Computers, Materials & Continua CMC, vol.61, no.1, pp.243-254, 2019

Fast Scene Reconstruction Based on Improved SLAM

Zhenlong Du[1, *], Yun Ma[1], Xiaoli Li[1] and Huimin Lu[2]

Abstract: Simultaneous location and mapping (SLAM) plays the crucial role in VR/AR application, autonomous robotics navigation, UAV remote control, etc. The traditional SLAM is not good at handle the data acquired by camera with fast movement or severe jittering, and the efficiency need to be improved. The paper proposes an improved SLAM algorithm, which mainly improves the real-time performance of classical SLAM algorithm, applies KDtree for efficient organizing feature points, and accelerates the feature points correspondence building. Moreover, the background map reconstruction thread is optimized, the SLAM parallel computation ability is increased. The color images experiments demonstrate that the improved SLAM algorithm holds better real-time performance than the classical SLAM.

Keywords: SLAM, thread optimization, scene reconstruction, feature point match.

1 Introduction

With the development of virtual reality (VR)/augmented reality (AR) technology and the hardware performance upgrading, more and more VR/AR applications have been involving into our life and bringing the great convenience to modern people. At the same time, VR/AR related technology has attracted the wide and extensive attention, and VR/AR requirements prompt the related investigation forward. Moreover, the scene localization and the mapping generation are required by automatous robotics navigation, it is urgent to capture the external environment information, reconstruct the previously unknown scene in real-time. In the paper the simultaneous localization and mapping (SLAM) [Zhou, Lian, Yang et al. (2018); Zhang, Liu, Dong et al. (2016); Zhang, He, Chen et al. (2016)] algorithm is investigated.

Although SLAM has made some progresses in recent years, it still encountered some difficulties in practical applications [Cui, McIntosh and Sun (2018)]. Till now, SLAM includes MonoSLAM [Davison, Reid, Molton et al. (2007); Bresson, Feraud, Aufrere et al. (2015)], parallel tracking and mapping(PTAM) [Klein and Murray (2007)], large-scale direct monocular SLAM(LSD-SLAM) [Engel, Schps and Cremers (2014)], EKF-SLAM [Barrau and Bonnabel (2015)], SLAM with RGB-D camera (RGBD-SLAM) [Kerl, Stuckler and Cremers (2015)], these SLAM methods include tracking, depth map estimation and map optimization, three stages. The traditional SLAM is difficult to

[1] School of Computer Science and Technology, Nanjing TECH University, Nanjing, 211816, China.

[2] School of Engineering, Kyushu Institute of Technology, Kyushu 804-8550, Japan.

[*] Corresponding Author: Zhenlong Du. Email: duzhl@njtech.edu.cn.

achieve high performance [Davison, Reid, Molton et al. (2007)], is not good at process camera with fast movement and severe jittering. The powerful chip occurrence improves SLAM performance, furthermore SLAM operates from the offline to online processing. The vision technology and the sensor promotion make the map construction more intuitive, especially the positioning in the previously unknown scene.

The paper presents an improved SLAM algorithm, which includes the feature point match acceleration based on KDtree, homography plane iterative estimation, and background process optimization for image prefetch, updation and expansion. The presented improved SLAM algorithm can handle camera with fast movement and rapid jittering, and fast reconstruct the prior unknown scene. Compared with the classical ORB-SLAM [Mur-Artal, Montiel and Tardos (2015)] and RGBD-SLAM [Kerl, Stuckler and Cremers (2015)], the improved SLAM algorithm could fast reconstruct the scene, optimize the camera trajectory according to the scene and camera posture, and achieve the lowest RMSE.

2 Related works

SLAM technique originally is applied to the autonomous robotics navigation, and it depends on the sensors such as laser range-finders and sonar for rapidly sensing the surrounding environment. Due to the camera holds the advantages of compact, accurate, noninvasive, cheap and ubiquitous, etc., the vision community has accumulated many achievements on structure-from-motion (SFM), recently sensor based SLAM has moved to the vision based SLAM.

LSD-SLAM based on monocular vision [Engel, Schps and Cremers (2014)] performs semi-dense mapping on large-scale scene, could construct the camera trajectory, and detect the scale drift when the scene changes significantly. The depth map can be constructed by iterative introducing the keyframe, and the good pixels are selected for modeling both the depth restoration and the depth map updating. LSD-SLAM achieves the consistent map via the constraint optimization. In large-scale environment, LSD-SLAM achieves the good semi-dense global consistency mapping, moreover it can run on CPU. Semi-direct visual odometry (SVO) [Forster, Pizzoli and Scaramuzza (2014)] directly on pixel intensities, estimates 3D points with the probabilistic mapping method that explicitly models outlier measurements, greatly eliminates the computation costs of feature point matching, can handle images at high rate acquisition.

Kalman filter is generally used for estimating the system state with maximum likelihood, it is employed for the scene point prediction in EKF-SLAM [Barrau and Bonnabel (2015)]. EKF-SLAM inevitability includes the error accumulation, when the current state prediction is beyond the threshold, the system could not achieve the real-time performance.

PTAM [Klein and Murray (2007)] is a keyframe-based monocular parallel SLAM algorithm, it adopts the two parallel threads, foreground threads mainly captures and matches the feature points and estimates the camera posture, while the background one mainly performs the map extension. FAST (features from accelerated segment testing) feature descriptor [Rosten, Porter and Drummond (2010)] is applied to extract the feature points within the region. The selected keyframes are cached in the keyframe queue, and the

mapping thread only extracts the feature points and reconstructs the 3D points from the keyframe queue. The camera tracking thread performs the feature points match, optimizes the camera posture of current frame according to the feature points correspondence.

3 Fast scene reconstruction via the improved SLAM

The improved SLAM adopts the parallel framework, the foreground thread manages the feature point match optimization and the local map expansion, the background thread performs the loop detection and improves the system efficiency. The improved SLAM algorithm includes the feature point match acceleration via KDtree, homography plane determination, and background thread optimization, mainly concentrates on the SLAM execution performance improvement.

3.1 Perspective transformation

3D point $P=[x_w, y_w, z_w, 1]^T$ is transformed to 2D point $[x_c, y_c, z_c, 1]^T$ by the acquisition device. Generally, operator takes the images with camera, mobile or Kinect. As Fig. 1 illustration, camera captures multiple 3D points $\mathbf{X}_p = \{P_1, P_2, P_3, \cdots\}$ within object, and the camera performs continuous acquisition from multiple angles, such as, camera postures $C_1, C_2, C_3, \cdots$. SLAM infers the camera position and posture from the successive images via multi-view geometry principle. The camera pose is composed of a 3×3 rotation matrix $\mathbf{R}_n$ and a translation vector $\mathbf{t}_n$. $P=[x_w, y_w, z_w, 1]^T$ is transformed from the world coordinate system to the local camera coordinate system as Eq. (1).

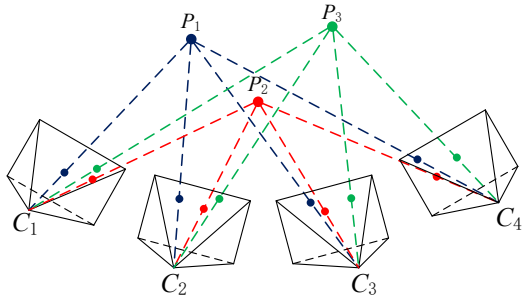


Figure 1: The camera takes object with multiple postures

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_n & \mathbf{t}_n \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \tag{1}$$

Eq. (1) is the homogeneous coordinate representation of perspective transformation. Eq. (2) is the nonhomogeneous coordinate representation of Eq. (1).

$$\begin{bmatrix} x \\ y \end{bmatrix} = \tau\left(\mathbf{K}\left(\mathbf{R}_i\left(\mathbf{X}_p - \mathbf{t}_i\right)\right)\right) \tag{2}$$

In which $\mathbf{K}$ is the camera parameter matrix, $\mathbf{R}_i$ is the rotation matrix at posture C_i, $\mathbf{t}_i$ is the camera translational vector at C_i. $\tau(\bullet)$ is a function as $[x/z, y/z] = \tau([x, y, z])^T$.

3.2 Feature points match acceleration

Points match [Gao, Xia, Zhang et al. (2018)] plays an important role in SLAM, it searches the matched points among images for determining the camera posture and predicting the map expansion. ORB (Oriented FAST and Rotated BRIEF) [Mur-Artal, Montiel and Tardos (2015)] feature descriptor bears the strong feature extraction and representation ability, it is applied in SLAM for the feature points match. SLAM need handle gigantic feature points and quickly find the matched feature points, then, the search strategy is crucial for SLAM. ORB-SLAM need artificially set the threshold for feature points match. If the threshold is set inappropriately, the number of matched points is readily influenced, reduces the matching accuracy. In the paper, KDtree is employed for accelerating the feature points match.

ORB-SLAM uses the brute force method for matching the feature points, as shown in Fig. 2, the computation costs is heavy and the real-time performance is difficult guaranteed. Inspired by the work [Forster, Carlone, Dellaert et al. (2017)], KDtree is exploited for improving SLAM execution efficiency. Additionally, for further improving the feature points match efficiency, region of interest (ROI) is utilized, it reduces the region with few feature points, as Fig. 3 depiction.

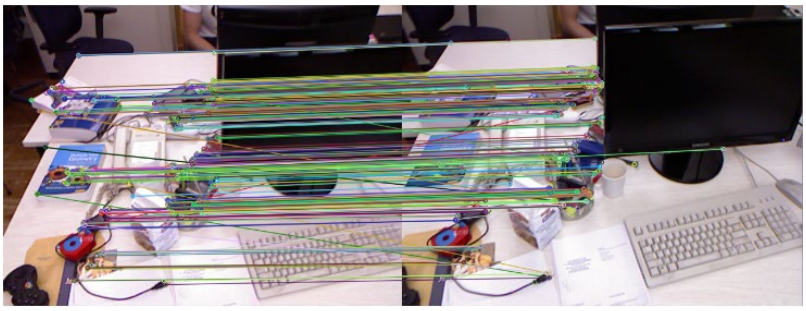


Figure 2: Conventional ORB-SLAM feature points match

KDtree includes the search tree building and the search speeding strategy. The search tree building establishes the search space based on the distance measurement on the feature points in image I_t and image I_{t+1}. Suppose m_i as the base point, KDtree searches the matched feature points under the measurement criteria. The search tree building constructs the candidate points for each feature point. KDtree has the special search speeding strategy, for any point m_i in I_t, it starts from the tree root node, firstly locates the starting branch based on the points similarity

measurement, then accesses the nodes of this branch for getting the mostly matched feature point. Meanwhile, backtracing is used to determine whether the branch holds the closer feature point. If the backtrace time is less than the threshold, the branch with the smallest distance is selected from the queue as points closer to m_i. The improved SLAM feature points correspondence procedure constructs matched feature point in I_{t+1} for any feature point in m_i in I_t.

Figure 3: Rich feature points region determination by ROI

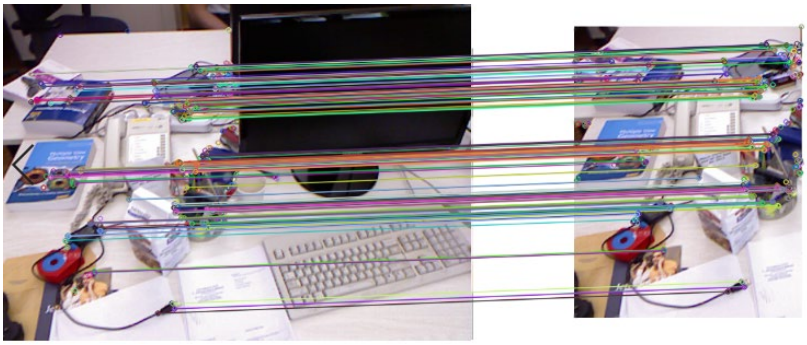


Figure 4: Feature points correspondence building by KDtree

Fig. 4 demonstrates that the improved feature points approach can build the feature points correspondence, and the used feature point number is smaller than the one of ORB-SLAM.

3.3 Homography plane determination

When feature points fall within the same plane or the parallax of two images is small, the camera posture is restored with aid of the homography plane. There exist some planar planes (such as tables, walls, etc.) in the indoor scenario.

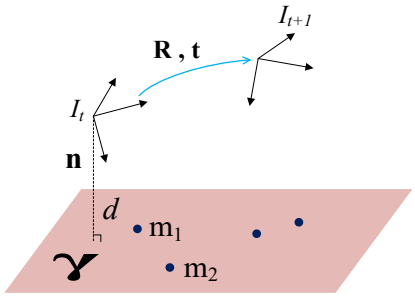


Figure 5: Homography plane

As the Fig. 5 showing, feature points $m_1 = (u_1, v_1, 1)^T$ and $m_2 = (u_2, v_2, 1)^T$ separately on the image I_t and I_{t+1} both fall within the plane γ, which follow the equation.

$$n^T \mathbf{X} + d = 0 \tag{3}$$

The plane is decided by $\gamma = \left[\mathbf{n}^T, d\right]^T$, where $\mathbf{n}$ is the unit normal vector in the camera coordinate system and d is the depth from the camera center to γ. Meanwhile, the transformation between feature points $m_1 = (u_1, v_1, 1)^T$ and $m_2 = (u_2, v_2, 1)^T$ separately on I_t and I_{t+1} is as the follow.

$$m_2 = \mathbf{K}\left(\mathbf{R} - \frac{\mathbf{t}\mathbf{n}^T}{d}\right)\mathbf{K}^{-1} m_1 \tag{4}$$

In which $\mathbf{K}$ is the camera intrinsic parameter matrix, $\mathbf{R}$ is the rotation matrix from I_t to I_{t+1}, $\mathbf{t}$ is the translation vector from I_t to I_{t+1}.

Assume the homography matrix $\mathbf{H}_{3\times 3}$ stands for $\mathbf{K}(\mathbf{R} - \frac{\mathbf{t}\mathbf{n}^T}{d})\mathbf{K}^{-1}$, then Eq. (4) has the following form.

$$\begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{V_{11}}{V_{33}} & \frac{V_{12}}{V_{33}} & \frac{V_{13}}{V_{33}} \\ \frac{V_{21}}{V_{33}} & \frac{V_{22}}{V_{33}} & \frac{V_{23}}{V_{33}} \\ \frac{V_{31}}{V_{33}} & \frac{V_{32}}{V_{33}} & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} \tag{5}$$

$u_2 = \frac{V_{11}u_1 + V_{12}v_1 + V_{13}}{V_{31}u_1 + V_{32}v_1 + 1}$ and $v_2 = \frac{V_{21}u_1 + V_{22}v_1 + V_{23}}{V_{31}u_1 + V_{32}v_1 + 1}$ can be derived from Eq. (5).

$$V_{11}u_1 + V_{12}v_1 + V_{13} - V_{31}u_1u_2 - V_{32}v_1u_2 - V_{33}u_2 = 0 \tag{6}$$

$$V_{21}u_1 + V_{22}v_1 + V_{23} - V_{31}u_1u_2 - V_{32}v_1v_2 - V_{33}v_2 = 0 \tag{7}$$

$\mathbf{H}$ is decided by Eq. (6) and Eq. (7). The improved SLAM exploits the homography feature tracking method for adapting the camera with strong rotation and fast movement.

Homography plane estimation is heavy computation procedure, furthermore the homography evaluation of any image to current one also bears the high computation. In the paper for improving SLAM efficiency, the keyframe F_k is served for the agent of

prefetch images, and the homography matrix between keyframe F_k and current image I_j is calculated, and it is expressed as the follow.

$$\mathbf{H}_{k\to j}^{P_L} = \mathbf{K}\left(\mathbf{R}_j\mathbf{R}_k^T + \frac{\mathbf{R}_j(\mathbf{t}_j - \mathbf{t}_k)\mathbf{n}_L^T\mathbf{R}_k^T}{d_L + \mathbf{n}_L^T\mathbf{R}_k\mathbf{t}_k}\right)\mathbf{K}^{-1} \tag{8}$$

In which $\mathbf{R}_j$ and $\mathbf{t}_j$ are separately the rotation matrix and translation vector of I_j, $\mathbf{H}_{k\to j}^{P} = \left\{\mathbf{H}_{k\to j}^{P_L} \mid L = 1,2,\cdots,N_{k\to j}^{P}\right\}$ represent the homography plane from F_k to I_j.

3.4 Background thread optimization

Background thread plays the important role in SLAM, it manages the region prefetch, updation and expansion. The traditional SLAM could generate a rather good result from the stable capture. For the inexperienced or novice operator sometimes manipulates SLAM, or the strong lens rotation and fast movement often occur, these captured data causes SLAM to lose keyframes or cannot achieve the matched feature points. At the same time, there exists some difference between the calculated feature point and the real point, the camera posture and the actual gesture. Latif et al. [Latif, Cadena and Neira (2013)] proposed a camera pose optimization method to correct the scale drift at the loop procedure. When the camera moves smoothly, a constant velocity motion model can be used to predict the camera pose location.

Object point P_j is projected to the pixel x_j in I_i under camera C_i, this perspective transformation is represented by $x_j = \boldsymbol{F}\left(C_i, P_j\right)$. In the paper, only the matched feature points are considered for being processed, thereafter x_i represents any feature point in any image I_i, it is the 2D point of P_j.

$$\arg\min_{C_1,\cdots,C_m,P_1,\cdots,P_n} \sum_{i=1}^{m}\sum_{j=1}^{n}\left\|\boldsymbol{F}(C_i,P_j) - \widetilde{m}_{ij}\right\|^2 \tag{9}$$

$\widetilde{m}_{ij}$ stands for all feature points to its scene positions the in all images, Eq. (9) attempts to achieve all feature points corresponding to its scene position as close as possible, it is employed for background thread optimization for scene reconstruction.

$$\arg\min_{C_j} \sum_{H(P_i,x_i)} \left\|\tau\left(K\left(R_j\left(P_i - p_j\right)\right)\right) - x_i\right\|_{\delta_h} \tag{10}$$

In which δ_h is the Huber loss function. Eq. (10) is optimized for scene prefetch by homography transformation.

The improved SLAM foreground thread calculates the local camera posture. If a certain amount of error is below a certain threshold, the prediction based on the prior information might cause the error accumulation. Although background thread optimization can maximize a posterior error, it does not well eliminate this kind of error.

4 Experiments

The improved SLAM algorithm proposed by the paper is implemented on the personal laptop with Intel(R) Core (TM)i5-6500 CPU@2.5 GHz, 8G RAM. The experiment deployment OS is 64-bit Ubuntu 16.04. The discussed algorithm runs online and handles the color images which are captured by the handhold Kinect within the indoor environment.

The routine hosted by the improved algorithm is robot operating system (ROS), which is open source code maintained by Open Source Robotics Foundation Inc. ROS is a flexible framework for developing robot related software, is a collection of cross-platform tools, libraries, and conventions that aim to simplify the task of handling complex and robust robot behavior. ROS execution threads cover the foreground and background threads, the foreground thread mainly captures and matches the feature points and estimates the camera posture through the homography tracking, while the background one mainly performs map extension, system loop detection and bundle adjustment (BA) [Vo, Narasimhan and Sheikh (2016)] optimization on the data obtained by the foreground thread.

The traditional SLAM prefers the gray images for the performance consideration and requires to input the gray images. Direct operating on color images brings on the more process data, requires the heavy computational cost, the interaction performance is influenced too. However, in the experiment the algorithm directly operates the color images, the entire data flow also is based on color images. Meanwhile the frame rate is 20 frames per second, the algorithm real-time performance is improved than the conventional SLAM.

In the paper the improved feature points match module is based on KDtree, it is used to rapidly match the feature points across frames via hierarchical manner with minimal matching error, greatly assures the real-time capability. Fig. 6 is the feature points match result by the improved SLAM algorithm.

Figure 6: Feature points obtained by the improved SLAM algorithm

For overall evaluating the algorithm performance, the videos involving rapid movement and strong rotation acquired by Kinect are testified by the experiment. The improved SLAM is able to process video with depth, as shown in Fig. 8, and the indoor scene is reconstructed with a sparse point cloud, and the red posture describes the keyframe location.

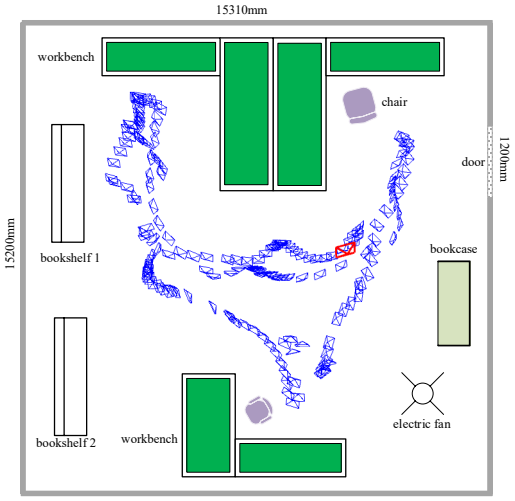
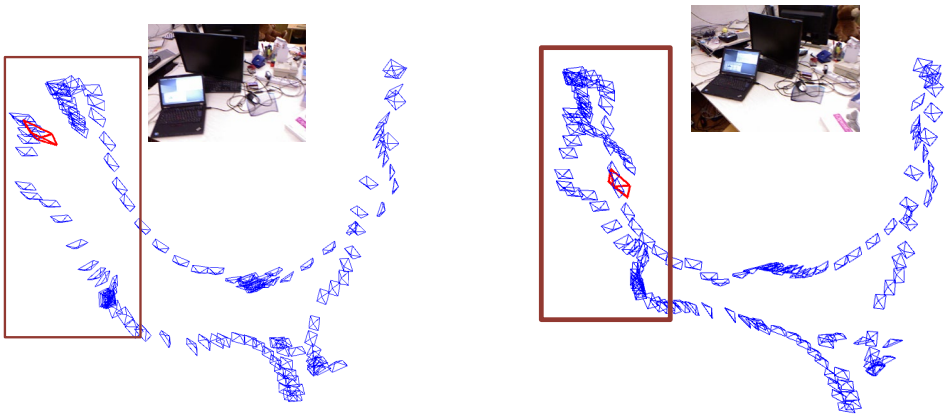


Figure 7: Scene layout



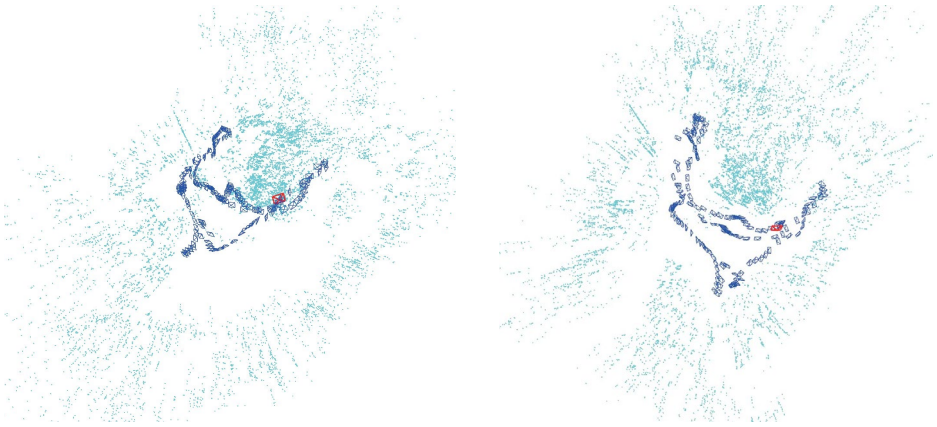
(a) Before optimization (b) After optimization

Figure 8: Camera trajectory optimization

Fig. 7 describes the experiment scene, which is a lab and includes the workbench, chair, bookcase, bookshelf and electric fan, the scene length is 15310 mm and the scene width is 15200 mm, the door is at the right wall and its width is 1200 mm. In this scene, all camera postures constitute the camera trajectory which is shown by blue sign, and the current camera posture is depicted by red symbol.

Within the same scene as Fig. 7, Fig. 8 shows the camera trajectory optimization result, Fig. 8(a) gives the camera trajectory without optimization, while Fig. 8(b) demonstrates the camera trajectory with optimization. From camera trajectory comparison within the two brown rectangles in Fig. 8(a) and Fig. 8(b), it observed that the camera trajectory without optimization is rough, while the camera trajectory with optimization is more compact.

Fig. 9 shows the reconstructed scene with 3D point cloud, Fig. 9(a) is the viewed from 45° view, and Fig. 9(b) is the viewed from right top. From two views of Fig. 9, it can be observed that the workbench, bookcase, bookshelf and chair are well reconstructed by the improved SLAM algorithm.



(a) from 45 view (b) top view

Figure 9: 3D point cloud of reconstructed scene

Four data sets, *Fr1/360*, *Fr1/floor* and *Fr1/desk* and one real-time *indoor* data Indoor downloaded from https://vision.in.tum.de/data/datasets/ are employed for evaluating the algorithm performance among *ORB-SLAM*, *RGBD-SLAM* and the improved SLAM by the paper. *RMSE* is used as the comparison measure in Tab. 1, it is observed that the improved SLAM approach achieves the lowest RMSE than *ORB-SLAM* and *RGBD-SLAM* in four datasets. Additionally, Tab. 1 shows that the proposed algorithm is more accurate than the original *ORB-SLAM* algorithm in positioning accuracy, it can fast restore depth map than *RGBD-SLAM* algorithm. The generated depth map by the improved SLAM algorithm is accurate and satisfies the real-time object insertion requirement, as Fig. 10 illustration.

Table 1: Algorithms performance comparison

Dataset	ORB-SLAM	RGBD-SLAM	Ours
Fr1/360	0.120	0.107	0.100
Fr1/floor	0.024	0.039	0.016
Fr1/desk	0.015	0.024	0.014
Indoor	0.066	0.089	0.048

(a) Cube insertion (b) Cube and arch introduction

Figure 10: Object real-time introduction

5 Conclusion

There exists monocular, stereo, RGB-D and ROS SLAM, these SLAM algorithms have been extensively investigated, and they can run on PC, mobile and robotics, three platforms. However, they still have the performance limitations, it is urgent for increasing SLAM real-time performance. With more types sensor involved by SLAM, more novel vision methods applied to SLAM, SLAM would be introduced and improved for handling more complicated scenario.

In the paper an improved SLAM algorithm is proposed in which KDtree is introduced for accelerating the feature points match, therefore the efficiency of depth map acquisition and the map reconstruction are improved. Moreover, background map expansion thread is optimized and SLAM performance is increased via parallel threads. Additionally, the improved SLAM method processes color videos, while the classical SLAM deals with gray videos.

With the big image/video emergence, such as, 4K, SLAM confronts to process much bigger images/videos, and its efficiency and performance improvement need to be investigated further.

Acknowledgement: This work is supported by the National Natural Science Foundation of China (Grant No. 61672279), Project of "Six Talents Peak" in Jiangsu (2012-WLW-023), and Open Foundation of State Key Laboratory of Hydrology-Water Resources and Hydraulic Engineering, Nanjing Hydraulic Research Institute, China (2016491411).

References

Barrau, A.; Bonnabel, S. (2015): Invariant filtering for pose EKF-SLAM aided by an IMU. *Proceedings of IEEE Conference on Decision and Control*, pp. 2133-2138.

Bresson, G.; Feraud, T.; Aufrere, R.; Checchin, P.; Chapuis, J. (2015): Real-time monocular slam with low memory requirements. *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 4, pp. 1827-1839.

Cui, Q.; McIntosh, S.; Sun, H. (2018): Identifying materials of photographic images and

photorealistic computer generated graphics based on deep CNNs. *Computers, Materials & Continua*, vol. 55, no. 2, pp. 229-241.

Davison, J.; Reid, D.; Molton, D.; Stasse, O. (2007): Monoslam: real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052-1067.

Engel, J.; Schps, T.; Cremers, D. (2014): LSD-SLAM: large-scale direct monocular SLAM. *Proceedings of 2014 European Conference on Computer Vision*, pp. 834-849.

Forster, C.; Carlone, L.; Dellaert, F.; Scaramuzza, D. (2017): On-manifold preintegration for real-time visual inertial odometry. *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1-21.

Forster, C.; Pizzoli, M.; Scaramuzza, D. (2014): SVO: fast semi-direct monocular visual odometry. *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 15-22.

Gao, Z. G.; Xia, S. X.; Zhang, Y. K.; Yao, R.; Zhao, J. Q. et al. (2018): Real-time visual tracking with compact shape and color feature. *Computers, Materials & Continua*, vol. 55, no. 3, pp. 509-521.

Kerl, C.; Stuckler, J.; Cremers, D. (2015): Dense continuous-time tracking and mapping with rolling shutter RGB-D cameras. *Proceedings of 2015 IEEE International Conference on Computer Vision*, pp. 2264-2272.

Klein, G.; Murray, D. (2007): Parallel tracking and mapping for small AR workspacess. *Proceedings of 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp. 1-10.

Latif, Y.; Cadena, C.; Neira, J. (2013): Robust loop closing over time for pose graph SLAM. *International Journal of Robotics Research*, vol. 32, no. 14, pp. 1611-1626.

Mur-Artal, R.; Montiel, J.; Tardos, J. (2015): ORB-SLAM: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147-1163.

Rosten, E.; Porter, R.; Drummond, T. (2010): Faster and better: a machine learning approach to corner detection. *IEEE Transactions on Robotics*, vol. 32, no. 1, pp. 105-119.

Vo, M.; Narasimhan S, G.; Sheikh, Y. (2016): Spatiotemporal bundle adjustment for dynamic 3D reconstruction. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1710-1718.

Zhang, G.; He, Y.; Chen, W.; Jia, J.; Bao, H. (2016): Multi-viewpoint panorama construction with wide-baseline images. *IEEE Transactions on Image Processing*, vol. 25, no. 7, pp. 3099-3111.

Zhang, G.; Liu, H.; Dong, Z.; Jia, J.; Wong, T. T. et al. (2016): Efficient non-consecutive feature tracking for robust structure-from-motion. *IEEE Transactions on Image Processing*, vol. 25, no. 12, pp. 5957-5970.

Zhou, T.; Lian, B.; Yang, S.; Zhang, Y.; Liu, Y. (2018): Improved GNSS cooperation positioning algorithm for indoor localization. *Computers, Materials & Continua*, vol. 56, no. 2, pp. 225-245.