2016 IEEE International Symposium on Mixed and Augmented Reality

Robust Keyframe-based Monocular SLAM for Augmented Reality

Haomin Liu[1] Guofeng Zhang[1,2*] Hujun Bao[1,2*]
[1]State Key Lab of CAD&CG, Zhejiang University
[2]The Collaborative Innovation Center for Industrial Cyber-Physical System, Zhejiang University

ABSTRACT

Keyframe-based SLAM has achieved great success in terms of accuracy, efficiency and scalability. However, due to parallax requirement and delay of map expansion, traditional keyframe-based methods easily encounter the robustness problem in the challenging cases especially for fast motion with strong rotation. For AR applications in practice, these challenging cases are easily encountered, since a home user may not carefully move the camera to avoid potential problems. With the above motivation, in this paper, we present RKSLAM, a robust keyframe-based monocular SLAM system that can reliably handle fast motion and strong rotation, ensuring good AR experiences. First, we propose a novel multi-homography based feature tracking method which is robust and efficient for fast motion and strong rotation. Based on it, we propose a real-time local map expansion scheme to triangulate the observed 3D points immediately without delay. A sliding-window based camera pose optimization framework is proposed, which imposes the motion prior constraints between consecutive frames through simulated or real IMU data. Qualitative and quantitative comparisons with the state-of-the-art methods, and an AR application on mobile devices demonstrate the effectiveness of the proposed approach.

Index Terms: I.4.8 [Image Processing and Computer Vision]: Scene Analysis—Tracking; H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—Artificial, augmented, and virtual realities

1 INTRODUCTION

Along with the popularity of mobile and wearable devices, augmented reality (AR) has received unprecedented attention in recent years. Seamless augmented reality requires accurate camera pose estimation and 3D reconstruction of the scene in real-time, which is the problem that SLAM (Simultaneous Localization and Mapping) aims to solve.

Due to the complexity of real cases, it is not easy for traditional SLAM methods to work well in practice. For example, a novice home user takes a mobile device to capture the scene and watch the AR effect. He may not have the knowledge about how to carefully move the mobile device to make the AR system work well, and just would like to freely rotate and move the mobile device to see what he want to see. Fast motion and strong rotation easily happen which are very challenging even for the state-of-the-art SLAM systems [17, 33, 6, 27].

Good AR experience requires that SLAM system can handle kinds of complex camera motion, allowing easy use for a novice home user. The frequency of camera lost is as small as possible even encountering unexpected conditions such as very fast camera motion with severe motion blur. Even if encountering tracking failure, the camera pose can be quickly relocalized to avoid long time waiting.

*Corresponding authors: {zhangguofeng,bao}@cad.zju.edu.cn

In this paper, we present RKSLAM, a robust keyframe-based monocular SLAM system that is able to simultaneously handle fast motion and strong rotation in a robust way. To the best of our knowledge, this problem has not been thoroughly investigated before but actually very important for achieving good AR experiences. The main contributions of our paper are as follows:

1. We propose an effective multi-homography based feature tracking method for both well-constrained and ill-conditioned 3D points, which is not only robust to fast motion and strong rotation, but also very efficient to allow real-time computation even on a mobile device.

2. We propose a novel sliding-window based local map expansion and optimization framework, which not only can triangulate and optimize the newly observed 3D points immediately without delay, but also can effectively optimize the camera poses through simulating IMU measurements with reasonable assumptions in AR applications. This method can significantly improve the robustness in the case of fast camera motion with severe blur. Real IMU measurements if available can be incorporated into the optimization framework to further improve the robustness and accuracy.

2 RELATED WORK

Visual SLAM can be categorized into filtering-based methods and keyframe-based methods. MonoSLAM [2] is a representative work of filtering-based SLAM. The camera motion parameters and 3D positions of landmarks are jointly represented as a probabilistic state. EKF is used to predict and update the joint state for each frame. The main drawback of MonoSLAM is the $O(N^3)$ computational complexity, where N is the number of landmarks, limiting its scalability to only a small space with hundreds of landmarks. In addition, the linearization error is easily accumulated by EKF. PTAM [17] addresses these limitations with a novel keyframe-based parallel tracking and mapping framework. Real-time performance is obtained by separating tracking from mapping, and high accuracy is guaranteed by performing bundle adjustment (BA) [35] over keyframes. As concluded in [31], keyframe-based BA outperforms filtering especially when N is large.

Many recent state-of-the-art SLAM systems adapt keyframe-based framework, such as RDSLAM [33] and ORB-SLAM [27]. RDSLAM [33] detects the appearance and structure change of the scene, and proposes a variant of RANSAC to support localization in dynamic scenes. ORB-SLAM [27] uses ORB features [30] for tracking, mapping, re-localization and loop detection. Pose graph optimization is also adopted to close loops. Different to most visual SLAM systems, LSD-SLAM [4] replaces the sparse point features with a semi-dense depth map which can be recovered in real-time on a CPU. Localization is performed by direct tracking based on the semi-dense depth map, improving the robustness in featureless environments. SVO [6] also uses direct tracking, but only on sparse point features, eliminating the need of costly feature extraction. With this scheme, very high frame rate can be achieved.

The major weakness of traditional keyframe-based methods is their robustness to fast motion and strong rotation. Firstly, in order

978-1-5090-3641-7/16 $31.00 © 2016 IEEE
DOI 10.1109/ISMAR.2016.24

to well constrain the 3D points, sufficient parallax among neighboring keyframes is typically required, which, however, is not satisfied in the case of strong rotation. Secondly, feature matching between large-baseline keyframes is computationally time-demanding thus can only be performed in the background with at least several frames delay for map expansion, which makes the tracking sensitive to fast camera motion.

Some of the above mentioned state-of-the-arts can handle strong rotation to a certain extent. In order to improve the tracking robustness for strong rotation, ORB-SLAM proposed to relax the large parallax requirement for keyframe insertion. The redundant keyframes and unstable 3D points will be deleted later for maintaining a sparse and accurate global map. Both LSD-SLAM and SVO proposed to use filtering technique for robust depth estimation. Filtering-based methods are generally robust to slow rotation, in which case the parallax is growing gradually from small to large. If the camera is purely rotating, any depth value would result in the same 2D projections. With parallax slowly growing, depth can be gradually updated and finally converged to a good estimate. However, it will become much more challenging if the camera moves fast. Firstly, the delay of map expansion is still not well addressed by these methods. Feature matching and depth filtering on background thread cannot catch up with fast camera movement, which eventually degrades or even fails tracking. Secondly, fast motion and strong rotation make robust feature tracking very difficult. The search range and patch distortion becomes unpredictable, making template warping based feature tracking used in PTAM and SVO unreliable. Invariant feature descriptors used by ORB-SLAM and RDSLAM are also sensitive to large perspective distortion and motion blur. ASIFT [25] proposed to address this problem by simulating different viewpoints. However, the expensive computation is not suitable for real-time application.

There are also some methods proposed to explicitly handle pure rotation problem. Gauglitz *et al.* [8] proposed to switch between 6 DoF tracking and panorama tracking between the current frame and latest keyframe, resulting in multiple separated 3D/panorama submaps. Pirchheim *et al.* [28] and Herrera *et al.* [10] shared the similar idea with [8], but were able to provide a global 3D map. In [28], the depth of each 2D feature is set to infinite for 6 DoF tracking. Herrera *et al.* [10] proposed to minimize the distance to epipolar line instead of re-projection error for 2D features. Both methods triangulate the 2D features into 3D once sufficient parallax is detected. However, pure rotation rarely happens in practice. Camera translation is inevitable even if the user tries the best to rotate the camera around its optical center. The translation will be considered as an additional rotation by panorama tracking, which causes drift. Secondly, the lack of 3D constraints easily causes scale inconsistency. Thirdly, fast camera motion cannot be handled by these methods.

Nowadays, many SLAM systems incorporate IMU data to improve robustness, called visual-inertial SLAM. Most visual-inertial SLAM methods are based on filtering. To break the scalability limitation of filtering-based methods, MSCKF [26] converts the binary constraints between 3D points and camera into the constraints among multiple cameras. Only cameras are included in the state vector instead of 3D points. The complexity is linear in the number of features. Li and Mourikis [21] analyzed the observability of MSCKF, and proposed to use first-estimate Jacobian [12] to fix the linearization point and improve the consistency of MSCKF. In [11], by explicitly enforcing the unobservable directions, the spurious information gain is successfully prevented and the inconsistency is reduced. There are also some visual-inertial SLAM methods based on non-linear optimization, in which the problem is formulated as a factor graph [19] and solved as a MAP estimation. Real-time performance is obtained by only maintaining a local map with old states marginalized [3, 20], or performing incremental smoothing

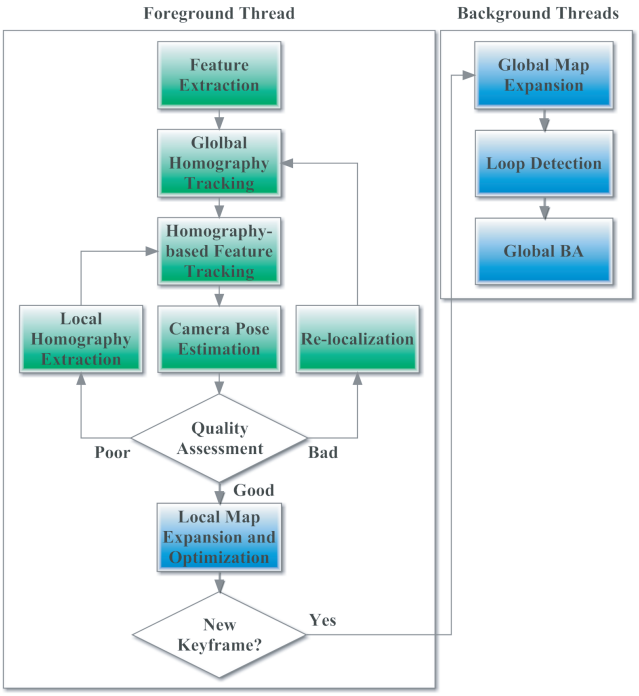


Figure 1: Framework of RKSLAM. Tracking and mapping tasks are green and blue blocks respectively. Note that unlike most keyframe-based systems in which all mapping tasks run in the background, we bring local map expansion and optimization to foreground, for handling fast motion and strong rotation.

with most old states unaffected [15, 13, 7]. These methods are mainly for robot navigation, in which the quality of IMU measurements is quite good.

By assuming a smooth motion, Lovegrove *et al.* [22] proposed to employ cumulative cubic B-splines to parameterize the camera trajectory. With this representation, they can estimate the pose as well as the corresponding first and second derivates at any time instance. IMU measurements can be easily incorporated without discrete-time approximation. With the continuous-time representation, the rolling shutter effects also can be faithfully compensated. Kerl *et al.* [16] also used the cumulative cubic B-splines for direct tracking and dense mapping with rolling shutter RGB-D cameras. However, fast motion and strong rotation violate the smooth motion assumption. In addition, during fast motion, motion blur usually causes the dominant deformation over rolling shutter. Meilland *et al.* [24] proposed a unified solution for estimating motion blur and rolling shutter deformations simultaneously. But this method is for an RGB-D camera, or for a monocular camera only when the 3D model is available.

3 Framework Overview

Similar to previous keyframe-based SLAM systems [17, 33, 27], our system also employs parallel tracking and mapping framework, as illustrated in Figure 1. Different to previous methods, we optimize *global map* in the background thread but optimize *local map* in the foreground thread for fast map expansion. *Global map* contains the selected keyframes and well-constrained 3D points which are observed in at least two keyframes with sufficient parallax. A set of 3D planes are also extracted during global mapping and included into the global map. *Local map* contains a sliding window of most recent frames and those 3D points which are not well-constrained but have at least one observation in the sliding window.

Before describing our method, we first introduce the convention for our mathematical formulation, which is adopted from [9]. We always use an italic symbol (e.g. x) to denote a scalar and a bold capital letter (e.g. $\mathbf{H}$) to denote a matrix. A bold letter such as $\mathbf{b}$ denotes a column vector, and its transpose $\mathbf{b}^\top$ denotes the corresponding row vector. In addition, the group/set of points, vectors or matrices are generally denoted by an italic capital letter, such as V. For a vector $\mathbf{x}$, we use superscript h to denote the corresponding vector in homogenous coordinate, i.e. $\mathbf{x}^h$.

In our system, each 3D point i contains the 3D position $\mathbf{X}_i$ and the local image patch $\mathscr{X}_i$. Each frame i stores its image I_i, camera pose $\mathbf{C}_i$, a set of 2D feature locations $\{\mathbf{x}_{ij}\}$, and a set of indexes to their corresponding 3D points V_i. The set of selected keyframes is denoted as $\{F_k|k=1\cdots N_F\}$. The camera pose $\mathbf{C}_i$ is parameterized as a quaternion $\mathbf{q}_i$ and the camera position $\mathbf{p}_i$. For a 3D point $\mathbf{X}_j$, the 2D projection in i-th image is computed as

$$\mathbf{x}_{ij} = \pi(\mathbf{K}(\mathbf{R}_i(\mathbf{X}_j - \mathbf{p}_i))) \tag{1}$$

where $\mathbf{K}$ is intrinsic matrix assumed to be known and constant, and $\mathbf{R}_i$ is the rotation matrix of $\mathbf{q}_i$. $\pi(\cdot)$ is the projection function $\pi([x,y,z]^\top) = [x/z, y/z]^\top$. Each plane i stores its parameters $\mathbf{P}_i$ and a set of indexes to the 3D points belonging to it, denoted as $V_{\mathbf{P}_i}$. We define the plane parameters $\mathbf{P}_i = [\mathbf{n}_i^\top, d_i]^\top$, where $\mathbf{n}_i$ represents its normal direction and d_i represents the signed distance from origin to the plane. For a 3D point $\mathbf{X}$ lying on the plane $\mathbf{P}_i$, we have $\mathbf{n}_i^\top \mathbf{X} + d_i = 0$.

The key component of RKSLAM is a novel feature tracking method based on three types of homographies, i.e. global homography, local homography and specific plane homography. Global homography describes the planar transformation between two images which can be estimated by global image alignment. We denote the global homography from keyframe F_k to current frame I_i as $\mathbf{H}^G_{k\to i}$. The set of global homographies for all keyframes is denoted as $H^G_i = \{\mathbf{H}^G_{k\to i}|k=1\cdots N_F\}$. Local homography describes the local alignment between two local parts of the images. For the frame pair (F_k, I_i), we can estimate a set of local homographies using the obtained correspondences with multiple RANSAC [5] procedures, denoted as $H^L_{k\to i} = \{\mathbf{H}^{L_j}_{k\to i}|j=1\cdots N^L_{k\to i}\}$.

The specific plane homography is similar to local homography, but defined for a specific 3D plane. For a 3D plane $\mathbf{P}_j$ visible in keyframe F_k, its corresponding homography from F_k to I_i can be derived as

$$\mathbf{H}^{\mathbf{P}_j}_{k\to i} = \mathbf{K}\left(\mathbf{R}_i\mathbf{R}_k^\top + \frac{\mathbf{R}_i(\mathbf{p}_i - \mathbf{p}_k)\mathbf{n}_j^\top \mathbf{R}_k^\top}{d_j + \mathbf{n}_j^\top \mathbf{R}_k \mathbf{p}_k}\right)\mathbf{K}^{-1}. \tag{2}$$

For each keyframe F_k, we derive a set of specific plane homographies, denoted as $H^P_{k\to i} = \{\mathbf{H}^{\mathbf{P}_j}_{k\to i}|j=1\cdots N^P_{k\to i}\}$. The 3D planes can be automatically extracted from the recovered 3D points (The algorithm will be introduced in Section 5.2).

As illustrated in Figure 1, the foreground thread processes the video stream in real-time. For each current frame I_i, we first extract FAST corners [29] and track the set of global homographies H^G_i to I_i. Then we use both the global homographies H^G_i and specific plane homographies $H^P_{k\to i}$ to track 3D points, obtaining a set of 3D-2D correspondences for camera pose estimation. Then we assess the tracking quality by the method of [17], which returns good, poor or bad. If it returns good tracking quality, we directly use the tracking result to expand and optimize the local map, then determine whether a new keyframe should be selected. For fast motion with large translation, the quality may be poor or even bad. In the case of poor quality where there are still some feature matches, most of which are well-constrained points, we estimate a set of local homographies $H^L_{k\to i}$ and try to re-match the lost tracked features. In the case of bad quality, we invoke the re-localization procedure as in [17]. Once re-localization succeeds, we use the re-localized keyframe for global homography tracking, then go to feature tracking again. The newly selected keyframe wakes up the background threads. The global map is expanded by adding the new keyframe and newly triangulated 3D points with local bundle adjustment. After that, we can expand existing 3D planes and extract new 3D planes with these points. Then we detect loops by matching features between the new keyframe and existing keyframes. Finally, BA is performed for the global map. It should be noted that the keyframe selection, re-localization and loop detection in our framework are stand-alone, which can be replaced with more advanced techniques. In our implementation, we directly use the corresponding algorithms in [17, 18] for keyframe selection and re-localization. For loop closure, we use the same algorithm as re-localization to find feature matches between a previous keyframe and the new keyframe, and then close the loop by global BA.

4 Tracking

We assume piecewise planarity of the scene, and extract multiple homographies to track the 3D points to the current frame. It is difficult for traditional methods to predict the search range and correct perspective distortion in the cases of fast motion and strong rotation, especially for those ill-conditioned points whose 3D positions are unreliable. We tackle this difficulty by exploiting the piecewise planarity of natural scenes. With multi-homography representation, search range can be correctly and tightly determined, and perspective distortion can be removed or alleviated.

4.1 Global and Local Homography Estimation

As introduced in Section 3, we extract three kinds of homographies. While global homography roughly aligns the whole image, local and specific plane homographies can precisely align the local regions. Specific plane homography $\mathbf{H}^{\mathbf{P}_j}_{k\to i}$ can be analytically derived according to (2). Here, we describe the algorithms for extracting global and local homographies, respectively.

We estimate the global homography between key frame F_k and current frame I_i by direct image alignment, using the small blurry image (SBI) as used in [18]:

$$\mathbf{H}^G_{k\to i} = \arg\min_{\mathbf{H}} \sum_{\mathbf{x}\in\Omega} ||\tilde{F}_k(\mathbf{x}) - \tilde{I}_i(\pi(\tilde{\mathbf{H}}\mathbf{x}^h))||_{\delta_\mathbf{I}} \tag{3}$$

where Ω is the domain of SBI, and superscript h denotes the vector in homogenous coordinate. $\tilde{F}_k$ and $\tilde{I}_i$ are SBI of F_k and I_i respectively. The tilde above the homography in $\tilde{\mathbf{H}}$ converts the homography $\mathbf{H}$ from the original image space to that of SBI. $||\cdot||_\delta$ is the Huber norm as

$$||e||_\delta = \begin{cases} \frac{||e||_2^2}{2\delta} & \text{if } ||e||_2 \le \delta \\ ||e||_2 - \frac{\delta}{2} & \text{otherwise.} \end{cases} \tag{4}$$

We maintain the global homography $\mathbf{H}^G_{k\to i}$ for each keyframe F_k. It is time-consuming to solve (3) for all keyframes and also unnecessary since most keyframes do not overlap with the current frame. In addition, since the baseline between F_k and I_i is generally large, directly solving (3) would bias the solution towards a major plane. Here, we introduce a novel method to address the above two problems.

It is reasonable to assume the baseline between consecutive frames is small so a global homography can approximately represent the transformation between them. By solving (3) between last frame I_{i-1} and current frame I_i, we can obtain $\mathbf{H}^G_{(i-1)\to i}$. So we have $\mathbf{H}^G_{k\to i} = \mathbf{H}^G_{(i-1)\to i}\mathbf{H}^G_{k\to(i-1)}$ by propagation. Since we already have a set of feature matches (denoted as $M_{k,i-1} = \{(\mathbf{x}_k, \mathbf{x}_{i-1})\}$) between

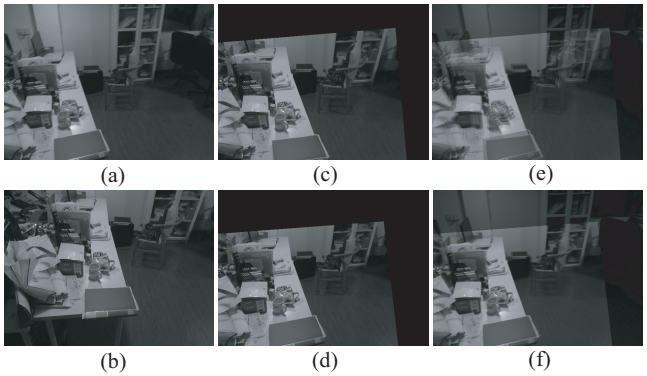


Figure 2: Comparison of global homography estimation. (a) The current image. (b) One selected keyframe to be matched. (c) Warping (b) to (a) by solving (3). (d) Warping (b) to (a) by solving (5). (e) The composition of (a) and (c). (f) The composition of (a) and (d).

F_k and I_{i-1}, we propose to optimize $\mathbf{H}^G_{k\to(i-1)}$ instead of optimizing $\mathbf{H}^G_{k\to i}$. We use $M_{k,i-1}$ to determine the overlapping keyframe set, denoted as K_{i-1}. We select the top 5 ranked keyframes with $|M_{k,i-1}| > 20$. For each keyframe F_k in K_{i-1}, we use $M_{k,i-1}$ to prevent error accumulation and bias, by modifying (3) as

$$\mathbf{H}^G_{k\to(i-1)} = \arg\min_{\mathbf{H}}\Big(\sum_{\mathbf{x}\in\Omega} ||\tilde{F}_k(\mathbf{x}) - \tilde{I}_{i-1}(\pi(\tilde{\mathbf{H}}\mathbf{x}^h))||_{\delta_I} + \sum_{(\mathbf{x}_k,\mathbf{x}_{i-1})\in M_{k,i-1}} ||\pi(\mathbf{H}\mathbf{x}^h_k) - \mathbf{x}_{i-1}||_{\delta_\mathbf{x}}\Big). \tag{5}$$

In our experiments, $\delta_I = 0.1$ and $\delta_\mathbf{x} = 10$. Compared to (3), (5) is able to interpolate multiple planar motions under large baseline. We use Gauss Newton algorithm to solve (3) and (5). Figure 2 gives a comparison. Figure 2(a) shows the current image, and (b) shows the selected keyframe. As can be seen, if we directly solve (3), the estimated global homography is poor as can be evidenced in (c) and (e). The alignment error is obvious. In contrast, the estimated global homography by solving (5) is much better as shown in (d) and (f).

If the tracking quality by global homographies and specific 3D plane homographies is poor, our system will further estimate local homographies with the estimated matches. We use a simple multi-homography extraction scheme [37, 14, 36] to estimate a set of local homographies. Specifically, we perform RANSAC [5] over $M_{k,i}$ to estimate the best local homography with maximum inlier matches, denoted as $\mathbf{H}^{L_1}_{k\to i}$. The inlier matches satisfying $\mathbf{H}^{L_1}_{k\to i}$ are excluded and the procedure is repeated for the remaining matches to estimate $\mathbf{H}^{L_2}_{k\to i}$. The above procedure is repeated until the number of remaining matches is less a threshold.

4.2 Robust Feature Tracking with Multi-Homography Representation

The estimated multiple homographies can be used to perform guided matching, i.e. feature prediction and patch warping which are challenging for traditional methods in the cases of fast motion and strong rotation. Firstly, since the positions for those ill-conditioned 3D points are invalid or inaccurate, the camera pose prediction is unreliable by a traditional motion model. Secondly, if there are strong perspective distortions, simple template matching or comparison with feature descriptors will become unreliable.

Compared to previous methods, we determine the search range for well-constrained and ill-conditioned 3D points in different ways. For well-constrained points whose 3D positions are reliable, we only need to predict the camera pose of current frame $\mathbf{C}_i$ to

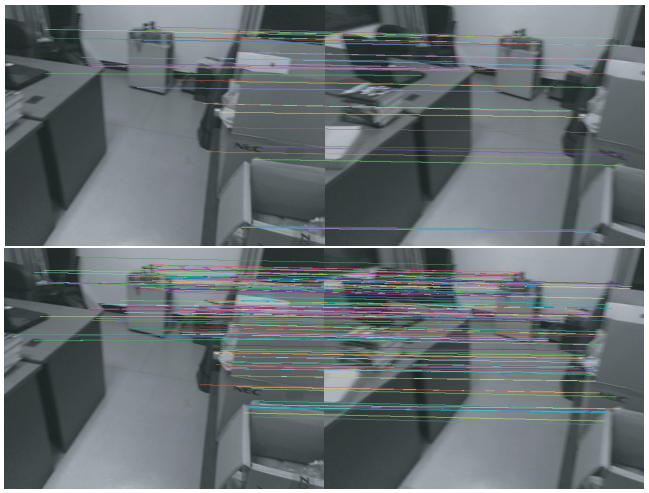


Figure 3: Local homography estimation. Top: 37 matches are obtained by global homography and specific plane homographies. Bottom: additional 153 matches are obtained by the estimated local homographies.

project the 3D points and determine the search location. We use the global homographies H^G_i to predict $\mathbf{C}_i$. As introduced in Section 4.1, the global homographies are derived from last frame I_{i-1}, and the overlapping keyframe set of last frame is K_{i-1}. Without loss of generality, we denote an arbitrary 2D feature in $F_k \in K_{i-1}$ as $\mathbf{x}_k$ and its corresponding 3D point as $\mathbf{X}_k$. By $\mathbf{x}_i = \pi(\mathbf{H}^G_{k\to i}\mathbf{x}^h_k)$, we can get a 3D-2D correspondence $(\mathbf{X}_k, \mathbf{x}_i)$. The set of all such 3D-2D correspondences is denoted as $V_{\mathbf{C}_i} = \{(\mathbf{X}_j, \mathbf{x}_j)\}$. We can predict the camera pose by solving

$$\arg\min_{\mathbf{C}_i} \sum_{(\mathbf{X}_j,\mathbf{x}_j)\in V_{\mathbf{C}_i}} ||\pi(\mathbf{K}(\mathbf{R}_i(\mathbf{X}_j - \mathbf{p}_i))) - \mathbf{x}_j||_{\delta_\mathbf{x}}. \tag{6}$$

For ill-conditioned points, we first try $\pi(\mathbf{H}^G_{k\to i}\mathbf{x}^h_k)$ directly as the search location. If the tracking quality is poor, then a set of local homographies $H^L_{k\to i}$ are estimated from the matched correspondences. We try each of the search locations in $\{\mathbf{H}^{L_j}_{k\to i}x_k | \mathbf{H}^{L_j}_{k\to i} \in H^L_{k\to i}\}$ for those unmatched features. For each search location, the search range is determined as a $r\times r$ rectangle region centered at the search location. We set $r = 10$ and $r = 30$ for well-constrained and ill-conditioned points respectively.

Our multi-homography representation not only can predict good starting points but also can correct perspective distrotion. Given a homography $\mathbf{H}_{k\to i}$, we warp the local patch around $\mathbf{x}_k$ in keyframe F_k to current frame I_i as

$$\mathscr{X}(\mathbf{y}) = I_k(\mathbf{H}^{-1}_{k\to i}(\pi(\mathbf{H}_{k\to i}\mathbf{x}^h_k + \mathbf{y}^h))) \tag{7}$$

where $\mathscr{X}(\mathbf{y})$ is the patch intensity at offset $\mathbf{y}$ relative to patch center. $\mathbf{y} \in (-W/2, W/2)\times(-W/2, W/2)$, W is the patch size and set to 8 in our experiments. For the points belonging to a 3D plane $\mathbf{P}_j$, we use the derived planar homography $\mathbf{H}^{\mathbf{P}_j}_{k\to i}$ (see (2)) for patch warping. For other points, similar to the prediction of search range, we first try global homography $\mathbf{H}^G_{k\to i}$, then each of $H^L_{k\to i}$ for those lost tracked features if tracking quality is poor. Considering that strong perspective change between consecutive frames seldom happens, it is unnecessary to re-compute (7) for each input frame. We can compute the change of the four corners for each warped patch, and re-compute (7) only if the change is larger than a threshold.

Given a correct and tight search range and undistorted patch, we can easily track the correspondence in the current frame. Similar to

PTAM, we compute zero-mean SSD scores at extracted FAST corners within the search region, and select the best one if the difference is less than a threshold. Note that our position prediction and patch distortion correction do not assume any prior motion model, which makes our approach robust to fast motion and strong rotation. In addition, it does not require expensive invariant descriptor computation or large-range patch searching, so that our local map expansion can be executed in the foreground thread without delay.

Some previous methods [37, 14, 36] also use multiple homographies to obtain point correspondences, but for different purposes. [37] uses expensive SIFT features for high quality structure-from-motion, which cannot run in real-time on a CPU. Both [14] and [36] aim to obtain dense correspondences on background pixels. In contrast, our method uses FAST corners and global image alignment to quickly obtain initial matches, and then estimate local homographies to increase the number of matches if necessary. In our method, since sufficient feature matches generally can be obtained by the global homography and specific plane homographies, local homography estimation is seldom invoked. Only in the case where very large parallax exists between F_k and I_i, and the overlapping region consists of multiple separated planes, for those ill-conditioned points, extensively trying each local homography becomes necessary. As shown in Figure 3, 37 matches are obtained by global and specific plane homographies, and additional 153 matches are obtained by local homographies.

5 Mapping

The proposed robust and efficient feature tracking allows real-time local map expansion and optimization, which is crucial to support fast motion and strong rotation. The foreground thread maintains a local map containing a sliding window of most recent frames, and the 3D points which are not well-constrained by keyframes but having at least one observation in the sliding window. These 3D points must be instantiated as soon as possible to track the subsequent camera poses, and must be refined continuously to reduce drift. We propose an efficient method for local map expansion and optimization, as described in Section 5.1. The optimization framework can be extended to incorporate real IMU data. The global map contains the keyframes, the well-constrained 3D points, and a set of extracted 3D planes. If new well-constrained 3D points are triangulated, we can use them to expand the existing 3D planes and extract new 3D planes. The details will be described in Section 5.2.

5.1 RealTime Local Map Expansion and Optimization

For a 2D feature $\mathbf{x}_k$ in F_k, once the correspondence $\mathbf{x}_i$ in I_i is found, we instantiate the 3D point $\mathbf{X}$ immediately. First, we compute the ray angle as

$$\alpha(i,k) = \text{acos}(\frac{\mathbf{r}_k^\top \mathbf{r}_i}{||\mathbf{r}_k|| \cdot ||\mathbf{r}_i||}) \tag{8}$$

where $\mathbf{r}_j = \mathbf{R}_j^\top \mathbf{K}^{-1} \mathbf{x}_j^h$. In ideal case where the parallax is sufficient with $\alpha(i,k) \geq \delta_\alpha$, we instantiate $\mathbf{X}$ by traditional triangulation algorithm [9]. We set $\delta_\alpha = 1^\circ$ in our experiments. Unfortunately in most cases, the parallax is insufficient at first observation. But on the other hand, any depth of $\mathbf{x}_k$ results in similar re-projection in I_i, so we could safely assign arbitrary depth to $\mathbf{x}_k$ at this stage. We choose the mean depth value d_k of keyframe F_k to initialize $\mathbf{X}$ as $\mathbf{X} = d_k \mathbf{K}^{-1} \mathbf{x}_k^h$, and optimize $\mathbf{X}$ as

$$\arg\min_{\mathbf{X}} ||\pi(\mathbf{K}\mathbf{R}_k(\mathbf{X}-\mathbf{p}_k)) - \mathbf{x}_k||_2^2 + ||\pi(\mathbf{K}\mathbf{R}_i(\mathbf{X}-\mathbf{p}_i)) - \mathbf{x}_i||_2^2. \tag{9}$$

Depth error could affect camera tracking once there is sufficient motion parallax. It is best to perform BA over the local map while each new frame comes, but not feasible for real-time performance. Instead we alternate the optimization for only 3D points or camera poses while fixing the other. This strategy is very effective in practice and can significantly reduce the computation to allow real-time performance even on a mobile device. At first glance this strategy seems to be suboptimal. But in fact, we only adjust those ill-conditioned 3D points here, keeping the well-constrained 3D points fixed. The well-constrained 3D points are optimized by global BA in the background and generally very accurate. Camera pose estimation benefits from these well-constrained 3D points, and in turn improves the accuracy of those ill-conditioned 3D points.

It is rather simple to optimize 3D points while fixing camera poses. Each 3D point $\mathbf{X}_i$ can be optimized independently by minimizing

$$\arg\min_{\mathbf{X}_i} \sum_{j \in V_{\mathbf{X}_i}} ||\pi(\mathbf{K}(\mathbf{R}_j(\mathbf{X}_i - \mathbf{p}_j))) - \mathbf{x}_{ij}||_{\delta_\mathbf{x}} \tag{10}$$

where $V_{\mathbf{X}_i}$ is the set of frame indexes for keyframes and the frames in sliding window in which $\mathbf{X}_i$ is observed.

Each camera i could also be optimized independently as

$$\arg\min_{\mathbf{C}_i} \sum_{j \in V_i} ||\pi(\mathbf{K}(\mathbf{R}_i(\mathbf{X}_j - \mathbf{p}_i))) - \mathbf{x}_{ij}||_{\delta_\mathbf{x}} \tag{11}$$

where V_i is the set of point indexes for the 3D points visible in frame i. However, for extremely severe blur caused by fast camera motion, any feature tracking method would fail. Those blurred frames are lack of constraints for reliable camera pose estimation. Since fast camera motion generally occurs occasionally and does not last for a long time, we can use neighboring frames to constrain the blurred frames. Specifically, we borrow the idea from visual-inertial SLAM [26, 11, 20]. Assuming that we have IMU measurements, namely the linear acceleration $\hat{\mathbf{a}}$ and rotational velocity $\hat{\boldsymbol{\omega}}$, expressed in local frame. The state of camera motion is augmented as

$$\mathbf{s} = [\mathbf{q}^\top \quad \mathbf{p}^\top \quad \mathbf{v}^\top \quad \mathbf{b}_\mathbf{a}^\top \quad \mathbf{b}_\omega^\top]^\top \tag{12}$$

where $\mathbf{v}$ is the linear velocity expressed in global frame, $\mathbf{b}_\mathbf{a}$ and $\mathbf{b}_\omega$ are time-varying bias of linear acceleration and rotational velocity respectively. The true linear acceleration $\mathbf{a}$ and rotational velocity $\boldsymbol{\omega}$ are

$$\mathbf{a} = \hat{\mathbf{a}} - \mathbf{b}_\mathbf{a} + \mathbf{n}_\mathbf{a} \tag{13}$$

$$\boldsymbol{\omega} = \hat{\boldsymbol{\omega}} - \mathbf{b}_\omega + \mathbf{n}_\omega \tag{14}$$

where $\mathbf{n}_\mathbf{a} \sim \mathcal{N}(\mathbf{0}, \sigma_{\mathbf{n}_\mathbf{a}}^2 \mathbf{I})$ and $\mathbf{n}_\omega \sim \mathcal{N}(\mathbf{0}, \sigma_{\mathbf{n}_\omega}^2 \mathbf{I})$ are Gaussian noise of the inertial measurements. $\mathbf{I}$ is the 3×3 identity matrix. In our experiments, we generally set $\sigma_{\mathbf{n}_\mathbf{a}} = 0.01 \text{ m}/(\text{s}^2\sqrt{\text{HZ}})$ and $\sigma_{\mathbf{n}_\omega} = 0.01 \text{ rad}/(\text{s}\sqrt{\text{HZ}})$ (we set $\sigma_{\mathbf{n}_\omega} = 0.001 \text{ rad}/(\text{s}\sqrt{\text{HZ}})$ if real IMU measurements are provided).The continuous time kinematic model [1] describing the time evolution of the state is

$$\begin{aligned} \dot{\mathbf{q}} &= \frac{1}{2}\boldsymbol{\Omega}(\boldsymbol{\omega})\mathbf{q} \\ \dot{\mathbf{p}} &= \mathbf{v} \\ \dot{\mathbf{v}} &= \mathbf{R}^\top \mathbf{a} \\ \dot{\mathbf{b}}_\mathbf{a} &= \mathbf{w}_\mathbf{a} \\ \dot{\mathbf{b}}_\omega &= \mathbf{w}_\omega \end{aligned} \tag{15}$$

where

$$\boldsymbol{\Omega}(\boldsymbol{\omega}) = \begin{bmatrix} -[\boldsymbol{\omega}]_\times & \boldsymbol{\omega} \\ -\boldsymbol{\omega}^\top & 0 \end{bmatrix} \tag{16}$$

and $[\cdot]_\times$ denotes the skew symmetric matrix. $\mathbf{b}_\mathbf{a}$ and $\mathbf{b}_\omega$ are modeled as random-walk process driven by the white Gaussian noise $\mathbf{w}_\mathbf{a} \sim \mathcal{N}(\mathbf{0}, \sigma_{\mathbf{w}_\mathbf{a}}^2 \mathbf{I})$ and $\mathbf{w}_\omega \sim \mathcal{N}(\mathbf{0}, \sigma_{\mathbf{w}_\omega}^2 \mathbf{I})$ respectively. In our experiments, we generally set $\sigma_{\mathbf{w}_\mathbf{a}} = 0.001 \text{ m}/(\text{s}^3\sqrt{\text{HZ}})$ and $\sigma_{\mathbf{w}_\omega} =$

0.0001 rad$/(\text{s}^2\sqrt{\text{HZ}})$. For discrete time state, we use the zero-th order integrator proposed in [34] for quaternion propagation:

$$\mathbf{q}_{i+1} = \mathbf{q}_{\Delta}(\boldsymbol{\omega}_i, t_{\Delta_i}) \otimes \mathbf{q}_i \tag{17}$$

where

$$\mathbf{q}_{\Delta}(\boldsymbol{\omega}, t_{\Delta}) = \begin{cases} \begin{bmatrix} \frac{\boldsymbol{\omega}}{|\boldsymbol{\omega}|} \sin(\frac{|\boldsymbol{\omega}|}{2} t_{\Delta}) \\ \cos(\frac{|\boldsymbol{\omega}|}{2} t_{\Delta}) \end{bmatrix} & \text{if } |\boldsymbol{\omega}| > \varepsilon_{\omega} \\ \begin{bmatrix} \frac{\boldsymbol{\omega}}{2} t_{\Delta} \\ 1 \end{bmatrix} & \text{otherwise} \end{cases} \tag{18}$$

t_{Δ_i} is the time interval between consecutive frames i and $i+1$. $\otimes$ is quaternion multiplication operator. ε_{ω} is a small value to avoid division by zero. In our experiments, we set $\varepsilon_{\omega} = 0.00001$. The true $\mathbf{q}_{\Delta}(\boldsymbol{\omega}_i, t_{\Delta_i})$ can be approximated as

$$\mathbf{q}_{\Delta}(\boldsymbol{\omega}_i, t_{\Delta_i}) \approx \begin{bmatrix} \frac{\tilde{\theta}}{2} \\ 1 \end{bmatrix} \otimes \mathbf{q}_{\Delta}(\hat{\boldsymbol{\omega}}_i - \mathbf{b}_{\omega_i}, t_{\Delta_i}) \tag{19}$$

where $\tilde{\theta}$ is a 3×1 error vector. Substituting (14) to (19) with rearranging, we have

$$\begin{aligned} \tilde{\theta} &\approx 2 \left[\mathbf{q}_{\Delta}(\hat{\boldsymbol{\omega}}_i - \mathbf{b}_{\omega_i} + \mathbf{n}_{\omega}, t_{\Delta_i}) \otimes \mathbf{q}_{\Delta}^{-1}(\hat{\boldsymbol{\omega}}_i - \mathbf{b}_{\omega_i}, t_{\Delta_i}) \right]_{1:3} \\ &\approx 2\mathbf{G}_{\omega}\mathbf{n}_{\omega} \end{aligned} \tag{20}$$

$\mathbf{G}_{\omega}$ is the Jacobian matrix with respect to the noise $\mathbf{n}_{\omega}$. Substituting (19) to (17) with rearranging, we have

$$\tilde{\theta} \approx 2 \left[\mathbf{q}_{i+1} \otimes \mathbf{q}_i^{-1} \otimes \mathbf{q}_{\Delta}^{-1}(\hat{\boldsymbol{\omega}}_i - \mathbf{b}_{\omega_i}, t_{\Delta_i}) \right]_{1:3} \tag{21}$$

Combining (20) and (21), we define the cost function and its covariance for the rotation component as

$$\begin{aligned} \mathbf{e}_{\mathbf{q}}(\mathbf{q}_i, \mathbf{q}_{i+1}, \mathbf{b}_{\omega_i}) &= \left[\mathbf{q}_{i+1} \otimes \mathbf{q}_i^{-1} \otimes \mathbf{q}_{\Delta}^{-1}(\hat{\boldsymbol{\omega}}_i - \mathbf{b}_{\omega_i}, t_{\Delta_i}) \right]_{1:3} \\ \Sigma_{\mathbf{q}} &= \sigma_{\mathbf{n}_{\omega}}^2 \mathbf{G}_{\omega} \mathbf{G}_{\omega}^{\top} \end{aligned} \tag{22}$$

Derivation for other components is trivial. The discrete time state propagate is

$$\begin{aligned} \mathbf{p}_{i+1} &= \mathbf{p}_i + \mathbf{v}_i t_{\Delta_i} + \mathbf{R}_i^{\top} \mathbf{p}_{\Delta}(\mathbf{a}_i, t_{\Delta_i}) \\ \mathbf{v}_{i+1} &= \mathbf{v}_i + \mathbf{R}_i^{\top} \mathbf{v}_{\Delta}(\mathbf{a}_i, t_{\Delta_i}) \\ \mathbf{b}_{\mathbf{a}_{i+1}} &= \mathbf{b}_{\mathbf{a}_i} + \mathbf{w}_{\mathbf{a}} t_{\Delta_i} \\ \mathbf{b}_{\omega_{i+1}} &= \mathbf{b}_{\omega_i} + \mathbf{w}_{\omega} t_{\Delta_i} \end{aligned} \tag{23}$$

where

$$\begin{aligned} \mathbf{p}_{\Delta}(\mathbf{a}, t_{\Delta}) &= \frac{1}{2} \mathbf{a} t_{\Delta}^2 \\ \mathbf{v}_{\Delta}(\mathbf{a}, t_{\Delta}) &= \mathbf{a} t_{\Delta} \end{aligned} \tag{24}$$

The cost functions and corresponding covariances are

$$\begin{aligned} \mathbf{e}_{\mathbf{p}}(\mathbf{q}_i, \mathbf{p}_i, \mathbf{p}_{i+1}, \mathbf{v}_i, \mathbf{b}_{\mathbf{a}_i}) &= \mathbf{R}_i(\mathbf{p}_{i+1} - \mathbf{p}_i - \mathbf{v}_i t_{\Delta_i}) - \mathbf{p}_{\Delta}(\hat{\mathbf{a}}_i - \mathbf{b}_{\mathbf{a}_i}, t_{\Delta_i}) \\ \mathbf{e}_{\mathbf{v}}(\mathbf{q}_i, \mathbf{v}_i, \mathbf{v}_{i+1}, \mathbf{b}_{\mathbf{a}_i}) &= \mathbf{R}_i(\mathbf{v}_{i+1} - \mathbf{v}_i) - \mathbf{v}_{\Delta}(\hat{\mathbf{a}}_i - \mathbf{b}_{\mathbf{a}_i}, t_{\Delta_i}) \\ \mathbf{e}_{\mathbf{b}_{\mathbf{a}}}(\mathbf{b}_{\mathbf{a}_i}, \mathbf{b}_{\mathbf{a}_{i+1}}) &= \mathbf{b}_{\mathbf{a}_{i+1}} - \mathbf{b}_{\mathbf{a}_i} \\ \mathbf{e}_{\mathbf{b}_{\omega}}(\mathbf{b}_{\omega_i}, \mathbf{b}_{\omega_{i+1}}) &= \mathbf{b}_{\omega_{i+1}} - \mathbf{b}_{\omega_i} \\ \Sigma_{\mathbf{p}} &= \frac{1}{4} \sigma_{\mathbf{n}_{\mathbf{a}}}^2 t_{\Delta_i}^4 \mathbf{I} \\ \Sigma_{\mathbf{v}} &= \sigma_{\mathbf{n}_{\mathbf{a}}}^2 t_{\Delta_i}^2 \mathbf{I} \\ \Sigma_{\mathbf{b}_{\mathbf{a}}} &= \sigma_{\mathbf{w}_{\mathbf{a}}}^2 t_{\Delta_i}^2 \mathbf{I} \\ \Sigma_{\mathbf{b}_{\omega}} &= \sigma_{\mathbf{w}_{\omega}}^2 t_{\Delta_i}^2 \mathbf{I} \end{aligned} \tag{25}$$

Based on these relative constraints between consecutive frames as defined in (22) and (25), we can define the following energy function with all the motion states $\mathbf{s}_1 \cdots \mathbf{s}_l$ in the sliding window as

$$\begin{aligned} \underset{\mathbf{s}_1 \cdots \mathbf{s}_l}{\arg\min} \sum_{i=1}^{l} \sum_{j \in V_{\mathbf{C}_i}} &||\pi(\mathbf{K}(\mathbf{R}_i(\mathbf{X}_j - \mathbf{p}_i))) - \mathbf{x}_{ij}||_{\delta_{\mathbf{x}}} + \sum_{i=1}^{l-1} ||\mathbf{e}_{\mathbf{q}}(\mathbf{q}_i, \mathbf{q}_{i+1}, \mathbf{b}_{\omega_i})||_{\Sigma_{\mathbf{q}}}^2 \\ &+ \sum_{i=1}^{l-1} ||\mathbf{e}_{\mathbf{p}}(\mathbf{q}_i, \mathbf{p}_i, \mathbf{p}_{i+1}, \mathbf{v}_i, \mathbf{b}_{\mathbf{a}_i})||_{\Sigma_{\mathbf{p}}}^2 + \sum_{i=1}^{l-1} ||\mathbf{e}_{\mathbf{v}}(\mathbf{q}_i, \mathbf{v}_i, \mathbf{v}_{i+1}, \mathbf{b}_{\mathbf{a}_i})||_{\Sigma_{\mathbf{v}}}^2 \\ &+ \sum_{i=1}^{l-1} ||\mathbf{e}_{\mathbf{b}_{\mathbf{a}}}(\mathbf{b}_{\mathbf{a}_i}, \mathbf{b}_{\mathbf{a}_{i+1}})||_{\Sigma_{\mathbf{b}_{\mathbf{a}}}}^2 + \sum_{i=1}^{l-1} ||\mathbf{e}_{\mathbf{b}_{\omega}}(\mathbf{b}_{\omega_i}, \mathbf{b}_{\omega_{i+1}})||_{\Sigma_{\mathbf{b}_{\omega}}}^2 \end{aligned} \tag{26}$$

where l is the size of sliding window, and $||\mathbf{e}||_{\Sigma}^2 = \mathbf{e}^{\top}\Sigma^{-1}\mathbf{e}$ is the squared Mahalanobis distance.

The above derivation assumes we have the inertial measurement $\hat{\mathbf{a}}_i$ and $\hat{\omega}_i$, but actually IMU sensor may be not available. For linear acceleration, we can safely set $\hat{\mathbf{a}}_i = 0$ because abrupt jump rarely happens for AR applications. However, for rotational velocity, since the user may frequently turn around to see the whole AR effect, we cannot set $\hat{\omega}_i = 0$. Instead we align the consecutive SBIs with feature matches to solve a best fit $\hat{\omega}_i$

$$\begin{aligned} \hat{\omega}_i = \underset{\omega}{\arg\min} (&\sum_{x \in \boldsymbol{\Omega}} ||\tilde{I}_i(\mathbf{x}) - \tilde{I}_{i+1}(\pi(\mathbf{K}\mathbf{R}_{\Delta}(\boldsymbol{\omega}, t_{\Delta_i})\mathbf{K}^{-1}\mathbf{x}^h))||_{\delta_{\mathbf{I}}} \\ &+ \sum_{(\mathbf{x}_i, \mathbf{x}_{i+1}) \in M_{i,i+1}} \frac{1}{\delta_{\mathbf{x}}} ||\pi(\mathbf{K}\mathbf{R}_{\Delta}(\boldsymbol{\omega}, t_{\Delta_i})\mathbf{K}^{-1}\mathbf{x}_i^h) - \mathbf{x}_{i+1}||_2^2) \end{aligned} \tag{27}$$

where $\mathbf{R}_{\Delta}(\boldsymbol{\omega}, t_{\Delta})$ is the rotation matrix of (18), and $M_{i,i+1}$ is the set of feature matches between image i and $i+1$. $\delta_{\mathbf{x}}$ controls the weight of the residual error of feature matches, and set to 10 in our experiments. We use Gaussian Newton algorithm to solve (27).

The above optimization framework can easily be extended to incorporate real IMU data. The only difference is that there may be multiple IMU measurements between two consecutive frames. Denote the set of IMU measurements between frame i and $i+1$ as $\{(\hat{\mathbf{a}}_{ij}, \hat{\boldsymbol{\omega}}_{ij}, t_{ij}) | j = 1 \cdots n_i\}$, where $\hat{\mathbf{a}}_{ij}$, $\hat{\boldsymbol{\omega}}_{ij}$ and t_{ij} are j-th linear acceleration, rotational velocity and time stamp respectively. Pre-integration techniques [23, 7] can be used to pre-integrate these IMU measurements, and replace the delta components of (18) and (24) with

$$\begin{aligned} \mathbf{q}_{\Delta}\left(\{(\hat{\boldsymbol{\omega}}_{ij} - \mathbf{b}_{\omega_i}, t_{ij})\}\right) &\approx \mathbf{q}_{\Delta}\left(\{(\hat{\boldsymbol{\omega}}_{ij} - \hat{\mathbf{b}}_{\omega_i}, t_{ij})\}\right) + \frac{\partial \mathbf{q}_{\Delta}}{\partial \mathbf{b}_{\omega}}(\mathbf{b}_{\omega} - \hat{\mathbf{b}}_{\omega}) \\ \mathbf{p}_{\Delta}\left(\{(\hat{\mathbf{a}}_{ij} - \mathbf{b}_{\mathbf{a}_i}, \hat{\boldsymbol{\omega}}_{ij} - \mathbf{b}_{\omega_i}, t_{ij})\}\right) &\approx \mathbf{p}_{\Delta}\left(\{(\hat{\mathbf{a}}_{ij} - \hat{\mathbf{b}}_{\mathbf{a}_i}, \hat{\boldsymbol{\omega}}_{ij} - \hat{\mathbf{b}}_{\omega_i}, t_{ij})\}\right) \\ &\quad + \frac{\partial \mathbf{p}_{\Delta}}{\partial \mathbf{b}_{\mathbf{a}}}(\mathbf{b}_{\mathbf{a}} - \hat{\mathbf{b}}_{\mathbf{a}}) + \frac{\partial \mathbf{p}_{\Delta}}{\partial \mathbf{b}_{\omega}}(\mathbf{b}_{\omega} - \hat{\mathbf{b}}_{\omega}) \\ \mathbf{v}_{\Delta}\left(\{(\hat{\mathbf{a}}_{ij} - \mathbf{b}_{\mathbf{a}_i}, \hat{\boldsymbol{\omega}}_{ij} - \mathbf{b}_{\omega_i}, t_{ij})\}\right) &\approx \mathbf{v}_{\Delta}\left(\{(\hat{\mathbf{a}}_{ij} - \hat{\mathbf{b}}_{\mathbf{a}_i}, \hat{\boldsymbol{\omega}}_{ij} - \hat{\mathbf{b}}_{\omega_i}, t_{ij})\}\right) \\ &\quad + \frac{\partial \mathbf{v}_{\Delta}}{\partial \mathbf{b}_{\mathbf{a}}}(\mathbf{b}_{\mathbf{a}} - \hat{\mathbf{b}}_{\mathbf{a}}) + \frac{\partial \mathbf{v}_{\Delta}}{\partial \mathbf{b}_{\omega}}(\mathbf{b}_{\omega} - \hat{\mathbf{b}}_{\omega}) \end{aligned}$$

where $\hat{\mathbf{b}}_{\mathbf{a}_i}$ and $\hat{\mathbf{b}}_{\omega_i}$ are the state of $\mathbf{b}_{\mathbf{a}_i}$ and $\mathbf{b}_{\omega_i}$ at the pre-integration time. The Jacobians are evaluated at these states, and calculated for once. More details can be found in [7]. Here we assume the gravity component has been excluded and $\hat{\mathbf{a}}_{ij}$ is the user-generated acceleration. It can be obtained by using sensor fusion algorithms like [34] to refine the raw data from accelerometer and gyroscope. In our experiments, we use the IMU measurements provided by an iPhone 6 which have been filtered.

Compared to previous visual-inertial SLAM methods [26, 11, 20], our method does not jointly optimize 3D points in (26), and does not marginalize states moved out of the sliding window either, which makes the optimization very efficient. Normal equations are rather sparse, with only non-zero elements along the diagonal band corresponding to consecutive frames. Again, global localization accuracy is guaranteed by the well-constrained 3D points optimized by BA in the background.

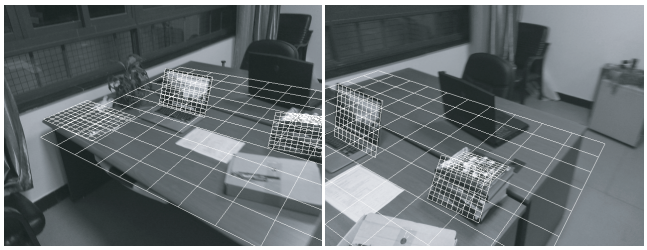


Figure 4: The extracted 3D planes by fitting the recovered 3D points.

5.2 3D Planes Extraction and Global Map Optimization

Similar to [17], we perform local and global BA to refine the map and camera poses of keyframes in the background. For each 3D point in the new keyframe F_k, we first check the maximal ray angle with existing keyframe by (8). If $\max_i \alpha(i,k) \geq \delta_\alpha$ and the 3D position is successfully triangulated, we mark the point well-constrained. Then we expand the existing 3D planes visible in F_k by the new well-constrained 3D points belonging to the plane. To decide whether a 3D point $\mathbf{X}$ belongs to a 3D plane $\mathbf{P}$, we could just check the point-to-plane distance $|\mathbf{n}^\top \mathbf{X} + d|$. However, satisfying the plane equation is only a necessary but not sufficient condition to decide whether $\mathbf{X}$ belongs to $\mathbf{P}$ because plane equation does not contain the plane boundary information. We assume points belonging to the same plane are close to each other in space, and check the point-to-plane distance only for neighbors of those points already assigned to the plane. We define the neighborhood by performing Delaunay triangulation over the 2D features on keyframe F_k. A 3D point $\mathbf{X}$ is added to the set $V_\mathbf{P}$ of plane $\mathbf{P}$ if it satisfies three conditions: 1) $\mathbf{X}$ has not been assigned to any plane, 2) $\mathbf{X}$ is connected to another point in $V_\mathbf{P}$, 3) $|\mathbf{n}^\top \mathbf{X} + d| < \delta_\mathbf{P}$. We set $\delta_\mathbf{P} = 0.01 d_k$ in our experiments, where d_k is the mean depth of keyframe F_k.

We use RANSAC algorithm [5] to extract new 3D planes. At each RANSAC iteration, we randomly sample three connected points to initialize an inlier 3D point set $V_\mathbf{P}$. A 3D plane hypothesis $\mathbf{P}$ is generated from $V_\mathbf{P}$. Then we start an inner loop, in which plane expansion and optimization are alternated iteratively. At each inner iteration, we check the point-to-plane distance for those 3D points $\mathbf{X}$ connected to the points in $V_\mathbf{P}$, and add $\mathbf{X}$ to $V_\mathbf{P}$ if $|\mathbf{n}^\top \mathbf{X} + d| < \delta_\mathbf{P}$. Then we optimize $\mathbf{P}$ using all points in $V_\mathbf{P}$ by minimizing

$$\arg\min_{\mathbf{P}} \sum_{\mathbf{X} \in V_\mathbf{P}} (\mathbf{n}^\top \mathbf{X} + d)^2 \tag{28}$$

and try to expand $V_\mathbf{P}$ again. The inner iteration is repeated until no points can be added to $V_\mathbf{P}$. We discard $\mathbf{P}$ and $V_\mathbf{P}$ if $|V_\mathbf{P}| < 30$. To avoid assigning a point to multiple planes, after RANSAC operation, we sort the extracted planes in descending order of the number of associated 3D point. Denote the sorted inlier set as $\{V_{\mathbf{P}_i} | i = 1 \cdots N_\mathbf{P}\}$. Starting from the first, for each $V_{\mathbf{P}_i}$, if a 3D point $\mathbf{X} \in V_{\mathbf{P}_i}$ is also included in previous planar point set $V_{\mathbf{P}_j}$ $(j < i)$, we remove this point from $V_{\mathbf{P}_i}$. Figure 4 shows the extracted 3D planes.

The 3D plane parameters are also refined in global BA by adding the point-to-plane constraints. We jointly optimize all the camera poses of keyframes, 3D points and 3D planes by minimizing the following energy function with Levenberg Marquart algorithm:

$$\begin{aligned} \arg\min_{\mathbf{C},\mathbf{X},\mathbf{P}} & \sum_{i=1}^{N_F} \sum_{j \in V_i} ||\pi(\mathbf{K}(\mathbf{R}_i(\mathbf{X}_j - \mathbf{p}_i))) - \mathbf{x}_{ij}||_{\delta_\mathbf{x}} \\ & + \sum_{i=1}^{N_\mathbf{P}} \sum_{j \in V_{\mathbf{P}_i}} ||\mathbf{n}_i^\top \mathbf{X}_j + d_i||_{\delta_\mathbf{P}}. \end{aligned} \tag{29}$$

It should be noted that other functions in our paper are also solved by Levenberg Marquart algorithm, except for (3), (5) and (27).

6 Experimental Results

Module	Time per frame
Feature extraction	~ 2 ms
Feature tracking	$2 \sim 8$ ms
Local map expansion and optimization	$2 \sim 4$ ms

Table 1: Process time per frame with a single thread.

We have conducted experiments with several challenging examples on both PC and mobile devices. The test sequences with 640×480 resolution are captured by an iPhone 6 mobile phone. Table 1 shows the time spent in different steps with a single thread on a desktop PC with a Intel(R) Core(TM) i7-4770K CPU @ 3.5GHz and 32GB memory. The feature tracking module generally only takes about 2 ms per frame except when local homography estimation or re-localization is invoked. So the computation in the foreground thread usually takes no more than 8 ms per frame. While running the recorded sequences, we set the frame rate to 60fps for allowing sufficient time for local BA to refine the map. The computation frame rate on an iPhone 6 can be around $20 \sim 50$ fps generally, which is fast enough for many mobile AR applications. The datasets, executable app and SDK can be found in our project webpage[1].

6.1 Qualitative Evaluation

We make a comparison with other state-of-the-art monocular SLAM systems, i.e. ORB-SLAM[2] [27], PTAM [17, 18], LSD-SLAM [4], RDSLAM [33], SVO [6], and DT-SLAM [10]. We directly use the implementation provided by the authors.

Figure 5 shows an indoor example where the camera captured a desktop with fast motion and strong rotation. The initialization ways of different methods may be quite different. PTAM and RD-SLAM require the user to specify two initial keyframes for initialization. ORB-SLAM and DT-SLAM can automatically select two keyframes with sufficient parallax for initialization. LSD-SLAM and SVO directly initialize the map with random or mean depth. Our system can either detect a known marker (e.g. an A4 paper) or use the specified one or two frames for initialization. So in the beginning, we moved the camera from side to side to ensure each SLAM system can successfully initialize the map and camera poses. Then the camera moved around the desktop with frequently sudden rotation and fast movement.

We first compare the results by our method with four different settings for sliding window based optimization: 1) discarding motion prior constraints from (26); 2) $\hat{\mathbf{a}}_i = 0$ and $\hat{\boldsymbol{\omega}}_i = 0$; 3) $\hat{\mathbf{a}}_i = 0$ and $\hat{\boldsymbol{\omega}}$ is estimated by solving (27); 4) using real IMU measurements. As shown in Figures 5(a) and (b), tracking frequently fails in the cases of fast motion and strong rotation if we discard motion prior constraints or directly set $\hat{\boldsymbol{\omega}}_i = 0$. Other SLAM systems also have this problem or suffer from serious drift as shown in (e)-(h). By incorporating motion prior constraints with rotational velocity estimation, our method can faithfully recover the camera poses even in the case of fast camera motion with severe blur as shown in Figure 5(c). With the real IMU measurements, the robustness can be further improved, as shown in (d). We found that PTAM work more robust than other methods (except our method) for this example. The reason may be that PTAM uses pyramid matching and small blur images which is not so sensitive to motion blur. In addition, fronto-parallel patch assumption and delayed triangulation with local map optimization in a background thread work well in this small scene example. The results by RDSLAM, SVO and DT-SLAM are included in the supplementary video.

[1] http://www.zjucvg.net/rkslam/rkslam.html

[2] We use the source code of ORB-SLAM2 from the website https://github.com/raulmur/ORB_SLAM2

Table 2: Localization error comparison in the TUM RGB-D dataset. From left to right: RMSE (cm) of keyframes, the starting ratio (i.e. dividing the initialization frame index by the total frame number), and the tracking success ratio after initialization.

Group	Sequence	RKSLAM	ORB-SLAM	PTAM	LSD-SLAM
A	fr1_xyz	0.61/0%/100%	1.05/0%/100%	1.29/0%/100%	7.64/0%/100%
A	fr2_xyz	0.43/0%/100%	0.23/0%/100%	0.29/0%/100%	6.32/0%/100%
A	fr3_sitting_xyz	1.98/0%/92%	1.31/5%/100%	X	9.12/0%/100%
B	fr1_desk	1.69/0%/100%	1.40/12%/100%	2.71/0%/44%	3.86/27%/100%
B	fr2_desk	10.10/0%/97%	0.78/6%/100%	0.55/0%/20%	17.41/0%/100%
B	fr3_long_office	2.48/0%/100%	2.17/0%/100%	0.82/0%/31%	36.04/30%/100%
C	fr1_rpy	1.26/0%/100%	5.53/4%/84%	X	3.26/0%/11%
C	fr2_rpy	0.41/0%/100%	0.23/32%/100%	0.56/0%/100%	3.71/0%/25%
C	fr3_sitting_rpy	1.44/0%/100%	0.19/93%/100%	2.44/0%/93%	3.36/0%/89%
D	fr1_360	11.81/0%/95%	8.16/5%/11%	X	8.25/0%/5%
D	fr2_360_hemisphere	17.48/0%/88%	12.27/1%/65%	76.50/0%/33%	25.64/0%/19%
D	fr2_pioneer_360	20.24/0%/86%	1.40/69%/46%	59.09/0%/98%	30.62/0%/41%

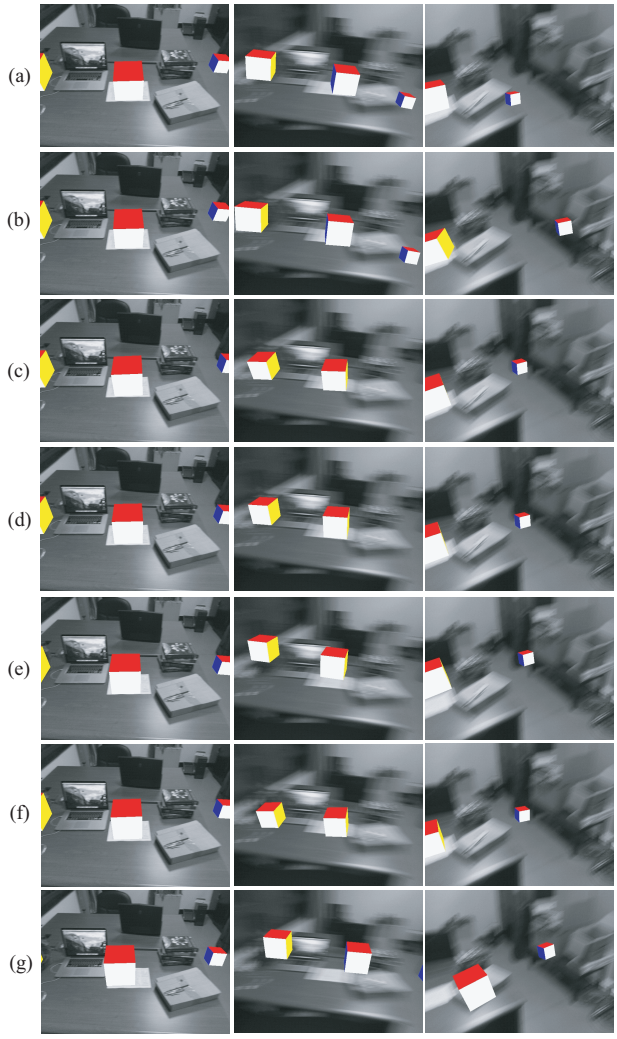


Figure 5: SLAM result comparison of an indoor example by inserting virtual cubes. (a) The result by RKSLAM without motion prior constraints. (b) The result by RKSLAM with setting $\hat{\mathbf{a}}_i = 0$ and $\hat{\boldsymbol{\omega}}_i = 0$. (c) The result by RKSLAM with rotational velocity estimation. (d) The result by RKSLAM with real IMU measurements. (e) The result by ORB-SLAM. (f) The result by PTAM. (g) The result by LSD-SLAM.

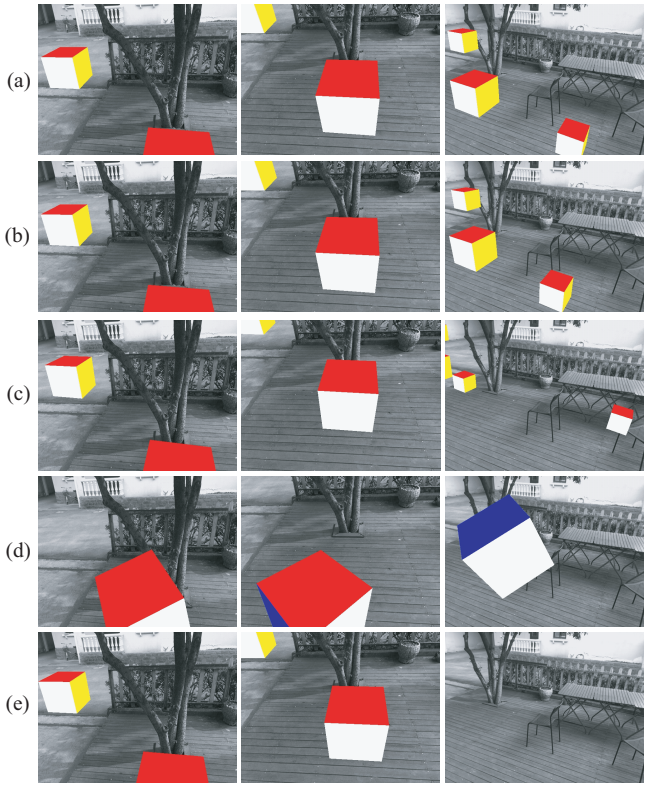
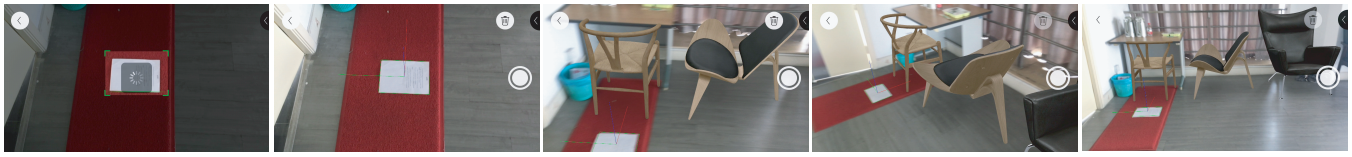


Figure 6: SLAM result comparison by inserting virtual cubes. (a) The result by RKSLAM with rotational velocity estimation. (b) The result by RKSLAM with real IMU measurements. (c) The result by ORB-SLAM. (d) The result by PTAM. (e) The result by LSD-SLAM.

Figure 6 shows another challenging outdoor example whose scale is much larger than the above indoor example. Again, our method works rather well in this challenging example and significantly outperforms previous methods. Even without IMU measurements, our method still can faithfully recover the camera poses in almost all frames. With real IMU measurements, the result is further slightly improved, as shown in Figure 6(b). Please refer to our supplementary video to watch more complete comparison and another indoor example.



(a) (b) (c) (d) (e)

Figure 7: AR application on an iPhone 6. (a) The camera center is toward the A4 paper on the ground. (b) The recognized A4 paper with map initialization. (c)-(e) Inserting three different chairs into the scene and watching them from three different viewpoints.

6.2 Quantitative Evaluation

We use the TUM RGB-D dataset [32] to perform quantitative evaluation and make comparison with ORB-SLAM, PTAM, and LSD-SLAM. We select 12 sequences, classified into 4 groups according to their motion type: A) simple translation; B) there are loops; C) slow and nearly pure rotation; D) fast motion with strong rotation. We only use RGB information for all methods. The RMSE of keyframes and the completeness of camera trajectories are computed and listed in Table 2. To evaluate the trajectory completeness, we compute the starting ratio (i.e dividing the initialization frame index by the total frame number), and the tracking success ratio after initialization.

We define tracking failure as if the camera pose is not successfully estimated (for other methods) or relative localization error to the keyframe with maximum common features is larger than 10cm (for our method, actually more strict). For the sequences in group A, most methods obtain good results. The success ratio of all other methods is low for the sequences in group D, which are most challenging and quite similar to our sequences. In contrast, our success ratio is much higher, and our RMSE is also good in most sequences. Although ORB-SLAM has lower RMSE in more sequences, the success ratio is rather low for the sequences in group D. For the sequences in group B, our RMSE is larger than that of ORB-SLAM which uses more expensive ORB features to better handle loop closure.

6.3 AR Application

Here we show an AR application that the user can use a mobile device to capture the scene and then insert one or multiple 3D furniture models into his home to "see" the effect of furniture placing without imagination. Since the real size of the recovered 3D structure by monocular SLAM is unknown, we need a method to precisely estimate the scale so that the 3D furniture models can be inserted into the scene with accurate size. However, even with the noisy IMU measurements provided by a mobile device, the real size is not easy to be precisely estimated immediately. A more reliable method is to use a known size marker. In our AR application system, we choose to use an A4 paper which can be easily obtained by users. Making the camera center toward the A4 paper in the beginning, our AR system can automatically detect its four edges with camera pose estimation, as shown in Figures 7(a) and (b). After initialization, the user can freely download the 3D models in the item list and insert them into the scene, as shown in Figures 7(c)-(e). Please refer to the supplementary video for the better presentation of this result.

7 Conclusions

In this paper, we have presented a novel keyframe-based monocular SLAM system which can robustly handle fast camera motion and strong rotation. In order to achieve this objective, we contribute a novel multi-homography based feature tracking method which can utilize the global and local planarity of the scenes to guide the matching. This feature tracking method not only can robustly handle fast camera motion and strong rotation but also very efficient to allow real-time computation even on a mobile device. In addition, we propose a novel sliding-window based local map expansion and optimization framework, which not only can triangulate and optimize the new 3D points immediately without delay but also can utilize the motion prior constraints among neighboring frames to improve the robustness of camera pose estimation by simulating IMU measurements. This framework also can be extended to incorporate real IMU measurements if available to further improve the robustness and accuracy.

Similar to other feature point based visual SLAM methods, if the scene is extremely textureless or there are many repeated textures/structures, our system may fail. If IMU measurements are available, this problem can be alleviated. In our future work, we also would like to combine quasi-dense matching or direct tracking method like [4] to alleviate this problem.

Acknowledgments

The authors would like to thank all the reviewers for their constructive comments to improve this paper, and Hangzhou Pink Elephant Digital Technology Co., Ltd for providing furniture 3D models. This work was supported in part by the National Key Technology Research and Development Program of the Ministry of Science and Technology of China under Grant 2014BAK14B01, NSF of China (Nos. 61232011 and 61272048), the Fundamental Research Funds for the Central Universities (Grant No. 2015XZZX005-05), and a Foundation for the Author of National Excellent Doctoral Dissertation of PR China (Grant No. 201245).

References

[1] A. B. Chatfield. *Fundamentals of high accuracy inertial navigation*. AIAA, 1997.

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

[3] T.-C. Dong-Si and A. I. Mourikis. Motion tracking with fixed-lag smoothing: Algorithm and consistency analysis. In *IEEE International Conference on Robotics and Automation*, pages 5655–5662. IEEE, 2011.

[4] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *13th European Conference on Computer Vision, Part II*, pages 834–849. Springer, 2014.

[5] 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*, 24(6): 381–395, 1981.

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

[7] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza. IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. In *Robotics: Science and Systems*, 2015.

[8] S. Gauglitz, C. Sweeney, J. Ventura, M. Turk, and T. Hollerer. Live tracking and mapping from both general and rotation-only camera motion. In *IEEE International Symposium on Mixed and Augmented Reality*, pages 13–22. IEEE, 2012.

[9] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2004.

[10] C. Herrera, K. Kim, J. Kannala, K. Pulli, J. Heikkila, et al. DT-SLAM: Deferred triangulation for robust SLAM. In *IEEE International Conference on 3D Vision*, volume 1, pages 609–616, 2014.

[11] J. A. Hesch, D. G. Kottas, S. L. Bowman, and S. I. Roumeliotis. Consistency analysis and improvement of vision-aided inertial navigation. *IEEE Transactions on Robotics*, 30(1): 158–176, 2014.

[12] G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis. Analysis and improvement of the consistency of extended kalman filter based SLAM. In *IEEE International Conference on Robotics and Automation*, pages 473–479, 2008.

[13] V. Indelman, S. Williams, M. Kaess, and F. Dellaert. Information fusion in navigation systems via factor graph based incremental smoothing. *Robotics and Autonomous Systems*, 61(8):721–738, 2013.

[14] Y. Jin, L. Tao, H. Di, N. I. Rao, and G. Xu. Background modeling from a free-moving camera by multi-layer homography algorithm. In *15th IEEE International Conference on Image Processing*, pages 1572–1575, 2008.

[15] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert. iSAM2: Incremental smoothing and mapping using the bayes tree. *International Journal of Robotics Research*, 31(2):216–235, 2012.

[16] C. Kerl, J. Stuckler, and D. Cremers. Dense continuous-time tracking and mapping with rolling shutter RGB-D cameras. In *IEEE International Conference on Computer Vision*, pages 2264–2272, 2015.

[17] G. Klein and D. W. Murray. Parallel tracking and mapping for small AR workspaces. In *6th IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 225–234, 2007.

[18] G. Klein and D. W. Murray. Improving the agility of keyframe-based SLAM. In *10th European Conference on Computer Vision, Part II*, pages 802–815. Springer, 2008.

[19] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

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

[21] M. Li and A. I. Mourikis. High-precision, consistent EKF-based visual-inertial odometry. *The International Journal of Robotics Research*, 32(6):690–711, 2013.

[22] S. Lovegrove, A. Patron-Perez, and G. Sibley. Spline fusion: A continuous-time representation for visual-inertial fusion with application to rolling shutter cameras. In *British Machine Vision Conference*. BMVA Press, 2013.

[23] T. Lupton and S. Sukkarieh. Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions. *IEEE Transactions on Robotics*, 28(1):61–76, 2012.

[24] M. Meilland, T. Drummond, and A. I. Comport. A unified rolling shutter and motion blur model for 3D visual registration. In *IEEE International Conference on Computer Vision*, pages 2016–2023, 2013.

[25] J.-M. Morel and G. Yu. ASIFT: A new framework for fully affine invariant image comparison. *SIAM Journal on Imaging Sciences*, 2(2):438–469, 2009.

[26] A. I. Mourikis and S. I. Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In *IEEE International Conference on Robotics and Automation*, pages 3565–3572. IEEE, 2007.

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

[28] C. Pirchheim, D. Schmalstieg, and G. Reitmayr. Handling pure camera rotation in keyframe-based SLAM. In *IEEE International Symposium on Mixed and Augmented Reality*, pages 229–238, 2013.

[29] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *9th European Conference on Computer Vision, Part I*, pages 430–443. Springer, 2006.

[30] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: an efficient alternative to SIFT or SURF. In *IEEE International Conference on Computer Vision*, pages 2564–2571. IEEE, 2011.

[31] H. Strasdat, J. M. Montiel, and A. J. Davison. Visual SLAM: why filter? *Image and Vision Computing*, 30(2):65–77, 2012.

[32] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *IEEE/RSJ International Conference on Intelligent Robot Systems*, pages 573–580, Oct. 2012.

[33] W. Tan, H. Liu, Z. Dong, G. Zhang, and H. Bao. Robust monocular SLAM in dynamic environments. In *IEEE International Symposium on Mixed and Augmented Reality*, pages 209–218, 2013.

[34] N. Trawny and S. I. Roumeliotis. Indirect kalman filter for 3D attitude estimation. Technical Report 2005-002, University of Minnesota, Dept. of Comp. Sci. & Eng., Mar. 2005.

[35] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustment–a modern synthesis. In *Vision algorithms: theory and practice*, pages 298–372. Springer, 1999.

[36] D. Zamalieva, A. Yilmaz, and J. W. Davis. A multi-transformational model for background subtraction with moving cameras. In *European Conference on Computer Vision*, pages 803–817. Springer, 2014.

[37] G. Zhang, Z. Dong, J. Jia, T. Wong, and H. Bao. Efficient non-consecutive feature tracking for structure-from-motion. In *11th European Conference on Computer Vision, Part V*, pages 422–435. Springer, 2010.