SOFT-SLAM: Computationally Efficient Stereo Visual SLAM for Autonomous UAVs

Igor Cvišić
University of Zagreb
Faculty of Electrical Engineering and Computing
HR-10000, Zagreb, Croatia
igor.cvisic@fer.hr

Josip Ćesić
University of Zagreb
Faculty of Electrical Engineering and Computing
HR-10000, Zagreb, Croatia
josip.cesic@fer.hr

Ivan Marković
University of Zagreb
Faculty of Electrical Engineering and Computing
HR-10000, Zagreb, Croatia
ivan.markovic@fer.hr

Ivan Petrović
University of Zagreb
Faculty of Electrical Engineering and Computing
HR-10000, Zagreb, Croatia
ivan.petrovic@fer.hr

克罗地亚

Abstract

Autonomous navigation of unmanned aerial vehicles (UAVs) in GPS-denied environments is a challenging problem, especially for small-scale UAVs characterized by small payload and limited battery autonomy. Possible solution to the aforementioned problem is vision-based simultaneous localization and mapping (SLAM), since cameras, due to their dimensions, low weight, availability, and large information bandwidth, circumvent all the constraints of UAVs. In this paper we propose a stereo vision SLAM yielding very accurate localization and a dense map of the environment developed with the aim to compete in the European Robotics Challenges (EuRoC) targeting airborne inspection of industrial facilities with small-scale UAVs. The proposed approach consists of a novel stereo odometry algorithm relying on feature tracking (SOFT), which is currently ranking first among all stereo methods on the KITTI dataset. Relying on SOFT for pose estimation, we build a feature-based pose graph SLAM solution, which we dub SOFT-SLAM. SOFT-SLAM has a completely separated odometry and mapping threads supporting large loop-closing and global consistency. It

提出了一整个我SLAM

1. Soft odometry
2. feature-based pose graph

该slam框架分为两个线程
线程1 一个单独的里程计
线程2 回环检测和global consstency

also achieves a constant-time execution rate of 20 Hz with deterministic results using only two threads of an onboard computer used in the challenge. The UAV running our SLAM algorithm obtained the highest localization score in the EuRoC Challenge 3, Stage IIa–Benchmarking, Task 2. Furthermore, we also present an exhaustive evaluation of SOFT-SLAM on two popular public datasets and compare it to other state-of-the-art approaches; namely, ORB-SLAM2 and LSD-SLAM. The results show that SOFT-SLAM obtains better localization accuracy on the majority of datasets sequences, while also having lower runtime.

1 Introduction

Autonomous unmanned aerial vehicles (UAVs) are receiving significant attention in many communities, including academia, industry, and consumer electronics. They are being used in a large spectrum of applications from search-and-rescue operations, surveillance and border control, industrial plant inspection, agriculture, construction, filmmaking, and recreational use. This is not surprising since they offer unparalleled capabilities in terms of ground coverage and maneuverability in comparison to land-based robots. However, these advantages come with caveats, especially for small-scale and high-speed UAVs in terms of battery autonomy and payload, thus usually requiring light-weight navigation sensors like cameras, especially in GPS-denied environments. In general, there are three types of UAVs with respect to the construction and flying principles: 1) fixed-winged UAVs whose wing surface generates the major lift; usually have higher payload and can cover larger distances, but cannot hover in one spot and provide precise sensor positioning, 2) rotary-wing UAVs that derive lift from the rotary-wing system, not much unlike the helicopter; they have the capabilities for precise positioning, but the propulsion system requires more complicated control and maintenance, and 3) multirotor UAVs that have the same capabilities as the rotary-wing, but are more stable and easier to control, although with somewhat shorter flight time and ranges. Regardless of the UAV application and type, one of the prerequisites for their autonomous operation is the challenging real-time localization with the available on-board sensors and limited computational resources.

The answer to the aforementioned challenge lies in simultaneous localization and mapping (SLAM), a subject which has been researched in the field of mobile robotics and autonomous systems for several decades now (Bailey and Durrant-Whyte, 2006a; Bailey and Durrant-Whyte, 2006b). SLAM consists of simultaneously estimating the state of the robot and the map of the environment. For the case of UAVs, the state is usually a 6D pose, although some other quantities, like velocities and sensor biases, can also be included. The map

of the environment depends on the used sensors and application requirements; they can be sparse consisting only of specific features, e.g., points, lines and corners, or dense with very accurate and precise metric information, e.g., 3D surveying laser range scanner maps. For the navigation of autonomous UAVs, sparse maps can be adequate for state estimation; however, for the case of cluttered areas, dense maps are necessary for obstacle/hazard avoidance. Given the time that SLAM has been the focus of research community, it is tempting to ask if SLAM is solved or even if robots truly need SLAM? In a recent paper by (Cadena et al., 2016) these difficult questions have been thoroughly analyzed and the authors assert that, indeed, robots do require SLAM, and that depending on the robot/environment/performance combination there are still unsolved challenges left in the field—one being in the field of high-speed vision-based autonomous UAVs, which exhibit fast dynamics and have limited sensor and computational resources.

As described in (Cadena et al., 2016), SLAM attempts to build a globally consistent representation of the environment at the same time leveraging ego-motion, i.e., odometry, and loop closing. Since this paper is concerned with vision-based localization for UAVs, we will focus mostly on vision-based approaches. Odometry is the process of estimating the robot position relative to its environment. It is a dead reckoning process, meaning that the robot determines its motion incrementally by integrating the current motion to the previously determined position. As such, it is prone to cumulative errors and the most common sensors used for land-based odometry are wheel encoders. If the surrounding scene has texture, one of the most accurate sensors for odometry is camera, and this process is then known as visual odometry (Nister et al., 2004). From two views taken from a single camera, rotation and translation up to a scale can be obtained. On the other hand, having two views from a calibrated stereo camera, both the rotation and translation between them can be exploited. A detailed survey of this topic is given in (Scaramuzza and Fraundorfer, 2011; Fraundorfer and Scaramuzza, 2012). Odometry based on motion integration is prone to drift, although recent methods achieve very accurate performance both with (Forster et al., 2016a; Usenko et al., 2016) and without inertial information (Cvišić and Petrović, 2015). This drift was the main reason behind the development of SLAM—a procedure that can correct for the odometry drift by detecting already visited places, i.e., by making loop closure. Without the loop closure part SLAM reduces to odometry, and as illustrated in (Cadena et al., 2016), in that case the robot would "interpret the world as an infinite corridor".

The most successful implementations of SLAM rely on laser scanners or (3D) cameras in static feature rich environments (Estrada et al., 2005; Kerl et al., 2013). SLAM methods can be roughly divided in *graph optimization* and *nonlinear filtering* approaches. Graph optimization approaches have received more attention in the recent years, while early nonlinear filtering approaches were based on the extended Kalman

filter (EKF). However, this is not to say that filtering approaches cannot achieve state-of-the-art performance (Mourikis and Roumeliotis, 2007). At the moment there are multiple libraries available for solving SLAM (Cadena et al., 2016), where for the present paper we use the general framework for graph optimization (g2o) by (Kümmerle et al., 2011). For a detailed survey and history of SLAM, please confer (Bailey and Durrant-Whyte, 2006a; Bailey and Durrant-Whyte, 2006b; Cadena et al., 2016). Concerning visual SLAM methods, they can also be divided in two groups: *feature-based* and *direct* methods. Feature based methods use features such as lines, edges, corners and detectors/descriptors such as Harris (Harris and Stephens, 1988), SIFT (Lowe, 1999), SURF (Bay et al., 2006), FAST (Rosten and Drummond, 2006), or ORB (Rublee et al., 2011) features to detect a sparse set of salient points from the real world and match them between the frames. Examples of sparse vision SLAM solutions with different types of features can be found in (Davison, 2003; Smith et al., 2006; Davison et al., 2007; Klein and Murray, 2007; Konolige and Agrawal, 2008; Eade and Drummond, 2009; Mur-Artal et al., 2015; Mur-Artal and Tardos, 2016; Pire et al., 2015; Leutenegger et al., 2015). Direct methods operate directly on pixel intensities. They use more information from the image then methods with features and therefore can eventually provide more accurate estimation in situations with poor lightning conditions, low texture and motion blur. Dense direct SLAM algorithms have been proposed by (Newcombe et al., 2011) and (Pizzoli et al., 2014); however, they are also computationally demanding and require GPU implementation for real-time performance. Furthermore, there are also semidense methods that exploit only image areas with strong edges (Engel et al., 2013; Engel et al., 2014; Engel et al., 2015; Usenko et al., 2016) and semi-direct methods that combine sparse features and direct methods (Forster et al., 2014; Forster et al., 2016b), both achieving real-time performance on CPUs. However, a common part of both feature based and direct methods is optimization over numerous features or pixel intensities, which may be a highly redundant procedure. Furthermore, by employing the bundle adjustment, redundancy grows exponentially. Since only three precise points are enough to compute the exact camera pose, one question naturally arises: Is it possible to select the best subset of features and compute the pose with minimal effort?

The main goal of the work presented in this paper was to develop a SLAM module yielding very accurate localization and a dense map of the environment that will be able to run in real-time on a UAV embedded computer still leaving enough processing power for the control and navigation algorithms in order to compete in Challenge 3 of the European Robotics Challenges (EuRoC). Since the provided on-board computer has 4 cores, we decided to design SLAM that will operate only on 2 cores, leaving the rest of the computational power for model predictive control based position controller, sensor fusion, and other critical tasks. Although a monocular camera and an IMU can be used for the state estimation part of SLAM, we decided to focus from the beginning on a stereo camera approach, since such a camera setup was expected at EuRoC. In this

paper we draw upon earlier work published in a conference paper (Cvišić and Petrović, 2015), i.e., the SOFT stereo visual odometry, and extend it to form a sparse pose graph SLAM solution with an environment map (SOFT-SLAM). The approach we propose is the most similar to the ORB-SLAM2—we also use the same features for odometry, relocalization and loop-closing, but with several important distinctions which are summarized in the following contributions:

1. Instead of the computationally more intensive bundle adjustment for *local localization*, i.e., local SLAM, we use SOFT visual odometry, which yields $\approx 0.8\%$ position error with respect to the traversed path and execution frequency of the proposed method of 20 Hz. Currently, SOFT ranks the highest on the KITTI dataset[1] among the visual odometry approaches[2].

2. The proposed SLAM framework, which supports large loop closings, completely decouples the odometry and mapping thread, yielding a constant runtime odometry with global consistency, i.e., an odometry that does not have to wait at any point for the mapping thread. Furthermore, unlike ORB-SLAM2, which due to non-deterministic nature of the multi-threaded implementation produces different results for the same data, SOFT-SLAM results are deterministic, i.e., it always yields equal output for the same data.

3. We use features from SOFT for loop closing that, as in the case of ORB-SLAM2, makes the system highly efficient, simple, and reliable on a sub-pixel accuracy. Even though SOFT features are not invariant to rotation and scale, we demonstrate on relevant public dataset that loop closing occurs frequently enough to achieve state-of-the-art results.

4. We perform a systematic evaluation on relevant public datasets containing indoor and outdoor environments; concretely, including UAV (EuRoC dataset) and car sequences (KITTI dataset).

Notably, we achieve better or equal camera localization accuracy on the majority of sequences from the EuRoC (Burri et al., 2016) and KITTI (Geiger et al., 2013) datasets in comparison to the other state-of-the-art approaches; namely, ORB-SLAM2 and LSD-SLAM. Furthermore, our UAV running SOFT-SLAM achieved the highest localization score on the EuRoC Challenge 3, Stage IIa–Benchmarking, Task 2.

The paper is organized as follows. Section 2 describes the related work. In Section 3 we present an overview of the SOFT-SLAM localization system. In Section 4 we present in details the odometry thread, while Section 5

[1] www.cvlibs.net/datasets/kitti/eval_odometry.php
[2] We are referring to the improved version of SOFT that has been dubbed SOFT2 in the KITTI evaluation table.

is devoted to the mapping thread. Section 6 presents exhaustive experimental results and comparison to the other state-of-the-art approaches. In the end, Section 7 concludes the paper.

2 Related work

Even though this paper is concerned with stereo visual odometry and SLAM, we still reference monocular SLAM approaches since many stereo methods rely upon them and they served as an inspiration for this paper as well. Visual odometry was introduced in the seminal paper (Nister et al., 2004) where solutions for relative motion estimation were presented for both monocular and stereo cameras. In (Davison et al., 2007) a real-time monocular SLAM was presented, dubbed MonoSLAM, based on EKF and a sparse map of high quality landmarks optimized towards enabling localization. Therein the authors used the Shi and Thomasi detector (Shi and Tomasi, 1994) asserting that such features, in comparison to SIFT, are more appropriate for continuous tracking due to their low computational cost. However, they also point out the advantages of SIFT when it comes to loop-closing with very weak priors. One of the main contributions of MonoSLAM was an active approach to feature mapping and measurement, meaning that, from the estimated feature localizations and pertinent uncertainties, a sensible strategy was devised for focusing only on the most relevant parts of images in order to decide where to search for matches. The authors also presented a general motion model for smooth 3D camera movement and a top-down solution for monocular feature initialization, where after the first measurement only a 3D semiinfinite line is set along which the feature must lie, after which only depth is estimated from future observations in a probabilistic manner. All this resulted with a sparse feature-based SLAM algorithm running at 30 Hz in a room-sized environment. Similarly to MonoSLAM, in this paper we also focus on computationally inexpensive features of high-quality, but by using IMU information to aid in the search for possible matches, thus reducing the execution time.

Another feature-based seminal work in monocular vision was presented in (Klein and Murray, 2007), called parallel tracking and mapping (PTAM) for augmented reality applications, wherein optimization for tracking and mapping was separated in a bundle adjustment framework (Briggs et al., 1999; Hartley and Zisserman, 2003). However, PTAM lacks large loop-closing and adequate handling of occlusions and is limited to small-scale operations (Mur-Artal et al., 2015). Building upon PTAM authors in (Pire et al., 2015) presented the stereo version of the algorithm. Therein, the authors also stressed the importance of the parallel framework decoupling the time-constrained pose estimation from the map building and refinement. For visual features they used the Shi and Thomasi detector coupled with the BRIEF descriptor (Calonder et al., 2010) and perform local bundle adjustment on a fixed number of queued keyframes. The optimization algorithm that

is used both in the pose estimation and bundle adjustment is the general framework for graph optimization (g2o). All this resulted in a sparse feature-based algorithm running in real-time. In this paper, we are also leveraging the PTAM idea of dividing the SLAM problem in two parallel different threads: the camera tracking, i.e., in our case the odometry to be precise, and the map optimization. However, in our case we do not even share the map between the threads, we only estimate the drift between the odometry thread and the map itself. This way we ensure constant time execution of the odometry and complete independence with respect to the mapping thread. In (Mur-Artal et al., 2015) a SLAM framework for monocular, stereo and RGBD cameras was proposed that uses the same set of features for the visual odometry, mapping, relocalization, and loop-closing. The main advantages of the used ORB features (Rublee et al., 2011) is that good performance can be obtained with respect to the algorithm complexity and feature robustness to rotation, illumination and (to a certain point) scale. The workhorse of the ORB-SLAM2 can be divided in two optimization procedures. The first procedure is the *local localization* optimization that performs bundle adjustment over both poses and features on a set of key frames, resulting with the relative displacement between the first and the last keyframe, i.e., the pose estimation. This approach is computationally very intensive and is, therefore, not used for the global optimization. The second procedure is concerned with the global consistency, i.e., the loop-closing algorithm. What ORB-SLAM2 does in this step is to select a sparse set of keyframes, the so called *essential graph*, and optimizes over them using only the poses. The result of this procedure is an updated sparse pose graph with a feature map consisting of ORB features. The optimization algorithm that is used in both procedures is g2o. The SLAM framework that we propose is similar to ORB-SLAM2—we also use the same features for odometry and loop-closing, but with several important distinctions which were elaborated in Section 1. Additionally, we also omit the mapping for localization purposes, because (i) we achieve precise localization without the mapping, hence we speed up the processing time, and (ii) for obstacle avoidance we must have a dense map anyway, thus no advantage would by gained by optimizing sparse map points—only valuable CPU power would be used. Furthermore, under the assumption that the odometry has a low position error, the search space for loop closing is constrained; therefore, we omit relocalization capability of ORB-SLAM2 using bags of words and instead implement a direct search for transformation on a small number of nearby frames.

Other recent visual odometry and SLAM approaches have concentrated on the *direct* approach where instead of focusing on salient image points, like the feature-based methods, image pixels intensities are used to infer about the camera motion in the environment. In (Engel et al., 2014) a large scale direct monocular SLAM (LSD-SLAM) was proposed, which uses direct image alignment coupled with a filtering-based estimation of semi-dense depth maps, and where the global map is represented as a pose graph consisting of keyframes

as vertices and similarity transforms as edges. The tracking part is achieved by minimizing the photometric error using a Gauss-Newton optimization in the algebra of the special Euclidean group, i.e., $\mathfrak{se}(3)$, while direct image alignment between two differently scale keyframes is performed in the algebra of the similarity group, i.e., $\mathfrak{sim}(3)$. In the end, the map is continuously optimized in the background using the g2o pose graph optimization. LSD-SLAM was extended to stereo images in (Engel et al., 2015), where authors used both baseline and multiview stereo in order to estimate the depth, and also proposed an affine lightning correction in order to make the direct approach more robust to illumination changes. More recently, the stereo LSD-SLAM has also been extended to *tightly* integrate IMU into direct image alignment with a non-linear energy minimization framework, thus alleviating one of the direct methods challenges—high non-convexity of the photometric error. All the LSD-SLAM variants in the end yield a semi-dense map of the environment. An approach combining direct tracking and image features, dubbed *semi-direct*, was presented in (Forster et al., 2014; Forster et al., 2016b) for monocular and multiple camera setups. It was later extended in (Forster et al., 2016a) to include IMU information by preintegrating inertial measurements between selected keyframes into a single relative motion constraint and at the same time properly addressing the manifold structure of the SO(3) and SE(3) groups. As discussed in (Mur-Artal et al., 2015) the advantages of direct methods are that they do not need features, thus they avoid pertaining artefacts, they are more robust to blur, low-texture areas and high frequency texture areas; however, disadvantages are that the photometric error is highly non-convex and limits the baseline of matches (typically narrower than those of feature-based methods), they can be sensitive to illumination changes and moving objects in the scene (which was noted in (Engel et al., 2015) for the case of the KITTI dataset). There also exist sparse feature-based methods coupling the IMU data with visual pose estimation. In (Konolige et al., 2010) a stereo visual odometry was presented that relies on new features for tracking called the center surround extrema (CenSurE) features showing better matching score than the FAST, SIFT and Harris corner features, while being computationally comparable to the FAST and Harris detectors. The authors have also coupled the IMU data with the visual odometry through an EKF, where the IMU was used as both an inclinometer (absolute roll and pitch) and an angular rate sensor (for incremental yaw). This is a *loosely* coupled approach that does not take correlations into account between the IMU data and visual odometry data; however, it was demonstrated that the long term visual odometry accuracy can be dramatically improved by loosely fusing data from the IMU. Another visual-inertial approach was presented in (Leutenegger et al., 2015) for monocular and stereo images. This is a keyframe-based method relying on a probabilistic cost function that combines reprojection error and *tightly* coupled IMU information.

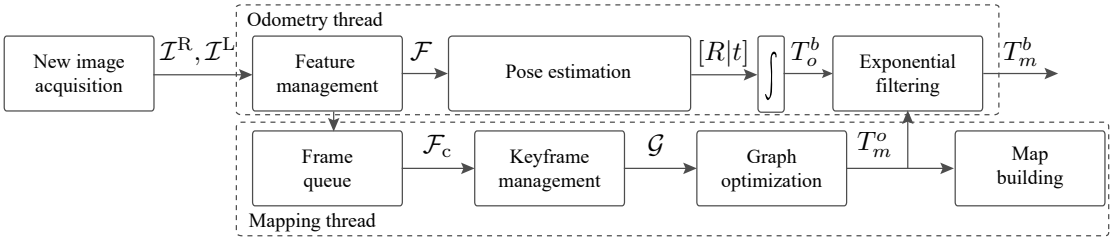

Figure 1: Structure of the SOFT-SLAM algorithm organized in two threads. After left ($\mathcal{I}^{\mathrm{L}}$) and right ($\mathcal{I}^{\mathrm{R}}$) image acquisition, the odometry thread operates so that a set of carefully selected features $\mathcal{F}$ is extracted, and an estimation procedure is performed to obtain the relative transformation $[R|t]$, which is then integrated to obtain the odometry-to-base transformation T_o^b. The mapping thread operates so that the keyframe management procedure takes a frame with accompanied features $\mathcal{F}$ available in the queue. After applying keyframe management logic, the designed graph $\mathcal{G}$ is optimized, resulting with the map-to-odometry transformation T_m^o. Finally, T_o^b and T_m^o are combined via exponential filtering yielding the map-to-base transformation T_m^b.

3 SOFT-SLAM: System Overview

The guideline we follow in our system design is to ensure computationally efficient autonomous navigation of UAVs. Hence we have to ensure real-time execution and high reliability of the SLAM algorithm so the control system ensuring the autonomy of the vehicle could rely on the online SLAM execution. Besides achieving high accuracy of the system, it is important that the algorithm spends nearly constant period of time for execution of each iteration. For these purposes we organize our algorithm in two main threads: the odometry thread and the mapping thread. A rough structure of the SOFT-SLAM algorithm is given in Fig. 1. The odometry thread is important from the perspective of the local controllability, whereas the mapping thread is important from the viewpoint of the overall operation execution. The two threads are finally connected through a filter that uses the output of the mapping thread and fuses it with the odometry information.

The odometry thread can be seen as a sequence of several constituent blocks. First, after a new stereo-pair acquisition, high-quality features are detected. The detection part is based on corner and blob masks on the gradient image followed by non-maximum suppression. Thereafter, the IMU is used to predict the relative displacement of the UAV in order to define a search radius for feature matching. Features are then matched in circular order, using the sum of absolute differences (SAD) of the patches in the image gradient, weighted with the distance to the predicted coordinates as proposed in (Geiger et al., 2011). The output of this step is a sparse feature set that can be further used within a RANSAC optimization procedure. A significance of the RANSAC approach is that our system consists of two different RANSAC optimization algorithms depending on the setup of the system. If an IMU is available, we then proceed by directly employing its gyroscopic measurements, and only execute the 1-point RANSAC optimization. Otherwise, if an IMU is not available,

the common practice is to use a 3-point algorithm (Fraundorfer and Scaramuzza, 2012). However, for certain setups, e.g. KITTI, we found out that computing the rotation from a single camera can improve localization precision at the expense of increased runtime (Cvišić and Petrović, 2015). Therefore, for KITTI we use the 5-point Nister algorithm (Nister et al., 2004) (5-point RANSAC) on the left stereo image for generating hypotheses and determining the best suited set of inliers. The result of the RANSAC is a set of inliers that are used in a Gauss-Newton optimization in order to yield the final relative orientation and translation. This procedure avoids optimizing both at the same time, thus yielding faster and more robust performance. By integrating the obtained relative transformations, we have a result of the cumulative odometry representing an *odometry-to-base* transform $\mathcal{T}_o^b$. More detailed explanation of the odometry thread is given in Section 4.

The mapping thread takes the output of the feature management block from the odometry thread and puts it into the frame queue for further use within the keyframe management step. This step checks whether a current keyframe is to be taken into consideration or is rather to be discarded, depending on the distances from the last stored keyframe. If it is taken into account, the algorithm checks for the possibility of loop closure. After the loop-closing detection, we run g2o on the sparse set of poses (features are not included), which yields an updated pose graph, thus correcting for the odometry drift. The proposed SLAM framework enables a complete separation of the odometry part and mapping part, thus enabling the odometry to run in constant time without having to wait for the mapping thread. The whole method still leaves enough resources so that a dense map of the environment can be constructed using a stereo method for computing depth maps, where we particularly employ the semi-global matching algorithm (Hirschmüller, 2011), and insert the result into the global octomap (Hornung et al., 2013). We consider this to be of great importance for high-speed UAV navigation—especially in indoor and cluttered environments. Note that the octomaps do not participate in the pose estimation process, but were necessary for collision avoidance in the EuRoC competition.

The output of the mapping thread is a correction that can be seen as a *map-to-odometry* transform, denoted by $\mathcal{T}_m^o$. More detailed explanation of the mapping thread is given in Section 5. The two threads are finally merged through an exponential filter and the result can be seen as representing a *map-to-base* transform $\mathcal{T}_m^b$, which is sent to the position controller.

4 Odometry thread

In this Section we provide description of each step in the odometry thread. We are motivated by (i) the application related to an autonomous UAV operation, where an IMU is part of the system setup, and by (ii) the significance of vision-based SLAM without inertial information. For this reason, we present two versions of the odometry algorithm: one using information from an IMU and one relying purely on vision. In the ensuing paragraphs we explicitly emphasize the differences between the two algorithms wherever appropriate.

4.1 Feature management

The odometry thread starts with a feature management step consisting of three parts: feature extraction and matching, tracking, and selection. As an input of this step we use a stereo image and an IMU measurement if available.

Feature management starts with extraction and matching of corner-like features in both left and right images of the stereo pair. For this purpose, in the vein of (Geiger et al., 2011), we utilize blob and corner masks on the gradient image, and apply the non-maximum suppression, thus obtaining a set of available features. The features are then used in the matching process, where the correspondences are determined through calculating the sum of absolute differences over a pattern of pixels around the detected maxima. The matching patch $\mathcal{M}$ is illustrated in Fig. 2. Since the SAD method is susceptible to outliers, we additionally filter the feature set by applying *circular matching*. By circular matching we mean that the features will be taken into account only if they are matched between both left and right images of two consecutive frames corresponding to steps $k-1$ and k, i.e., images denoted as $\mathcal{I}^{\mathrm{L}}_{k-1}$, $\mathcal{I}^{\mathrm{R}}_{k-1}$, $\mathcal{I}^{\mathrm{L}}_{k}$ and $\mathcal{I}^{\mathrm{R}}_{k}$. The circular algorithm takes a feature from one frame and finds the best match in an another frame, sequentially following the order $\mathcal{I}^{\mathrm{L}}_{k-1} \rightarrow \mathcal{I}^{\mathrm{R}}_{k-1} \rightarrow \mathcal{I}^{\mathrm{R}}_{k} \rightarrow \mathcal{I}^{\mathrm{L}}_{k} \rightarrow \mathcal{I}^{\mathrm{L}}_{k-1}$. If an IMU is available we apply the prediction from step $k-1$ to step k; otherwise, we use the constant velocity model assumption and apply the last available relative transformation. Finally, if the feature is successfully matched through the entire sequence, i.e., the last matched feature is the same as the beginning one, the circle is closed and the feature is considered as being stable and therefore kept as a high-quality point for further analysis. Otherwise, the circle is not closed and the feature is rejected. Additionally, if the circle is successfully closed, we calculate the normalized cross correlation (NCC) on a rectangular patch around each feature position, which is applied as an additional check together with the SAD-based circular matching. Since NCC is significantly slower than SAD, we apply it only for gating purposes to reject a certain amount of surviving features with a low NCC score after the SAD-based circular

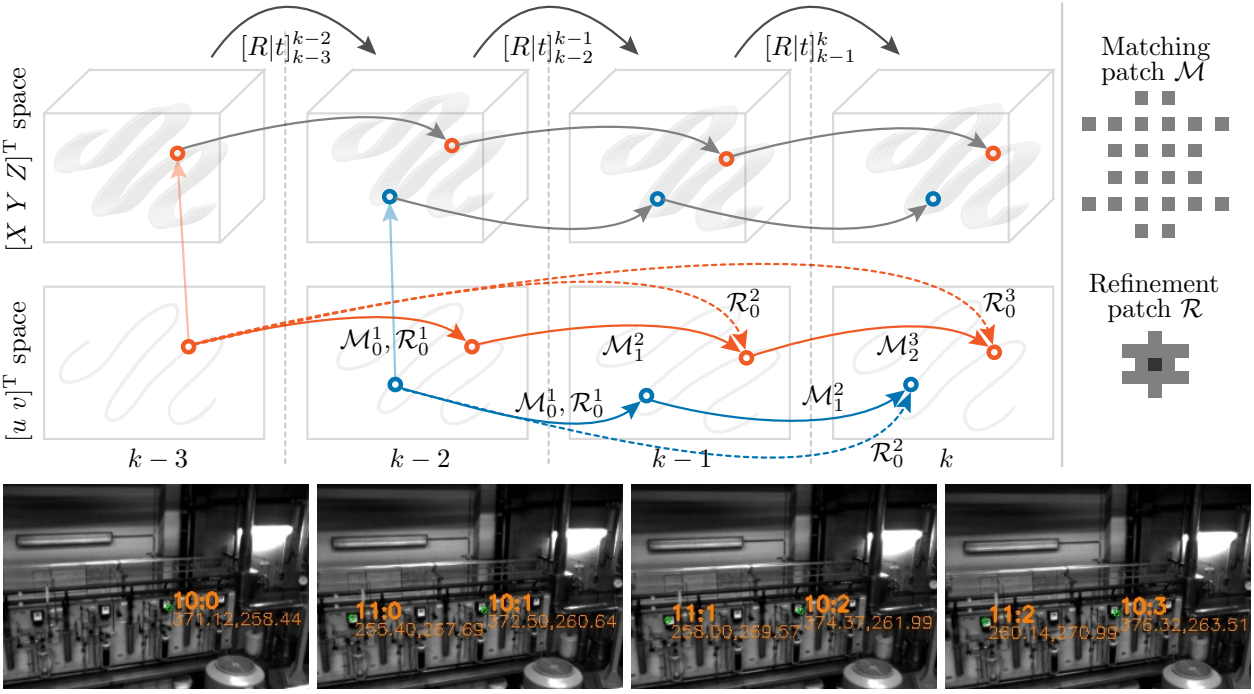

Figure 2: The upper illustration consists of two sequences. The first sequence represents an example of the real-world (3D) propagation of positions of two features (blue and orange) over time. The second sequence represents its propagation in the image space, where solid connections appear when matching features, while dashed connections appear during the refinement procedure. Note that the refinement is always applied with respect to the initial feature appearance, while matching relates two consecutive steps. In the upper right corner, the patches used for matching ($\mathcal{M}$) and refinement ($\mathcal{R}$) are presented. Therein, the level of gray intensity represents larger weighting for a given pixel. An example of the real image sequence filled with their feature appearances is given in the bottom sequence, where id, age and subpixel position is associated to each feature.

matching has been performed.

The resulting set of features can now be tracked through time and we associate several properties to each of them including: (i) unique identifier, (ii) age, (iii) current position in image space, (iv) feature strength, (v) class, and (vi) initial descriptor. The tracking algorithm for each feature consists of:

- the initial step—where a unique identifier and a descriptor are joined to the pertaining feature, and the age of the feature is set to 0
- the tracking—where in each iteration the position of the feature in the current frame is refined on a subpixel level with respect to the feature position in the initial frame, using the refinement patch $\mathcal{R}$ illustrated in Fig. 2.

By using the initial value of the descriptor during the entire feature lifetime the drift is reduced, since allowed

dissimilarity between initial and current appearance is kept constrained. However, an intensive change in perspective relative to the first appearance of the feature can result in death of the particular feature track. Another option would be to warp the initial feature according to an a priori pose estimate. However, besides slightly increasing the computational complexity, warp would also require a computation of a patch normal. Since this computation could be an overhead for real-time application, a common practice is to assume that patch normal is pointing at the camera. While this assumption could prolong the feature life, it would also inevitably introduce an error that would grow with feature age. Generally, features that are tracked for a longer period of time are considered being more reliable, but possibly biased due to the changed perspective during its life; therefore, one should try to maintain equal distribution of both young and veteran features. It is possible to prolong the feature lifetime by updating the descriptor through time, but on the other hand, the feature could quickly lose coincidence with the initial real-world point and the procedure would be more prone to drift. Additionally, some features with strong image appearances can appear even lacking the correspondence in the real world, e.g., intersection of a line in the foreground with a line in the background (Shi and Tomasi, 1994). Given that, we decided to still keep the initial value of the descriptor throughout the whole lifetime, thus risking earlier feature track death, but having a more reliable feature set. Furthermore, we propagate the refined position on a subpixel level while establishing matches between current and previous occurrence in the vein of a Markov process, i.e., assuming that information about displacement relative to the first occurrence is preserved in the refined feature position. In particular, in our implementation we use the parabolic fitting approach (Geiger et al., 2011). The tracking procedure is illustrated in Fig. 2 together with an exemplary feature set from a real image sequence.

Rather than estimating ego-motion using a large set of features, in the further processing we keep a smaller but carefully selected set of features. We propose a selection policy that provides both *spatial* and *temporal variety* of strong features across the image. We particularly choose features such that the number of far and near features is well balanced, and that the features are uniformly distributed over the entire image (Kitt et al., 2010). This is achieved by performing bucketing where an image is divided into rectangles/buckets. In each bucket, only a limited number of features is retained, while all others are discarded. Our experiments showed that such distribution of features improves localization accuracy. The procedure is given as follows:

- the image is convolved with a 5×5 corner patch and a 5×5 blob patch, where maxima and minima of these two convolved images represent image coordinates of four classes of features: corner max, corner min, blob max, and blob min. Feature descriptor is then constructed from values of the surrounding pixels in the image gradient, both horizontal and vertical.

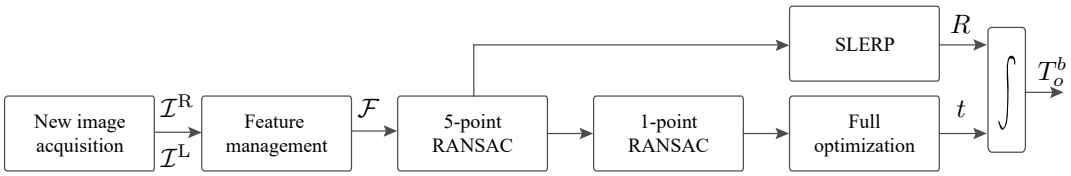

Figure 3: Pipeline of the pure vision based SOFT visual odometry algorithm. After left ($\mathcal{I}^{\mathrm{L}}$) and right ($\mathcal{I}^{\mathrm{R}}$) image acquisition, a set of carefully selected features $\mathcal{F}$ is extracted. Next, the estimation procedure involving several steps is executed. This results with relative transformation $[R|t]$, which is then integrated to obtain the odometry-to-base transformation T_o^b.

- the image is divided into 50×50 pixels, areas called buckets, and within each bucket features are separated into 4 classes, namely corner max/min and blob max/min
- within each class, the features are sorted using the following comparison function

$$\mathrm{select}(f_1, f_2) = \begin{cases} \mathrm{stronger}(f_1, f_2)\,, & \text{if } \ \mathrm{age}(f_1) = \mathrm{age}(f_2) \mid \mathrm{age}(f_1) > a_{th} \mid \mathrm{age}(f_2) > a_{th}\,, \\ \mathrm{older}(f_1, f_2)\,, & \text{if } \ \mathrm{age}(f_1) \neq \mathrm{age}(f_2) \ \&\ \mathrm{age}(f_1) \leq a_{th} \ \&\ \mathrm{age}(f_2) \leq a_{th}\,, \end{cases} \tag{1}$$

  where f_1 and f_2 are compared features, and a_{th} is an age threshold, while the strength is obtained as a result of the filtering of input image with blob and corner masks,
- respecting the selection criteria, the first feature from each class is put into the final list and the step is repeated until the desired number of features is selected.

By this, we prefer old features only up to age a_{th}. Once a feature gets older, the policy starts preferring a stronger one and the feature continues living only if it is strong enough, and is not privileged because of its age anymore. This way we allow new strong features to enter the selection, preventing the veterans to dominate the odometry. *Spatial variety* policy is achieved first by bucketing and second by favoring diverse classes of features in the final feature set, while *temporal variety* policy is realized with sorting selection function that maintains balance between young and old features. By following the logic of this careful selection, we ensure that the odometry is based on a heterogeneous set of features, which turned out to be an important tool for ego-motion estimation accuracy.

4.2 Pose estimation without inertial information

The odometry estimation pipeline without an IMU is given in Fig. 3. Since we rely on the monocular 5-point method (Nister et al., 2004), the approach is not affected by imperfect calibration between the left and right camera. The entire ego-motion estimation is performed in two steps: first, the rotation is estimated using

the 5-point method, and second, the resulting rotation is used for estimating translation via minimization of reprojection errors in the image plane. For computing the relative rotation between time steps $k-1$ and k we use only the left camera of the stereo pair. The epipolar constraint between two views of a calibrated camera can be expressed as

$$q'^{\mathrm{T}} E q = 0\,, \tag{2}$$

where the matrix $E = [t]_\times R \in \mathbb{R}^{3\times 3}$ is the essential matrix, $[t]_\times$ is the corresponding skew-symmetric matrix, while $q \leftrightarrow q'$ are normalized corresponding image coordinates between the two views. Since the scale in monocular vision is unobservable, eight unknown parameters have to be determined. The solution to this problem is approached first by picking five correspondences and constructing the pertaining five equations, and then using the following properties of the essential matrix

$$\det(E) = 0\,, \tag{3}$$

$$2EE^{\mathrm{T}}E - \mathrm{tr}(EE^{\mathrm{T}})E = 0\,. \tag{4}$$

The solution to such a defined system lays in one of the roots of the tenth degree polynomial (Nister et al., 2004). The 5-point method is herein used in conjunction with the RANSAC algorithm as follows. An essential matrix is calculated for a number of 5-point subsets chosen from the full set of points, hence forming corresponding hypotheses. Each of the obtained essential matrices is tested so that the number of inliers among all the points is determined, and the hypothesis with the largest number of inliers is chosen as the final solution, which we denote as $R_{k-1}^{k^r}$.

In our experiments, we also tested an approach with an additional refinement where the best previously obtained 5-point solution is updated with a larger set of inliers for the hypothesis at hand via an optimization. However, it turned out that the average odometry error increased. Depending on the configuration of inliers this method may converge to a local minima, resulting with rotation shifted from the original 5-point solution (Nister et al., 2004). Additionally, the original 5-point method has (i) a closed-form solution and does not depend on the initial guess and (ii) uses minimal set of points hence the probability that an outlier affects the final solution is minimized.

For smoothing purposes we propose to repeat the 5-point RANSAC optimization between the last and several previous frames. In particular, we repeat the algorithm at most 3 times depending on the amount of surviving features between steps k and $k-l$, where $l \in \{1,2,3\}$. Given that, the last relative rotation

between $k-1$ and k can be expressed in different ways as follows

$$R_{k-1}^{k'} = R_{k-1}^{k^r} \tag{5}$$

$$R_{k-1}^{k''} = \left(R_{k-2}^{k-1}\right)^{-1} R_{k-2}^{k^r} \tag{6}$$

$$R_{k-1}^{k'''} = \left(R_{k-3}^{k-2} R_{k-2}^{k-1}\right)^{-1} R_{k-3}^{k^r} \tag{7}$$

where the superscript r denotes application of the 5-point RANSAC, and R_{k-l}^{k-l+1}, $l \in \{2,3\}$ denote the previous relative odometry transformations. Finally, the estimated transformations $R_{k-1}^{k'}$, $R_{k-1}^{k''}$ and $R_{k-1}^{k'''}$ are fused using spherical linear interpolation (SLERP) approach (Dam et al., 1998). It is applied iteratively as follows

$$\text{iteration 1: } R_{k-1}^{k} = R_{k-1}^{k'}\,, \tag{8}$$

$$\text{iteration 2: } R_{k-1}^{k} = R_{k-1}^{k}\left(\left(R_{k-1}^{k}\right)^{-1} R_{k-1}^{k''}\right)^{\frac{1}{2}}, \tag{9}$$

$$\text{iteration 3: } R_{k-1}^{k} = R_{k-1}^{k}\left(\left(R_{k-1}^{k}\right)^{-1} R_{k-1}^{k'''}\right)^{\frac{1}{3}}. \tag{10}$$

It is not necessary that all the iterations are executed. For example, if too few features appearing in step $k-l$, $l \in \{1,2,3\}$, are still alive in step k, only $l-1$ iterations will be executed. Here we slightly abuse the notation, since we do not actually use the rotation matrices, but quaternion representation of rotations, since SLERP operates in quaternion space. This rotation is used as an input for the translation estimation part, where we employ the 1-point stereo method.

The 1-point stereo method relies on reconstruction of 3D point clouds from the previous and current frames. In (Nister et al., 2004) it was shown that the motion estimation obtained by minimizing image reprojection error is more accurate than the one obtained by minimizing Euclidean error in 3D space. This is due to the fact that feature position uncertainties are not taken into account during minimization. Furthermore, in Euclidean space error covariances are highly anisotropic, while in the image space errors are more uniformly distributed between dimensions (Badino et al., 2013). Therefore, the 3D point cloud is triangulated from the previous view and then reprojected onto the image plane of the current view via following formulae

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \pi(X; R, t) = \begin{bmatrix} f & 0 & c_u \\ 0 & f & c_v \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} R \,|\, t \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{11}$$

where $[u\ v\ 1]^{\mathrm{T}}$ are homogeneous image coordinates, π is projection function, f is the focal length, $[c_u\ c_v]^{\mathrm{T}}$ is the image principal point, $[R\,|\,t]$ are rotation and translation between two views, and $[X\ Y\ Z\ 1]^{\mathrm{T}}$ are homogeneous coordinates of the 3D world point. The translation is then iteratively calculated by minimizing the following cost function

$$\arg\min_{t} \left(||x_i^l - \pi^l(X; R, t)||^2 + ||x_i^r - \pi^r(X; R, t)||^2 \right), \tag{12}$$

for the i-th point of the previously obtained feature set. This optimization is performed fast enough so that it can be executed for each point of the feature set. Therefore, each point can be considered as a hypothesis that is then tested on the full feature set. The one with the largest set of inliers is chosen for the outlier rejection approach, by which we obtain the final feature set serving as an input for the full optimization. For this purpose, we apply the full Gauss-Newton optimization approach. The cost function is given as

$$\arg\min_{t} \sum_{i=1}^{n} w_i \left(||x_i^l - \pi^l(X; R, t)||^2 + ||x_i^r - \pi^r(X; R, t)||^2 \right), \tag{13}$$

where w_i is the weight of the i-th point. The weighting function is based on the one presented in (Geiger et al., 2011) with an additionally proposed age-based factor

$$w_i = w_{\text{lens}} w_{\text{age}}, \text{ where } w_{\text{lens}} = \left(\frac{|u_i - c_u|}{|c_u|} + p_{\text{lens}} \right)^{-1} \text{ and } w_{\text{age}} = 1 + \log(\text{age}_i + 1), \tag{14}$$

where parameter p_{lens} depends on the stereo camera and lens setup. By applying this weighting function we give more significance to features located closer to the image center in the horizontal direction, while reducing the impact of features located further to the image horizontal margins. By this, we finish the estimation of the relative motion between the two steps that consists of the rotation part resulting from 5-point algorithm. This motion is finally integrated to have the full odometry to base transformation.

4.3 IMU aided pose estimation

If an IMU is part of the system setup we can directly employ its measurements and avoid the need for the 5-point RANSAC approach, thus reducing computational complexity. Pipeline of the underlying algorithm is given in Fig. 4. We employ IMU measurements only for rotation, and again, similarly to the approach without an IMU, use the 1-point optimization approach for estimating the translation.

Furthermore, we also use the Kalman filter for estimating the gyroscope bias. The prediction step of the

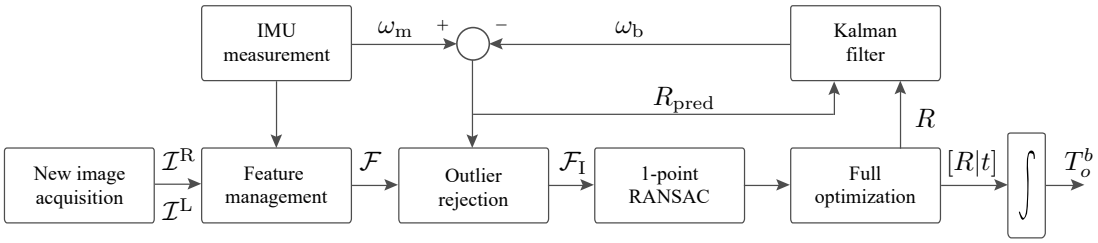

Figure 4: Pipeline of the IMU aided SOFT visual odometry algorithm. After left ($\mathcal{I}^{\mathrm{L}}$) and right ($\mathcal{I}^{\mathrm{R}}$) image acquisition, a set of carefully selected features $\mathcal{F}$ is extracted. Next, based on the measured (ω_{m}) and estimated (ω_{b}) gyroscope bias, outlier rejection is applied, obtaining the feature set $\mathcal{F}_{\mathrm{I}}$. Finally, 1-point RANSAC and full optimization are then used for determining translation t and orientation R, respectively. The transformation $[R\,|\,t]$, is then integrated to obtain the odometry-to-base transformation T_o^b.

Kalman filter is performed once a new IMU reading is obtained, while the update step is executed each time a new image is captured. As a measurement within the filter we use the final output of the visual odometry. The described Kalman filter for bias estimation has been previously presented in (Cvišić and Petrović, 2015).

The IMU measurement together with the estimated gyroscope bias is used for outlier rejection, which could be caused by appearing dynamic objects, and for narrowing the feature matching search space, thus resulting in a reduction of the execution time. Afterwards, we perform the 1-point method in order to compute the translation vector as presented in Sec. 4.2. Each of the obtained $[R\,|\,t]$ hypotheses is tested so that the number of inliers among all the points is determined, and again, the hypothesis with the largest number of inliers is chosen for the final outlier rejection, thus yielding the final feature set serving as the input to the full optimization. For full optimization, we apply the Gauss-Newton in order to determine both the rotation and translation, which is in contrast to the optimization without an IMU, where we only aimed at computing the translation vector, as shown by (13).

4.4 Exponential filtering

Having the odometry $\mathcal{T}_o^b$ estimated, and using the last available odometry drift estimation $\mathcal{T}_m^o$, which is presented in detail in Sec. 5, we are ready for updating the *map-to-base* transform $\mathcal{T}_m^b$. However, for the case of a real-world operating UAV, as in the EuRoC Challenge examples, the update should not occur instantaneously, because the sudden shift in the pose could negatively affect the stability of the platform. Therefore, the estimated pose of the UAV, now represented by a quaternion and translation vector pair, is

updated via an exponential filter as follows

$$R^b_{m,k} = R^b_{m,k-1}\Big(\big(R^b_{m,k-1}\big)^{-1} R^{b,z}_{m,k}\Big)^{\frac{1}{\alpha}} \tag{15}$$

$$t^b_{m,k} = t^b_{m,k-1} + \alpha(t^{b,z}_{m,k} - t^b_{m,k-1}), \tag{16}$$

where the $[R^b_{m,k} \,|\, t^b_{m,k}]$ pair represents the current pose of the UAV in the world map at time $k-1$, $[R^{b,z}_{m,k} \,|\, t^{b,z}_{m,k}]$ is the new 'measured' pose after correcting for the drift, and $\alpha = 0.05$. Note that we abuse notation here since we do not actually update rotation matrices but quaternions using SLERP as in Section 4.2. With the current setting of $\alpha = 0.05$ the described procedure amounts to low-pass filtering, but as stated earlier this is only relevant for real-world UAV operation—in the case of offline SLAM testing, e.g., on the KITTI dataset, this does not affect the final result.

5 Mapping thread

The mapping thread operates independently of the odometry thread, but however uses queued outputs of the feature management procedure. The main reason the mapping thread can operate independently is that its output serves for drift evaluation, while the drift is assumed to progress in the vein of a Wiener process. Therefore, once a new output of the mapping thread is available, i.e., a new *map-to-odometry* correction T^o_m is estimated, it can be used no matter how old the queued frame is. The mapping thread is organized in several steps as depicted in Fig. 2. It starts by taking a frame from the queue, continues with keyframe management accounting for a loop closing possibility, and accordingly prepares the poses for graph optimization. A few examples of the environment maps built by SOFT-SLAM are given in Fig. 5.

5.1 Keyframes management

Keyframe management relies on the feature set resulting from the extraction step described in Sec. 4.1, obtained after applying the non-maximum suppression procedure. We denote this input feature set as a *frame*. The pipeline for keyframe management procedure is given in Fig. 6.

Each frame is considered as a potential keyframe by checking whether the distance between the previous keyframe and the current frame exceeds a predefined threshold in either translation (d_{th}) or rotation (θ_{th}). If the distance is below the given threshold, we discard the current frame and take the next one from the frame queue. If the distance exceeds the predefined threshold, we check for the possibility of loop closing

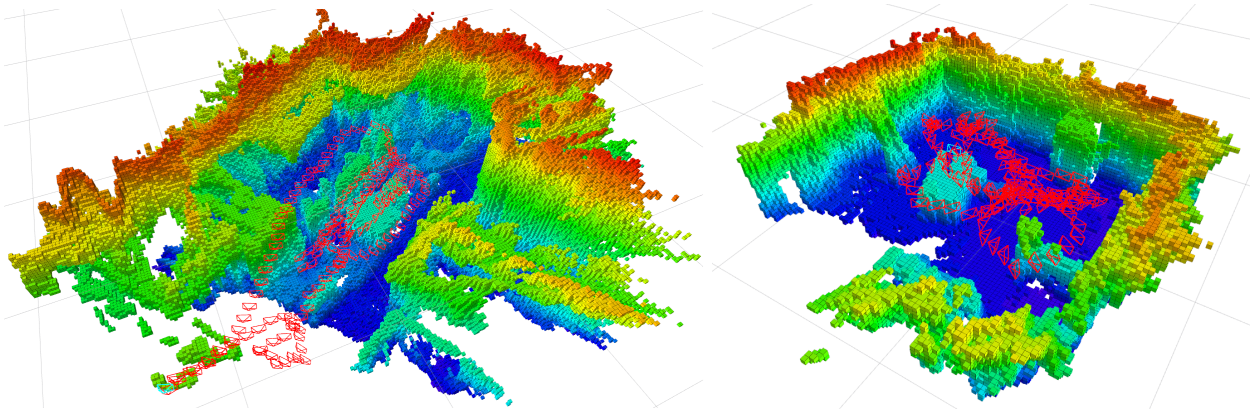

Figure 5: Examples of the resulting octomaps built during the execution of the SOFT-SLAM algorithm on the EuRoC dataset. The maps are built in real-time during the SOFT-SLAM execution on MH_01 (left) and V1_01 (right). The octomaps are not part of the pose estimation process, but were necessary for collision avoidance in the EuRoC competition.

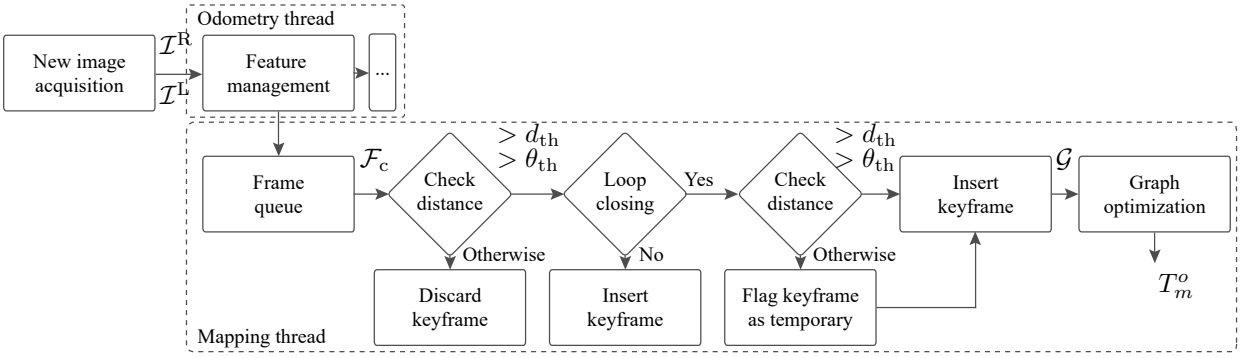

Figure 6: The pipeline of the keyframe management procedure. The threshold parameters in translation (d_{th}) and rotation (θ_{th}) components condition the logic of the keyframe management. Finally, after obtaining final graph $\mathcal{G}$, the graph optimization is used for determining the map-to-odometry transformation T_m^o.

(which is described in details in Sec. 5.2). If loop closing is not detected we insert the current frame as new keyframe and associate the current pose and covariance to it. The covariance is initialized such that the translation and rotation parts are proportional to the distance between the previous and the new keyframe with proportionality factors p_d and p_θ. In other words, we follow the odometry logic: higher the displacement, higher the uncertainty. On the other hand, if the loop closing is detected, we check for the distance of the new keyframe to the closest keyframe in the map of keyframes using the information from the odometry. If this distance exceeds any of the two thresholds, d_{th} or θ_{th}, we insert this keyframe into the full map of keyframes. Otherwise, we flag it as a temporary keyframe that will be removed from the full map of keyframes once the graph optimization is finished. Before continuing to the graph optimization, we have to initialize new edges, formed between the new keyframe and any such keyframes with whom loop closing was detected. As in the vein of the previous covariance initialization, we use the proportionality factors p_d and p_θ, multiplying the

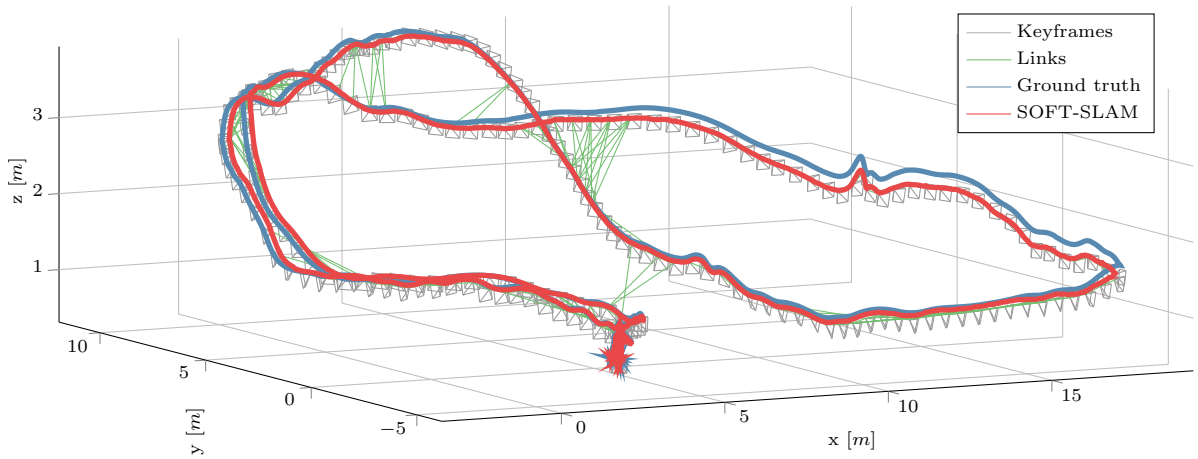

Figure 7: Visualization of keyframes and links resulting from the SOFT-SLAM execution for the trajectory MH-04 of the EuRoC dataset.

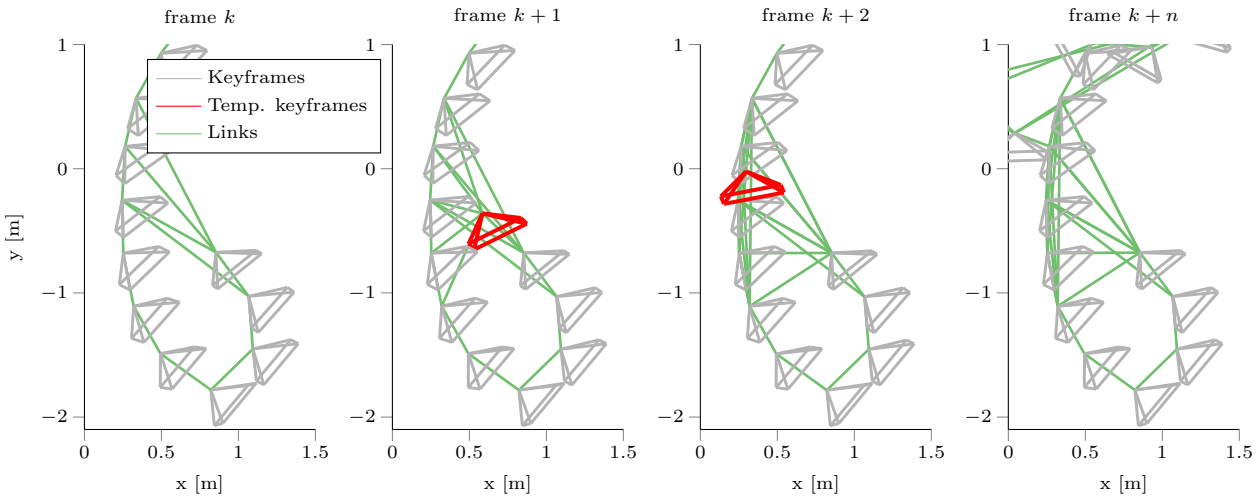

Figure 8: Visualization of evolution of keyframe map through several steps, appearing during the SOFT-SLAM execution for the trajectory V1-02 of the EuRoC dataset. At steps $k+1$ and $k+2$ temporary keyframes were inserted, while after several steps at $k+n$, all temporary keyframes were removed and only regular keyframes remained.

distance between the estimated odometry information and those joined to the keyframes detected for loop closing. An example of visualization of the map of keyframes and established links for a trajectory from the EuRoC dataset is given in Fig. 7 and Fig. 8. As the reader will notice, in this paragraph we introduced the concept of temporary keyframes. Temporary keyframe is a frame that is temporarily necessary for loop closing and graph optimization, but becomes an overhead for in the long-term, because the graph already contains nearby keyframe carrying very similar information. After every graph optimization, all temporary keyframes are removed, thus ensuring constrained number of keyframes in constrained space.

5.2 Loop closing

The loop closing procedure starts with determining candidate keyframes that will be considered for loop closing. For this purpose we employ the radius and angle search based on the pose of the new keyframe. In particular, only keyframes that are placed within the predefined radius and angle will be considered as candidates for loop closing, hence reducing the complexity of this step. Once having the set of candidates, we evaluate each of them so that comparing histograms of each keyframe with the one of the new keyframe. The histograms consists of four classes, i.e., corner max/min and blob max/min, while for the distance/similarity measure between two histograms we use SAD. After having determined the similarity information, we keep n closest keyframes as candidates for loop closing, while discarding the others. We now evaluate each candidate with the following procedure:

- determine correspondences between the candidate keyframe and the new keyframe using circular matching and NCC
- perform 3-point RANSAC (Scaramuzza and Fraundorfer, 2011) for determining the best suitable hypothesis transformation and accordingly apply the outlier rejection
- if the remaining number of inliers is larger than the threshold, we apply the full refinement optimization by which we obtain the final linking transformation
- additionally, we check if the final linking distance is too large, and eventually stop with the loop closing.

Having keyframes arranged in a pose graph, we use the Levenberg-Marquard algorithm implemented in g2o (Kümmerle et al., 2011) for the optimization. In order to prepare the map of keyframes for graph optimization, besides poses for the initialization and pose constraints as edges of the graph, g2o also requires edge covariance matrices, i.e., the uncertainties of the relative pose constraints. For this purpose, here we again use a straightforward heuristic—the relative rotation and translation uncertainty is proportional to the magnitude of the relative motion, with proportionality factors p_d and p_θ, respectively.

6 Experimental results and discussion

For experimental results we evaluate SOFT-SLAM on two public datasets; namely, the EuRoC dataset (Burri et al., 2016) and the KITTI dataset (Geiger et al., 2011). For the EuRoC dataset, since IMU

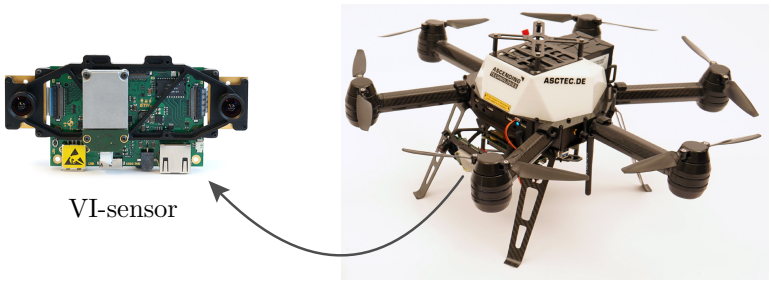

Figure 9: AscTec Neo (Ascending Technologies GmbH, 2017) multirotor unmanned aerial vehicle equipped with the VI-sensor (Nikolic et al., 2014).

measurements are readily available, we use the odometry thread procedure described in Section 4.3. Since IMU is not available for the KITTI dataset, in the odometry thread we use the pose estimation based on the procedure described in Section 4.2. We evaluate the proposed method using two different metrics: (i) *relative* metric proposed in (Geiger et al., 2013) that computes average relative translation and rotation errors over a sliding window of different lengths, and (ii) *absolute* metric proposed in (Sturm et al., 2012) that aligns the trajectories prior to computing absolute translation root-mean-square-error (RMSE). The former metric is more odometry-oriented, while the latter one is more SLAM-oriented. Furthermore, we also present localization results from the EuRoC Challenge 3, Stage II.

6.1 EuRoC dataset

The EuRoC dataset contains eleven sequences recorded with the AscTec Neo small-scale hexacopter UAV (Ascending Technologies GmbH, 2017), shown in Fig. 9, which is equipped with a visual-inertial (VI) sensor unit (Nikolic et al., 2014). The VI sensor unit was mounted in a front-down looking position providing WVGA stereo grayscale images with a baseline of 11 cm at a rate of 20 Hz and inertial data at a rate of 200 Hz. The dataset provides two types of sequences: the first batch was recorded in a large machine hall, while the second batch was recorded in a room equipped with a motion capture system. Depending on the texture, brightness, and UAV dynamics the sequences are classified as easy, medium, and difficult. In the case of the EuRoC dataset, the UAV revisits the environment, hence loop closing occurs quite often, which is especially significant from the perspective of SLAM evaluation.

We compare our SOFT-SLAM on the EuRoC sequences with ORB-SLAM2 and LSD-SLAM. SOFT-SLAM is extensively compared with ORB-SLAM2 on a laptop with an Intel Core i7 processor, at 2.4 GHz, since the authors of ORB-SLAM2 made the source code publicly available. On the other hand, since the code for stereo LSD-SLAM is not available, we only refer to its results for three sequences of the dataset as given in

Table 1: Results of SOFT-SLAM compared with ORB-SLAM2 and LSD-SLAM on the EuRoC dataset using the absolute metric (as used in EuRoC). Results marked by '*' indicate that modified calibration parameters were used as opposed to publicly available parameters. Subscripts abs, μ, m, and M denote absolute position error, mean, minimum, and maximum values, respectively. ORB-SLAM2 has two absolute position error values (minimum and maximum) due its non-deterministic nature of results.

	SOFT-VO	SOFT-SLAM				ORB-SLAM2					LSD-SLAM
Metric	t_{abs} [cm]	t_{abs} [cm]	T_{μ} [ms]	T_{m} [ms]	T_{M} [ms]	$t_{abs,m}$ [cm]	$t_{abs,M}$ [cm]	T_{μ} [ms]	T_{m} [ms]	T_{M} [ms]	t_{abs} [cm]
V1_01	8.9	**4.2**	28	17	**40**	8.7	8.8	56	35	106	6.6
V1_02	9.7	**3.4**	26	19	**36**	6.4	6.6	54	26	121	7.4
V1_03	10.1	**5.7**	24	15	**34**	7.2	9.4	54	27	124	8.9
V2_01	13.6	**7.2***	26	16	**35**	6.1	7.6	53	31	116	-
V2_02	27.6	**6.9***	25	16	**39**	5.6	7.3	63	22	163	-
V2_03	72.4	**17.3***	24	16	**35**	-	-	-	-	-	-
MH_01	10.0	**2.8**	29	20	**41**	4.0	4.3	70	36	390	-
MH_02	5.6	**4.2**	29	18	**42**	4.3	5.0	65	36	312	-
MH_03	17.0	**3.8**	27	19	**38**	3.5	4.0	66	34	249	-
MH_04	28.1	**9.6**	27	15	**43**	7.1	13.8	56	29	117	-
MH_05	20.3	**5.8**	29	17	**42**	5.3	15.3	59	27	298	-

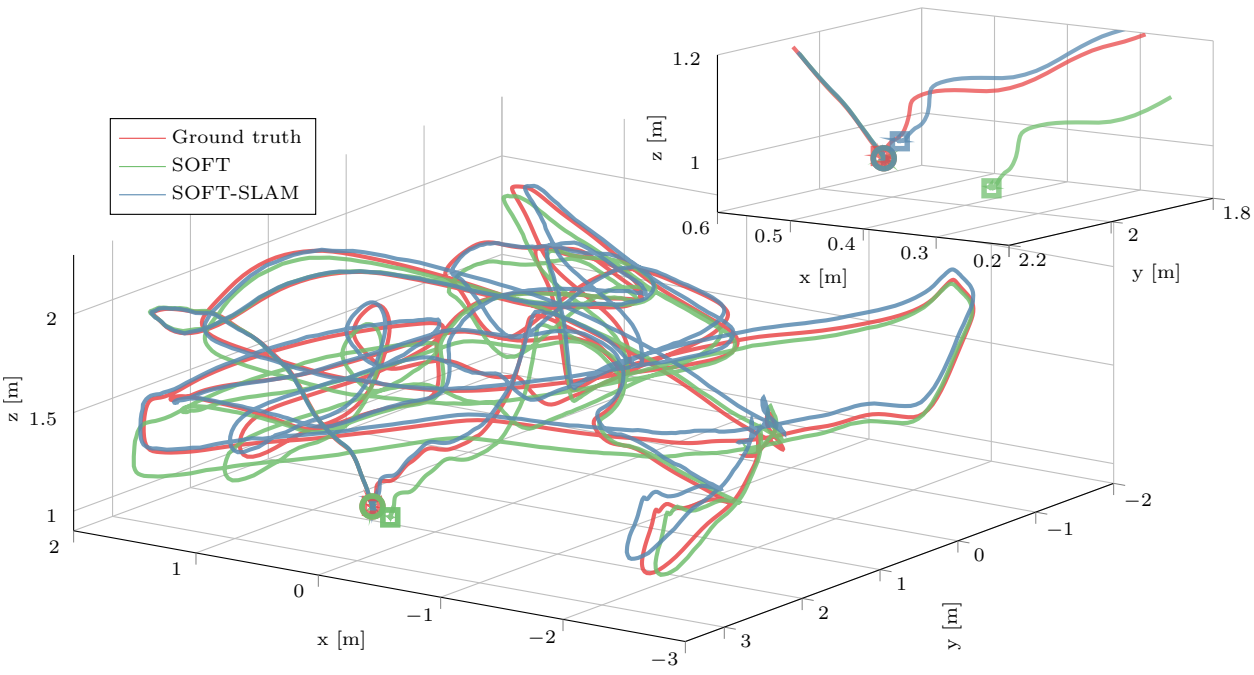

Figure 10: The results of SOFT-SLAM algorithm for sequence V1_02 of the EuRoC dataset. Circles denote starting points, while squares represent ending points of the trajectories.

(Engel et al., 2015). Table 1 presents (i) results for the absolute metric and (ii) runtimes for each algorithm on each track of the EuRoC dataset.

Table 2: Number of keyframes, links, and loop closings for SOFT-SLAM for all the sequences of the EuRoC dataset. The number of loop closings can be computed as the difference of the number of links and keyframes increased by one.

Seq.	V1_01	V1_02	V1_03	V2_01	V2_02	V2_03	HM_01	MH_02	MH_03	MH_04	MH_05
Keyframes	132	166	158	80	188	181	166	161	286	202	219
Links	231	348	256	90	285	224	252	275	495	315	315
Loop closings	100	183	99	11	98	44	87	115	210	114	97

It is important to note that the authors of ORB-SLAM2 claim slightly better results then presented in Table 1 (e.g., $t_{\text{abs}}(\text{V1\_01}) = 3.5\,\text{cm}$, $t_{\text{abs}}(\text{MH\_01}) = 3.5\,\text{cm}$ (Mur-Artal and Tardos, 2016)). However, the original paper reports only median results. Due to the non-deterministic nature of the multi-threading system of the ORB-SLAM2 algorithm, which produces different results in each run, we ran it 5 times on each trajectory of the dataset and present minimum $t_{\text{abs,m}}$ and maximum $t_{\text{abs,M}}$ errors respecting the absolute metric. Note that this does not apply to SOFT-SLAM, which achieves deterministic results, i.e., each run on the same dataset always yields the same result. Furthermore, even though the median result might have the greatest probability of occurrence, due to the general nature of the UAV applications, we employ the absolute error and runtime comparisons based of the worst case runs. Alongside the maximal runtime, we also provide information of the mean T_μ and minimal T_{m} runtime for each trajectory. We bold the result with the minimal absolute error and the worst case runtime respecting each of the 11 testing scenarios. In Fig. 11 we also show SOFT-SLAM and ORB-SLAM2 runtimes for two different EuRoC sequences, where we can notice lower runtime of SOFT-SLAM and that on occasions ORB-SLAM2 can exhibit runtime spikes. Alongside the execution runtimes, we further illustrate the approach to keyframe management and loop closing, by providing the number of keyframes, links, and loop closings for all the EuRoC sequences in Table 2.

We have additionally noted that the publicly available calibration parameters given for the V2 set of the tracks are wrong. A clear indicator for this was that the epipolar lines in these tracks were not aligned. Therefore, we had to apply an additional rotation of 0.24 deg between cameras to the original calibration files to align epipolar lines and achieve the results presented in the Table 1. On the other hand, ORB-SLAM2 proves to be robust to this calibration related issues. Furthermore, ORB-SLAM2 had problems with processing the V2_03 due to severe motion blur (Mur-Artal and Tardos, 2016). The SOFT-SLAM was able to process the sequence, but achieved relatively larger error with respect to the other sequences of the dataset due to missing images in V2_03, motion blur, and lack of texture in parts of this sequence.

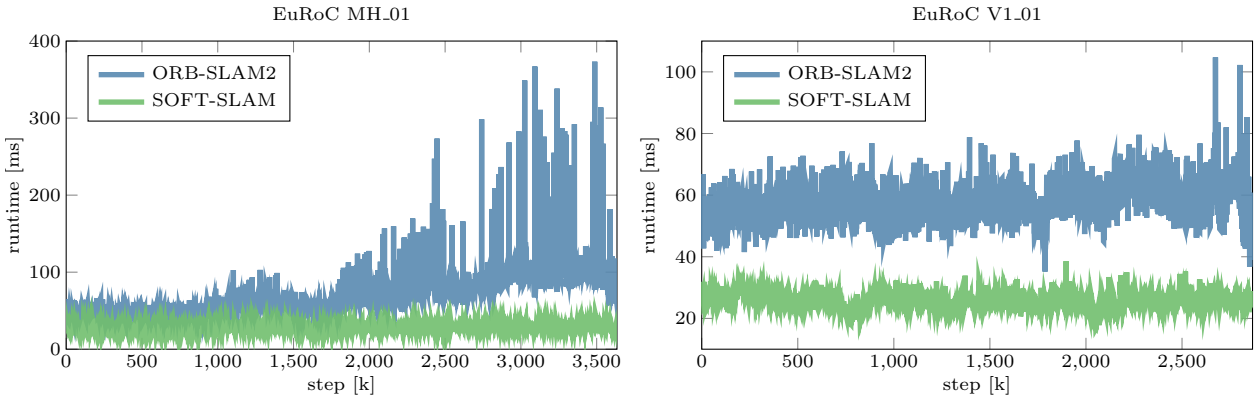

Figure 11: Runtimes of SOFT-SLAM compared with ORB-SLAM2 on the trajectories MH_01 (left) and V1_01 (right) of the EuRoC dataset.

6.2 Live real-world experiment - EuRoC competition

Besides the EuRoC dataset, we also provide results achieved during the EuRoC Challenge 3, Stage IIa–Benchmarking, Task 2 tests for the Task 2: State estimation and controller accuracy. This experiment included an on-sight autonomous UAV flight, while the results were obtained by the EuRoC automated script and supervised by the EuRoC commission. The flight was officially recorded and the video is available as the accompanying material to this paper. The Task 2 evaluated trajectory following based on localization using only the VI-sensor (Nikolic et al., 2014). The experiment was done in a labyrinth-like room as follows:

- initialization step; while airborne, the UAV gets a message from the system about its ground truth position, which the system obtains from the VICON motion capture, hence the UAV is allowed to align its odometry frame of reference with the VICON
- after the initialization step, the target trajectory represented in the VICON reference frame is sent to the UAV, and the trajectory tracking algorithm based on the on-board localization is started
- besides the autonomous UAV operation, the system tracks the executed trajectory and computes absolute position error using the VICON system.

For evaluation purposes, the trajectory was divided in 4 connected sub-trajectories, representing 4 separately evaluated subtasks. The complexity of the subtasks successively increased, and teams could achieve partial score even if they did not complete all the subtasks. Additionally, if the UAV violated safety distance to any of the obstacles, the ongoing task would be stopped, and no points would be gained for the subtask. The BladeHunters team (called UNIZG-FER in the Simulation part of the Challenge) relying on SOFT-SLAM

Table 3: Performance results of the BladeHunters team in EuRoC Challenge 3, Stage IIa–Benchmarking, Task 2 of EuRoC, where SOFT-SLAM localization was used. The BladeHunters team achieved the highest score among the 5 teams participating in this stage of the EuRoC competition. Results represents absolute translation RMSE.

Team [Method]	subtask 1	subtask 2	subtask 3	subtask 4
BladeHunters [SOFT-SLAM]	15.78 cm	15.19 cm	17.72 cm	13.56 cm

succeeded to finish the whole trajectory without breaking the safety distance to any of the obstacles in the room, while achieving the highest score for Task 2 among all the five participating teams. Table 3 shows the absolute translation RMSE for each of the subtasks achieved by the BladeHunters team. Note that the error given in Table 3 arises both from state estimation and controller accuracy. However, no additional information from the competition is available and therefore we cannot extract the sole state estimation error. In addition, it is worth noting that several processes together with SOFT-SLAM were running on-board using limited CPU resources. This includes a model predictive controller, sensor fusion and other important processes which altogether made this result possible. Several snapshots of the BladeHunters Task 2 execution (top), and the truly executed trajectory (bottom), are given in Fig. 12.

6.3 KITTI dataset

To further test effectiveness of SOFT-SLAM and demonstrate its applicability for other types of vehicles except UAVs, we tested the algorithm on the KITTI dataset, which is the most popular odometry benchmark dataset providing a very thorough analysis of rotational and translational error in urban, rural, and highway scenarios. It contains eleven sequences recorded by a car equipped with various sensors; including, four cameras, a Velodyne laser range scanner, and an accurate inertial and GPS navigation system. The four cameras set-up stereo pairs (color and grayscale) at a baseline of 0.54 m providing images at a rate of 10 Hz with a resolution somewhat smaller than the original 1392×512 pixels due to rectification. Table 4 provides a snapshot of the current ranking of different algorithms on evaluation sequences, including the proposed SOFT visual odometry.

We compare our SOFT-SLAM on all the KITTI sequences with ORB-SLAM2 and LSD-SLAM. Note that only sequences 00, 02, 05, 06, 07, and 09 contain loops, hence other sequences basically evaluate just the odometry. In particular, as in the vein of (Mur-Artal and Tardos, 2016), in Table 5 we use two different metrics, (i) absolute translation RMSE t_{abs} (Sturm et al., 2012), and (ii) average relative translation t_{rel} and rotation r_{rel} errors (Geiger et al., 2013).

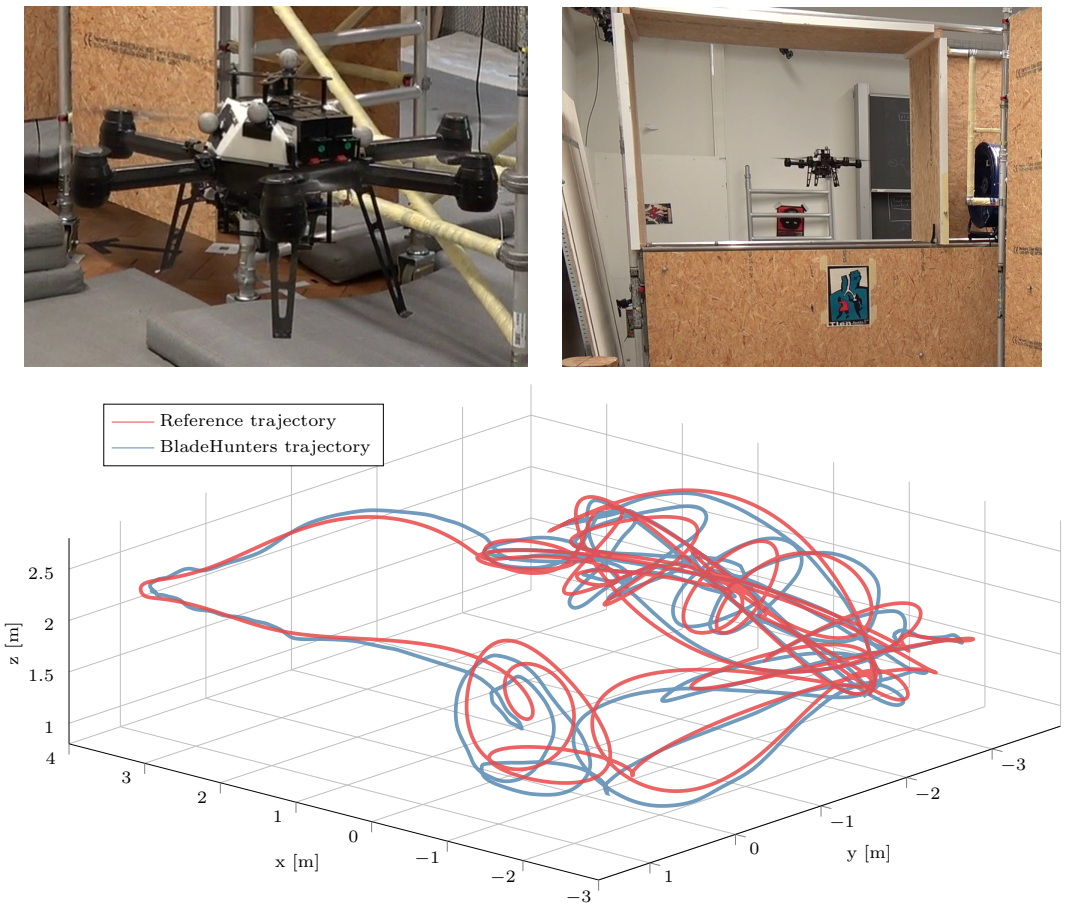

Figure 12: A snapshot of the experiment during the EuRoC competition with the UAV in-flight (top) and the result of the EuRoC EuRoC Challenge 3, Stage IIa–Benchmarking, Task 2 achieved by BladeHunters team based on SOFT-SLAM (bottom). The quantitative results for the executed trajectory are given in Table 3.

In general, SOFT-SLAM achieves better or equal performance on two-thirds of the error measures for all of the eleven sequences. It is also worth noting that compared to the SOFT visual odometry, the difference in accuracy is evident on sequences containing loop closing trajectories.

6.4 Discussion

The core idea of the proposed approach was to develop a highly accurate localization module that runs in real- and constant-time on a small-scale UAV, like the AscTec Neo used in EuRoC, and still leave enough processing power for control and navigation algorithms. Because of these constraints, the algorithm development required an economical approach. All of this resulted in SOFT-SLAM: a stereo visual sparse feature-based pose graph SLAM solution running at the frequency of 20 Hz on two threads on the AscTec Neo onboard computer (Intel Core i7-5557U, 4M cache, 3.40 GHz, 16 GB DDR3). When compared to stereo ORB-SLAM2

Table 4: Current snapshot of the KITTI ranking table consisting of the state-of-the-art algorithms in stereo visual odometry and SLAM.

Method	Setting	Translation	Rotation	Runtime
SOFT2 [Odometry thread (Section 4)]	st	0.81 %	0.0022 [deg/m]	0.1 s / 2 cores
GDVO	st	0.86 %	0.0031 [deg/m]	0.09 s / 1 core
HypERROCC	st	0.88 %	0.0027 [deg/m]	0.25 s / 2 cores
SOFT (Cvišić and Petrović, 2015)	st	0.88 %	0.0022 [deg/m]	0.1 s / 2 cores
RotRocc (Buczko and Willert, 2016a)	st	0.88 %	0.0025 [deg/m]	0.3 s / 2 cores
EDVO	st	0.89 %	0.0030 [deg/m]	0.1 s / 1 core
svo2	st	0.94 %	0.0021 [deg/m]	0.2 s / 1 core
ROCC (Buczko and Willert, 2016b)	st	0.98 %	0.0028 [deg/m]	0.3 s / 2 cores
cv4xv1-sc (Persson et al., 2015)	st	1.09 %	0.0029 [deg/m]	0.145 s / GPU
MonoROCC	st	1.11 %	0.0028 [deg/m]	1 s / 2 cores
ORB-SLAM2 (Mur-Artal and Tardos, 2016)	st	1.15 %	0.0027 [deg/m]	0.06 s / 2 cores
svo	st	1.16 %	0.0030 [deg/m]	0.1 s / 2 core
NOTF (Deigmoeller and Eggert, 2016)	st	1.17 %	0.0035 [deg/m]	0.45 s / 1 core
S-PTAM (Pire et al., 2015)	st	1.19 %	0.0025 [deg/m]	0.03 s / 4 cores
S-LSD-SLAM (Engel et al., 2015)	st	1.20 %	0.0033 [deg/m]	0.07 s / 1 core

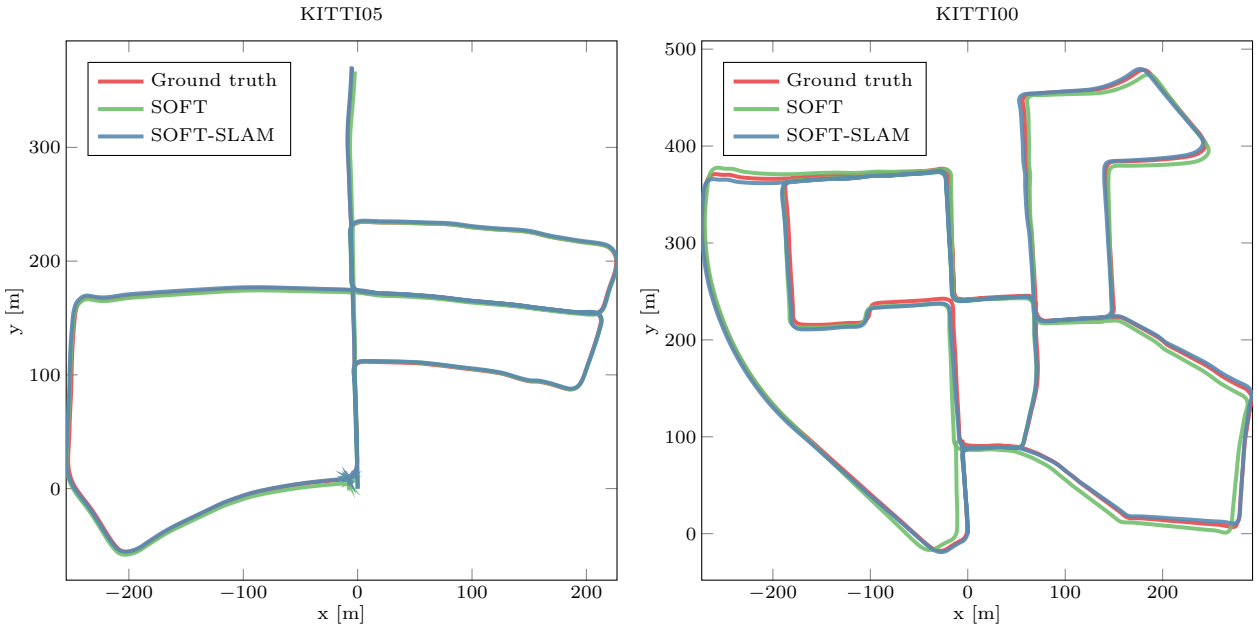

Figure 13: Examples of the results of SOFT visual odometry and SOFT-SLAM algorithms tested on KITTI dataset. Reconstructed trajectories are KITTI05 (left) and KITTI00 (right).

and LSD-SLAM on the EuRoC dataset, SOFT-SLAM achieves smaller absolute error for the majority of tracks (Table 1). However, it should be noted that all the algorithms have a centimeter-range error across all tracks with precision suitable for autonomous indoor UAV operation. Furthermore, in comparison to ORB-SLAM2, SOFT-SLAM has significant advantages in terms of processing speed and CPU-load, which

Table 5: Results of SOFT-SLAM compared with ORB-SLAM2 and LSD-SLAM on the KITTI dataset using both relative (as used in KITTI) and absolute (as used in EuRoC) metric, where t represents translation error, while r represents rotation error.

Metric	SOFT-VO			SOFT-SLAM			ORB-SLAM2			LSD-SLAM		
	t_{rel} [%]	r_{rel} [%]	t_{abs} [m]	t_{rel} [%]	r_{rel} [%]	t_{abs} [m]	t_{rel} [%]	r_{rel} [%]	t_{abs} [m]	t_{rel} [%]	r_{rel} [%]	t_{abs} [m]
KIT.00	0.63	0.24	2.97	0.66	**0.22**	1.20	0.70	0.25	1.3	**0.63**	0.26	**1.0**
KIT.01	0.96	0.18	3.02	**0.96**	**0.18**	**3.02**	1.39	0.21	10.4	2.36	0.36	9.0
KIT.02	0.74	0.20	6.00	1.36	**0.23**	5.09	**0.76**	**0.23**	5.7	0.79	0.23	**2.6**
KIT.03	0.70	0.23	0.54	**0.70**	0.23	**0.54**	0.71	**0.18**	0.6	1.01	0.28	1.2
KIT.04	0.50	0.18	0.35	0.50	0.18	0.35	0.48	**0.13**	**0.2**	**0.38**	0.31	**0.2**
KIT.05	0.47	0.20	1.61	0.43	0.17	**0.77**	**0.40**	**0.16**	0.8	0.64	0.18	1.5
KIT.06	0.38	0.18	1.06	**0.41**	**0.14**	**0.52**	0.51	0.15	0.8	0.71	0.18	1.3
KIT.07	0.36	0.23	0.35	**0.36**	**0.24**	**0.33**	0.50	0.28	0.5	0.56	0.29	0.5
KIT.08	0.78	0.21	2.28	**0.78**	**0.21**	**2.28**	1.05	0.32	3.6	1.11	0.31	3.9
KIT.09	0.74	0.17	2.82	**0.59**	**0.18**	**1.34**	0.87	0.27	3.2	1.14	0.25	5.6
KIT.10	0.68	0.26	0.93	0.68	**0.26**	**0.93**	**0.60**	0.27	1.0	0.72	0.33	1.5

makes it more desirable for flight operations. More concretely, the advantages are as follows

1. It uses only 2 cores compared to 3+1 cores of ORB-SLAM2, so it can provide more CPU power for other essential and time-critical algorithms responsible for keeping the UAV airborne

2. Processing time is more than halved—it can easily process a 20 Hz image stream

3. Processing time is more constant-time and always below 50 ms (above 20 Hz)—ORB-SLAM2 has spikes lasting up to ≈ 0.3 s which could be fatal for a UAV. SOFT-SLAM produces exactly the same result after each run, which is very convenient for testing, and also makes the system more reliable during the flight.

4. Map size does not affect processing time of the main thread.

The workhorse of SOFT-SLAM is the visual odometry that was tailored from the start to the stereo setup. This is in contrast to the ORB-SLAM2 and LSD-SLAM, which were first developed for monocular setups. The odometry is based on an intricate sparse feature management procedure that balances over time feature saliency and persistence through frames in order to yield a small set of high-quality features. From our experience we believe that this management procedure is the main reason for the high accuracy of the visual odometry.

Although SOFT-SLAM was developed primarily for EuRoC, which included highly dynamical motions in 6D in smaller indoor environments, thus often revisiting parts of the environment, it also performs well in outdoor scenarios as validated by the results on the KITTI dataset. In Fig. 14 we show profiles of ego-motion translational and angular velocities for two sequences, one from the KITTI and one from the EuRoC dataset, exhibited by the vehicle and UAV, respectively. The motion is quite diverse, for example, on the KITTI dataset the vehicle can exhibit high translational velocity, while on the EuRoC dataset UAV commonly exhibits more dynamic in angular rates than translational velocities. This can be seen in Table 5 where in comparison to ORB-SLAM2 and LSD-SLAM, it achieves better or equal performance on two-thirds of the error measures. This leads us to the conclusion that SOFT-SLAM is a highly efficient, versatile, and accurate stereo visual SLAM solution.

The drawback of SOFT-SLAM is that it does not apply a probabilistically rigorous treatment of the uncertainties in the system, which would certainly lead to a mathematically cleaner approach to the problem and even better performance. Furthermore, we do not perform map optimization—the output of our system is an octomap, whose update can be computationally time consuming, so in some sense SOFT-SLAM can be seen as a stereo vision localization architecture that outputs an octomap of the environment. It could also be argued that the selected features are not invariant to rotation, thus could impede loop-closing performance due to weaker invariance to viewpoint. From our experience on the EuRoC and KITTI datasets, and from the EuRoC Challenge 3 the loop-closing occurs frequently enough to have state-of-the-art performance; however, the features do not possess invariance properties of ORB features and, unlike ORB-SLAM2, we did not aim at offering also a solution to the kidnapped robot relocalization problem.

7 Conclusion

In this paper we have presented a stereo visual pose graph SLAM solution for small-scale high-speed UAVs. The main motif for the work proposed in this paper was to develop a highly accurate localization algorithm able to run in real-time and to build a dense map of the environment in order to participate in the EuRoC Challenge 3 targeting airborne inspection of industrial facilities. Given that, the development of the module necessitated an economical approach to the problem. The first part of the presented approach was SOFT, stereo visual odometry based on careful feature selection and tracking—it is also the foundation of the proposed SLAM framework and currently ranks first on the KITTI dataset. Furthermore, the odometry is completely independent of the mapping and can run at constant-time—the only information shared between the odometry and mapping is the estimated drift from the map origin.

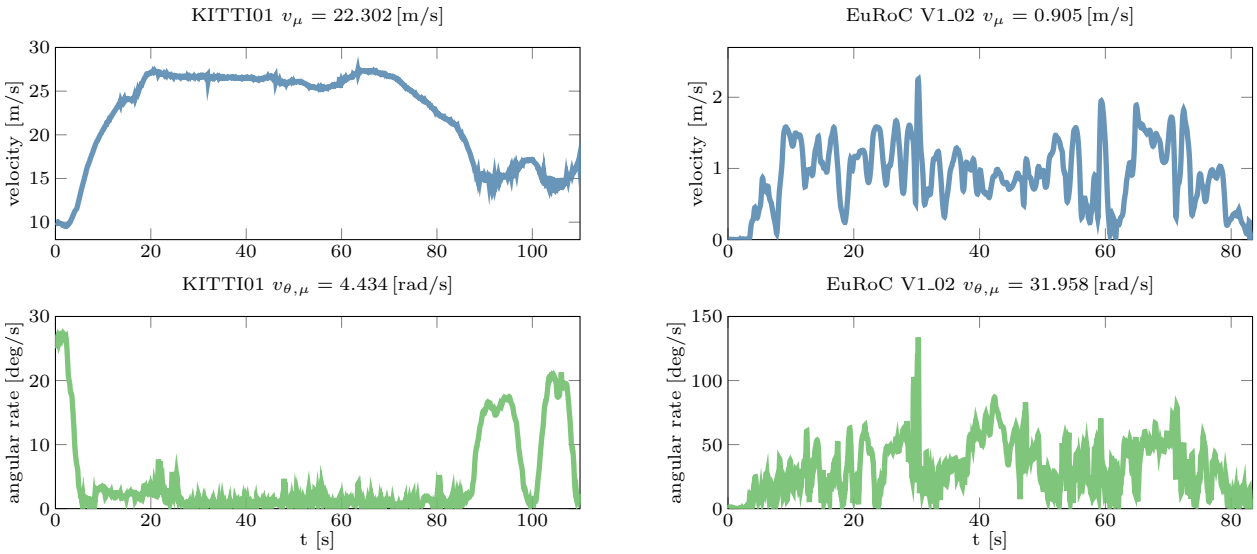

Figure 14: Ego-motion velocity profiles from KITTI01 (left) and EuRoC V1_02 (right) sequences.

The mapping part is based on a sparse set of keyframes, although we slightly misuse the term. Keyframes are not used in estimating depth from relative motion, thus our keyframes only serve as pose nodes also containing the pertaining sparse features. Keyframes are linked with relative pose constraints which are fed to the g2o optimization algorithm once loop-closing is detected. We continuously search for loop-closing using the same features as in the odometry. As in the ORB-SLAM2 case, this makes the approach computationally less intensive since features are already precomputed. A heuristic procedure is used for keyframe management in order to ensure adequate distribution of keyframes in the environment. The resulting solution, dubbed SOFT-SLAM, achieves constant-time execution running on two threads at 20 Hz on a UAV onboard computer used in EuRoC, thus leaving enough processing power for model-predictive control, sensor fusion, and other critical algorithms. In the EuRoC Challenge 3, Stage IIa–Benchmarking, Task 2, our UAV running SOFT-SLAM had the highest localization score among the five participating teams. Furthermore, in the paper we also present an exhaustive comparison of SOFT-SLAM with ORB-SLAM2 and LSD-SLAM on the EuRoC and KITTI datasets. The results demonstrate equal or better performance on the majority of the datasets sequences, thus leading us to the conclusion that SOFT-SLAM is a highly accurate and efficient vision-based SLAM solution.

Acknowledgments

This work has been supported from the Unity Through Knowledge Fund under Grant 24/15 (Cooperative Cloud based Simultaneous Localization and Mapping in Dynamic Environments, cloudSLAM) and the Eu-

ropean Robotics Challenges (EuRoC) funded from the European Union's Seventh Framework Programme under Grant no. 608849.

References

Ascending Technologies GmbH (Accessed February 1, 2017). `www.asctec.de/en/`.

Badino, H., Yamamoto, A., and Kanade, T. (2013). Visual odometry by multi-frame feature integration. In *IEEE International Conference on Computer Vision Workshops*, pages 222–229.

Bailey, T. and Durrant-Whyte, H. (2006a). Simultaneous localization and mapping (SLAM): Part I. *IEEE Robotics and Automation Magazine*, 13(2):99–108.

Bailey, T. and Durrant-Whyte, H. (2006b). Simultaneous localization and mapping (SLAM): Part II. *IEEE Robotics and Automation Magazine*, 13(3):108–117.

Bay, H., Tuytelaars, T., and Van Gool, L. (2006). SURF: Speeded up robust features. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3951 LNCS:404–417.

Briggs, B., McLauchlan, P. F., Harteley, R. I., and Fitzgibbon, A. W. (1999). Bundle adjustment — a modern synthesis. In *International Workshop on Vision Algorithms*, pages 298–372.

Buczko, M. and Willert, V. (2016a). Flow-decoupled normalized reprojection error for visual odometry. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 1161–1167.

Buczko, M. and Willert, V. (2016b). How to distinguish inliers from outliers in visual odometry for high-speed automotive applications. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 478–483.

Burri, M., Nikolic, J., Gohl, P., Schneider, T., Rehder, J., Omari, S., Achtelik, M. W., and Siegwart, R. Y. (2016). The EuRoC micro aerial vehicle datasets. *The International Journal of Robotics Research*, 35(10):1157–1163.

Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Reid, I. D., and Leonard, J. J. (2016). Past, present, and future of simultaneous localization and mapping: towards the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332.

Calonder, M., Lepetit, V., Strecha, C., and Fua, P. (2010). BRIEF: Binary robust independent elementary features. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6314 LNCS(PART 4):778–792.

Cvišić, I. and Petrović, I. (2015). Stereo odometry based on careful feature selection and tracking. In *European Conference on Mobile Robots (ECMR)*, page 6.

Dam, E. B., Koch, M., and Lillholm, M. (1998). Quaternions, interpolation and animation. Technical report, Department of Computer Science, University of Copenhagen.

Davison, A. J. (2003). Real-time simultaneous localisation and mapping with a single camera. In *International Conference on Computer Vision (ICCV)*, pages 1403–1410.

Davison, A. J., Reid, I. D., Molton, N. D., and Stasse, O. (2007). MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067.

Deigmoeller, J. and Eggert, J. (2016). Stereo visual odometry without temporal filtering. In *German Conference on Pattern Recognition (GCPR)*, pages 166–175. Springer.

Eade, E. and Drummond, T. (2009). Edge landmarks in monocular SLAM. *Image and Vision Computing*, 27(5):588–596.

Engel, J., Sch, T., and Cremers, D. (2014). LSD-SLAM: Direct monocular SLAM. In *European Conference on Computer Vision (ECCV)*, pages 834–849.

Engel, J., Stückler, J., and Cremers, D. (2015). Large-scale direct SLAM with stereo cameras. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 141–148.

Engel, J., Sturm, J., and Cremers, D. (2013). Semi-dense visual odometry for a monocular camera. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1449–1456.

Estrada, C., Neira, J., and Tardós, J. (2005). Hierarchical SLAM: Real-time accurate mapping of large environment. *IEEE Transactions on Robotics*, 21(4):588–596.

Forster, C., Carlone, L., Dellaert, F., and Scaramuzza, D. (2016a). On manifold preintegration for real-time visual–inertial odometry. *IEEE Transactions on Robotics*, page 21.

Forster, C., Pizzoli, M., and Scaramuzza, D. (2014). SVO: Fast semi-direct monocular visual odometry. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 15–22.

Forster, C., Zhang, Z., Gassner, M., Werlberger, M., and Scaramuzza, D. (2016b). SVO: Semi-direct visual odometry for monocular and multi-camera systems. *IEEE Transactions on Robotics*, pages 1–17.

Fraundorfer, F. and Scaramuzza, D. (2012). Visual odometry part II: matching, robustness, optimization, and applications. *IEEE Robotics & Automation Magazine*, 19(2):78–90.

Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, 32(11):1231–1237.

Geiger, A., Ziegles, J., and Stiller, C. (2011). StereoScan: Dense 3D Reconstruction in Real-time. In *Intelligent Vehicles Symposium (IV)*.

Harris, C. and Stephens, M. (1988). A combined corner and edge detector. In *Procedings of the Alvey Vision Conference 1988*, pages 147–151.

Hartley, R. and Zisserman, A. (2003). *Multiple View Geometry in Computer Vision*. Cambridge University Press.

Hirschmüller, H. (2011). Semi-global matching motivation, developments and applications. In *Proceedings of the 53rd Photogrammetric Week*, pages 173–184.

Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., and Burgard, W. (2013). OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*. Software available at `http://octomap.github.com`.

Kerl, C., Sturm, J., and Cremers, D. (2013). Dense visual SLAM for RGB-D cameras. In *IEEE International Conference on Intelligent Robots and Systems*, pages 2100–2106.

Kitt, B., Geiger, A., and Lategahn, H. (2010). Visual odometry based on stereo image sequences with RANSAC-based outlier rejection scheme. In *Intelligent Vehicles Symposium (IV)*.

Klein, G. and Murray, D. (2007). Parallel tracking and mapping for small AR workspaces. In *EEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*.

Konolige, K. and Agrawal, M. (2008). FrameSLAM: From bundle adjustment to real-time visual mapping. *IEEE Transactions on Robotics*, 24(5):1066–1077.

Konolige, K., Agrawal, M., and Solà, J. (2010). Large-scale visual odometry for rough terrain. In *Springer Tracts in Advanced Robotics*, volume 66, chapter Robotics R, pages 201–212.

Kümmerle, R., Grisetti, G., Rainer, K., Strasdat, H., Konolige, K., and Burgard, W. (2011). g2o: A general framework for graph optimization. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3607–3613.

Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R., and Furgale, P. (2015). Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3):314–334.

Lowe, D. G. (1999). Object recognition from local scale-invariant features. *Proceedings of the Seventh IEEE International Conference on Computer Vision*, 2(8):1150–1157.

Mourikis, A. I. and Roumeliotis, S. I. (2007). A multi-state constraint Kalman filter for vision-aided inertial navigation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3565–3572.

Mur-Artal, R., Montiel, J. M. M., and Tardos, J. D. (2015). ORB-SLAM: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163.

Mur-Artal, R. and Tardos, J. D. (2016). ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *arXiv*.

Newcombe, R. A., Lovegrove, S. J., and Davison, A. J. (2011). DTAM: Dense tracking and mapping in real-time. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2320–2327.

Nikolic, J., Rehder, J., Burri, M., Gohl, P., Leutenegger, S., Furgale, P. T., and Siegwart, R. (2014). A synchronized visual-inertial sensor system with FPGA pre-processing for accurate real-time SLAM. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 431–437.

Nister, D., Naroditsky, O., and Bergen, J. (2004). Visual odometry. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*.

Persson, M., Piccini, T., Felsberg, M., and Mester, R. (2015). Robust stereo visual odometry from monocular techniques. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 686–691.

Pire, T., Fischer, T., Civera, J., De Cristoforis, P., and Berlles, J. J. (2015). Stereo parallel tracking and mapping for robot localization. In *IEEE International Conference on Intelligent Robots and Systems*, volume 2015-Decem, pages 1373–1378.

Pizzoli, M., Forster, C., and Scaramuzza, D. (2014). REMODE: Probabilistic, monocular dense reconstruction in real time. In *IEEE International Conference on Robotics and Automation*, pages 2609–2616.

Rosten, E. and Drummond, T. (2006). Machine learning for high-speed corner detection. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3951 LNCS:430–443.

Rublee, E., Rabaud, V., and Konolige, K. (2011). ORB: an efficient alternative to SIFT or SURF. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2564–2571.

Scaramuzza, D. and Fraundorfer, F. (2011). Visual odometry: Part I: The First 30 Years and Fundamentals. *IEEE Robotics and Automation Magazine*, 18(4):80–92.

Shi, J. and Tomasi, C. (1994). Good features to track. In *IEEE International Conference on Computer Vision and Patter Recognition (CVPR)*.

Smith, P., Reid, I., and Davison, A. (2006). Real-time monocular SLAM with straight lines. In *Proceedings of the British Machine Vision Conference 2009*, pages 17–26.

Sturm, J., Engelhard, N., Endres, F., Burgard, W., and Cremers, D. (2012). A benchmark for the evaluation of RGB-D SLAM systems. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 573–580.

Usenko, V., Engel, J., Stückler, J., and Cremers, D. (2016). Direct visual-inertial odometry with stereo cameras. In *International Conference on Robotics and Automation (ICRA)*, pages 1885–1892.