Survey on Monocular Odometry for Conventional Smartphones

Gergely Hollósi, Csaba Lukovszki, István Moldován, Sándor Plósz and Frigyes Harasztos

Abstract—**In the last decade huge amount of research work has been put to realize indoor visual localization with personal smartphones. Considering the available sensors and their capabilities monocular odometry may provide a solution, even regarding strict requirements of augmented reality applications. This paper is aimed to give an overview on the state of the art results regarding monocular visual indoor localization. For this purpose it presents the necessary basics of computer vision and reviews the most promising solutions for different topics.**

Index Terms—**Computer vision, Visual Monocular Odometry, SLAM, Survey**

I. INTRODUCTION

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

Besides, this topic presents a 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 the Structure from Motion (SFM) [1] [2] problem, where the main goal was 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 the problem formulation does not focus on the relative pose estimation of sequential images. Moreover the family of SLAM (Simultaneous Localization and Mapping) algorithms focuses on the environment modeling (map building) and the relative camera pose estimation simultaneously [3]. To overcome the real time and accuracy requirements these solutions induced the PTAM (Parallel Tracking and Mapping) [4]. In the meantime, the problem has been also targeted by another application field, the odometry. The original requirement of the monocular Visual Odometry (VO) [5] [6] was to accurately determine the relative pose of a rover.

Manuscript received May 17, 2016, revised June 24, 2016.

G. Hollósi, Cs. Lukovszki, I. Moldován, S. Plósz are with the Department of Telecommunications and Media Informatics, Budapest University of Technology and Economics, Magyar Tudósok krt. 2., Budapest 1117, Hungary (email: {hollosi, lukovszki, moldovan, plosz}@tmit.bme.hu., url: http://www.tmit.bme.hu) Authors are also work for Pillar Europa Non-Profit ltd., Ady Endre u. 19., 1014 Budapest, Hungary (url: http://pillareuropa.hu)

F. Harasztos is the Director at Flaxcom Holding co. ltd. Mogyordi t 53., 1149 Budapest, Hungary (email: harasztos.frigyes@flaxcom.hu, url: flaxcom.hu)

This work and related research has been funded by the PIAC-13-1-2013-0226 (Organic Localization) project.

In this paper authors are engaged 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 the authors focus on the capabilities of the recent smartphones. Common smartphones are equipped with a thin-lens perspective camera, that can be modeled with an ideal pin-hole model [7], and they are also equipped with IMU (Inertial Measurement Unit) integrating gyroscope and accelerometer, while having reasonable capacity for storage and processing. Regarding the motion of the device the following discussion considers 6dof (degree-of-freedom).

II. THEORETICAL BACKGROUND

Monocular visual odometry tries to determine the pose and location of a device mainly using visual perception aided by a couple of auxiliary sensors (e.g. gyroscope 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.

A. 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 $\pi(\cdot)$ is the projection model. It is essential to mention that in case of monocular systems the $\pi(\cdot)$ 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 huge drawback of loosing the depth information while recording frames.

In practice the projection model is considered to be linear in homogeneous space, i.e. it can be represented by a matrix product (commonly referred to as 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:

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

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

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

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

$$\begin{aligned} \mathbf{x}_k &= \mathbf{K}(f)\left[\mathbf{I}|\mathbf{0}\right]\mathbf{T}_k^{-1}\mathbf{X} = \\ &= \mathbf{K}(f)\left[\mathbf{R}_k^{-1}| - \mathbf{R}_k^{-1}\mathbf{C}_k\right]\mathbf{X} = \\ &= \mathbf{K}(f)\mathbf{P}_k^e\mathbf{X} \end{aligned} \tag{5}$$

where $\mathbf{P}_k^e$ 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 $\mathbf{P}_k^e$ extrinsic camera matrices or the $\mathbf{T}_k$ rigid-body transformation of the cameras mainly based on (but not exclusively) the visual information encoded in frames.

B. 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) [8]. Radial distortion depends on the radial distance from the radial distortion center (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 centers (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.

C. 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.

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. subpixel accuracy), however, the latter requires some kind of filtering or interpolating algorithm, that in some cases can cause information loss.

2) Feature detection: Feature based methods are working 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 [9], from that the most widely used are the Harris detector [10], the Scale-invariant feature transform (SIFT) which is based on Laplacian of Gaussian filters [11], the Maximally Stable Extremal Regions (MSER) [12], the Features from Accelerated Segment Test (FAST), Oriented FAST and Rotated BRIEF (ORB) [13]. Considering the overall requirements SIFT is the most promising, however due to its high complexity strict constraints restrict its application in mobile environments.

III. 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 [7].

A. Theory

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

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 to be calibrated [14]. For 5 or 4 points the [15] 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 [16]. The P3P solution yields to finite number of solutions using only three point pairs as the smallest subset of points pairs [17]. The P3P solution has the advantage of using only three points in a RANSAC (Random Sample Consensus) framework to eliminate outlier point pairs thereby decreasing the required number of iterations.

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 the parameters of a model and classify samples as inliers and outliers. The most noted algorithm is RANSAC which is widely used in the literature [18].

3) Relative pose estimation: The basic terms in relative pose estimation are the *fundamental matrix* and the *essential matrix*, both can be computed from projection pairs. The fundamental matrix is a 3×3 matrix ($\mathbf{F}$) satisfying $\mathbf{x}'^T\mathbf{F}\mathbf{x} = 0$, where projections ($\mathbf{x}$ and $\mathbf{x}'$) are of the same world point in two different images. The essential matrix ($\mathbf{E}$) uses the normalized image coordinates so it can be computed from the intrinsic camera matrix ($\mathbf{K}$) and the fundamental matrix as $\mathbf{E} = \mathbf{K}'^T\mathbf{F}\mathbf{K}$. The essential matrix is applicable to recover the pose of the cameras by decomposition [7]. A lot of methods are known to determine the relative pose of the cameras: the 8 point algorithm [19], the 7 point algorithm [7], 6 point algorithm [20] and 5 point algorithms [21] [22]. It is essential to mark that these algorithms differ in handling degenerate configurations (i.e. coplanar objects or cylinder containing the projection centers) and are unable to recover the scale of the set-up.

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 is the minimization of *reprojection 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 which contains the projection (π_n), distortions (K_n) and rigid body transformation (T_n) of the camera (i.e. intrinsic and extrinsic behavior). 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 the pixel measurements are $\overline{\mathbf{u}}_{n,m}$ then the optimization of reprojection error equals the expression

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

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

As it is obvious from Eq. 8 that the reprojection error is not linear 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* [23]. Bundle adjustment is widely used in SLAM, SfM and odometry problems to refine a coarse solution or optimize the map 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* [24] [7].

B. Implementations

All of the solutions and implementations use the algorithms mentioned above but combine them in quite different ways.

1) PTAM: SLAM methods has 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 [4]. 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 relocaliser [25].

PTAM detects FAST features on a scale pyramid to provide scale invariance and uses these feature points to recover the geometry. The 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 keyframes 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.

The PTAM solution is capable to track the camera pose accurately and 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).

2) ORB-SLAM: ORB-SLAM realizes a rather complex visual odometry solution, however, it is based basically on feature detection and point geometry [26]. As its name suggests it uses ORB features to gather image information and provides odometry and 3d reconstruction simultaneously. Besides, ORB-SLAM provides re-localization 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 keyframe chosen by the tracking thread. The *loop closing* thread is watching for map points to reoccur using bag of words model, and when it founds one, the loop closing corrects the loop by similarity transformation (see Fig. 1).

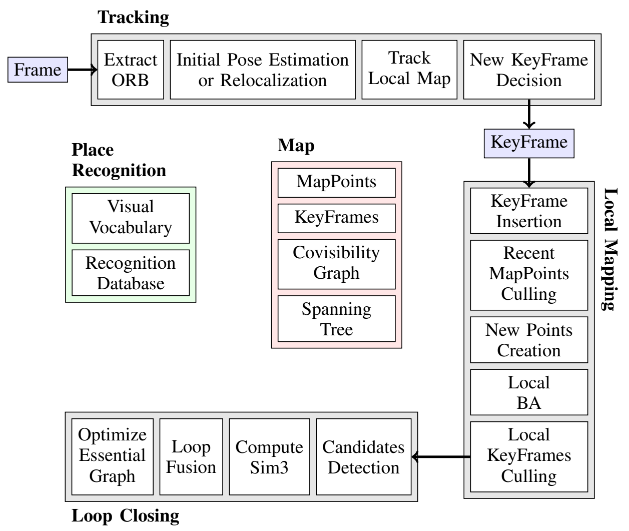

Fig. 1. The architecture of ORB-SLAM 2 system. ORB SLAM 2 runs on three threads parallel to each other. The *Tracking* thread does the real-time pose estimation, the *Local Mapping* thread creates new map points and optimize the local map and the *Loop Closing* thread tries to find loops in the odometry and fixes it.

ORB-SLAM applies ORB feature detection as it provides rotation and scale invariance and 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. The 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:

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

where M is the model (H for homography and F for fundamental matrix), d^2_{cr} and d^2_{rc} 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 reproject the map onto the frame, recovering more feature matches. The last step is the keyframe 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 keyframe 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 reprojection error. To maintain compact reconstruction ORB-SLAM removes redundant keyframes.
4) Loop closing happens parallel to tracking and mapping and uses bag of words representation and co-visibility information to detect loop candidates [27]. 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.

IV. 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.

A. 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 [28]:

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

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 discretized 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.

B. DTAM

Dense Tracking and Mapping (DTAM) uses the photo-consistency theory in a special way to provide dense maps and real-time visual odometry [29]. The main idea behind dense mapping is to sum the photometric error along a ray from the camera center and find the d distance which minimize 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.

The 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 that real-time performance requires huge computing capacity and massive GPU utilization.

C. LSD-SLAM

Large-Scale Direct Monocular SLAM (LSD-SLAM) uses direct methods combined with a probabilistic approach to track camera movements and build dense maps real-time [30]. The LSD-SLAM has a scale-aware image alignment algorithm which directly estimates the similarity transformation between two keyframes 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 keyframe then it decides whether the actual keyframe should be replaced by the new frame. In case of replacement it initializes a new depth map otherwise it propagates the depth map of the current keyframe. At every keyframe replacement LSD-SLAM runs a 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 keyframe and the new frame I_j. In the cost function $r_p(\cdot)$ is the photometric error, σ_{r_p} is the variance of the photometric error and $\|\cdot\|_\delta$ expresses 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 keyframe 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 keyframe selected by the LSD-SLAM process. The depth map is initialized with the depth map of the previous keyframe or with a random depth map if no keyframe exists. For each new frame the depth map is propagated as in [31], namely if the inverse depth for a pixel was d_0 then for the new frame it is approximated as

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

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 keyframes by incorporating the depth residual into the error function shown above. This method penalizes deviations in inverse depth between keyframes and helps to estimate the scaled transformation between them.

D. SVO

The Fast Semi-Direct Monocular Odometry (SVO) is a great example of a hybrid solution for visual odometry using direct and feature based algorithms as well [32]. The 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 on parallel threads, i.e. calculating the movement trajectory at each frame real-time and select keyframes which can be used for mapping on the mapping thread. As the mapping thread uses features, bundle adjustment can be used to minimize reprojection error and construct accurate maps.

The tracking thread projects the 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 [33]) then by local bundle adjustment to apply the epipolar constraints.

The unique solution of the SVO is the fact that no depth map is computed but for each feature point on a keyframe a depth-filter is assigned which estimates the feature depth in a probabilistic way. First, the mapping thread decides the new frame to be a keyframe or not. Feature extraction is executed on new keyframes and to each feature is assigned a freshly initialized depth-filter. On interframes (i.e. not keyframes) the depth-filters for the features 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 then a dense one. The depth filters are a 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.

V. Filter-based solutions

In the real-time applications the relative pose estimation should be seamless, which can not be guaranteed just by image processing. To overcome this problem motion models are introduced to estimate the camera state between pose estimations. One on the most reliable solution is demonstrated as the MonoSLAM [34] for smooth camera motion in a desk-scale local environment.

In indoor application the most reasonable choice for motion estimation is to combine measurements of IMU, gyroscope and accelerometer with the measurements from projective camera images of the environment.

The filter based family of visual odometry algorithms fuses inertial IMU measurements with visual feature observations. In these methods, the current IMU poses and positions of visual landmarks are jointly estimated. These approaches share the same basic principles with camera-only localization based on bundle-adjustment. These solutions in the most cases integrate inertial data from IMU and pose estimations from camera measurements. These combined techniques are characterized as *loosely coupled* and *tightly coupled* systems. In loosely coupled systems [35] [36] [37] 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 [38] [39]. However loosely coupled systems limit computational complexity, in the following we focus on tightly coupled techniques due to its ability to reach higher consistency between camera poses and map of landmarks.

A. 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

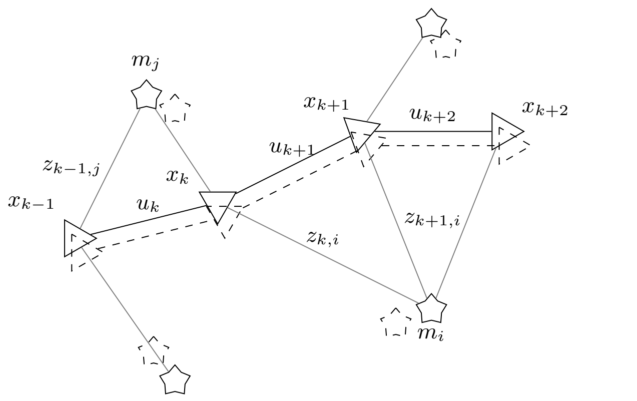

Fig. 2. The probabilistic SLAM problem. The triangles show the robot poses while stars represent landmarks. We depicted the true values with solid lines and the estimated values with dashed lines. The observations always made between the true location and the true landmark position.

even harder to solve if we consider the noise of the sensor measurements. 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 system state parameters, including the camera location and orientation at a discrete time ($\mathbf{x}_k$) based on observations ($\mathbf{z}_k = \{\mathbf{z}_{ik}\}$) from a given location on the environment landmarks, 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 $\mathbf{u}_k$ system state (see Fig. 2). The problem of relative pose estimation is given then in the probabilistic form as follows. [3]

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

The calculation of position probability distribution is done iteratively starting from $P(\mathbf{x}_{k-1}, \mathbf{m}|\mathbf{z}_{1:k-1}, \mathbf{u}_{1:k-1}, \mathbf{x}_0)$ 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.

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

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

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

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

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

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

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

B. The IMU model

In indoor applications practically gyroscope and accelerometer measurements can be used to determine actual relative pose, and in filter algorithms for filter state propagation. All these measurements are stressed with local measurement noise, distortion and biases. The accelerometer measures actual acceleration ($\mathbf{a}_{m,\mathcal{I}} \in \mathbb{R}^3$) in the IMU orientation frame ($\mathcal{I}$), and its model can be formulated as follows.

$$\mathbf{a}_{m,\mathcal{I}}(t) = \mathbf{T}_a\mathbf{R}_{\mathcal{I}\mathcal{G}}(t)(\mathbf{a}_\mathcal{G}(t) - \mathbf{g}) + \mathbf{a}_b(t) + \mathbf{a}_n(t) \quad (23)$$

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

Regarding the gyroscope, it measures rotational velocity ($\boldsymbol{\omega}_{m,\mathcal{I}} \in \mathbb{R}^3$) in the IMU orientation frame, its realistic model looks like the following:

$$\boldsymbol{\omega}_{m,\mathcal{I}}(t) = \mathbf{T}_g\boldsymbol{\omega}_\mathcal{I}(t) + \mathbf{T}_s\mathbf{a}_\mathcal{I}(t) + \boldsymbol{\omega}_b(t) + \boldsymbol{\omega}_n(t) \quad (24)$$

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

In practice, due to their insignificant effects scale and misalignment and acceleration influence is considered idealistic ($\mathbf{T}_a = \mathbf{T}_g = \mathbf{I}, \mathbf{T}_s = \mathbf{0}$).

C. Extended Kalman Filter (EKF)

The Bayesian technique can be solved by EKF, where the motion or state transition model (Eq. 19) is formalized by the following relation.

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

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

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

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k, \mathbf{m}) + \mathbf{v}_k \quad (26)$$

where $\mathbf{h}$ function describes the relation between the actual state $\mathbf{x}_k$ and the map landmarks $\mathbf{m}$ with the projected point of landmark $\mathbf{z}_k$. 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 ($\mathbf{x}_{IMU}$) and the auxiliary section related to the observation model related to the certain solution ($\mathbf{x}_{aux}$).

$$\mathbf{x} = [\mathbf{x}_{IMU}, \mathbf{x}_{aux}] \quad (27)$$

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

$$\mathbf{P}_k = \begin{bmatrix} \mathbf{P}_{IMU} & \mathbf{P}_{IMU,aux} \\ \mathbf{P}^T_{IMU,aux} & \mathbf{P}_{aux} \end{bmatrix} \quad (28)$$

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

$$\begin{aligned} \hat{x} &\leftarrow \mathbf{f}(\hat{x}, \mathbf{u}) \\ \mathbf{P} &\leftarrow \mathbf{F}\mathbf{P}\mathbf{F}^T + \mathbf{Q} \end{aligned} \quad (29)$$

where the $\mathbf{F}$ is the Jacobian of f function and evaluated at the estimate $\hat{x}_k$, thus $\mathbf{F} = \frac{\partial \mathbf{f}(\hat{x}, \mathbf{u})}{\partial \mathbf{x}}|_{\hat{x}_k}$.

Based on the visual observations the correction is formulated in the following equations, that describes the residual, the Kalman gain, respectively.

$$\begin{aligned} \mathbf{r} &= \mathbf{z} - \mathbf{h}(\mathbf{x}) \\ \mathbf{K} &= \mathbf{P}\mathbf{H}^T(\mathbf{H}\mathbf{P}\mathbf{H}^T + \mathbf{R})^{-1} \end{aligned} \quad (30)$$

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

$$\begin{aligned} \hat{\mathbf{x}} &\leftarrow \hat{\mathbf{x}} + \mathbf{K}\mathbf{r} \\ \mathbf{P} &\leftarrow (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P} \end{aligned} \quad (31)$$

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

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

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

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

D. Particle Filter

The bayesian propagation and measurement equations (see Eq. 22 and Eq. 21) cannot be solved in a 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 [40]. 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 approximate it someway. The solution is to find an approximate importance distribution $\Pi(\mathbf{x}|\mathbf{z}_{1:k})$ from which it is easy to draw samples. These kind of techniques are called importance sampling methods. *Particle filter* is the method of using *sequential importance resampling* algorithm. This forms the posterior distribution with a couple of $w_k^{(i)}$ weights.

$$p(\mathbf{x}_k|\mathbf{z}_{1:k}) \approx \sum_{i=1}^{N} w_k^{(i)} \delta(\mathbf{x}_k - \mathbf{x}_k^{(i)}) \tag{36}$$

where $\delta(\cdot)$ is the Dirac-delta.

E. Solutions

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

$$\mathbf{x}_k = [\mathbf{x}_{IMU,k}, \mathbf{p}_{f_{1,k}}^T ... \mathbf{p}_{f_{n,k}}^T]^T \tag{37}$$

The 3D features can be parametrized traditionally using (x, y, z) coordinates, the anchored homogeneous parametrization [41], and the inverse-depth parametrization [42]. Although the former one is straightforward, the latter two increases the consistency and accuracy.

The EKF-SLAM uses the ”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 actual image features. Assuming a calibrated perspective camera, the observation of feature i on the actual image at time step k is expressed by the following equation describing the actual observation.

$$\mathbf{z}_{i,k} = \mathbf{h}(\mathbf{x}_{IMU,k}, \mathbf{p}_{f_{i,k}}) = \frac{1}{z_{f_i,\mathcal{C}_k}} \begin{bmatrix} x_{f_i,\mathcal{C}_k} \\ y_{f_i,\mathcal{C}_k} \end{bmatrix} + \mathbf{n}_{i,k} \tag{38}$$

where $\mathbf{n}_{i,k}$ is the measurement noise, and the $\mathbf{p}_{f_i,\mathcal{C}_k} = [x_{f_i,\mathcal{C}_k}, y_{f_i,\mathcal{C}_k}, z_{f_i,\mathcal{C}_k}]$ describes the observed feature position in the 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 transformation between the IMU ($\mathcal{I}$) and the camera ($\mathcal{C}$) frames.

$$\mathbf{p}_{f_i,\mathcal{C}_k} = \mathbf{R}_{\mathcal{C}\mathcal{I}}\mathbf{R}_{\mathcal{I}_k\mathcal{G}}(\mathbf{p}_{f_i,\mathcal{G}} - \mathbf{p}_{\mathcal{I}_k,\mathcal{G}}) + \mathbf{p}_{\mathcal{I},\mathcal{C}} \tag{39}$$

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 linearizes it around the actual state ($\hat{\mathbf{x}}_{IMU,k}$) as:

$$\mathbf{r}_{i,k} = \mathbf{z}_{i,k} - \mathbf{h}(\hat{\mathbf{x}}_{IMU,k}, \hat{\mathbf{p}}_{f_i,\mathcal{C}_k}) \simeq \mathbf{H}_{i,k}(\hat{\mathbf{x}}_k)\tilde{\mathbf{x}}_k + \mathbf{n}_{i,k} \tag{40}$$

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

$$\gamma_i = \mathbf{r}_i^T(\mathbf{H}_i\mathbf{P}_i\mathbf{H}_i^T + \sigma^2\mathbf{I})^{-1}\mathbf{r}_i \tag{41}$$

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

The 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 to the actual number of the landmarks, thus the complexity is $\mathcal{O}(n^2)$. In practice the actual map can consists of thousands of features, thus the EKF-SLAM becomes computationally intractable for large areas.

To provide first-aid to this problem Sola proposed a method, when the state and covariance matrices are updated by only the actual observed features. [43]

2) MSCKF: The fundamental advantage of filter-based algorithms is they account for the correlations that exist between the pose of the camera and the 3D position of the observed features. Besides, the main limitation is its high computational complexity, even when only hundreds of features are considered during calculations.

The motivation of Multi-State Constraint Filter (MSCKF) is the introduction of consecutive camera poses into the state instead of feature positions. This is first done by Nister [44], however this method does not incorporate inertial measurements. Sliding window-based solutions appear also in other solutions. [45]

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.

$$\mathbf{x}_k = [\mathbf{x}_{imu,k}, \mathbf{q}_{\mathcal{G}\mathcal{C}_1}^T, \mathbf{p}_{\mathcal{C}_1,\mathcal{G}} ... \mathbf{q}_{\mathcal{G}\mathcal{C}_N}^T, \mathbf{p}_{\mathcal{C}_N,\mathcal{G}}]^T \tag{42}$$

Since the *time update* is common for EKF-based pose estimation, the difference is maintained during the *measurement update* step. When new image arrives features are tracked among the last N camera poses. The update process considers each single feature f_j that has been observed from the set of N_j camera poses ($\mathbf{q}_{\mathcal{G}\mathcal{C}_i}^T, \mathbf{p}_{\mathcal{C}_i,\mathcal{G}}$).

The estimated feature position $\hat{\mathbf{p}}_{f_j,\mathcal{G}}$ in the global frame is triangulated from camera poses using feature observations. Usually a least-square minimization is used with inverse-depth parametrization. [42] The residual $\mathbf{r}_j^{(j)}$ is then defined as the difference between re-projections of estimated feature $\hat{\mathbf{p}}_{f_j,\mathcal{G}}$ and the real feature observations.

$$\mathbf{r}_i^{(j)} = \mathbf{z}_i^{(j)} - \hat{\mathbf{z}}_i^{(j)} \tag{43}$$

On the other hand the residual can be approximated by linearising about 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

$$\mathbf{r}^{(j)} \simeq \mathbf{H}_{\mathbf{x}}\tilde{\mathbf{x}} + \mathbf{H}_f^{(j)}\tilde{\mathbf{p}}_{f_j,\mathcal{G}} \quad (44)$$

Since the 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 of 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 the columns of which form the basis of the left null-space of $\mathbf{H}_f$, so we get:

$$\mathbf{r}_o^{(j)} \simeq \mathbf{A}^{(j)T}\mathbf{H}_{\mathbf{x}}^{(i)}\tilde{\mathbf{x}}^{(j)} + \mathbf{A}^{(j)T}\mathbf{n}^{(j)} = \mathbf{H}_o^{(j)}\tilde{\mathbf{x}}^{(j)} + \mathbf{n}_o^{(j)} \quad (45)$$

Also by stacking residuals into a single vector from observations from each f_j features, we obtain:

$$\mathbf{r}_o = \mathbf{H}_{\mathbf{x}}\tilde{\mathbf{x}} + \mathbf{n}_o \quad (46)$$

To reduce the computational complexity during update QR decomposition is applied on $\mathbf{H}_{\mathbf{x}}$ [46]. After determining the $\mathbf{T}_H$ upper triangular matrix and its corresponding unitary matrix whose columns form bases for the rand and null-space of $\mathbf{H}_{\mathbf{x}}$, $\mathbf{Q}_1$, the residual is then reformulated as the following:

$$\mathbf{r}_n = \mathbf{Q}_1^T\mathbf{r}_o = \mathbf{T}_H\tilde{\mathbf{x}} + \mathbf{n}_n \quad (47)$$

Based on the above measures, the residual $\mathbf{r}_n$ and the measurement Jacobian $\mathbf{T}_H$ the basic EKF update is used (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, while EKF is appropriate also for the estimation of these parameters. The MSCKF 2.0 [46] introduces these parameters $(\mathbf{q}_{IC}, \mathbf{p}_{C,I})$ into the state. Besides, global orientation errors are considered and an improved linearizion and calculation of Jacobians are provided to improve the observability and in increase accuracy and stability.

The MSCKF model later is extended with estimation of rolling shutter camera properties [47] and temporal calibration [48], while algorithm is provided for on-line self-calibration [49], as well.

Regarding the computational complexity it is easy to realize that instead of the EKF-SLAM the complexity basically depends more on the registered camera states than the observed number of features. However the calculation of $\mathbf{T}_H$ depends on the number of features ($\sim d$) and the columns of the $\mathbf{Q}_1$ (r). The other crucial factor is determined by the computation of covariance matrix update. The cost of the MSCKF update is then calculated 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 the window size. To overcome this limitation the authors designed a hybrid MSCKF-EKF SLAM solution, where MSCKF is applied only for short, while long features are inserted into the state vector. [50]

3) FastSLAM: The FastSLAM implements PF method, however the high dimensional state-space of the SLAM problem makes it computationally infeasible to apply particle filters directly on the Bayesian-equations. FastSLAM solves this problem by applying a factorization to the posterior distribution as follows [51]:

$$p(\mathbf{x}_{1:k}, \mathbf{m}|\mathbf{z}_{0:k}, \mathbf{u}_{0:k}, \mathbf{x}_0) = p(\mathbf{x}_{1:k}|\mathbf{z}_{0:k}, \mathbf{u}_{0:k}, \mathbf{x}_0) \cdot \prod_k p(\mathbf{m}_k|\mathbf{x}_{1:k}, \mathbf{z}_{0:k}, \mathbf{u}_{0:k}, \mathbf{x}_0) \quad (48)$$

The estimation thus can be done in two steps: first we estimate the posterior of the path trajectories then – based on the trajectory estimated – we estimate the locations of the K landmarks independently. The path estimation is done by a modified particle estimator using Monte Carlo method, while the estimation of the landmarks is achieved by Kalman-filters. Because landmarks are conditioned on the path estimation if M particle is used to estimate the trajectory then KM two dimensional Kalman-filter is 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 $\mathcal{O}(M \log K)$. This way the resampling of particles can happen much faster than implemented naively.

The 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 [52].

FastSLAM has a more efficient extension called FastSLAM 2.0 which uses another proposal distribution including the current landmark observations and this way calculating the importance weights differently [53].

VI. 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, some 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 parallelization opportunities to speed up computations.

The 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 [54]. The widespread OpenCV[1] library has also GPU support for various algorithms using CUDA and OpenCL. Not only feature detection and extraction but bundle adjustment can be parallelized to be ca. 30 times faster than native implementations such as the Multicore Bundle Adjustment project shows [55].

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

VII. EVALUATION

Beside the solutions described in this work, a huge amount of implementations are available. 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 accessible. The most widely used datasets are the KITTI dataset [56], the RGB-D dataset [57] and New College Data Set [58], 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 the performance evaluation of available implementations is published at http://www.cvlibs.net/datasets/kitti/eval_odometry.php.

VIII. CONCLUSION

A 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 a significant evolution, from SfM to the real-time SLAM approaches. However to face the real-time requirements, filter-based solutions tightly coupling inertial measurements with visual odometry are emerging. Through embedding inertial measurements from IMU for motion estimation to the projective geometry principles, these approaches are promising for the future implementations, however they suffer from the long-lasting state parameter estimations.

REFERENCES

[1] H. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," vol. 293, no. 10, pp. 133 – 135.

[2] C. H. J. Pike, "3d positional integration from image sequences," in *Proc. Alvey Vision Conf.*, p. 87 90.

[3] T. B. H. Durrant-Whyte, "Simultaneous localization and mapping: part i," vol. 13, no. 2, pp. 99 – 110.

[4] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, Nov 2007, pp. 225–234.

[5] F. F. Davide Scaramuzza, "Visual odometry, tutorial," vol. 18, no. 4, pp. 80 – 92.

[6] D. S. Friedrich Fraundorfer, "Visual odometry: Part ii: Matching, robustness, optimization, and applications," vol. 19, no. 2, pp. 78 – 90.

[7] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, ISBN: 0521540518, 2004.

[8] S. S. Beauchemin and R. Bajcsy, "Modelling and removing radial and tangential distortions in spherical lenses," in *Multi-Image Analysis: 10th International Workshop on Theoretical Foundations of Computer Vision*, March 2000, pp. 1–21.

[9] J. Chao, A. Al-Nuaimi, G. Schroth, and E. Steinbach, "Performance comparison of various feature detector-descriptor combinations for content-based image retrieval with jpeg-encoded query images," in *Multimedia Signal Processing (MMSP), 2013 IEEE 15th International Workshop on*, Sept 2013, pp. 029–034.

[10] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings of the 4th Alvey Vision Conference*, 1988, pp. 147–151.

[11] D. Lowe, "Object recognition from local scale-invariant features," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 2, 1999, pp. 1150–1157 vol.2.

[12] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust wide-baseline stereo from maximally stable extremal regions," *Image and Vision Computing*, vol. 22, no. 10, pp. 761 – 767, 2004.

[13] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International Conference on Computer Vision*, Nov 2011, pp. 2564–2571.

[14] Y. Abdel-Aziz, H. Karara, and M. Hauck, "Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry*," *Photogrammetric Engineering and Remote Sensing*, vol. 81, no. 2, pp. 103 – 107, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0099111215303086

[15] B. Triggs, "Camera pose and calibration from 4 or 5 known 3d points," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 1, 1999, pp. 278–284 vol.1.

[16] V. Lepetit, F.Moreno-Noguer, and P.Fua, "Epnp: An accurate o(n) solution to the pnp problem," *International Journal Computer Vision*, vol. 81, no. 2, 2009.

[17] L. Kneip, D. Scaramuzza, and R. Siegwart, "A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, June 2011, pp. 2969–2976.

[18] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981. [Online]. Available: http://doi.acm.org/10.1145/358669.358692

[19] R. I. Hartley, "In defense of the eight-point algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 6, pp. 580–593, Jun 1997.

[20] O. Pizarro, R. Eustice, and H. Singh, "Relative pose estimation for instrumented, calibrated imaging platforms," in *Proceedings of Digital Image Computing Techniques and Applications*, Sydney, Australia, December 2003, pp. 601–612.

[21] H. Li and R. Hartley, "Five-point motion estimation made easy," in *18th International Conference on Pattern Recognition (ICPR'06)*, vol. 1, 2006, pp. 630–633.

[22] D. Nistér, "An efficient solution to the five-point relative pose problem," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 6, pp. 756–777, Jun. 2004. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2004.17

[23] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, *Vision Algorithms: Theory and Practice: International Workshop on Vision Algorithms Corfu, Greece, September 21–22, 1999 Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, ch. Bundle Adjustment — A Modern Synthesis, pp. 298–372. [Online]. Available: http://dx.doi.org/10.1007/3-540-44480-7_21

[24] M. A. Lourakis and A. Argyros, "SBA: A Software Package for Generic Sparse Bundle Adjustment," *ACM Trans. Math. Software*, vol. 36, no. 1, pp. 1–30, 2009.

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

[26] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: a versatile and accurate monocular SLAM system," *CoRR*, vol. abs/1502.00956, 2015. [Online]. Available: http://arxiv.org/abs/1502.00956

[27] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, ser. CVPR '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 2161–2168. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2006.264

[28] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for rgb-d cameras," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, May 2013, pp. 3748–3754.

[29] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "Dtam: Dense tracking and mapping in real-time," in *Proceedings of the 2011 International Conference on Computer Vision*, ser. ICCV '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 2320–2327. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2011.6126513

[30] J. Engel, T. Schöps, and D. Cremers, *Computer Vision – ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part II*. Cham: Springer International Publishing, 2014, ch. LSD-SLAM: Large-Scale Direct Monocular SLAM, pp. 834–849. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-10605-2_54

[31] J. Engel, J. Sturm, and D. Cremers, "Semi-dense visual odometry for a monocular camera," in *2013 IEEE International Conference on Computer Vision*, Dec 2013, pp. 1449–1456.

[32] C. Forster, M. Pizzoli, and D. Scaramuzza, “Svo: Fast semi-direct monocular visual odometry,” in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 15–22.
[33] S. Baker and I. Matthews, “Lucas-kanade 20 years on: A unifying framework,” *International Journal of Computer Vision*, vol. 56, no. 3, pp. 221–255, 2004. [Online]. Available: http://dx.doi.org/10.1023/B:VISI.0000011205.11775.fd
[34] N. D. M. Andrew J. Davison, Ian D. Reid and O. Stasse, “Monoslam: Real-time single camera slam,” vol. 29, no. 6, pp. 1052 – 1067.
[35] J. F. M. S. I. Roumeliotis, A. E. Johnson, “Augmenting inertial navigation with image-based motion estimation,” in *IEEE International Conference on Robotics and Automation, 2002. Proceedings. ICRA ’02.*, vol. 4, pp. 4326 – 4333.
[36] A. K. J. Tardif, M. G. M. Laverne and M. Laverne, “A new approach to vision-aided inertial navigation,” in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4161 – 4168.
[37] R. S. Stephan Weiss, “Real-time metric state estimation for modular vision-inertial systems,” in *IEEE International Conference on Robotics and Automation (ICRA), 2011*, pp. 4531 – 4537.
[38] S. S. Taragay Oskiper, Zhiwei Zhu and R. Kumar, “Visual odometry system using multiple stereo cameras and inertial measurement unit,” in *IEEE Conference on Computer Vision and Pattern Recognition 2007*, pp. 1 – 8.
[39] M. A. Kurt Konolige and J. Solà, “Large-scale visual odometry for rough terrain,” in *Robotics Research - The 13th International Symposium, ISRR 2007*, pp. 201 – 212.
[40] S. Srkk, *Bayesian Filtering and Smoothing*. New York, NY, USA: Cambridge University Press, 2013.
[41] J. Solà, “Consistency of the monocular ekf-slam algorithm for three different landmark parametrizations,” in *IEEE International Conference on Robotics and Automation (ICRA), 2010*, pp. 3513 – 3518.
[42] J. M. M. M. Javier Civera, Andrew J. Davison, “Inverse depth parametrization for monocular slam,” vol. 24, no. 5, pp. 932 – 945.
[43] J. S. e. a. Cyril Roussillon, Aurlien Gonzalez, “Rt-slam: A generic and real-time visual slam implementation,” in *8th International Conference, ICVS 2011, Lecture Notes in Computer Science*, vol. 6962, pp. 31 – 40.
[44] “Visual odometry for ground vehicle applications,” *Journal of Field Robotics*, vol. 23, no. 1, pp. 3–20, 2006.
[45] J. L. Lee E Clement, Valentin Peretroukhin and J. Kelly, “The battle for filter supremacy: A comparative study of the multi-state constraint kalman filter and the sliding window filter,” in *12th Conference on Computer and Robot Vision (CRV), 2015*, pp. 23 – 30.
[46] A. M. M. Li, “High-precision, consistent ekf-based visual-inertial odometry,” vol. 32, no. 6, pp. 690 – 711.
[47] B. H. K. Mingyang Li and A. I. Mourikis, “Real-time motion tracking on a cellphone using inertial sensing and a rolling-shutter camera,” in *IEEE International Conference on Robotics and Automation (ICRA), 2013*, pp. 4712 – 4719.
[48] A. I. M. Mingyang Li, “3-d motion estimation and online temporal calibration for camera-imu systems,” in *IEEE International Conference on Robotics and Automation (ICRA), 2013*, pp. 5709 – 5716.
[49] X. Z. Mingyang Li, Hongsheng Yu and A. I. Mourikis, “High-fidelity sensor modeling and self-calibration in vision-aided inertial navigation,” in *IEEE International Conference on Robotics and Automation (ICRA, 2014*, pp. 409 – 416.
[50] A. M. M. Li, “Optimization-based estimator design for vision-aided inertial navigation,” in *Proceedings of the Robotics: Science and Systems Conference*.
[51] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, “Fastslam: A factored solution to the simultaneous localization and mapping problem,” in *Eighteenth National Conference on Artificial Intelligence*. Menlo Park, CA, USA: American Association for Artificial Intelligence, 2002, pp. 593–598. [Online]. Available: http://dl.acm.org/citation.cfm?id=777092.777184
[52] T. Bailey, J. Nieto, and E. Nebot, “Consistency of the fastslam algorithm,” in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, May 2006, pp. 424–429.
[53] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, “FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges,” in *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*. Acapulco, Mexico: IJCAI, 2003.
[54] C. Wu, “SiftGPU: A GPU implementation of scale invariant feature transform (SIFT),” http://cs.unc.edu/~ccwu/siftgpu, 2007.
[55] C. Wu, S. Agarwal, B. Curless, and S. M. Seitz, “Multicore bundle adjustment,” in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, June 2011, pp. 3057–3064.
[56] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the kitti vision benchmark suite,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.
[57] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, “A benchmark for the evaluation of rgb-d slam systems,” in *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.
[58] M. Smith, I. Baldwin, W. Churchill, R. Paul, and P. Newman, “The new college vision and laser data set,” *The International Journal of Robotics Research*, vol. 28, no. 5, pp. 595–599, May 2009. [Online]. Available: http://www.robots.ox.ac.uk/NewCollegeData/

Gergely Hollósi is a researcher at Dept. of Telecommunications and Media Informatics (TMIT) of Budapest University of Technology and Economics (BME). Gergely received his M.Sc. in the Budapest University of Technology and Economics (BME) in 2009. He is actively researching and developing indoor localization systems based on various technologies including radio-, IMU- and visual-based solutions.

Csaba Lukovszki is a researcher at Dept. of Telecommunications and Media Informatics (TMIT) of Budapest University of Technology and Economics (BME). He has received M.Sc. in electrical engineering in 1998 and finished Ph.D. course in 2002. He is a lecturer of courses and leader of innovative application-driven research and development projects in the field of indoor positioning with strong cooperation with industrial partners. He is the author of 8 journal papers and 23 conference papers.

István Moldován is a Research Fellow at the Budapest University of Technology and Economics, at the Department of Telecommunications and Media Informatics. In 1996, he received an M.Sc. degree in Automation and Industrial Informatics from Technical University of Tirgu-Mures, Romania. His research interests include network management, embedded systems, simulation and performance evaluation of computer networks. He is lecturing in communication networks.

Sándor Plósz is a researcher at Dept. of Telecommunications and Media Informatics (TMIT) of Budapest University of Technology and Economics (BME). He received his M.Sc. in Informatics at the Budapest University of Technology and Economics (BME) in 2009 and finished the Ph.D. course in 2012. He has been a researcher at the University, author of several journal of conference papers in the topics of dependable embedded systems, security of industrial systems and vision-based localization.