UNIVERSITÄT
KOBLENZ · LANDAU
Fachbereich 4: Informatik

An On-Board Stereo Visual-Inertial Odometry System for an Unmanned Aerial Vehicle

Masterarbeit
zur Erlangung des Grades
Master of Science
im Studiengang Computervisualistik

vorgelegt von

Stephan Manthe

Betreuer: Prof. Pascual Campoy, Centro Automática y Robótica, Universidad Politecnica de Madrid
Betreuer: M.Sc. Elektronik und Automatisierung, Adrián Carrio, Centro Automática y Robótica, Universidad Politecnica de Madrid
Erstgutachter: Prof. Dr.-Ing. Dietrich Paulus, Institut für Computervisualistik, Fachbereich Informatik, Universität Koblenz-Landau
Zweitgutachter: Dipl.-Inform. Frank Neuhaus, Institut für Computervisualistik, Fachbereich Informatik, Universität Koblenz-Landau

Koblenz, im April 2017

Kurzfassung

Die Selbstlokalisierung und Bewegungsschätzung gehört zu den Hauptaufgaben eines unbemannten Luftfahrzeugs (englisch unmanned aerial vehicle, UAV). Durch diese Fähigkeiten erlangt ein solches System zu einen Mehrwert, da es ihm ermöglicht autonom zu agieren. Für ein höchstmögliches Maß an Autonomie in jeglicher Umgebung soll ein solches System nicht auf externe Referenzsysteme wie z. B. Satellitennavigationssysteme angewiesen sein. Dies kann mit einem Sensorsystem an Bord eines UAVs, das eine Stereokamera und eine inertiale Messeinheit (englisch inertial measurement unit, IMU) verbindet, erreicht werden. Ein stereo visueller-inertialer Odometrie-Algorithmus kann dann die Daten beider Sensoren zur Bewegungsschätzung fusionieren.

In dieser Arbeit wird vor dem Hintergrund des Anwendungsfalls auf einem UAV ein solcher Algorithmus entwickelt. Hierzu werden zunächst Eigenschaften von Kameras beschrieben, die zur Erklärung benötigt werden. Der hier vorgestellte Ansatz verwendet FAST-Merkmale in Kombination mit dem Rotated-BRIEF Deskriptor und ein Verfahren zum Merkmaltracking, um Bildkorrespondenzen zu bilden. Auf diesen aufbauend wird dann beschrieben, wie mittels der Lösung eines nichtlinearen Optimierungsproblems eine Bewegungsschätzung berechnet werden kann. Im Anschluss hieran werden die Eigenschaften einer IMU und die enge Integration in das nichtlineare Optimierungsproblem beschrieben. Die in C++ erfolgte Implementierung des beschriebenen Algorithmus wird dann bezüglich seiner Präzision in der Bewegungsschätzung mit und ohne Verwendung einer IMU evaluiert. Abschließend wird die Arbeit zusammengefasst und ein Ausblick auf Weiterentwicklungsmöglichkeiten gegeben.

Abstract

Self localization and motion estimation is a major part of an unmanned aerial vehicle (UAV). Through this capability such a system achieves a major additional benefit, since it enables the UAV to operate autonomously. For the highest degree of autonomy in any environment such a system should also not depend on external reference systems like satellite navigation systems. This can be achieved by a sensor system on board of the UAV, that consists of a stereo camera and an inertial measurement unit (IMU). A stereo visual-inertial odometry algorithm can then fuse the data of both sensors for motion estimation.

In this work a stereo visual-inertial odometry algorithm is developed in context of application with UAVs. In order to accomplish this, the properties of cameras, needed by a stereo visual odometry algorithms, are described. The algorithm presented here uses FAST features in combination with the Rotated-BRIEF descriptor and an approach for feature tracking in order to obtain image correspondences. Based on these it is described how it is possible to obtain a motion estimation by solving a nonlinear motion estimation problem. Subsequently, the properties of an IMU and the tight integration into an optimization problem are described. The C++ implementation of the described algorithm is then evaluated regarding its precision in motion estimation with and without the use of an IMU. Finally, the work is summarized and an outlook on possibilities for improvements is given.

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Die Vereinbarung der Arbeitsgruppe für Studien- und Abschlussarbeiten habe ich gelesen und anerkannt, insbesondere die Regelung des Nutzungsrechts.

Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden. ja ☐ nein ☐

Der Veröffentlichung dieser Arbeit im Internet stimme ich zu. ja ☐ nein ☐

Koblenz, den 9. April 2017

Contents

Chapter 1

Introduction

Visual odometry is a technology which is used to estimate the motion of a mobile system that is equipped with one or more cameras. It analyzes the image streams of the cameras that depict the changing environment due to motion. During the analysis of the streams full 6D poses of the camera system between points in time are estimated. By concatenating these relative poses visual odometry computes a complete trajectory starting at an initial pose.

Visual odometry with two parallel cameras looking in the same direction is called stereo visual odometry. Compared to the monocular variant, stereo visual odometry has the advantage that a metric scale can be recovered. This kind of motion estimation can be combined with other motion estimation technologies in order to achieve a higher accuracy. Inertial measurement units (IMU) combine an accelerometer and a rate-gyroscope that measure linear acceleration and angular velocity. The combination of a stereo camera, an IMU and an algorithm that processes the data of both sensors is denoted as a stereo visual-inertial odometry system. The goal of this combination is to increase the accuracy of the system, estimating motion more accurately than any of the sensors on its own could. The main task of this system is to compute its motion over time. Many approaches additionally estimate a 3D model of the environment in form of a point cloud. This is due to the fact that it is needed in most approaches to derive the motion. A benefit of this 3D model is that it can be post processed in order to use it as a map.

Application areas for this systems can be found in air, ground and under water robotics. For the purpose of being fully autonomous, robots should not rely exclusively on global navigation satellite systems (GNSS) like Galileo, GPS or GLONASS. The reason for that is that robots are often employed in environments where GNSS is unavailable, such as in indoor environments. Such an environment is shown in Figure 1.1, where a quadrotor navigates autonomously through a corridor. It uses a stereo camera mounted on its top for navigation.

Figure 1.1: Quadrotor UAV with Stereo Camera Flying Indoor. This image shows a quadrotor UAV with a stereo camera mounted on its top for indoor navigation. Since the robot operates inside of a building, no GNSS signals can be received. Instead, the UAV navigates using the stereo camera.

The tasks of robots often require a precise knowledge about their position and orientation, enabling them to complete their tasks safely and quickly. This skill is essential, especially for unmanned aerial vehicles (UAV), since they are very mobile and often fly long distances. These systems also have limited resources in terms of payload and energy capacity. Therefore, it is important to develop light navigation systems with low power consumption. Widely used navigation systems in ground robotics that are based on 3D laser range finders are often heavier as well as bigger than visual-inertial systems, decreasing the battery-constrained flight time when used on UAVs. Additionally, UAVs are often equipped with IMUs as well as cameras. That makes visual odometry in combination with those sensors even more interesting for navigation.

During the computation of the relative poses small errors are made, which accumulate over time and cause the estimated pose to differ more and more from the true pose as time proceeds. This effect is called drift. Visual odometry algorithms have a strong similarity to visual simultaneous localization and mapping (V-SLAM) algorithms. Both types of algorithms accumulate relative poses for motion estimation. The difference is that SLAM algorithms try to minimize the

drift by potentially taking all the data into account that was previously captured. This means that a drift can be corrected by a SLAM algorithm at a later point in time by closing loops. Stereo visual-inertial odometry algorithms do not correct these accumulated errors afterwards. Only a few stereo visual-inertial odometry algorithms compute motion by taking images from more than two different points in time into account. The amount of images is then often limited to a small number like five images. The advantages of this are mainly a less complex problem that has to be solved as well as lower computational costs.

1.1 State of the Art

In this section the state of the art in solving the different aspects of visual odometry is described. First an overview of motion estimation methods for UAVs is given. Then, the state of the art in landmark detection and processing is described, followed by the latest work on image based motion estimation algorithms. Finally, methods for integrating an inertial measurement unit into the motion estimation algorithms are introduced. Since visual odometry and V-SLAM algorithms are very similar and solutions can be transferred between them, also V-SLAM approaches are partially presented.

Most UAVs fuse the data of many different sensors into a single pose to do motion estimation. This pose is more accurate than a pose, which can be computed from a single sensor. Very commonly used sensors are IMUs which are usually a combination of a gyroscope, an accelerometer, and magnetometer, which are able to precisely measure the attitude of an UAV at a high. However, position estimation only based on the data of an IMU is inaccurate [Woo07]. Because of that, their measurements are often fused with the data of other sensors for motion estimation.

2D laser scanners are often used for localization and mapping with UAVs [KNS+14, BKZ+15]. They provide a 2D profile of the environment, but since this profile depends strongly on the pose of the UAV, their navigation and mapping capabilities in 3D are limited. To avoid this problem, they are sometimes actuated or two laser scanners at the same time are used [BKZ+15]. This makes such a system expensive, heavy and large which is not advantageous for UAVs and impossible to carry for micro aerial vehicles (MAVs).

Other common sensor combinations are a sensor that measures the distance from the UAV to the ground, in combination with a ground facing camera. The distance sensors can be a sonar [HMTP13] or a laser altimeter for example. Knowing the altitude and the attitude of an UAV motion can then be computed with a metric scale by using an optical flow algorithm in combination with a motion

model. One advantage of those sensors is that the motion computation can be done at a high frequency on an embedded device [FJQBMT13].

Motion estimation and localization based on cameras is very common for UAVs [FPS14, FCC15, PSLDLP+16, KSS08, AD15]. In addition to the optical flow approach, several other algorithmic approaches exist to estimate the motion with single and multiple cameras. The basic schema of these algorithms is to find corresponding image features over subsequent images. From these features the three dimensional structure of the environment as well as the motion of a moving camera can be computed. An overview of these structure from motion methods can be found in Hartley and Zisserman's standard work Multiple View Geometry [HZ03]. The state of the art in feature detection, feature matching, motion estimation as well as integrating an IMU into these algorithms is given in the following three paragraphs.

Landmark Detection Most visual odometry algorithms use salient image features, which can be redetected in subsequent images with a high probability. In many cases key points or interest points, which can be found by corner detectors, are used as image features [GZS11, CP15, MAMT15, MAT16]. The *Harris corner detector* [HS88] is an older basic corner detector which is still in use by recent visual odometry algorithms [ZKS14, NRB+14]. It examines the structure of the auto-correlation matrix of an image point. This structure can be analyzed to make a statement about how good a particular point can be distinguished from others. The good *features to track* corner detector by Shi and Tomasi [ST94] is similar to the Harris corner detector. Both work on the auto-correlation matrix, but differ in how they interpret it.

Rosten and Drummond developed the *Features from Accelerated Segment Test* or also called *FAST* corner detector [RD06, RPD10] which has been used very often in recent visual odometry approaches [ABH+09, FCC15, NRB+14, AD15, FPS14, SCG13, YHGD14, PPFM15]. This detector makes use of a previously learned decision tree, which speeds up the computation of corners. Its short computational time makes it attractive for autonomous robotics with limited computational power and real time requirements. The ORB corner detector makes use of a modified FAST corner detector. This corner detector computes FAST features in a scale space and adds an orientation to the key points. The ORB corner detector is particularly used by the V-SLAM algorithms ORB-SLAM [MAMT15] and ORB-SLAM2 [MAT16].

Geiger et. al. [GZS11] presented a filter based corner detector and a filter based blob detector. The maxima and minima of the filter responses are used as key points. This approach has been also recently used in [CP15].

In order to achieve equally distributed key points over the image, the bucketing technique [ZDFL95] is used by many visual odometry algorithms [KGL10, CP15, FCC15]. The equal distribution improves the results of the pose computation which is based on visual landmarks. Bucketing divides an image into equally sized regions. In each region only the n most distinctive image features are kept for further processing.

In some artificial environments, such as a white corridor, no texture exist on which key points can be detected. Some alternative approaches then try to use lines as features. They can be combined with key points [GOBGJ16] or used without additional features for visual odometry [HFB16, WW13].

Marker based approaches use artificial markers as landmarks for motion estimation [PSLDLP+16, ESWS11]. The advantage of such algorithms is that the landmarks can be detected very reliably. Furthermore an identification number can be extracted from some markers. This makes it easy to find the same marker in different images, if multiple markers are used [GJSMCMJ14, WO16]. The disadvantage of this approach is that the environment has to be equipped with markers beforehand.

Matching One approach to find corresponding key points is called matching. This technique uses two or more sets of key points from different images and attempts to find corresponding points, avoiding wrong matches. Recent investigations focused on the matching stage in order to achieve a trajectory reconstruction with a low drift and high accuracy [CP15, BW16b]. Approaches which find matches between key points often use an descriptor. A descriptor is an n-dimensional vector of numbers that describes this point. In order to compute a descriptor, an algorithm processes a region around a key point. Key point descriptors which are often used in visual odometry or V-SLAM are BRIEF [CLSF10] or ORB [RRKB11]. Especially ORB became popular recently due to its application in ORB-SLAM [MAMT15] and ORB-SLAM2 [MAT16]. Additionally, the popular key point descriptors SIFT [Low99] and SURF [BTVG06] that also provide a blob detector were mentioned in [SF11] to be used for visual odometry. However, in recent visual odometry approaches they are not used anymore. In order to find corresponding key points, the distance between feature descriptors is compared. The actual correspondences are then found by exploiting different matching strategies. The circular matching strategy assumes a feature only to be matched correctly, if it was matched in a closed circle over stereo image pairs, captured at subsequent points in time [CP15]. Furthermore, the epipolar geometry of a stereo camera, which is described for example in [HZ03], is a popular standard approach to shrink the search space with constraints [KSS08, UESC16, CP15, SF11].

Also similarity measures like the normalized cross correlation [KSS08, CP15, NNB04] and the sum of absolute differences [UESC16, CP15, NNB04] are used by visual odometry and V-SLAM approaches. Cvišić and Petrović [CP15] make use of both measures in their matching strategy.

Instead of detecting key points in two images and matching them, key points can also be detected in one image and then be searched in a consecutive frame. This assumes that the motion between two consecutive frames is small [Sze10]. A popular algorithm for this is the KLT-tracker [LK81, TK91, ST94]. This algorithm is used in [ZKS14, YHGD14, FCC15] for example.

Direct methods utilize image patches for correspondence creation [LDIG11, NLD11, SMR07, FPS14, ESC14, EKC16]. The quality of the patch alignment is measured by the photometric error which compares the intensity values of two patches. During an optimization process the photometric error has to be computed many times, often including warping and integrating the image patches. This is computationally expensive but the computation costs for detecting key points and descriptors can be saved [FPS14]. Furthermore, the patches can be aligned with high accuracy at a sub-pixel level [IA99].

Motion Estimation from Visual Features The 5-point algorithm is used to compute the essential matrix from five 2D to 2D point correspondences. This matrix describes the geometric relation between two cameras, and allows the extraction of the rotation matrix and the direction vector of the relative pose between two cameras [HZ03]. This technique has been recently used in [CP15] to compute the rotation matrix from a relative pose.

More often approaches are used which minimize a nonlinear optimization problem to compute a relative pose [NRB+14, ZKS14, FPS14, FCDS15b, CP15, UESC16]. These approaches make use of a mathematical camera model that describes how an image is generated in a camera. The values that are generated by the camera model are then compared with the observed values of the real image. The difference is then expressed as a cost value. By adapting the relative pose and other parameters of the camera model the cost value is minimized during an optimization process. The so-called reprojection error measures the distance between a 2D point, which was projected to the image in the mathematical model, and the observation from the real image [HZ03]. Approaches based on the reprojection error usually only adapt the pose of the cameras and the reconstructed 3D points in a mathematical model [WW13, FPS14, CP15]. In addition to the reprojection error, cost terms can be added that do not require the reconstruction of 3D points [ZKS14, FCDS15a]. Direct approaches optimize the absolute difference between pixel values from the real image and the image which is generated from the camera model. This difference is called photometric error [UESC16].

For optimization the Levenberg-Marquardt algorithm, which is a combination of the Gauss-Newton Method and Gradient Descent can be used [SS02, ZKS14, UESC16]. An implementation of this method is provided by the Ceres-Solver library [AMO17]. Also, recent methods [EKC16, ZKS14, FCDS15a] increasingly make use of the incremental smoothing and mapping algorithm iSAM [KRD08] and iSAM2 [KJR+12]. Those algorithms are able to efficiently add data to an already existing optimization problem. This is achieved by only updating those parts of the optimization problem that are affected by the update [KJR+12].

IMU integration Recently, the integration of an IMU into visual odometry algorithms has become popular. This results from the gain of accuracy in motion estimation algorithms due to the complementary characteristics of cameras and IMUs. Additionally, both sensors are cheap and are already installed on many robots.

Visual-inertial odometry algorithms can be divided into loosely and tightly coupled approaches. Loosely coupled approaches compute motion for IMU and camera separately. After that both computed motion estimates are fused into one which is more precise. This is generally done by the use of Kalman Filters [KSS08, TGL+10, SSA13]. Those approaches have the advantage that visual odometry algorithms can easily be replaced by different implementations. However, by combining IMU data and visual measurements in this way, the pose computation by IMU and visual data can not profit from each other directly. This problem is tackled by tightly coupled approaches [NRB+14, AD15, FCDS15a, UESC16, EGH16, KESL17, CP15]. They can be subdivided into optimization and filtering based approaches [UESC16]. Optimization based approaches use an IMU motion model in an optimization problem to compute the most likely motion, together with visual measurements. Since the IMU delivers data at a high frequency, the optimization problem grows very fast, causing high computation times. This can be avoided by integrating the IMU measurements between two camera frames to a single motion estimation, which can then be used during optimization [SS02, FCDS15a, UESC16]. Another approach to avoid this problem is proposed in [NRB+14]. There the IMU measurements are marginalized if the number of measurements exceed a certain threshold. Tightly coupled filter based approaches make use of a Kalman Filter. In contrast to the loosely coupled filter based approaches, there the visual measurements and the measurements from the IMU are directly incorporated into the Kalman Filter [AD15, LM13].

1.2 Goals of the Thesis

This section describes the goals of the thesis at hand. It was written in the Vision 4 UAV group of the Technical University Madrid in cooperation with the Active Vision Group from the University Koblenz-Landau. Both groups work with robots, which can benefit from pose estimation by stereo visual-inertial odometry. The Vision 4 UAV group of Professor Pascual Campoy mainly works with quadrotor UAVs. Their pose estimation is currently based on laser scanners or optical markers, which works well. Nevertheless, a reliable stereo visual-inertial odometry system for the UAVs would provide new opportunities. Already mounted cameras and the on-board IMU could be used for navigation in addition to other tasks. In the best case a laser could be replaced by a stereo visual-inertial odometry system which would save weight and free capacities. Since markers would not be required anymore, the system would become more flexible and the preparation overhead would be reduced.

The main goal of the thesis is to develop a stereo visual-inertial algorithm that is integrated into the software framework *Aerostack*[1] of the Vision 4 UAV group. This framework is used to control UAVs with a high degree of autonomy. The new algorithm should provide a suitable alternative to the currently used approaches for localization in Aerostack. Furthermore, the programmed software should be modular, so that it can also be used with other robots, e. g. those of the Active Vision Group. Also an overview of the main ideas for stereo visual-inertial odometry should be given in this thesis. The implementation of the algorithm and the evaluation building up on this supports this idea since they give new insights.

1.3 Outline

The thesis at hand is structured into eight chapters. In the second Chapter 2 an explanation to the camera model will be given, that explains how 3D objects are projected to the sensor of a camera. Also, there will be an explanation for the geometric relations between two cameras. In Chapter 3 it will be described how key points can be extracted from an image and how image correspondences between these can be obtained. Building up on the two previous chapters, the motion estimation by stereo visual odometry is explained in Chapter 4. Afterwards, the properties of an IMU and the integration of this sensor into the visual odometry algorithm will be explained in Chapter 5. With the theoretical knowledge of the previous chapters the developed algorithm and its integration into the Aerostack framework is presented in Chapter 6. In Chapter 7 the presented algorithm is evaluated before in Chapter 8 the thesis will be summarized.

[1] Aerostack web page: http://www.aerostack.org

Chapter 2

Camera Geometry

The goal of stereo visual odometry is to compute the motion of a moving stereo camera over time. This is achieved by simultaneously reconstructing a 3D model that contains the pose of a camera at different points in time as well as the 3D geometry, which was recorded by the camera during that motion. In most cases the *pinhole camera model* is used that describes how a world point in 3D space is projected to a 2D pixel on the sensor of a perspective camera. The other way round it can also be used to reproject a pixel from an image to a ray in 3D space.

The *epipolar geometry* describes the projective geometry between two views [HZ03] and therefore provides the theoretical basis of reconstruction algorithms. It introduces the concept of the *essential matrix*, that can be computed from the relative pose between two cameras or at least five 2D to 2D point correspondences. Due to its versatile properties, it will be used in this thesis during the triangulation of 3D points, the search of key point correspondences, and the reconstruction of a relative pose between two cameras.

In order to triangulate 3D points from pixels, a second image from a perspective camera is needed, showing the same 3D geometry from a different view. During the triangulation, the intersection of the rays that are defined by the reprojection of the two pixels are searched to determine the position of a 3D point. However, for two cameras whose relative pose is only defined up to a scale factor, the metric position of the triangulated point can not be defined. This is usually the case for pure monocular visual odometry. The use of a stereo camera has the advantage that the metric pose between the left and the right camera can be derived during a calibration procedure. This enables to triangulate 3D points with the same metric scale.

In the following first Section 2.1 first the pinhole camera model is described. Building up on this the epipolar geometry is explained in Section 2.2 before in Section 2.3 the triangulation of 3D points will be introduced.

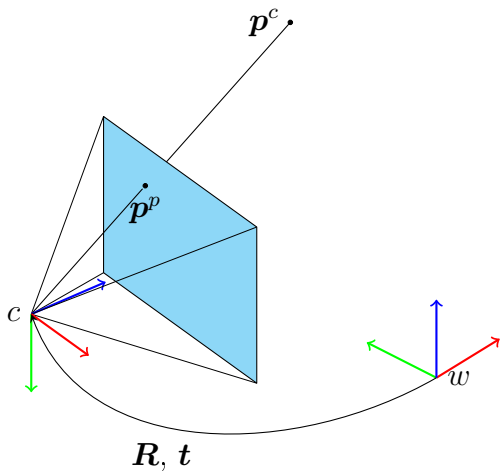


Figure 2.1: Illustration of the camera model. This figure shows the projection of a 3D point onto the image plane and the relation between camera and world coordinates is visualized. The origin of the camera coordinate system c is determined with respect to the world coordinate system w. This relation is determined by a rotation matrix $\boldsymbol{R}$ and a translation vector $\boldsymbol{t}$. A point $\boldsymbol{p}^c$ in camera coordinates can be projected to a point $\boldsymbol{p}^p$, which lies in the light blue image plane. Note that the image plane is placed in front of the camera.

2.1 Camera Model

In order to model a camera a fixed reference coordinate system w in which the pose of a camera and 3D objects can be defined is needed. The pose of the camera is measured with the camera coordinate system c with respect to w. It is a right handed coordinate system whose z-axis points into the direction of camera view and the x- , y-axis point right and down respectively. Its relation to w is described by a rotation matrix $\boldsymbol{R}$ and a translation vector $\boldsymbol{t}$ that are called the extrinsic camera parameters. In Figure 2.1 these relations are visualized. This figure shows also the different coordinate systems whose axes are colored in red for the x-axis, green for the y-axis and blue for the z-axis. This color scheme will be used consistently over all figures in this work.

Before a world point $\boldsymbol{p}^w$ can be projected onto the camera sensor it has to be transformed into a point

$$\boldsymbol{p}^c = \boldsymbol{R} \cdot \boldsymbol{p}^w + \boldsymbol{t} \tag{2.1}$$

in camera coordinates. The camera coordinate system is also the center of a central projection, which is called optical center. To project a point $\boldsymbol{p}^c = (x_c, y_c, z_c)^{\mathrm{T}}$ to

the sensor of a camera a perspective division is applied to $\boldsymbol{p}^c$ first. It projects the point $\boldsymbol{p}^c$ to a homogeneous point

$$\tilde{\boldsymbol{p}}^i = \begin{pmatrix} x_c/z_c \\ y_c/z_c \\ 1 \end{pmatrix} = \begin{pmatrix} x_c/z_c \\ y_c/z_c \\ z_c/z_c \end{pmatrix} \tag{2.2}$$

in the homogeneous image coordinate system i. During this projection the depth information of this point is lost, since the z_c value can not be recovered after the perspective division. The intrinsic camera parameters are defined by the focal lengths f_x, f_y and the center of projection whose coordinates are given by c_x and c_y. These parameters can be used to transform $\tilde{\boldsymbol{p}}^i$ into homogeneous pixel coordinates. This transformation is expressed with the calibration matrix $\boldsymbol{K}$ as

$$\tilde{\boldsymbol{p}}^p = \boldsymbol{K} \cdot \tilde{\boldsymbol{p}}^i = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \tilde{\boldsymbol{p}}^i. \tag{2.3}$$

The projection of a point $\boldsymbol{p}^c$ into pixel coordinates is also visualized in Figure 2.1. Here $\boldsymbol{p}^c$ is projected along the black ray directly to a point $\boldsymbol{p}^p$ in pixel coordinates.

Some lenses, especially ones with a wide field of view, cause a significant distortion of the captured image that can not be neglected. It can be modeled by a distortion function dist, which models the distortion with a second or higher order polynomial in image coordinates. The distortion function dist transforms $\tilde{\boldsymbol{p}}^i$ from homogeneous image coordinates into the distorted point

$$\tilde{\boldsymbol{p}}^d = \text{dist}(\tilde{\boldsymbol{p}}^i) \tag{2.4}$$

in the homogeneous distorted image coordinate system d. In Equation 2.3 $\tilde{\boldsymbol{p}}^d$ then replaces $\tilde{\boldsymbol{p}}^i$ and is transformed into pixel coordinates as follows:

$$\tilde{\boldsymbol{p}}^p = \boldsymbol{K} \cdot \tilde{\boldsymbol{p}}^d \ . \tag{2.5}$$

The coefficients of the polynomial in the distortion function are called the distortion coefficients. They are usually derived together with the parameters of the calibration matrix $\boldsymbol{K}$ during a calibration process of the camera [HZ03, SF11].

Since the geometric concepts and algorithms building up on the explained projection pipeline rely on undistorted coordinates, it is important to be able to reverse the distortion. This is similar to the distortion done in image coordinates. In a first step a distorted pixel $\tilde{\boldsymbol{p}}^p$ is transformed into $\tilde{\boldsymbol{p}}^d$ by making use of the inverse camera matrix $\boldsymbol{K}^{-1}$:

$$\tilde{\boldsymbol{p}}^d = \boldsymbol{K}^{-1}\tilde{\boldsymbol{p}}^p \ . \tag{2.6}$$

The inverse distortion function dist^{-1} can then be applied to $\tilde{\boldsymbol{p}}^d$ so that

$$\tilde{\boldsymbol{p}}^i = \text{dist}^{-1}(\tilde{\boldsymbol{p}}^d) \tag{2.7}$$

is computed.

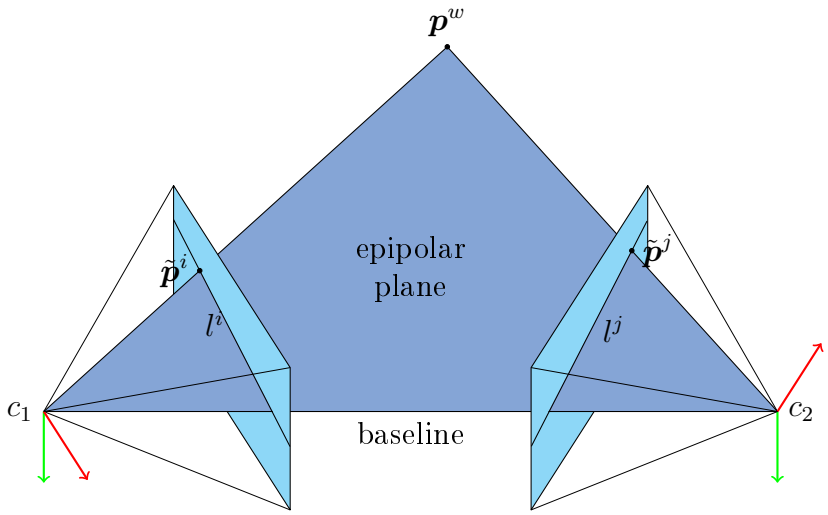


Figure 2.2: Illustration of the epipolar geometry. The figure shows a schematic visualization of the epipolar geometry. Two cameras with their coordinate systems c_1 and c_2 are visualized. By connecting the two origins of the camera coordinate systems and an additional 3D point $\boldsymbol{p}^w$, the epipolar plane is determined. The projection of $\boldsymbol{p}^w$ in image coordinates lies in a plane that is parallel to the image plane at $z_c = 1$. The two epipolar lines l^i and l^j result from the intersection of these planes with the epipolar plane.

2.2 Epipolar Geometry

The epipolar geometry describes the extrinsic and intrinsic relation between two projective cameras. This can be explained visually by intersecting planes, derived from two cameras and a 3D point $\boldsymbol{p}^w$ lying in front of these cameras. As it is shown in Figure 2.2, the origins of the camera coordinate systems c_1 and c_2 can be connected by a line called baseline. By connecting c_1 and c_2 with $\boldsymbol{p}^w$ two more lines can be generated. All three of this lines lie in one plane, which is called epipolar plane. Through intersecting the epipolar plane with another plane which is parallel to the image plane at $z_c = 1$, two more homogeneous lines l^i and l^j can be generated for each camera. These lines are part of the epipolar plane as well and are called *epipolar lines*. Via the epipolar plane the points $\tilde{\boldsymbol{p}}^i$ and $\tilde{\boldsymbol{p}}^j$ can be transferred onto their corresponding epipolar lines l^j and l^i in the opposite plane. This relation is expressed by the essential matrix $\boldsymbol{E}$. One way to derive $\boldsymbol{E}$ is to compute it from the relative pose between c_1 and c_2. This is done by using the rotation matrix $\boldsymbol{R}$ and the translation vector $\boldsymbol{t} = (t_x, t_y, t_z)^{\mathrm{T}}$, which transform a

3D point from c_1 to c_2 coordinates. By making use of the hat operator $\wedge$, $\boldsymbol{t}$ is mapped to a skew symmetric matrix

$$\boldsymbol{t}^{\wedge} = \begin{pmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{pmatrix} . \tag{2.8}$$

and $\boldsymbol{E}$ is then defined as:

$$\boldsymbol{E} = \boldsymbol{t}^{\wedge}\boldsymbol{R} . \tag{2.9}$$

The points $\tilde{\boldsymbol{p}}^i$ and $\tilde{\boldsymbol{p}}^j$ in homogeneous image coordinates, can then be transferred by

$$l^j = \boldsymbol{E}\tilde{\boldsymbol{p}}^i \tag{2.10}$$

$$l^i = \boldsymbol{E}^{\mathrm{T}}\tilde{\boldsymbol{p}}^j \tag{2.11}$$

to their corresponding epipolar lines. A 3D point along the ray determined by the center of the camera coordinate system c_1 and $\boldsymbol{p}^w$ is therefore projected to l^j as long as it is in the field of view of the camera belonging to c_2. The same holds true in reverse for a 3D point that is moved on the ray determined by $\tilde{\boldsymbol{p}}^j$ and $\boldsymbol{p}^w$. A point in the first image therefore creates a corresponding epipolar line in the second image.

Furthermore, it can be verified by the so called epipolar constraint

$$\begin{aligned} \tilde{\boldsymbol{p}}^{j\mathrm{T}}\boldsymbol{E}\tilde{\boldsymbol{p}}^i &= 0 \\ \tilde{\boldsymbol{p}}^{j\mathrm{T}}l^i &= 0 \end{aligned} \tag{2.12}$$

that two corresponding points lie exactly in the same epipolar plane. Given two image points, one in either image and each of them originating from the same object, the epipolar constraint predicts them to lie on the same epipolar plane. This constraint exploits that the dot product of a homogeneous line and homogeneous point is zero if the point lies on the line.

2.3 Sparse 3D Reconstruction from Two Views

During the projection of a 3D structure to an image as explained in Section 2.1 the depth information of this structure is lost. This information can be recovered by making use of two images captured from different perspectives. This requires the relative pose between the two cameras, whose coordinate systems are denoted with c_1 and c_2. To recover the 3D information for one 3D point two corresponding 2D points $\boldsymbol{p}^i$ and $\boldsymbol{p}^j$ belonging to c_1 and c_2 must be known. An algorithm that tries to reconstruct a 3D point puts a ray through the origin of c_1 and $\boldsymbol{p}^i$ for the first camera and a second ray through c_2 and $\boldsymbol{p}^j$ for the second camera. It

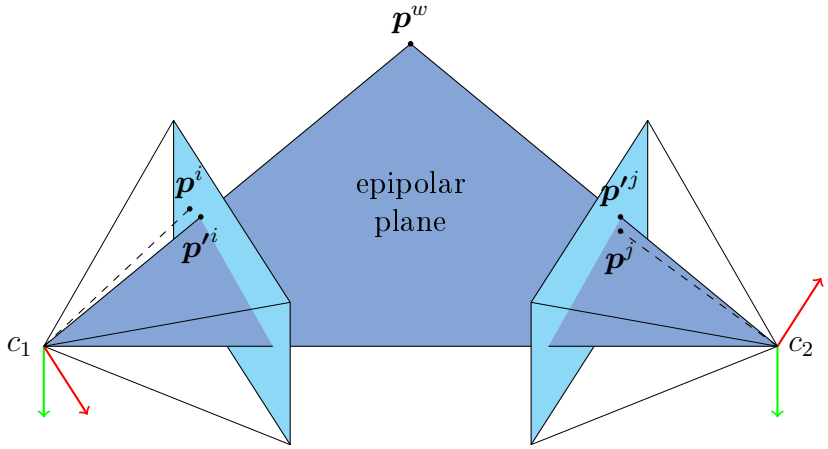


Figure 2.3: Illustration of a 3D point triangulation. This figure shows a pair of corresponding 2D points $\boldsymbol{p}^i$ and $\boldsymbol{p}^j$ seen by two cameras from different perspectives. Due to inaccuracies $\boldsymbol{p}^i$ and $\boldsymbol{p}^j$ do not satisfy the epipolar constraint and get corrected until they lie as corrected points $\boldsymbol{p}'^i$ and $\boldsymbol{p}'^j$ in the same epipolar plane. By laying a ray through the corrected points and their corresponding center of projection two rays are created. They intersect at the point at which the 3D point $\boldsymbol{p}^w$ is created.

then computes the intersection of these two rays. This is done by solving a linear system of equations, which is determined by the 2D point correspondences and the relative pose between the cameras.

In practice the exact intersection is never found due to uncertainties

- in the estimated relative pose of both cameras,
- the calibration of the cameras or the camera model or
- in the spatial localization of the key points [SF11].

Therefore the algorithm tries to estimate a point which lies between these two rays. However some of the reconstructed points are reconstructed due to these uncertainties with high inaccuracies and a negative z-value. The reason for this is that $\boldsymbol{p}^i$ and $\boldsymbol{p}^j$ do not lay exactly on the same epipolar plane. Hartley and Zisserman propose to use an optimal triangulation method under the assumption that the noise of the 2D points is normal distributed [HS97]. Like illustrated in Figure 2.3, this method searches for two corrected points $\boldsymbol{p}'^i$ and $\boldsymbol{p}'^j$ that minimize

$$\underset{\boldsymbol{p}'^i, \boldsymbol{p}'^j}{\operatorname{argmin}} \|\boldsymbol{p}'^i - \boldsymbol{p}^i\|^2 + \|\boldsymbol{p}'^j - \boldsymbol{p}^j\|^2 \text{ subject to } \tilde{\boldsymbol{p}}^{j\mathrm{T}} \boldsymbol{E} \tilde{\boldsymbol{p}}^i = 0. \tag{2.13}$$

These points $\boldsymbol{p}'^i$ and $\boldsymbol{p}'^j$ are then used for reconstructing the 3D point $\boldsymbol{p}^w$ by solving the linear system of equations.

(a) A reconstructed point cloud. **(b)** Image that was used for reconstruction.

Figure 2.4: Visualization of a point cloud reconstruction and an image, which was used for this reconstruction. In Figure (a) a point cloud, which was reconstructed from several stereo image pairs can be seen. In the front part of the point cloud the reconstruction of a marker pattern can be seen, next to a visualization of stereo camera coordinate systems. The marker pattern corresponds to the one in Figure (b). This precise image was used to reconstruct the point cloud in Figure (a).

In order to compute a point cloud the above described method is repeated many times for different point correspondences. The result of such a reconstruction for several stereo image pairs can be seen in Figure 2.4a. An image from which this point cloud was reconstructed is shown in Figure 2.4b.

The scale of the reconstructed points depends on the length or scale of vector $\boldsymbol{t}$ that describes the relative position between the cameras that was used for triangulation. In case of visual odometry with monocular cameras the absolute scale of a translation vector $\boldsymbol{t}$ between two cameras is often unknown. For a stereo camera instead the metric scale of $\boldsymbol{t}$ is known from a previous calibration procedure. Therefore 3D points can be triangulated with a metric scale.

Chapter 3

Feature Detection, Matching, and Tracking

The basis of visual odometry is the ability to track landmarks as image features from consecutive images. Due to the change of image features in overlapping images, which is caused by the motion of a camera, it is possible to reconstruct this motion. Therefore salient features in images have to be detected first. It is important to detect particular features that can be relocated in a consecutive image with a high probability. Without this ability a motion computation is not possible. One type of image features are key points, which can be for example corners in images. More concrete in this thesis *FAST* corners are used as key points, which were already mentioned in Section 1.1 as a state of the art corner detector. They are located at the intersection of two edges and can therefore be precisely localized.

Additionally, it has been found to be important for the precision of the motion estimation that key points are equally distributed over the image. This is typically not guaranteed by a feature detector itself. A bucketing algorithm [NNB04, KGL10] can be used to improve this.

In order to be able to re-detect key points in another image, different techniques exist. One is to find matches between two independently detected sets of key points from two different images. For matching the key points are described by an key point descriptor algorithm with a vector of features. These features describe the key point by the appearance of the image region around it. The *Rotated BRIEF* (Binary Robust Independent Elementary Features) [RRKB11] descriptor here used describes a key point by using a bit string whose bits represent the results of binary tests. To find corresponding key points between the left and the right image of the stereo camera a constrained matching strategy is used in order to decide which key points belong together. Another technique to find key point correspondences is to detect key points in one image and to track them by their appearance in a second

image. For this the image patch around the key point is compared with spatially close image patches in the second image. This approach assumes that key points do not move too far in subsequent images and is therefore well-suited for image streams, where the expected motion of a key point is small [Sze10].

In the first Section 3.1 of this chapter the FAST corner detector is described in detail, followed by the bucketing algorithm in Section 3.2. After that the Rotated BRIEF feature descriptor is explained in Section 3.3 that can be used in combination with the constrained matching strategy explained in Section 3.4. Finally key point tracking is explained in Section 3.5.

3.1 FAST Corner Detector

The FAST corner detector searches for corners in an image I. This is done in several consecutive steps including a preparatory machine learning phase. Without edge treatment, each pixel $\boldsymbol{p} \in I$ in pixel coordinates is examined to be a corner point. Around a pixel $\boldsymbol{p}_{\text{test}}$ that is tested to be a corner a Bresenham circle of radius three is defined. Figure 3.1 shows an image of a window in the left part of the figure. On the right a detail of a corner of this window is shown to a pixel grid enlarged. There the pixel $\boldsymbol{p}_{\text{test}}$ can be seen in the middle of the pixel grid. The Bresenham circle around $\boldsymbol{p}_{\text{test}}$ is visualized with pixels that have a bold white border. Each of these pixels is referenced by a number between 1 and 16. To determine whether $\boldsymbol{p}_{\text{test}}$ is a corner a high speed test is performed, that checks if $\boldsymbol{p}_{\text{test}}$ can . It compares the image value $\mathrm{I}(\boldsymbol{p}_{\text{test}})$ at position $\boldsymbol{p}_{\text{test}}$ with the values of the pixels on the Bresenham circle at position 1, 5, 9 and 13. For a threshold t the pixel $\boldsymbol{p}_{\text{test}}$ passes the high speed test, if at least three of the values at these positions are all smaller than $\mathrm{I}(\boldsymbol{p}_{\text{test}}) - t$ or all greater than $\mathrm{I}(\boldsymbol{p}_{\text{test}}) + t$. If the first test is passed by $\boldsymbol{p}_{\text{test}}$ a second test is executed, otherwise $\boldsymbol{p}_{\text{test}}$ is not assumed to be a corner. In this test a continued arc consisting out of $a = 12$ pixels have to be smaller than $\mathrm{I}(\boldsymbol{p}_{\text{test}}) - t$ or greater than $\mathrm{I}(\boldsymbol{p}_{\text{test}}) + t$. This test is visualized in Figure 3.1 on the pixel grid exemplary as a dotted line. When this test is also passed by $\boldsymbol{p}_{\text{test}}$, it is assumed to be a corner.

Although this algorithm could already be used as a corner detector the authors Rosten and Drummond [RD06] criticize four weaknesses of this approach:

1. The high-speed test does not generalize well for $a < 12$. A smaller a is needed because with $a = 12$ only corners with an acute angle are found.

2. The choice and ordering of the fast test pixels contains implicit assumptions about the distribution of feature appearance.

3. Knowledge from the first 4 tests is discarded. This could be used during the second test in order to save computational effort.

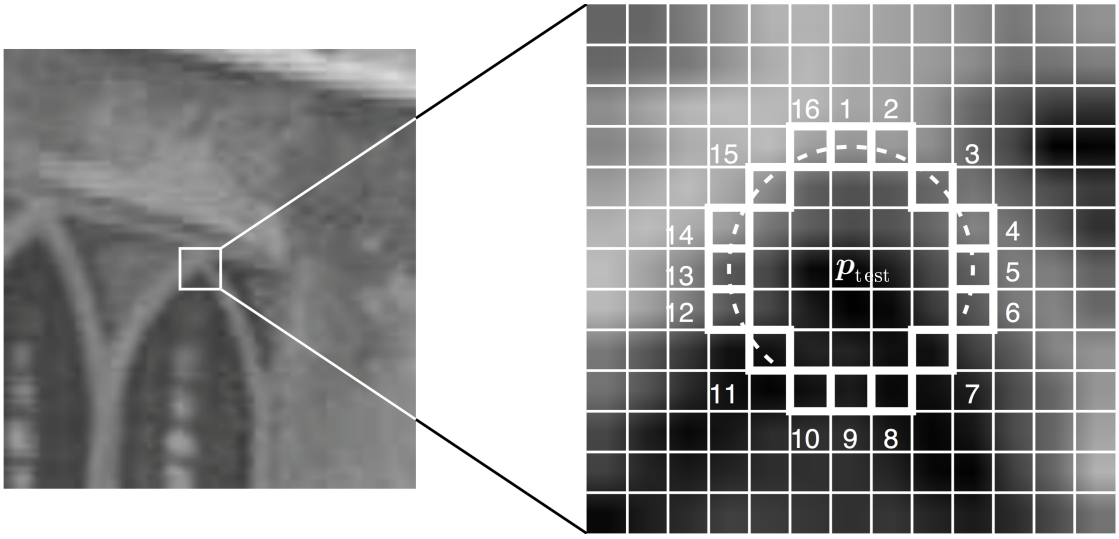


Figure 3.1: Illustration of the FAST corner detection. The image on the right shows a detail from the image on the left enlarged up to pixel level. First a pixel $\boldsymbol{p}_{\text{test}}$ is chosen to be tested and around it a Bresenham circle is defined, highlighted by pixels with a border. In a high speed test the pixel 1, 5, 9 and 13 are tested. If this test is passed, 12 pixel in a row, marked by the dotted line, have to fulfill the test conditions of a second test [RD06].

4. Multiple corners are detected adjacent to one another, but only one precise position of the corner is required. This could lead to mistakes during matching.

In order to tackle the first three problems, the authors propose an offline approach to train a decision tree. The decision tree is then used to decide whether or not a pixel is a corner.. For training a set of training images is needed. These images are then used only by the second test to detect corners. Based on the detected corners it can be determined which pixel on the Bresenham circle contain the most information gain to decide if $\boldsymbol{p}_{\text{test}}$ is a corner. This information can then be encoded into a decision tree that will be used to find corners in images. For practical application the resulting decision tree is converted into program code of the programming language C, containing nested if and else branches. Finally the code is compiled and optimized by a compiler. Using a precomputed and optimized decision tree makes the FAST corner detector computationally very efficient. However a lot of adjacent corners are detected by this algorithm. In order to tackle this problem a corner response function V is introduced, which is computed for every detected corner. This function uses the set of pixels C on the

Bresenham circle around the detected corner. The pixels $\boldsymbol{c} \in C$ are then divided into two more sets

$$\begin{aligned} P_B &= \{\boldsymbol{c} \in C | I(\boldsymbol{c}) \geq I(\boldsymbol{p}_{\text{test}}) + t\} \text{ and} \\ P_D &= \{\boldsymbol{c} \in C | I(\boldsymbol{c}) \leq I(\boldsymbol{p}_{\text{test}}) - t\} \end{aligned} \tag{3.1}$$

for the bright and the dark pixels on the Bresenham circle. The response function V is then defined as:

$$V(\boldsymbol{p}_{\text{test}}) = \max \left\{ \sum_{\boldsymbol{c} \in P_B} |I(\boldsymbol{c}) - I(\boldsymbol{p}_{\text{test}})| - t, \sum_{\boldsymbol{c} \in P_D} |I(\boldsymbol{p}_{\text{test}}) - I(\boldsymbol{c})| - t \right\}. \tag{3.2}$$

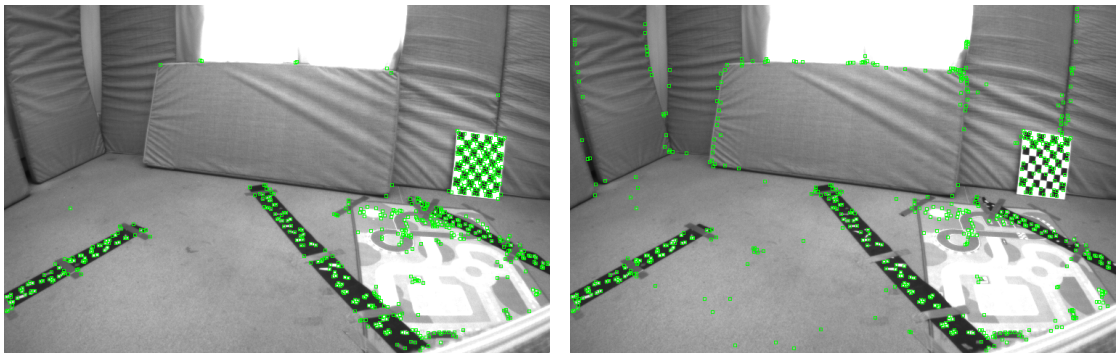
In order to obtain the final corners, a non maximum suppression is performed on the evaluations of V [RD06].

An example of extracted FAST corners for a threshold $t = 30$ and $a = 9$ can be seen in Figure 3.2a. The corners are marked with green squares in the image. It can be noticed that in some image regions more corners are detected than in others.

3.2 Feature Bucketing

The bucketing algorithm aims at achieving an equal distribution of key points over the image. Interest point detectors often use a threshold to determine which pixels will be a key point and which will not. Since some regions in images have higher contrast and others a lower one, different thresholds for different image regions would often be appropriate. However in practical application often only one heuristic threshold is chosen and applied for corner detection on all images during visual odometry. From this threshold it is not clear how many key points will be detected in an image if for example light conditions or the type of texture change. In the example in Figure 3.2a adjacent key points were extracted on the play carpet and the chessboard. In the upper left part of the same image hardly any key points were detected, even if there are possibilities to detect some.

The bucketing algorithm tries to solve this problem by subdividing the whole image into non overlapping squares, called buckets. Within each bucket, the algorithm keeps only the n key points with the highest corner values with respect to a corner response function. In this case the response function indicates a key point by returning high values. This allows to choose a lower threshold t that makes sure that in images with low contrast still key points can be detected. Additionally, the maximum number of key points and therefore the computational effort of subsequent algorithms is limited. Furthermore a more uniform distribution of key points over the image is achieved. A result of the bucketing algorithm based on



(a) FAST threshold 30 without bucketing **(b)** FAST threshold 15 with bucketing

Figure 3.2: Visualization of detected key points with and without bucketing. In (a) 604 key points where extracted by the FAST algorithm with $t = 35$ and $a = 9$. In this image a lot of key points are gathered at the chessboard pattern and at the play carpet, where strong image values are present. In (b) the initial key points where extracted with $t = 15$ and $a = 9$. The bucketing algorithm was applied afterwards which finally lead to 541 more uniformly distributed key points.

FAST features can be seen in Figure 3.2b. To achieve a more uniform distribution of key points the FAST threshold was set to $t = 15$. After that the bucketing algorithm with 10 horizontal and 10 vertical buckets was applied. A maximum of 10 key points was allowed in each bucket. Comparing Figure 3.2a in which 604 key points were detected with Figure 3.2b with 541 key points it can be noted that the distribution is much more uniform. Many adjacent key points on the chessboard pattern or the play carpet have been removed. However also new key points in the upper left region of the image have been added due to the lower detection threshold. A disadvantage of this is that also a lot of key points where detected on edges. Their positions can not be as precisely located as those of corners, which could lead to a more in precise motion estimation.

3.3 Rotated BRIEF Feature Descriptor

Rotated BRIEF is a binary feature descriptor, which describes key points by using a bit pattern. It is part of ORB (oriented FAST and Rotated BRIEF), a combination of an extended FAST key point detector and a rotation invariant BRIEF [CLSF10] descriptor. In this thesis only Rotated BRIEF is explained since it is used in combination with plain FAST corners in the developed visual odometry algorithm. In the following, the BRIEF descriptor is described first. The following

part will explain Rotated BRIEF, which adds rotation invariance and a learning stage to BRIEF.

BRIEF describes a key point $\boldsymbol{k}$ by comparing intensity values of smoothed image patches. Let the function g return the value of the center pixel from a 9×9 patch that was smoothed by a Gaussian filter. Then a binary test

$$\tau(\mathrm{g}; \boldsymbol{p}, \boldsymbol{q}) = \begin{cases} 1 : \mathrm{g}(\boldsymbol{p}) < \mathrm{g}(\boldsymbol{q}), \\ 0 : \mathrm{g}(\boldsymbol{p}) \geq \mathrm{g}(\boldsymbol{q}) \end{cases}, \tag{3.3}$$

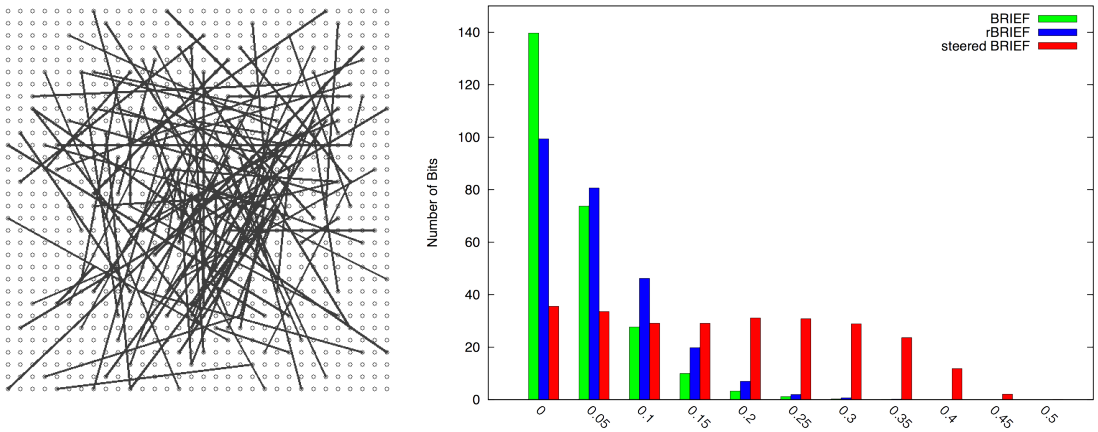
can be defined, where $\boldsymbol{p}$ and $\boldsymbol{q}$ are pixel locations and $\mathrm{g}(\boldsymbol{p})$ returns the smoothed patch value for pixel $\boldsymbol{p}$. The function τ can then be applied to $n = 256$ different pixel location pairs $(\boldsymbol{p}, \boldsymbol{q})$ in a larger patch of size 31×31 around $\boldsymbol{k}$. The results of the function are then stored as a bit string:

$$b(\boldsymbol{k}) = \sum_{1 \leq i \leq n} 2^{i-1} \tau(\mathrm{g}; \boldsymbol{p}_i, \boldsymbol{q}_i), \tag{3.4}$$

which serves as the BRIEF descriptor of a key point $\boldsymbol{k}$. Comparison of these binary descriptors can be carried out efficiently by computing their Hamming distance.

The authors of BRIEF suggested several possibilities to choose the pixel position pairs. The one used during the development of Rotated BRIEF, is generated by choosing two normally distributed pixels inside of the large patch. An example of the point pairs that are produced by this procedure are visualized in Figure 3.3a. Once the patch pairs are generated they are used for every computed descriptor to make them comparable. If a key point is redetected that is rotated, also the image region around the key point is rotated. As a result, the descriptor changes strongly and is not very similar anymore to the initial descriptor. This happens since the image patch of the descriptor is not rotated with the key point. A solution that avoids this problem rotates the patch pattern by the orientation provided by the previously detected key point. Since a computation of the exact rotation for every key point is computationally expensive a look up table of patch patterns is generated. It contains 30 patch patterns that discretize the orientation of the key point in steps of 12 degrees. The authors denote this descriptor as Steered BRIEF.

Even if Steered BRIEF is rotationally invariant some good properties of BRIEF are lost. One of these properties is that the variance of single features has decreased. Patch pairs with a mean close to 0.5 have a high variance and return different values for different inputs when they are used with τ. Figure 3.3b shows a histogram over the number of features and their distances from 0.5 for the different BRIEF descriptors. BRIEF has many more feature patches that are close to 0.5 than Steered BRIEF. In order to restore this property a machine learning



(a) Patch connections [CLSF10] **(b)** Histogram of bit mean values [RRKB11]

Figure 3.3: Visualization of patch pairs for BRIEF and a histogram of bit mean values. Figure (a) shows 128 pixel location pairs. The points in the image visualize pixels of the large image patch around a key point. From the inside of this patch, pairs of pixels are visualized by lines. By making use of the location pairs and a binary function a key point descriptor is then computed. The histogram in Figure (b) shows the distribution of patch pairs by their distance to a mean of 0.5. It is obvious that Rotated BRIEF has much more patch pairs close to a mean of 0.5 than Steered BRIEF. This indicates that the features react differently to different inputs, which is needed for a good descriptor.

approach is employed to select the most descriptive image patch pairs. This is achieved by generating a set of training patches, containing all possible pairs of non overlapping patches. With those patch pairs τ is executed on a large number of key points. Then the mean and the variance of the results for each patch pair are examined. Also uncorrelated patch pairs for the descriptor are desired. The machine learning algorithm therefore generates a feature descriptor that contains patch pairs with a mean close to 0.5 and that contains patches that are uncorrelated.

The resulting descriptor is called Rotated BRIEF. Figure 3.3b shows that the number of features with a mean close to 0.5 has increased a lot in comparison to steered Brief. However, the number of patch pairs close to 0.5 from Rotated BRIEF is still lower than compared to BRIEF. This results from the fact that if the key point is rotated, while the BRIEF descriptor not, totally different image regions are compared by the features [RRKB11].

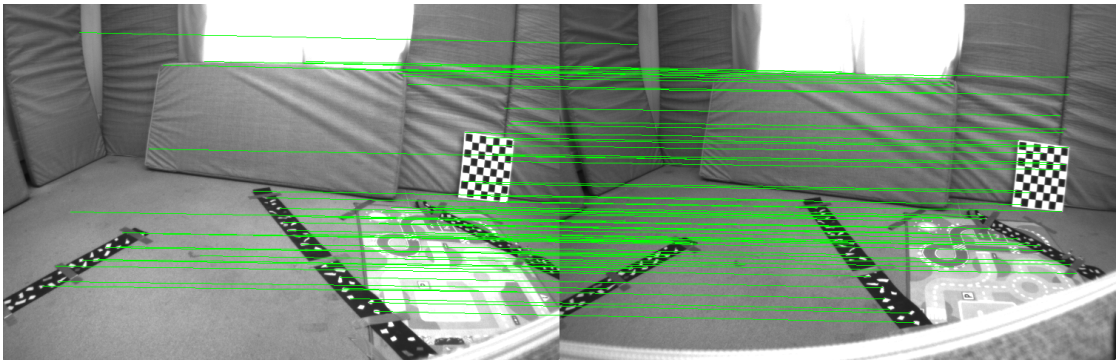


Figure 3.4: Visualization of feature matches. In the visualization 71 feature matches are visualized as green lines. At the ends of each line a key point was detected.

3.4 Constrained Feature Matching

Feature matching tries to find corresponding features between two sets of independently computed feature descriptors. In order to find the best matching descriptors, the distances between pairs of descriptors from the two sets are compared. In the case that for one descriptor in the first set several other descriptors with a similar distance in the second set exists it is not sure which correspondence is correct. This is especially a problem in images with repetitive image structures that cause similar feature descriptors. A matching algorithm then has the choice to either reject similar matches, causing a decrease in the number of correspondences, or to accept them, increasing the number of incorrect correspondences. In case of a stereo camera the relative pose between both cameras is precisely known from a prior extrinsic calibration. This enables computation of the essential matrix $\boldsymbol{E}$ to reduce the number of possible candidates for correspondences by making use of the epipolar constraint. For a key point $\tilde{\boldsymbol{p}}^i$ in I the corresponding epipolar line l^j is computed as shown in Equation 2.10. During the correspondence search only key points that lie on this epipolar line are considered to be a correct match. Due to inaccuracies in the intrinsic and extrinsic calibration of the camera model, the camera model itself and the key point location, the practical implementation allows points up to a certain distance as correct matches. This procedure reduces the number of possible matches and therefore eliminates unfortunately possible ambiguities, which results in more correct matches. However it is still possible that wrong correspondences are made. These are called outlier matches whereas the correct matches are called inlier matches.

Figure 3.4 shows a result of the constrained matching procedure. In total 71 matches where found between the left and the right image pair of a stereo camera. Each match is visualized as a green line between the key points.

3.5 Feature Tracking

Another technique for finding features over subsequent images is to detect them in a first image and to track them in a consequent image by using their appearance. This assumes a small spatial motion of a tracked point in pixel coordinates. Consequently also the appearance of a region around the tracked point is assumed to be similar.

For finding a point from a first image I in a second image J again, a quadratic image patch around the tracked point in I is defined. The values inside the image patch are then compared with regions in J. This assumes that the displacement of the tracked point in the image is relative small and thus the appearance is expected to be similar too. For comparing two image patches, distance measures like the sum of squared differences can be used [Sze10]. The error between the image patches is then used as a cost function during an optimization step. During the optimization the corresponding image patch with the highest similarity to the original image patch is identified. This kind of tracking algorithm is broadly called the Kanade, Lucas, Tomasi tracker (KLT tracker). It was first developed in [LK81] and then improved in [TK91, ST94]. Based on the original concept a pyramidal version of the KLT tracker has been developed. It creates a image pyramid for I containing I and smaller versions of it. The same procedure is executed for J. Starting at the smallest scale in the image pyramid the KLT-tracker searches the point correspondence. This leads to a coarse but easier tracking, since fewer pixel have to be examined and the spatial distance between the corresponding points is smaller. The result of this execution of the algorithm is then used as initialization for the next larger pyramid scale, where the image correspondences are refined. This execution should compute a more precise location of the corresponding point because of the higher resolution of the image. The process is iterated until the pyramid layer of the original image is reached. The presented coarse to fine tracking can speed up the tracking process and makes the algorithm work better with larger motions of tracked points [Sze10, Bou01].

Chapter 4

Motion Estimation for Stereo Visual Odometry

The motion estimation part of a stereo visual odometry algorithm processes image correspondences of consecutive stereo image pairs in order to derive motion of the sensor system over time. This is done by estimating the relative transformation between the position and attitude of the camera at the points in time at which stereo image pairs were captured. The trajectory of the stereo camera can be derived by iteratively concatenating these poses.

A problem during this process is that the image correspondences can contain outliers. They potentially cause great errors during motion estimation and increase the drift of the estimated trajectory. In order to estimate motion in the presence of outliers the motion estimation algorithm has to be robust against them. This means that outliers have to be identified in order to give them no influence to the motion estimation.

Even inlier correspondences can be faulty due to possibly inaccurate localization of detected image correspondences. Therefore, the most likely relative motion of the camera given the inlier correspondences should be derived. The information from the inliers has to be fused in a way that the most likely motion is computed. In this work the motion estimation is done by solving a nonlinear optimization problem, where all known information from image correspondences can be incorporated at the same time. An additional advantage of this approach is that data about motion from the IMU can be integrated, which will be explained in Chapter 5.

The remainder of this chapter is structured as follows. First an overview about nonlinear optimization in relation to visual odometry is given in Section 4.1. After that more formal and detailed introduction to the motion estimation problem is given in Section 4.2. The two following sections build on the formulation introduced in that section. Section 4.3 then explains a possible method to compute

an initial guess for the motion estimation problem. Finally in Section 4.4 it is explained how a nonlinear optimization problem can be constructed for motion estimation.

4.1 Nonlinear Optimization

During the previous sections measurements in form of key point correspondences have been derived from images. The measurements are faulty and a mathematical motion model has to be used that incorporates these errors. Due to the errors in the measurements it is not clear how the correct state of the model looks like but the most likely state can be estimated. This can be achieved by constructing a least squares problem that quantifies how good a model state fits to measurements using a sum of squared differences . A solution to this problem can be computed by a nonlinear optimization algorithm that tries to find the state that minimizes the sum of squared differences.

The computation of the single summands of the sum is done by so called cost functions. They predict measurements for a state $\boldsymbol{x}$ of a mathematical model. The cost function compares the predicted measurements with the real measurements and then computes the difference between them in form of a residual vector $r(\boldsymbol{x})$. Since different types of cost functions can be used they also often quantify the calculated residuals in different units and with different confidences. Therefore they have to be standardized in order to be comparable. The standardization is done by making use of a covariance matrix $\boldsymbol{\Sigma}$, which specifies the inaccuracy of a measurement as a normal distribution [TMHF99]. With this information the objective function that has to be minimized is defined as the sum over the squared standardized residuals:

$$\underset{\boldsymbol{x}}{\operatorname{argmin}} \frac{1}{2} \sum_{i=1}^{n} \|r_i(\boldsymbol{x})\|^2_{\boldsymbol{\Sigma}_i}, \tag{4.1}$$

where n is the number of cost functions in the problem, i is their index, and $\|\cdot\|^2_{\boldsymbol{\Sigma}}$ denotes the squared Mahalanobis distance.

In some cases the observed measurements are outliers. For a parameter estimation that is close to the correct solution, these outlier measurements cause a significant larger residual error than it is done by inliers. This error value of outliers grows quadratically. This can cause that an optimization algorithm is not able to find a correct optimal solution since it worth more to minimize the fast reducing errors of outliers than the already small ones of inliers. One strategy to avoid this is the use of loss functions. Loss functions reduce the influence of large

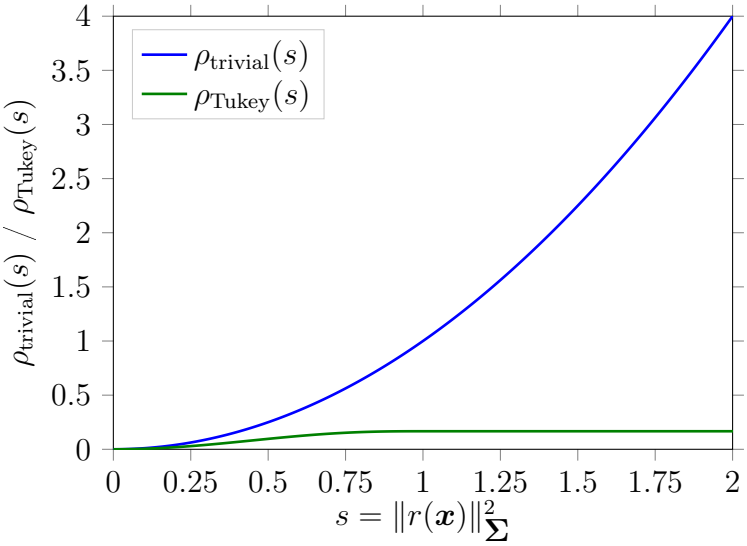


Figure 4.1: Plots of the trivial and the Tukey loss function. The x-axis shows the length of the residual vectors $s = \|r(\boldsymbol{x})\|^2_{\boldsymbol{\Sigma}}$. The corresponding costs after applying the cost function are shown on the y-axis.

errors and make the optimization to a certain extent robust against outliers. To them the non negative scalar

$$s = \|r(\boldsymbol{x})\|^2_{\boldsymbol{\Sigma}}\,, \tag{4.2}$$

is passed as an argument. In this work the trivial and the Tukey loss function will be used. Plots of these loss functions can be seen in Figure 4.1.

The trivial loss function:

$$\rho_{\text{trivial}}(s) = s, \tag{4.3}$$

returns simply its parameter s and does not add any robustness to the optimization problem. Here the opposite is the case, due to the squaring in Equation 4.2 larger errors grow faster than small ones. The use of this loss function makes sense when there are no outliers in the measurements.

The Tukey loss function is defined as:

$$\rho_{\text{Tukey}}(s) = \begin{cases} \frac{a^2}{6}\left(1 - \left(1 - \frac{s}{a^2}\right)^3\right) & s \leq a^2 \\ \frac{a^2}{6} & s > a^2 \end{cases}, \tag{4.4}$$

where a is a scale factor, that is chosen in dependence of the current application of the loss function [AMO17]. It computes values that are smaller than s in general and which do not increase for $s > a^2$. Which measurements are classified as outliers is therefore determined by the value of s. Since outliers will result

in constant error values the optimization algorithm can compute state updates that decrease residual errors for inliers without that the residual errors for outliers increases. In case of the Tukey loss function this avoids that outliers that cause a s that is greater than a^2 have influence on the solution of the optimization result.

Finally the nonlinear optimization problem where every standardized residual vector belongs to a loss function ρ_i can be defined:

$$\underset{\boldsymbol{x}}{\operatorname{argmin}} \frac{1}{2} \sum_{i=1}^{n} \rho_i(\|r_i(\boldsymbol{x})\|^2_{\boldsymbol{\Sigma}_i}). \tag{4.5}$$

It is used to estimate a state that minimizes the sum over the errors. This state will be the final solution of the optimization problem.

The optimal state can be computed by an optimization algorithm of which some where already mentioned in Section 1.1. In this work the Levenberg-Marquardt algorithm is used which interpolates between gradient descent and the Gauss-Newton method during optimization. In order to solve the optimization problem it needs an initialization of this problem first that is close to the final solution. If the initialization is bad it can happen that the algorithm does not converge to a global minimum due to the nonlinearity of the optimization problem. Starting at the initial state the algorithm adapts this one iteratively with state updates, in order to approach an optimal solution. The adaption is done several times until a maximum number of iterations has been reached. Additionally, a preliminary abortion is possible if the state vector changes only by a amount that is smaller than a predefined threshold. The small change indicates that a good solution of the state vector is found. After the algorithm has aborted the state from its last iteration is used as the solution of the optimization problem [TMHF99].

4.2 Problem Definition

In order to measure motion with a stereo camera a coordinate frame, with which the position of the stereo camera is represented, is needed. This will be without loss of generality the coordinate frame of the left camera c_l. The fixed relative pose $\boldsymbol{S}$ between the c_l and the coordinate frame c_r of the right camera is derived during a preceding calibration procedure. During this procedure, the intrinsic camera parameters of both cameras are derived as well. These information can be used to triangulate 3D points from image correspondences.

Stereo visual odometry computes for two successive stereo image pairs from time $k-1$ and k a relative pose between these points in time. Here this is done by making use of key point correspondences, which can be obtained as described in Chapter 3. They are used to derive the relative pose of the stereo camera between

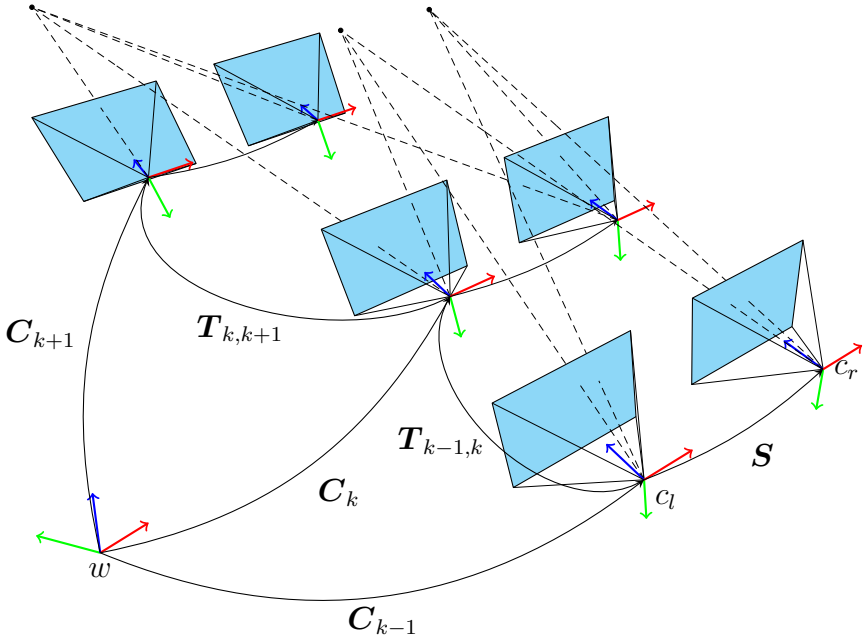


Figure 4.2: Illustration of the stereo visual odometry motion estimation. A stereo camera consists out of two cameras with the coordinate frames c_l and c_r, whose fixed relative pose is given by the transformation $\boldsymbol{S}$. The motion of that camera is measured with respect to the fixed frame w. In order to measure the motion, at time $k = 0$ the initial absolute pose $\boldsymbol{C}_0$ of the stereo camera is given. From there on relative poses are computed using point correspondences. By accumulating these poses new absolute poses can be computed [SF11].

time k and $k-1$ as a rigid transformation. This transformation consists out of a rotation $\boldsymbol{R}_{k-1,k}$ and a translation $\boldsymbol{t}_{k-1,k}$ that are potentially estimated separately. The rigid transformation $\boldsymbol{T}_{k-1,k} \in \mathbb{R}^{4\times4}$ is defined as follows:

$$\boldsymbol{T}_{k-1,k} = \begin{pmatrix} \boldsymbol{R}_{k-1,k} & \boldsymbol{t}_{k-1,k} \\ \boldsymbol{0} & 1 \end{pmatrix}. \tag{4.6}$$

The motion measurement starts at time $k = 0$ and is measured with respect to a fixed frame of reference w. For this a predefined initial absolute pose $\boldsymbol{C}_0 \in \mathbb{R}^{4\times4}$ that determines the transformation of the camera with respect to w is needed. Note that here the transformations are defined inverse to those of Chapter 2. The reason for this is that different conventions in the field of motion estimation and computer vision are used. Inverting the transformation matrix will transform between both conventions.

Starting with, $\boldsymbol{C}_0$, stereo visual odometry accumulates the relative poses to compute the following absolute poses of the stereo camera. Therefore a absolute pose $\boldsymbol{C}_k$ at time k is defined as:

$$\boldsymbol{C}_k = \boldsymbol{C}_{k-1}\boldsymbol{T}_{k-1,k}. \tag{4.7}$$

The camera trajectory starting at time $k = 0$ to the current time n is then defined as the set of absolute camera poses $C_{0:n} = \{\boldsymbol{C}_0, ..., \boldsymbol{C}_n\}$ [SF11]. This process is illustrated in Figure 4.2.

The estimation of the relative poses is done by solving an nonlinear optimization problem. Its state

$$\boldsymbol{x} = [\boldsymbol{R}_{k-1,k}, \boldsymbol{t}_{k-1,k}] \tag{4.8}$$

represents the currently estimated relative pose $\boldsymbol{T}_{k-1,k}$.

4.3 Visual Motion Initialization

As mentioned in Section 4.1 a reasonably good initialization of the nonlinear optimization problem is needed in order to enable it to converge to a correct solution. In case of stereo visual odometry a good relative pose estimate of $\boldsymbol{T}_{k-1,k}$ is needed. One possibility is to compute an initialization by making use of the essential matrix. As already mentioned in Chapter 2 it can be computed from five 2D to 2D point correspondences. This is known as the 5-point problem and the algorithm that solve these problems are called *5-point algorithms* [Nis04, LH06]. Since sometimes outlier correspondences are created it can happen that these are used during the computation of the essential matrix. In this cases it is possible that the computed essential matrix strongly differs from the correct one. In order to avoid this the 5-point algorithm is typically used in combination with a RANSAC algorithm. This one works as follows. From a set of more than five 2D to 2D point correspondences sets of five point correspondences are chosen at random. For each of these sets a hypotheses for the essential matrix is computed. These hypothesis are scored by verifying how good the essential matrix fits to all 2D to 2D point correspondences. One method for scoring a hypothesis is to count the number of inliers, that fall below a certain point to epipolar line distance, over all point correspondences. Finally the hypothesis with the highest number of inliers is chosen as a result. Since this method is probabilistic it is not guaranteed that it produces always the same solution. But for a high number of hypothesis generations the solution tends to be stable. For the 5-point algorithm this number depends on the expected number of outliers, the number of total point correspondences and the probability with which a correct solution should be chosen [Nis04, SF11].

From the resulting essential matrix then the rotation matrix $\boldsymbol{R}_{k-1,k}$ and a translation vector $\tilde{\boldsymbol{t}}_{k-1,k}$ that is defined up to a scale factor can be computed [HZ03].

For the initial pose the computed $\boldsymbol{R}_{k-1,k}$ can be used directly but $\tilde{\boldsymbol{t}}_{k-1,k}$ has to be scaled to its estimated correct length. The scale can be computed from a 3D point $\boldsymbol{p}^c = (x^c, y^c, z^c)^{\mathrm{T}}$, which was triangulated with the correct scale from the stereo camera and a 3D point $\boldsymbol{p'}^c = (x'^c, y'^c, z'^c)^{\mathrm{T}}$ with incorrect scale. The point $\boldsymbol{p'}^c$ was triangulated from a 2D to 2D point correspondence of the left camera over time. For this $\boldsymbol{R}_{k-1,k}$ and $\tilde{\boldsymbol{t}}_{k-1,k}$ with incorrect scale were used. In a next step the scale factor α of $\boldsymbol{p'}^c$ is computed as:

$$\alpha = \frac{\frac{x^c}{x'^c} + \frac{y^c}{y'^c} + \frac{z^c}{z'^c}}{3}. \tag{4.9}$$

It can be used to scale $\tilde{\boldsymbol{t}}_{k-1,k}$ to its correct length. However, for being robust against outliers the scale is computed for all point correspondences. Then heuristically the 5% of the smallest and 5% of the biggest scale factors are discarded, since it is assumed that they contain outliers. After that the mean $\tilde{\alpha}$ over all remaining scale factors is computed. The scalar $\tilde{\alpha}$ is then used to compute an initial estimate of the correct scaled translation vector:

$$\boldsymbol{t}_{k-1,k} = \tilde{\alpha} \cdot \tilde{\boldsymbol{t}}_{k-1,k}. \tag{4.10}$$

The method to derive the initial pose here presented is inspired by the method that is described in [SF11] to derive a consistent scaled motion of a monocular camera from 2D to 2D point correspondences. This method uses also triangulated 3D points but without a metric and it uses another formula to estimate scale. Also the method for deriving the translation vector in [CP15] is similar in terms that it uses reconstructed 3D points triangulated with a metric scale from a previous stereo image pair. Here during an optimization procedure the full translation vector whose scale is derived from the triangulated 3D points is estimated. However the here presented method is to the best of our knowledge not presented in the literature and therefore assumed to be new.

4.4 Cost Functions for Visual Motion Estimation

In the presented work feature correspondences with and without depth information are utilized for motion estimation in order to profit from both. This idea is inspired by Zhang et. al.[ZKS14] who did this in a similar way. For this two different types of cost functions are used.

The first type of cost function computes the reprojection error, that quantifies the distance between a predicted projection of a 3D point and its corresponding observation. This quantification of how good the current state of the motion model fits to the measured key points is done in the left stereo image, which was captured

at time k. A basic requirement for this is, that a 3D to 2D point correspondence is known, where $\boldsymbol{p}_{k-1}^{c_l}$ was triangulated at time $k-1$ and its corresponding 2D key point $\boldsymbol{p}_k^i$ at time k in the left stereo image is known. By making use of the currently estimated motion transformation $\boldsymbol{p}_{k-1}^{c_l}$ is transformed into

$$\boldsymbol{p}_k^{c_l} = \boldsymbol{R}_{k-1,k}^T \boldsymbol{p}_{k-1}^{c_l} - \boldsymbol{R}_{k-1,k}^T \boldsymbol{t}_{k-1,k} \tag{4.11}$$

in camera coordinates c_l of the left stereo camera at time k. In a next step $\boldsymbol{p}_k^{c_l}$ is projected into its predicted projection $\boldsymbol{p}'^i$in image coordinates. To compute the residual error the corresponding measurements that were originally detected in pixel coordinates have to be transformed into image coordinates as described by Equation 2.6 and 2.7. Finally the two-dimensional residual vector for the left camera:

$$r\left(\boldsymbol{x}; \boldsymbol{p}_{k-1}^{c_l}, \boldsymbol{p}_k^i\right) = \boldsymbol{p}'^i - \boldsymbol{p}^i \tag{4.12}$$

can be defined. An important property of these cost functions is that its residual prediction error decreases only if the translation of the relative pose matches the right scale. This enforces a correct scale of the relative pose during optimization.

The second used type of cost function exploits the epipolar constraint to improve the motion estimation. The idea of the cost function is that a 2D to 2D point correspondence fulfills the epipolar constraint. The cost function first computes from the current estimated state the essential matrix $\boldsymbol{E}_{k-1,k}$ by making use of Equation 2.9. A homogeneous point $\tilde{\boldsymbol{p}}_k^i$ in image coordinates, which was detected at time k is transformed into a epipolar line

$$l_{k-1}^i = \boldsymbol{E}_{k-1,k} \tilde{\boldsymbol{p}}_k^i \tag{4.13}$$

at time $k-1$. The point to line distance $d(l_{k-1}^i, \tilde{\boldsymbol{p}}_{k-1}^i)$ between the epipolar line l_{k-1}^i and $\tilde{\boldsymbol{p}}_{k-1}^i$ in image coordinates can then be used as a measure of how good the estimated relative pose fits to the epipolar constraint. The same measure can also be computed the other way around. For computation of the epipolar line l_k^i the transposed essential matrix $\boldsymbol{E}_{k-1,k}^T$ in combination with $\tilde{\boldsymbol{p}}_{k-1}^i$ can be used as described in Equation 2.11. The point to line distance $d(l_k^i, \tilde{\boldsymbol{p}}_k^i)$ describes as a second measure how good the pose estimation fits. The resulting cost function computes the two-dimensional residual

$$e(\boldsymbol{x}; \tilde{\boldsymbol{p}}_{k-1}^i, \tilde{\boldsymbol{p}}_k^i) = \left(d(l_k^i, \tilde{\boldsymbol{p}}_k^i), d(l_{k-1}^i, \tilde{\boldsymbol{p}}_{k-1}^i)\right)^{\mathrm{T}}. \tag{4.14}$$

Since for the computation of the essential matrix the scale of $\boldsymbol{t}_{k-1,k}$ can be arbitrary the presented cost function has only influence on the direction of $\boldsymbol{t}_{k-1,k}$. Nevertheless the incorporation of this cost function into the optimization problem adds more information about the correct relative pose.

The uncertainty of the key point locations is described as a normal distributed with zero mean. Its standard deviation was chosen heuristically as

$$\sigma_{\text{er}} = \frac{1.0\text{px}}{f}, \tag{4.15}$$

where f denotes the average of the focal lengths f_x and f_y in pixels. The division by f is necessary to transform the pixel value into image coordinates. With σ_{er} the covariance matrix

$$\boldsymbol{\Sigma}^{\text{er}} = \text{diag}(\sigma_{\text{er}}^2, \sigma_{\text{er}}^2), \tag{4.16}$$

is defined. It is used to standardize $e(\boldsymbol{x}; \text{d})$ and $r(\boldsymbol{x}, \text{c})$.

Chapter 5

IMU Preintegration on a Manifold

An inertial measurement unit is a sensor system that is able to measure inertial forces to derive information about its motion. The sensors that are contained in this system are three orthogonal rate-gyroscopes for measuring angular velocity and three orthogonal accelerometers for measuring linear acceleration. This enables to derive a fully 3D pose from the IMU measurements over time.

Quadrotors typically use IMUs in combination with a stereo camera that are microelectromechanical systems (MEMS) . They are cheap, lightweight and have a small form factor but come with the trade off that their measurements are to inaccurate in order to use them alone for a precise motion estimation. For example Woodman showed in [Woo07] that after one minute the position error for his pure IMU based navigation system is around 150 meter.

The benefit of combining an IMU with a stereo camera comes from the complementary properties of sensing motion of both sensors. An IMU measures the forces that affect it self, when it is moved and belongs therefore to the category of *propriocetive* sensors. In contrast a camera measures the appearance of the environment as an image and belongs to the *exteroceptive* sensors. The motion measurements from both sensors are therefore independent but should predict the same motion. This makes it possible to fuse them into a single motion estimation problem and take advantage of the measurements from both sensors.

As mentioned in Section 1.1 approaches to combine an IMU with cameras can be divided into loosely and tightly coupled ones. In this work a tightly coupled approach is presented. The integration will be achieved by adding two additional cost functions to the in Section 4.4 to a nonlinear optimization problem. Since an IMU usually produces measurements at a much higher frequency than a stereo camera the nonlinear optimization problem would grow rapidly by integrating every single IMU measurement. To avoid this and make it possible to find a proper solution in an acceptable time the IMU measurements will be preintegrated based on the approach by Forster et. al. [FCDS15a].

Due to the complexity of the integration of an IMU in such an algorithm only the integration of the gyroscope is presented. The position estimation of the visual-inertial sensor unit will instead be derived from image correspondences. However the precision of the stereo visual-inertial algorithm will gain in attitude and position accuracy.

In the following, an introduction to the special orthogonal group SO(3) will be given in Section 5.1. This is needed since it is used during the definition of the residuals for the gyroscope. After that the characteristics of a MEMS Gyroscope will be introduced in Section 5.2. Building up on the characteristics of a gyro it can be explained more in detail, why it is useful to preintegrate the gyroscope measurements in Section 5.3. In the last Section 5.4 then the gyroscope preintegration and its incorporation into a nonlinear optimization problem is explained.

5.1 Special Orthogonal Group SO(3)

The special orthogonal group SO(3) is the group of 3D rotation matrices. Formally this group can be defined as

$$\mathrm{SO}(3) = \{\boldsymbol{R} \in \mathbb{R}^{3\times 3} : \boldsymbol{R}^{\mathrm{T}}\boldsymbol{R} = \boldsymbol{Id}_3, \det(\boldsymbol{R}) = 1\}, \tag{5.1}$$

with the matrix multiplication as group operation and the transposed matrix as inverse. A property of the group is that it is a smooth manifold. This means that the group behaves locally like a Euclidean space but not globally. An illustration of this relationship can be seen in Figure 5.1. The space on the sphere behaves not Euclidean but a limited tangent plane at some point on the sphere does. In theory the tangent plane is part of the tangent space belonging to the manifold, which is also denoted as Lie algebra or $\mathfrak{so}(3)$. The Lie algebra consists out of skew symmetric matrices, which can be used to express small rotations in an Euclidean space. By making use of the hat operator $\wedge$ which maps a vector $\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z)$ to a skew symmetric matrix:

$$\boldsymbol{S} = \boldsymbol{\omega}^{\wedge} = \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix}, \tag{5.2}$$

the so called *exponential map* can be defined. The exponential map $\exp : \mathfrak{so}(3) \to \mathrm{SO}(3)$ maps an element, which is close to identity of $\mathfrak{so}(3)$, to a rotation matrix in SO(3) and is defined as

$$\exp(\boldsymbol{\phi}^{\wedge}) = \boldsymbol{Id}_3 + \frac{\sin\|\boldsymbol{\phi}\|}{\|\boldsymbol{\phi}\|}\boldsymbol{\phi}^{\wedge} + \frac{1-\cos(\|\boldsymbol{\phi}\|)}{\|\boldsymbol{\phi}\|}(\boldsymbol{\phi}^{\wedge})^2. \tag{5.3}$$

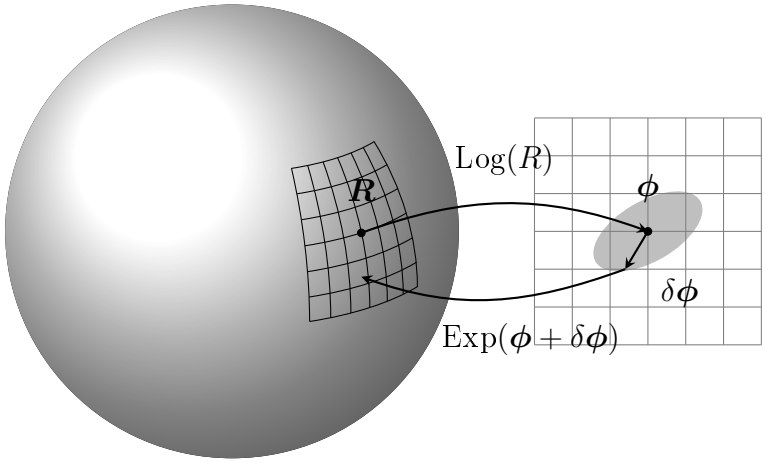


Figure 5.1: Illustration of a manifold as a surface of a sphere. A manifold behaves globally not Euclidean but locally like the surface of the sphere in the figure does. By making use of the logarithm map Log a rotation matrix $\boldsymbol{R}$ can be mapped from SO(3) to a vector in $\boldsymbol{\phi} \in \mathbb{R}^3$, where a uncertainty of $\boldsymbol{\phi}$ can be defined as a normal distribution with zero mean. A small correction $\boldsymbol{\phi} + \delta\boldsymbol{\phi}$ of $\boldsymbol{\phi}$ can then be mapped back to SO(3) by the exponential map Exp.

It will later be used to define a rotation matrix from the measurements of a triple gyroscope. A detailed derivation of the exponential map can be found in [Woo07].

The inverse of the exponential map is the logarithm map $\log : \mathrm{SO}(3) \to \mathfrak{so}(3)$. It maps a rotation matrix $\boldsymbol{R} \neq \boldsymbol{Id}_3$ but close to $\boldsymbol{Id}_3$ to a skew symmetric matrix:

$$\log(\boldsymbol{R}) = \frac{\varphi \cdot (\boldsymbol{R} - \boldsymbol{R}^{\mathrm{T}})}{2\sin(\varphi)} \text{ with } \varphi = \cos^{-1}\left(\frac{\mathrm{tr}(\boldsymbol{R}) - 1}{2}\right). \tag{5.4}$$

From the obtained skew symmetric matrix $\boldsymbol{S} \in \mathfrak{so}(3)$ then its corresponding vector $\boldsymbol{\omega}$ can be extracted by making use of the vee operator $\vee$:

$$\boldsymbol{\omega} = \boldsymbol{S}^{\vee}. \tag{5.5}$$

For notational convenience additional versions of the exponential and logarithm map that operate direct on vectors:

$$\begin{array}{lcccccccc} \mathrm{Exp} & : & \mathbb{R}^3 & \to & \mathrm{SO}(3) & ; & \boldsymbol{\phi} & \to & \exp(\boldsymbol{\phi}^{\wedge}) \\ \mathrm{Log} & : & \mathrm{SO}(3) & \to & \mathbb{R}^3 & ; & \boldsymbol{R} & \to & \log(\boldsymbol{R})^{\vee}, \end{array} \tag{5.6}$$

are used in the following.

5.2 Characteristics of MEMS Gyroscopes

With the relative pose $\boldsymbol{G}$ between the frame of the left camera c_l and the frame g of the gyroscope the absolute pose of the gyroscope can be computed. This

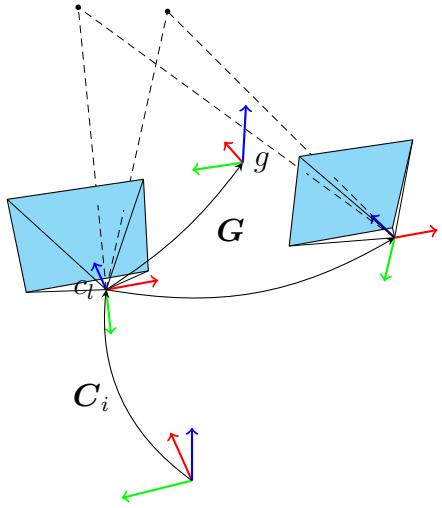


Figure 5.2: Illustration of the inertial stereo camera model. The relative pose $\boldsymbol{G}$ specifies the geometric relation between the left camera frame c_l and the gyro frame g. By concatenating the absolute pose $\boldsymbol{C}_i$ of the stereo camera and the relative pose $\boldsymbol{G}$, the absolute pose of the gyroscope can be computed.

relation is shown in Figure 5.2, where the gyroscope frame is shown in relation to the already from Section 4.2 known coordinate frames of a stereo camera. The relative pose $\boldsymbol{G}$ can be derived during a calibration process. A calibration toolbox like Kalibr[1] can be used, for this. Here the calibration will not be explained in more detail since it is beyond the scope of this thesis.

A triple axis MEMS gyroscope consists out of three orthogonally arranged single-axis rate gyroscopes. They measure the rotational rate for their axis for one point in time. The measurements are then summarized as a single measurement by the triple axis gyroscope. Consequently direct information about 3D angular motion can be obtained from a gyroscope. The measurements are affected by small errors but do not contain outliers as long as the values that have to be measured are in the measurement range of the gyroscope. However the small measurement errors are not negligible and lead to a drift when the orientation of the gyroscope is tracked. The mathematical model, which is used to describe gyroscope measurements describes the errors by a sensor bias $\mathbf{b}(t)$ and additive white noise $\boldsymbol{\eta}(t)$ at time t.

The sensor bias is a slow varying offset from the real angular velocity. It can be observed when the gyroscope is not undertaken any rotation. A simple method to compute an initial gyroscope bias is to average the gyroscope output over a few seconds. However the gyroscope bias changes slowly over time and is therefore

[1]Kalibr web page: https://github.com/ethz-asl/kalibr

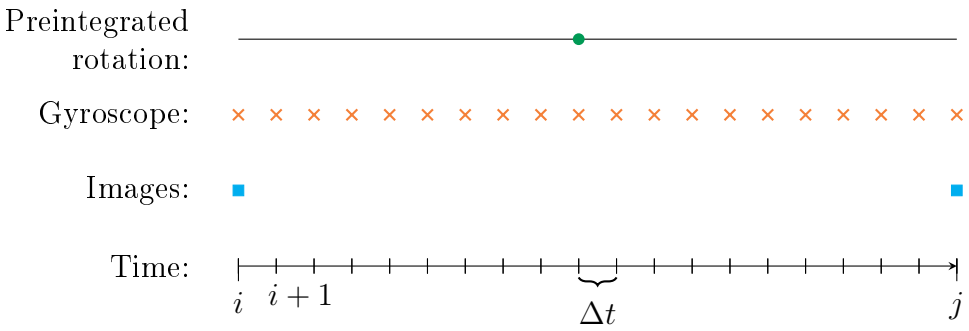


Figure 5.3: Illustration of data rates during preintegration [FCDS15a]. Two stereo image pairs visualized by the two cyan squares are captured at two points in time i and j. The gyroscope measurements, visualized as orange crosses, sampled after every time period Δt. To reduce computational effort the sampled gyroscope measurements are summarized as a preintegrated rotation measurement, which is visualized as green circle with a line.

modeled as a random walk. A random walk consist out of a series of steps, where for each step the direction and size is randomly determined [Woo07]. Therefore the bias is assumed as a normally distributed random variable with bias stability $\boldsymbol{\eta}^{\mathrm{b}}$, which expresses how quickly the bias changes over time. The additive white noise fluctuates much faster than the sampling rate of the sensor. It consists out of uncorrelated random values with a zero mean and the noise strength $\boldsymbol{\eta}$. Like the bias change it is also modeled as a random walk.

By making use of $\mathbf{b}(t)$ and $\boldsymbol{\eta}(t)$ the gyroscope measurement at time t can be defined as

$$\tilde{\boldsymbol{\omega}}(t) = \boldsymbol{\omega}(t) + \mathbf{b}(t) + \boldsymbol{\eta}(t), \tag{5.7}$$

where $\boldsymbol{\omega}(t)$ is the angular velocity without any noise. A more detailed introduction to the gyroscope noise model can be found in the appendix of [Cra06].

The data rate at which the angular rate from a gyroscope is received lies usually between 100Hz and 1kHz. This is much higher than the image rate of a stereo camera [FCDS15b], which causes several gyroscope measurements to be made between two image pairs. An illustration of the different frequencies at which images and gyroscope measurements are received can be seen in Figure 5.3. For a period of time Δt between two points in time t and $t+\Delta t$ the discrete measurement $\tilde{\boldsymbol{\omega}}(t)$ is assumed to be constant.

5.3 Naive Gyroscope Attitude Tracking

In order to track the attitude of the gyroscope its discrete measurements have to be integrated over time. The measurements are considered to be constant for a period of time Δt between two gyroscope measurements. Since it is known that the measurements are affected by white noise and a bias, the estimated noise term as well as the bias at time t can be considered. This is done by reformulating Equation 5.7 to:

$$\boldsymbol{\omega}(t) = \tilde{\boldsymbol{\omega}}(t) - \mathbf{b}(t) - \boldsymbol{\eta}^{\mathrm{d}}(t), \tag{5.8}$$

where $\boldsymbol{\eta}^{\mathrm{d}}$ denotes the discrete white noise. Its covariance is a function of the sampling rate and relates to the continuous noise $\boldsymbol{\eta}$ by $\mathrm{Cov}(\boldsymbol{\eta}^{\mathrm{d}}(t)) = \frac{1}{\Delta t}\mathrm{Cov}(\boldsymbol{\eta}(t))$ [FCDS15a]. With a known attitude $\boldsymbol{R}(t)$ the attitude at $t + \Delta t$ can be computed as:

$$\boldsymbol{R}(t + \Delta t) = \boldsymbol{R}(t)\mathrm{Exp}((\tilde{\boldsymbol{\omega}}(t) - \mathbf{b}(t) - \boldsymbol{\eta}^{\mathrm{d}}(t))\Delta t). \tag{5.9}$$

From Equation 5.9 it is already possible to derive a cost function, which could be integrated into an nonlinear optimization problem. But this would increase the number of cost functions as well as the number variables in the state of the mathematical model. It would increase by values that determine the current orientation of the gyroscope and its bias estimate for every orientation measurement. In consequence the motion estimation would become much more computationally complex.

5.4 Gyroscope Preintegration

The gyroscope preintegration has the goal to reduce the number of states that need to be estimated during the nonlinear optimization. Therefore the set $G_{i,j}$ of gyroscope measurements between time i and j of two consecutive stereo image pairs can be summarized to one integrated rotation measurement:

$$\boldsymbol{R}_{i,j} = \prod_{k=i}^{j-1} \mathrm{Exp}((\tilde{\boldsymbol{\omega}}_k - \mathbf{b}_k - \boldsymbol{\eta}_k^{\mathrm{d}})\Delta t). \tag{5.10}$$

This product still contains the white noise and slowly changing bias terms, which cause $\boldsymbol{R}_{i,j}$ to be inaccurate if they are not correctly estimated.

To solve this problem the white noise is isolated in a first step. Therefore $\mathbf{b}_i$ is assumed to be constant between i and j:

$$\mathbf{b}_i = \mathbf{b}_{i+1} = ... = \mathbf{b}_{j-1}. \tag{5.11}$$

Then Equation 5.10 is transformed by making use of a first order approximation into

$$\boldsymbol{R}_{i,j} \simeq \tilde{\boldsymbol{R}}_{i,j}(\mathbf{b}_i)\mathrm{Exp}(-\delta\boldsymbol{\phi}_{i,j}) \ , \tag{5.12}$$

where $\tilde{\boldsymbol{R}}_{i,j}(\mathbf{b}_i) = \prod_{k=i}^{j-1} \mathrm{Exp}(\tilde{\boldsymbol{\omega}}_k - \mathbf{b}_i)$ is called the *preintegrated rotation measurement* computed for the constant bias $\mathbf{b}_i$ and $\delta\boldsymbol{\phi}_{i,j}$ is the isolated noise term. This term expresses how the noise has been propagated over time. It enables to define the uncertainty in $\mathfrak{so}(3)$, being normally distributed with zero mean.

In $\mathfrak{so}(3)$ the uncertainty is defined by a covariance matrix $\boldsymbol{\Sigma}^{\eta}_{i,j}$, which is iteratively computed during the preintegration. A complete derivation of the iterative computation of $\boldsymbol{\Sigma}^{\eta}_{i,j}$ can be found in the supplementary material of [FCDS15a].

To integrate the slowly-varying bias changes into this model it would be possible to recompute the preintegrated rotation measurement every time when the bias changes. While this would be computationally very expensive it is possible to approximate a change of $\tilde{\boldsymbol{R}}_{i,j}(\mathbf{b}_i)$ caused by a small correction $\delta\mathbf{b}$ of the bias with a first order expansion:

$$\boldsymbol{R}_{i,j}(\mathbf{b}_i) \simeq \tilde{\boldsymbol{R}}_{i,j}(\mathbf{b}_i)\mathrm{Exp}\left(\frac{\partial \bar{\boldsymbol{R}}_{i,j}}{\partial \mathbf{b}}\delta\mathbf{b}\right), \tag{5.13}$$

where $\frac{\partial \bar{\mathbf{R}}_{i,j}}{\partial \mathbf{b}}$ is the partial derivation of $\boldsymbol{R}_{i,j}(\mathbf{b}_i)$ with respect to $\mathbf{b}_i$. It describes how the preintegrated rotation measurement changes due to bias changes and can be iteratively computed with the preintegrated rotation measurement. A complete derivation of $\frac{\partial \bar{\mathbf{R}}_{i,j}}{\partial \mathbf{b}}$ can be found in the supplementary material to [FCDS15a].

With the integration of the bias updates in Equation 5.13 now the residual for the preintegrated rotation measurement can be defined. Let $\boldsymbol{R}_i$ and $\boldsymbol{R}_j$ be the estimated orientation of the gyroscope at time i respectively j. Then the residual vector for the preintegrated rotation measurement is defined as:

$$g(\boldsymbol{x}; G_{i,j}) = \mathrm{Log}\left(\left(\tilde{\boldsymbol{R}}_{i,j}(\mathbf{b}_i)\mathrm{Exp}\left(\frac{\partial \bar{\boldsymbol{R}}_{i,j}}{\partial \mathbf{b}}\delta\mathbf{b}\right)\right)^T \boldsymbol{R}_i^T \boldsymbol{R}_j\right). \tag{5.14}$$

It computes the difference between the predicted relative orientation $\boldsymbol{R}_i^T\boldsymbol{R}_j$ of the gyroscope and the preintegrated rotation measurement in $\mathfrak{so}(3)$ by making use of the logarithm map. In order to model the uncertainty of the preintegrated rotation measurement the previous computed covariance matrix $\boldsymbol{\Sigma}^{\eta}_{i,j}$ can now be used to standardize $g(\boldsymbol{x}, G_{i,j})$ with the Mahalanobis distance as described in Section 4.1.

As a last step the bias updates have to be integrated into the optimization problem. This needs an extension of the state $\boldsymbol{x}$, to which the bias estimate $\mathbf{b}_i$ is added. For integration of the bias it is assumed that it changes over the time interval $\Delta t_{i,j}$ by a discrete noise $\boldsymbol{\eta}^{\mathrm{bd}}$. The bias $\mathbf{b}_j$ at time j is then defined as:

$$\mathbf{b}_j = \mathbf{b}_i + \boldsymbol{\eta}^{\mathrm{bd}}. \tag{5.15}$$

As mentioned in Section 5.2 the bias change is modeled as a random walk. The noise of the random walk has zero mean and covariance $\mathbf{\Sigma}_{i,j}^{\mathrm{bd}} = \Delta t_{i,j} \mathrm{Cov}(\boldsymbol{\eta}^{\mathrm{b}})$. With this the residual:

$$h(\boldsymbol{x}) = \mathbf{b}_j - \mathbf{b}_i, \tag{5.16}$$

can be defined, which is standardized with $\mathbf{\Sigma}_{i,j}^{\mathrm{bd}}$.

Chapter 6

Stereo Visual-Inertial Odometry for UAVs

In the previous chapters different parts of the stereo visual-inertial odometry problem and possible algorithmic solutions to them have been explained. Based on these algorithms a concrete procedure of a stereo visual-inertial odometry algorithm can be defined.

To use this algorithm in practice it was implemented in modern *C++11*. In order to let a software framework make use of the algorithm, interfaces that enable to pass IMU and stereo image data to the algorithm are needed. Vice versa the algorithm can pass computed poses and status informations to the software framework.

As a main goal of this thesis that was previously defined in Section 1.2 the developed algorithm should be integrated into the Aerostack framework. Due to this the algorithm was integrated into Aerostack by making use of the mentioned interfaces. This enables an UAV that works with Aerostack to use the presented stereo visual-inertial algorithm for navigation.

The rest of the chapter is structured as follows. In Section 6.1 the connections of the algorithms that solve the single part problems for stereo visual-inertial odometry is explained. After that a short introduction to the main characteristics of Aerostack is given in Section 6.2 followed by an description of how the algorithm was integrated into Aerostack in Section 6.3.

6.1 Stereo Visual-Inertial Odometry Algorithm

The schema drawing in Figure 6.1 gives an overview about the developed algorithm and should support the following explanations.

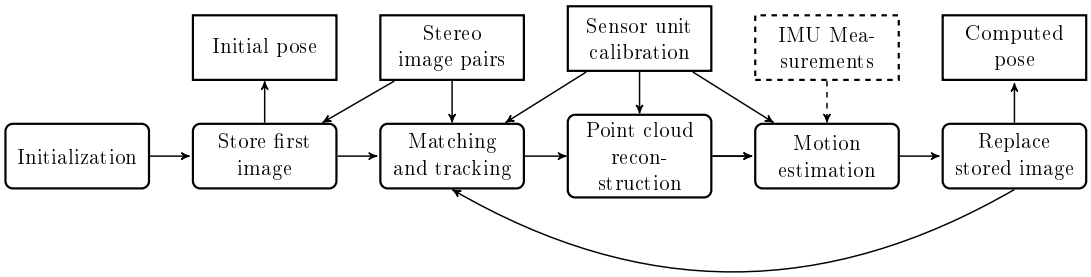


Figure 6.1: Stereo visual-inertial odometry pipeline. The figure visualizes the single algorithmic steps and data flows during the execution of the stereo visual-inertial odometry algorithm. Therefore boxes with rounded corners indicate algorithmic processes and boxes with top corners data. The data flow is shown by arrows. Since the visual odometry algorithm is able to estimate motion without the data of the IMU, IMU measurements are surrounded by a dashed frame that indicates that they are optional.

Before the stereo visual-inertial algorithm can operate the sensor unit has to be calibrated. During this separated process the distortion coefficients, the intrinsic camera parameters, and the extrinsic camera parameters of the stereo camera are determined. In case that a stereo visual-inertial sensor unit is used the relative pose of the IMU to the stereo camera has to be computed also. Additionally, an initial value for the gyroscope bias has to be estimated before every start of the algorithm. It might be different with every start up of the IMU especially after a longer time. The wrong initialized bias would result in a drift of the computed trajectory that is stronger than for a correct initialized bias.

In the first step during operation of the algorithm it is initialized. Process parameters are loaded by this and variables are set to an predefined value. Here especially the initial pose of the camera system with respect to a global coordinate system has to be mentioned. This pose is important since the odometry algorithm will compute the motion relative to it, as explained in Section 4.

The first stereo image pair from the stereo camera needs special treatment. This is since no previous image pair exists to which a relative pose can be computed. Therefore it is stored until another stereo image pair is received from the stereo camera. However, the initial absolute pose is provided by the algorithm.

When a second stereo image pair at time j is passed to the algorithm, it starts to process the stored image pair from time i. On both images FAST key points are detected in a first step that is followed by bucketing for both images. For the key points from the left and the right image then Rotated BRIEF descriptors are computed. By making use of the descriptors and the calibration data of the sensor

unit a constrained matching is performed to obtain key point correspondences between the left and right image. Additionally, tracking between the stored and the newly received left image is performed.

For key points which could be tracked and matched at the same time point correspondences with key points in three images exist. These key point correspondences can be used to triangulate 3D points using corresponding points of the left and right stereo image from time i. The 3D points are then in the coordinate frame of the left stereo camera. Since the key points in the stored image that belong to the 3D points are known it is possible to deduce the corresponding key points in the left stereo image at time j. In this way 3D to 2D point correspondences can be stored in a set $C_{i,j}$. For key points that were only tracked in the left stereo image between time i and j the resulting correspondences are stored in the set $D_{i,j}$.

The motion estimation of the stereo visual odometry algorithm starts with the computation of an initial relative pose. This is done using the algorithm presented in Section 4.3. It processes 2D to 2D point correspondences for which also 3D to 2D point correspondences exist. After that a nonlinear optimization problem is solved in order to refine the initial pose. It minimizes a sum over the cost functions presented in Section 4.4 and 5.4 by adapting the in Equation 4.8 defined state. Therefore it is defined as

$$\underset{\boldsymbol{x}}{\operatorname{argmin}} \frac{1}{2} \sum_{\mathrm{c} \in C_{i,j}} \rho_{\mathrm{Tukey}}(\|r\left(\boldsymbol{x}; \mathrm{c}\right)\|^2_{\boldsymbol{\Sigma}^{\mathrm{er}}}) + \frac{1}{2} \sum_{\mathrm{d} \in D_{i,j}} \rho_{\mathrm{Tukey}}(\|e\left(\boldsymbol{x}; \mathrm{d}\right)\|^2_{\boldsymbol{\Sigma}^{\mathrm{er}}}). \tag{6.1}$$

In order to make the motion estimation robust the Tukey loss function is applied to the individual summands.

Gyroscope measurements can optionally be integrated into the nonlinear motion estimation problem. The integration is simply done by adding the residual from Equation 5.14. In order to estimate the gyroscope bias correctly a cost function that computes the residual from Equation 5.16 is added too. Since there are no outliers for the gyroscope measurements and its bias estimate the trivial loss function is applied to their computed residuals. The optimization problem for the stereo visual-inertial odometry algorithm is therefore defined as

$$\begin{aligned} \underset{\boldsymbol{x}}{\operatorname{argmin}} & \frac{1}{2} \sum_{c \in C_{i,j}} \rho_{\mathrm{Tukey}} \left(\|r\left(\boldsymbol{x}; \mathrm{c}\right)\|^2_{\boldsymbol{\Sigma}^{\mathrm{er}}}\right) + \frac{1}{2} \sum_{d \in D_{i,j}} \rho_{\mathrm{Tukey}} \left(\|e\left(\boldsymbol{x}; d\right)\|^2_{\boldsymbol{\Sigma}^{\mathrm{er}}}\right) + \\ & \frac{1}{2} \left(\rho_{\mathrm{trivial}} \left(\|g(\boldsymbol{x}; G_{i,j})\|^2_{\boldsymbol{\Sigma}^{\eta}_{i,j}} \right) + \rho_{\mathrm{trivial}} \left(\|h(\boldsymbol{x})\|^2_{\boldsymbol{\Sigma}^{\mathrm{bd}}_{i,j}} \right) \right). \end{aligned} \tag{6.2}$$

After motion estimation the estimated relative pose is concatenated with the previously computed absolute pose and the new pose is provided by the algorithm. Furthermore, the last received stereo image pair is stored. This is the final processing step for one stereo image pair. After receiving a new stereo image pair

the algorithm starts again with the matching and tracking procedure for another iteration.

6.2 Aerostack Framework

Aerostack is an open source software framework for aerial robotics. Its goal is to provide versatile software that can easily be used in a broad field of applications for UAVs with a high degree of autonomy. A special feature of Aerostack is its software architecture that allows a human operator to potentially control multiple robotic agents that can interact as a swarm. The architecture of the software that runs on a robotic agent can be described as five stacked layers.

- The *social layer* contains a communication system. It provides capabilities for human-robot interaction as well as for robot-robot interaction over a network connection.
- The supervision system of the *reflective layer* supervises the internal state of a robotic agent and tries to react on possible events or problems like an unexpected obstacle. The current state can be reported to other robots or the human operator through the social layer.
- In the *deliberative layer* abstract solutions to complex tasks like the computation of a trajectory are generated by a planning system. It incorporates feedback from the reflective layer and provides state informations to it at the same time.
- The *executive layer* generates concrete behavior sequences from the abstract solutions of the deliberative layer by an executive system. Furthermore, a situation awareness system integrates an internal state of the robotic agent from sensor information.
- The *reactive layer* communicates with hardware interfaces and reads sensor information, which are passed to the executive layer. Furthermore, it translates the behavior sequences from the executive system into specific commands for the hardware interface.

In each layer run several programs that are called processes in the Aerostack terminology. Each of these processes is responsible for one specific task during the operation of an UAV like for example obstacle recognition. For the inter process communication Aerostack utilizes the open source Robot Operating System (ROS). The key components of a ROS systems are a roscore and so called ROS nodes that correspond to processes in Aerostack. The roscore connects the ROS nodes using

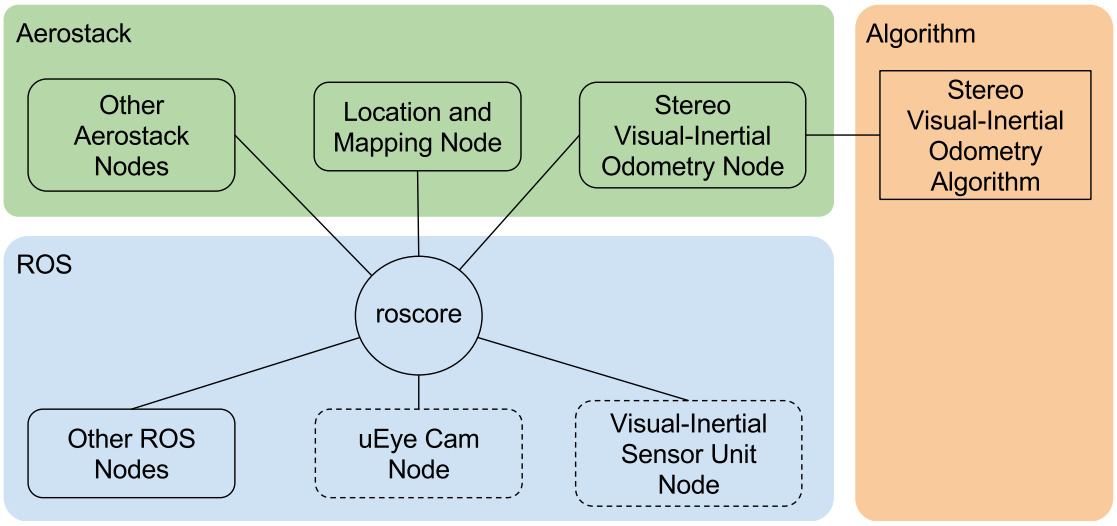


Figure 6.2: Schema of the algorithm integration into Aerostack. A roscore organizes the communication between ROS nodes. The stereo visual-inertial odometry algorithm is integrated in a stereo visual-inertial odometry node that implements an Aerostack interface. It receives input data via ROS topics from ROS-nodes that interface with hardware, visualized by the dotted border. The computed visual odometry pose is then sent to a location and mapping node.

ROS topics. On each topic a specific type of data in form of ROS messages can be sent between nodes. Based on this modular architecture Aerostack provides many nodes whose tasks range from interfacing with UAV hardware to nodes that execute a complete mission plan. Additionally, due to the fact that Aerostack is a ROS based framework also ROS nodes, provided by the ROS community, can easily interface with it [SLFB+16].

6.3 Aerostack Integration

The implementation of the stereo visual-inertial algorithm is realized in modern *C++11*. Figure 6.2 visualizes the mode of implementation. An instance of the class `StereoVisualInertialOdometer` stores parameters for the stereo visual-inertial odometry algorithm and implements it. The gyroscope measurements and the stereo image pairs can be added to the algorithm through two separated methods. When a stereo image pair is added the computation of a relative pose is triggered automatically except for the first image pair. In case of the first stereo image pair the initial pose is provided directly by the algorithm.

In order to interface with Aerostack a stereo visual-inertial odometry node creates an object of type `StereoVisualInertialOdometry` that inherits from the

abstract Aerostack class `DroneProcess`. The abstract methods that had to be implemented for this are responsible to initialize, start, and stop the odometry algorithm. While the node is running it receives IMU measurments and stereo image pairs through messages and passes them to a `StereoVisualInertialOdometer` object. After processing, the computed absolute pose is send to a location and mapping node on the executive layer.

For synchronized stereo image pairs a visual-inertial sensor unit node sends an electric trigger signal to two iDS uEye cameras. Their images are read and sent as ROS messages by two ROS uEye Cam nodes. Additionally, the visual-inertial sensor unit node is prepared to send IMU measurements. All these hardware related nodes work on the reactive layer of Aerostack.

Chapter 7

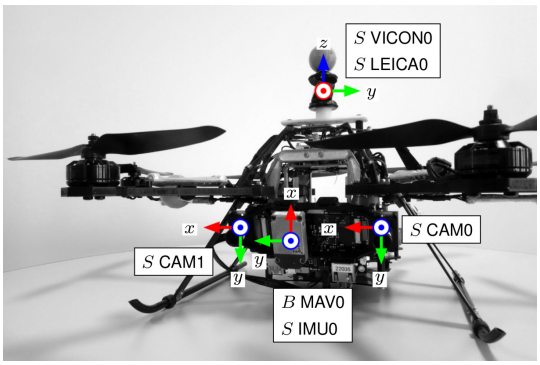
Evaluation

Evaluation of stereo visual-inertial odometry algorithms is often done by making use of existing datasets. They provide the data necessary for the algorithm to work as well as the ground truth trajectory to which the computed results of the algorithm can be compared. This evaluation method has the advantage that it allows for easy comparison based on common error measures. They have different properties and are used to measure different quality criteria.

In the thesis at hand the visual odometry datasets of the KITTI Vision Benchmark Suite and the European Robotics Challenge (EuRoC) datasets[1] [BNG+16] are used for evaluation. Since the KITTI datasets only provide images of a stereo camera, the evaluation focuses on the EuRoC datasets which also provide IMU data. The KITTI datasets provide a benchmark list that uses its own error measurements, which were also implemented in this work. For evaluation of the EuRoC datasets the *absolute trajectory error* (ATE) will be used. It was introduced by [SEE+12] to measure the global consistency of trajectories from SLAM algorithms. However, recently [KDB17] use this method for evaluation of visual odometry algorithms on the EuRoC datasets and was therefore chosen. Additionally it provides the possibility to evaluate with ground truth trajectories which do not contain orientation information. This is partially the case for the EuRoC datasets.

The remainder of this chapter is structured as follows. In the first Section 7.1 the properties of the used datasets will be presented. After that in Section 7.2 two common error measurements, which are used during the evaluation will be explained. Building up on these explanations an evaluation of the stereo visual odometry algorithm with the KITTI datasets is done in Section 7.3. In the next Section 7.4 the influence of the gyroscope to the trajectory reconstruction will be

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



(a) The EuRoC hexacopter with sensor unit

(b) EuRoC machine hall environment

Figure 7.1: Images of the EuRoC hexacopter and machine hall environment. In Figure (a) shows the hexacopter, which was used to record the datasets. In front the visual-inertial sensor unit is shown with plotted frames of reference for the single sensors. At the top of the hexacopter the marker for pose tracking is mounted. Figure (b) gives an representative overview of the machine hall environment [BNG+16].

evaluated on the EuRoC datasets. Finally in Section 7.5 a runtime evaluation is done.

7.1 Datasets

The odometry datasets of the KITTI Vision Benchmark Suite were recorded from a moving car in urban environments. The 22 datasets are split into eleven training datasets and eleven datasets for evaluation. A special feature of the KITTI Vision Benchmark Suite is that algorithms can be uploaded for evaluation on a web page. For all submitted algorithms a leader board is provided that enables to compare the performance of different visual odometry methods. The stereo camera system of the car was equipped with two grayscale and two color cameras, which capture stereo image pairs with 10 frames per second. During the tests presented in this thesis the rectified grayscale stereo image pairs with a resolution of 1241×376 pixels where used. They have the property that epipolar lines are horizontal and that corresponding points have the same vertical position. As a result of rectification the images are sharp in the image center but blurred in the border areas. Moreover, the images show moving objects like vehicles or pedestrians that can cause outlier image correspondences. The ground truth information of the datasets is provided as a 6D pose. These were computed from a GPS receiver with an integrated IMU. Note that the IMU is only used to compute a accurate ground truth trajectory and is not provided in the odometry datasets [GLU12].

The EuRoC datasets contain data records captured with the hexacopter shown in Figure 7.1a. It is equipped with a visual-inertial sensor unit. The synchronized sensor system consists of two monochrome cameras with a resolution of 768 × 480 pixel capturing images at 20 Hz and an IMU providing gyroscope as well as accelerometer data at 200 Hz. The eleven datasets were captured in three different environments with different properties. Vicon room 1 (VC1) datasets are captured in a bright room which already has been shown in Figure 2.4b and 3.2b. In contrast images from Vicon room 2 (VC2) dataset were captured in the same room with covered windows and different obstacles. From both environments exist respectively datasets with three levels of difficulty (V1-1 and V2-1 easy; V1-2 and V2-2 medium; V1-3 and V2-3 difficult) resulting from slow to fast motion speeds and good to bad lighting conditions. As an additional challenge slightly moving curtains are visible on some datasets. They can cause outliers that are hard to detect due to their small variations in image space. As ground truth of these datasets a full 6D pose is provided captured by a Vicon motion capture system. The remaining 5 datasets were recorded in a machine hall (MH) environment. These datasets only contain the ground truth position captured by a Leica laser tracker. The degrees of difficulty are here also divided into three levels (MH-1 and MH-2 easy; MH-3 medium; MH-4 and MH-5 difficult) caused by different motion speeds and lighting conditions. A representative image of the machine hall environment is shown in Figure 7.1b.

7.2 Error Measurements

The KITTI Visual odometry and SLAM evaluation uses its own error measures to evaluate translational and rotational error separately. In order to define these errors let $\boldsymbol{p}_i \in SE(3)$ be the estimated absolute pose for a stereo image pair i and $\hat{\boldsymbol{p}}_i \in SE(3)$ the corresponding absolute ground truth pose. The relative pose $\boldsymbol{\delta}_{i,j} \in SE(3)$ between two poses $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ of a stereo image pair j is then defined with the inverse pose composition operator $\ominus$ as

$$\boldsymbol{\delta}_{i,j} = \boldsymbol{p}_j \ominus \boldsymbol{p}_i. \tag{7.1}$$

The operations $\angle(\cdot)$ and $\|\cdot\|$ extract then the angle of the rotation respectively the Euclidean norm in meter of a pose.

Additionally, a set F of stereo image pairs $(i,j) \in F$ has to be defined. For this set $\|F\|$ denotes the overall Euclidean length of all (i,j). Based on this a rotational error measurement

$$e_{\text{rot}}(F) = \frac{1}{\|F\|} \sum_{(i,j)\in F} \angle(\boldsymbol{\delta}_{i,j} \ominus \hat{\boldsymbol{\delta}}_{i,j}) \tag{7.2}$$

with the unit [deg/m] and a translational error measurement

$$e_{\text{trans}}(F) = \frac{1}{\|F\|} \sum_{(i,j)\in F} \|\boldsymbol{\delta}_{i,j} \ominus \hat{\boldsymbol{\delta}}_{i,j}\| \tag{7.3}$$

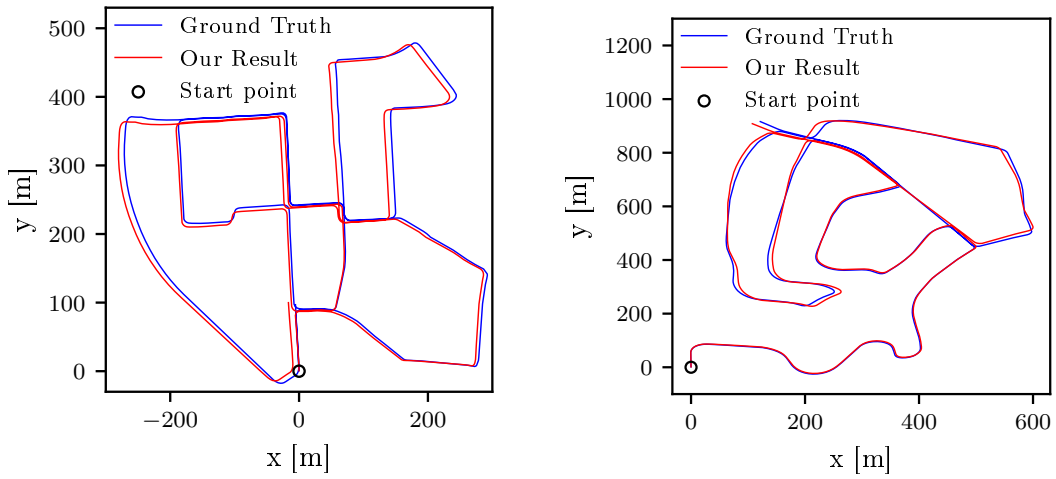
which defines the error in percent of $\|F\|$. In the case of the KITTI error measures i and j must not be consecutive stereo image pairs. By analyzing the code of the odometry development kit from the KITTI web page it can be seen that i is increased in steps of 10 image pairs. Than for each i, j is chosen in a way that $\|\boldsymbol{\delta}_{i,j}\|$ is greater than a segment length $l \in \mathbb{R}^+$. In case of KITTI this is done for all $l \in \{100, 200, 300, 400, 500, 600, 700, 800\}$, where l has the unit meter [GLU12]. The error values that are presented for the ranking in the benchmark list are defined as the average of the error measures for all datasets.

The *absolute trajectory error* (ATE) was introduced by [SEE+12] to measure the global consistency of trajectories from SLAM algorithms. In a first step, the algorithm aligns the computed trajectory with the ground truth trajectory in a way that the squared mean distance between the positions of corresponding poses on both trajectories are minimal. For instance, the method developed by Horn [Hor87] allows for this. From all distances between these poses the root mean square error is computed. Since the ATE measures the global consistency, it would compute for a trajectory that contains a large pose error in its middle but perfect estimated relative poses at all other places a high error value. In contrast a trajectory that contains several small rotation errors can achieve a better ATE error value.

7.3 Accuracy on KITTI Datasets

The evaluation of the stereo visual odometry algorithm was executed on two KITTI training datasets. The first dataset is KITTI 00 and has a total length of approximately 3723 meters. KITTI 02 is the second dataset and has a length of approximately 5065 meters.

The reconstructed trajectories of both datasets shows Figure 7.2 as a projection into the xy-plane. Note that in this common type of visualization the drift of the trajectory in direction of the z-axis is not visible. For both datasets the accumulation and the resulting drift is clearly notable in the visualization of the trajectory. However, the algorithm does not loose the track at any place of the reconstructed trajectories. The computed error measures for both datasets are shown in Table 7.1. Notably, the result on KITTI 00 is better than on KITTI 03 for the ATE. Nevertheless, for the two KITTI error measurements the algorithm is better on the dataset KITTI 03. This opposed result comes from the characteristics of the KITTI 00 dataset. It has many overlapping trajectory paths on shorter



(a) KITTI 00 in xy-plane

(b) KITTI 02 in xy-plane

Figure 7.2: Reconstructed trajectories in xy-plane for datasets KITTI 00 and KITTI 02 The algorithm is able to track the trajectory over the full dataset in both cases despite a small drift.

trajectory length in combination with a smaller spatial extension. This makes it possible to align ground truth and reconstructed trajectory close to each other.

The computed KITTI error measurements enable a coarse classification of the algorithm into the KITTI leader board as an orientation of its performance. Note that this comparison has due to the fact that the leader board was computed on more and other datasets only a limited validity. An extraction of the leader board containing only visual odometry methods can be found in Appendix A. The leader board is sorted ascendingly by the KITTI translation error. For this sorting the presented method can be classified approximately to place 10. The best method concerning the translation error is SOFT2 with a value of 0.81%. It is obviously based on SOFT[CP15] that achieves a translation error of 0.88%. The SOFT algorithm uses a more complex feature matching and tracking stage in combination with a motion estimation that utilizes the 5 point-algorithm and nonlinear optimization. Also the ROCC[BW16b] (Robust Outlier Criterion for Camera-based Odometry) method that was developed in the context of automotive applications has to be mentioned. It achieves a translation error of 0.98% and focuses on feature selection, which is done by selecting outliers in an iteratively manner. For this a new normalized reprojection error measure is used. Of this method, two more variants RotROCC[BW16a] with 0.88% and HypERROCC with 0.88% perform better than the method presented in this thesis. For HypERROCC also no reference is given but due to the name of the method and the name of the

dataset	ATE [m]	$e_{\text{trans}}(F)\%$	$e_{\text{rot}}(F)$[deg/m]
KITTI 00	5.12	1.00	0.0032
KITTI 02	8.01	0.94	0.0031

Table 7.1: Evaluation results on KITTI datasets.

	V1-1	V1-2	V1-3	V2-1	V2-2	V2-3
stereo visual	0.66	0.47	-	0.55	0.70	-
stereo visual-inertial	0.21	0.19	-	0.42	0.68	-

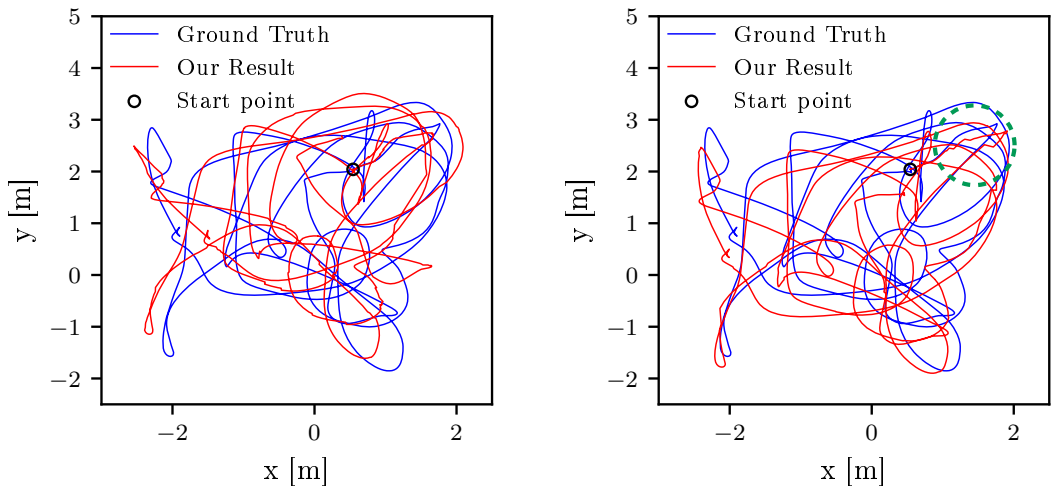
Table 7.2: ATE values for the results of the EuRoC Vicon room 1 and 2 datasets in meter.

author it can be assumed that it is a modification of ROCC. The other methods better than place 10 achieve similar results but for them no references are given.

The rotation error of the developed algorithm lies for KITTI 00 at 0.0031 [deg/m] and for KITTI 02 at 0.0032 [deg/m]. These values are slightly higher than the values of the top methods in the benchmark list but still comparable.

7.4 Influence of the Gyroscope

The gyroscope measurments of the EuRoC datasets allow evaluating the gyroscope influence on the accuracy of the visual odometry algorithm. On all datasets the algorithm was ran with processing and without processing gyroscope data. During each test, all images and gyroscope data were processed by the algorithms in order to obtain comparable results. Table 7.2 displays the ATE values for the results of the Vicon room datasets. Excepted for the datasets V1-3 and V2-3 the stereo visual odometry algorithm and the stereo visual-inertial algorithm are able to preserve the global consistency of the reconstructed trajectory in a way that it makes sense to measure the accuracy with the ATE. In all four cases the algorithms compute several times a wrong translational part of the relative poses. This happens due to the lack of correct key point correspondences and makes an interpretation of the ATE difficult and potentially misleading. One reason for the translational error is that the brightness between two consecutive stereo image pairs can change strongly in dataset V1-3. The camera is in these cases often directly oriented towards large windows which cause the auto exposure to change quickly and therefore the brightness of the images also. In case of V2-3, the key point correspondences are not found due to strong motion blur, different auto exposure settings for the left and the right stereo camera, and only little texture when the camera is oriented towards a white wall.



(a) V1-2 stereo visual odometry result **(b)** V1-2 stereo visual-inertial odometry result

Figure 7.3: Reconstructed trajectories in xy-plane for the dataset EuRoC V1-2. The stereo visual as well as the stereo visual-inertial odometry algorithm are both able to reconstruct the whole trajectory. Nevertheless, the trajectory in Figure (b) reconstructed by the stereo visual-inertial algorithm runs nearly in parallel to the ground truth trajectory. A place at which small jumps of the trajectory are notable is marked with a green dashed circle.

For all other Vicon room datasets, the algorithms are able to track the trajectory, which is according to the ATE measures with the stereo visual-inertial algorithm more precise than for the stereo visual odometry algorithm. The reconstructed trajectories for V1-2 can be seen in Figure 7.3. In these visualizations, the difference of the reconstruction quality can be seen clearly. The result from the stereo visual odometry algorithm in Figure 7.3a has more drift than the one of the stereo visual-inertial algorithm in Figure 7.3b. Therefore the reconstructed trajectory in Figure 7.3b runs close to the ground truth trajectory. However, sometimes small jumps in the trajectories are noticeable. These are in the trajectory of the stereo visual-inertial algorithm more pronounced than for the trajectory of the stereo visual algorithm. At these points, the optimization algorithm of the motion estimation converges to a wrong local minimum.

For the datasets from Vicon room 2 the ATE values are in general higher than for Vicon room 1. One reason for this is a permanent drift in the z-axis and less key point correspondences due to the darker images with lower contrast. At the same time, these datasets also reveal improvements of the integration of the gyroscope.

	MH1	MH2	MH3	MH4	MH5
stereo visual	0.13	0.20	0.49	2.62	1.20
stereo visual-inertial	0.18	0.11	0.31	0.33	0.41

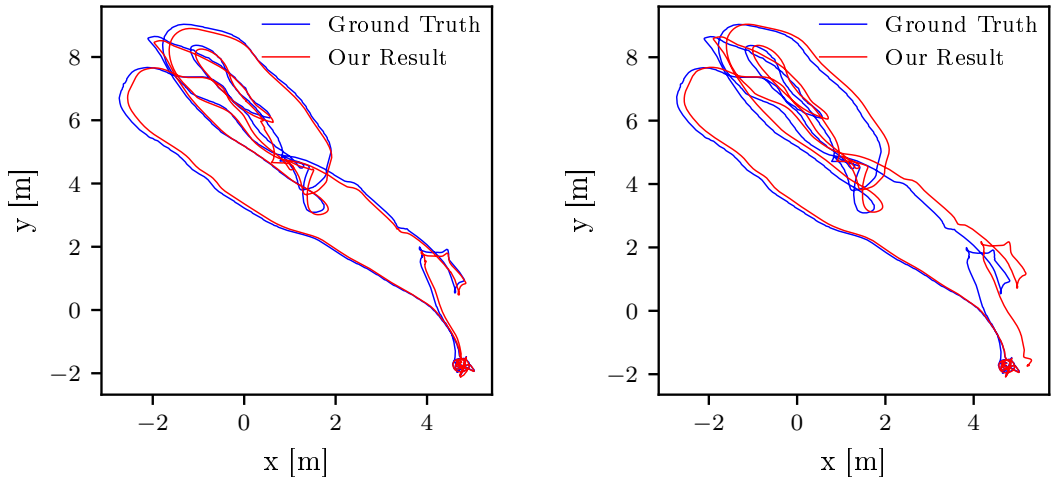
Table 7.3: ATE values for the results of the EuRoC machine hall datasets in meter.

On the machine hall datasets both algorithms were able to track the complete trajectories of the UAV. The reason for this is that the feature tracking and matching of these datasets is easier since the optical flow is in general slower than in case of the Vicon room datasets. In Table 7.3 the ATE values for the machine hall datasets are shown. For both algorithms the ATE values rise with the degree of difficulty of the datasets. Except for MH1 the ATE values improved by integrating the gyroscope data into the visual odometry algorithm. The slight deterioration might result from a wrongly estimated initial gyroscope bias value. The reconstructed trajectories for MH1 can be seen in Figure 7.4. Since the ground truth data does not provide orientation information, the first 1100 absolute sensor poses were used to compute a transformation that align both trajectories by using Horns method. It is possible to observe the drift of the computed trajectory by aligning the ground truth and the computed trajectory with this transformation. In case of Figure 7.4 it can be noticed that the accumulated drift for the trajectory of the stereo visual-inertial algorithm is much greater than for the one from the stereo visual odometry algorithm.

For all other datasets, the ATE values improved due to the incorporation of the gyroscope into visual odometry algorithm. Especially, the more difficult datasets show strong improvements. Here the stereo visual odometry algorithm has problems in case of difficult lightning conditions, which causes a drop in the amount of image correspondences. The incorporation of the gyroscope helps the algorithm to compute better rotation estimates which also improves the solutions for the translation estimation. A good example for this is dataset MH4. The visualization of the reconstructed trajectories in Figure 7.5a was generated with the same method as for Figure 7.4. In this visualization a green dotted circle marks the area with only a few image correspondences. The quality of the reconstructed trajectory in this area is much less accurate than in other parts of the reconstruction. In comparison, the corresponding section of the reconstructed trajectory in Figure 7.5b is much more similar to the ground truth trajectory.

7.5 Runtime Evaluation

The runtime evaluation was conducted on a consumer notebook with a Intel® Core™ i7-4850HQ 2.30GHz CPU and 16 Gb random access memory. Of the self-



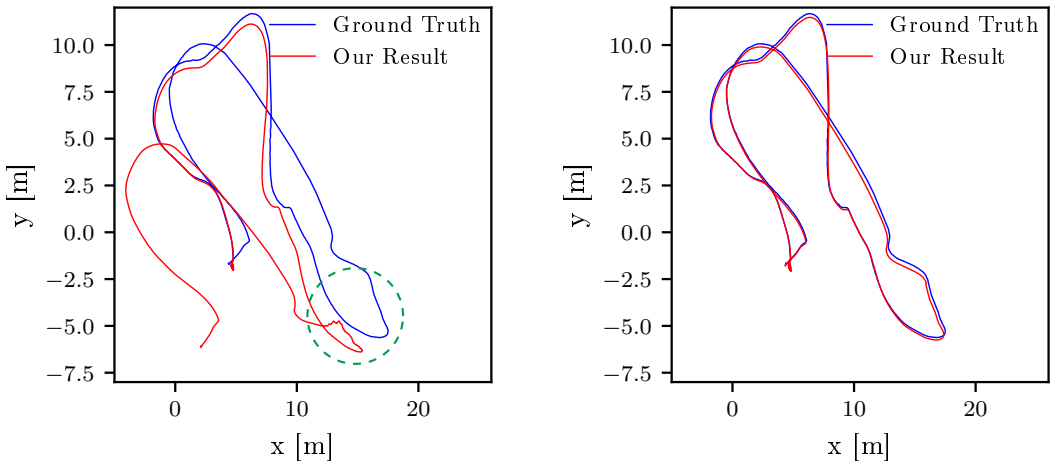
(a) MH1 stereo visual odometry result **(b)** MH1 stereo visual-inertial odometry result

Figure 7.4: Alignment of the reconstructed trajectory with the ground truth trajectory in the xy-plane for dataset EuRoC MH1. It can be noticed that the reconstructed trajectory of the stereo visual-inertial odometry algorithm accumulates more drift.

written software only the feature matching is parallelized. However, several used function calls of the OpenCV library like the 5-point RANSAC algorithm are internally parallelized. Also Ceres-Solver used during the optimization procedure 8 threads.

In order to evaluate the runtime of the algorithm the datasets KITTI 00 and Vicon room 1 were chosen. On these datasets the stereo visual-inertial odometry was run and the processing times for key point matching, key point triangulation, pose initialization and pose refinement were measured. The results of these measurements are shown in Figure 7.6.

By comparing the execution times it reveals that the total processing time for the KITTI 00 dataset is longer than on EuRoC V1-1. The main reason for this time intensity is that the method creates nearly twice as many image correspondences (on average 1206 image correspondences) than EuRoC V1-1 (on average 655 image correspondences). The different amount of image correspondences result from the matching settings which were chosen differently due to the different characteristics of the datasets. As a result, the times of key point matching, pose initialization, and pose refinement are higher for KITTI 00 than for EuRoC V1-1. During the nonlinear optimization more cost functions that make use of the epipolar constraint are used. They are expensive to compute and cause high run times. Additionally, the computation time increases also through the high number of state updates that



(a) MH4 stereo visual odometry result **(b)** MH4 stereo visual-inertial odometry result

Figure 7.5: Alignment of the reconstructed trajectory with the ground truth trajectory in the xy-plane for dataset EuRoC MH4. The stereo visual odometry algorithm accumulates more drift. It can also be noted that it has problems to reconstruct the trajectory in the area marked with the green dashed circle. In this area bad light conditions are present. However, the stereo visual-inertial algorithm is able to reconstruct the trajectory precisely.

were needed during the nonlinear optimization. Only the triangulation is faster for KITTI 00 than for EuRoC V1-1. The optimal triangulation method needs much less time for the rectified images from KITTI 00 than for the those from EuRoC V1-01.

The time spent for key point matching, triangulation, and pose initialization are for the stereo visual and the stereo visual-inertial odometry algorithm on the EuRoC V1-1 dataset similar. This meets common expectations, since both test runs used the same parameters. The pose refinement of the stereo visual-inertial algorithm is 6 ms faster than for the stereo visual algorithm. During the preintegration of the gyroscope measurements more computational effort is needed by the visual-inertial algorithm. However, less state updates during the nonlinear optimization are needed which saves time.

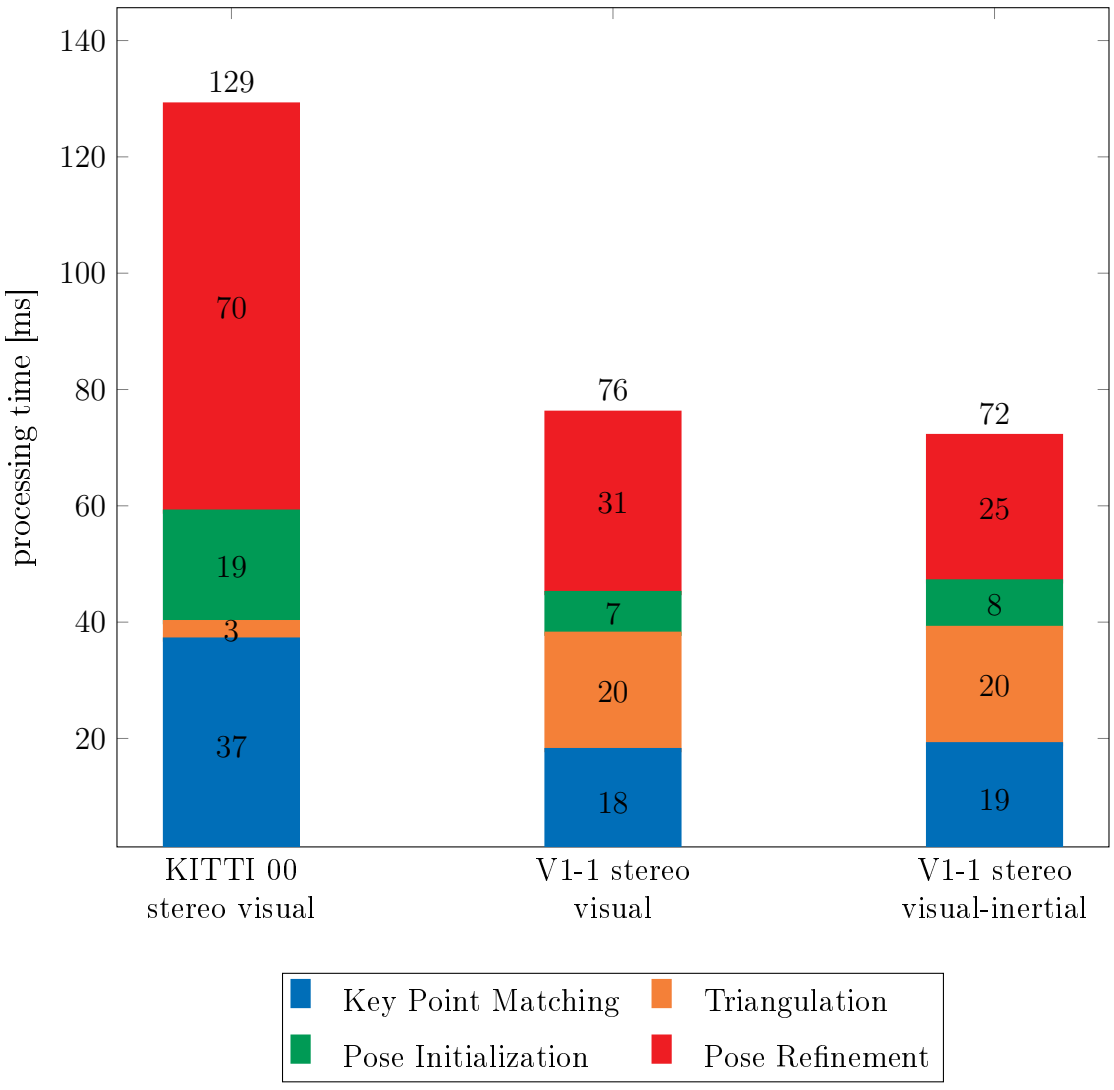


Figure 7.6: Stacked bar chart of the algorithm processing time. The stereo visual algorithm runs slower on the KITTI 00 than on the EuRoC V1-1 dataset due to more image correspondences. On the EuRoC V1-1 dataset the stereo visual-inertial algorithm is 4 ms faster than the stereo visual algorithm.

Chapter 8

Conclusion

The thesis at hand presented an adaption of Zhang's algorithm to a stereo visual-inertial odometry algorithm. As one of the main goals, this algorithm was integrated into the Aerostack framework which is used to control UAVs.

The first chapter of the thesis started by giving an overview to stereo visual inertial odometry in application with UAVs. After that, different solutions to the single subproblems used by state-of-the-art stereo visual-inertial odometry algorithms were given. In the following chapter the camera model and the epipolar geometry which describes the geometric relations between two cameras were explained. Based on the epipolar geometry the optimal triangulation method which enables a triangulation of 3D points from 2D point correspondences between two images was introduced.

The following chapter addressed the search of corresponding image features between two images. For the detection and description of salient image features the FAST corner detector and the rotated BRIEF key point descriptor were introduced. In order to find key point matches between the left and the right stereo image a constrained matching approach based on the epipolar geometry was derived. This approach was not used to derive image correspondences between two successive frames however, because the KLT-tracker yielded more accurate correspondences.

The next chapter discussed the motion estimation based on image features for stereo visual odometry. In the first section concepts of nonlinear optimization were introduced on which the motion estimation in this thesis is based. The next section presented to the best of our knowledge a new method to initialize the motion estimation with a metric correct scaled translation. Subsequently, cost functions that are used to quantify how good the estimated motion fits to detected feature correspondences in the nonlinear optimization problem were defined. For additional robustness against outliers, the Tukey loss function was applied to suppress incorrect correspondences.

The previously defined stereo visual motion estimation allows an extension to a stereo visual-inertial motion estimation which was described in the following chapter. The approach chosen here did this by tightly integrating gyroscope measurements over time. This integration was done by making use of the properties of the special orthogonal group SO(3). Its property of being a manifold played an important role during the definition of the cost functions of the gyroscope.

Based on these preliminary explanations the whole stereo visual-inertial odometry algorithm was illustrated in the subsequent chapter. After that the concepts of the Aerostack framework for controlling UAVs were presented. This was complemented by a description of how the developed algorithm was integrated into Aerostack.

The presented visual odometry algorithm was evaluated on two KITTI datasets which allowed a coarse assessment of its precision in comparison to other algorithms. Since these datasets did not provide IMU measurements the comparison of the stereo visual and the stereo visual-inertial algorithm was performed on EuRoC datasets. With these tests it was shown that the integration of the gyroscope into the stereo visual odometry algorithm has positive effects on the runtime and the accuracy of trajectory reconstruction.

There are several improvements of the stereo visual-inertial odometry algorithm possible. One approach would be to improve the feature matching and tracking. This could be done by integrating a more complex matching strategy into the algorithm like circular matching. It could be applied separately to different feature types like blobs and key points in order to match them, similar to what is suggested by Cvišić et. al. [CP15]. IMU data could also be integrated into the process in order to support matching and tracking by reducing the search space.

The motion estimation itself could be improved by integrating a windowed bundle adjustment. This would require the selection of key frames on which bundle adjustment is applied. It should reduce the drift over the whole trajectory and increase the accuracy of the reconstructed point cloud.

Furthermore, the integration of the IMU could be extended by making use of the accelerometer. The preintegration as it was presented for the gyroscope measurements can be done in a similar manner with an accelerometer. Consequently, the position and the velocity of the visual-inertial sensor unit would be estimated by the IMU as well. In case of the integration of the accelerometer it would also be interesting to determine whether its predicted change in position can be used to initialize the motion estimation together with the preintegrated gyroscope measurements.

Appendix A

KITTI Odometry Benchmark

The KITTI odometry benchmark gives an overview of the performance of current odometry algorithms in general. It contains visual monocular and stereo visual odometry methods as well as laser based odometry methods. However, here for comparison only an over view of the visual methods is given. The datasets from which the benchmark list was generated is not available for public. In order to evaluate an algorithm on these datasets it must be uploaded on an evaluation server. Since this thesis does not focus on the evaluation on KITTI datasets this was not done. Hence the presented algorithm is not listed in the leader board.

Column description:

- Method: Contains the name of the used algorithm or method.
- Translation: The translation error, which was defined in section 7.2.
- Rotation: The rotation error, which was defined in section 7.2.
- Runtime: Runtime of the algorithm in seconds and additional information about the number of used CPU cores. If the algorithm works on a graphics card this is indicated by the abbreviation GPU.

The table was downloaded on 21st of March 2017.

	Method	**Translation**	**Rotation**	**Runtime**
1	SOFT2	0.81 %	0.0022 [deg/m]	0.1 s / 2 cores
2	LSLAM	0.82 %	0.0020 [deg/m]	0.2 s / 4 cores
3	GDVO	0.86 %	0.0031 [deg/m]	0.09 s / 1 core
4	HypERROCC	0.88 %	0.0027 [deg/m]	0.25 s / 2 cores
5	SOFT	0.88 %	0.0022 [deg/m]	0.1 s / 2 cores
6	RotRocc	0.88 %	0.0025 [deg/m]	0.3 s / 2 cores
7	EDVO	0.89 %	0.0030 [deg/m]	0.1 s / 1 core

	Method	Translation	Rotation	Runtime
8	svo2	0.94 %	0.0021 [deg/m]	0.2 s / 1 core
9	ROCC	0.98 %	0.0028 [deg/m]	0.3 s / 2 cores
10	cv4xv1-sc	1.09 %	0.0029 [deg/m]	0.145 s / GPU
11	MonoROCC	1.11 %	0.0028 [deg/m]	1 s / 2 cores
12	ORB-SLAM2	1.15 %	0.0027 [deg/m]	0.06 s / 2 cores
13	svo	1.16 %	0.0030 [deg/m]	0.1 s / 2 core
14	NOTF	1.17 %	0.0035 [deg/m]	0.45 s / 1 core
15	S-PTAM	1.19 %	0.0025 [deg/m]	0.03 s / 4 cores
16	S-LSD-SLAM	1.20 %	0.0033 [deg/m]	0.07 s / 1 core
17	VoBa	1.22 %	0.0029 [deg/m]	0.1 s / 1 core
18	CBSLAM	1.24 %	0.0029 [deg/m]	0.04 s / 1 cores
19	SLUP	1.25 %	0.0041 [deg/m]	0.17 s / 4 cores
20	FRVO	1.26 %	0.0038 [deg/m]	0.03 s / 1 core
21	MFI	1.30 %	0.0030 [deg/m]	0.1 s / 1 core
22	TLBBA	1.36 %	0.0038 [deg/m]	0.1 s / 1 Core
23	2FO-CC	1.37 %	0.0035 [deg/m]	0.1 s / 1 core
24	StereoSFM	1.51 %	0.0042 [deg/m]	0.02 s / 2 cores
25	SSLAM	1.57 %	0.0044 [deg/m]	0.5 s / 8 cores
26	eVO	1.76 %	0.0036 [deg/m]	0.05 s / 2 cores
27	Stereo DWO	1.76 %	0.0026 [deg/m]	0.1 s / 4 cores
28	D6DVO	2.04 %	0.0051 [deg/m]	0.03 s / 1 core
29	SSLAM-HR	2.14 %	0.0059 [deg/m]	0.5 s / 8 cores
30	MEVO	2.40 %	0.0060 [deg/m]	1 s / 1 core
31	VISO2-S	2.44 %	0.0114 [deg/m]	0.05 s / 1 core
32	GT_VO3pt	2.54 %	0.0078 [deg/m]	1.26 s / 1 core
33	BoofCV-SQ3	2.54 %	0.0073 [deg/m]	0.14 s / 1 core
34	VO3pt	2.69 %	0.0068 [deg/m]	0.56 s / 1 core
35	TGVO	2.94 %	0.0077 [deg/m]	0.06 s / 1 core
36	SGPVO	3.11 %	0.0097 [deg/m]	0.2 s / 1 core
37	VO3ptLBA	3.13 %	0.0104 [deg/m]	0.57 s / 1 core
38	PLSVO	3.26 %	0.0095 [deg/m]	0.20 s / 2 cores
39	AvgSLAM	3.26 %	0.0101 [deg/m]	0.1 s / 4 cores
40	CFORB	3.73 %	0.0107 [deg/m]	0.9 s / 8 cores
41	VOFS	3.94 %	0.0099 [deg/m]	0.51 s / 1 core
42	VOFSLBA	4.17 %	0.0112 [deg/m]	0.52 s / 1 core
43	IO	6.55 %	0.0315 [deg/m]	5 s / 1 core
44	ST	97.70 %	0.2710 [deg/m]	0.2 s / 8 cores

Appendix B

List of Tables

Appendix C

List of Figures

Appendix D

Bibliography

[ABH+09] Achtelik, Markus ; Bachrach, Abraham ; He, Ruijie ; Prentice, Samuel ; Roy, Nicholas: Stereo vision and laser odometry for autonomous helicopters in GPS-denied indoor environments. In: *Proceedings of SPIE The International Society for Optical Engineering* 1 (2009), S. 733219–733219–10

[AD15] Abeywardena, Dinuka ; Dissanayake, Gamini: Tightly-coupled model aided visual-inertial fusion for quadrotor micro air vehicles. In: *Field and Service Robotics* Springer, 2015, S. 153–166

[AMO17] Agarwal, Sameer ; Mierle, Keir ; Others: *Ceres Solver*. http://ceres-solver.org, 2017. – Version 1.12

[BKZ+15] Beul, Marius ; Krombach, Nicola ; Zhong, Yongfeng ; Droeschel, David ; Nieuwenhuisen, Matthias ; Behnke, Sven: A high-performance MAV for autonomous navigation in complex 3D environments. In: *Unmanned Aircraft Systems (ICUAS), 2015 International Conference on* IEEE, 2015, S. 1241–1250

[BNG+16] Burri, Michael ; Nikolic, Janosch ; Gohl, Pascal ; Schneider, Thomas ; Rehder, Joern ; Omari, Sammy ; Achtelik, Markus W. ; Siegwart, Roland: The EuRoC micro aerial vehicle datasets. In: *The International Journal of Robotics Research* (2016)

[Bou01] Bouguet, Jean-Yves: Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. In: *Intel Corporation* 5 (2001), Nr. 1-10, S. 4

[BTVG06] Bay, Herbert ; Tuytelaars, Tinne ; Van Gool, Luc: Surf: Speeded up robust features. In: *European conference on computer vision* Springer, 2006, S. 404–417

[BW16a] Buczko, Martin ; Willert, Volker: Flow-decoupled normalized reprojection error for visual odometry. In: *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on* IEEE, 2016, S. 1161–1167

[BW16b] Buczko, Martin ; Willert, Volker: How to distinguish inliers from outliers in visual odometry for high-speed automotive applications. In: *2016 IEEE Intelligent Vehicles Symposium (IV)* IEEE, 2016, S. 478–483

[CLSF10] Calonder, Michael ; Lepetit, Vincent ; Strecha, Christoph ; Fua, Pascal: Brief: Binary robust independent elementary features. In: *European conference on computer vision* Springer, 2010, S. 778–792

[CP15] Cvišić, Igor ; Petrović, Ivan: Stereo odometry based on careful feature selection and tracking. In: *Mobile Robots (ECMR), 2015 European Conference on* IEEE, 2015, S. 1–6

[Cra06] Crassidis, John L.: Sigma-point Kalman filtering for integrated GPS and inertial navigation. In: *IEEE Transactions on Aerospace and Electronic Systems* 42 (2006), Nr. 2, S. 750–756

[EGH16] Eckenhoff, Kevin ; Geneva, Patrick ; Huang, Guoquan: High-accuracy preintegration for visual-inertial navigation / University of Delaware. 2016 (RPNG-2016-001). – Research report

[EKC16] Engel, J. ; Koltun, V. ; Cremers, D.: Direct Sparse Odometry. (2016), July

[ESC14] Engel, Jakob ; Schöps, Thomas ; Cremers, Daniel: LSD-SLAM: Large-scale direct monocular SLAM. In: *European Conference on Computer Vision* Springer, 2014, S. 834–849

[ESWS11] Eberli, Daniel ; Scaramuzza, Davide ; Weiss, Stephan ; Siegwart, Roland: Vision based position control for MAVs using one single circular landmark. In: *Journal of Intelligent & Robotic Systems* 61 (2011), Nr. 1-4, S. 495–512

[FCC15] Fu, Changhong ; Carrio, Adrian ; Campoy, Pascual: Efficient visual odometry and mapping for Unmanned Aerial Vehicle using ARM-based stereo vision pre-processing system. In: *2015 International Conference on Unmanned Aircraft Systems, ICUAS 2015*, 2015, S. 957–962

[FCDS15a] Forster, Christian ; Carlone, Luca ; Dellaert, Frank ; Scaramuzza, Davide: IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. In: *Proceedings of Robotics: Science and Systems* Georgia Institute of Technology, 2015, S. 6–15

[FCDS15b] Forster, Christian ; Carlone, Luca ; Dellaert, Frank ; Scaramuzza, Davide: On-manifold preintegration theory for fast and accurate visual-inertial navigation. In: *IEEE Transactions on Robotics* (2015), S. 1–18

[FJQBMT13] Fei, Wang ; Jin-Qiang, CUI ; Ben-Mei, CHEN ; Tong, H L.: A comprehensive UAV indoor navigation system based on vision optical flow and laser FastSLAM. In: *Acta Automatica Sinica* 39 (2013), Nr. 11, S. 1889–1899

[FPS14] Forster, Christian ; Pizzoli, Matia ; Scaramuzza, Davide: SVO: Fast semi-direct monocular visual odometry. In: *Proceedings - IEEE International Conference on Robotics and Automation*, 2014, S. 15–22

[GJSMCMJ14] Garrido-Jurado, S. ; Salinas, R. M. ; Madrid-Cuevas, F.J. ; Marín-Jiménez, M.J.: Automatic generation and detection of highly reliable fiducial markers under occlusion. In: *Pattern Recognition* 47 (2014), Nr. 6, S. 2280 – 2292

[GLU12] Geiger, Andreas ; Lenz, Philip ; Urtasun, Raquel: Are we ready for autonomous driving? the KITTI vision benchmark suite. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2012, S. 3354–3361

[GOBGJ16] Gomez-Ojeda, Ruben ; Briales, Jesus ; González-Jiménez, Javier: PL-SVO: Semi-direct monocular visual odometry by combining points and line segments. In: *Int. Conf. on Intelligent Robots and Systems (IROS)* IEEE/RSJ, 2016, S. 4211–4216

[GZS11] Geiger, Andreas ; Ziegler, Julius ; Stiller, Christoph: StereoScan: Dense 3d reconstruction in real-time. In: *IEEE Intelligent Vehicles Symposium, Proceedings*, 2011, S. 963–968

[HFB16] Holzmann, Thomas ; Fraundorfer, Friedrich ; Bischof, Horst: Direct stereo visual odometry based on lines. In: *Proceedings of the 11th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, 2016*, 2016, S. 1–11

[HMTP13] Honegger, Dominik ; Meier, Lorenz ; Tanskanen, Petri ; Pollefeys, Marc: An open source and open hardware embedded metric optical flow cmos camera for indoor and outdoor applications. In: *Robotics and Automation (ICRA), 2013 IEEE International Conference on* IEEE, 2013, S. 1736–1741

[Hor87] Horn, Berthold K.: Closed-form solution of absolute orientation using unit quaternions. In: *JOSA A* 4 (1987), Nr. 4, S. 629–642

[HS88] Harris, Chris ; Stephens, Mike: A combined corner and edge detector. In: *Alvey vision conference* Bd. 15 Citeseer, 1988, S. 10–5244

[HS97] Hartley, Richard I. ; Sturm, Peter: Triangulation. In: *Computer vision and image understanding* 68 (1997), Nr. 2, S. 146–157

[HZ03] Hartley, Richard ; Zisserman, Andrew: *Multiple view geometry in computer vision*. Cambridge university press, 2003

[IA99] Irani, Michal ; Anandan, P: About direct methods. In: *International Workshop on Vision Algorithms* Springer, 1999, S. 267–277

[KDB17] Krombach, Nicola ; Droeschel, David ; Behnke, Sven: Combining Feature-Based and Direct Methods for Semi-dense Real-Time Stereo Visual Odometry. In: *Intelligent Autonomous Systems 14: Proceedings of the 14th International Conference IAS-14*. Cham : Springer International Publishing, 2017, S. 855–868

[KESL17] Kasyanov, Anton ; Engelmann, Francis ; Stückler, Jörg ; Leibe, Bastian: Keyframe-based visual-inertial online SLAM with relocalization. In: *CoRR* abs/1702.02175 (2017)

[KGL10] Kitt, Bernd ; Geiger, Andreas ; Lategahn, Henning: Visual odometry based on stereo image sequences with RANSAC-based

outlier rejection scheme. In: *IEEE Intelligent Vehicles Symposium, Proceedings*, 2010, S. 486–492

[KJR+12] Kaess, M. ; Johannsson, H. ; Roberts, R. ; Ila, V. ; Leonard, J. J. ; Dellaert, F.: iSAM2: Incremental smoothing and mapping using the Bayes tree. In: *The International Journal of Robotics Research* 31 (2012), Nr. 2, S. 216–235

[KNS+14] Klingbeil, Lasse ; Nieuwenhuisen, Matthias ; Schneider, Johannes ; Eling, Christian ; Droeschel, David ; Holz, Dirk ; Läbe, Thomas ; Förstner, Wolfgang ; Behnke, Sven ; Kuhlmann, Heiner: Towards autonomous navigation of an UAV-based mobile mapping system. In: *Proc. of Int. Conf. on Machine Control & Guidance (MCG)*, 2014, S. 136–147

[KRD08] Kaess, Michael ; Ranganathan, Ananth ; Dellaert, Frank: iSAM: Incremental smoothing and mapping. In: *IEEE Transactions on Robotics* 24 (2008), Nr. 6, S. 1365–1378

[KSS08] Kelly, Jonathan ; Saripalli, Srikanth ; Sukhatme, Gaurav S.: Combined visual and inertial navigation for an unmanned aerial vehicle. In: *Springer Tracts in Advanced Robotics* Bd. 42, 2008, S. 255–264

[LDIG11] Lovegrove, Steven ; Davison, Andrew J. ; Ibanez-Guzmán, Javier: Accurate visual odometry from a rear parking camera. In: *Intelligent Vehicles Symposium (IV), 2011 IEEE* IEEE, 2011, S. 788–793

[LH06] Li, Hongdong ; Hartley, Richard: Five-point motion estimation made easy. In: *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on* Bd. 1 IEEE, 2006, S. 630–633

[LK81] Lucas, Bruce D. ; Kanade, Takeo: An Iterative Image Registration Technique with an Application to Stereo Vision. In: *Proceedings of the 7th International Joint Conference on Artificial Intelligence, IJCAI '81, Vancouver, BC, Canada, August 24-28, 1981*, 1981, S. 674–679

[LM13] Li, Mingyang ; Mourikis, Anastasios I.: High-precision, consistent EKF-based visual-inertial odometry. In: *The International Journal of Robotics Research* 32 (2013), Nr. 6, S. 690–711

[Low99] Lowe, David G.: Object recognition from local scale-invariant features. In: *Computer vision, 1999. The proceedings of the seventh IEEE international conference on* Bd. 2 Ieee, 1999, S. 1150–1157

[MAMT15] Mur-Artal, Raul ; Montiel, Jose Maria M. ; Tardos, Juan D.: ORB-SLAM: a versatile and accurate monocular SLAM system. In: *IEEE Transactions on Robotics* 31 (2015), Nr. 5, S. 1147–1163

[MAT16] Mur-Artal, Raul ; Tardos, Juan D.: ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. In: *arXiv preprint arXiv:1610.06475* (2016)

[Nis04] Nistér, David: An efficient solution to the five-point relative pose problem. In: *IEEE transactions on pattern analysis and machine intelligence* 26 (2004), Nr. 6, S. 756–770

[NLD11] Newcombe, Richard A. ; Lovegrove, Steven J. ; Davison, Andrew J.: DTAM: Dense tracking and mapping in real-time. In: *Computer Vision (ICCV), 2011 IEEE International Conference on* IEEE, 2011, S. 2320–2327

[NNB04] Nistér, David ; Naroditsky, Oleg ; Bergen, James: Visual odometry. In: *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on* Bd. 1 Ieee, 2004, S. I–I

[NRB+14] Nikolic, Janosch ; Rehder, Joern ; Burri, Michael ; Gohl, Pascal ; Leutenegger, Stefan ; Furgale, Paul T. ; Siegwart, Roland: A synchronized visual-inertial sensor system with FPGA pre-processing for accurate real-time SLAM. In: *Robotics and Automation (ICRA), 2014 IEEE International Conference on* IEEE, 2014, S. 431–437

[PPFM15] Persson, Mikael ; Piccini, Tommaso ; Felsberg, Michael ; Mester, Rudolf: Robust stereo visual odometry from monocular techniques. In: *Intelligent Vehicles Symposium (IV), 2015 IEEE* IEEE, 2015, S. 686–691

[PSLDLP+16] Pestana, Jesús ; Sanchez-Lopez, Jose L. ; De La Puente, Paloma ; Carrio, Adrian ; Campoy, Pascual: A vision-based quadrotor multi-robot solution for the indoor autonomy challenge of the 2013 international micro air vehicle competition. In: *Journal of Intelligent & Robotic Systems* 84 (2016), Nr. 1-4, S. 601–620

[RD06] Rosten, Edward ; Drummond, Tom: Machine learning for high-speed corner detection. In: *European conference on computer vision* Springer, 2006, S. 430–443

[RPD10] Rosten, Edward ; Porter, Reid ; Drummond, Tom: Faster and better: A machine learning approach to corner detection. In: *IEEE transactions on pattern analysis and machine intelligence* 32 (2010), Nr. 1, S. 105–119

[RRKB11] Rublee, Ethan ; Rabaud, Vincent ; Konolige, Kurt ; Bradski, Gary: ORB: An efficient alternative to SIFT or SURF. In: *Computer Vision (ICCV), 2011 IEEE International Conference on* IEEE, 2011, S. 2564–2571

[SCG13] Song, Shiyu ; Chandraker, Manmohan ; Guest, Clark C.: Parallel, real-time monocular visual odometry. In: *Robotics and Automation (ICRA), 2013 IEEE International Conference on* IEEE, 2013, S. 4698–4705

[SEE+12] Sturm, Jürgen ; Engelhard, Nikolas ; Endres, Felix ; Burgard, Wolfram ; Cremers, Daniel: A benchmark for the evaluation of RGB-D SLAM systems. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems* IEEE, 2012, S. 573–580

[SF11] Scaramuzza, Davide ; Fraundorfer, Friedrich: Visual odometry : Part I: The first 30 years and fundamentals. In: *IEEE Robotics & Automation Magazine* 18 (2011), Nr. 4, S. 80–92

[SLFB+16] Sanchez-Lopez, Jose L. ; Fernández, Ramón A S. ; Bavle, Hriday ; Sampedro, Carlos ; Molina, Martin ; Pestana, Jesus ; Campoy, Pascual: Aerostack: An architecture and open-source software framework for aerial robotics. In: *Unmanned Aircraft Systems (ICUAS), 2016 International Conference on* IEEE, 2016, S. 332–341

[SMR07] Silveira, Geraldo ; Malis, Ezio ; Rives, Patrick: An efficient direct method for improving visual SLAM. In: *Robotics and Automation, 2007 IEEE International Conference on* IEEE, 2007, S. 4090–4095

[SS02] Strelow, Dennis ; Singh, Sanjiv: Optimal motion estimation from visual and inertial measurements. In: *Applications of Com-*

puter Vision, 2002.(WACV 2002). Proceedings. Sixth IEEE Workshop on IEEE, 2002, S. 314–319

[SSA13] Sirtkaya, Salim ; Seymen, Burak ; Alatan, A A.: Loosely coupled kalman filtering for fusion of visual odometry and inertial navigation. In: *Information Fusion (FUSION), 2013 16th International Conference on* IEEE, 2013, S. 219–226

[ST94] Shi, Jianbo ; Tomasi, C.: Good features to track. In: *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1994, S. 593–600

[Sze10] Szeliski, Richard: *Computer vision : Algorithms and applications*. London; New York : Springer Science & Business Media, 2010

[TGL+10] Tardif, Jean-Philippe ; George, Michael ; Laverne, Michel ; Kelly, Alonzo ; Stentz, Anthony: A new approach to vision-aided inertial navigation. In: *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on* IEEE, 2010, S. 4161–4168

[TK91] Tomasi, Carlo ; Kanade, Takeo: Detection and tracking of point features. School of Computer Science, Carnegie Mellon Univ. Pittsburgh, 1991 (CMU-CS-91-132). – Research report

[TMHF99] Triggs, Bill ; McLauchlan, Philip F. ; Hartley, Richard I. ; Fitzgibbon, Andrew W.: Bundle adjustment—a modern synthesis. In: *International workshop on vision algorithms* Springer, 1999, S. 298–372

[UESC16] Usenko, Vladyslav ; Engel, Jakob ; Stückler, Jörg ; Cremers, Daniel: Direct visual-inertial odometry with stereo cameras. In: *Int. Conf. on Robotics and Automation*, 2016, S. 1885–1892

[WO16] Wang, John ; Olson, Edwin: AprilTag 2: Efficient and robust fiducial detection. In: *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on* IEEE, 2016, S. 4193–4198

[Woo07] Woodman, Oliver J.: An introduction to inertial navigation / University of Cambridge, Computer Laboratory. 2007 (UCAM-CL-TR-696). – Research report

[WW13] Witt, Jonas ; Weltin, Uwe: Robust stereo visual odometry using iterative closest multiple lines. In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, November 3-7, 2013*, IEEE, 2013, S. 4164–4171

[YHGD14] Yanhua, Jiang ; Huiyan, Chen ; Guangming, Xiong ; Davide, Scaramuzza: ICP stereo visual odometry for wheeled vehicles based on a 1DOF motion prior. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2014, S. 585 – 592

[ZDFL95] Zhang, Zhengyou ; Deriche, Rachid ; Faugeras, Olivier ; Luong, Quang-Tuan: A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. In: *Artificial intelligence* 78 (1995), Nr. 1-2, S. 87–119

[ZKS14] Zhang, Ji ; Kaess, Michael ; Singh, Sanjiv: Real-time depth enhanced monocular odometry. In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems* IEEE, 2014, S. 4973–4980