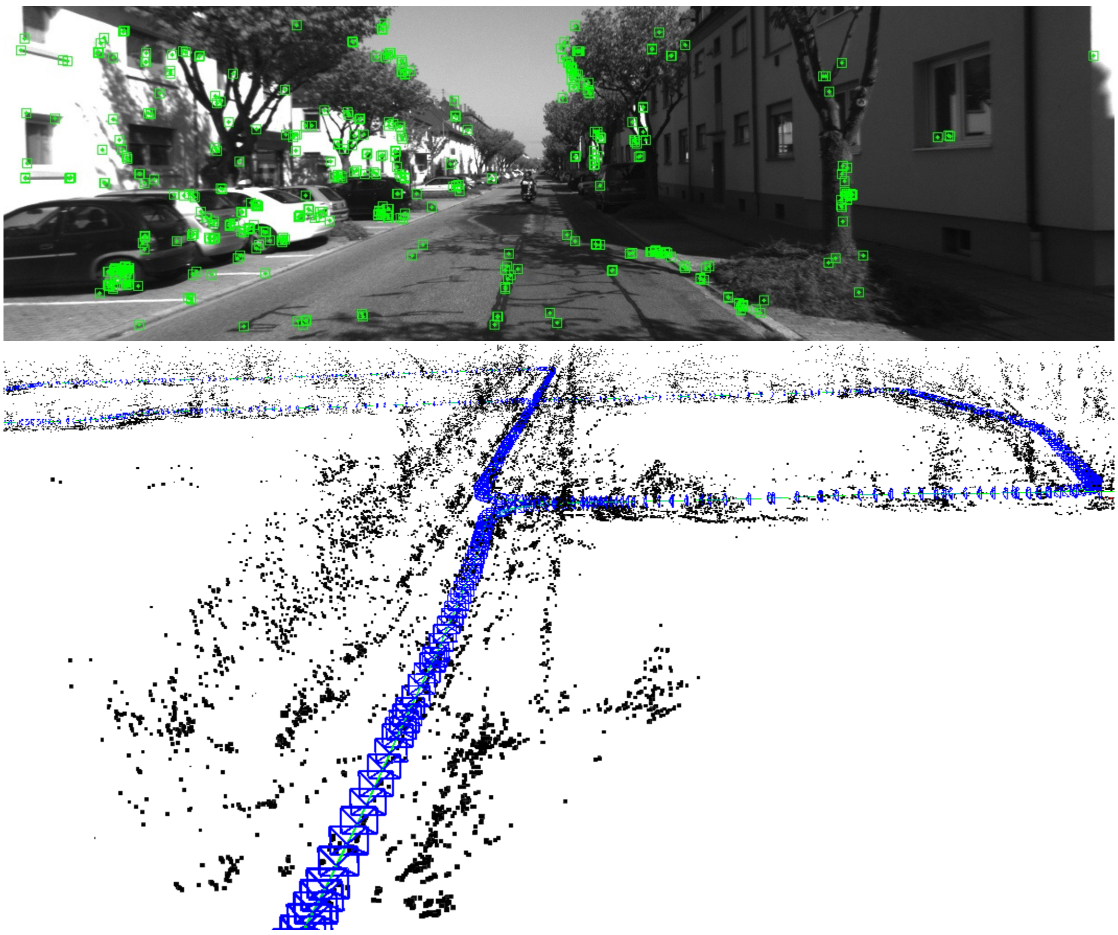


Figure 6.15: Sparse feature-based 3D Reconstruction of KITTI 00 by ORB-SLAM. The top view shows an exemplary scene, where ORB features are tracked. The lower image shows the sparse map, that is obtained by tracking ORB features.

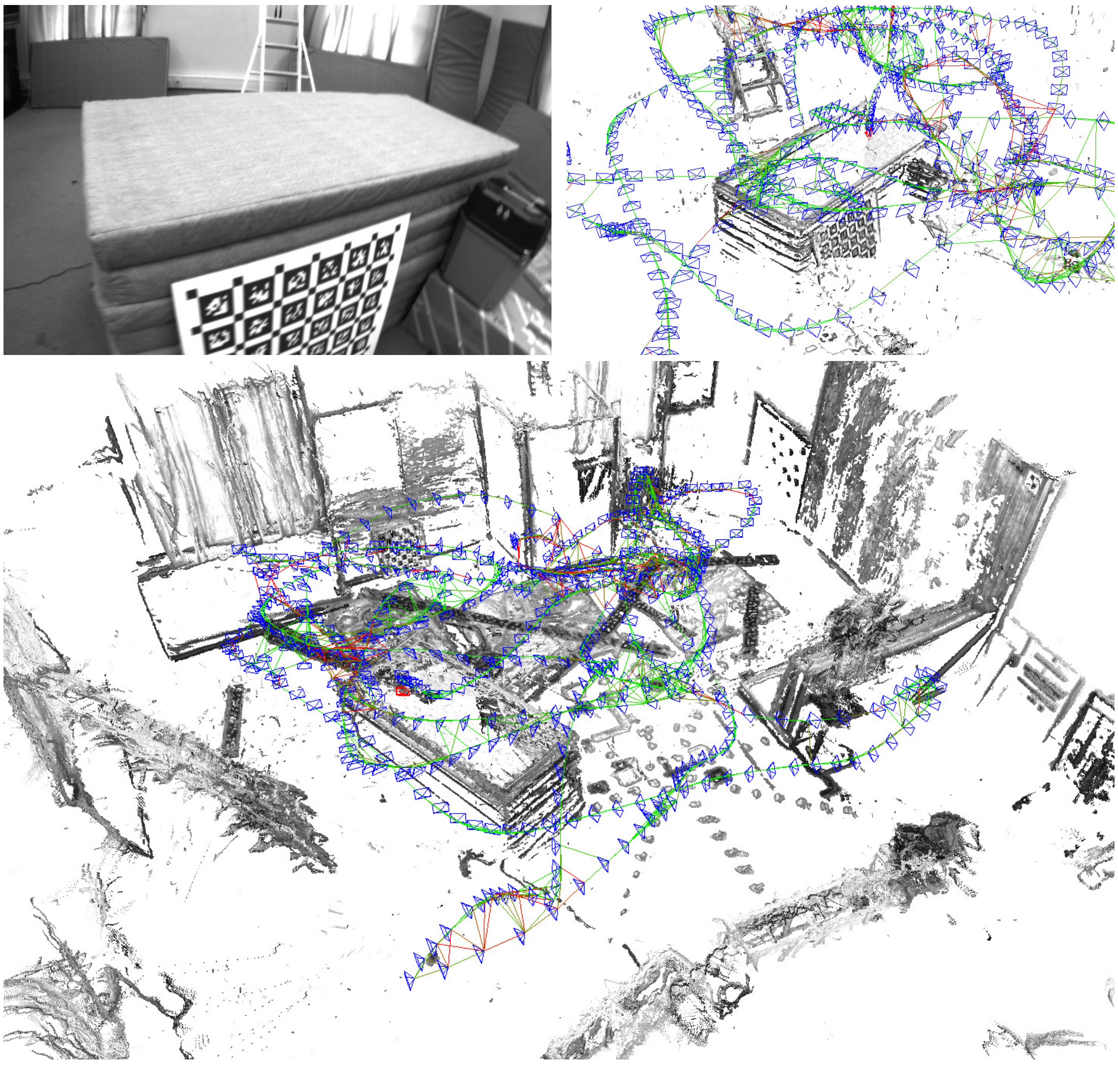


Figure 6.16: Semi-dense 3D Reconstruction of the EuRoC Dataset V1_02 with medium difficulty. Results show a globally consistent semi-dense map. The depicted key frame graph visualizes the trajectory. Key frames are shown in blue, while edge-constraints are shown in green and red, depending on their confidence.

7 Conclusion

In this thesis, we proposed a novel semi-direct stereo SLAM method which combines direct image alignment with feature-based matching.

By combining direct image alignment and feature-based matching, our method can track images with high frame rate in real-time. The feature-based matching serves as initialization for direct tracking on key frames. Thereby, our approach is robust to large motions and self-rotations, whereas pure direct methods tend to fail.

The performance of the method has been evaluated quantitatively in terms of accuracy and runtime. In experimental evaluation on challenging datasets, we show that our approach is well-suited for autonomous indoor and outdoor navigation. Moreover, the evaluation shows that in the majority of the experiments, our approach achieves higher accuracy than state-of-the-art methods without forfeiting performance. The method runs in real-time on the original resolution of the KITTI dataset without the necessity to reduce the image resolution. By using feature-based matching as prior for the direct alignment our method is computationally less expensive as convergence is obtained faster and thus well-suited for the use on a MAV to estimate the egomotion in real-time. Experiments on the EuRoC dataset show that our approach is robust to aggressive flight maneuvers with strong rotations and motion blur. While other methods tend to lose track at the highly dynamic movements of the MAV, our method recovers the full trajectory and simultaneously builds a semi-dense map.

Experiments on flights captured from our MAV, that is equipped with non-rigid fish eye stereo, show limitations of our approach. Strong distortions, induced by the projective rectification onto a plane, and a non-rigid stereo camera rig lead to inferior behavior than to datasets captured with a rigidly attached camera rig and normal lenses. We cope with these problems by introducing a different weighting scheme and by re-estimating the extrinsics of our camera system on the flight, following that our method was able to recover the path with an average ATE below 0.8 m, while other stereo methods completely fail or produce an ATE above 1 m. Aswhile on the KITTI dataset our method achieves a percentage improvement of 60% to the second best method, we achieve an improvement of 43% to the second best on the EuRoC dataset.

Additionally, we compare the performance of full SLAM to the performance of pure visual odometry. Although, our visual odometry shows an average relative error below 1%, the accuracy of the absolute trajectory is improved up to 80% by visual SLAM. We show that all estimates can be further improved by SLAM.

The qualitative evaluation of our approach shows that we are able to reconstruct an accurate semi-dense 3D map of the world. In direct comparison to the sparse map of ORB-SLAM, our reconstruction contains far more details and is well suited for mapping obstacles.

Runtime evaluation shows that the constraint search and the global pose graph optimization pose the bottleneck of the system. As tracking of new frames runs in a separate thread with 30 Hz, real-time constraints can be fulfilled.

Since our method works very well for datasets captured with a normal lens, but is less accurate on datasets captured with fish eye lenses, we believe that the rectification of spherical images onto a plane might not be optimal for fish eye lenses. For a next step, we thereby suggest to employ a different camera model that is better suited to model the spherical projection. However, as such models are usually non-linear, this might lead to less efficient behavior.

To conclude, the semi-direct visual SLAM developed in this thesis shows accurate behavior in complex scenarios while still operating in real-time on high-resolutions. It is therefore well-suited for autonomous systems that need a reliable and fast state estimation. Experiments show that in terms of robustness and accuracy, our approach outperforms state-of-the-art methods, which often lose track at dynamic motions. For future work, integration of our 3D reconstruction to an OctoMap would be possible to better map the occupancy of the surroundings. Moreover, our method is build very modular so that other feature-based approaches, like ORB-SLAM or visual-inertial methods, instead of LIBVISO2, could be evaluated. Furthermore, extracted sparse points of the feature-based method, could also be fused into the depth map. All in all, we developed a semi-dense semi-direct visual SLAM method for stereo cameras, that achieves accurate results and builds consistent 3D reconstructions of the environment.

Bibliography

[1] S. Abraham. *TCC - A software for Test field based self-Calibration of multi-Camera-systems, documentation.* 2004 (cit. on p. 33).

[2] S. Abraham and W. Foerstner. “Fish-eye-stereo calibration and epipolar rectification”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 59.5 (2005), pp. 278–288 (cit. on p. 11).

[3] M. Achtelik, M. Achtelik, S. Weiss, and R. Siegwart. “Onboard IMU and Monocular Vision Based Control for MAVs in Unknown In- and Outdoor Environments”. In: *Proc. of the IEEE International Conference on Robotics and Automation (ICRA).* 2011 (cit. on p. 24).

[4] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. “Speeded-Up Robust Features (SURF)”. In: *Comput. Vis. Image Underst.* 110.3 (June 2008), pp. 346–359 (cit. on p. 19).

[5] M. Beul, N. Krombach, Y. Zhong, D. Droeschel, M. Nieuwenhuisen, and S. Behnke. “A high-performance MAV for autonomous navigation in complex 3D environments”. In: *Unmanned Aircraft Systems (ICUAS), 2015 International Conference on.* 2015 (cit. on p. 5).

[6] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua. “BRIEF: Computing a Local Binary Descriptor Very Fast”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.7 (2012), pp. 1281–1298 (cit. on p. 19).

[7] A. Comport, E. Malis, and P. Rives. “Accurate Quadrifocal Tracking for Robust 3D Visual Odometry”. In: *Robotics and Automation, 2007 IEEE International Conference on.* Apr. 2007, pp. 40–45 (cit. on p. 26).

[8] A. Davison, I. Reid, N. Molton, and O. Stasse. “MonoSLAM: Real-Time Single Camera SLAM”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 29.6 (June 2007), pp. 1052–1067 (cit. on p. 24).

[9] D. Droeschel, M. Nieuwenhuisen, M. Beul, J. Stueckler, D. Holz, and S. Behnke. “Multi-Layered Mapping and Navigation for Autonomous Micro Aerial Vehicles”. In: *Journal of Field Robotics* (2015). to appear (cit. on p. 63).

[10] J. Engel, T. Schöps, and D. Cremers. “LSD-SLAM: Large-Scale Direct Monocular SLAM”. In: *Proc. of the European Conference on Computer Vision (ECCV).* Sept. 2014 (cit. on pp. 1, 27, 35, 36, 51).

[11] J. Engel, J. Stueckler, and D. Cremers. "Large-Scale Direct SLAM with Stereo Cameras". In: *Proc. of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS).* to appear. Sept. 2015 (cit. on p. 27).

[12] J. Engel, J. Sturm, and D. Cremers. "Semi-Dense Visual Odometry for a Monocular Camera". In: *Proc. of the IEEE Int. Conference on Computer Vision (ICCV).* Sydney, Australia, Dec. 2013 (cit. on p. 38).

[13] C. Forster, M. Pizzoli, and D. Scaramuzza. "SVO: Fast Semi-Direct Monocular Visual Odometry". In: *IEEE International Conference on Robotics and Automation (ICRA).* 2014 (cit. on p. 27).

[14] W. Förstner. "A feature based correspondence algorithm for image matching". In: *ISP Comm. III.* 1986 (cit. on p. 19).

[15] A. Geiger, P. Lenz, and R. Urtasun. "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite". In: *Conference on Computer Vision and Pattern Recognition (CVPR).* 2012 (cit. on pp. 51, 52, 69).

[16] A. Geiger, M. Roser, and R. Urtasun. "Efficient Large-Scale Stereo Matching". In: *Asian Conference on Computer Vision (ACCV).* 2010 (cit. on p. 44).

[17] A. Geiger, J. Ziegler, and C. Stiller. "StereoScan: Dense 3D Reconstruction in Real-time". In: *Intelligent Vehicles Symposium (IV).* 2011 (cit. on pp. 25, 35, 51).

[18] C. G. Harris and J. M. Pike. "3D Positional Integration from Image Sequences". In: *Image Vision Comput.* 6.2 (May 1988), pp. 87–90 (cit. on p. 19).

[19] M. Irani and P. Anandan. "About Direct Methods". In: *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice.* ICCV '99. London, UK, UK: Springer-Verlag, 2000, pp. 267–277 (cit. on pp. 21, 22, 26, 57).

[20] S. Kammel, J. Ziegler, B. Pitzer, M. Werling, T. Gindele, D. Jagzent, J. Schröder, M. Thuy, M. Goebl, F. v. Hundelshausen, O. Pink, C. Frese, and C. Stiller. "Team AnnieWAY's autonomous system for the 2007 DARPA Urban Challenge". In: *Journal of Field Robotics* 25.9 (2008), pp. 615–639 (cit. on p. 52).

[21] J. Kannala and S. Brandt. "A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 28.8 (Aug. 2006), pp. 1335–1340 (cit. on p. 33).

[22] C. Kerl, J. Sturm, and D. Cremers. “Dense Visual SLAM for RGB-D Cameras”. In: *Proc. of the Int. Conf. on Intelligent Robot Systems (IROS).* 2013 (cit. on p. 27).

[23] G. Klein and D. Murray. “Parallel Tracking and Mapping for Small AR Workspaces”. In: *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR’07).* Nara, Japan, Nov. 2007 (cit. on pp. 24, 27).

[24] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. “G2o: A general framework for graph optimization”. In: *Robotics and Automation (ICRA), 2011 IEEE International Conference on.* May 2011, pp. 3607–3613 (cit. on pp. 39, 48).

[25] A. S. Lab. *The EuRoC MAV Dataset.* ETH Zürich. 2015. URL: `http://projects.asl.ethz.ch/datasets/doku.php?id=kmavvisualinertialdatasets` (visited on 01/18/2016) (cit. on p. 61).

[26] D. G. Lowe. “Distinctive Image Features from Scale-Invariant Keypoints”. In: *Int. J. Comput. Vision* 60.2 (Nov. 2004), pp. 91–110 (cit. on p. 19).

[27] B. D. Lucas and T. Kanade. “An Iterative Image Registration Technique with an Application to Stereo Vision”. In: *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2.* IJCAI’81. Vancouver, BC, Canada, 1981, pp. 674–679 (cit. on p. 26).

[28] J. Maye, P. Furgale, and R. Siegwart. “Self-supervised calibration for robotic systems”. In: *Intelligent Vehicles Symposium (IV), 2013 IEEE.* June 2013, pp. 473–480 (cit. on p. 33).

[29] L. Meier, P. Tanskanen, L. Heng, G. H. Lee, F. Fraundorfer, and M. Pollefeys. “PIXHAWK: A Micro Aerial Vehicle Design for Autonomous Flight Using Onboard Computer Vision”. In: *Auton. Robots* 33.1-2 (Aug. 2012), pp. 21–39 (cit. on p. 5).

[30] H. P. Moravec. “Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover”. AAI8024717. PhD thesis. Stanford, CA, USA, 1980 (cit. on p. 25).

[31] T. Mori and S. Scherer. “First Results in Detecting and Avoiding Frontal Obstacles from a Monocular Camera for Micro Unmanned Aerial Vehicles”. In: *Proc. of the IEEE Int. Conference on Robotics and Automation (ICRA).* 2013 (cit. on p. 23).

[32] R. Mur-Artal, J. Montiel, and J. Tardos. “ORB-SLAM: A Versatile and Accurate Monocular SLAM System”. In: *Robotics, IEEE Transactions on* 31.5 (Oct. 2015), pp. 1147–1163 (cit. on pp. 25, 51).

[33] R. A. Newcombe, S. Lovegrove, and A. J. Davison. "DTAM: Dense tracking and mapping in real-time". In: *IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011.* 2011, pp. 2320–2327 (cit. on p. 27).

[34] M. Nieuwenhuisen, D. Droeschel, M. Beul, and S. Behnke. "Autonomous Navigation for Micro Aerial Vehicles in Complex GNSS-denied Environments". In: *Journal of Intelligent & Robotic Systems* (2015), pp. 1–18 (cit. on p. 5).

[35] J. Nikolic, J. Rehder, M. Burri, P. Gohl, S. Leutenegger, P. T. Furgale, and R. Siegwart. "A synchronized visual-inertial sensor system with FPGA pre-processing for accurate real-time SLAM". In: *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014.* 2014, pp. 431–437 (cit. on p. 60).

[36] D. Nister, O. Naroditsky, and J. Bergen. "Visual odometry". In: *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on.* Vol. 1. June 2004 (cit. on pp. 15, 25).

[37] T. Pire, T. Fischer, J. Civera, P. De Cristóforis, and J. Jacobo berlles. "Stereo Parallel Tracking and Mapping for robot localization". In: *Proc. of The International Conference on Intelligent Robots and Systems (IROS) (Accepted).* 2015 (cit. on p. 51).

[38] S. Ross, N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. Bagnell, and M. Hebert. "Learning Monocular Reactive UAV Control in Cluttered Natural Environments". In: *Proc. of the IEEE Int. Conference on Robotics and Automation (ICRA).* 2013 (cit. on p. 23).

[39] E. Rosten and T. Drummond. "Machine Learning for High-speed Corner Detection". In: *Proceedings of the 9th European Conference on Computer Vision - Volume Part I.* ECCV'06. Graz, Austria, 2006, pp. 430–443 (cit. on pp. 19, 24).

[40] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. "ORB: An Efficient Alternative to SIFT or SURF". In: *Proceedings of the 2011 International Conference on Computer Vision.* ICCV '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 2564–2571 (cit. on p. 25).

[41] D. Scaramuzza and F. Fraundorfer. "Visual Odometry [Tutorial]". In: *Robotics Automation Magazine, IEEE* 18.4 (Dec. 2011), pp. 80–92 (cit. on pp. 15, 19, 23).

[42] D. Scaramuzza, A. Martinelli, and R. Siegwart. "A Toolbox for Easily Calibrating Omnidirectional Cameras". In: *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on.* Oct. 2006, pp. 5695–5701 (cit. on p. 31).

[43] K. Schauwecker and A. Zell. "On-Board Dual-Stereo-Vision for the Navigation of an Autonomous MAV". In: *Journal of Intelligent & Robotic Systems* 74.1-2 (2014), pp. 1–16 (cit. on p. 23).

[44] K. Schmid, P. Lutz, T. Tomic, E. Mair, and H. Hirschmüller. "Autonomous Vision-based Micro Air Vehicle for Indoor and Outdoor Navigation". In: *Journal of Field Robotics* 31.4 (2014), pp. 537–570 (cit. on p. 23).

[45] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. "A Benchmark for the Evaluation of RGB-D SLAM Systems". In: *Proc. of the International Conference on Intelligent Robot Systems (IROS).* Oct. 2012 (cit. on p. 51).

[46] S. Weiss, M. Achtelik, S. Lynen, M. Chli, and R. Siegwart. "Real-time onboard visual-inertial state estimation and self-calibration of MAVs in unknown environments". In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on.* May 2012, pp. 957–964 (cit. on p. 23).

[47] S. Weiss, D. Scaramuzza, and R. Siegwart. "Monocular-SLAM-based navigation for autonomous micro helicopters in GPS-denied environments". In: *Journal of Field Robotics* 28.6 (2011), pp. 854–874 (cit. on pp. 24, 25).