FastORB-SLAM: Fast ORB-SLAM method with Coarse-to-Fine Descriptor Independent Keypoint Matching

Qiang Fu[12], Hongshan Yu[1], Xiaolong Wang[23], Zhengeng Yang[1],
Hong Zhang[2] *Fellow, IEEE*, Ajmal Mian[4]
[1]National Engineering Laboratory for Robot Visual Perception and Control Technology, Hunan University, China
[2]Department of Computing Science, University of Alberta, Canada
[3]School of Mathematics and Information Science, Shaanxi Normal University, Xi'an, China
[4]Department of Computer Science & Software Engineering, The University of Western Australia, Australia

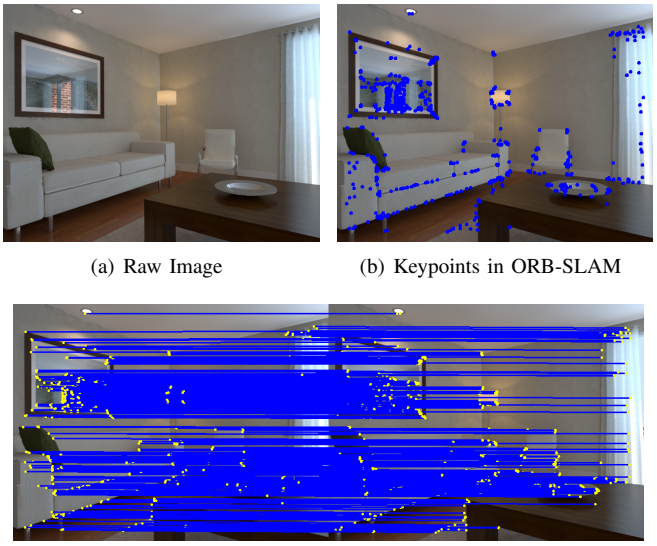
Indirect methods for visual SLAM are gaining popularity due to their robustness to varying environments. ORB-SLAM2 [1] is a benchmark method in this domain, however, the computation of descriptors in ORB-SLAM2 is time-consuming and the descriptors cannot be reused unless a frame is selected as a keyframe. To overcome these problems, we present FastORB-SLAM which is light-weight and efficient as it tracks keypoints between adjacent frames without computing descriptors. To achieve this, a two stage coarse-to-fine descriptor independent keypoint matching method is proposed based on sparse optical flow. In the first stage, we first predict initial keypoint correspondences via a uniform acceleration motion model and then robustly establish the correspondences via a pyramid-based sparse optical flow tracking method. In the second stage, we leverage motion smoothness and the epipolar constraint to refine the correspondences. In particular, our method computes descriptors only for keyframes. We test FastORB-SLAM with an RGBD camera on *TUM* and *ICL-NUIM* datasets and compare its accuracy and efficiency to nine existing RGBD SLAM methods. Qualitative and quantitative results show that our method achieves state-of-the-art performance in accuracy and is about twice as fast as the ORB-SLAM2.

***Index Terms*—Keypoint Matching, Optical Flow, Motion Model, ORB SLAM, Visual SLAM.**

I. Introduction

VISUAL simultaneous localization and mapping (SLAM) has been an active field of research in recent years [1]–[8]. SLAM provides a powerful solution for mobile robots to estimate six degrees-of-freedom (DoF) pose (position and orientation) and recover the 3D structure of the surroundings from a camera's image stream. Visual SLAM is gaining importance in many application areas [9], such as virtual reality (VR), augmented reality (AR), unmanned aerial vehicle (UAV) or unmanned ground vehicle (UGV) navigation, and autonomous mobile robots.

High-accuracy and low-computational cost are the two core requirements of visual SLAM [10]–[17]. Current methods are divided into photometric-based direct methods, e.g., DSO [4] and SVO [5], and feature-based indirect methods [18]–[20]. Direct methods recover pose by minimizing the pixels' photometric errors. On the other hand, indirect methods leverage discriminative image features to recover camera pose by minimizing the reprojection errors between the feature correspondences, and implement loop closure (relocation) to eliminate the global drift based on the feature descriptors. Point-based methods track discriminative keypoints along successive frames and then recover the camera motion trajectory. These methods are robust because the discriminative keypoints are relatively invariant to changes in viewpoint and illumination. In the last several years, many indirect SLAM methods were proposed for real-time applications [8], [18]. Among these, ORB-SLAM2 is considered to be the current state-of-the-art



(a) Raw Image (b) Keypoints in ORB-SLAM

(c) Keypoint Matching without Descriptor (Ours)

Fig. 1. Illustration of our keypoint matching method on two adjacent frames from the *ICL-NUIM* dataset [51]. ORB-SLAM takes ∽16 ms to extract keypoints (∽8ms for detection + ∽8ms for description) under default parameters (1000 numbers), whereas our method takes only ∽12 ms to establish reliable keypoint correspondences without computing descriptors.

This work was supported in part by the National Natural Science Foundation of China under Grant 61973106 and Grant U1813205, U1913202, in part by the China Scholarship Council under Grant 201906130082, in part by the Key Research and Development Project of Science and Technology Plan of Hunan Province under Grant 2018GK2021, in part by the Key Project of Science and Technology Plan of Changsha City under Grant kq1801003, in part by the Hunan Natural Science Foundation under Grant 2017JJ3118, in part by Aviation Science Fund Grant 201705W1001, in part by the Key Research and Development Project of Science and Technology Plan of Chenzhou City.
Corresponding author: Hongshan Yu
This work is constructed at Alberta of University.
Any question, please feel free to contact me: cn.fq@qq.com.
Video Demo: https://youtu.be/bFWTT-kGEQ0.

SLAM method and build on many excellent works, e.g., first real-time visual SLAM, PTAM [21], fast place recognition, BoW2 [22], and efficient graph-based bundle adjustment (BA) *covisibility graph* [23]. Therefore, ORB-SLAM2 yields better accuracy and robustness than other existing solutions.

Mainstream indirect methods such as ORB-SLAM2 implement three threads: *Tracking*, *Local Map* and *Loop Clousure*. The *Tracking* thread establishes keypoint correspondences in adjacent frames based on descriptor matching, and then estimates and outputs camera pose in real time. Once a current frame is selected as a keyframe, the last two threads are activated to refine camera motion but not in real time. The *Tracking* part is considered as the foundation of any SLAM system, as it not only has an immediate impact on accuracy and robustness but also provides association information for the other two threads. Naturally, it takes up most of the computational resources.

We observe that the computation of keypoint descriptors in indirect methods is time-consuming and the descriptors are not reused except in the case of keyframes. This wastes significant computational sources. If we can establish reliable keypoint correspondences without computing descriptors between adjacent frames (or equivalently in *Tracking*), it will greatly reduce the computational cost without loss of precision.

Based on the above, this paper presents FastORB-SLAM, an efficient light-weight visual SLAM system. Unlike indirect methods such as ORB-SLAM2, our method computes descriptors only when the frame is selected as a keyframe.

To establish reliable keypoint correspondences between adjacent frames without descriptors, our keypoint matching method is designed into two stages: The first stage is for robust keypoint matching, we firstly predict initial keypoint correspondences by a uniform acceleration model, and then pyramid-based optical flow tracking algorithm is implemented to establish robust keypoint correspondences. The second stage is for inlier refinement, we firstly leverage motion smoothness constraint to filter out outliers, and then adopt epipolar constraint to refine the correspondences again.

In summary, our main contributions are as follow:

- We present FastORB-SLAM, a novel, complete, light-weight, and robust SLAM system that is developed based on ORB-SLAM2 and sparse optical flow, which can output high-accurate 3D pose estimation, e.g., Fig. 2.
- A novel coarse-to-fine keypoint matching method is proposed, which can establish reliable keypoint correspondences between adjacent frames without descriptors.
- We study a uniform acceleration model to predict keypoint correspondences, which does not only improve the accuracy of the keypoint matching but also potentially reduces the computation of searching correspondences.
- The proposed FastORB-SLAM was tested using an RGB-D camera as input, with almost all representative open-source RGB-D SLAM systems in terms of location accuracy (RMSE) and computation time over a dozen sequences from the well-known *TUM* [50] and *ICL-NUIM* [51] datasets. The qualitative and quantitative results show our method achieves SOTA performance.
- Our method is about twice as fast as the ORB-SLAM2 with highly competitive location accuracy. A demo is provided to demonstrate it[1].

The remainder of the paper is organized as follows: Section II introduces related work, then the FastORB-SLAM system architecture and implementation steps are presented in Section III. Section IV introduces the coarse-to-fine descriptor-independent keypoint matching method in detail. Experiment setup is described in Section V. Finally, we conclude remarks and the highlights of future works in Section VI.

II. Related Work

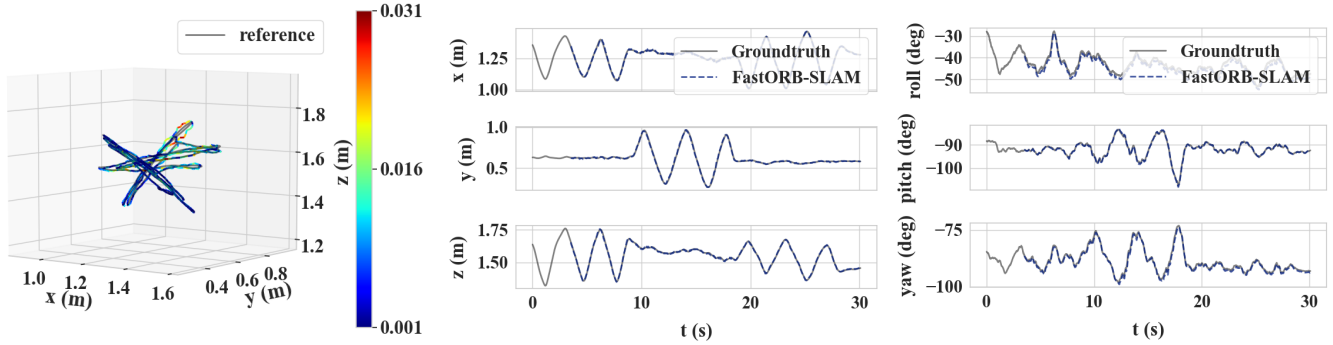
High-accuracy and low-computational cost are the two core requirements of visual SLAM [2], [3]. Camera motion in SLAM is regarded as rigid motion (translation and rotation) [12], it can be obtained by constantly estimating the transformation matrix between consecutive frames, and the matrix also can be used for map registration. Current visual SLAM methods are divided into photometric-based direct SLAM methods and feature-based indirect SLAM methods:

Photometric-based Direct SLAM. This group of methods solves the pose estimation problem by minimizing the images pixel-level intensities errors [5], which is originally inspired by the optical flow algorithm [24]. Recent representative works can be divided into semi-direct methods (SVO) [5] proposed by ETH Zurich and sparse direct methods (DSO) [4] proposed by TUM.

Forster *et al.* firstly proposed SVO, a two-thread framework including *Tracking* and *Local Mapping*, where it tracks sparse pixels at FAST corners to recover motion in *Tracking*, and refines pose in *Local Mapping*. SVO uses a *depth filter* to estimate pixel depth value and filter outliers. The way it works is that it models the triangulated depth observations using a Gaussian+Uniform distribution: if the triangulated depths of the same feature point are close within a small range then the mean and variance of the depth values can be obtained using a Gaussian distribution, whereas if the depth values spread out, then they follow a uniform distribution. If a feature contains a lot of outliers, it will be filtered out, as it does not converge to a Gaussian distribution with a small variance. To solve this problem, Loo *et al.* proposed CNN-SVO method [7], where he uses a mono depth prediction network to predict depth value at corners, greatly improves the robustness.

Engel *et al.* firstly proposed DSO, in which a novel probabilistic model is presented to directly minimize photometric error without computing keypoints or descriptors. Subsequently, as CNN-SVO did, Yang *et al.* leveraged deep depth prediction to improve the performance of DSO [25], [26]. Wang *et al.* proposed Stereo DSO, in which depth value is estimated by multiple view geometry [27]. Schubert *et al.* adopt a rolling shutter model to improve robustness [28], Von *et al.* fused a Inertial Measurement Unit (IMU) to improve robustness in quick movement scenes [29], similar works include [30]–[32]. And, Gao *et al.* added a *Loop Closure* thread to eliminate global drift errors [33]. Lots of experiments show the direct

[1]https://youtu.be/bFWTT-kGEQ0 or bilibili.com/video/BV1wT4y1j7hf



(a) 3-D motion trajectory error (b) translation errors along time (c) rotational errors along time

Fig. 2. The example of location accuracy experiments. In this example, FastORBSLAM runs in representative *TUM-fr1-xyz* sequence, produces highly competitive location accuracy (translational RMSE = 0.010 m). (a) denotes camera 3-D motion trajectory, and error is described via a colormap map on the right. (b) and (c) denote translation and rotational errors along time in x, y, z-axis direction, respectively. More experiments with representative SOTA solutions are presented in Section V-B.

methods have an obvious advantage on computing speed [4], however, it is easy to produce poor robustness and accuracy.

Feature-based Indirect SLAM. This group of methods leverages salient image features (like point or line features) to recover and refine camera motion by minimizing reprojection errors of the features correspondences [18], [20], [21].

Georg *et al.* proposed PTAM, the first real-time feature indirect SLAM method, from the University of Oxford, which adopts two parallel threads to estimate pose for real-time in *Tracking* thread, and refine the camera motion in *Local Mapping* thread. Subsequently, lots of works were proposed based on PTAM for real-time applications [8], [14]. Among them, ORB-SLAM2 is known as the current SOTA apporach as it yields unprecedented performance [2]. In addition to the mentioned two parallel threads, ORB-SLAM2 added an *Loop Closure* thread to search a global constraint. The thread is develop on bag of words (BoWs) model [22] and *covisibility graph* [23], the former is used to measure similarity of two frames, the latter is used for efficient large-scale BA.

Subsequently, many works were proposed based on ORB-SLAM2. Point-based methods presumably produce poor location accuracy, even fail in low-texture scenes where the methods cannot track enough keypoints. Gomez-Ojeda *et al.* [18] and Fu *et al.* [20] proposed to combine line features into ORB-SLAM2 system, to improve robustness in low-texture scenes. To meet the requirement of pose estimation in dynamic scenes, Bescos *et al.* proposed Dyna-SLAM [38], in which it added a preprocessing step to cull dynamic objects via a Mask-RCNN network. Besides, researchers also extended ORB-SLAM2 to some applications, such as robot navigation [39], [40], semantic SLAM [41], [42], etc. Newly, ORB-SLAM3 was released on arXiv in August 2020 [43], it focused on the integration of ORB-SLAM and IMU information.

Indirect methods can be summed up as follow: extract sparse features; match them in successive frames based on descriptor distance; recover camera motion, refine pose and structure through minimizing reprojection errors in feature correspondences. Compared with direct methods, indirect methods takes more computation resources to extract indirect features. It is a double-edged sword that robust feature extractor makes the system more robust, whereas it is time-consuming.

Summary. A complete and robust SLAM system (direct or indirect methods) should include three threads: *Tracking*, *Local Mapping*, and *Loop Closure*. *Tracking* runs in front-end, it output current camera pose for real-time. *Local Mapping* and *Loop Closure* run in back-end for non-real time, they are designed to refine (optimize) camera motion and structure via local or global constraint. *Loop Closure* is an essential thread for improving robustness in life-time operation as it provides a powerful constraint to correct glocal accumulation errors, moreover, it can be used for relocation when system fail to track efficient features at some time [34]–[36].

Whether minimizing photometric errors (direct methods) or reprojection errors (indirect methods), it boils down a non-linear least-squares optimization problem, which can be efficiently addressed via the BA [18]. Once correspondences are established, pose estimation or refinement problem can be solved through the BA optimization. Therefore, it plays an extremely important role for visual SLAM to establish accurate feature correspondences.

III. System Overview

This paper presents FastORB-SLAM, a complete, robust, and light-weight visual SLAM system. Unlike ORB-SLAM2 establishes keypoint correspondences in adjacent frames based on descriptor matching, this system does it via a coarse-to-fine descriptor-independent matching method. The descriptors are computed only when a frame is selected as a keyframe, whereas ORB-SLAM2 computes them for every frame.

Compared with SVO, there are three main differences: First, our method adopts different keypoint detectors; Second, SVO cannot implement a loop closure operation as it did not extract descriptors; The last but not least, SVO tracks keypoints to recover motion by directly minimizing photometric errors, it has a problem that if keypoint correspondences contain a lot of outliers, these outliers will lead to a terrible location accuracy (more discussion in Section II). Correspondingly, our method establishes keypoint correspondences and deals with the outliers problem via an explicit coarse-to-fine descriptor-independent keypoint matching method, and then recovers

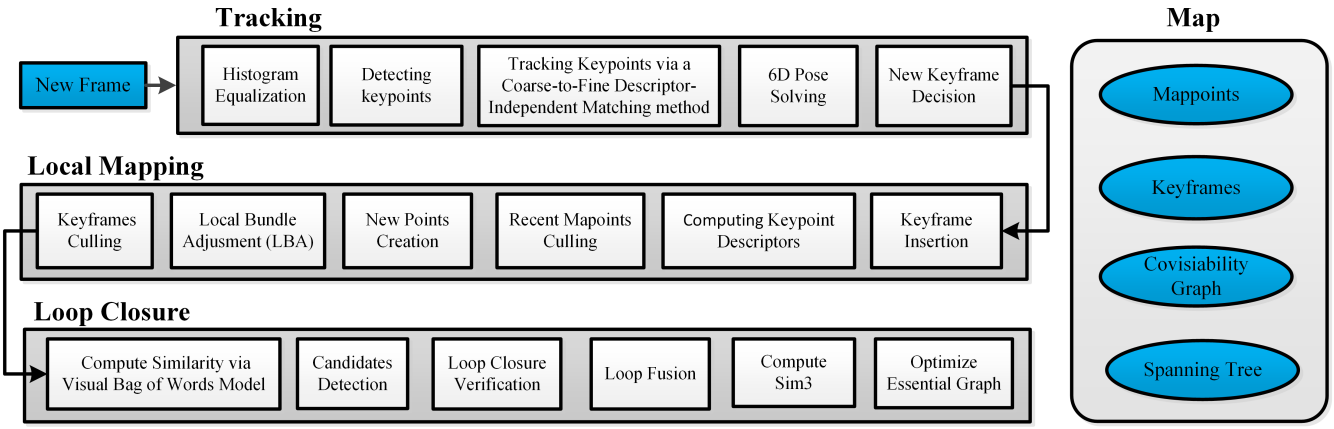


Fig. 3. The proposed FastORB-SLAM system overview. FastORB-SLAM is built based on ORB-SLAM2, consists of three threads: *Tracking*, *Local Mapping*, and *Loop Closure*. *Tracking* quickly estimates and outputs 6D camera pose for real time. *Local Mapping* add a new keyframe and optimizes local keyframes by BA optimization. *Loop Closure* is constantly checking loops and correcting the drift with global BA optimization. The Map structure contains information of keyframe, mappoints, *covisiable graph*, and spanning tree. The compact structure is designed for efficient computation [2], it remains useful observations and culls useless information timely for avoiding redundant computation.

camera motion by minimizing the reprojection errors between the correspondences.

The general structure of FastORB-SLAM is depicted in Fig. 3. In a nutshell, the structure is developed strongly based on the scheme first proposed by ORB-SLAM [13] and also implements three different threads: *Tracking*, *Local Mapping*, and *Loop Closure*. Camera pose estimation and optimization are implemented based on a *Map* structure. *Tracking* runs in front-end, it does not only output real-time camera pose estimation, but also provides observation information between frames for the other two threads. The two threads run in back-end (non-real time), they are activated when a frame is selected as a keyframe, to eliminate local or global drift errors for high-accuracy pose estimation. More details are given below:

Map. A compact map structure is designed for efficient computation when the system optimizes camera pose [2], it remains useful observations and culls useless information timely. The structure consists of:

- Keyframes. Each keyframe contains camera pose parameters, observed keypoints, and descriptors.
- Mappoints. Each mappoint consists of a 3-D landmark that is observed by the corresponding keypoint, and its 3-D positions in the world coordinate system.
- *Covisibility graph*. This graph contains covisibility information of keyframes [23], where each node represents a keyframe, and the edges between keyframes are created only if they share a minimum number of landmarks (this paper sets it as 20). Implementing local BA means to optimize the current keyframe and its neighbor keyframes (nodes) in the graph, which allows for a very fast refinement operation.
- Spanning tree. Spanning tree stands for the minimum connected representation of a graph that includes all the keyframes. Once a spanning tree is established, a corresponding *essential graph* is created. Different from *covisibility graph*, the edge in *essential graph* is created only when two keyframes share over 100 landmarks, so it is more sparse. Spanning tree and *essential graph* were proposed by ORB-SLAM [2] for a fast global BA.

Optimizing graph is equal to optimize keyframes pose (nodes) based observation constraints (edges), controlling graph scale (nodes and edges) is equal to control computation scale.

Tracking. This thread outputs the real-time pose estimation and provides the observation information between frames for the other two thread.

Firstly, we preprocess each image by the adaptive histogram equalization algorithm proposed in [44] (See Fig. 5) to reduce the illumination effect. Secondly, the keypoints are detected by using an improved ORB algorithm proposed in [1]. Thirdly, initial camear pose is predicted via a uniform acceleration motion model. Forth, the keypoint correspondences are established via a coarse-to-fine descriptor-independent matching method, which will be introduced at length in Section IV. Moreover, as lots of SOTA systems did, the keypoint correspondences are searched from the last frame, the nearest keyframe, and local Map to find more correspondences. Finally, once the correspondences are established, the pose estimation is refined by a BA optimization. Compared with ORB-SLAM2, our efficiency in this thread comes from two aspects:

- Not need to compute keypoint descriptors.
- Not need to detect keypoints when the inlier number is enough (such as 200 or 300), see Fig. 4.

After obtaining camera pose of the current frame, the system judges whether the current frame is a keyframe: pass over 20 frames, track least 50 keypoints, or *Local mapping* is idle.

Local Mapping. This thread is activated, when a frame is selected as a keyframe, to compute keypoint descriptors for the keyframe. Next, this thread looks for previous keyframes in the *Covisibility graph* that connect the current keyframe based on descriptor matching, meanwhile, all mappoints seen by those keyframes are also found. Finally, we can create the corresponding graph structure in *Covisibility graph*. Once the graph is created, the current keyframe and connected

Fig. 4. Examples for keypoints (inliers) on continuous frames. Before detecting keypoints, incoming image is preprocessed using an adaptive histogram equalization algorithm for reducing illumination effect. From this figure, our system can track enough keypoints all the time. Notably, we do not need to extract keypoints if inlier number is enough, for example, we only detect keypoints in the first column and the last column. In this sequence (*ICL-NUIM-Office 3*), FastORB-SLAM yields better location accuracy than ORB-SLAM2 with less computation time, please see Table II, Fig. 10 and 11.

Fig. 5. Example for prerpocessing images via adaptive histogram equalization algorigm for reducing illutration effect. Top column represents original images from *ICL-NUIM* dataset and bottom column represents images after equalization. It takes $\backsim$1.5 ms for a frame.

keyframes pose are optimized for eliminating local drift errors. Note that this thread only optimized the keyframes location that are observed by the current frame, which is regarded as a local BA process. After the local BA, redundant keyframes are discarded to control the graph scale.

Loop Closure. After *Local Mapping*, this thread is activated. It is designed to eliminate global drift errors through a powerful loop closure constraint for glocal BA optimization. In this work, we follow the loop closure work of ORB-SLAM2. Firstly, we adopt the DBoW2 model [22] to search and measure the similarity between keyframes. Secondly, when a loop close constraint is established, a spanning tree that contains all nodes (keyframes) is generated, it stands for the minimum connected representation of a graph, in which for each node, only one parent node and one child node connect it. Thirdly, *essential graph* is created according to the spanning tree. Finally, the global BA is implemented for optimizing the *essential graph*.

In addition, ORB-SLAM2 loads a text-format dictionary [22] for loop detection ($\backsim$3000 ms). This work converts the dictionary to binary format, thus our system can quickly load the dictionary on startup ($\backsim$30 ms).

Camera Motion Model. Camera motion is regarded as 6D rigid body transformation including position and orientation in this paper. We describe it based on Lie Group and Lie Algebra [5]:

$$SE(3) = \left\{ \boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{\rho} \\ \boldsymbol{\phi} \end{bmatrix} \in \mathbb{R}^6, \boldsymbol{\rho} \in \mathbb{R}^3, \boldsymbol{\phi} \in SO(3) \right\}. \quad (1)$$

where $\boldsymbol{\rho}$ denotes a 3×1 translation vector, $\boldsymbol{\phi} \in SO(3)$ denotes rotation matrix. Let $\boldsymbol{T_{cw}}$ and $\boldsymbol{T_{rw}}$ be current frame pose and reference (previous) camera pose in world coordinate system. Let T_r^c be relative motion transformation between the two frames, and $\boldsymbol{T_{cw}}, \boldsymbol{T_{rw}}, \boldsymbol{T_c^r} \in SE(3)$. As we know, $SE(3)$ is a finite dimensional smooth manifold such that the multiplication $SE(3) \times SE(3) \rightarrow SE(3)$. Thus we have:

$$\boldsymbol{T_{cw}} = \boldsymbol{T_c^r T_{rw}} \quad , \quad \boldsymbol{T_c^r} = \boldsymbol{T_{cw} T_{rw}}^{-1}. \quad (2)$$

In particular, camera motion is coded using Sophus library.

IV. Coarse-to-Fine Descriptor-Independent Keypoint Matching Method

Observe that two adjacent frames in time-varying sequence have two characteristics: small baseline distance and brightness invariant, based on the observation, a two-stage, coarse-to-fine, and descriptor-independent keypoint matching method is presented to establish reliable keypoint corrspondences in this section. Notably, the descriptors are extracted only when the frame is selected as a keyframe, see also system overview in Fig. 3. The coarse-to-fine method is divided into two stages:

First Stage is for robust keypoint matching:

- First, predict keypoint correspondences by an efficient motion model, which gives algorithm a good initial guess and also potentially reduces the computation of searching the correspondences;

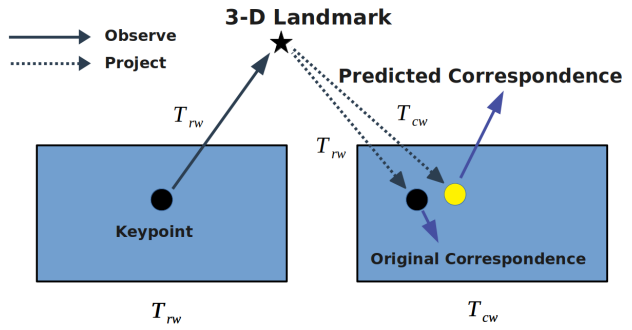


Fig. 6. The illustration of predicting a keypoint correspondence by projecting the 3-D landmark to current frame. Given the pose $\boldsymbol{T_{rw}}$ of reference frame, the predicted pose $\boldsymbol{T_{cw}}$ of the current frame, a keypoint on the reference frame and its 3-D landmark, the predicted (initial) correspondence on the current frame is obtained according to the projection model. Note that in specific example of this paper (see Section IV-B), $\boldsymbol{T_{cw}} = \boldsymbol{T^{\star}_{cw}}$, $\boldsymbol{T_{rw}} = \boldsymbol{T_{cw-1}}$.

Algorithm 1 Descriptor-Independent Keypoint Matching with Motion Prediction Model.

Input: The reference frame I_r, A keypoint on the reference frame $I_r(x, y)$; The current frame I_c; Last three frame pose $\boldsymbol{T_{cw-1}}$, $\boldsymbol{T_{cw-2}}$, $\boldsymbol{T_{cw-3}}$;

Output: The Movement Vector $\boldsymbol{m}(dx, dy)$, and keypoint correspondence $I_r(x, y) \rightarrow I_c(x + dx, y + dy)$;

1: Model camera motion as a uniform acceleration motion model, then predict current frame velocity $\boldsymbol{V_c}$ by Equation (16) from $\boldsymbol{T_{cw-1}}$, $\boldsymbol{T_{cw-2}}$, $\boldsymbol{T_{cw-3}}$.
2: According to $\boldsymbol{V_c}$, predict current frame pose $\boldsymbol{T^{\star}_{cw}}$ by Equation (17), and cast $\boldsymbol{T^{\star}_{cw}} \in SE(3) \rightarrow \boldsymbol{T^{\star}} \in \mathbb{R}^{3\times4}$;
3: According to $\boldsymbol{T^{\star}}$, predict initial keypoint correspondence $I_c(x^{\star}, y^{\star})$ by Equation (18);
4: Solve movement vector $\boldsymbol{m}(dx, dy)$ via **Algorithm 2**;
5: **return** $\boldsymbol{m}(dx, dy)$ and keypoint correspondence $I_r(x, y) \rightarrow I_c(x + dx, y + dy)$;

- Then, establish keypoint correspondences in an 8-level pyramid structure based on sparse optical flow algorithm.

More specifically, we implement **Algorithm 1** for all keypoints in the first stage to robustly establish coarse keypoint correspondences.

Second Stage is for inlier refinement:

- First, leverage camera motion smoothness constraint to filter out outliers;
- Then, adopt RANSAC-based fundamental matrix method refine keypoint correspodences again.

In the rest of this section, we firstly models (formulates) the descriptor-independent keypoint matching problem. Subsequently, special operation steps are described at length.

A. Problem Model

In this work, the goal of the descriptor-independent keypoint matching method is defined as:

Goal: given a keypoint (x, y) in reference frame I_r, find its corresponding location $(x + dx, y + dy)$ in current frame I_c, or equivalently, find the movement vector $\boldsymbol{m} = (dx, dy)$.

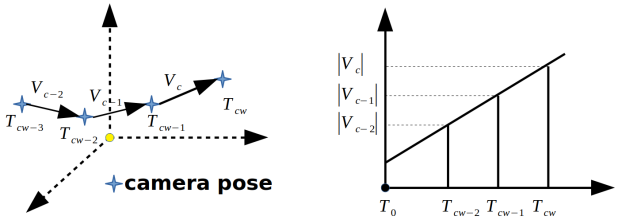


Fig. 7. The illustration of uniform acceleration motion model. This model is used to predict velocity. $\boldsymbol{T_{cw}}$ denotes the camera pose in the world coordinate system, $\boldsymbol{V}$ denotes velocity, $|\boldsymbol{V}|$ denotes scalar value of velocity. For intuitive understanding, we plot velocity scalar change along camera pose on the right, and then we have $|\boldsymbol{V_c}| = 2|\boldsymbol{V_{c-1}}| - |\boldsymbol{V_{c-2}}|$.

Thus the correspondence in I_r and I_c can be established by:

$$I_r(x, y) \leftrightarrow I_c(x + dx, y + dy). \tag{3}$$

Theoretical Foundation. The matching method works on two assumptions:

- Gray Level Invariant: Pixel intensities invariant between adjacent frames.
- Neighborhood Similarity: Consistent motion in neighboring of a point.

Let $I_{x,y,t}$ be a grayscale value of keypoint coordinate (x, y) at t on the first frame, after dt time, the keypoint moves to $(x + dx, y + dy)$ on the next frame.

Assumption 1: Gray level invariant. This assumption means the intensities of two corresponding keypoints between adjacent frames from time-varying sequences do not change. We have:

$$I(x + dx, y + dy, t + dt) = I(x, y, t), \tag{4}$$

by using Taylor expansion, the left hand side becomes:

$$\begin{aligned} I(x + dx, y + dy, t + dt) \approx & I(x, y, t) + \\ & \frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt, \end{aligned} \tag{5}$$

therefore an equation can be obtained:

$$\frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt = 0, \tag{6}$$

by dividing dt, we have:

$$\frac{\partial I}{\partial x}\frac{\mathrm{d}x}{\mathrm{d}t} + \frac{\partial I}{\partial y}\frac{\mathrm{d}y}{\mathrm{d}t} = -\frac{\partial I}{\partial t}, \tag{7}$$

where dx/dt and dy/dt denote the speed of x-axis and y-axis respectively, and $\partial I/\partial x$ and $\partial I/\partial y$ respectively denotes gradient of x-direction and y-direction at the point. $\partial I/\partial t$ denotes gradient along time. Then we have:

$$\begin{bmatrix} \boldsymbol{I_x} & \boldsymbol{I_y} \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{bmatrix} = -\boldsymbol{I_t}, \tag{8}$$

where $\boldsymbol{u} = dx/dt$, $\boldsymbol{v} = dy/dt$, $\boldsymbol{I_x} = \partial I/\partial x$, $\boldsymbol{I_y} = \partial I/\partial y$, $\boldsymbol{I_t} = \partial I/\partial t$. $\boldsymbol{I_x}$, $\boldsymbol{I_y}$ and $\boldsymbol{I_t}$ are image gradients in x, y and $time$ axes, which are all known. Finally, the problem of keypoint matching is translated into solve keypoint movement over the time. However, this equation cannot be solved with two unknown variables equation $(\boldsymbol{u}, \boldsymbol{v})$. Therefore, We make the second assumption.

Assumption 2: Neighborhood Similarity. This assumption means that all pixels in an $\omega * \omega$ size of patch around the keypoint have consistent movement $(\boldsymbol{u}, \boldsymbol{v})$. We have:

$$\begin{bmatrix} \boldsymbol{I_x} & \boldsymbol{I_y} \end{bmatrix}_k \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{bmatrix} = -(\boldsymbol{I_t})_k, k = 1, ..., \omega^2. \tag{9}$$

Equation 9 is over-determined, thus the problem of keypoints tracking becomes to solve two unknown variables with k equations. In which, $(\boldsymbol{u}, \boldsymbol{v})$ can be estimated via least square fit method.

Note that considering t is a fixed scalar between the two adjacent frames, for example, Kinect 1 captures images at 30hz, which means $t = 1/30$ s, in this work, we believe:

$$(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{m}(dx, dy) \tag{10}$$

B. Correspondences Predication with Motion Model

Recall the **Goal** in last Section (IV-A), if given a good initial guess to solve the movement vector $\boldsymbol{m}$ (see Fig. 6), it is not only able to improve robustness of keypoints tracking, but also potentially reduces the iterative optimization computation. Base on it, we predict keypoint correspondences on current frame by a motion model.

First, we cast the camera motion to a uniform acceleration motion model (like Fig. 7 shows), and then estimate the velocity between reference (last) frame and current frame. Next, the current frame pose can be predicted via the velocity. Finally, initial keypoint correspondences are obtained by projecting the 3-D landmarks that are observed by the keypoints in reference frame, to current frame using the predicted current camera pose (see like Fig. 6). The specific operations are given below:

Notation: Let $\boldsymbol{T_{cw}}, \boldsymbol{T_{cw-1}} \in SE(3)$ be current frame pose and last camera pose in world coordinate system. $\boldsymbol{T_{cw}^{cw-1}}$ denotes the relative motion transformation matrix (translation and rotation) between the two frames. As t is a fixed scalar between adjacent frame, for example, 30hz means $t = 1/30$, in this work we define velocity $\boldsymbol{V_c}$ as:

$$\boldsymbol{V_c} = \boldsymbol{T_{cw}^{cw-1}} = \boldsymbol{T_{cw}}\boldsymbol{T_{cw-1}}^{-1} \in SE(3). \tag{11}$$

This equation can obtained from Equation (1). Now, the problem of velocity prediction is transformed to predict relative transformation matrix between current frame and reference frame.

Motion Model. Like Fig. 7 shows, we assume camera motion conform a uniform acceleration motion model, thus velocity $\boldsymbol{V_c}$ from last frame to current frame can be solved from previous three frame poses:

$$\boldsymbol{V_c} = f(\boldsymbol{T_{cw-1}}, \boldsymbol{T_{cw-2}}, \boldsymbol{T_{cw-3}}), \tag{12}$$

where $\boldsymbol{T_{wc-1}}$, $\boldsymbol{T_{cw-2}}$, and $\boldsymbol{T_{cw-3}}$ represent the pose of the corresponding three frames.

Or equivalently, the increment between two adjacent velocities should be equal:

$$\boldsymbol{V_{c-1}} \otimes \boldsymbol{V_c} = \boldsymbol{V_{c-1}} \otimes \boldsymbol{V_{c-2}}, \tag{13}$$

where $\otimes$ represents the increment operation of velocity between two velocities. The velocity is $\in SE(3)$, which is a finite dimensional smooth manifold such that the multiplication $SE(3) \times SE(3) \to SE(3)$. Thus, the increment operation can be computed by:

$$\begin{aligned} \boldsymbol{V_{c-1}} \otimes \boldsymbol{V_c} &= \boldsymbol{V_c}\boldsymbol{V_{c-1}}^{-1} \\ \boldsymbol{V_{c-2}} \otimes \boldsymbol{V_{c-1}} &= \boldsymbol{V_{c-1}}\boldsymbol{V_{c-2}}^{-1} \end{aligned} \tag{14}$$

where:

$$\begin{aligned} \boldsymbol{V_{c-1}} &= \boldsymbol{T_{cw-1}}\boldsymbol{T_{cw-2}}^{-1}, \\ \boldsymbol{V_{c-2}} &= \boldsymbol{T_{cw-2}}\boldsymbol{T_{cw-3}}^{-1}. \end{aligned} \tag{15}$$

Finally, combine above the equations, we have:

$$\begin{aligned} \boldsymbol{V_c} &= \boldsymbol{V_{c-1}}\boldsymbol{V_{c-2}}^{-1}\boldsymbol{V_{c-1}} \\ &= \boldsymbol{T_{cw-1}}\boldsymbol{T_{cw-2}}^{-1}\boldsymbol{T_{cw-3}}\boldsymbol{T_{cw-2}}^{-1}\boldsymbol{T_{cw-1}}\boldsymbol{T_{cw-2}}^{-1}. \end{aligned} \tag{16}$$

Correspondences Prediction. Once the transformation matrix (velocity) is estimated, a predicted (initial) current frame pose $\boldsymbol{T_{cw}^{\star}}$ can be predicted by:

$$\boldsymbol{T_{cw}^{\star}} = \boldsymbol{V_c}\boldsymbol{T_{cw-1}}. \tag{17}$$

For intuitive understanding, in this subsection we uses a 3×4 matrix to denote the transformation instead of Lie Group and Lie Algebra (See Equation 2), that is $\boldsymbol{T_{cw}^{\star}} \in SE(3) \to \boldsymbol{T^{\star}} = \begin{bmatrix} \boldsymbol{R}, \boldsymbol{t} \end{bmatrix} \in \mathbb{R}^{3\times4}$, where $\boldsymbol{R}$ is a 3×3 rotaion matrix, $\boldsymbol{t}$ is a 3×1 translation matrix.

Like Fig. 6 shows. Given a keypoint $I_r(x, y)$ in reference frame, a 3-D landmark is observed in world coordinate system. Thus, by projecting the 3-D landmark (X, Y, Z) to current frame with $\boldsymbol{T}$, an initial guess of keypoint correspondence $I_c(x^{\star}, y^{\star})$ on the current frame can be obtained:

$$\boldsymbol{p^{\star}} \propto \boldsymbol{K}\boldsymbol{T^{\star}}\boldsymbol{P} = s\boldsymbol{K}\boldsymbol{T^{\star}}\boldsymbol{P}, \tag{18}$$

where

$$\boldsymbol{p^{\star}} = \begin{bmatrix} x^{\star} \\ y^{\star} \\ 1 \end{bmatrix}, \boldsymbol{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \boldsymbol{P} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{19}$$

where s represents the scale factor, $\boldsymbol{K}$ denotes Camera Intrinsic Matrix that consists of focal length f_x, f_y and principal point offset c_x, c_y. This matrix is determined in advance by camera calibration. For all keypoitns, the initial (predicted) correspondences can be obtained by Equation (18).

Thus, we can obtain a initial keypoint correspondence $I_c(x^{\star}, y^{\star})$, then Equation (3) (**Goal**) is translated into:

$$I_r(x, y) \leftrightarrow I_c(x + dx, y + dy) \leftrightarrow I_c(x^{\star} + dx, y^{\star} + dy). \tag{20}$$

C. Movement Vector Solving based on an 8-Level Image Pyramid

Movement Vector Solving: First, the grayscale residual function $\epsilon(\boldsymbol{m})$ is defined as:

$$\begin{aligned} \epsilon(\boldsymbol{m}) &= \epsilon(dx, dy) \\ &= \sum_{x=-\omega_x}^{\omega_x} \sum_{y=-\omega_y}^{\omega_y} (I_r(x, y) - I_c(x^{\star} + dx, y^{\star} + dy)), \end{aligned} \tag{21}$$

where ω_x, ω_y set the size of integration window to $(2\omega_x + 1) \times (2\omega_y + 1)$, this paper sets $\omega_x = \omega_y = 2$. For such a patch size, the problem becomes that using 25 equations (points) to solve two variant (dx, dy).

Then, the object function is modeled as:

$$dx, dy = \arg\min_{dx,dy} \sum_{i}^{k} ||I_r(x_i, y_i) - I_c(x_i^\star + dx, y_i^\star + dy)||^2 , \tag{22}$$

where k denotes the pixels in integration window, in this work, $i = 25$, $i \in 1, 2, ..., k$. This equation can be solved through an iterative Lucas-Kanade (KLT) method [24].

Second, one essential observation in Lucas-Kanade method is that image derivatives I_x and I_y can be computed directly from the reference image I_r in the neighborhood of the point independently from the second image I_c, based on it, the gradient expression is defined as:

$$\frac{\partial \epsilon(\boldsymbol{m})}{\partial \boldsymbol{m}} = \sum_{i=1}^{k} \begin{bmatrix} I_x(x,y)I_x(x,y) & I_x(x,y)I_y(x,y) \\ I_x(x,y)I_y(x,y) & I_y(x,y)I_y(x,y) \end{bmatrix} \tag{23}$$

where

$$\begin{aligned} I_x(x,y) &= \frac{\partial I_r(x_i, y_i)}{\partial x} = \frac{I_r(x_i + 1, y_i) - I_r(x_i - 1, y_i)}{2} \\ I_y(x,y) &= \frac{\partial I_r(x_i, y_i)}{\partial y} = \frac{I_r(x_i, y_i + 1) - I_r(x_i, y_i - 1)}{2} \end{aligned} \tag{24}$$

where $I_x(x, y)$ and $I_y(x, y)$ denote the image derivatives of the position (x_i, y_i) in x and y axes, respectively.

Finally, we define a accuracy evaluation function $\epsilon(w)$ to determine when the iterative will terminate. We have:

$$\epsilon(w) = \frac{\sum_{i=1}^{k} ||I_r(x_i, y_i) - I_c(x_i + dx, y_i + dy)||}{k}. \tag{25}$$

This equation denotes average grayscale residual between windows (patches). Let N_{iter} be iterations, thus the termination condition is designed as:

$$\epsilon(w) < w_{errormin} \| N_{iter} > N_{itermax} \tag{26}$$

where $w_{errormin}$ denotes the minimum value of the window error and $N_{itermax}$ denotes the maximum value of iterations. Experimentally, it is a good option to take $w_{errormin} = 0.02$, $N_{itermax} = 10$.

Pyramid Model: The movement vector is computed in a image pyramid structure to improve robustness.

First, given ORB algorithm detects keypoints in an 8-level image pyramid with scale ratio = 1.2, we used the same image pyramid structure.

Let the pyramid levels be $L = 1, ..., L_m$, where $L_m = 8$ is the deepest pyramid level, $\boldsymbol{m^L}$ be the keypoint movement vector on the I^L-th image pyramid, observe that $\boldsymbol{m} = \boldsymbol{m^1}$. The steps is described in **Algorithm 2**.

D. Inlier Refinement

From the previous steps, we can establish robust keypoint correspondences between the reference and current frame, however, there are possibly wrong matching pairs. Therefore, in this stage, we adopt two efficient tips to refine inliers:

1) implement motion smoothness constraint that proposed in [45], [46] to filter out outliers;
2) implement epipolar constraint to refine inliers again.

Motion smoothness constrain takes ∽0.15 ms for ∽1000 keypoints, and epipolar constraint is implement via RANSAC-based fundamental matrix method, which takes ∽0.25ms for ∽1000 keypoints.

Algorithm 2 Solve Movement Vector Based on an 8-Level Image Pyramid.

Input: The reference frame I_r and current frame I_c; The keypoint on reference frame $I_r(x, y)$; The initial (predicted) keypoint correspondence $I_c(x^\star, y^\star)$;

Output: The Movement Vector $\boldsymbol{m}(dx, dy)$ to establish Equation (20);

1: Describe I_c and I_r in an 8-level Pyramid: $L_1, ..., L_m$, scale ratio = 1.2, observer that $L_m = 8$ denote the deepest layer;
2: Compute movement vector $\boldsymbol{m}^{L_m}$ at L_m via an iterative Lucas-Kanade method, for the iterative optimization process:
- Object function: Equation (22);
- Gradient: Equation (23);
- Termination condition: Equation (26).

3: Propagate the result $\boldsymbol{m^{L_m}}$ to upper layer $L_m - 1$ as an initial guess for keypoints movement $\boldsymbol{m^{L_m - 1}}$;
4: Refine movement vector $\boldsymbol{m^{L_m - 1}}$ at level $L_m - 1$ by Equation (22);
5: Propagate the result $\boldsymbol{m^{L_m - 1}}$ to level $L_m - 2$ and so on up to the 1-th level, then get $\boldsymbol{m^1}$;
6: **return** $\boldsymbol{m}(dx, dy) = \boldsymbol{m^1}$;

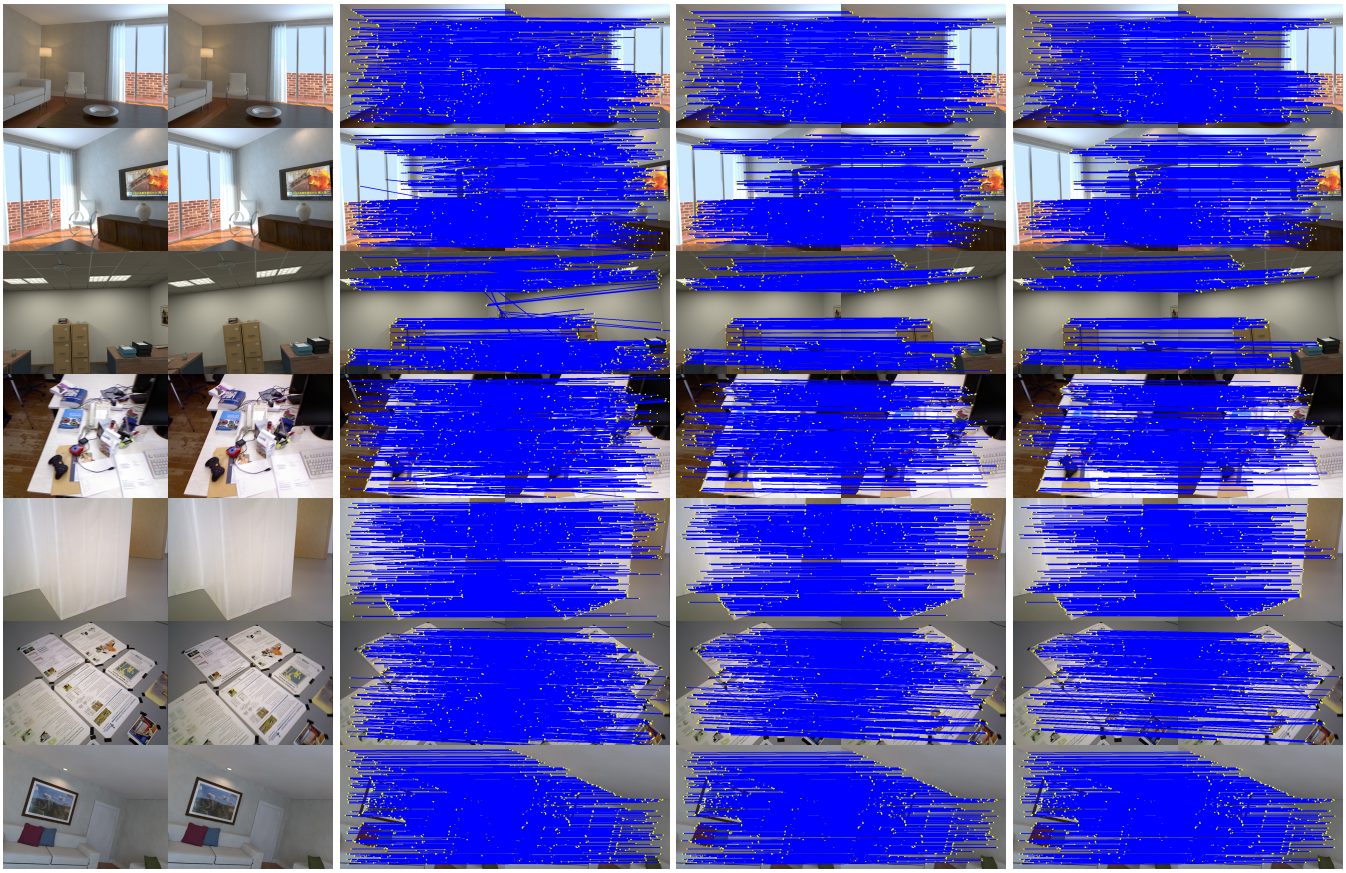
V. Experiments

In this section, we evaluate the performance of FastORB-SLAM using an RGB-D camera in terms of location accuracy and efficiency. We first test the proposed keypoint matching method to demonstrate that our method can establish reliable keypoint correspondences in Section V-A), which builds a foundation for high-accuracy pose estimation . Next, we compare the proposed system with with almost all (9) open-source RGB-D SLAM systems to demonstrate the performance of FastORB-SLAM in Section V-B. In the experiments, qualitative and quantitative comparisons will be presented, related setups are given below:

Dataset: Two well-known RGB-D public datasets: *TUM* dataset [50] and *ICL-NUIM* dateset [51].

Hardware: All experiments were performed on a laptop computer (Intel Core i7-10710U CPU @1.10 GHz without GPU parallelization).

Software: FastORB-SLAM was implemented using C++ on Ubuntu 18.04 LTS (Operate System), and key codes depend on OpenCV 3.4, Sophus, Eigen, and g2o library [52].



(a) Images Pairs (b) KLT (w/o Motion Model) (c) Motion Model (Ours) (d) Inlier Refinement (Ours)

Fig. 8. Examples of ours method in keypoints tracking. We detect 1000 keypoints for every images. In (a), the left and right images represent reference frame and current frame, respectively. (b) and (c) represent the results produced by KTL (w/o motion model) and ours (w/motion model), respectively. (d) represents the results after inlier refinement including motion smoothness and epipolar constraint (Ours).

TABLE I
KEYPOINTS TRACKING COMPARISON OF RATIO [%] AND TIME [MS]. ROW MEANS ROW NUMBER IN FIG 8

	Row 1		Row 2		Row 3		Row 4		Row 5		Row 6		Row 7		Average	
	Ratio	Time	Ratio	Time	Ratio	Time	Ratio	Time	Ratio	Time	Ratio	Time	Ratio	Time	Ratio	Time
KLT (w/o motion model)	0.86	0.44	0.80	0.71	0.62	4.74	0.77	1.26	0.82	0.66	0.70	2.53	0.99	0.21	**0.79**	**1.50**
Ours (w/ motion model)	0.90	0.26	0.88	0.25	0.83	0.27	0.84	0.32	0.84	0.34	0.83	0.40	0.99	0.21	**0.87**	**0.29**

Ratio represents inliers ratio, the inliers number is keypoints number on current frame after epipolar constraint verification. Time represents the time of the epipolar verification. Two algorithms both spent 6-7 ms on keypoints matching, we did not count it in this Table. Row n represents n-th row in Fig 8. From this table, our method yields a higher accuracy with less computation time.

A. Keypoint Matching

In this section, we evaluate the performance of the coarse-to-fine descriptor-independent keypoint matching method in terms of inlier ratio and time consumption. Our keypoint matching method includes two stages: robust keypoint matching and inlier refinement. Beacuse the former stage is developed based on sparse optical flow method, specifically KTL method [24], we adopt it as the baseline method to compare. Their biggest difference is that we study a model model to predict keypoint correspondences as an initial guess (See Section IV), therefore, we test its effect in the following experiments. By the way, KLT and our method are implemented in the same image pyramid: $L_m = 8$, scale ratio = 1.2, because the keypoints (ORB) are extracted in such a pyramid model. Besides, identical iteration termination condition is adopted: $w_{errormin} = 0.02$, $N_{itermax} = 10$.

Qualitative keypoint matching results are presented in Fig. 8, in which all images are selected from ICL-NUIM (Row, 1-3 and 7) and TUM (Row 4-6) datasets. In Fig. 8(a), the left and right represent reference frame and current frame, respectively. Fig. 8(b) and Fig. 8(c) represent the corresponding results produced by KTL method and ours method, respectively. Fig. 8(d) represent the result after inlier refinement (ours). Notably, 1-6 rows represent relatively long baseline distance between two images, row 7 represent relatively small baseline distance. The threshold of keypoint number is set to 1000. From the

TABLE II
CAMERA LOCALIZATION RMSE [M] ERROR AND AVERAGE TIME [S] COMPARISON IN *ICL-NUIM* DATASET

	Office 0		Office 1		Office 2		Office 3		Living 0		Living 1		Living 2		Living 3	
	RMSE	Time	RMSE	Time	RMSE	Time	RMSE	Time	RMSE	Time	RMSE	Time	RMSE	Time	RMSE	Time
ORB-SLAM2	0.038	0.021	**0.070**	0.024	**0.011**	0.020	0.066	0.021	**0.006**	0.021	0.101	0.024	**0.015**	0.021	0.013	0.022
Ours	**0.034**	**0.013**	0.080	**0.013**	0.015	**0.014**	**0.037**	**0.012**	0.010	**0.014**	**0.026**	**0.015**	0.016	**0.013**	**0.009**	**0.012**

All statics are collected via real reproduction test. Median over 5 executions for each sequence. RMSE represents translation RMSE [m]. Time represents average time consumption [s] each frame. See **Office 3** and **Living 1**, our method yields much higher location accuracy than ORB-SLAM2 with less computation cost. In the other sequences, ours method is highly competitive, too.

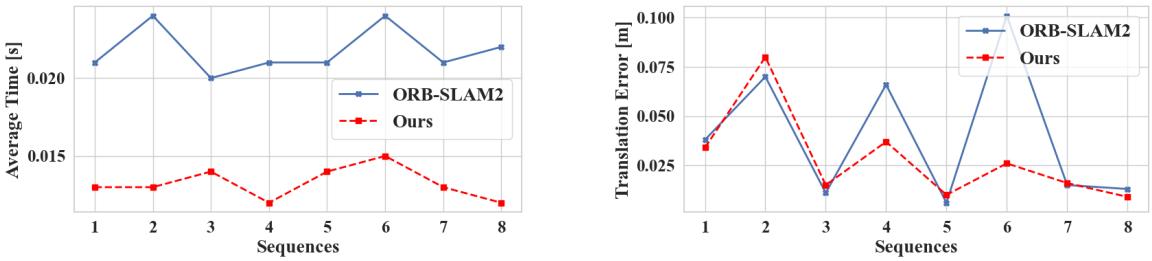
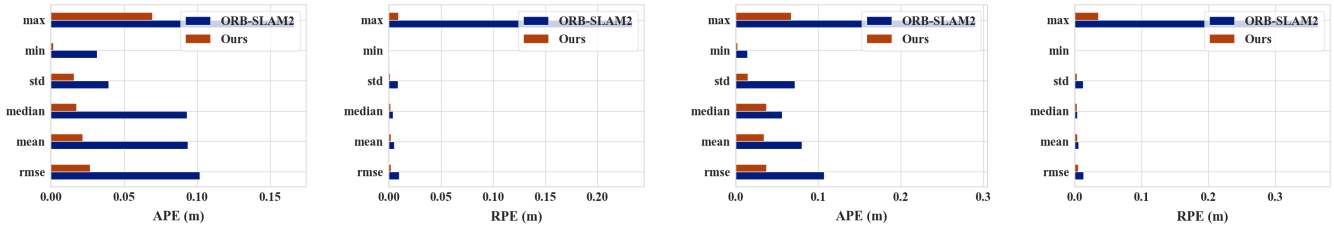


Fig. 9. Average Time and Translation Error Comparison in all 8 sequences from *ICL-NUIM* dateset. Note that lower is better. Sequences 1-8 represent Office 0-3 and Living 0-3 sequences from *ICL-NUIM* dataset in turn. More details are presented in Table II. From this figure, our method produces a highly competitive location accuracy with much lower time consumption in these sequences. In Sequence 4 (Office 3) and Sequence 6 (Living 1), ORB-SLAM2 has a big drift error. In actual run, we observe that it does not track effective keypoints in the two sequences, we further present the error statistic in Fig. 10 and 3-motion trajectory comparison in Fig 11.



(a) *ICL-NUIM-Living 1* (b) *ICL-NUIM-Office 3*

Fig. 10. Two examples for specific APE and RPE comparison. From this figure, our method produce a much better accuracy than ORB-SLAM2 in the two sequences.

figure, our method produce higher accuracy than the KLT method visually. Next, we quantify the results.

Quantitative results are collected in Table I. The "Ratio" represents inliers ratio, following [45], [46], it is computed by N_{inlier}/N_{Total}, where $N_{Total} = 1000$ and N_{inlier} represents the keypoints number after epipolar constraint verification. The "Time" represents the time of the epipolar constraint. In view of the same magnitude time consumption (6-7 ms) of the two algorithms for 1000 keypoints, we did not count it in this Table. Row n represents the n-th row of the Fig 8. From this table, we can conclude as follow:

- Lower inliers ratio produces more time computation when uses epipolar constraint (RANSAC-based foundation matrix) to filter out outliers, e.g., see Row 1, KLT (inlier ratio = 0.80) takes 0.71 ms, in the other hand, our method (0.88) only takes 0.25 ms.
- Good guess can improve the accuracy of the keypoint matching. Our method (with motion model) produces higher average inlier ratio = 0.87, whereas KLT = 0.79.
- Our method yields higher accuracy than the KTL method with less time consumption.

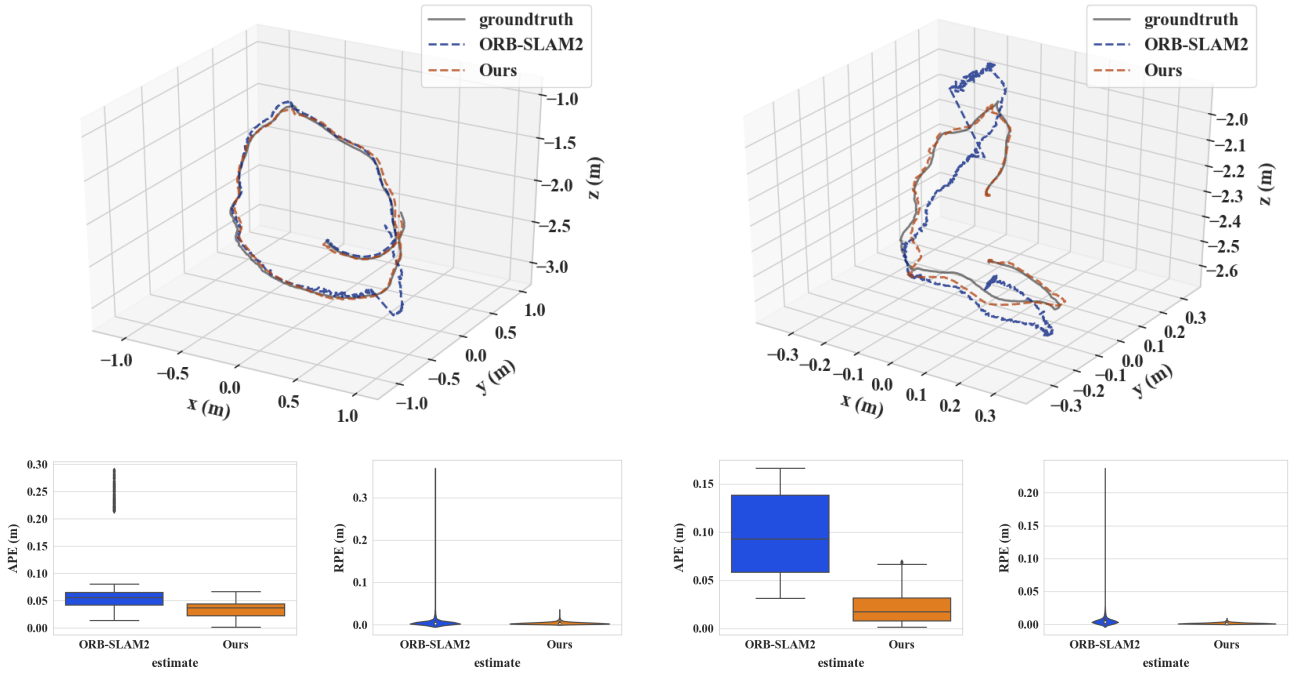
Overall, these experiments show that our method can establish reliable keypoint correspondences without descriptors, which builds a foundation for high-accuracy pose estimation. As we known, better keypoint correspondences, higher location accuracy. Next, we evaluate the system performance.

B. Location Accuracy and Efficiency Experiments

In this section, we evaluate the performance of FastORB-SLAM in terms of location accuracy and time comsuption.

ICL-NUIM dataset. Given our system is developed based on the benchmark ORB-SLAM2, we first compare our system with it. Related experiments are conducted in all 8 sequences from the *ICL-NUIM* dataset in terms of location accuracy and time consumption.

Quantitative and qualitative results are presented in Table II and Fig. 9, respectively. All statics are collected in real reproduction test, RMSE represents translation root mean square



(a) Comparison in *ICL-NUIM-Office 3* (b) Comparison in *ICL-NUIM-Living 1*

Fig. 11. The location accuracy comparison of ORB-SLAM2 and Ours in *ICL-NUIM-Office 3* (Left) and *ICL-NUIM-Living 1* (Right) sequences. Top row represents 3-D motion trajectory. APE and RPE error comparsion are presented in bottom row. From this figure, our methods yields a much better accuracy than ORB-SLAM2 in the two sequences including a low-texture episode.

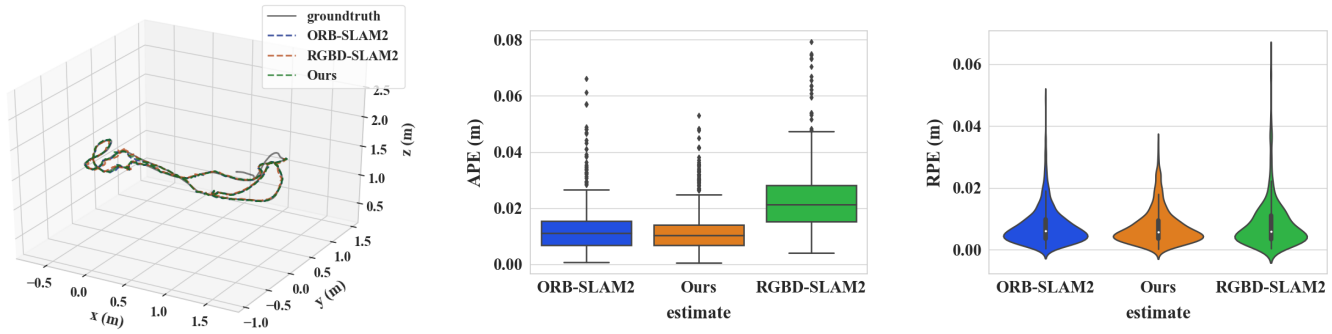
TABLE III
CAMERA LOCALIZATION RMSE (M) ERROR COMPARISON IN *TUM* DATASET

Sequence	Ours	ORB-SLAM2	ElasticFusion	Kintinuous	DVO-SLAM	RGBD-SLAM2	BundleFusion	BAD-SLAM	Lei *et al.*	Fu *et al.*
fr1-xyz	**0.010**	0.012	0.016	0.018	0.023	0.012	0.012	–	–	0.011
fr1-desk	**0.014**	0.015	0.020	0.037	0.021	0.026	0.016	0.017	0.021	0.020
fr1-desk2	0.025	**0.022**	0.048	0.071	0.046	0.025	–	–	–	0.009
fr1-room	0.050	0.047	0.068	0.075	**0.043**	0.087	–	–	–	–
fr2-desk	0.009	**0.008**	0.071	0.034	0.017	0.057	–	–	–	0.009
fr2-xyz	**0.006**	**0.006**	0.011	0.029	0.018	0.026	0.011	0.011	0.013	0.007
fr2-large	0.181	0.140	–	–	–	X	–	–	–	**0.102**
fr3-office	0.011	**0.008**	0.017	0.030	0.035	–	0.022	0.017	0.027	0.018
fr3-nst	**0.018**	0.019	**0.018**	0.031	0.018	X	X	–	0.018	0.021

All statics are collected in ORB-SLAM2 [1], RGBD-SLAM2 [14], BundleFusion [15], BAD-SLAM [16], Lei et al. [17], Fu et. al [20], and reproduction test. "X" means this system failed or lost its position at some point of this sequence. "–" represent that we cannot obtain the value in the papers. From the table, our method achieved SOTA performance, taking *fr1-xyz* as an example, and we further present an intuitive result in Fig. 2.

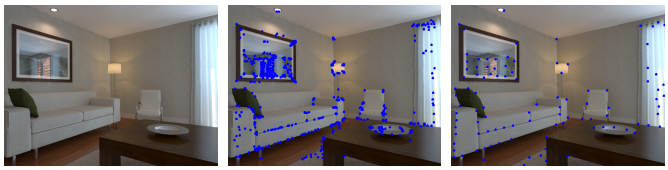
error (RMSE) [m]. Time represents average time consumption [m] in each frame. The specific value is determined over 5 executions for each sequence. Based on these results, we can conclude as follow:

- ORB-SLAM2 and our method both show high robustness, as they both successfully run in all 8 sequences .
- In terms of time computation, our method has an obvious advantage over ORB-SLAM2, see the left Fig. 9 or Time statics in Table II, e.g., in Living 3 sequence: 0.22 (ORB-SLAM2) VS 0.12 (Ours). The frame per second (FPS) of our method is up to 84 HZ nearly. The main reason is that we do not extract keypoint descriptors in *Tracking*.
- In the **Living 1** and **Office 3** sequences, our method obtains much higher location accuracy than ORB-SLAM2. For comparison, we plot the trajectory, APE, and RPE in Fig. 11 and 10. Observe that ORB-SLAM2 does not track effective keypoints in the low-texture part of the two sequences (see video demo), so it produces a big drift error. Correspondingly, ours method can track reliable keypoints all the time due to two reasions: The coarse-to-fine keypoint matching method can establish enough keypoint correspondences; Every image is preprocessed by the adaptive histogram equalization algorithm proposed in [44].
- Besides, our method also produces comparable accuracy in the other sequences.
- In general, our method runs nearly twice faster than ORB-SLAM2 with highly competitive accuracy in the



(a) Motion Trajectory Comparison (b) Absolute Pose Error (c) Relative Pose Error

Fig. 12. The location accuracy comparison of ORB-SLAM2, RGBD-SLAM2 and Ours in *TUM-fr1-desk* sequence. (a) represents 3-D motion trajectory, (b) and (c) respectively represent APE and RPE. From this figure, ORB-SLAM2 and ours yield better accuracy in this sequence.



(a) ORB (b) ORB-SLAM2 (c) Future work

Fig. 13. The keypoints extration compariosn of original ORB algorithm, ORB-SLAM2, and future work.

8 sequences.

TUM dataset. Given lots of RGB-D systems have used the *TUM* as the test dataset, we compare FastORB-SLAM with these representative RGB-D SLAM systems: ORB-SLAM2, DVO-SLAM [13], RGBD-SLAM2 [14], BundleFusion [15], BAD-SLAM [16], Lei *et al.* [17], Fu *et al.* [20], ElasticFusion [48], and Kintinuous [49].

Quantitative results are presented in Table III. In which, location accuracy is evaluated through the translation RMSE. All statics are from [1], [14]–[17], [20], and real reproduction experiments. "X" means this system failed or lost its position in this sequence. "–" represent that we cannot obtain corresponding value in the paper. From Table III, We can conclude:

- ORB-SLAM2 and our method produce better location accuracy than other solutions, as they all both obtained the best accuracy in 4 sequences while other solutions yield one best performance at most, such as DVO-SLAM, ElasticFusion, and Fu *et al.*.
- Take fr1-desk as an example, we further plot 3D motion trajectory of ORB-SLAM2, RGBD-SLAM2, and our method in Fig. 12(a), where ORB-SLAM2, RGBD-SLAM2, and our method all yield acceptable motion trajectory. Further, the comparisons of absolute pose error (APE) and relative pose error (RPE) are presented in Fig. 12(b) and Fig. 12(c), respectively. In which, our method yields the best accuracy in terms of APE and RPE.

Discussion: In the experiments, we test FastORB-SLAM system with 9 RGB-D systems in 17 sequences (8 from *ICL-NUIM*, 9 from *TUM*). Generally speaking, the quantitative results presented in Table III show that our method and ORB-SLAM2 are better than other 8 systems due to the productive system structure design (See Fig. 3). Although ORB-SLAM2 has a bigger drift error in the Living 1 and Office 3 sequences of *ICL-NUIM* than ours, but it maintains a high standard in other all 15 sequences. We cannot declare that our method is definitely better than ORB-SLAM2, but at least we can say that our method is highly competitive in terms of accuracy and robustness. And most importantly our method has an obvious advantage over time computation (See Fig. 9(a)).

In addition, This paper also demonstrates that the brightness invariant between adjacent frames is a reasonable assumption in time-varying sequences, which can be used to speed up the process of keypoint matching between (small baseline distance) adjacent frames without extracting descriptors.

VI. Conclusion

In this paper, we present FastORB-SLAM, a novel, light-weight visual SLAM system. This system is developed based on ORB-SLAM2 and optical flow algorithm. Compared with ORB-SLAM2, our method has an obvious advantage over computation speed as it do not need to extract descriptors in $Tracking$ thread.

In experiments, we demonstrate FastORB-SLAM can produce SOTA performance in indoor scenes with an RGB-D camera in terms of location accuracy and efficiency. Compared with ORB-SLAM2, our method runs nearly twice faster than ORB-SLAM2 with highly competitive accuracy. A video demo is made to demonstrate it.

For future work, we plan to improve FastORB-SLAM from two aspects:

- Adopt fewer but more homogeneous keypoints (see Fig. 13), which will presumably further improve computation efficiency.
- Extend FastORB-SLAM to multisensor SLAM (such as stereo cameras or IMU), which will improve robustness in challenging scenes, e.g., outdoor scenes or quick movement.

References

[1] Mur-Artal R, Tards J D. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras[J]. IEEE Transactions on Robotics, 2017, 33(5): 1255-1262.

[2] Mur-Artal R, Montiel J M M, Tardos J D. ORB-SLAM: a versatile and accurate monocular SLAM system[J]. IEEE transactions on robotics, 2015, 31(5): 1147-1163.

[3] Forster C, Pizzoli M, Scaramuzza D. SVO: Fast semi-direct monocular visual odometry[C]//2014 IEEE international conference on robotics and automation (ICRA). IEEE, 2014: 15-22.

[4] Engel J, Koltun V, Cremers D. Direct sparse odometry[J]. IEEE transactions on pattern analysis and machine intelligence, 2017, 40(3): 611-625.

[5] Forster C, Zhang Z, Gassner M, et al. SVO: Semidirect visual odometry for monocular and multicamera systems[J]. IEEE Transactions on Robotics, 2016, 33(2): 249-265.

[6] Tateno K, Tombari F, Laina I, et al. Cnn-slam: Real-time dense monocular slam with learned depth prediction[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017: 6243-6252.

[7] Loo S Y, Amiri A J, Mashohor S, et al. CNN-SVO: Improving the mapping in semi-direct visual odometry using single-image depth prediction[C]//2019 International Conference on Robotics and Automation (ICRA). IEEE, 2019: 5218-5223.

[8] Sumikura S, Shibuya M, Sakurada K. OpenVSLAM: A Versatile Visual SLAM Framework[C]//Proceedings of the 27th ACM International Conference on Multimedia. 2019: 2292-2295.

[9] G. Salem, J. Krynitsky, M. Hayes, T. Pohida and X. Burgos-Artizzu, "Three-Dimensional Pose Estimation for Laboratory Mouse From Monocular Images," in IEEE Transactions on Image Processing, vol. 28, no. 9, pp. 4273-4287, Sept. 2019, doi: 10.1109/TIP.2019.2908796.

[10] Y. Wang, B. Zhang and C. Peng, "SRHandNet: Real-Time 2D Hand Pose Estimation With Simultaneous Region Localization," in IEEE Transactions on Image Processing, vol. 29, pp. 2977-2986, 2020, doi: 10.1109/TIP.2019.2955280.

[11] Zhou H, Zhang T, Lu W. Vision-based pose estimation from points with unknown correspondences[J]. IEEE transactions on image processing, 2014, 23(8): 3468-3477.

[12] Yu H, Fu Q, Yang Z, et al. Robust robot pose estimation for challenging scenes with an RGB-D camera[J]. IEEE Sensors Journal, 2018, 19(6): 2217-2229.

[13] Kerl C, Sturm J, Cremers D. Dense visual SLAM for RGB-D cameras[C]//2013 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2013: 2100-2106.

[14] Endres F, Hess J, Sturm J, et al. 3-D mapping with an RGB-D camera[J]. IEEE transactions on robotics, 2013, 30(1): 177-187.

[15] Dai A, Niener M, Zollhfer M, et al. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration[J]. ACM Transactions on Graphics (ToG), 2017, 36(4): 1.

[16] Dai A, Niener M, Zollhfer M, et al. Schops T, Sattler T, Pollefeys M. Bad slam: Bundle adjusted direct rgb-d slam[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2019: 134-144.

[17] Han L, Xu L, Bobkov D, et al. Real-time global registration for globally consistent rgb-d slam[J]. IEEE Transactions on Robotics, 2019, 35(2): 498-508.

[18] Gomez-Ojeda R, Moreno F A, Zuiga-Nol D, et al. PL-SLAM: a stereo SLAM system through the combination of points and line segments[J]. IEEE Transactions on Robotics, 2019, 35(3): 734-746.

[19] Wen S, Zhao Y, Zhang H, et al. Joint optimization based on direct sparse stereo visual-inertial odometry[J]. Autonomous Robots, 2020: 1-19.

[20] Fu Q, Yu H, Lai L, et al. A Robust RGB-32D SLAM System With Points and Lines for Low Texture Indoor Environments[J]. IEEE Sensors Journal, 2019, 19(21): 9908-9920.

[21] Klein G, Murray D. Parallel tracking and mapping for small AR workspaces[C]//2007 6th IEEE and ACM international symposium on mixed and augmented reality. IEEE, 2007: 225-234.

[22] Glvez-Lpez D, Tardos J D. Bags of binary words for fast place recognition in image sequences[J]. IEEE Transactions on Robotics, 2012, 28(5): 1188-1197.

[23] Strasdat H, Davison A J, Montiel J M M, et al. Double window optimisation for constant time visual SLAM[C]//2011 international conference on computer vision. IEEE, 2011: 2352-2359.

[24] Birchfield S. KLT: An implementation of the Kanade-Lucas-Tomasi feature tracker[J]. http://www. ces. clemson. edu/ stb/klt/, 2007.

[25] Yang N, Wang R, Stuckler J, et al. Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry[C]//Proceedings of the European Conference on Computer Vision (ECCV). 2018: 817-833.

[26] Yang N, Stumberg L, Wang R, et al. D3VO: Deep Depth, Deep Pose and Deep Uncertainty for Monocular Visual Odometry[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020: 1281-1292.

[27] Wang R, Schworer M, Cremers D. Stereo DSO: Large-scale direct sparse visual odometry with stereo cameras[C]//Proceedings of the IEEE International Conference on Computer Vision. 2017: 3903-3911.

[28] Schubert D, Demmel N, Usenko V, et al. Direct sparse odometry with rolling shutter[C]//Proceedings of the European Conference on Computer Vision (ECCV). 2018: 682-697.

[29] Von Stumberg L, Usenko V, Cremers D. Direct sparse visual-inertial odometry using dynamic marginalization[C]//2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2018: 2510-2517.

[30] Qin T, Li P, Shen S. Vins-mono: A robust and versatile monocular visual-inertial state estimator[J]. IEEE Transactions on Robotics, 2018, 34(4): 1004-1020.

[31] Qin T, Pan J, Cao S, et al. A general optimization-based framework for local odometry estimation with multiple sensors[J]. arXiv preprint arXiv:1901.03638, 2019.

[32] Usenko V, Demmel N, Schubert D, et al. Visual-inertial mapping with non-linear factor recovery[J]. IEEE Robotics and Automation Letters, 2019.

[33] Gao X, Wang R, Demmel N, et al. LDSO: Direct sparse odometry with loop closure[C]//2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2018: 2198-2204.

[34] Dai Z, Huang X, Chen W, et al. A Comparison of CNN-Based and Hand-Crafted Keypoint Descriptors[C]//2019 International Conference on Robotics and Automation (ICRA). IEEE, 2019: 2399-2404.

[35] Hou Y, Zhang H, Zhou S. BoCNF: efficient image matching with Bag of ConvNet features for scalable and robust visual place recognition[J]. Autonomous Robots, 2018, 42(6): 1169-1185.

[36] Wang X, Peng G, Zhang H. Combining multiple image descriptions for loop closure detection[J]. Journal of Intelligent and Robotic Systems, 2018, 92(3-4): 565-585.

[37] Rublee E, Rabaud V, Konolige K, et al. ORB: An efficient alternative to SIFT or SURF[C]//2011 International conference on computer vision. Ieee, 2011: 2564-2571.

[38] Bescos B, Fcil J M, Civera J, et al. DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes[J]. IEEE Robotics and Automation Letters, 2018, 3(4): 4076-4083.

[39] Sujiwo A, Ando T, Takeuchi E, et al. Monocular vision-based localization using ORB-SLAM with LiDAR-aided mapping in real-world robot challenge[J]. Journal of robotics and mechatronics, 2016, 28(4): 479-490.

[40] Buyval A, Afanasyev I, Magid E. Comparative analysis of ROS-based monocular SLAM methods for indoor navigation[C]//Ninth International Conference on Machine Vision (ICMV 2016). International Society for Optics and Photonics, 2017, 10341: 103411K.

[41] Zhao Z, Mao Y, Ding Y, et al. Visual Semantic SLAM with Landmarks for Large-Scale Outdoor Environment[J]. arXiv preprint arXiv:2001.01028, 2020.

[42] Webb A M, Brown G, Lujn M. ORB-SLAM-CNN: Lessons in Adding Semantic Map Construction to Feature-Based SLAM[C]//Annual Conference Towards Autonomous Robotic Systems. Springer, Cham, 2019: 221-235.

[43] Campos, Carlos, et al. "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM." arXiv preprint arXiv:2007.11898 (2020).

[44] Stark J A. Adaptive image contrast enhancement using generalizations of histogram equalization[J]. IEEE Transactions on image processing, 2000, 9(5): 889-896.

[45] Bian J W, Lin W Y, Matsushita Y, et al. Gms: Grid-based motion statistics for fast, ultra-robust feature correspondence[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017: 4181-4190.

[46] Bian, J W, Lin W Y, et al. GMS: Grid-Based Motion Statistics for Fast, Ultra-robust Feature Correspondence. International Journal of Computer Vision, 128: 15801593, 2020. doi.org/10.1007/s11263-019-01280-3.

[47] Xu G, Zhang Z. Epipolar geometry in stereo, motion and object recognition: a unified approach[M]. Springer Science Business Media, 2013.

[48] Whelan T, Salas-Moreno R F, Glocker B, et al. ElasticFusion: Real-time dense SLAM and light source estimation[J]. The International Journal of Robotics Research, 2016, 35(14): 1697-1716.

[49] Whelan T, Kaess M, Fallon M, et al. Kintinuous: Spatially extended kinectfusion[J]. 2012.

[50] Sturm J, Engelhard N, Endres F, et al. A benchmark for the evaluation of RGB-D SLAM systems[C]//2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2012: 573-580.

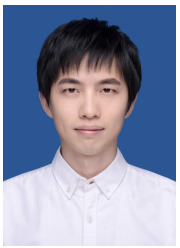
[51] Handa A, Whelan T, McDonald J, et al. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM[C]//2014 IEEE international conference on Robotics and automation (ICRA). IEEE, 2014: 1524-1531.
[52] Kmmerle R, Grisetti G, Strasdat H, et al. g 2 o: A general framework for graph optimization[C]//2011 IEEE International Conference on Robotics and Automation. IEEE, 2011: 3607-3613.

Qiang Fu was born in China. He is currently working toward a Ph.D. degree with the National Engineering Laboratory for Robot Visual Perception and Control, College of Electrical And Information Engineering, Hunan University, China, under the supervision of Prof. Hongshan Yu.

He is also currently a Visiting Scholar with the University of Alberta, Robotic and Vision Group, Edmonton, AB, Canada, under the supervision of Prof. Hong Zhang (IEEE Fellow). His research interests include mobile robot, visual SLAM, computer vision.

Hongshan Yu received the B.S., M.S., and Ph.D. degrees in control science and technology in electrical and information engineering from Hunan University, Changsha, China, in 2001, 2004, and 2007, respectively.

From 2011 to 2012, he was a Post-Doctoral Researcher with the Laboratory for Computational Neuroscience, University of Pittsburgh, USA. He is currently a Professor with Hunan University and Associate Dean of the National Engineering Laboratory for Robot Visual Perception and Control. His research interests include autonomous mobile robot and machine vision, with over 30 publications in these areas.



Xiaolong Wang received his B.Sc. degree in Applied Mathematics, M.Sc. degree in Computational Mathematics, and Ph.D. degree in Applied Mathematics from Northwestern Polytechnical University (NWPU), China. He was a visiting doctoral student in the department of computing science at University of Alberta, Canada. From 2019 to 2020, he was a Post-Doctoral Researcher at the University of Alberta, Robotic and Vision Group, under the supervision of Prof. Hong Zhang (IEEE Fellow).

He is currently a lecturer in School of Mathematics and Information Science, Shaanxi Normal University, China. His research is in the area of visual SLAM system and collaborative perception.

Zhengeng Yang received the B.S. and M.S. degrees from Central South University, Changsha, China, in 2009 and 2012, respectively. He is currently working toward his Ph.D. degree with the National Engineering Laboratory for Robot Visual Perception and Control, Hunan University, China, under the supervision of Prof. Hongshan Yu. He is also currently a Visiting Scholar with the University of Pittsburgh, Pittsburgh, PA, USA.

His research interests include computer vision, image analysis, and machine learning.

Hong Zhang (Fellow, IEEE) received the B.S. degree from Northeastern University, Boston, MA, USA, in 1982, and the Ph.D. degticlree from Purdue University, West Lafayette, IN, USA, in 1986, both in electrical engineering.

He conducted research at the University of Pennsylvania, Philadelphia, PA, USA, as a PostDoctoral Fellow. He is currently a Professor with the Department of Computing Science, University of Alberta, Edmonton, AB, Canada. His research interests include robotics, computer vision, and image processing, with over 200 publications in these areas. For the past 15 years, he has expended considerable effort in the study of mobile robot navigation with visual sensing.

Dr. Zhang is a fellow of the Canadian Academy of Engineering in recognition of his accomplishments. He held a prestigious NSERC Industrial Research Chair from 2003 to 2017. He was the General Chair of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), in Vancouver, Canada, and is on the Senior Program Committee (SPC) of IROS 2018, IROS 2019, and ICRA 2019. He is serving as the Secretary of IEEE Robotics and Automation Society (20182019), and has been appointed as the Editor-in-Chief of the IROS Conference Editorial Board for a term of three years from 2020 to 2022. He has served on the editorial boards of several international journals including the IEEE TRANSACTIONS ON CYBERNETICS and the MDPI journal of Robotics. He is a Principal Investigator in the NSERC Canadian Robotics Networks, the NCFRN from 2012 to 2017, and the NCRN from 2018 to 2023, whose mandate is to develop the science and technologies to allow mobile robots to work in challenging environments and to generate and communicate critical information to humans.

Ajmal Mian is a Professor of Computer Science at The University of Western Australia. He has received two prestigious fellowships and several research grants from the Australian Research Council and the National Health and Medical Research Council of Australia with a combined funding of over $12 million. He was the West Australian Early Career Scientist of the Year 2012 and has received several awards including the Excellence in Research Supervision Award, EH Thompson Award, ASPIRE Professional Development Award, Vice-chancellors Mid-career Award, Outstanding Young Investigator Award, the Australasian Distinguished Dissertation Award and various best paper awards. He is an Associate Editor for 3 journals that are ranked A* by CORE i.e. TNNLS, TIP and PR. His research interests are in computer vision, 3D deep learning, shape analysis, face recognition, human action recognition and video analysis.