2018 24th International Conference on Pattern Recognition (ICPR)
Beijing, China, August 20-24, 2018

Em-SLAM: a Fast and Robust Monocular SLAM Method for Embedded Systems

Yirui Wu[†][‡]*, Zhikai Li[‡]*, Shivakumara Palaiahnakote[§], and Tong Lu[‡]
[†]College of Computer and Information, Hohai University, Nanjing, China
[‡]National Key Lab for Novel Software Technology, Nanjing University, Nanjing, China
[§]Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur, Malaysia
{wuyirui@hhu.edu.cn; lizk@smail.nju.edu.cn; shiva@um.edu.my; lutong@nju.edu.cn}

***Abstract*—Simultaneous Localization and Mapping (SLAM) is difficult to deploy in the embedded systems due to its high computation cost and stable input requirements. Building on excellent algorithms of recent years, we present Em-SLAM, a monocular SLAM method which is fast and robust in the embedded system. We present Em-SLAM in three stages comprising *initial pose estimation*, *iterative pose optimization and correspondences*, and *mapping with nearest frame queue*. During the first stage, we perform stable initial pose estimation based on the matched ORB features extracted around the selected key points. Regarding initial pose and corresponding key points as input, the second stage of Em-SLAM iteratively optimizes these inputs values by tracking key points in the new frames. At the last stage, we firstly determine keyframes with the help of the proposed nearest frame queue and then design a greedy search algorithm to find matched ORB features between keyframes, which are adopted for compact and robust map reconstruction. Due to the special designs for the embedded systems, Em-SLAM demonstrates a high accurate and fast performance on the embedded system for all SLAM tasks: tracking, mapping and loop closing. We evaluate Em-SLAM on he most popular datasets by comparing with one latest SLAM method.**

I. Introduction

Simultaneous Localization and Mapping (SLAM) [1], [2], [3] is one of most active research areas in the fields of pattern recognition and robotics. SLAM performs the task of constructing a map of an unknown environment while simultaneously estimating the pose of an agent via visual clues. Recently, SLAM have attracted lots of attentions from researchers and companies, since they can be deployed on different types of automatic agents, from mobile robots to drones, for navigation [4], map reconstruction [5] and so on.

We classify current SLAM methods into two categories, namely direct methods [6], [7] and feature-based methods [1], [8]. The main difference between two categories lies in estimating poses and constructing map through corresponding pixels or matched features. Without additional feature extraction and matching, direct methods have shown faster performance than feature-based methods. Generally, run-time computation cost of the direct method is linearly related with the number of corresponding pixels, which is often a small value in direct methods. However, the computation of photometric error, *i.e.* the difference of the projected intensity values of one map point in two successive frames, is sensitive, as it involves warping and integrating large image regions [3]. Wrong initial pose estimation could lead the photometric error to reach the local extremum, resulting in unreliable results. On the contrary, the feature-based methods could offer a more reliable pose estimation even with large inter-frame movement, due to the availability of robust feature detectors and descriptors. However, extracting and matching features between frames requires large computation cost, making the feature-based method not suitable for resource limited systems, such as the embedded systems. Moreover, the camera shaking and large inter-frame movement greatly affect the robustness during the deployment of SLAM on embedded systems, since embedded systems with smaller shape may behave in a more unstable way than large robots. Therefore, either current feature-based methods or direct methods can't guarantee the fast and robust performance in the embedded systems.

* indicates equal contribution.

With the idea of utilizing the advantages of feature-based and direct methods to improve the performance in the embedded systems, we propose Em-SLAM, which utilizes matched key points between frames to estimate the initial pose at initialization, and performs fast tracking to compute the robust results of both pose and reconstructed map at runtime. Since key points extracted at initialization are adopted as the initial correspondences for runtime tracking, the runtime computation cost is linearly related with the number of extracted key points, which is determined by several parameters. The proposed method thus supports instant re-deployment on embedded systems with different computation resource by simply adjusting parameters.

The main contribution of the paper is to propose Em-SLAM, which could perform pose estimation and map reconstruction in embedded systems by involving the strength of both feature-based and direct methods. The proposed method could achieve robust and fast performance in run-time, due to stable initial pose estimation, fast iterative pose optimization and reliable map reconstruction. The proposed method also supports instant re-deployment on different embedded systems by simply adjusting parameters. Moreover, we propose a structure, named as nearest frame queue (short for NFQ), which serves to find the matching keyframe with less time and avoids the situation of loss tracking.

II. Related Work

A. SLAM Problem Definition

SLAM is the problem of determine the robot poses s_i and a map of the environment m_i from the feature positions f_j and image feature measurements z_{ij}:

$$p(s_i, m_i | f_j, l_{ij}) \tag{1}$$

where s_i denotes the robots pose at time i consisted by a robots x-y coordinate in the plane and its heading direction, m_i refers to the reconstructed map at time t, f_j represents the positions of the features in the jth frame and l_{ij} represents the image feature measurements - the observation of feature f_j when robot is with pose s_i. Applying Bayes' rule to sequentially updating s_i or m_i, we could rewrite Equ. 1 as

$$p(s_i | f_j, l_{ij}, m_i) \quad and \quad p(m_i | f_j, l_{ij}, s_i) \tag{2}$$

The classical mapping methods can be found in [1], [9], [10] and we mainly focus on the part of pose estimation. As defined in [11], the best way to estimate pose is to construct a probability model with the structure of a probabilistic Markov chain, where s_i and f_j are variables, and $l_{i,j}$ are represented by edges in the chain graph. In SLAM, this network will continuously grow as new pose and measurement variables are added at every time step, and new features will be added whenever new parts of a scene are explored for the first time.

B. Monocular SLAM

Monocular SLAM methods could be classified as either feature-based or direct method.

The standard approach of a feature based method is to firstly extract and match a sparse set of salient image features in nearby frames. After matching, the approach robustly recovers and refines camera motion and map structure through epipolar geometry and reprojection error minimization, respectively. PTAM [8] is the most representative feature-based SLAM algorithm that achieves robustness through camera tracking and mapping quantity of features. PTAM is further improved with edge features, a rotation estimation step during tracking and a better relocalization method [12]. Recently, ORB-SLAM [1], [13], which is currently the state of the art method, utilizes the extraction of sparse ORB features from the input image to carry out four tasks: tracking, mapping, relocalization and loop closing. Note that ORB-SLAM could compute a sparse reconstruction of the scene as well as to estimate the camera pose with the help of employing local bundle adjustment and pose graph optimization. However, directly deploying ORB-SLAM in the embedded system could result in system crash and lag performance, due to its high computation requirement.

Direct methods estimate the map structure and pose directly from intensity key values among frames. The first approaches [14], [15] utilize the filter to jointly estimate the map feature locations and the camera pose. However, it has the drawbacks of wasting computation in processing consecutive frames with little new information and the accumulation of linearization errors. Later methods [16], [17] use nonlinear least squares

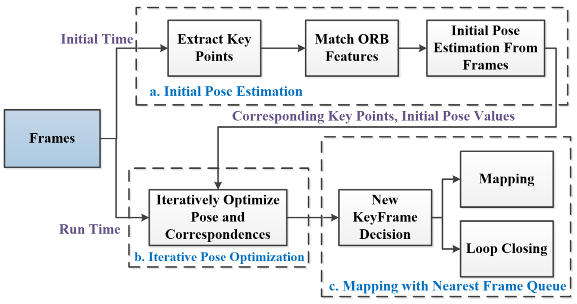
Initial Time
Extract Key Points
Match ORB Features
Initial Pose Estimation From Frames
a. Initial Pose Estimation
Corresponding Key Points, Initial Pose Values
Frames
Iteratively Optimize Pose and Correspondences
New KeyFrame Decision
Mapping
Loop Closing
Run Time
b. Iterative Pose Optimization
c. Mapping with Nearest Frame Queue

Fig. 1. The framework of the proposed Em-SLAM consists of three steps: (a) initial pose estimation, (b) iterative pose optimization and (c) mapping with nearest frame queue.

optimization instead to estimate the surface normals of the patches, which allows tracking a patch over a wide range of viewpoints. Recently, Engel et .al [6] combines a fully direct probabilistic model with consistent, joint optimization of all model parameters, achieving real-time performance even in the embedded systems. However, the photometric error of directed method is large with strong inter-frame movements, results in unstable pose estimation. Most relevant to our methods, Forster et. al [3] propose a semi-direct monocular visual odometry algorithm by coupling direct and feature-based methods, which could obtain impressive results in quadracopters at high frame-rates without requiring to extract features in frames.

III. The Proposed Method

The structure of the proposed Em-SLAM is shown in Fig. 1, which consists of three steps, *i.e.* initial pose estimation, iterative pose optimization and mapping with nearest frame queue. Note that first step and the later two steps are performed at initialization and at runtime, respectively. During the initial time, the proposed method firstly finds key points by either gradients or contrast, and then performs initial pose estimation based on the matched ORB features, which are extracted around key points in nearby frames. During the run time, the proposed method tracks key points in input frames to iteratively optimize the corresponding key points and initial pose values, computed by the initialization step. After tracking, the proposed method decides whether to accept it as a new keyframe with the help of the proposed nearest frame queue. If accepted, the ORB features of the new keyframe will be matched with former keyframes with a greedy search algorithm. All matched ORB features together construct a robust and compact map. Besides, the new keyframe will be used to detect a loop.

A. Initial Pose Estimation From Frames

In this subsection, the proposed method aims to estimate the initial pose based on matching features from nearby frames at initialization. The proposed method firstly finds key points by either gradient or contrast, and then extracts ORB features [18] around the key points. After extraction, the proposed method adopts a constant velocity motion model [19] to

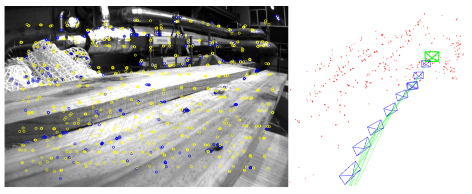
(a) (b)

Fig. 2. (a) An example of the extracted key points, where blue and yellow points represent key points with high contrast and large gradient values; (b) An example to represent the process of mapping, where red points refer to map points, green lines represent the trajectory of the camera and blue triangles represent keyframes.

perform pose initialization for one frame based on the matched ORB features. To stabilize the initial pose estimation during the shaking period of camera, the proposed method searches matches in a wide range of nearby frames and constructs a weight scheme to achieve a robust pose estimation.

Feature-based methods generally can't achieve desirable matching results among little texture or motion blurred frames, since few features could be obtained on such frames. We thus fuse points with either large gradients or high contrast to construct the set of key points K as follows:

$$\begin{aligned} O =&\{p| \sum_{x\in b(p)} |I_s(x) - I_s(p)| > \varepsilon_{p,s}, s = 1, ..., l\} \\ &\bigcup \{q|G_s(q) - \frac{1}{c} \sum_{x\in b(q)} G_s(x) > \varepsilon_{q,s}, s = 1, ..., l\} \end{aligned} \tag{3}$$

where function $b(p)$ and $b(q)$ refer to a square window centered at pixel p and q respectively, I and G refer to the intensity and gradients map of the inputting frame, s represents the scale level of the resized frames computed by image pyramid, l is the level number of the pyramid (We set $l = 8$ by experiments), c is the number of pixels inside the window, and $\varepsilon_{p,s}$, $\varepsilon_{q,s}$ are two region-adaptive thresholds based on the mean value of intensity and gradient inside the window, representing with $\varepsilon_{p,s} = \alpha \cdot \frac{1}{c}\sum_{x\in b(q)} I_s(x)$ and $\varepsilon_{q,s} = \beta \cdot \frac{1}{c}\sum_{x\in b(q)} G_s(x)$. It's noted the parameters α and β determine the number of key points. Recall that the runtime computation cost of the proposed method is linearly related with the number of the extracted key points, users could adjust parameters α and β to modify the run-time computation cost, making it fit with different types of embedded systems. From Equ. 3, we could know p are pixels with high contrast computed following the idea of FAST corner detection [20] and q are pixels with larger gradients by comparing with the mean gradient of the pixels inside the window. It's noted we utilize the intensity property of images and the first order of intensity values to find p and q, respectively. With such derivative operations, the proposed method could find more key points with local maximal property, achieving more robust pose estimation result. We show an example of the extracted key points in Fig. 2 (a), where we could see the proposed method successfully obtains key points even in plain regions.

After finding key points, ORB descriptors are computed on O to represent the neighboring information around the key points. The ORB descriptor is widely used in feature matching due to its robust, fast and view-invariant performance. The proposed method then uses a constant velocity motion model to initially predict the camera pose based on the set of matched ORB features in two successive frames, represented as $M_{i,i+1}$ where i represents the order number of the frames. However, not enough matches lead to violated and unstable motion model, resulting in wrong pose initialization results. We thus propose to perform a wider search around key points in the last frame to guarantee enough matched ORB features between two successive frames.

Pose estimated by two successive frames may not be accurate if the camera shakes, since the adopted motion model is designed to compute the pose based on frames with constant speed. We thus propose to search more matches in nearby frames for robust pose estimation. Note that we achieve nearby frames by the proposed nearest frame queue, which is efficient in obtaining the set of nearby frames due to its basic structure *i.e.* queue. After matching, we construct a weight scheme to compute a convinced pose estimation r_i for the ith frame based on the poses estimated by nearby frames, which could be represented as:

$$r_i = \sum_{j=1}^{n} \omega_{i,j} f_v(M_{i,i+j}) \tag{4}$$

where n is the number of nearby frames (We set $n = 10$ by experiments), function $f_v()$ represents the constant velocity motion model, $M_{i,i+j}$ represents the matched ORB features between the ith and $i+j$th frame, and $\omega_{i,j}$ refers to the weight corresponding to the pose estimated based on the ith and jth frame. Note that we generate ω by one-dimensional gaussian weight function, which assigns larger weight value to the pose estimated by closer frame. Finally, we stop the initialization step until we achieve a stable pose estimation satisfying the following criterion:

$$r_{ini} = \{r_i|\ |r_i - r_{i-1}| < \gamma r_{i-1}\} \tag{5}$$

where γ is a preset and large threshold value.

B. *Iterative Pose Optimization*

In this subsection, the proposed method propose to track key points in input frames to iteratively optimize the corresponding key points and initial pose. During each iteration, we first optimize pose based on the neighboring information of key points and then optimize positions of corresponding key points in new frame through alignment of the corresponding feature-patches. Finally, we minimize the reprojection residuals to compute the robust pose values.

Inspired by [3], we first compute the optimized camera poses $\tilde{r}_i$ by minimizing the photometric error between the input corresponding key points as follows:

$$\tilde{r}_i = \arg\min_{r_i} \frac{1}{2} \sum_{e\in O_{i,i-1}} \parallel \delta(r_i, P_i(e)) \parallel^2 \tag{6}$$

where $O_{i,i-1}$ refers to set of the corresponding key points between the $i-1$th and ith frame, function $\delta()$ calculates the photometric error as defined in [21], $P_i(e)$ denotes small reference pathes of 3×3 pixels around the input key point e in frame I_i. Since Equ. 6 is nonlinear in $\tilde{r}_i$, we solve it in an iterative Gauss-Newton procedure. Note that we extend the calculation regions for robust optimization and the input r_i and $O_{i,i-1}$ could be computed by either initialization or optimization in the last iteration.

Next, we first perform Lucas - Kanade tracking [22] to roughly compute the positions of corresponding key points $O_{i,i-1}$ for the new frame, and then refine the positions by minimizing the photometric error of the patch in the current frame with respect to the reference patch in the keyframe:

$$\tilde{m} = \arg\min_{m} \frac{1}{2} \parallel P_i(m) - A_i \cdot P_r(m) \parallel, \forall m \in O_{i,i-1} \quad (7)$$

where m and $\tilde{m}$ are the positions of key point in set $O_{i,i-1}$ before and after refinement respectively, I_r refers to the keyframe which observes point with the closest observation angle and A_i is an affine warping matrix, which will be applied to the reference patch $P_r(m)$ due to its large size. Equ. 7 is solved using the inverse compositional Lucas - Kanade.

Finally, we again optimize the camera pose r_i to minimize the reprojection residuals as follows:

$$\tilde{r}_i = \arg\min_{r_i} \frac{1}{2} \sum_{e \in O_{i,i-1}} \parallel e - \pi(r_i, p_e) \parallel^2 \quad (8)$$

where p_e refers to a 3D point corresponding to the key point e and $\pi(r_i, p_e)$ is the projection function, which projects p_e to frame plain by the estimated pose r_i. It's noted that function $\pi()$ is determined by the intrinsic camera parameters which are known from calibration. Equ. 8 is essentially the well-known problem of motion-only Bundle Adjustment [11], which is solved by using an iterative Gauss-Newton least-squares minimization algorithm in the proposed method.

C. Mapping With Nearest Frame Queue

In this subsection, the proposed method utilizes the nearest frame queue to help seek a proper frame as keyframe. Once a keyframe is determined, we propose a greedy search algorithm to find matched ORB features between keyframes for map updating. Meanwhile, we adopt the new keyframe to check whether a loop is detected.

To insert the current frame into the keyframe database, it must meet the conditions about the number of key points and similarity *i.e.* current frame must track at least 40 key points and track less than 85% key points than any keyframe in the keyframe database. It's noted that we design simple conditions to speed up the process of inserting keyframes. Due to the settled low thresholds, the inserting could be robust to challenging camera movements and typically rotations as well.

Direct applying such conditions requires the implements of local culling, which is used by [1] and [10], to maintain a small enough keyframe database so that a compact enough map reconstruction. However, local keyframe culling is time-consuming and may delete already reconstructed map points when culling the corresponding keyframe. We thus propose NFQ to help seek a proper frame as keyframe without such culling scheme. The key idea of utilizing NFQ for searching lies in the fact that nearby frames are similar in features so that culling is possible to occur in nearby frames. Pre-saving and searching strategy relieve the repeated procedures of culling in neighbourhood so that reduce the computing time. Moreover, when camera shakes, the current frame may have a higher degree of matching with the previous n-th frame (when the relative movement is smaller). We thus utilize NFQ to find a keyframe with proper matching degree without entering the tracking loss situation. Specifically, we firstly put the current frame into NFQ represented by N, whose size is settled before. When NFQ is full, we search it to determine the set of new keyframes K_n by measuring and selecting with:

$$\begin{aligned} K_n =& \{i | v_i < 90\%, where\ v_i = max f_\delta(O_i, O_j), \\ & i \in N, \forall j \in N \cup K_{ref}, i \neq j\} \end{aligned} \quad (9)$$

where v_i represents the similarity when comparing the ith frame with other frames in the keyframe database and NFQ, K_{ref} and N are the sets of frames in the keyframe database and NFQ respectively, O_i refers to the set of key points corresponding to the ith frame, function $f_\delta()$ counts the similar ORB features extracted on two sets of key points.

Once there are an optimized camera pose and several keyframes, an initial local map is reconstructed using the covisibility graph of keyframes following the semi-dense 3D mapping method proposed in [1]. We represent such process in Fig. 2(b). After inserting a new frame into keyframe database, the proposed method processes the new keyframe and performs local BA [23] to achieve an optimal reconstruction in the surroundings of the camera pose. New correspondences for unmatched ORB in the new keyframe are searched in connected keyframes in the covisibility graph to triangulate new points. However, due to the existence of large inter-frame movement, the feature descriptions of the projected pixels of the same 3D map point may change greatly after long-distance move, which makes the accurate matching between two long-distance ORB features and further the robust and compact enough mapping hard to realize. We thus introduce a greedy search algorithm, represented in Algorithm 1, to find matched key points in long distance. In Algorithm 1, function $size()$ get the size of the set, σ and τ refers to the index of keyframe

Algorithm 1 Greedy search for matched key points in long distance

```
Input: New ORB feature $z_p$, set of ORB features $R = \{R_i | i \in K_{ref}\}$
Step1: Set $S = z_p$ and $\sigma = size(K_{ref})$
Step2: Search in ORB set of keyframe $R_\sigma$, compute $\varphi_\sigma = \max f_{sim}(S, z_\tau), \forall z_\tau \in R_\sigma$ and the corresponding $z_{\sigma,\tau}$
Step3: If $\varphi_\sigma <= \varphi_{\sigma-1}, ..., \varphi_{\sigma-x}$
$\sigma = \sigma - 1$ (if $\sigma = 0$, stop) and Go to Step2
Else $\sigma = \sigma - 1$ (if $\sigma = 0$, stop), $S = z_{\sigma,\tau}$, and Go to Step2
Output: New correspondence for mapping $\{z_p, z_{\sigma,\tau}\}$
```

TABLE I
COMPARISON OF TRANSLATION RMSE (CM) AND PER FRAME PROCESSING TIME (MS) ON TUM RGB AND EUROC DATASET WITH THE PROPOSED METHOD, ORB-SLAM2 [13] AND LSD-SLAM [9].

	PC						Embedded System					
Sequence	$R_{p,1}$	$R_{p,2}$	$R_{p,3}$	$T_{p,1}$	$T_{p,2}$	$T_{p,3}$	$R_{e,1}$	$R_{e,2}$	$R_{e,3}$	$T_{e,1}$	$T_{e,2}$	$T_{e,3}$
Fr1/xyz	0.9360	**0.5927**	*fail*	**20.47**	24.54	*fail*	0.9451	**0.6012**	*fail*	**61.36**	89.94	*fail*
Fr2/xyz	0.4252	**0.2401**	3.177	18.82	25.04	**10.38**	0.4278	**0.2396**	3.175	59.21	87.68	**37.37**
Fr2/desk	1.056	**0.9730**	*fail*	**18.29**	23.41	*fail*	1.173	**0.9834**	*fail*	**53.48**	82.95	*fail*
Fr3/office	**1.180**	2.328	13.18	18.56	30.86	**9.651**	**1.219**	2.349	13.16	58.12	109.0	**34.74**
MH01easy	**4.379**	4.399	13.25	19.62	39.91	**15.24**	**4.432**	*fail*	13.31	59.02	*fail*	**54.86**
MH02easy	**3.180**	3.857	11.41	21.26	38.87	**15.23**	**3.276**	*fail*	11.46	66.02	*fail*	**53.34**
MH03medium	5.213	**4.091**	*fail*	**18.16**	35.27	*fail*	**5.250**	*fail*	*fail*	**54.48**	*fail*	*fail*
MH04difficult	*fail*	**6.576**	*fail*	*fail*	**35.35**	*fail*	*fail*	*fail*	*fail*	*fail*	*fail*	*fail*

and ORB feature respectively, function $f_{sim}()$ computes the similarity between two ORB features and x is the number of searching depth (We set $x = 10$ by experiments). The key idea behind Algorithm. 1 is an assumption of transmittal. In other words, we have an assumption, *i.e.* if ORB features z_a, z_b and z_c satisfy $z_a = z_b$ and $z_b = z_c$, then we can conclude $z_a = z_c$. Since directly matching between z_a and z_c is difficult, we could transform it by searching medial ORB feature z_b in a greedy sense. Moreover, once a new keyframe is inserted, we propose to follow the idea of [1] to perform loop detection.

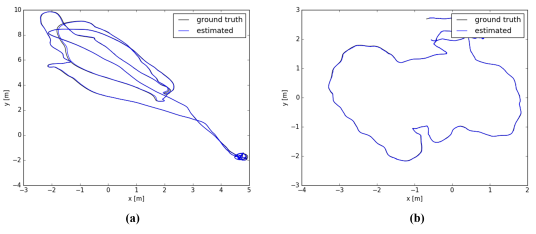

Fig. 3. (a) and (b) represent the Trajectory (blue) estimated by the proposed method and groundtruth (black) for EuRoC MH02easy and TUM RGB Fr3/office.

IV. EXPERIMENTS

A. Dataset and Evaluation

In experiments, we consider two databases, *i.e.* TUM RGB dataset [24] and EuRoC dataset [25], to evaluate the proposed method. Note that TUM RGB dataset is a well-known benchmark dataset to evaluate the general performance of the system including localization accuracy, map reconstruction and processing time, while EuRoc dataset pay special attention on performance with large scale scene and different levels of difficulties. In fact, sequences from EuRoc dataset are classified as easy, medium and difficult depending on cameras speed, illumination and scene texture. Based on these data, we utilize the absolute translation RMSE R, proposed in [24], to evaluate the localization accuracy for one given sequence:

$$R = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \| \, trans(r_i, r_{i,g}) \, \|^2} \tag{10}$$

where n is the number of frames, r_i and $r_{i,g}$ represent the predicted pose values estimated by a SLAM method and ground-truth pose for the ith frame respectively, and function $trans()$ calculates the translational components of the relative pose error between the estimated and ground-truth pose. Note that we follow the idea of [24] to only use translation RMSE for evaluation, since comparing translation RMSE is enough to prove that the proposed method is fast in performance.

B. Performance Analysis

We perform all experiments on an Intel Core i7-4700MQ (4 cores @ 2.60GHz) PC with 8Gb of RAM and an ARM Cortex-A9 (4 cores @ 1.60GHz) Embedded system with 2Gb of RAM. Table. I gives the localization accuracy and the time comparison, where subscripts 1, 2 and 3 correspond to the proposed method, ORB-SLAM2[13] and LSD-SLAM[9] respectively, subscripts p and e refer to performance on PC and the embedded system respectively, R is utilized to measure the localization error, T is the per frame processing time to measure time performance, and the first four rows and other rows correspond to sequences in TUM RGB and EuRoC dataset. Recall that ORB-SLAM is a feature-based method that achieves significant results on tracking, mapping, relocalization and loop closing, while LSD-SLAM is able to build large scale semi-dense maps using direct methods in real time. The codes for two comparative methods are available online so that we build them for experiments.

From Table. I, we find the proposed method are able to process all the sequences, except for MH04difficult. In MH04difficult, the camera shakes so greatly that the proposed directly tracking with key points may lose. Meanwhile, LSD-SLAM fails in four cases. In fact, we find LSD-SLAM is not robust due to its high computation cost of generating semi-dense map ORB-SLAM2 fails with four sequences from EuRoC dataset when implemented in the embedded system. This is due to the reason the computation cost of ORB-SLAM2 increase greatly so that the embedded system couldn't support its running when dealing with difficult sequences. The proposed method introduces the fast iterative optimization on pose and correspondences between key points to relieve the computation burden, which guarantees its flexibility especially in the embedded system. Moreover, the iterative optimization leads to the fast performance in running, which is proved by the fact that $\{T_{p,1}, T_{e,1}\}$ achieved by the proposed method is much smaller than that achieved by ORB-SLAM2 $\{T_{p,2}, T_{e,2}\}$ and a slightly larger than that achieved by LSD-SLAM $\{T_{p,3}, T_{e,3}\}$. In fact, steps of feature extraction and matching increase the computation of the proposed method when comparing with LSD-SLAM. On the other hand, such designs are helpful to purge a more robust and accurate localization result, which is presented by the statics that $\{R_{p,1}, R_{e,1}\}$ obtained by the proposed method is nearly the same as that achieved by ORB-SLAM2 $\{R_{p,2}, R_{e,2}\}$ and much smaller than that achieved by LSD-SLAM $\{R_{p,3}, R_{e,3}\}$. By extracting more types of key points for optimization and design of the nearest keyframe queue, we also notice more robust performance of the proposed method comparing with ORB-SLAM2, which is proved by $R_{p,1}$ obtained by the proposed method slightly increases when dealing with easy, medium and difficult sequences form EuRoC dataset. However, $R_{p,2}$ obtained by ORB-SLAM2 raises greatly when the difficulty level of the input sequence increases. Fig. 3 shows examples of computed trajectories compared to the ground-truth and more computation examples are shown on the attached video.

C. Implementation

The proposed Em-SLAM incorporates three parallelled running threads, *i.e.* tracking, local mapping and loop closing,. Recall the steps illuminated in Fig. 1, step (a), (b) and new key frame decision module in step (c) could be implemented as functions of the tracking thread. Local mapping thread refers to the implementation of the mapping module in step (c), while loop closing thread represents the implementation of the loop closing module in step (c). Considering mapping and looping threads might cause delay of the SLAM system during runtime, we further propose a strategy, *i.e.* stop mapping and looping if we find no more changes in the reconstructed map druing a period of time. Using this strategy will make Em-SLAM suitable for indoor environment with less computation.

V. Conclusion

In this paper, we propose Em-SLAM, a fast and robust monocular SLAM method, specially designed for the embedded systems. We firstly perform stable initial pose estimation based on the matched ORB features. Regarding corresponding key points as input, the proposed method iteratively optimizes inputs values by tracking key points in the new frames. Finally, we determine keyframes in NFQ and perform a greedy search algorithm to find matched ORB features. Comparative experimental results on a popular dataset illustrate the effectiveness of Em-SLAM. Our future work includes the exploration on dense map reconstruction with improved Em-SLAM.

Acknowledgment

This work was supported by the Natural Science Foundation of China under Grant 61702160, Grant 61370091, Grant 61672273, the Fundamental Research Funds for the Central Universities under Grant 2016B14114, the Science Foundation of JiangSu under Grant BK20170892, the Science Foundation for Distinguished Young Scholars of Jiangsu under Grant BK20160021 and the open Project of the National Key Lab for Novel Software Technology in NJU under Grant KFKT2017B05.

References

[1] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Trans. Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[2] K. Tateno, F. Tombari, I. Laina, and N. Navab, "CNN-SLAM: real-time dense monocular SLAM with learned depth prediction," in *Proceedings of CVPR*, 2017, pp. 6565–6574.

[3] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: fast semi-direct monocular visual odometry," in *Proceedings of ICRA*, 2014, pp. 15–22.

[4] M. Milford and G. F. Wyeth, "Seqslam: Visual route-based navigation for sunny summer days and stormy winter nights," in *Proceedings of IEEE ICRA*, 2012, pp. 1643–1649.

[5] 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. 6, pp. 1052–1067, 2007.

[6] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *CoRR*, vol. abs/1607.02565, 2016.

[7] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *CoRR*, vol. abs/1708.03852, 2017.

[8] G. Klein and D. W. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proceedings of International Symposium on Mixed and Augmented Reality, ISMAR*, 2007, pp. 225–234.

[9] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: large-scale direct monocular SLAM," in *Proceedings of ECCV*, 2014, pp. 834–849.

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

[11] H. Strasdat, J. M. M. Montiel, and A. J. Davison, "Visual SLAM: why filter?" *Image Vision Comput.*, vol. 30, no. 2, pp. 65–77, 2012.

[12] G. Klein and D. W. Murray, "Improving the agility of keyframe-based SLAM," in *Proceesdings of ECCV*, 2008, pp. 802–815.

[13] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: an open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Trans. Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[14] N. Molton, A. J. Davison, and I. D. Reid, "Locally planar patch features for real-time structure from motion," in *Proceedings of BMVC*, 2004, pp. 1–10.

[15] H. Jin, P. Favaro, and S. Soatto, "A semi-direct approach to structure from motion," *The Visual Computer*, vol. 19, no. 6, pp. 377–394, 2003.

[16] G. F. Silveira, E. Malis, and P. Rives, "An efficient direct approach to visual SLAM," *IEEE Trans. Robotics*, vol. 24, no. 5, pp. 969–979, 2008.

[17] A. Pretto, E. Menegatti, and E. Pagello, "Omnidirectional dense large-scale mapping and navigation based on meaningful triangulation," in *Proceedings of IEEE ICRA*, 2011, pp. 3289–3296.

[18] E. Rublee, V. Rabaud, K. Konolige, and G. R. Bradski, "ORB: an efficient alternative to SIFT or SURF," in *Proceedings of IEEE ICCV*, 2011, pp. 2564–2571.

[19] C. C. De Wit, H. Olsson, K. J. Astrom, and P. Lischinsky, "A new model for control of systems with friction," *IEEE Transactions on automatic control*, vol. 40, no. 3, pp. 419–425, 1995.

[20] E. Rosten and T. Drummond, "Fusing points and lines for high performance tracking," in *Proceedings of IEEE ICCV*, 2005, pp. 1508–1515.

[21] S. Baker and I. A. Matthews, "Lucas-kanade 20 years on: A unifying framework," *International Journal of Computer Vision*, vol. 56, no. 3, pp. 221–255, 2004.

[22] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of IJCAI*, 1981, pp. 674–679.

[23] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustmentła modern synthesis," in *International workshop on vision algorithms*. Springer, 1999, pp. 298–372.

[24] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 573–580.

[25] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," *I. J. Robotics Res.*, vol. 35, no. 10, pp. 1157–1163, 2016.