13th International Conference on Advanced Computational Intelligence (ICACI)
Wanzhou, China, May 14-16, 2021

2021 13th International Conference on Advanced Computational Intelligence (ICACI) | 978-1-6654-1254-4/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICACI52617.2021.9435915

ORB-SLAM2S: A Fast ORB-SLAM2 System with Sparse Optical Flow Tracking

Yufeng Diao
School of Automation
Chongqing University
Chongqing, China
diaoyufeng@cqu.edu.cn

Ruping Cen
School of Automation
Chongqing University
Chongqing, China
cenruping@cqu.edu.cn

Fangzheng Xue
School of Automation
Chongqing University
Chongqing, China
xuefangzheng@cqu.edu.cn

Xiaojie Su*
School of Automation
Chongqing Unversity
Chongqing, China
suxiaojie@cqu.edu.cn

***Abstract*—This paper presents ORB-SLAM2S, a fast and complete simultaneous localization and mapping (SLAM) system based on ORB-SLAM2 for monocular, stereo, and RGB-D cameras. The system works, ensuring accuracy simultaneously, in real-time on standard central processing units (CPU) at a faster speed in small and large indoor and outdoor environments. The system includes a lightweight front-end which is a sparse optical flow method for non-keyframes to avoid the extraction of keypoints and descriptors that allows for high-speed real-time performance. For keyframes, a feature-based method is used to ensure the accurate trajectory estimation almost the same as ORB-SLAM2. The evaluation of famous public sequences shows that our method achieves almost the same state-of-the-art accuracy as ORB-SLAM2 and faster speed performance which is 3~5 times that of ORB-SLAM2, being in most cases the faster SLAM solution. As proved by experiments, the system provides a fast and lightweight visual SLAM while ensuring accuracy for low-cost mobile devices.**

Keywords*—*visual SLAM, real-time performance, trajectory accuracy

I. INTRODUCTION

Simultaneous localization and mapping (SLAM) was first put forward in the field of robots [1-3]. It means that the robot starts from the unknown location of the unknown environment, locates its position and posture by repeatedly observing the environmental characteristics in the movement process, and then constructs the surrounding environment's incremental map according to its position. Compared with other sensors, the camera has the advantages of low hardware cost and rich visual information, which has attracted many scholars and companies. The SLAM system only using camera as external sensor is called visual SLAM (VSLAM) [3] that provides a real-time pose for mobile robots.

With the rapid development of robots, unmanned aerial vehicles, unmanned driving, virtual reality (VR), and augmented reality (AR) in recent years, VSLAM has become a hot research topic in the academic and application fields and has been considered as the key technology to realize autonomous mobile robots, VR and AR [4]. The commercialization of VSLAM to AR, VR, and other mobile devices requires a lot of computing power, even expensive GPU. Therefore, a lightweight, fast, real-time, and accurate system is still the key problem of VSLAM. Real-time is a key performance index of VSLAM technology to provide a faster real-time optimal posture when the mobile robot, AR and VR, are running dynamically with a strong focus on real-time operation.

Most feature-based VSLAM systems [5] can perform in real-time on a standard CPU at about 10~30 HZ using monocular, stereo, or RGB-D cameras, and include loop detection and relocalization capabilities based on descriptors. However, the feature-based method takes a lot of time to extract invariant corners and robust descriptors for each frame, which becomes the main factor limiting the real-time performance. The direct method [6] and the optical flow method [7] avoids extracting and matching descriptors so that a faster speed performance is obtained. However, they lose the function of loop detection and relocalization based on descriptors.

Therefore, this study proposes ORB-SLAM2S based on ORB-SLAM2 [8]. By combining feature-based and sparse optical flow methods, all these issues are solved and allow for not only retain the trajectory accuracy of the feature-based method but also obtain the faster real-time performance of the sparse optical flow method. The main contributions of this paper are as follows:

- A complete, light-weight, accurate, and fast VSLAM system for monocular, stereo, and RGB-D cameras, including loop closing, and relocalization;
- By using a light-weight front-end combining feature-based and sparse optical flow methods, the faster real-time performance than ORB-SLAM2 is achieved, ensuring accuracy simultaneously;
- The stereo, RGB-D, and monocular results show that ORB-SLAM2S achieves a faster speed performance while ensuring accuracy.

The paper is organized as follows: Section II introduces the related work, Section III describes the system, then presents the evaluation results in Section IV, and conclusions are finally made in Section V.

II. RELATED WORK

Trajectory accuracy and real-time performance are two important indexes of VSLAM. According to the need for feature points and descriptors, the current VSLAM system is divided into the feature-based method, direct method, and optical flow method. In this part, the different methods' trajectory accuracy and real-time performance will be discussed.

A. *Feature-based Method*

The general method of feature-based SLAM extracts a series of sparse and stable edges [9], blocks [10], or corners [11-13] from each frame, then some schemes extract invariant descriptors corresponding to features, and use descriptors to match features. With these matched keypoints, the motion and structure of the camera are recovered by multi-view geometry. In this kind of SLAM algorithm, due to the existence of robust descriptors of keypoints, the feature-based method can establish the common view relationship for a longer time and add more constraint relations, such as covisibility graph [14], relocalization, and loop detection, so that this method can obtain better robustness and trajectory accuracy. However, the extraction of keypoints and invariant descriptors is extraordinarily time-consuming, which is a common fault of the feature-based method. ORB-SLAM [8], [11], one of the most perfect and easy-to-use systems in the modern SLAM system, represents the mainstream feature-based methods' peak. ORB-SLAM2 system extracts each frame's features and descriptors but only retains and uses for location, mapping, and optimization in keyframes. For non-keyframes, the keypoints and descriptors will be discarded after solving the pose, which has no contribution to the mapping and trajectory accuracy. Therefore, only using keyframe's keypoints and descriptors can almost guarantee all the advantages of the feature-based method. For non-keyframes, it is unnecessary to extract the keypoints and descriptors with high time consumption costs.

In ORB-SLAM2S, features are extracted only for selected keyframes, which ensures the capabilities of loop closing and relocalization and reduces the computation time significantly. Therefore, ORB-SLAM2S achieves almost the same state-of-the-art accuracy as ORB-SLAM2.

B. *Direct Method*

The direct method does not extract keypoints and descriptors. It only uses the nonlinear optimization tools such as G2O [15] to minimize the photometric error and get the pose while building a semi-dense map [6]. However, this method is greatly affected by illumination. The gray gradient's nonconvexity makes the direct method easy to fall into the local extremum and can not get the optimal solution. The real-time performance of the direct method is determined by the number of tracking points. When the number of tracking points is small, the speed is so fast. For instance, in the sparse direct (such as DSO [16]), when the number of tracking points is large, the real-time performance will decline. Meanwhile, the trajectory accuracy is limited by the number of tracking points. To obtain higher accuracy, only quantity correction quality can be used. In some methods, they use a semi-direct way to track the apparent texture features to maintain the balance between sparsity and trajectory accuracy, such as SVO [17]. In short, due to the lack of keypoints and descriptors, the direct method lacks the ability of loop detection, which makes our final map results not globally consistent.

C. *Optical Flow Method*

The sparse optical flow method is more mature among optical flow methods in the SLAM field. This method only extracts stable keypoints (such as FAST [18]) when necessary and does not extract descriptors. Under the assumption of invariable gray value, the corresponding matching points are obtained by minimizing the photometric error, and the pose is solved by matching point pairs. This method's advantage is that there is no need to extract descriptors and only extract a small number of stable keypoints, which is a trade-off between speed and accuracy. However, due to the lack of descriptors, this method lacks loop closing and relocalization.

In ORB-SLAM2S, a sparse optical flow method is used for non-keyframes tracking from frame to frame with sub-pixel precision to avoid the extraction of descriptors that allows for high-speed real-time performance. Therefore, our method is light-weight and works with standard central processing units (CPUs) at a faster real-time performance.

III. ORB-SLAM2S

The general overview of the ORB-SLAM2S system is shown in Fig. 1. ORB-SLAM2S is built on feature-based ORB-SLAM2, whose main content is summarized here for the convenience of readers. The three main parallel threads work as follows:

1) *Tracking*: the tracking thread combines sparse optical flow method (optical flow tracking module in Fig. 1) and feature-based method (feature-based tracking module in Fig. 1) to find matching with the local map and applies motion-only bundle adjustment (BA) to minimize reprojection error, so as to track each frame;

2) *Local Mapping*: the local mapping thread manages and optimizes the local covisibility graph and keyframe, and implements local BA;

3) *Loop Closing*: the loop closing thread detects large loops and corrects cumulative drift by performing pose optimization. After optimizing the pose graph, the thread enables the fourth thread to execute full BA and calculates the optimal motion and sparse point cloud solution.

Note that our system is only different from ORB-SLAM2 in the tracking thread. In the rest of this section, We will present how to combine sparse optical flow with the feature-based method to speed up and ensure accuracy at the same time in the tracking thread. For a detailed description of other system blocks, ORB-SLAM2's publication [8][11] is referred to see.

A. *System Bootstrapping*

When the system startup, the map initialization based on the feature-based method will work. It uses the first frame to create the first keyframe, sets its pose as the origin, and creates the initial map from all the keypoints. With the map and keyframes, the optical flow tracking is used to solve only the pose of each frame by tracking keyframes indirectly without feature points and descriptors extracted. When the number of tracked feature points is small, this frame is considered to be a keyframe and the feature-based tracking module will work.

The tracking thread schematic diagram is shown in Fig. 2, in which map points are generated by keyframes and then tracked by the optical flow method indirectly.

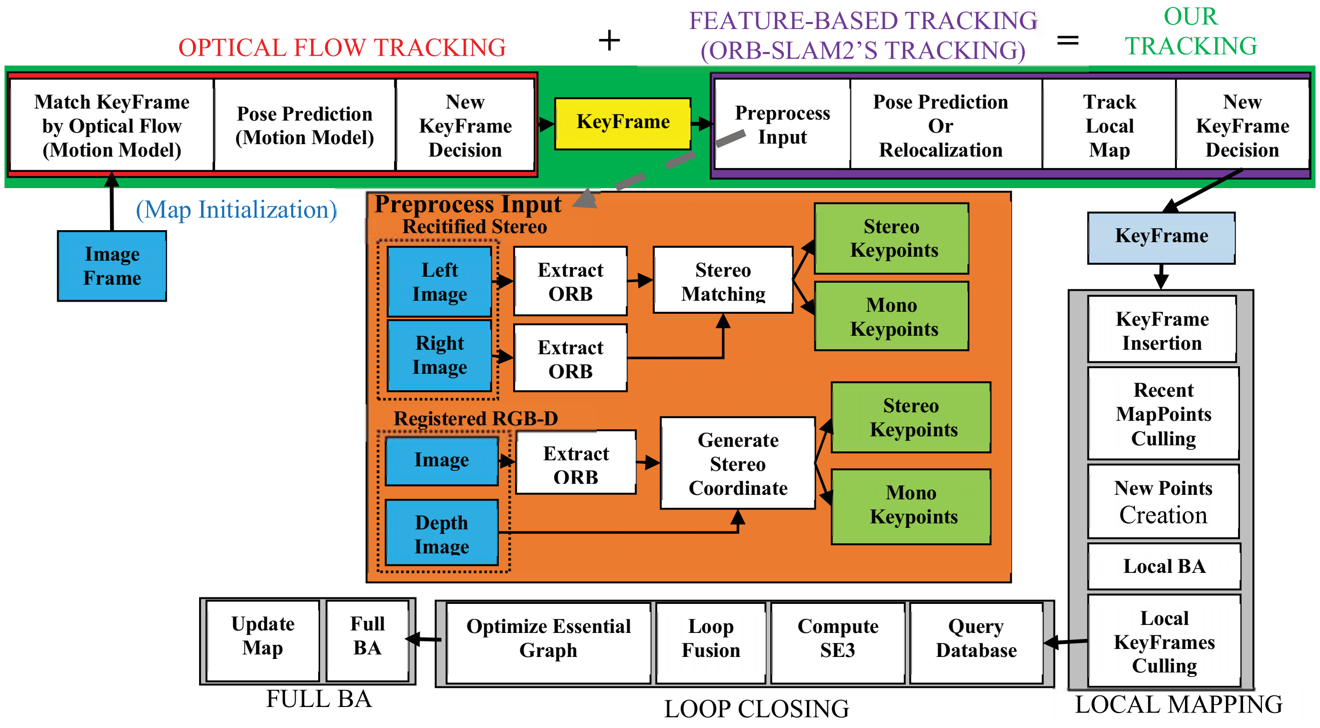

Fig. 1. ORB-SLAM2S system is developed with three main thread parallel structures, including tracking, local mapping and loop closing threads, which can create a fourth thread to perform full BA after a loop closure. our tracking thread (green block) adopts a sparse optical flow method for non-keyframes, as shown in the red block; in keyframes, the feature-based method is used, as shown in the purple block.

For non-keyframes, optical flow is used to track the keyframes to establish a matching with the map points of keyframes. In the tracking process (shown by the arrow in Fig. 2), the keyframes' map points tracked in the first frame are passed through by optical flow from frame to frame to indirectly track the keyframe. In this process, no new keypoints and descriptors will be extracted. The advantages of this strategy are as follows:

1) due to the existence of covisibility graph and loop closing, the accuracy of feature-based keyframes is higher. Therefore, the keyframes are tracked indirectly instead of directly solving the motion increment from frame to frame to avoid the accumulated pose drift.

2) for the sensitivity of optical flow to illumination, the stability of the optical flow algorithm is ensured through frame-to-frame tracking instead of keyframe-to-frame.

Only keypoints and descriptors are extracted from keyframes, instead of extracting every frame as the original system, to achieve the acceleration and maintain accuracy.

B. Track KeyFrame with Optical Flow

Optical flow: parse optical flow is used in the optical flow tracking module. The pixel gray of the reference frame can be written as $I_r(u,v)$. Suppose that the reference point moves to $(u+\Delta u, v+\Delta v)$ at current frame with pixel gray $I_c(u+\Delta u, v+\Delta v)$, and the pixel gray remains unchanged, then the optical flow problem can be defined as [19]:

$$\{\Delta u,\Delta v\}=\underset{\Delta u,\Delta v}{\arg\min}\sum_{u=-w}^{w}\sum_{v=-w}^{w}\| I_r(u,v)-I_c(u+\Delta u,v+\Delta v)\|_2^2 \qquad (1)$$

where $\{\Delta u, \Delta v\}$ are the movement increment and w represents a square block with (u,v) as the center and $(2w+1)$ as the side length.

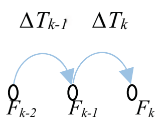

Fig. 3. In the uniform motion model, we assume that the motion ΔT_{k-1} at the last moment is the same as the current moment, $\Delta T_k \approx \Delta T_{k-1}$.

Uniform Motion Model: So far, the matching point pairs are obtained by the optical flow method by formula (1), but the assumption of the invariable gray value is too ideal, it is only suitable for local small range matching. To overcome this shortcoming, a uniform motion model is used as shown in Fig. 3. The motion increment from $k-1$ to $k-2$ frame is defined as $\Delta T \in SE(3)$, assuming that the motion increment of the current frame is the same as the last frame, then there is:

$$T_{cw}^{k\,*} = \Delta T T_{cw}^{k-1} \qquad (2)$$

where $T_{cw}^{k\,*}$ represents the predicted camera pose of the current frame under the assumption of the constant speed model and T_{cw}^{k-1} means the camera pose of the last frame. Project the map point $P_w(x, y, z)$ in the world coordinate system, generated by the keyframes, into the current frame's image coordinate system according to our predicted posture, then the current frame's image coordinates $P_c^*(u^*, v^*)$ will be predicted:

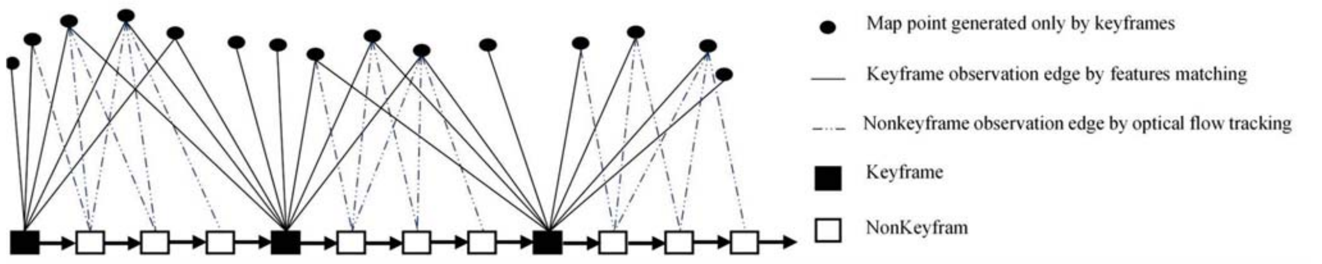
Map point generated only by keyframes
Keyframe observation edge by features matching
Nonkeyframe observation edge by optical flow tracking
Keyframe
NonKeyfram

Fig. 2. In the tracking thread operation diagram of the ORB-SLAM2S system, for the keyframe, we extract the keypoints and descriptors, and get the map points and pose according to the descriptor matching triangulation and pose solution. For non-keyframes, the optical flow is used to solve the matching and only solve the pose, which has no contribution to the map.

$$P_c^* = \frac{1}{z} K T_{cw}^{k*} P_w \tag{3}$$

where K is the camera's internal parameters, which can be obtained by camera calibration, and z is the depth of the map point in the camera coordinate system.

The predicted coordinates P_c^* (u^*, v^*) at the current frame image will serve as the initial value of our optical flow iteration in (1):

$$\{\Delta u, \Delta v\} = \arg\min_{\Delta u, \Delta v} \sum_{u=-w}^{w} \sum_{v=-w}^{w} \| I_r(u,v) - I_c(u^* + \Delta u, v^* + \Delta v) \|_2^2 \tag{4}$$

where u^* and v^* respectively represent the abscissa and ordinate of the predicted coordinates P_c^* (u^*, v^*) in (3).

This efficient model avoids falling into local extreme value or not satisfying the assumption of gray level invariance due to excessive motion. At the same time, it speeds up the iterative solution of least squares.

C. Pose Prediction

The system performs BA to optimize the camera's pose $T_{cw} \in SE(3)$ (motion-only BA). Motion-only BA minimizes the reprojection error between matched 3-D points $P_w^i \in \mathbb{R}^3$ in world coordinates and keypoints P_{uv}^i, with $i \in \chi$ the set of all matches:

$$\{T_{cw}\} = \arg\min_{T_{cw}} \sum_{i \in \chi} \eta \left(\left\| P_{uv}^i - K T_{cw} P_w^i \right\|_\Sigma^2 \right) \tag{5}$$

where K is the camera's internal parameter, Σ is the covariance matrix, and η is the robust kernel function.

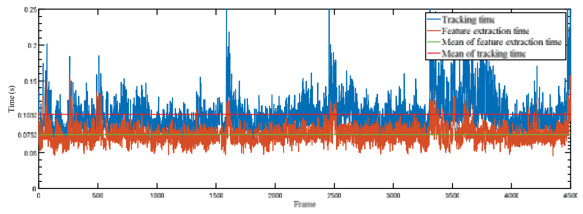

Fig. 4. The time consumption results of ORB-SALM2 system using the stereo camera on the KITTI 00 sequence dataset [20].

D. New KeyFrame Decision

In our optical flow tracking module, when the number of tracking map points is small, it will be considered as a keyframe and the feature-based tracking module is enabled. The system can work well when the threshold is set to 200. In the feature point tracking module, the strategy of the original system is followed, using descriptor matching to establish a covisibility graph, and culling the redundant keyframes.

E. Feature-Based Tracking

As shown in Fig. 4, the feature-based method consumes lots of time, and the mean time spent on extracting ORB features accounts for 73% of the whole tracking thread. Therefore, the tracking module based on features is only used in keyframes. When the optical flow tracking module sends a keyframe, the feature-based tracking will be enabled. As shown in the Input Processing module (orange block in Fig. 1), the ORB features will be extracted, and use the BRIEF descriptor for matching; next use the result of the optical flow method as the initial value to optimize the pose; then track the local map to ensure better local consistency, and finally remove the redundant image frame and send it to the local mapping thread.

Note that the difference from the original system is that the pose of the current frame has been calculated through the optical flow method, so the descriptor matching and pose optimization will be faster. This will be shown in section IV.

IV. Evaluation

The system has been evaluated in detail on stereo, RGB-D, and monocular cameras in two popular datasets, including indoor and outdoor large and small scenes. It is compared with the original state-of-the-art system. Our comparison experiments run in the notebook computer with Intel Core i5 dual-core CPUs and 16GB memory. Each frame's time consumption and trajectory accuracy comparison are shown on each dataset in this section. Each frame's time consumption and trajectory accuracy comparison are shown on each dataset in this section. Two standard metrics are used, including the absolute pose root-mean-square error (RMSE) tabs proposed in [21], and the mean time consumption of the tracking module.

A. TUM RGB-D Dataset

The TUM dataset [21] is an indoor sequence from the RGB-D sensor, which contains the different textures, lighting, and structural conditions to evaluate VSLAM. The evaluation results are shown in Table I. The results show that the trajectory accuracy of our system is almost the same as the original system, and the speed performance is improved about 3~5 times.

TABLE I
RGB-D CAMERA'S COMPARISON OF ACCURACY and TIME CONSUMPTION IN TUM DATASET

Sequence	ORB-SLAM2S mean time (ms)	ORB-SLAM2S RMSE (m)	ORB-SLAM2 mean time (ms)	ORB-SLAM2 RMSE (m)
fr1/desk	**11.0185**	0.02167	37.5365	**0.01587**
fr1/room	**14.0279**	**0.05361**	44.1161	0.07988
fr1/rpy	**14.6708**	0.04352	42.2226	**0.02117**
fr1/xyz	**9.6090**	0.01069	45.1799	**0.00979**
fr2/desk	**7.4022**	0.01502	46.8412	**0.00963**

Comparison of Absolute Pose RMSE (m) and Time Consumption in tracking thread.

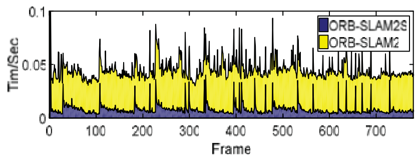

Fig. 5. Time consumption comparison for RGB-D in TUM fr1/xyz.

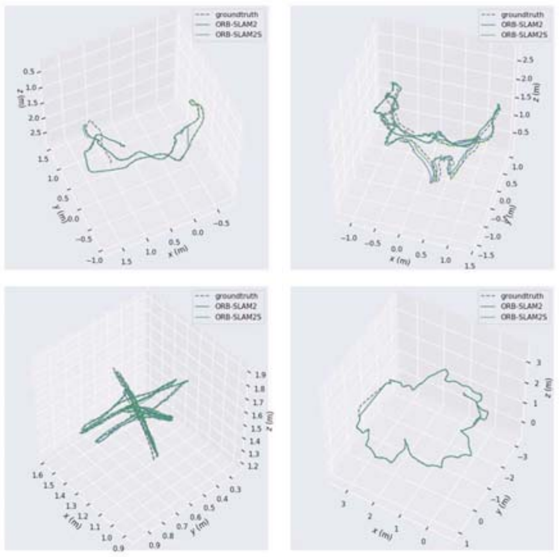

Fig. 6. Estimated trajectory and ground truth in TUM fr1/desk, fr1/room, fr1/xyz, and fr2/desk.

TABLE II
STEREO CAMERA IN THE KITTI DATASET

Sequence	ORB-SLAM2S mean time (ms)	ORB-SLAM2S RMSE (m)	ORB-SLAM2 mean time (ms)	ORB-SLAM2 RMSE (m)
00	**29.549**	1.533611	139.835	**1.263961**
01	**-**	-	**183.617**	**9.609990**
02	**36.433**	**5.825880**	138.246	6.051866
03	**21.369**	**0.598045**	139.835	0.898246
04	**37.361**	0.322265	124.961	**0.232865**
05	**25.253**	0.888850	129.752	**0.737796**
06	**34.534**	**0.629387**	141.444	0.657803
07	**20.562**	1.285786	112.928	**0.559112**
08	**25.865**	4.430031	124.422	**3.489849**
09	**30.352**	4.099652	122.417	**3.012105**
10	**23.408**	1.569769	108.509	**1.003910**

Comparison of Absolute Pose RMSE (m) and Time Consumption in tracking thread. "–" indicates bad accuracy.

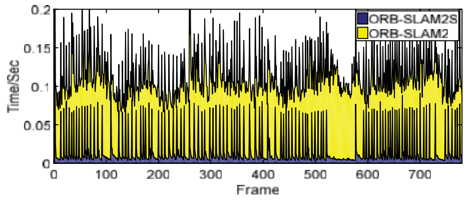

Fig. 7. Time consumption comparison for stereo in KITTI 00.

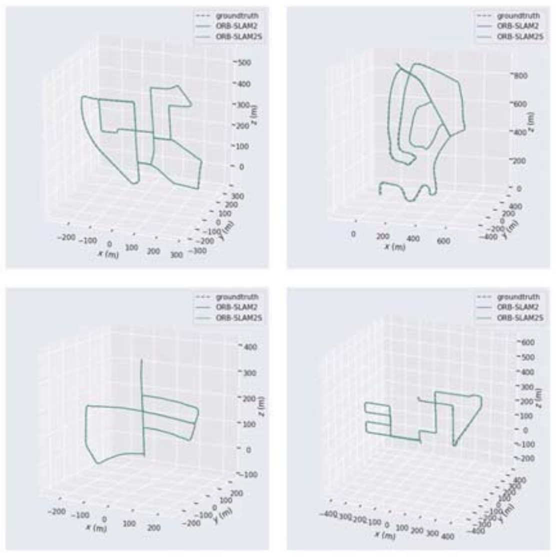

Fig. 8. Estimated trajectory and ground truth for stereo in KITTI 00.

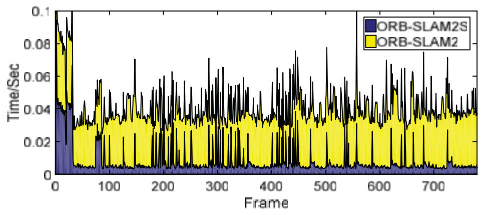

Fig. 9. Time consumption comparison for monocular in TUM fr1/xyz.

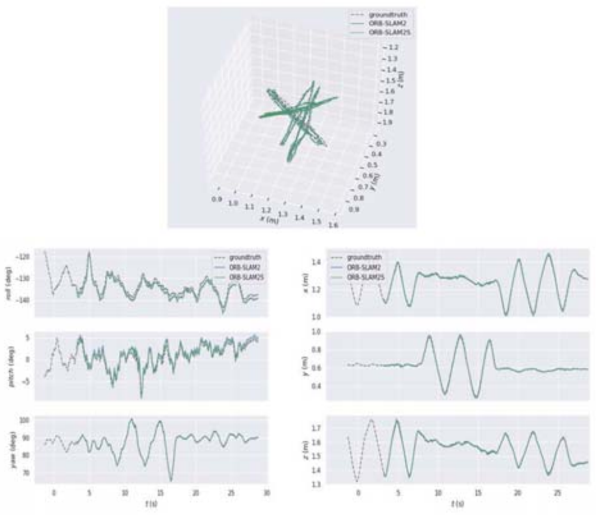

Fig. 10. Estimated trajectory and 6 degrees of freedom including angle and translation for monocular in TUM fr1/xyz.

Figure 5 shows the time consumption of each frame in the tracking module. Fig. 5 indirectly reflects the working process of our system. Firstly, there are many spaced burrs, which are time-consuming keyframes based on the features. Most of the frames belong to low-cost non-keyframes at the bottom, and there is no need to extract keypoints and descriptors. Secondly, from the keyframe to the keyframe, there is a triangle shape of time decreasing with the big head and small tail. That is because the keypoints are gradually lost in the process of indirect transfer tracking. When there are few tracked keypoints, the feature-based tracking is started. Finally, It is noted that the time consumption of our keyframe is slightly lower than the original system because our keyframe insertion is not so dense so that the multi-thread is not congested. At the same time, our keyframe takes the optical flow tracking calculation result as the initial value, which effectively reduces the matching range of descriptors and the iterative time of later solution. Some examples of estimated trajectories are shown in Fig. 6.

B. *KITTI Stereo Datasets*

The KITTI dataset [20] includes stereo sequences with a resolution after rectification of 1240×376 pixels collected by a car in urban and highway environments. Sequences 00, 02, 05, 06, 07 and 09 contain loops. ORB-SLAM2S can detect all loops just like the original system, which verifies that our optical flow tracking module does not affect the feature-based loop detection function. Based on the published ground truth, the evaluation results of all 11 sequences are shown in Table II. The trajectory accuracy of our system is slightly worse than the original system, but the speed performance is improved about 3~5 times. In sequence 01, ORB-SALM2S obtains bad trajectory accuracy, because the sequence contains dynamic moving objects, and the optical flow algorithm does not eliminate it well. The time consumption details of each frame are shown in Fig. 7. Some examples of estimated trajectories are shown in Fig. 8.

C. *TUM Monocular Dataset*

In monocular evaluation, the depth map of the TUM dataset is ignored and only uses the monocular image as the input data. Fig. 9 shows the running time consumption of each frame in detail in TUM fr1/xyz and shows that the speed performance is greatly improved. The example in TUM fr1/xyz shows in Fig. 10 that the trajectory accuracy of our system is almost the same as the original system.

V. Conclusion

In this paper, a complete, light-weight, accurate, and fast VSLAM system has been presented for monocular, stereo, and RGB-D cameras, including loop closing, and relocalization on standard CPUs. As proved by experiments, this system aims to provide a fast and lightweight VSLAM while ensuring accuracy. By using a lightweight front-end combining feature-based and sparse optical flow methods, faster real-time performance than ORB-SLAM2 has been achieved, ensuring accuracy simultaneously.

Future expansion may include constructing the real-time navigation map based on the stereo and RGB-D cameras, such as the octree map for space robots like aircraft and the grid map for plane robots.

References

[1] H. Durrant-Whyte, and T. Bailey. “Simultaneous Localization and Mapping: Part I,” *IEEE Robotics & Automation Magazine,* vol. 13, no. 2, pp. 99-110, 2006.

[2] C. Cadena, L. Carlone, H. Carrillo, et al. “Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age,” *IEEE Transactions on Robotics,* vol. 32, no. 6, pp. 1309-1332, 2019.

[3] D. Zou, P. Tan, and W. Yu. “Collaborative visual SLAM for multiple agents: A brief survey,” *Virtual Reality & Intelligent Hardware*, vol. 1, no. 5, pp. 461-482, 2019.

[4] L. Jinyu, Y. Bangbang, C. Danpeng, et al, “Survey and evaluation of monocular visual-inertial SLAM algorithms for augmented reality,” *Virtual Reality & Intelligent Hardware*, vol. 1, no. 4, pp.386-410, 2019.

[5] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, “ORB-SLAM: A versatile and accurate monocular SLAM system,” *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147-1163, 2017.

[6] J. Engel, T. Schps, and D. Cremers, “LSD-SLAM: Large-scale direct monocular SLAM,” *European Conference on Computer Vision*, pp. 834-849, 2014.

[7] T. Pandey, D. Pena, J. Byrne, et al, “Leveraging deep learning for visual odometry using optical flow,” *Sensors*, vol. 21, no. 4, pp. 1313-1317, 2021.

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

[9] R. Gomez-Ojeda, D. Zuñiga-Noël, F. Moreno, et al, “PL-SLAM: A stereo SLAM system through the combination of points and line segments,” *IEEE Transactions on Robotics*, vol. 35, no. 3, pp. 734-746, 2019.

[10] A. J. Davison, I. D. Reid, N. D. Molton, et al, “MonoSLAM: real-time single camera SLAM,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp.1052-1067, 2007.

[11] E. Rublee, V. Rabaud, K. Konolige, et al, “ORB: An efficient alternative to SIFT or SURF,” *IEEE International Conference on Computer Vision*, pp. 2564-2571, 2012.

[12] D. G. Lowe, “Distinctive Image Features from Scale-Invariant Key-points,” *International Journal of Computer Vision,* vol. 60, no. 2, pp. 91-110, 2004.

[13] H. Bay, A. Ess, T. Tuytelaars, et al, “SURF: Speeded up robust features,” *European Conference on Computer Vision*, vol. 110, no. 3, pp. 440-417, 2006.

[14] H. Strasdat, A. J. Davison, J. M. M. Montiel, et al, “Double window optimisation for constant time visual SLAM,” *International Conference on Computer Vision*, pp. 2352-2359, 2011.

[15] G. Grisetti, R. Kümmerle, H. Strasdat, et al, “G2o: A general framework for (hyper) graph optimization,” *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 9-13, 2011.

[16] J. Engel, V. Koltun, and D. Cremers, “Direct sparse odometry,” *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 40, no. 3, pp. 611-625, 2018.

[17] C. Forster, Z. Zhang, M. Gassner, et al, “SVO: Semi-direct visual odometry for monocular and multicamera systems,” *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 249-265, 2017.

[18] E. Rosten, and T. Drummond, “Machine learning for high-speed corner detection,” *European conference on computer vision*, pp. 430-443, 2006.

[19] B. Simon, R. Gross, and I. Matthews, “Lucas-Kanade 20 years on: A unifying framework,” *International Journal of Computer Vision*, vol. 56, no. 3, pp. 221-255, 2004.

[20] A. Geiger, P. Lenz, C. Stiller, et al, “Vision meets robotics: The KITTI dataset,” *The International Journal of Robotics Research,* vol. 32, no. 11, pp. 1231-1237, 2013.

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