# FMD Stereo SLAM: Fusing MVG and Direct Formulation Towards Accurate and Fast Stereo SLAM

**Conference Paper** · January 2019

CITATIONS
0

READS
196

**3 authors:**

Fulin Tang
Chinese Academy of Sciences
**6** PUBLICATIONS **7** CITATIONS
SEE PROFILE

Heping Li
Chinese Academy of Sciences
**34** PUBLICATIONS **102** CITATIONS
SEE PROFILE

Yihong Wu
Chinese Academy of Sciences
**80** PUBLICATIONS **864** CITATIONS
SEE PROFILE

# FMD Stereo SLAM: Fusing MVG and Direct Formulation Towards Accurate and Fast Stereo SLAM

Fulin Tang, Heping Li, Yihong Wu*

***Abstract*— We propose a novel stereo visual SLAM framework considering both accuracy and speed at the same time. The framework makes full use of the advantages of key-feature-based multiple view geometry (MVG) and direct-based formulation. At the front-end, our system performs direct formulation and constant motion model to predict a robust initial pose, reprojects local map to find 3D-2D correspondence and finally refines pose by the reprojection error minimization. This front-end process makes our system faster. At the back-end, MVG is used to estimate 3D structure. When a new keyframe is inserted, new mappoints are generated by triangulating. In order to improve the accuracy of the proposed system, bad mappoints are removed and a global map is kept by bundle adjustment. Especially, the stereo constraint is performed to optimize the map. This back-end process makes our system more accurate. Experimental evaluation on EuRoC dataset shows that the proposed algorithm can run at more than 100 frames per second on a consumer computer while achieving highly competitive accuracy.**

## I. INTRODUCTION

In recent years, visual odometry(VO) [1] and simultaneous localization and mapping(SLAM) [2] have been more and more popular because of their wide applications in augmented reality(AR), unmanned aerial vehicles(UAVs), unmanned ground vehicles(UGVs) and robot visual navigation. There are many different sensor types for VO and SLAM, such as monocular cameras [3], stereo cameras [4] and RGB-D cameras [5]. Generally, stereo VO/SLAM can directly estimate the global scale by using the length of the baseline between left camera and right camera. However, monocular visual odometry can not directly estimate the global scale and has to rely on other sensors, such as IMU [6], to estimate the global scale. Although RGB-D visual odometry can directly estimate the global scale, depth capture from RGB-D cameras is limited within a certain range. In this paper, we focus our attention on stereo visual SLAM, which is widely used in the field of robotics.

At present, two kinds of studies on SLAM, geometric SLAM and learning SLAM, are very active [7]. Due to the wide successful use of deep learning, learning SLAM has attracted many interests. It depends on lots of labeled datasets and cannot be applied well in a strange environment. In geometric SLAM, keyframe-based SLAM gains in popular [8]. Keyframe-based SLAM includes key-feature-based SLAM and direct SLAM. Direct SLAM considers the whole image or some pixels with larger gradients and is more robust in poorly-textured environments. But its running speed is affected greatly by the photometric error optimizations. The initial value range is reasonable and then the optimization can be convergent to an accurate value faster. Key-feature-based SLAM extracts key corners or edges and then makes matching to compute maps and camera poses by using multiple view geometry (MVG) theory [9] including epipolar geometry, triangulation, structure from motion, bundle adjustment etc. If the extracted key features and their matching are reliable, SLAM will have higher accuracy. Therefore, key-feature-based SLAM and direct SLAM have their own advantages. Moreover, multiple view geometry has the rigourous theory for 3D computer vision. However, most of the existing SLAM systems are either fast or accurate. There are seldom methods considering both speed and accuracy at the same time. In this paper we would like to fuse all these advantages of MVG, key-feature-based SLAM, and direct SLAM to give a practical SLAM system with both higher accuracy and faster speed.

SVO [10],[11], which is called semi-direct visual odometry, combines key features and direct formulation. It can run at more than 300 frames per second on a consumer computer. SVO uses the depth filter [12] to estimate depth. The depth filter may not converge to true depth for its noisy initialization that will affect later localization accuracy. ORBSLAM [13],[14], a feature-based typical system using MVG, detects FAST corners [15] and extracts ORB descriptors [16] on each frame. It is more accurate than SVO. But it runs at about 20 frames per second on a consumer computer. Recently, direct sparse odometry (DSO) fuses bundle adjustment and sparse pixels with larger gradients achieving high accuracy [17]. Its speed on a consumer computer is similar with ORBSLAM.

Inspired by these excellent VO/SLAM systems, we fuse their advantages to give a practical stereo SLAM system taking into account both accuracy and speed. We propose a novel framework. At the front-end, speed is considered. Our system performs the photometric error minimization and constant motion model to predict initial pose, reprojects local map to find 3D-2D correspondence and finally refines poses by the reprojection error minimization. At the back-end, accuracy is considered. A global map is kept by bundle adjustment in our system. The global map can not only greatly improve the localization accuracy but also can be used as

*The authors are with National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, University of Chinese Academy of Sciences, China. The corresponding author is Yihong Wu, yhwu@nlpr.ia.ac.cn. This work was supported by the National Natural Science Foundation of China under Grant Nos. 61836015, 61572499, 61421004.

an off-line map for visual navigation. Especially, to further improve the accuracy of the proposed system, the stereo constraint is used to optimize the global map. Experimental results on EuRoC dataset show that our algorithm achieves highly competitive accuracy while running at more than 100 frames per second on a consumer computer, proving that our algorithm is suitable well to indoor complex cluttered environments and short-baseline stereo cameras.

The remainder of this paper is organized as follows. In Section II, we review the related work. An overview of the proposed framework is given in Section III. In Section IV, we describe the tracking part in detail. The mapping procedure will be introduced in Section V. We provide the experimental comparison with some state-of-the-art algorithms in Section VI. Finally, in Section VII, we describe conclusions and future work.

## II. RELATED WORK

In this section, we describe the related work on S-LAM/VO methods. Davison et al [18] first presented a real-time monocular slam system, which is based on filter theory. Klein and Murray [19] used MVG to propose real-time Parallel Tracking And Mapping (PTAM), which is based on keyframes. After a few years, filter-based SLAM and keyframe-based SLAM were compared by [20]. Later, keyframe-based SLAM attracts more and more interests. It includes key-feature-based SLAM, direct SLAM, semi-direct SLAM.

Key-feature-based methods use MVG to estimate 3D structure and to optimize on reprojection errors. A lead method was given in Mouragnon et al [21]. PTAM [19] uses FAST corners points [15] as features, which is suitable for small desktop workspace in application. Later, PTAM was extended to stereo PTAM [22]. Strasdat et al. [23] presented a filter-based approach for feature initialisation within keyframe-based SLAM to integrate scales. A double window optimisation for constant time visual SLAM was proposed in [24]. Mei et al. [4] developed a relative SLAM approach for large-scale mapping in constant-time using stereo. Mur-Artal et al presented ORBSLAM [13] which uses ORB for correspondences. In the tracking thread, PnP [25] or frame-to-frame tracking is used to compute the pose of current frame. And then the pose is refined by tracking local map to find more 3D-2D correspondence. In the local mapping thread, a sparse 3D map is constructed by matching ORB descriptors and triangulating the matching pairs. ORBSLAM2 [14] is an extension to ORBSLAM, which includes monocular slam system, stereo slam system, and RGB-D slam system.

Direct methods use filter of inverse depth parametrization [26] to estimate 3D structure. The camera pose is estimated directly by minimizing an photometric error measure that is based on the image's pixel-level intensities [27]. DTAM [28] is a system for real-time tracking and reconstruction by GPU which uses all pixels in the image. LSD-SLAM [29] uses part of pixels in the image and directly operates on image intensities both for tracking and mapping. It can runs in real-time on a CPU and even on a modern smartphone. DSO [17] is a novel direct sparse visual odometry for real-time tracking and reconstruction by CPU which only uses sparse pixels in the image.

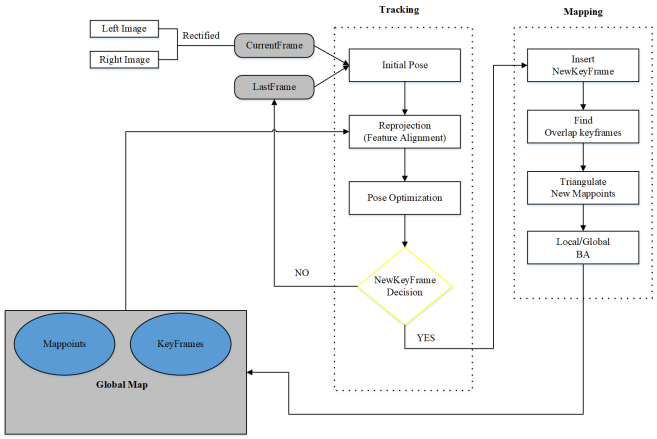



Fig. 1. The proposed framework: tracking and mapping pipelines.

Semi-direct methods combine key-feature-based methods and direct methods. SVO [10],[11] extracts features to initialize new 3D points for keyframes and uses feature correspondences as an implicit result of direct motion estimation.

## III. THE PROPOSED FRAMEWORK

We give a brief overview of the proposed framework. The whole system is split into tracking thread and mapping thread, which is shown in Fig. 1.

In the tracking thread, the camera pose is estimated. The first step is to get predicted pose of current frame by sparse model-based image alignment. If sparse model-based image alignment fails, we use constant motion model to predict the pose of current frame. The second step is to reproject local map to current frame and find 3D-2D correspondence. The third step is to refine the pose of current frame by reprojection error minimization. The final step is to judge the current frame whether is a keyframe. If it is a keyframe, we perform stereo matching to get depth and insert it to the mapping thread.

In the mapping thread, MVG is used to estimate 3D structure. First, we find the keyframes which are close to current keyframe. Then we perform feature matching to find 2D-2D correspondence between keyframes. After that, triangulation is performed to generate new mappoints. Finally, we optimize the local map including the mappoints and the poses of keyframes. After a period of time, we refine the pose and reconstruction globally by using global bundle adjustment.

## IV. TRACKING

Let us denote the intensity image in the $k$-th frame as $I_k$. The camera pose with respect to the world's reference coordinate in the $k$-th timestamp is denote as $\mathbf{T}_{k,w}$. $C_k$ and $C_{k-1}$ are the camera coordinate systems at two consecutive

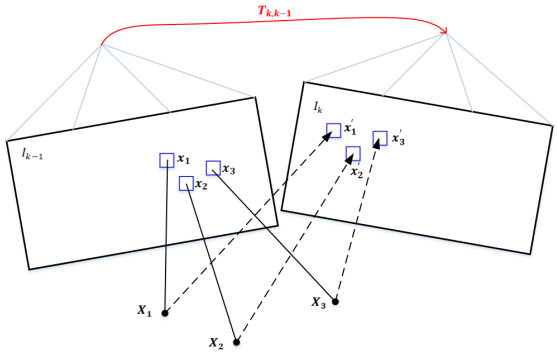


Fig. 2. Sparse model-based image alignment is performed to find the relative camera pose $\mathbf{T}_{k,k-1}$ that minimizes the photometric difference between image patches corresponding to the same 3D point(blue squares). This figure is adapted from [10].

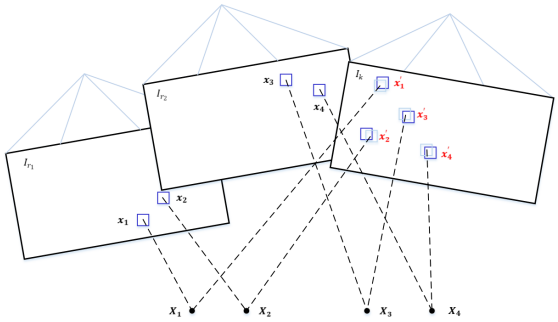


Fig. 3. Reprojection : because of inaccuracies in the 3D point and camera pose estimation, the 2D position of each point is optimized individually to minimize the photometric error in its patch. This figure is adapted from [10].

frames, which are related by the relative pose transformation in the special Euclidean group $\mathbf{T}_{k,k-1}(\xi) \in \mathrm{SE}(3)$. $\xi \in \mathfrak{se}(3)$ is the 6-vector of coordinates in the Lie algebra $\mathfrak{se}(3)$. We denote the feature point in the image as $\mathbf{x}$, its space point in the camera coordinate as ${}_c\mathbf{X}$ and its space point in the world coordinate as ${}_w\mathbf{X}$. The camera reprojection model is denote as $\pi$:

$$ {}_c\mathbf{X} = \mathbf{T}_{k,w} * {}_w\mathbf{X}, \tag{1} $$

$$ \mathbf{x} = \pi({}_c\mathbf{X}), \tag{2} $$

where $\mathbf{x}$ is 2D position of feature point.

### A. Initial Pose

In order to get initial pose of current frame, the relative camera pose between two consecutive frames, $\mathbf{T}_{k,k-1}(\xi)$, is estimated by sparse model-based image alignment, which is similar to [10]. Then the initial pose of current frame can be computed with

$$ \mathbf{T}_{k,w} = \mathbf{T}_{k,k-1} * \mathbf{T}_{k-1,w}. \tag{3} $$

The photometric error is used to estimate the relative camera pose $\mathbf{T}_{k,k-1}$ in sparse model-based image alignment. This is depicted in Fig. 2. First, we find space points observed by the previous frame. Then, the space points from the previous frame is reprojected to the current frame. The intensity residual $\delta I$ is defined by the photometric difference between pixels observing the same 3D point. It can be computed by projecting 3D point ${}_c\mathbf{X}$ from the previous image $I_{k-1}$ and subsequently projecting it into the current image. The residual is as follows:

$$ \delta I(\mathbf{T}, \mathbf{x}) = I_k(\pi(\mathbf{T} * {}_c\mathbf{X})) - I_{k-1}(\mathbf{x}) \quad \mathbf{x} \in \Omega, \tag{4} $$

where $\mathbf{T}$ is short for $\mathbf{T}_{k,k-1}$. $\mathbf{x}$, which is correspondence to ${}_c\mathbf{X}$, is the location of the feature point (Fast Corners) in the previous frame. $\Omega$ is the set of image points for which the space points are known at time $k-1$. We use $4*4$ patches to describe the feature point. Because of small frame-to-frame motions and small patch-sizes, we do not warp the pathes. The $4*4$ patches around the feature point is denoted as the vector $I(\mathbf{x}_i)$. Minimizing the photometric error of all patches is performed to obtain the relative camera pose $\mathbf{T}_{k,k-1}$:

$$ \mathbf{T}_{k,k-1} = \underset{\mathbf{T}_{k,k-1}}{\arg\min} \frac{1}{2} \sum_{i \in \Omega} \| \delta I(\mathbf{T}_{k,k-1}, \mathbf{x}_i) \|^2, \tag{5} $$

which is depicted in Fig. 2. We solve it in an iterative Gauss-Newton method and $\mathbf{T}_{k,k-1}$ is parameterized with the $6*1$ vector $\xi \in \mathfrak{se}(3)$. In order to accurate the computation, we perform the inverse compositional formulation [30] of intensity residual, where the jacobian matrix is computed only once. Similar to [30], the update step is performed as follows.

$$ \mathbf{T}_{k,k-1} = \mathbf{T}_{k,k-1} * \mathbf{T}(\xi)^{-1}, \tag{6} $$

where $\xi$ is mapped to SE(3) by the exponential map [31]:

$$ \mathbf{T}(\xi) = exp(\xi). \tag{7} $$

During the above process, if sparse model-based image alignment fails, we use constant motion model to estimate the initial pose of current frame:

$$ \mathbf{T}_{k,w} = \mathbf{T}_{k-1,w} * \mathbf{T}_{k-2,w}^{-1} * \mathbf{T}_{k-1,w}. \tag{8} $$

### B. Reprojection

In Section A, by only using the information from the previous frame and the current frame, we obtain initial pose which is not accurate. And the accuracy of current pose can be improved. To reduce the drift, information from map is used to refine the pose of current frame.

First, we find keyframes which are close to current frame. Then, all space points of close keyframes which are visible from their estimated camera poses are projected into the current image. As it is depicted in Fig. 3, we get an estimate of the corresponding 2D feature positions $\mathbf{x}_i'$. Similar to the last step, the patch is used to describe 2D feature. We use $8*8$ patch to describe 2D feature, which is different from the last step. What's more, because the distance between the current frame and the close keyframes are typically farther away than the distance between the current frame and the previous frame, a larger patch is used. An affine warping $\mathbf{A}_i$ is applied to the reference patch in keyframes. Finally,

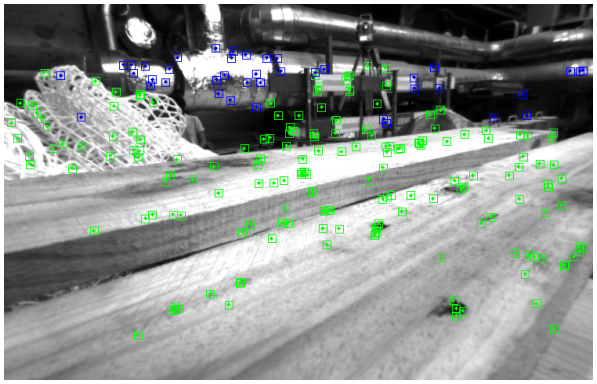


Fig. 4. Tracking points in EuRoC [32]. Green points have a depth less than 40 times the stereo baseline, while blue points are further away. In other words, green points are from stereo keypoints and blue points are from monocular keypoints.

according to corresponding patches, the 2D location of each feature is refined individually by minimizing the photometric error between the patch in the current frame and the patch in the reference keyframe r:

$$\mathbf{x}_i^{'} = \arg\min_{\mathbf{x}_i^{'}} \frac{1}{2} \| I_k(\mathbf{x}_i^{'}) - \mathbf{A}_i * I_r(\mathbf{x}_i) \|. \tag{9}$$

Also, we solve it in an iterative Gauss-Newton method and perform the inverse compositional formulation [30] of intensity residual to accurate optimization.

### C. Pose Optimization

In Section B, we refine 2D positions of feature points which are corresponding to space points in the world coordinate. However, these 3D-2D correspondences violate the epipolar constraints. Therefore, minimizing reprojection error is performed to refine the pose of current frame:

$$\mathbf{T}_{k,w} = \arg\min_{\mathbf{T}_{k,w}} \frac{1}{2} \sum_i \| \mathbf{x}_i - \pi(\mathbf{T}_{k,w} *_w \mathbf{X}) \|^2. \tag{10}$$

The pose of current frame is only optimized, which is known as motion-only BA [20]. It can be solved iteratively with Gauss-Newton method or Levenberg-Marquardt method.

### D. New Keyframe Decision

In this step, we decide whether the current frame is considered as a new keyframe. To decide the current frame as a new keyframe, all the following conditions must be satisfied:

1). There are more than 50 inlier points tracked in the current frame.

2). The mapping thread must be idle.

3). A minimum visual change is met. The parallax angle between current frame and last keyframe is more than two degrees.

Compared with ORBSLAM [13], our strategies of selecting keyframes are faster and more effective.

Once the new keyframe is selected, we construct the new keyframe. We extract ORB features on the left image and right image which are rectified. In our system, monocular keypoints are defined by two coordinates $\mathbf{x}_m = (u_L, v_L)$, which is on the left image and the right image. Stereo keypoints are defined by three coordinates $\mathbf{x}_s = (u_L, v_L, u_R)$, where $(u_L, v_L)$ is on the left image and $u_R$ is the horizontal coordinate on the right image.

Stereo matching between monocular keypoints on the left image and keypoints on the right image is performed to generate stereo keypoints. If a monocular keypoint $(u_L, v_L)$ on the left image is matched with monocular keypoint $(u_R, v_R)$ on the right image, the depth can be computed as:

$$d = \frac{f_x * b}{u_L - u_R}, \tag{11}$$

where $f_x$ is the horizontal focal length and b is the length of baseline between left camera and right camera. If the computed depth is less than 40 times the length of stereo baseline, the monocular keypoint $(u_L, v_L)$ on the left image is considered as the stereo keypoint $(u_L, v_L, u_R)$. Once there are stereo keypoints with depth, we use the stereo keypoints to create new mappoints.

The remaining keypoints which are monocular keypoints on the left image are matched between two keyframes. Triangulating from multiple views is performed to generate new mappoints, which is detailed in Section V. In summary, close space points are generated by stereo matching to compute the depth. Far space points are generated by triangulating from multiple views. In other words, close space points are corresponding to stereo keypoints. Far space points are corresponding to monocular keypoints. It is depicted in Fig. 4.

## V. MAPPING

In this section, when a new keyframe is inserted, the mapping thread is performed.

### A. Inserting New Keyframe

When current frame is considered as a new keyframe in the tracking thread, it is inserted into the mapping thread. In the new keyframe, although there are 3D-2D correspondences which are from the tracking thread, there are not descriptors for some 2D features. We extract ORB for the 2D features, which will help in the data association for relocalizaiton. Finally, the new keyframe is added into the map.

### B. Finding Overlap Keyframes

In order to generate far space points, finding overlap keyframes with a new keyframe is very important. Here, to find overlap keyframes with a new keyframe fast and effectively, pose graph is not generated in the map. We just calculate the distance between a new keyframe and other keyframes to decide which keyframe is overlapping with the new keyframe. The closest ten keyframes are considered as overlap keyframes with the current keyframe.

### C. Triangulating

In this step, because most of the time the depth filter does not converge with true depths for its noisy initialization and fast motion, which has a bad effect on localization accuracy, we do not use a depth filter to estimate depths. In our system, triangulating is performed to estimate depths and generate new mappoints. We match ORB features between a current keyframe and overlap keyframes. At this stage, some outliers are rejected by epipolar constraint, non-maximum suppression and cross check. After matching, the matching pairs are triangulated to generate new space points. Depth positivity in both cameras, parallax and reprojection errors are checked to discard outliers.

### D. Bundle Adjustment

We perform local bundle adjustment to optimize a local window of keyframes and space points from the local window of keyframes. In order to keep the algorithm efficient, 10 keyframes are fixed to perform local bundle adjustment. After a period of time, global bundle adjustment is performed to optimize all keyframes and all mappoints so that a global map is kept. Especially, stereo constraint is performed to improve the system. The projection function $\pi_m$ for far space points corresponding to monocular keypoints and $\pi_s$ for near space points corresponding to stereo keypoints are defined as follows:

$$\pi_m\left(\begin{bmatrix} {}_c\mathbf{X}(1) \\ {}_c\mathbf{X}(2) \\ {}_c\mathbf{X}(3) \end{bmatrix}\right) = \begin{bmatrix} f_x * \frac{{}_c\mathbf{X}(1)}{{}_c\mathbf{X}(3)} + c_x \\ f_y * \frac{{}_c\mathbf{X}(2)}{{}_c\mathbf{X}(3)} + c_y \end{bmatrix}, \tag{12}$$

$$\pi_s\left(\begin{bmatrix} {}_c\mathbf{X}(1) \\ {}_c\mathbf{X}(2) \\ {}_c\mathbf{X}(3) \end{bmatrix}\right) = \begin{bmatrix} f_x * \frac{{}_c\mathbf{X}(1)}{{}_c\mathbf{X}(3)} + c_x \\ f_y * \frac{{}_c\mathbf{X}(2)}{{}_c\mathbf{X}(3)} + c_y \\ f_x * \frac{{}_c\mathbf{X}(1)-b}{{}_c\mathbf{X}(3)} + c_x \end{bmatrix}, \tag{13}$$

where ($f_x$, $f_y$) is the focal length, ($c_x$, $c_y$) is the principal point, and b is the length of the baseline.

In terms of local bundle adjustment, the optimization problem is represented as follows:

$$\{{}_w\mathbf{X}^i, \mathbf{T}_j | i \in P_l, j \in K_l\} = \underset{{}_w\mathbf{X}^i, \mathbf{T}_j}{\arg\min} \sum_{j \in K_l} \sum_{i \in P_l} \rho(E(i,j)), \tag{14}$$

$$E(i,j) = \|x_m^i - \pi_m(\mathbf{T}_j * {}_w\mathbf{X}^i)\|^2, \tag{15}$$

$$E(i,j) = \|x_s^i - \pi_s(\mathbf{T}_j * {}_w\mathbf{X}^i)\|^2, \tag{16}$$

where $K_l$ is the local window of keyframes and $P_l$ is all the space points observed in the local window of keyframes. $E(i,j)$ is reprojection error, which is calculated by two ways. We use (15) to calculate reprojection error for monocular keypoints and (16) to calculate reprojection error for stereo keypoints.

The process of global bundle adjustment is similar to local bundle adjustment, except the first keyframe whose pose is fixed in the optimization. We use Levenberg-Marquardt method, which is implemented in ceres [33], to perform local bundle adjustment and global bundle adjustment. After the optimization, outliers are removed to keep an accurate map.

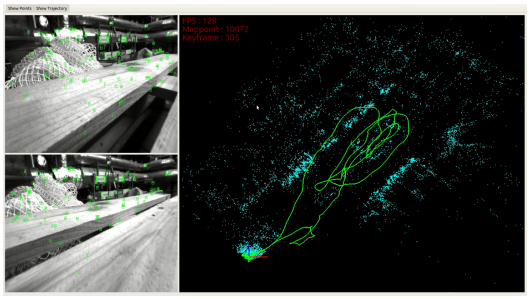
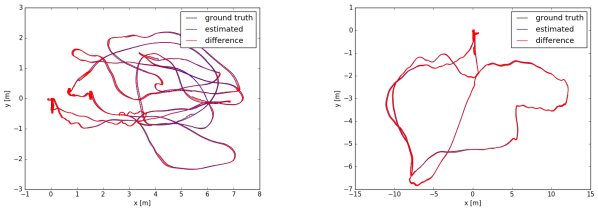


Fig. 5. Visualization of the implementation of our system on MH_01_easy. The left top image is the left image and the left bottom image is the right image. The green points on the left image and right image are 2D inlier points which are corresponding to 3D points. In the right, the green curve is the camera trajectory and the blue points are space points. As it is shown, a global map is kept.




(a) MH_03_medium (b) MH_05_difficult

(c) V1_01_easy (d) V2_01_easy

Fig. 6. Estimated trajectory (black) and ground truth (red) on EuRoC MH_03_medium, MH_05_difficult, V1_01_easy and V2_01_easy.

## VI. EXPERIMENTS

We evaluate our system on public dataset EuRoC [32]. Accuracy of our system is evaluated by using the benchmark tool [34]. The used setup is an Intel Xeon(R) E5-1620(eight cores @ 3.50GHz) and 16 GB RAM. Since our method is semi-direct and uses MVG, we compare our method with these two state-of-the-art methods : a semi-direct method SVO [10],[11] and a key-feature-based method ORBSLAM [13],[14]. In terms of accuracy and speed, the results of comparison prove our method is fast and accurate.

On EuRoC dataset, there are eleven sequences provided in total. There were collected on-board a Micro Aerial Vehicle (MAV) flying around two different rooms and a large industrial environment. The stereo camera has a baseline whose length is about 11cm and captures images at 20 Hz. The sequences are classified as easy, medium and difficult depending on MAV's speed, illumination and scene texture. The dataset contains millimeter accurate position ground truth from a laser tracking system. We test our system on

TABLE I
FRAMES LOCALIZATION ERROR COMPARISON BY THE BENCHMARK TOOL [34] ON EUROC DATASET, WHERE THE ERROR IS ABSOLUTE TRAJECTORY ERROR (RMSE: CENTIMETER)

| Sequence | Length /Duration | OURS | SVO stereo | ORBSLAM stereo without loop |
|---|---|---|---|---|
| MH_01_easy | 80.6m/182s | **3.80** | 8.00 | 4.03 |
| MH_02_easy | 73.5m/150s | **3.76** | 8.00 | 4.16 |
| MH_03_medium | 130m/132s | 5.36 | 29.00 | **4.78** |
| MH_04_difficult | 91.7m/99s | **9.20** | 267.00 | 40.49 |
| MH_05_difficult | 97.6m/111s | **9.30** | 43.00 | 10.27 |
| V1_01_easy | 58.6m/144s | 8.72 | **5.00** | 8.85 |
| V1_02_medium | 75.9m/83.5s | 20.11 | **9.00** | 9.75 |
| V1_03_difficult | 79.0m/105s | 53.28 | 36.00 | **16.44** |
| V2_01_easy | 36.5m/112s | 8.85 | 9.00 | **6.21** |
| V2_02_medium | 83.2m/115s | **7.67** | 52.00 | 7.96 |
| V2_03_difficult | 86.1m/115s | X | X | X |

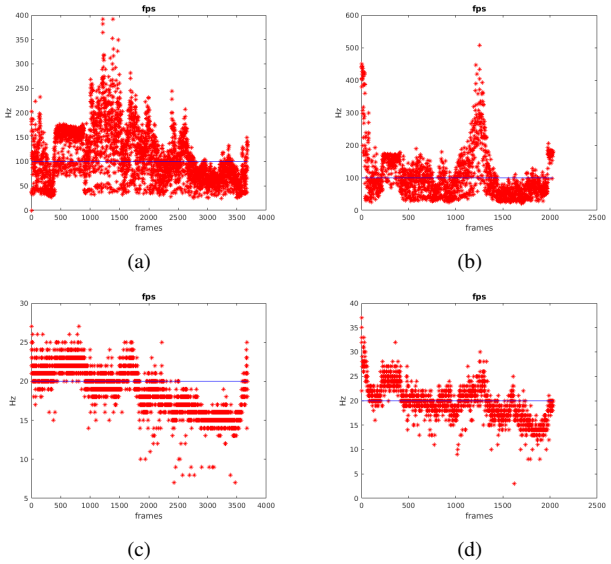



Fig. 7. (a) and (b) show frame rate of our system on EuRoC MH_01_easy and MH_04_difficult. (c) and (d) show frame rate of stereo ORBSLAM on EuRoC MH_01_easy and MH_04_difficult.

all sequences. The visualization of the implementation of our system around a large industrial environment on MH_01_easy dataset is shown in Fig. 5. Fig. 6 shows examples of computed trajectories compared with the ground truth. We use the absolute trajectory error (ATE) as the evaluation to compare the accuracy of our method with SVO and ORBSLAM. And the result is shown in TABLE I.

### A. Accuracy

We compare the absolute trajectory errors of our system with those of SVO and ORBSLAM. The result is shown in TABLE I. The absolute trajectory errors of SVO are from [11]. Stereo ORBSLAM without closure loop is performed in our computer to obtain its absolute trajectory errors. There are seven sequences where the accuracies of our system are better than those of stereo SVO. There are two reasons. First, we perform triangulation to create new mappoints instead of the depth filter. The depth filter may not converge with large and rapid motions. Second, in our system, we keep a global map while in SVO there is a local map with 10 keyframes. On the other hand, from the table, there are three sequences where the accuracies of SVO are better than those of our system. The main reason is that motion priors (IMU) is used to predict the initial pose in SVO [11] but ours is without any priors. Compared with stereo ORBSLAM [14] without closure loop in accuracy, our system has six sequences with higher accuracies.

### B. Runtime Evaluation

As shown in [10],[11], SVO can run at more than 300 frames per second on a consumer computer on EuRoC. We perform stereo ORBSLAM system on our computer to test the speed. The results show that stereo ORBSLAM can run at about 20 frames per second averagely. However, our system can run at more than 100 frames per second on the same consumer computer. Fig. 7 shows the speeds of ours and ORBSLAM. In SVO, number of the tracked points on each frame is limited within 100. Different from SVO, our system makes the optimization at the back-end. We extract ORB features and perform stereo matching only in the keyframes while each frame is extracted ORB features and performed stereo matching in stereo ORBSLAM. Our system considers both accuracy and speed at the same time. It combines advantages of SVO which is faster and of ORBSLAM which is more accurate. Therefore, our system is practical which can meet well some vision tasks in robotics, augmented reality, virtual reality, and so on.

## VII. CONCLUSIONS AND FUTURE WORK

In this study, we proposed a novel framework for stereo visual SLAM, which fuses key-feature-based MVG and direct formulation by making full use of advantages from SVO [10],[11] and ORBSLAM [13],[14]. At the front-end, we perform semi-direct method to track the camera pose, which makes our system faster. At the back-end, MVG is used to estimate 3D structure. We perform triangulation to create new points and use stereo constraint to carry out the optimization, which makes our system more accurate. In our novel framework, accuracy and speed are taken into account at the same time, which makes our system more practical. Because of the efficiency: faster speed, higher accuracy, and lower consumption, our system is being implemented on robot platforms for visual navigation.

Despite the great results, some promotions are needed in the future. We will use feature lines and edges to improve the robustness of the system. We will add closure loop detection to deal with large city environments. And using IMU sensor is also a good choice to assist vision localization. Furthermore, the framework will be adapted to a monocular visual SLAM.

## REFERENCES


[1] D. Nistr, O. Naroditsky, and J. Bergen, "Visual odometry," in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004, pp. I-I.

[2] S. Thrun and J. J. Leonard, "Simultaneous localization and mapping," in Spring handbook of robotics. Spring, 2008, pp. 871-889.

[3] J. Engle, J. Sturm, and D. Cremers, "Semi-dense visual odometry for a monocular camera," in Proceedings of the IEEE International Conference on Computer Vision, 2013, pp. 1449-1456.

[4] C. Mei, G. Sibley, M. Cummins, P. Newman, and I. Reid, "Rslam: a system for large-scale mapping in constant-time using stere," International Journal of Computer Vision, vol. 94, no. 2, pp. 198-214, 2011.

[5] C. Kerl, J. Stueckler, and D. Cremers, "Dense continuous-time tracking and mapping with rolling shutter RGB-D cameras," in Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 2264-2272.

[6] P. Corke, J. Lobo, and J. Dias, "An introduction to inertial and visual sensing," The International Journal of Robotics Research, vol. 26, no. 6, pp. 519-535, 2007.

[7] Y. Wu, F. Tang, H. Li, "Image based camera localization: an overview," Accepted by Visual Computing for Industry, Biomedicine and Art, 2018.

[8] G. Younes, D. Asmar, E. Shammas, J. Zelek, "Keyframe-based monocular SLAM: design, survey, and future directions," Robotics and Autonomous Systems, vol. 98, pp. 67-88, 2017.

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

[10] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: fast semi-direct monocular visual odometry," in Robotics and Automation(ICRA), 2014 IEEE International Conference on. IEEE, 2014, pp. 15-22.

[11] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, "SVO: semi-direct visual odometry for monocular and multi-camera systems," IEEE Transactions on Robotics, vol. 33, no. 2, pp. 249-265, 2017.

[12] M. Pizzoli, C. Forster, and D. Scaramuzza, "REMODE: probabilistic, monocular dense reconstruction in real time," in Proceedings of the IEEE International Conference on Robotics and Automation, 2014.

[13] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: a versatile and accurate monocular slam system," IEEE Transactions on Robotics, vol. 31, no. 5, pp. 1147-1163, 2015.

[14] R. Mur-Artal and J. D. Tardos, "ORB-SLAM2: an open-source SLAM system for monocular, stereo, and RGB-D cameras," IEEE Transactions on Robotics, vol. 33, no. 5, pp. 1255-1262, 2017.

[15] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in European Conference on Computer Vision. Spring, 2006, pp. 430-433.

[16] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: an efficient alternative to sift or surf," in Proceedings of the IEEE International Conference on Computer Vision, 2011, pp. 2564-2571.

[17] J. Engle, V. Koltun, and D. Cremers, "Direct sparse odometry," IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017.

[18] 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, vol. 29, no. 7, pp. 1052-1067, 2007.

[19] 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. IEEE, 2007, pp. 225-234.

[20] H. Strasdat, J. M. Montiel, and A. J. Davison, "Visual SLAM: why filter," Image and Vision Computing, vol. 30, no. 2, pp. 75-77, 2012.

[21] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd, "Real-time localization and 3D reconstruction," In Proceedings of the IEEE Conference of Vision and Pattern Recognition, 2006, pp. 363-370.

[22] T. Pire, T. Fischer, J. Civera, P. De Cristóforis, and J. J. Berlles, "Stereo parallel tracking and mapping for robot localization," in Intelligent Robots and System(IROS), 2015 IEEE/RSJ International Conference on. IEEE, 2015, pp. 1373-1378.

[23] H. Strasdat, J. Montiel, and A. J. Davison, "Scale drift-aware large scale monocular SLAM," Robotics: Science and Systems, 2010.

[24] H. Strasdat, A. J. Davison, J. M. Montiel, and K. Konolige, "Double window optimisation for constant time visual SLAM," In Proceedings of the IEEE International Conference on Computer Vision, 2011, pp. 2352-2359.

[25] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: an accurate o(n) solution to the pnp problem," International Journal of Computer Vision, vol. 81, no. 2, pp. 155-166, 2009.

[26] J. Civera, A. J. Davison, and J. M. Montiel, "Inverse depth parametrization for monocular slam," IEEE Transactions on Robotics, vol. 24, no. 5, pp. 932-945, 2008.

[27] M. Irani an P. Anandan, "All about direct methods," In Proceedings of Workshop Vis. Algorithms: Theory Pract. 1999, pp. 267-277.

[28] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: dense tracking and mapping in real-time," in Computer Vision(ICCV), 2011 IEEE International Conference on. IEEE, 2011, pp. 15-22.

[29] J. Engle, T Schöps, and D. Cremers, "LSD-SLAM: large-scale direct monocular slam," in European Conference on Computer Vision. Spring, 2014, pp. 834-849.

[30] S. Baker and I. Matthews, "Lucas-Kanade 20 Years on: A Unifying Framework: Part 1," International Journal of Computer Vision, vol. 56, no. 3, pp. 221-255, 2002.

[31] Y. ma, S. Soatto, J. Kosecka, and S. S. Sastry, An Invitation to 3D Vision: From Images to Geometric Models. Spring Verlag, 2005.

[32] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," The International Journal of Robotics Research, vol. 35, no. 10, pp. 1157-1163, 2016.

[33] S. Agarwal and K. Mierle, "Ceres solver, http://ceres-solver.org," Google, 2016.

[34] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems, 2012.