DE GRUYTER OPEN Acta Univ. Sapientiae, Informatica **8,** 2 (2016) 186–215

DOI: 10.1515/ausi-2016-0009

Monocular indoor localization techniques for smartphones

Gergely HOLLÓSI
Budapest University of Technology and Economics
email: hollosi@tmit.bme.hu

Csaba LUKOVSZKI
Budapest University of Technology and Economics
email: lukovszki@tmit.bme.hu

István MOLDOVÁN
Budapest University of Technology and Economics
email: moldovan@tmit.bme.hu

Sándor PLÓSZ
Budapest University of Technology and Economics
email: plosz@tmit.bme.hu

Frigyes HARASZTOS
Flaxcom Holding co. ltd.
email: harasztos.frigyes@flaxcom.hu

Abstract. In the last decade huge research work has been put to the indoor visual localization of personal smartphones. Considering the available sensor capabilities monocular odometry provides promising solution, even reflecting requirements of augmented reality applications. This paper is aimed to give an overview of state-of-the-art results regarding monocular visual localization. For this purpose essential basics of computer vision are presented and the most promising solutions are reviewed.

Computing Classification System 1998: F.1.1, G.1.3, I.2.10, I.4.1, I.4.8, I.4.9, K.8.1
Mathematics Subject Classification 2010: 68-02, 68U10, 68T45
Key words and phrases: computer vision, visual odometry, SLAM, indoor localization

1 Introduction

Due to the increasing capabilities and penetration, more and more applications are available on smart-phones and assist our everyday activities. In the last decade huge research work was put to the indoor location-based applications, from which the augmented reality based applications demand the highest requirements mostly expressed in accuracy, real-time and seamless localization. Based on the sensors that are available in recent smartphones and their computational and storage capabilities, a real-time implementation of monocular visual relative pose estimation seems to be a key to achieve the overall goal.

Besides, this topic presents great research interest, and high effort has been put on providing scalable and accurate solutions to satisfy the real-time requirements. Traditionally, the problem of visual pose estimation is discussed as Structure from Motion (SFM) [34] [17] problem, where the main goal is the off-line reconstruction of a 3D structure from pictures taken from different viewpoints. During the reconstruction process the viewpoints of the camera are also calculated, but problem formulation does not focus on relative pose estimation. The family of SLAM (Simultaneous Localization and Mapping) algorithms focuses on the environment modelling (map building) and relative camera pose estimation simultaneously [9]. To overcome the real time and accuracy requirements these solutions induced the PTAM (Parallel Tracking and Mapping) algorithm [22]. In the meantime, the problem also targeted by another application field, the odometry. The original requirement of the monocular Visual Odometry (VO) [50] [14] was to accurately determine the relative pose of a rover.

In this paper authors attempt to give a theoretical overview of the monocular odometry problem and its solutions. Also, some of the implementations are emphasized that seem to able to cope with the strict requirements even in mobile environments.

During the discussion, authors focus on capabilities of recent smartphones. Common smartphones are equipped with a thin-lens perspective camera, which can be modelled with an ideal pin-hole model [20], and they are also equipped with IMU (Inertial Measurement Unit) integrating gyroscope and accelerometer. Reasonable capacity for storage and processing. Regarding the motion of the device the following discussion suggests 6DOF (degree-of-freedom).

2 Theoretical background

Monocular visual odometry tries to determine pose and location of a device mostly using visual perception aided by couple of auxiliary sensors (e.g. gyro-

scope or acceleration sensor). The common implementation of visual perception is a monocular camera which provides continuous stream of frames at a variable or uniform time instants.

2.1 Projection model

The camera has a couple of internal parameters which are typically fixed and known a priori (e.g. by calibration). The most important characteristic of the camera is the projection model which projects three dimensional world points onto the image:

$$\mathbf{u} = \pi(\mathbf{p}_{\mathcal{C}}) \tag{1}$$

where $\mathbf{p}_{\mathcal{C}} = \left[x_{\mathcal{C}}, y_{\mathcal{C}}, z_{\mathcal{C}}\right]$ is a three dimensional world point in the reference frame of the camera, $\mathbf{u} = \left[x, y\right]$ is the projected point and $\pi()$ is the projection model. It is essential to mention that in case of monocular systems the $\pi()$ projection model is invertible only when the depth $d_{\mathbf{u}}$ of the model point is known:

$$\mathbf{p}_{\mathcal{C}} = \pi^{-1}(\mathbf{u}, d_{\mathbf{u}}) \tag{2}$$

We can see that monocular systems have the big drawback of loosing the depth information while recording frames.

In practice projection model is considered to be linear in homogeneous space , i.e. it can be represented by a matrix product (commonly referred to the pinhole camera model). Let $\mathbf{X}_{\mathcal{C}} = \left[X, Y, Z, 1\right]^T$ be the homogeneous coordinates of a three dimensional point in the reference frame of the camera. In this case the projection model can be expressed with a K intrinsic camera matrix:

$$\mathbf{x} = K(f)\left[I_{3\times3}|0_{3\times1}\right]\mathbf{X}_{\mathcal{C}} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\mathbf{X}_{\mathcal{C}} \tag{3}$$

where f is the focal length of the camera and $\mathbf{x} = \left[\lambda x, \lambda y, \lambda\right]^T$ are the homogeneous coordinates of the two dimensional projection. It is easy to see that the projection model is not invertible.

To represent camera movement in world frame we assign a T_k rigid-body transformation to each frame I_k at k time instants which contains orientation ($\mathbf{R}_k$) and location ($\mathbf{C}_k$) of the camera. The transformation can be expressed as a 4×4 matrix as

$$T_k = \begin{bmatrix} \mathbf{R}_k & \mathbf{C}_k \\ 0 & 1 \end{bmatrix} \tag{4}$$

A fixed world point $\mathbf{X} = \left[X, Y, Z, 1\right]$ can be projected at the k-th image frame as

$$\mathbf{x}_k = K(f)\left[I|0\right]T_k^{-1}\mathbf{X} = K(f)\left[R_k^{-1}|-R_k^{-1}\mathbf{C}_k\right]\mathbf{X} = K(f)P_k^e\mathbf{X} \tag{5}$$

where P_k^e is commonly called as the extrinsic matrix describing the world-to-camera transformation. Eq. (5) is the most basic and substantial constraint in the monocular visual odometry systems.

The goal of the monocular visual odometry algorithms is to determine the P_k^e extrinsic camera matrices or the T_k rigid-body transformation of the cameras mainly based on (but not exclusively) the visual information encoded in frames.

2.2 Projection distortion

An accurate algorithm must take into consideration that the projection model of the classical pinhole camera is only an approximation. Real cameras always have some non-linear distortion which is basically modelled as *radial* distortion, however, other distortion models also exist (i.e. tangential distortion)[4]. Radial distortion depends on the radial distance from the radial distortion centre (typically the principal point) and it is represented as an arbitrary function:

$$\hat{x} = x_c + L(r)(x - x_c) \qquad \hat{y} = y_c + L(r)(y - y_c) \tag{6}$$

where $r^2 = (x - x_c)^2 + (y - y_c)^2$ is the radial distance and x_c, y_c are the radial centres (commonly considered as zero). In practice, $L(r)$ is represented as a Taylor-series

$$L(r) = 1 + \kappa_1 r + \kappa_2 r^2 + \kappa_3 r^3 + \cdots \tag{7}$$

where κ_i are the radial distortion coefficients. In practice only the lower coefficients $(\kappa_1, \kappa_2, \kappa_3)$ are used.

2.3 Visual information retrieval

Visual odometry solutions are based on visual information encoded in the sequence of image frames. We can distinguish two widespread methods: intensity based *direct* methods and *feature* based methods.

2.3.1 Direct methods

In general, direct methods uses the $I_k(\mathbf{u})$ intensity map of the image, which represents the brightness of the image pixel coordinate or – rarely – the RGB

vector. The intensity map can be either quantized (i.e. pixel accuracy) or continuous (i.e. sub-pixel accuracy), however, the latter requires some kind of filtering or interpolating algorithm that in some cases can cause information loss.

2.3.2 Feature detection

Feature based methods works on point projections using *feature detection* and *feature extraction* algorithms that are able to detect and match the same points on different images without preliminary geometric knowledge. This way, visual odometry solutions are simplified to use only projections of real 3D landmarks. The efficiency of these algorithms can be measured by their invariance and speed. Invariance means that the detector can detect features which can be successfully matched even if the feature is rotated, scaled or suffered other transformations (e.g. affine transformation). There are a couple of such algorithms overviewed in [5], from which the most widely used are the Harris detector [18], the Scale-invariant feature transform (SIFT) that bases on Laplacian of Gaussian filters [36], the Maximally stable extremal regions (MSER) [37], the Features from accelerated segment test (FAST) and Oriented FAST and Rotated BRIEF (ORB) [48]. Considering the overall requirements SIFT is the most promising, however, due to its high complexity, strict constraints should be taken into account during its application in mobile environments.

3 Feature based solutions

Feature based solutions have the attribute to detect features on the frames first then match them to the previous frame resulting in projection *tracks* over a couple of sequential frames. These tracks can then be used to compute the geometry of the scene and to recover the camera translations and orientations. This method utilizes only point geometry models and correspondences, this way the well established framework of multiple view geometry can be applied [20].

3.1 Theory

The most important term here is pose estimation, which is the process of estimating the extrinsic (and sometimes the intrinsic) matrix from point correspondences. Depending on the point pairs we can establish two types of pose estimation: in case of 3D-2D point pairs (i.e. the world points and their pro-

jections) it is called *absolute pose estimation* and in case of 2D-2D point pairs (i.e. the projection pairs on two images) we call it *relative pose estimation*.

3.1.1 PnP problem

The absolute pose estimation problem is generally called Perspective-n-Point (PnP) problem, which has a couple of methods presented. The classical method for $n > 6$ point pairs is the DLT (Direct Linear Transform) method, but it is known to be unstable and requires the camera calibration [1]. For 5 or 4 points the [55] uses a polynomial technique which allows it to work well even in case of coplanar points. The EPnP solution is accurate for an arbitrary $n \geq 4$ point pairs and can handle planar and non-planar cases [26]. The P3P leads to to finite number of solutions using only three point pairs as the smallest subset of points pairs [24]. The P3P solution has the advantage of using only three points in a RANSAC framework to eliminate outlier point pairs decreasing the required number of iterations.

3.1.2 Random Sample Consensus

Since feature matching is prone to result false matches, a method is required to overcome this issue. It is common in image processing to use the minimal sample set to recover model parameters and classify samples to inliers and outliers. The most noted algorithm is the Random Sample Consensus (RANSAC) method, which is widely used in further solutions [12].

3.1.3 Relative pose estimation

The basic terms in relative pose estimation are the *fundamental matrix* and the *essential matrix*, both can be computed from 3D feature projection pairs. The fundamental matrix is a 3×3 matrix ($\mathbf{F}$) satisfying $\mathbf{x}'^{\mathsf{T}}\mathbf{F}\mathbf{x} = 0$, where projections ($\mathbf{x}$ and $\mathbf{x}'$) are of the same world point in two different images. The essential matrix ($\mathbf{E}$) uses normalized image coordinates, so it can be computed from the intrinsic camera matrix ($\mathbf{K}$) and the fundamental matrix as $\mathbf{E} = \mathbf{K}'^{\mathsf{T}}\mathbf{F}\mathbf{K}$. The essential matrix is applicable to recover the pose of the cameras by decomposition [20]. A lot of methods are known to determine the relative pose of the cameras: the 8 point algorithm [19], the 7 point algorithm [20], 6 point algorithm [45] and 5 point algorithms [27][42]. It is essential to mark that these algorithms differ in handling degenerate configurations (i.e. coplanar objects or cylinder containing the projection centres) and are unable to recover the scale of the set-up.

3.1.4 Bundle adjustment

The fundamental algorithms, like the relative and absolute pose estimation and triangulation, find the right solution only in case of noiseless measurements. Otherwise, they minimize the algebraic error which has no physical meaning. It can be proven that the maximum likelihood (ML) solution of these problems are the minimization of *re-projection error*. If we have N cameras and M points in space, then we can assign a $\theta_n(K_n, T_n, \pi_n)$ projection model to each camera that contains the projection (π_n), distortions (K_n) and rigid body transformation (T_n) of the camera (i.e. intrinsic and extrinsic behaviour). For a $\mathbf{p}_m$ point in space the projection for the camera n yields to $\mathbf{u}_{m,n} = \theta_n(\mathbf{p}_m)$. If pixel measurements are $\bar{\mathbf{u}}_{n,m}$, then the optimization of re-projection error equals the expression

$$\arg\min_{\theta_n, \mathbf{p}_m} \sum_{n,m} |\bar{\mathbf{u}}_{n,m} - \theta_n(\mathbf{p}_m)|^2 \tag{8}$$

meaning minimization of the euclidean-distance between the measurements and the re-projected points.

As it is obvious from Eq. (8), the re-projection error is not linear so we need an iterative Newton-like solution to solve the minimization problem. The process of solving Eq. (8) with Levenberg-Marquardt iteration is specially called *bundle adjustment* [56]. Bundle adjustment is widely used in SLAM, SFM and odometry problems to refine a coarse solution or co-optimize the map of landmarks and camera poses calculated before.

It is worth to mention that the special form of the projection equation yields to a sparse matrix, which can be utilized to speed up the bundle adjustment and relax the memory and processing requirements. This method is called *sparse bundle adjustment* [35][20].

3.2 Implementations

The solutions and implementations use the algorithms shown above but combine them in quite different ways.

3.2.1 PTAM

SLAM methods have the controversial problem of running at real-time speed while building an accurate map by a slow non-linear optimization process (i.e. bundle adjustment). Parallel Tracking and Mapping (PTAM) solves this problem by running two threads: one for the real-time tracking and one for the

map building [22]. PTAM was designed to work in small-scale, e.g. to provide desk-scale augmented reality. PTAM has several extensions implemented, like new initializer based on homography or a re-localiser [23].

PTAM detects FAST features on a scale pyramid to provide scale invariance and uses these feature points to recover the geometry. PTAM applies the 5-point algorithm to recover the initial camera relative pose (i.e. the fundamental matrix) and to construct the initial map. Hence, the process of PTAM odometry can be briefly described as follows:

- Tracking runs on its own thread and starts by detecting FAST features. A motion model is used to estimate the camera a-priori pose followed by projecting map points onto the image to detect feature matches and finally camera pose is refined from all the matches.

- The mapping thread selects key-frames at regular intervals based on a couple of conditions, then the thread triangulates new points and registers new projections. To refine the map, PTAM applies local and global bundle adjustments periodically.

PTAM solution is capable to track the camera pose accurately and in real-time thanks to the decoupled tracking and mapping processes, but its performance is limited by the number of landmarks registered in the map. This way PTAM is suitable only for small workspaces. One of the drawbacks of PTAM is the simple initialization process of the 5-point algorithm which is sensitive to planar degeneracy. It is worth to mention that PTAM does not employ any methods to recover the accumulated odometry error (i.e. loop closing).

3.2.2 ORB-SLAM

ORB-SLAM realizes a rather complex visual odometry solution, however, it is based basically on feature detection and point geometry [40]. As its name suggests it uses ORB features to gather image information and provides odometry and 3D reconstruction simultaneously. Besides, ORB-SLAM provides relocalization and loop closing capabilities in order to make the process more accurate.

ORB-SLAM works pretty much like PTAM by running three threads parallel to provide real-time odometry. The *tracking* thread is responsible for real-time motion estimation by detecting ORB features and camera pose recovery. The *local mapping* thread calculates the 3D reconstruction of the map in the background for every key-frame chosen by the tracking thread. The

loop closing thread is watching for map points to reoccur using bag of words model, and when it finds one the loop closing corrects the loop by similarity transformation.

ORB-SLAM applies ORB feature detection as it provides rotation and scale invariance. It is fast enough to maintain real-time performance, while it is suitable for both large-scale (i.e. distant frames) and small-scale (i.e. subsequent frames) matching. The great innovation in ORB SLAM is that it uses ORB for every part of the process: tracking, mapping and loop closing are executed on ORB features. ORB-SLAM system provides visual odometry as follows:

1. The ORB-SLAM starts with an automatic initialization method to retrieve the initial pose and map by extracting the ORB features, matching them and computing corresponding fundamental matrix and homography (i.e. the two dimensional projective transformation) in the same time. It computes a score to both the homography and the fundamental matrix as:

$$S_M = \sum_i \left(\rho_M\left(d_{cr}^2(\mathbf{x}_c^i, \mathbf{x}_r^i, M)\right) + \rho_M(d_{rc}^2(\mathbf{x}_c^i, \mathbf{x}_r^i, M))\right)$$
$$\rho_M(d^2) = \begin{cases} \Gamma - d^2 & \text{if} \quad d^2 < T_M \\ 0 & \text{if} \quad d^2 \geq T_M \end{cases} \tag{9}$$

   where M is the model (H for homography and F for fundamental matrix), d_{cr}^2 and d_{rc}^2 are the symmetric transfer errors, T_M is the outlier rejection threshold based on the χ^2 test at 95% ($T_H = 5.99$, $T_F = 3.84$, assuming a standard deviation of 1 pixel in the measurement error). Γ is a score compensating constant. ORB-SLAM recover initial pose and map from homography if

$$\frac{S_H}{S_H + S_F} > 0.45 \tag{10}$$

   Otherwise, it uses the fundamental matrix. After recovering pose and map, it starts a non-linear optimization (bundle adjustment) to refine the initial model.

2. After map initialization tracking tries to match ORB features of the current frame to the ORB features of the previous frame through a guided search employing a constant velocity model. The pose is then refined by non-linear optimization. After pose estimation ORB-SLAM tries to re-project the map onto the frame recovering more feature matches. The

last step is the key-frame decision which judges that the current frame should be passed to the local mapping thread. This step utilizes a couple of complex conditions.

3. Parallel to tracking, every key-frame is processed to provide a consistent map that is able to refine the tracking process and provides input to loop closing. Briefly, local mapping triangulates new point candidates having passed a restrictive map point culling test and uses local bundle adjustment to minimize re-projection error. To maintain compact reconstruction ORB-SLAM removes redundant key-frames

4. Loop closing happens parallel to tracking and mapping and uses bag of words representation and co-visibility information to detect loop candidates [43]. In case of loop detection it computes the similarity transformation accumulated while tracking to distributes the error along the whole path.

ORB-SLAM has been proven to be a robust and accurate solution even in large-scale areas and can successfully track ad-hoc movements while providing stable map initialization in case of a lost track. ORB-SLAM requires at least 20 frames per second to work well, which can hardly be satisfied using ORB feature detection on embedded devices like smartphones without exploiting massive GPU calculations.

4 Direct solutions

The principle behind direct solutions states that using the image intensities results in better odometry accuracy because it exploits all the information embedded in the frames while feature based solutions discard image information over feature points. The most important term of direct solutions is the *photo-consistency* discussed in the next section.

4.1 Photo-consistency theory

From a mathematical perspective, photo-consistency means that given two images I_1 and I_2, an observed point $\mathbf{p}$ by the two cameras yields to the same brightness in both images [21]:

$$I_1(\mathbf{u}) = I_2(\tau(\xi, \mathbf{u})) \tag{11}$$

where $\mathbf{u}$ is the projection of $\mathbf{p}$, $\tau(\cdot)$ is the *warping* function, which depends on $\pi(\cdot)$ (see Eq. (1)). The warping function maps a pixel coordinate from the first image to the second one given the camera motion ξ. Here, the motion ξ can be represented in any minimal representation (e.g. twist coordinates). Given the residual function for any $\mathbf{u}$ point in the Ω image domain

$$r(\xi, \mathbf{u}) = I_2(\tau(\xi, \mathbf{u})) - I_1(\mathbf{u}) \tag{12}$$

which depends on ξ and assuming independent pixel noise, the Maximum Likelihood (ML) solution is a classical minimization problem:

$$\xi_{ML} = \arg\min_{\xi} \int_{\Omega} r^2(\xi, \mathbf{u}) \, d\mathbf{u} \tag{13}$$

The problem is obviously non-linear, so the common solution is to run iterative minimization algorithms like Newton-Gauss method over a discretised image. To speed up the integration process, the integration can be run over a couple of selected patches instead of every pixels in the images.

4.2 DTAM

Dense Tracking and Mapping (DTAM) uses the photo-consistency theory in a special way to provide dense maps and real-time visual odometry [41]. The main idea behind dense mapping is to sum the photometric error along a ray from the camera centre and find the d distance that minimizes the sum, thus finding the depth parameter for that pixel. The summing is made along a couple of short baseline frames $m \in I(r)$ for a r reference frame:

$$C_r(\mathbf{u}, d) = \frac{1}{|I(r)|} \sum_{m \in I(r)} \|r_r(I_m, \mathbf{u}, d)\|_1 \tag{14}$$

where $\|\cdot\|_1$ is the L1 norm and the photometric error is

$$r_r(I_m, \mathbf{u}, d) = I_m(\tau(d, \mathbf{u}_i)) - I_r(\mathbf{u}_i) \tag{15}$$

Note that the only change in the equation is the parameter d. DTAM showed that minimizing the cost yields to a correct estimation of pixel depth which can be used to build dense maps.

The tracking part of the DTAM solution provides 6DOF estimation and basically happens the same way as shown in Eq. (13) with a couple of extensions to provide robust tracking with occlusion detection.

DTAM is robust and accurate visual odometry solution with excellent mapping capabilities. It is not only capable of handling occlusions but can track the movements even in case of total lost in focus and keep on tracking even for fast and random movements. The only drawback of the solution is real-time performance requires huge computing capacity and massive GPU utilization.

4.3 LSD-SLAM

Large-Scale Direct Monocular SLAM (LSD-SLAM) uses direct methods combined with a probabilistic approach to track camera movements and build dense map real-time [10]. LSD-SLAM has scale-aware image alignment algorithm which estimate directly the similarity transformation between two key-frames to provide scale consistent maps and odometry.

The main process of the LSD-SLAM is as follows: at every new frame it tries to estimate the movement relative to the current key-frame, then it decides whether the actual key-frame should be replaced by the new frame. In case of replacing, it initializes a new depth map, otherwise it propagates the depth map of the current key-frame. At every key-frame replacement LSD-SLAM runs map optimization, which is essential to create accurate dense maps.

LSD-SLAM uses image patches to recover pose around pixels with large intensity gradients. The tracking process is composed of two steps: estimation of rigid body transformation and depth map propagation. The former one is a weighted optimization of the variance-normalized photometric error

$$E_p(\xi_j) = \sum_{p \in \omega_{D_i}} \left\| \frac{r_p^2(\mathbf{u}, \xi_j)}{\sigma^2_{r_p(\mathbf{u},\xi_j)}} \right\|_\delta \tag{16}$$

for an existing key-frame and the new frame I_j. In the equation $r_p(\cdot)$ is the photometric error, σ_{r_p} is the variance of the photometric error and $\|\cdot\|_\delta$ means the Huber-norm. Apart from normalization by variance, this is a classical photometric error based odometry solution as in Eq. (13).

The biggest difference to other direct solutions is that the depth information for a key-frame is calculated in a probabilistic way, i.e. it is refined as new frames received. An inverse depth map and a depth map variance map is assigned to every key-frame selected by the LSD-SLAM process. The depth map is initialized with the depth map of the previous key-frame or with a random depth map if no key-frame exists. For each new frame the depth map is propagated as in [11], namely if the inverse depth for a pixel was d_0 then

for the new frame it is approximated as

$$\begin{aligned} d_1 &= (d_0^{-1} - t_z)^{-}1 \\ \sigma_{d_1}^2 &= \left(\frac{d_1}{d_0}\right)^4 \sigma_{d_0}^2 + \sigma_p^2 \end{aligned} \tag{17}$$

where σ_p is the prediction uncertainty and t_z is the camera translation along the optical axis.

LSD-SLAM also contains solution for the problem of scale-drift over long trajectories, which is the major source of error in the family of SLAM solutions. LSD-SLAM thus aligns two differently scaled key-frames by incorporating the depth residual into the error function shown above. This method penalizes deviations in inverse depth between key-frames and helps to estimate the scaled transformation between them.

4.4 SVO

Fast Semi-Direct Monocular Odometry (SVO) is a great example of hybrid solutions for visual odometry using direct and featured based algorithms as well [13]. SVO combines the probabilistic approach of depth map with the computationally attractive feature based concept as the name suggests providing real-time odometry and sparse mapping.

The basic process of SVO is tracking and mapping that are implemented on parallel threads, i.e. calculating the movement trajectory at each frame real-time and select key-frames that can be used for mapping on the mapping thread. As the mapping thread uses features, bundle adjustment can be used to minimize re-projection error and construct accurate maps.

The tracking thread projects 3D points of the map onto the new frame and uses the vicinity of the projected points in the image to estimate the motion relative the previous frame by photometric error optimization. The pose is refined by aligning the frame to the whole map (using Lucas-Kanade algorithm [3]) then by local bundle adjustment to apply the epipolar constraints.

SVO is unique in the way that no depth map is computed but for each feature point on a key-frame a depth-filter is assigned that estimates the feature depth in a probabilistic way. First, the mapping thread decides whether a new frame a key-frame or not. Feature extraction is executed on new key-frames and to each feature a freshly initialized depth-filter is assigned. On inter-frames (i.e. not key-frames) the feature depth-filters are updated until they converge to the estimated value and the variance is small enough. Converged depth filter are converted to map points by triangulation.

Thanks to the feature based mapping process SVO has proven to be faster than other direct solutions, however, the result is a sparse map rather than a dense one. The depth filters are capable of detecting outlier measurement and the map is always consistent and correct because triangulation happens only when the filters converged. As the SVO uses couple of small patches around features to estimate motion, it is capable of running real-time as well.

5 Filter-based solutions

In real applications relative pose estimation should be seamless, which cannot be guaranteed by solutions based only on image processing. To overcome this requirement motion models are applied to estimate the camera state between visual pose estimations. The first reliable solution is MonoSLAM [8], which introduces an extrapolated motion model. MonoSLAM is thus applicable for smooth camera motion, but can cover only desk-scale local environment.

The most reasonable choice for motion estimation is to combine measurements of IMU with projections of real 3D landmarks of the local environment. The filter based family of visual odometry algorithms fuses inertial IMU measurements with visual feature observations. In these methods, the current camera pose and positions of visual landmarks are jointly estimated, sharing the same basic principles with camera-only localizations. These combined techniques can be categorized as *loosely coupled* and *tightly coupled* systems. In loosely coupled systems [46] [54] [57] inertial and camera measurements are processed separately before being fused as a single relative pose estimate, while tightly coupled systems process all the information together [44] [25]. However, loosely coupled systems limits computational complexity, in the following we are focused on tightly coupled techniques due to its ability to reach higher consistency between camera poses and map of landmarks.

5.1 Theory

The original relative pose estimation problem is hard due to its nature. The algorithms use a map containing visual information to localize, while relative pose is necessary to construct and update the visual map. The problem becomes even harder to solve if we consider the noise of the sensors. Various probabilistic methods are used to deal with the uncertainty introduced by measurement noise, Extended Kalman Filter (EKF), Particle Filter (PF), which are all based on Bayesian technique for random value estimation of the system state parameters, including camera locations and orienta-

tions at discrete time instants ($\mathbf{x}_k$) based on observations of the landmarks ($\mathbf{z}_k = \{\mathbf{z}_{ik}\}$) from a given camera viewpoint, in other words the map points ($\mathbf{m} = \{\mathbf{m}_1, \mathbf{m}_2, ..., \mathbf{m}_n\} = \mathbf{m}_{1:n}$), while the camera location is controlled independently of the system state by control parameters ($\mathbf{u}_k$). The problem of relative pose estimation is given then in the probabilistic form as follows [9].

$$\mathsf{P}(\mathbf{x}_k, \mathbf{m} | \mathbf{z}_{0:k}, \mathbf{u}_{0:k}, \mathbf{x}_0) \tag{18}$$

The calculation of position probability distribution is done iteratively starting from $\mathsf{P}(\mathbf{x}_{k-1}, \mathbf{m} | \mathbf{z}_{1:k-1}, \mathbf{u}_{1:k-1}, \mathbf{x}_0)$ with input of the actual control $\mathbf{u}_k$ and measurement $\mathbf{z}_k$ using Bayesian Theorem. The computation from one side requires the *state transition* or *motion model* for the camera that describes the new state regarding the control input.

$$\mathsf{P}(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{u}_k) \tag{19}$$

Secondly, the *observation model* describes the probability of making and observation $\mathbf{z}_k$ when a camera and landmark locations are known.

$$\mathsf{P}(\mathbf{z}_k | \mathbf{x}_k, \mathbf{m}) \tag{20}$$

The iteration is then implemented in a standard two-step recursive process. The first step is the *time update* that *propagates* state in time according to the control.

$$\mathsf{P}(\mathbf{x}_k, \mathbf{m} | \mathbf{z}_{0:k-1}, \mathbf{u}_{0:k}, \mathbf{x}_0) = \int \mathsf{P}(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{u}_k) \cdot \mathsf{P}(\mathbf{x}_{k-1}, \mathbf{m} | \mathbf{z}_{0:k-1}, \mathbf{u}_{0:k-1}, \mathbf{x}_0) \, \mathrm{d}\mathbf{x}_{k-1} \tag{21}$$

The second step conveys the *measurement* or *update* when based on the actual state-dependent measurements *correction* is done on the actual state.

$$\mathsf{P}(\mathbf{x}_k, \mathbf{m} | \mathbf{z}_{0:k}, \mathbf{u}_{0:k}, \mathbf{x}_0) = \frac{\mathsf{P}(\mathbf{z}_k | \mathbf{x}_k, \mathbf{m}) \mathsf{P}(\mathbf{x}_k, \mathbf{m} | \mathbf{z}_{0:k-1}, \mathbf{u}_{0:k}, \mathbf{x}_0)}{\mathsf{P}(\mathbf{z}_k | \mathbf{z}_{0:k-1}, \mathbf{u}_{0:k})} \tag{22}$$

The above principle is implemented in various ways assuming different terms on the model and random value distributions.

5.2 The IMU model

In practice, gyroscope and accelerometer measurements can be used to estimate actual relative pose based on the kinematic model. This is done during

the timely filter state propagation. All these measurements are stressed with local measurement noise, distortion and biases. The accelerometer measures actual acceleration ($\mathbf{a}_{\mathrm{m},\mathcal{I}} \in \mathbb{R}^3$) in the IMU orientation frame ($\mathcal{I}$) and its model can be formulated as follows.

$$\mathbf{a}_{\mathrm{m},\mathcal{I}}(\mathrm{t}) = \mathbf{T}_{\mathrm{a}}\mathbf{R}_{\mathcal{I}\mathcal{G}}(\mathrm{t})(\mathbf{a}_{\mathcal{G}}(\mathrm{t}) - \mathbf{g}) + \mathbf{a}_{\mathrm{b}}(\mathrm{t}) + \mathbf{a}_{\mathrm{n}}(\mathrm{t}) \tag{23}$$

where $\mathbf{a}_{\mathcal{G}}$ is the real acceleration in global orientation frame, $\mathbf{g}$ is gravitational acceleration. $\mathbf{R}_{\mathcal{I}\mathcal{G}}$ represents rotational transformation between IMU frame ($\mathcal{I}$) and global frame ($\mathcal{G}$), while $\mathbf{T}_{\mathrm{a}}$ shape matrix comprises gyroscope axis misalignments and scale errors of bases. The measurement noise $\mathbf{a}_{\mathrm{n}}$ is modelled as a zero mean Gaussian random variable, $\mathbf{a}_{\mathrm{n}} \sim \mathcal{N}(\mathbf{0}, \mathbf{N}_{\mathrm{a}})$, and the bias $\mathbf{a}_{\mathrm{b}}$ changes over the time and is modelled as a random walk process driven by its own noise vector $\mathbf{a}_{w\mathrm{n}} \sim \mathcal{N}(\mathbf{0}, \mathbf{N}_{w\mathrm{a}})$.

Regarding gyroscope, it measures rotational velocity ($\boldsymbol{\omega}_{\mathrm{m},\mathcal{I}} \in \mathbb{R}^3$) in IMU orientation frame, its realistic model can be figured out as below.

$$\boldsymbol{\omega}_{\mathrm{m},\mathcal{I}}(\mathrm{t}) = \mathbf{T}_{\mathrm{g}}\boldsymbol{\omega}_{\mathcal{I}}(\mathrm{t}) + \mathbf{T}_{\mathrm{s}}\mathbf{a}_{\mathcal{I}}(\mathrm{t}) + \boldsymbol{\omega}_{\mathrm{b}}(\mathrm{t}) + \boldsymbol{\omega}_{\mathrm{n}}(\mathrm{t}) \tag{24}$$

where $\boldsymbol{\omega}_{\mathcal{I}}$ is the real rotational velocity in IMU orientation frame, $\mathbf{T}_{\mathrm{g}}$ is the shape matrix, while $\mathbf{T}_{\mathrm{s}}\mathbf{a}_{\mathcal{I}}$ represents influence of acceleration upon rotational velocity.

In practice, due to their insignificant effects scale, misalignment and acceleration influence is considered idealistic ($\mathbf{T}_{\mathrm{a}} = \mathbf{T}_{\mathrm{g}} = \mathbf{I}, \mathbf{T}_{\mathrm{s}} = \mathbf{0}$). Most of the real implementations assume biases and noises during modelling.

5.3 Extended Kalman Filter (EKF)

The Bayesian technique can be solved by EKF, which is applicable for nonlinear systems, while noise is considered as Gaussian. In EKF, the motion or state transition model Eq. (19) is formalized by the following equation.

$$\mathbf{x}_{\mathrm{k}} = \mathbf{f}(\mathbf{x}_{\mathrm{k}-1}, \mathbf{u}_{\mathrm{k}}) + \mathbf{w}_{\mathrm{k}} \tag{25}$$

where $\mathbf{f}$ function models vehicle kinematics in function of actual state $\mathbf{x}_{\mathrm{k}-1}$ and actual control input $\mathbf{u}_{\mathrm{k}}$ and $\mathbf{w}_{\mathrm{k}}$ is an additive zero mean Gaussian noise with covariance $\mathbf{Q}_{\mathrm{k}}$ ($\mathbf{w}_{\mathrm{k}} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_{\mathrm{k}})$).

On the other side, EKF implements the generic observation model Eq. (20) by the following equation.

$$\mathbf{z}_{\mathrm{k}} = \mathbf{h}(\mathbf{x}_{\mathrm{k}}) + \mathbf{v}_{\mathrm{k}} \tag{26}$$

where $\mathbf{h}$ function describes relation between actual state $\mathbf{x}_k$ and state-dependent measurements $\mathbf{z}_k$. The $\mathbf{v}_k$ is again an additive zero mean Gaussian error of observation with covariance $\mathbf{R}$ ($\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$).

The system state vector of filter-based visual odometry solutions can be divided into the part related to the motion estimation ($\mathbf{x}_{base}$) and the auxiliary section related to the observation model related to the certain solution ($\mathbf{x}_{aux}$). The base part $\mathbf{x}_{base}$ comprises parameters necessary to describe kinetic and dynamic state it also contains parameters necessary for modelling gyroscope and accelerometer. The auxiliary part $\mathbf{x}_{aux}$, in visual based systems, is necessary to describe the visual observation model. In real implementations it can contain real 3D positions or projected positions of map landmarks ($\mathbf{m}$) or even consecutive camera positions and orientations.

$$\mathbf{x}_k = [\mathbf{x}_{base,k}, \mathbf{x}_{aux,k}] \tag{27}$$

The related state covariance matrix ($\mathbf{P}$) is also can be divided into parts related to the motion model ($\mathbf{P}_{base}$), to the observation model ($\mathbf{P}_{aux}$), and describes the relation between these parameters ($\mathbf{P}_{base,aux}$).

$$\mathbf{P}_k = \begin{bmatrix} \mathbf{P}_{base,k} & \mathbf{P}_{base,aux,k} \\ \mathbf{P}^T_{base,aux,k} & \mathbf{P}_{aux,k} \end{bmatrix} \tag{28}$$

During time update the state vector estimate and related covariance matrix are updated according to the following equations.

$$\begin{aligned} \hat{\mathbf{x}}_{k|k-1} &= \mathbf{f}(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k) \\ \mathbf{P}_{k|k-1} &= \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}^T_k + \mathbf{Q}_k \end{aligned} \tag{29}$$

where the $\mathbf{F}$ is the Jacobian of $\mathbf{f}$ function and evaluated around at the state estimate $\hat{\mathbf{x}}_k$ and actual control input $\mathbf{u}_k$, $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\hat{\mathbf{x}}_k, \mathbf{u}_k)$.

Based on the visual observations, correction is formulated according to following equations that describe the residual, the Kalman gain, respectively.

$$\begin{aligned} \mathbf{r}_k &= \mathbf{z}_k - \mathbf{h}(\hat{\mathbf{x}}_{k|k-1}) \\ \mathbf{K}_k &= \mathbf{P}_{k|k-1} \mathbf{H}^T_k (\mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}^T_k + \mathbf{R}_k)^{-1} \end{aligned} \tag{30}$$

According to the residual and the Kalman gain estimated state and covariance matrix updates are defined as the followings.

$$\begin{aligned} \hat{\mathbf{x}}_{k|k} &= \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \mathbf{r}_k \\ \mathbf{P}_{k|k} &= (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k|k-1} \end{aligned} \tag{31}$$

Although, due to its flexibility and moderate complexity, EKF is the most widely used Bayesian filter, during its application some of the drawbacks have to be considered. Since EKF linearises the actual non-linear characteristics, the choice of point of linearisation affects its stability, while special care has to be taken to the estimation of noise variances.

Considering 6DOF kinematic properties of the smart-phone, application requires from the filter state to store actual orientation, position, velocity and gyroscope and accelerometer bias parameters, at least. According to this consideration, kinematic part of the filter state is defined by the following vector.

$$\mathbf{x} = [\mathbf{q}_{\mathcal{GI}}, \mathbf{p}_{\mathcal{I},\mathcal{G}}, \mathbf{v}_{\mathcal{I},\mathcal{G}}, \boldsymbol{\omega}_b, \mathbf{a}_b]^T \tag{32}$$

During state propagation using the gyroscope-accelerometer measurement pair, the nominal values of kinetic part of the state should follow the kinetic equations below.

$$\begin{aligned} \dot{\mathbf{q}}_{\mathcal{GI}} &= \frac{1}{2}\mathbf{q}_{\mathcal{GI}} \otimes (\boldsymbol{\omega}_m - \boldsymbol{\omega}_b), \quad \dot{\mathbf{p}}_{\mathcal{I},\mathcal{G}} = \mathbf{v}_{\mathcal{I},\mathcal{G}}, \\ \dot{\mathbf{v}}_{\mathcal{I},\mathcal{G}} &= \mathbf{R}_{\mathcal{GI}}(\mathbf{a}_m - \mathbf{a}_b) + \mathbf{g}, \quad \dot{\boldsymbol{\omega}}_b = \mathbf{0}, \quad \dot{\mathbf{a}}_b = \mathbf{0} \end{aligned} \tag{33}$$

5.4 Particle filter

The Bayesian propagation and measurement equations (see Eq. (22) and Eq. (21)) cannot be solved in closed form for the SLAM problem. For Gaussian-distribution the solution can be approximated with various Kalman-filters but the exact solution for strongly non-linear models can only be found by numerical integration.

Given a $\mathbf{g}(\mathbf{x}) : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ function, the expectation over a posterior distribution:

$$E[\mathbf{g}(\mathbf{x})|\mathbf{z}_{1:k}] = \int \mathbf{g}(\mathbf{x}) P(\mathbf{x}|\mathbf{z}_{1:k}) \, d\mathbf{x} \tag{34}$$

can be approximated by drawing N independent random samples $\mathbf{x}^{(i)}$ form the $p(\mathbf{x}|\mathbf{z}_{1:k})$ distribution:

$$E[\mathbf{g}(\mathbf{x})|\mathbf{z}_{1:k}] \approx \frac{1}{N}\sum_{i=1}^{N} \mathbf{g}(\mathbf{x}^{(i)}) \tag{35}$$

This type of numerical calculation of integrals is called Monte Carlo method [49]. However, in case of the Bayesian models it is not possible to draw samples from $P(\mathbf{x}|\mathbf{z}_{1:k})$, so we need to use importance sampling in order to approximate the distribution.

5.4.1 Importance sampling

To overcome the issue of not having the $P(\mathbf{x}|\mathbf{z}_{1:k})$ distribution, we can construct an importance distribution $\Pi(\mathbf{x}|\mathbf{z}_{1:k})$ from which it is easy to draw samples [33]. It is straightforward that

$$\int \mathbf{g}(\mathbf{x})P(\mathbf{x}|\mathbf{z}_{1:k})\,d\mathbf{x} = \int \left[\mathbf{g}(\mathbf{x})\frac{P(\mathbf{x}|\mathbf{z}_{1:k})}{\Pi(\mathbf{x}|\mathbf{z}_{1:k})}\right]\Pi(\mathbf{x}|\mathbf{z}_{1:k})\,d\mathbf{x} \tag{36}$$

By using this form we can establish a Monte Carlo approximation of the expectation by drawing samples from the importance distribution as $\mathbf{x}^{(i)} \sim \Pi(\mathbf{x}|\mathbf{z}_{1:k})$ and calculating the sum

$$\begin{aligned} E[\mathbf{g}(\mathbf{x})|\mathbf{z}_{1:k}] &\approx \frac{1}{N}\sum_{i=1}^{N}\frac{P(\mathbf{x}^{(i)}|\mathbf{z}_{1:k})}{\Pi(\mathbf{x}^{(i)}|\mathbf{z}_{1:k})}\mathbf{g}(\mathbf{x}^{(i)}) \\ &= \sum_{i=1}^{N}\tilde{w}^{(i)}\mathbf{g}(\mathbf{x}^{(i)}) \end{aligned} \tag{37}$$

where $\tilde{w}^{(i)}$ is defined as

$$\tilde{w}^{(i)} = \frac{1}{N}\frac{P(\mathbf{x}^{(i)}|\mathbf{z}_{1:k})}{\Pi(\mathbf{x}^{(i)}|\mathbf{z}_{1:k})} \tag{38}$$

By using normalized weights and applying Bayes-rule we can replace the posterior distribution $P(\mathbf{x}^{(i)}|\mathbf{z}_{1:k})$ with the prior and the likelihood function. The normalized weights $w^{(i)}$ can be written as

$$\begin{aligned} w^{*(i)} &= \frac{P(\mathbf{z}_{1:k}|\mathbf{x}^{(i)})P(\mathbf{x}^{(i)})}{\Pi(\mathbf{x}^{(i)}|\mathbf{z}_{1:k})} \\ w^{(i)} &= \frac{w^{*(i)}}{\sum_{j=1}^{N} w^{*(j)}} \end{aligned} \tag{39}$$

Finally, the posterior probability density function can be formed by using the Dirac delta function $\delta(\cdot)$:

$$p(\mathbf{x}|\mathbf{z}_{1:k}) \approx \sum_{i=1}^{N} w^{(i)}\delta(\mathbf{x}-\mathbf{x}^{(i)}) \tag{40}$$

5.4.2 Sequential importance sampling

Specifically, for state space models shown in Eq. (19) and Eq. (20), the modified version of the importance sampling algorithm can be used to calculate the expectation and probability density efficiently. The sequential importance sampling algorithm uses a weighted set of particles $\{(w_k^{(i)}, \mathbf{x}_k^{(i)})\}$, which contains samples from an importance distribution and their weights for every k time instant. The distribution can be approximated with the particles as

$$p(\mathbf{x}_k|\mathbf{z}_{1:k}) \approx \sum_{i=1}^{N} w_k^{(i)} \delta(\mathbf{x}_k - \mathbf{x}_k^{(i)}) \tag{41}$$

The weights can be calculated at every time instant with the equation

$$w_k^{(i)} \propto w_{k-1}^{(i)} \frac{P(\mathbf{z}_k|\mathbf{x}_k^{(i)}) P(\mathbf{x}_k^{(i)}|\mathbf{x}_{k-1}^{(i)})}{\Pi(\mathbf{x}_k^{(i)}|\mathbf{x}_{0:k-1}^{(i)}, \mathbf{z}_{1:k})} \tag{42}$$

where $w_k^{(i)}$ shall be normalized to sum to unity [49].

Sequential importance sampling still has the problem of having too many particles with almost zero weights, thanks to the special properties of the distributions used in SLAM techniques. This situation is called degeneracy problem and has a great impact on using them in real applications. To avoid degeneracy problem a re-sampling method called *sequential importance re-sampling* is used [16]. The main idea behind re-sampling is to draw new samples from the discrete distribution defined by the weights and use them as new particles. This way particles with relevant weights get duplicated and particles with small weights get removed. Commonly, the sequential importance re-sampling algorithm is referred to as *particle filter*.

5.5 Solutions

5.5.1 EKF-SLAM

In EKF-SLAM algorithms filter state vector contains current IMU state $\mathbf{x}_{base}$ and the observed feature 3D positions ($\mathbf{p}_{f_i}$). Thus the filter state vector is defined as follows.

$$\mathbf{x}_k = [\mathbf{x}_{base,k}, \mathbf{p}_{f_{1,k}}^T ... \mathbf{p}_{f_{n,k}}^T]^T \tag{43}$$

The 3D feature can be parametrized traditionally using (x, y, z) coordinates, anchored homogeneous parametrization [52] or the inverse-depth parametrization [6]. However, the former one is straightforward the latter two increases the consistency and accuracy.

EKF-SLAM uses "standard" propagation method of states ($\mathbf{x}_k$) and covariance matrix ($\mathbf{P}_k$) based on the IMU inertial measurements as described above, while the *update process* is calculated on the observed image features. Assuming a calibrated perspective camera, observation of feature i on the actual image at time step k is described by the following equation that describes the actual observation.

$$\mathbf{z}_{i,k} = \mathbf{h}(\mathbf{x}_{IMU,k}, \mathbf{p}_{f_{i,k}}) = \frac{1}{z_{f_i,\mathcal{C}_k}} \begin{bmatrix} x_{f_i,\mathcal{C}_k} \\ y_{f_i,\mathcal{C}_k} \end{bmatrix} + \mathbf{n}_{i,k} \tag{44}$$

where $\mathbf{n}_{i,k}$ is the measurement noise, and $\mathbf{p}_{f_i,\mathcal{C}_k} = [x_{f_i,\mathcal{C}_k}, y_{f_i,\mathcal{C}_k}, z_{f_i,\mathcal{C}_k}]$ describes observed position of feature f_i in camera orientation frame $\mathcal{C}_k$, and this position is described by the following equation and the $\mathbf{p}_{\mathcal{I},\mathcal{C}}$ and $\mathbf{R}_{\mathcal{C}\mathcal{I}}$ are the fixed position and rotation transformations between the IMU ($\mathcal{I}$) and the camera ($\mathcal{C}$) frames.

$$\mathbf{p}_{f_i,\mathcal{C}_k} = \mathbf{R}_{\mathcal{C}\mathcal{I}}\mathbf{R}_{\mathcal{I}_k\mathcal{G}}(\mathbf{p}_{f_i,\mathcal{G}} - \mathbf{p}_{\mathcal{I}_k,\mathcal{G}}) + \mathbf{p}_{\mathcal{I},\mathcal{C}} \tag{45}$$

Assuming that the actual position of the IMU frame is $\mathbf{p}_{\mathcal{I}_k,\mathcal{G}}$, EKF-SLAM defines a residual as the difference between the real observation $\mathbf{z}_{i,k}$ of the feature i and the projection of the estimated feature position ($\hat{\mathbf{p}}_{f_i,\mathcal{C}_k}$), and linearise it around the actual state ($\hat{\mathbf{x}}_{IMU,k}$) as:

$$\mathbf{r}_{i,k} = \mathbf{z}_{i,k} - \mathbf{h}(\hat{\mathbf{x}}_{IMU,k}, \hat{\mathbf{p}}_{f_i,\mathcal{C}_k}) \simeq \mathbf{H}_{i,k}(\hat{\mathbf{x}}_k)\tilde{\mathbf{x}}_k + \mathbf{n}_{i,k} \tag{46}$$

The $\mathbf{H}_{i,k}(\hat{\mathbf{x}}_k)$ is the Jacobian matrix of $\mathbf{h}$ with respect to the actual filter state estimate ($\hat{\mathbf{x}}_k$).

When $\mathbf{r}_{i,k}$ and $\mathbf{H}_{i,k}$ are computed, the outlier detection is done using Mahalanobis gating. If it succeeds the test using residual and observation Jacobian, Kalman gain and state innovation are computed according to basic EKF rules (see Eq. (31)). For Mahalanobis gating we compute the following:

$$\gamma_i = \mathbf{r}_i^T(\mathbf{H}_i\mathbf{P}_i\mathbf{H}_i^T + \sigma^2\mathbf{I})^{-1}\mathbf{r}_i \tag{47}$$

Then it is compared to the threshold given by the 96 percent of the χ^2 distribution of dimensions equal to the residual vector.

Observation update step requires that all landmarks and joint-covariance matrix must be updated every time an image is registered by the camera. Considering the complexity of the EKF-SLAM, it is straightforward that the computational complexity is dominated by cubic of actual number of landmarks, thus the complexity is $\mathcal{O}(n^2)$. In practice, a map can consists of thousands of features, thus the EKF-SLAM becomes computationally intractable for large areas.

To provide the first-aid to this problem Sola proposed a method, when state and covariance matrices are updated by the actual observed feature numbers, in this way making a step to cover the real-time requirements. [47]

5.5.2 MSCKF

The fundamental advantage of filter-based algorithms is that they account for the correlations exist between pose of the camera and 3D positions of the observed features. On the other hand, the main limitation is its high computational complexity.

The motivation of Multi-State Constraint Filter (MSCKF) is the introduction of consecutive camera poses into state instead of observable feature landmarks, as it is first introduced by Nister [43], however this method does not incorporate inertial measurements. Sliding window-based solutions also appear in other papers [7].

Assuming that, N of the camera poses are included in the EKF state vector at time step k the MSCK state vector has the following form.

$$\mathbf{x}_k = [\mathbf{x}_{base,k}, \mathbf{q}_{\mathcal{GC}_1}^T, \mathbf{p}_{\mathcal{C}_1,\mathcal{G}} \ldots \mathbf{q}_{\mathcal{GC}_N}^T, \mathbf{p}_{\mathcal{C}_N,\mathcal{G}}]^T \tag{48}$$

Since *time update* is common for EKF-based pose estimators, the difference is maintained during *measurement update* step, when new image is arrived and features are tracked among the last N camera poses. The update process is based on each single feature f_j that has been observed from the set of N_j camera poses $(\mathbf{q}_{\mathcal{GC}_i}^T, \mathbf{p}_{\mathcal{C}_i,\mathcal{G}})$ that has been also available in the state vector.

The estimated feature position $\hat{\mathbf{p}}_{f_j,\mathcal{G}}$ in the global frame is triangulated from these N camera poses using feature observations. During this process, usually, a least-square minimization used with inverse-depth parametrization [6]. Then, the residual is defined as the difference between re-projections of estimated feature $\hat{\mathbf{p}}_{f_j,\mathcal{G}}$ to the N_j cameras and the real feature observation is defined as $\mathbf{r}_j^{(j)}$.

$$\mathbf{r}_i^{(j)} = \mathbf{z}_i^{(j)} - \hat{\mathbf{z}}_i^{(j)} \tag{49}$$

On the other hand, the residual can be approximated by linearising around the estimates of the camera poses and the feature positions, where $\mathbf{H}_{\mathbf{x}_i}$ and $\mathbf{H}_{f_j}^{(j)}$ are the Jacobians of the measurement $\mathbf{z}_i^{(j)}$ with respect to the state and the feature position, respectively. After stacking the residuals for each N_j measurements of the f_j features we get the following equation.

$$\mathbf{r}^{(j)} \simeq \mathbf{H}_{\mathbf{x}}\tilde{\mathbf{x}} + \mathbf{H}_f^{(j)}\tilde{\mathbf{p}}_{f_j,\mathcal{G}} \tag{50}$$

Since actual state estimate $\mathbf{x}$ is used for estimation of $\hat{\mathbf{p}}_{f_j,\mathcal{G}}$, the error of state $\tilde{\mathbf{x}}$ and of feature position $\tilde{\mathbf{p}}_{f_j,\mathcal{G}}$ are correlated. The solution to this problem is projecting $\mathbf{r}^{(j)}$ on the left null-space of the matrix $\mathbf{H}_f^{(j)}$. Define $\mathbf{A}^{(j)}$ as the unitary matrix whose columns form the basis of the left null-space of $\mathbf{H}_f$, we get:

$$\mathbf{r}_o^{(j)} \simeq \mathbf{A}^{(j)\mathsf{T}}\mathbf{H}_x^{(i)}\tilde{\mathbf{x}}^{(j)} + \mathbf{A}^{(j)\mathsf{T}}\mathbf{n}^{(j)} = \mathbf{H}_o^{(j)}\tilde{\mathbf{x}}^{(j)} + \mathbf{n}_o^{(j)} \tag{51}$$

By also stacking residuals into a single vector from observations from each f_j features, we obtain the following single form of equation.

$$\mathbf{r}_o = \mathbf{H}_x\tilde{\mathbf{x}} + \mathbf{n}_o \tag{52}$$

To reduce the computational complexity during update QR decomposition is applied of $\mathbf{H}_x$ [31]. After determining the $\mathbf{T}_H$ upper triangular matrix and its corresponding unitary matrix whose columns form bases for the null-space of $\mathbf{H}_x$, $\mathbf{Q}_1$. The residual is then reformulated as the following.

$$\mathbf{r}_n = \mathbf{Q}_1^\mathsf{T}\mathbf{r}_o = \mathbf{T}_H\tilde{\mathbf{x}} + \mathbf{n}_n \tag{53}$$

Based on the above measures, the residual $\mathbf{r}_n$ and the measurement Jacobian $\mathbf{T}_H$ the basic EKF update is applied (see Eg. (31)).

The correct co-operation between image based relative observations and inertial measurements requires to exactly know the transformation between camera and IMU orientation frames. In most of the solutions this transformation assumed to be known exactly, EKF is appropriate also to estimate these parameters. The improvements in MSCKF 2.0 [31] introduces these parameters $(\mathbf{q}_{IC}, \mathbf{p}_{C,I})$ to the state parameters. Besides, in this paper global orientation errors are considered and improved linearisation and calculation of Jacobians are provided. These methods improve the observability and increase accuracy and stability while estimating relative orientation and positions.

MSCKF model is also augmented with estimation of rolling shutter camera properties [28] and temporal calibration [30], while algorithm is provided for on-line self-calibration [32].

Regarding the computational complexity, it is easy to realize that instead of EKF-SLAM, complexity fundamentally depends on the number of registered camera states not on the number of observed features. However, calculation of $\mathbf{T}_H$ depends on the number of features ($\sim d$) and the columns of the $\mathbf{Q}_1$ (r). Other crucial factor is determined by the computation of covariance matrix update. The cost of MSCKF update is then determined by $\max(\mathcal{O}(r^2 d), \mathcal{O}(m^3))$, where m is the size of the state vector.

One can see that since MSCKF uses sliding window for camera states, tracked features can be observed only for a time limited to window size. To overcome this limitation the authors designed a hybrid MSCKF-EKF SLAM solution, where the optimality found on using MSCKF for short features and long feature track is inserted to the state as it is done in EKF SLAM [29].

5.5.3 FastSLAM

FastSLAM implements PF method, however, high dimensional state-space of the SLAM problem makes it computationally infeasible to apply particle filters on the Bayesian-equations directly. FastSLAM solves this problem by applying a factorization to the posterior distribution as follows. [38]:

$$\mathrm{p}(\mathbf{x}_{1:\mathrm{k}}, \mathbf{m}|\mathbf{z}_{0:\mathrm{k}}, \mathbf{u}_{0:\mathrm{k}}, \mathbf{x}_0) = \mathrm{p}(\mathbf{x}_{1:\mathrm{k}}|\mathbf{z}_{0:\mathrm{k}}, \mathbf{u}_{0:\mathrm{k}}, \mathbf{x}_0) \prod_{\mathrm{k}} \mathrm{p}(\mathbf{m}_\mathrm{k}|\mathbf{x}_{1:\mathrm{k}}, \mathbf{z}_{0:\mathrm{k}}, \mathbf{u}_{0:\mathrm{k}}, \mathbf{x}_0) \quad (54)$$

Estimation thus can be done in two steps: first we estimate the posterior of path trajectories then – based on estimated trajectory – we estimate the locations of the K landmarks independently. The path estimation is done by a modified particle estimator using Monte Carlo method, while estimation of landmarks is achieved by Kalman-filters. Because landmarks are conditioned on path estimation, if M particle is used to estimate the trajectory, then KM two dimensional Kalman-filters are required to estimate the landmarks.

FastSLAM runs time linear in the number of landmarks, however, the implementation of FastSLAM uses a tree representation of particles to run in $\mathcal{O}(\mathsf{M} \log \mathsf{K})$. This way, the re-sampling of particles can happen much faster than using native implementation.

FastSLAM can handle huge amounts of landmarks – as extensive simulation has shown – and is at least as accurate as EKF-SLAM. However, the biggest problem of FastSLAM is the inability to forget the past (i.e. the pose and measurement history) and this way the statistical accuracy is lost [2].

FastSLAM has a more efficient extension called FastSLAM 2.0, which uses another proposal distribution including current landmark observations and this way calculating importance weights differently [39].

6 Implementation aspects

It is essential for visual odometry and SLAM algorithms to run real-time. Recent smartphones are equipped with a considerable amount of resources,

like multiple cores of CPU and GPU. To face to the real-time requirements by utilizing parallel resources, a couple of algorithms decouple real-time and background tasks. The computational burden is still really high for embedded devices. Fortunately, these algorithms give way to a lot of parallelisation to speed up computations.

Feature extraction is also much faster if done parallel, e.g. SiftGPU reported to extract SIFT features at 27 FPS on a nVidia 8800GTX card [58]. The widespread OpenCV[1] has also GPU support for various algorithms using CUDA and OpenCL. Not only feature detection and extraction but bundle adjustment can be parallelised to be ca. 30 times faster than native implementation as the Multi-core Bundle Adjustment project shows [59].

7 Evaluation

Beside the solutions described in this work, a huge amount of implementations are available (see https://openslam.org). For the prudent comparison of the methods, algorithms and real implementations, widely known datasets are used. These datasets provide huge amount of video frames of different trajectories with ground truth, containing mainly grayscale and RGB images but often RGB-D and laser data is also accessible. The most widely used datasets are the KITTI dataset [15], the RGB-D dataset [53] and New College Data Set [51], from those the KITTI odometry dataset consists of 22 stereo sequences (which can also be used as a monocular data) and a comprehensive evaluation of different SLAM methods listing accuracy and speed.

Regarding the KITTI dataset a huge list about performance evaluation of available implementations is published at http://www.cvlibs.net/datasets/kitti/eval_odometry.php.

8 Conclusion

Huge variety of algorithms and solutions are currently available to tackle the strict requirements of the accurate and real time visual indoor positioning that augmented reality-based applications demand. These algorithms build on the results of research work on computer vision from the last decades, which went through big evolution, from SFM to the real-time SLAM approaches. However, to face to real-time requirements filter-based solutions tightly coupling inertial measurements with visual odometry are emerging. Through embedding inertial

[1] OpenCV can be found at http://opencv.org

measurements from IMU for motion estimation to the projective geometry principles, these approaches are promising for future implementations, however they suffer from the capability of long-lasting state parameter estimations.

Acknowledgements

This publication and related research was supported by the PIAC-13-1-2013-0226 (Organic Localization) project. The project is supported by the Hungarian Government, and financed by the Research and Technology Innovation Fund.

References

[1] Y. I. Abdel-Aziz, H. M. Karara, M. Hauck, Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry, *Photogrammetric Engineering and Remote Sensing* **81**, 2 (2015) 103–107. ⇒191

[2] T. Bailey, J. Nieto, E. Nebot, Consistency of the fastslam algorithm, *Proc. of IEEE International Conference on Robotics and Automation (ICRA'06)*, Orlando, FL, USA, 2006, pp. 424–429. ⇒209

[3] S. Baker, I. Matthews, Lucas-kanade 20 years on: A unifying framework, *International Journal of Computer Vision (IJCV)* **56**, 3 (2004) 221–255. ⇒198

[4] S. S. Beauchemin, R. Bajcsy, Modelling and removing radial and tangential distortions in spherical lenses, *Proc. of International Workshop on Theoretical Foundations of Computer Vision (TFCV'00), Multi-Image Analysis, Lecture Notes in Computer Science*, Dagstuhl Castle, Germany, 2000, pp. 1–21. ⇒189

[5] J. Chao, A. Al-Nuaimi, G. Schroth, E. Steinbach, Performance comparison of various feature detector-descriptor combinations for content-based image retrieval with jpeg-encoded query images, *Proc. of International Workshop on Multimedia Signal Processing (MMSP'13)*, Pula, Sardinia, Italy, 2013, pp. 29–34. ⇒190

[6] J. Civera, A. J. Davison, J. M. M. Montiel, Inverse depth parametrization for monocular slam, *IEEE Transactions on Robotics* **24**, 5 (2008) 932–945. ⇒205, 207

[7] L. E. Clement, V. Peretroukhin, J. Lambert, J. Kelly, The battle for filter supremacy: A comparative study of the multi-state constraint kalman filter and the sliding window filter, *Proc. of Conference on Computer and Robot Vision (CRV'15)*, Halifax, Nova Scotia, 2015, pp. 23–30. ⇒207

[8] A. J. Davison, I. D. Reid, N. D. Molton, O. Stasse, Monoslam: Real-time single camera slam, *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* **29**, 6 (2007) 1052–1067. ⇒199

[9] H. Durrant-Whyte, T. Bailey, Simultaneous localization and mapping: part i, *IEEE Robotics & Automation Magazine* **13**, 2 (2006) 99–110. ⇒187, 200

[10] J. Engel, T. Schöps, D. Cremers, Lsd-slam: Large-scale direct monocular slam, *Proc. of 13th European Conference on Computer Vision (ECCV'14)*, volume 2, Springer International Publishing, Zurich, Switzerland, 2014, pp. 834–849. ⇒197

[11] J. Engel, J. Sturm, D. Cremers, Semi-dense visual odometry for a monocular camera, *Proc. of IEEE International Conference on Computer Vision (ICCV'13)*, Sydney, Australia, 2013, pp. 1449–1456. ⇒197

[12] M. A. Fischler, R. C. Bolles, Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography, *Communications of the ACM* **24**, 6 (1981) 381–395. ⇒191

[13] C. Forster, M. Pizzoli, D. Scaramuzza, Svo: Fast semi-direct monocular visual odometry, *Proc. of IEEE International Conference on Robotics and Automation (ICRA'14)*, Hong Kong, China, 2014, pp. 15–22. ⇒198

[14] F. Fraundorfer, D. Scaramuzza, Visual odometry, part ii: Matching, robustness, optimization, and applications, *IEEE Robotics & Automation Magazine* **19**, 2 (2012) 78–90. ⇒187

[15] A. Geiger, P. Lenz, R. Urtasun, Are we ready for autonomous driving? the kitti vision benchmark suite, *Proc. of Conference on Computer Vision and Pattern Recognition (CVPR'12)*, IEEE, Providence, RI, USA, 2012, pp. 3354–3361. ⇒210

[16] N. Gordon, Novel approach to nonlinear/non-gaussian bayesian state estimation, *IEE Proceedings F (Radar and Signal Processing)* **140** (1993) 107–113(6). ⇒205

[17] C. Harris, J. Pike, 3d positional integration from image sequences, *Proc. of the Alvey Vision Conference*, Manchester, UK, 1987, pp. 87–90. ⇒187

[18] C. Harris, M. Stephens, A combined corner and edge detector, *Proc. of the 4th Alvey Vision Conference*, Manchester, UK, 1988, pp. 147–151. ⇒190

[19] R. I. Hartley, In defense of the eight-point algorithm, *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI'97)* **19**, 6 (1997) 580–593. ⇒191

[20] R. I. Hartley, A. Zisserman, *Multiple View Geometry in Computer Vision*, (2nd edition), Cambridge University Press, Cambridge, England, UK, 2004. ⇒187, 190, 191, 192

[21] C. Kerl, J. Sturm, D. Cremers, Robust odometry estimation for rgb-d cameras, *Proc. of IEEE International Conference on Robotics and Automation (ICRA'13)*, Karlsruhe, Germany, 2013, pp. 3748–3754. ⇒195

[22] G. Klein, D. Murray, Parallel tracking and mapping for small ar workspaces, *Proc. of 6th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR 2007)*, Nara, Japan, 2007, pp. 225–234. ⇒187, 193

[23] G. Klein, D. Murray, Improving the agility of keyframe-based SLAM, *Proc. of 10th European Conference on Computer Vision (ECCV'08)*, Marseille, France, 2008, pp. 802–815. ⇒193

[24] L. Kneip, D. Scaramuzza, R. Siegwart, A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation, *Proc. of IEEE Conference on Computer Vision and*

Pattern Recognition (CVPR'11), Colorado Springs, USA, 2011, pp. 2969–2976. ⇒191

[25] K. Konolige, M. Agrawal, J. Solà, Large-scale visual odometry for rough terrain, *Proc. of International Symposium on Robotics Research*, Hiroshima, Japan, 2007, pp. 201–212. ⇒199

[26] V. Lepetit, F.Moreno-Noguer, P.Fua, Epnp: An accurate o(n) solution to the pnp problem, *International Journal Computer Vision* **81**, 2 (2009) 155–166. ⇒191

[27] H. Li, R. Hartley, Five-point motion estimation made easy, *Proc. of 18th International Conference on Pattern Recognition (ICPR'06)*, volume 1, Hong Kong, China, 2006, pp. 630–633. ⇒191

[28] M. Li, B. H. Kim, A. I. Mourikis, Real-time motion tracking on a cellphone using inertial sensing and a rolling-shutter camera, *Proc. of IEEE International Conference on Robotics and Automation (ICRA'13)*, Karlsruhe, Germany, 2013, pp. 4712–4719. ⇒208

[29] M. Li, A. I. Mourikis, Optimization-based estimator design for vision-aided inertial navigation, *Proc. of the Robotics: Science and Systems Conference*, Sydney, NSW, Australia, 2012, pp. 504–509. ⇒209

[30] M. Li, A. I. Mourikis, 3-d motion estimation and online temporal calibration for camera-imu systems, *Proc. of IEEE International Conference on Robotics and Automation (ICRA'13)*, Karlsruhe, Germany, 2013, pp. 5709–5716. ⇒208

[31] M. Li, A. I. Mourikis, High-precision, consistent ekf-based visual-inertial odometry, *International Journal of Robotics Research* **32**, 6 (2013) 690–711. ⇒208

[32] M. Li, H. Yu, X. Zheng, A. I. Mourikis, High-fidelity sensor modeling and self-calibration in vision-aided inertial navigation, *Proc. of IEEE International Conference on Robotics and Automation (ICRA'14)*, Hong Kong, China, 2014, pp. 409–416. ⇒208

[33] J. S. Liu, *Monte Carlo Strategies in Scientific Computing*, Springer Publishing Company, Incorporated, 2008. ⇒204

[34] H. Longuet-Higgins, A computer algorithm for reconstructing a scene from two projections, *Nature* **293**, 10 (1981) 133–135. ⇒187

[35] M. A. Lourakis, A. Argyros, SBA: A software package for generic sparse bundle adjustment, *ACM Transactions on Mathematical Software (TOMS)* **36**, 1 (2009) 1–30. ⇒192

[36] D. G. Lowe, Object recognition from local scale-invariant features, *Proc. of IEEE International Conference on Computer Vision (ICCV'99)*, volume 2, Kerkyra, Greece, 1999, pp. 1150–1157. ⇒190

[37] J. Matas, O. Chum, M. Urban, T. Pajdla, Robust wide-baseline stereo from maximally stable extremal regions, *Image and Vision Computing* **22**, 10 (2004) 761–767. ⇒190

[38] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, Fastslam: A factored solution to the simultaneous localization and mapping problem, *Proc. of National Conference on Artificial Intelligence*, American Association for Artificial Intelligence (AAAI), Edmonton, Alberta, Canada, 2002, pp. 593–598. ⇒209

[39] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, FastSLAM 2.0: An improved

particle filtering algorithm for simultaneous localization and mapping that provably converges, *Proc. of International Joint Conference on Artificial Intelligence (IJCAI-03)*, Acapulco, Mexico, 2003, pp. 1151–1156. ⇒209

[40] R. Mur-Artal, J. M. M. Montiel, J. D. Tardós, ORB-SLAM: a versatile and accurate monocular SLAM system, *IEEE Transactions on Robotics* **31**, 5 (2015) 1147–1163. ⇒193

[41] R. A. Newcombe, S. J. Lovegrove, A. J. Davison, Dtam: Dense tracking and mapping in real-time, *Proc. of International Conference on Computer Vision (ICCV'11)*, IEEE Computer Society, Washington, DC, USA, 2011, pp. 2320–2327. ⇒196

[42] D. Nistér, An efficient solution to the five-point relative pose problem, *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI'04)* **26**, 6 (2004) 756–777. ⇒191

[43] D. Nister, H. Stewenius, Scalable recognition with a vocabulary tree, *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, IEEE Computer Society, Seattle, WA, USA, 2006, pp. 2161–2168. ⇒195, 207

[44] T. Oskiper, Z. Zhu, S. Samarasekera, R. Kumar, Visual odometry system using multiple stereo cameras and inertial measurement unit, *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'07)*, Minneapolis, MN, USA, 2007, pp. 1–8. ⇒199

[45] O. Pizarro, R. Eustice, H. Singh, Relative pose estimation for instrumented, calibrated imaging platforms, *Proc. of Digital Image Computing Techniques and Applications*, Sydney, Australia, 2003, pp. 601–612. ⇒191

[46] S. I. Roumeliotis, A. E. Johnson, J. F. Montgomery, Augmenting inertial navigation with image-based motion estimation, *Proc. of IEEE International Conference on Robotics and Automation (ICRA'02)*, volume 4, Washington, DC, USA, 2002, pp. 4326–4333. ⇒199

[47] C. Roussillon, A. Gonzalez, J. Solà, J.-M. Codol, N. Mansard, S. Lacroix, M. Devy, Rt-slam: A generic and real-time visual slam implementation, *Proc. of International Conference of Computer Vision Systems (ICVS'11), Lecture Notes in Computer Science* **6962** (2011) 31–40. ⇒207

[48] E. Rublee, V. Rabaud, K. Konolige, G. Bradski, Orb: An efficient alternative to sift or surf, *Proc. of International Conference on Computer Vision (ICCV'11)*, Barcelona, Spain, 2011, pp. 2564–2571. ⇒190

[49] S. Särkkä, *Bayesian Filtering and Smoothing*, (1st edition), Cambridge University Press, New York, NY, USA, 2013. ⇒203, 205

[50] D. Scaramuzza, F. Fraundorfer, Visual odometry, part i: The first 30 years and fundamentals, *IEEE Robotics & Automation Magazine* **18**, 4 (2011) 80–92. ⇒187

[51] M. Smith, I. Baldwin, W. Churchill, R. Paul, P. Newman, The new college vision and laser data set, *The International Journal of Robotics Research* **28**, 5 (2009) 595–599. ⇒210

[52] J. Solà, Consistency of the monocular ekf-slam algorithm for three different land-

mark parametrizations, *Proc. of IEEE International Conference on Robotics and Automation (ICRA'10)*, Anchorage, Alaska, 2010, pp. 3513–3518. ⇒205

[53] J. Sturm, N. Engelhard, F. Endres, W. Burgard, D. Cremers, A benchmark for the evaluation of rgb-d slam systems, *Proc. of International Conference on Intelligent Robot Systems (IROS'12)*, Vilamoura, Algarve, Portugal, 2012, pp. 573–580. ⇒210

[54] J. Tardif, M. G. M. Laverne, A. Kelly, M. Laverne, A new approach to vision-aided inertial navigation, *Proc. of International Conference on Intelligent Robots and Systems (IROS'10)*, Taipei, Taiwan, 2010, pp. 4161–4168. ⇒199

[55] B. Triggs, Camera pose and calibration from 4 or 5 known 3d points, *Proc. of IEEE International Conference on Computer Vision (ICCV'99)*, volume 1, Kerkyra, Greece, 1999, pp. 278–284. ⇒191

[56] B. Triggs, P. F. McLauchlan, R. I. Hartley, A. W. Fitzgibbon, Bundle adjustment — a modern synthesis, *Proc. of International Workshop on Vision Algorithms: Theory and Practice*, Springer Berlin Heidelberg, Corfu, Greece, 1999, pp. 298–372. ⇒192

[57] S. Weiss, R. Siegwart, Real-time metric state estimation for modular vision-inertial systems, *Proc. of IEEE International Conference on Robotics and Automation (ICRA'11)*, Shanghai, China, 2011, pp. 4531–4537. ⇒199

[58] C. Wu, SiftGPU: A GPU implementation of scale invariant feature transform (SIFT), `http://cs.unc.edu/~ccwu/siftgpu`, 2007. ⇒210

[59] C. Wu, S. Agarwal, B. Curless, S. M. Seitz, Multicore bundle adjustment, *Proc. of Conference on Computer Vision and Pattern Recognition (CVPR'11)*, Colorado Springs, USA, 2011, pp. 3057–3064. ⇒210

Received: May 17, 2016 • Revised: November 11, 2016