A visual simultaneous localization and mapping approach based on scene segmentation and incremental optimization

Xiaoguo Zhang®, Qihan Liu, Bingqing Zheng, Huiqing Wang and Qing Wang

Abstract
Existing visual simultaneous localization and mapping (V-SLAM) algorithms are usually sensitive to the situation with sparse landmarks in the environment and large view transformation of camera motion, and they are liable to generate large pose errors that lead to track failures due to the decrease of the matching rate of feature points. Aiming at the above problems, this article proposes an improved V-SLAM method based on scene segmentation and incremental optimization strategy. In the front end, this article proposes a scene segmentation algorithm considering camera motion direction and angle. By segmenting the trajectory and adding camera motion direction to the tracking thread, an effective prediction model of camera motion in the scene with sparse landmarks and large view transformation is realized. In the back end, this article proposes an incremental optimization method combining segmentation information and an optimization method for tracking prediction model. By incrementally adding the state parameters and reusing the computed results, high-precision results of the camera trajectory and feature points are obtained with satisfactory computing speed. The performance of our algorithm is evaluated by two well-known datasets: TUM RGB-D and NYUDv2 RGB-D. The experimental results demonstrate that our method improves the computational efficiency by 10.2% compared with state-of-the-art V-SLAMs on the desktop platform and by 22.4% on the embedded platform, respectively. Meanwhile, the robustness of our method is better than that of ORB-SLAM2 on the TUM RGB-D dataset.

Keywords
Visual SLAM, large view transformation, scene segmentation, incremental optimization, bundle adjustment

Date received: 27 November 2019; accepted: 6 November 2020

Introduction
In recent years, the visual simultaneous localization and mapping (V-SLAM) technology has developed significantly. As a low-cost solution, V-SLAM technology has lower requirements on its hardware system. Therefore, it has preliminary applications in many fields, such as robot industry, virtual reality, and autonomous driving. However, the state-of-the-art V-SLAMs are usually sensitive to
the sparse landmarks in the environment and large view transformation of camera motion. Due to the decrease of the matching rate of feature points, the tracking thread often has a large drift or even causes failures on the loop detection and the back-end optimization process,\textsuperscript{9–11} which may finally result in instability or even collapse of the whole V-SLAM system.\textsuperscript{11}

Aiming at the above problems, this article proposes an improved V-SLAM method based on scene segmentation and incremental optimization. Different from those mainstream V-SLAM systems using all feature point information in the keyframe, this article selects the landmarks near the current line-of-sight vector to match with the last motion direction of the camera so as to improve the positioning accuracy. In the front-end module, by adding scene segmentation, camera motion direction to the tracking thread, and segmenting the camera trajectory, the effective prediction of camera motion in the road segments with sparse landmarks and large view transformation is realized. In the back-end optimization module, this article proposes an incremental optimization method combining camera motion direction and angle. By incrementally updating the state parameters and reusing the computed results, high-precision results of the camera track and feature points are obtained with satisfactory computing speed. The experimental results demonstrate that our method improves the computational efficiency by 10.2\% compared with state-of-the-art V-SLAMs on the desktop platform and by 22.4\% on the embedded platform, respectively. Meanwhile, the robustness of our method is better than ORB-SLAM\textsuperscript{23,4} on the TUM RGB-D dataset.\textsuperscript{1}

**Literature review**

Existing mainstream V-SLAMs could be roughly divided into three categories: based on filter, based on keyframe bundle adjustment (BA), and based on direct method.

Regarding the filter-based V-SLAM systems, Davison et al.\textsuperscript{12} proposed a most representative SLAM system named mono SLAM, in which the front end is able to track sparse map feature points and determine their poses in real time, and the back end uses an extended Kalman filter to update its mean and variance. To apply filtering-based SLAM in large scenes, Gu and Liu\textsuperscript{13} proposed a robust visual inertial odometry, and in its extended Kalman filter update process, the direct intensity error is used as the visual metric, however, such kinds of algorithms are only suitable for the environment with rich visual feature points. In general, filter-based V-SLAM system has poor performance in large scale and high dynamic scenes, which are likely to cause cumulative errors.\textsuperscript{14}

The first V-SLAM system based on keyframe BA, with its name parallel tracking and mapping (PTAM), was proposed by Klein and Murray.\textsuperscript{15} PTAM supports keyframe and back-end nonlinearly optimization, which uses a multi-thread strategy to track and map the scene, however, the main issue of PTAM is that its efficiency could be strongly affected in a medium range scenario with thousands of points. Currently, ORB-SLAM\textsuperscript{2} proposed by Mur-Artal et al.\textsuperscript{3,4} is one of the best keyframe-BA based V-SLAM systems, in which based on the improvement of PTAM, the ORB feature is used to track the surrounding environment, and loop-detection is established to eliminate cumulative errors. Similar to filter-based V-SLAM systems, keyframe-BA-based V-SLAM is dependent on the richness of environmental features, therefore, tracking failures and mapping accuracy errors often occur in the scene with sparse landmarks and large view transformation.

Instead of using keypoints, large-scale direct SLAM (LSD-SLAM), as the most-widely known direct method V-SLAM, directly operates on image intensities both for tracking and mapping, which are able to recover the semidense map in the scene.\textsuperscript{16} Engel et al.\textsuperscript{17} proposed direct sparse odometry (DSO) in 2017, which improves the scale of dense map restoration, and is more robust than LSD. The key problem of such kind of V-SLAMs is that they are usually sensitive to outdoor scene illumination changes and dynamic interference, so the robustness of location and mapping is not strong enough.\textsuperscript{13}

It is generally believed that the large view transformation of camera and sparse landmarks are two big technical challenges of V-SLAM. Morel and Yu\textsuperscript{18} proposed affine scale-invariant feature transform (ASIFT) algorithm in 2009. ASIFT uses affine sampling to simulate the angle deformation, which makes the algorithm to have the ability of anti-angle transformation and antiscale transformation, however, it could not provide real-time performance. Based on ASIFT, after surf (ASURF), after ORB (AORB), and AFREAK use SURF, ORB, and FREAK algorithms to replace SIFT algorithm, respectively, which are able to improve the operation speed.\textsuperscript{19} Han\textsuperscript{20} proposed a visual slam method to integrate the information of 3D structure, which uses the 3D point cloud structure to render virtual keyframes, expanding the scene coverage of matching view angle, as a result, the recall rate of loopback detection is improved in changing scenes.

In the view of the situation with sparse landmarks in the environment and large view transformation of camera motion, Pumarola et al.\textsuperscript{21} proposed a SLAM algorithm combining semantic information and line features, which sample the straight lines by matching the 3D points. However, the accuracy of the algorithm is not ideal and has high computational complexity since some of those 3D points are obtained using interpolation method.\textsuperscript{22} Zhang et al.\textsuperscript{23} proposed an RGB-D V-SLAM method based on line features, which use Mahalanobis distance to measure the distance from a 3D point to a 3D straight line and use random sample consensus (RANSAC) algorithm and maximum likelihood estimation to estimate the parameters of 3D lines so that it guarantees the accuracy of coordinate extracted in 3D line segments. Nerurkar et al.\textsuperscript{24} proposed C-KLAM based on the keyframe constraint. In the algorithm, during the back-end optimization, the sparse structure of the correlation information matrix is maintained through marginalization, and the
information matrix of nonkeyframes is projected into other adjacent keyframes to realize the optimization of camera pose and feature points. This method has strong robustness, but it relies on IMU or other inertial sensors to provide accurate camera pose output. Kaess et al. proposed iSAM2 based on incremental direct inference method with strong robustness for landmark points, which uses Bayes tree to solve the nonlinear optimization equation in an incremental way. Zhang et al. proposed a tracking method for discontinuous features through extracting the discontinuous SIFT features, in which K-means and matching matrices are used to cluster SIFT features, consequently feature points are segmented and fused. This method is suitable for large-scale scenes, however, its real-time performance is poor due to the large computing cost. Liu et al. proposed an RGB-D SLAM system using an incremental cluster adjustment algorithm, and it implements local and global BA in a unified optimization framework, which can save a lot of computing time in the back-end optimization of discontinuous segments and improve the processing speed. The disadvantage is that based on the direct method, the RGB-D SLAM system is sensitive to illumination changes and has the problems of feature point drift when there are few landmarks.

Overall speaking, V-SLAM technology has developed rapidly in recent years. However, when the V-SLAM technology is used in the situation of road segments with sparse landmarks or large view transformation, the number of matching feature points may be sharply reduced, which often leads to the problems of large drift or tracking failure in the V-SLAM system. These problems limit the further applications of V-SLAM.

Aiming at the aforementioned problem, this study proposes an improved V-SLAM method based on scene segmentation and incremental optimization strategy. The main contributions of this article are as follows:

1. In the front end, this article proposes a scene segmentation algorithm combined with camera motion direction and angle. By segmenting the camera trajectory and adding the camera motion direction to the tracking thread, an effective prediction model of camera motion in the scene with sparse landmarks and large view transformation is realized.
2. In the back end, this article proposes an incremental optimization method combining segmentation information and a prediction model optimization for unstable tracking state. By incrementally adding the state parameters and reusing the computed results, high-precision results of the camera trajectory and feature points are obtained with satisfactory computing speed.

**Framework**

The framework of the algorithm in this article is shown in Figure 1. Similar to ORB-SLAM2, our algorithm also includes three modules: a tracking thread, a local mapping thread, and a loop detection thread.
As shown in Figure 1, in the front end, a scene segmentation decision method is adopted. Then, the information of camera motion direction and angle is used to establish a prediction model, which can track the sparse landmarks more effectively. Consequently, more robust keyframes are selected by this algorithm. At the same time, in the back end, combined with the incremental optimization method, a prediction model of unstable tracking state and an optimization method of V-SLAM are adopted. By incrementally adding the state parameters to be optimized and reusing the existing calculation results, high-precision results of the camera trajectory and feature points are obtained with satisfactory computing speed.

**Tracking with landmarks and feature points.** Firstly, the uniform motion model\(^{24}\) of the visual camera is established

\[
x_{i+1} = F(x_i, u_i + w_i)
\]

where \(F\) is the motion equation of the camera between two frames, and \(x_i\) and \(x_{i+1}\) denote the camera poses at time steps \(i\) and \(i+1\), respectively. \(u_i\) is the measured control input, and in this article, it is related to camera motion direction. \(w_i\) is the zero-mean, white Gaussian measurement noise with covariance \(Q_i\).

At time step \(i\), the camera observation model is

\[
z_{ij} = H(x_i, f_j) + v_{ij}
\]

where \(z_{ij}\) denotes the camera’s observation of the landmark, \(f_j\) is the landmark’s coordinate, \(H\) is a general nonlinear measurement function, and \(v_{ij}\) is the zero-mean white Gaussian measurement noise with covariance \(Q'_{ij}\).

Suppose in the road segment \(\text{Seg}\), there are \(M\) landmark positions \(f_{j0}, j = 1, \ldots, M\), observed from these poses and \(N\) camera poses \(x_i, i = 0, 1, \ldots, N\). Then, the estimation \(\hat{x}_{0:N, f_{1:M}, \text{Seg}}\) of camera pose and landmark in segment \(\text{Seg}\) can be obtained by the maximize posterior probability estimates of all camera poses \(x_{0:N}\), all landmarks \(f_{1:M}\), and all available measurements \(Z_{0:N}, u_{0:N-1}\). It is given by

\[
\hat{x}_{0:N, f_{1:M}, \text{Seg}} = \arg\max_{x_{0:N}, f_{1:M}} p(x_{0:N}, f_{1:M}, Z_{0:N}, u_{0:N-1})
\]

where \(Z_i\) denotes a set of observation models for other landmarks at each camera pose \(x_i\). Combined with the above model, under the Gaussian and independent noise assumptions, the overall error model can be established as follows

\[
E(x_{0:N}, f_{1:M} | Z_{0:N}, u_{0:N-1}) = \sum_{i=0}^{N} \frac{1}{2} ||x_{i+1} - F(x_i, u_i)||_Q''^2
\]

\[
+ \sum_{z_{ij} \in Z_{0:N}} \frac{1}{2} ||z_{ij} - H(x_i, f_j)||_Q''^2 \triangleq \sum_{i=0}^{N} E_p(x_i, x_{i+1}; u_i) \quad (3)
\]

\[
+ \sum_{z_{ij} \in Z_{0:N}} E_M(x_i, f_j; z_{ij})
\]
Scene segmentation based on camera motion direction and angle. In the tracking thread of existing V-SLAMs, the large view transformation and the sparse road landmarks often lead to errors in the tracking thread, and finally, give rise to the drift of optimization iteration direction. Therefore, some researchers put forward a strategy based on scene segmentation optimization, which uses different clustering methods based on IMU and pose information to realize scene segmentation. It can not only reduce the error accumulation in the tracking thread but also improve the accuracy of BA optimization. Inspired by the above ideas, this article proposes a scene segmentation method combined with camera motion direction and angle.

The stable tracking algorithm in this article is basically the same as ORB-SLAM2, which uses the uniform motion model to predict the camera pose. Our algorithm is different from ORB-SLAM2 when the number of matching feature points sharply decreases. In this case, ORB-SLAM2 will compute the bag of words (BoW) vector of the current keyframes and find matched features based on the BoW dictionary. Instead, our algorithm does not use all the feature points information in the keyframes but uses the last camera motion direction and angle as constraints, at the same time, selects the feature points near the current line-of-sight vector to optimize the camera pose. The specific steps are as follows.

Firstly, camera positions $\hat{x}_i$ and $\hat{x}_{i+1}$ at time steps $i$ and $i+1$ are projected to a plane, respectively, to obtain $x'_i \sim K \hat{x}_i$, $x'_{i+1} \sim K \hat{x}_{i+1}$, where $K$ is the camera intrinsics matrix. Meanwhile, the number of tracking feature points $n_{\text{feature}}$ and the keyframes generation frequency $f_{\text{frame}}$ are determined in the tracking thread. By comparing them with weighted thresholds $(\overline{n_{\text{feature}}}, \overline{f_{\text{frame}}})$, which are average tracking feature points and keyframes generation frequency, respectively, segment computation will be executed to prevent the accumulation of local errors from affecting the optimization of localization and mapping results.

Secondly, in the segment decision phase, select the vector $\overrightarrow{x'_i x'_{i+1}}$ of the position of the projection plane at the current time and the position at the previous time within a certain confidence range. For the vectors $\overrightarrow{x'_i x'_{i+1}}$, arranged in time sequence, compute the included angle in the positive direction of the adjacent vectors. If the angle is larger than $\frac{\pi}{2}$ continuously, it indicates that there is an inflection point or tracking instability in the segment when the camera moves. At this time, use the K-means to cluster the direction vectors within the confidence range. Select the direction vectors $\overrightarrow{x'_i x'_{i+1}}$ and $\overrightarrow{x'_j x'_{j+1}}$ of two included angles $(\angle - \frac{\pi}{4}, \angle + \frac{\pi}{4})$ as the main direction of scene segmentation, and divide the scene into two segments, where $\angle$ is the average of vector angles within the confidence range. At the same time, use RANSAC algorithm to fit the rest of the location points into the two scene segments. The specific process can be represented by Algorithm 1.

**Prediction model combined with segmentation information.**

Based on the scene segmentation rule of “strict in and wide out,” after the scene segmentation, a reliable and efficient camera motion prediction model is needed in the case of discontinuous segments with sparse or even missing landmarks so as to select feature points as accurately as possible and maintain the tracking state until entering the stable road segments with dense landmarks.

After entering this kind of road segments, it is very important to select the appropriate road segments. Firstly, based on the direction constraint of the camera’s main motion angle in the segment, the landmarks which are close to the distance $d(\overrightarrow{f'_i x'_i x'_{i+1}}) \leq \overrightarrow{x'_i x'_{i+1}}$ of the direction are selected. After that, when tracking these road markings, the prediction model is as follows

$$\begin{align*}
x_i &= \overrightarrow{x_i x_{i+1}} \\
S_i &= S_e \times S_i \\
S_{i+1} &= S_e \times S_i 
\end{align*}$$

where $x_i$ is the predicted increment of camera pose $x_i$ in the main motion direction, $x_i(0.1,0.1,0.1)$, $S_e$ is the lie algebra representation of the predicted increment, $S_i, S_{i+1}$ is the lie algebra form of the absolute camera pose at $i,i+1$, respectively, and $S_{i+1}$ is the relative pose between time steps $i,i+1$. **Table 1. Configuration of the experimental platform.**

| Hardware       | Desktop                                      | HUAWEI-Hisilicon Kirin 970               |
|----------------|----------------------------------------------|------------------------------------------|
| CPU            | Intel Core i7-7700K(4.2 GHz)                 | ARM Cortex-A73 MPCore4, ARM Cortex-A53 MPCore4 |
| GPU            | NVIDIA GTX 1060 3 GB                         | ARM Mali-G72 MP12 GPU                    |
| Storage        | 521 GB                                       | 64 GB UFS 2.1                            |
| RAM            | 32 GB                                        | 4 GB LPDDR4X 1866MHz                     |
| OS             | Linux Ubuntu 16.04.                          | Linux Lubuntu (based on Ubuntu 16.04)    |
Through the prediction of camera motion, the calculation of the landmark \( f_{k_i+1} \) at the next moment can be obtained. Then, the error objective function of the prediction model can be established as follows:

\[
E_{k_i} = \sum_{i \in N} \left( \left\| f_{k_i}^t - f_{k_i}^b \right\| \right)^2 + \sum_{i \in N} \left\| d(f_{k_i}^t, \hat{x}_{i+1}) \right\|^2 \tag{5}
\]

where \( f_{k_i}^t \) denotes the calculated value of \( k \) landmark at time step \( i + 1 \), \( f_{k_i}^b \) denotes the actual observed value of \( k \) landmark at time step \( i + 1 \). \( d(f_{k_i}^t, \hat{x}_{i+1}) \) is the Euclidean distance between the selected landmark and the main direction of motion.

By minimizing the error objective function, we can effectively select landmarks to maintain the tracking state in the segments in the scene with large view transformation and sparse landmarks. The process will not end until the scene segments with normal tracking state are entered.

### Algorithm 1. Scene segmentation based on camera motion direction and angle

**Input**: Camera position at different times \( \{\hat{x}_i\} \)

- Number of feature points \( \{n_{\text{feature}}\} \), Frequency of keyframe generation \( \{f_{\text{frame}}\} \)
- Average tracking feature points \( n_{\text{feature}} \) and average keyframe generation frequency \( f_{\text{frame}} \)

**Output**: Scene segmentation main direction \( \hat{x}_i^t \hat{x}_{i+1}^b \) and \( \hat{x}_j^t \hat{x}_{j+1}^b \)

**Specific position of scene segment** \( \hat{x}_i, \hat{x}_j \)

**For** Camera position at anytime

Calculate the plane projection and get \( \hat{x}_i \sim K\hat{x}_i \)

**While** during tracking \( n_{\text{feature}} < 0.2n_{\text{feature}} \text{ and } f_{\text{frame}} < 0.1f_{\text{frame}} \) or \( n_{\text{feature}} > 1.1n_{\text{feature}} \text{ and } f_{\text{frame}} > 1.5f_{\text{frame}} \)

Select the time sequence arrangement vector \( \hat{x}_i^t \hat{x}_{i+1}^b \) of camera position projection plane within the confidence range \( \delta \)

Calculate the angle \( \{\theta_i\} \) in the positive direction of adjacent vectors

**If** \( \{\theta_i|\theta_i \in \theta\} > \pi \)

Using k-means clustering to cluster direction vectors in confidence range

Select the direction vectors \( \hat{x}_i^t \hat{x}_{i+1}^b \) and \( \hat{x}_j^t \hat{x}_{j+1}^b \) of two included angles \( \angle - \frac{\pi}{2} \angle + \frac{\pi}{2} \) as the main direction of scene segmentation

Use RANSAC algorithm to fit the rest of the location points into the two scene segments

**Else**

Remove \( \theta_i \) from the set \( \{\theta\} \)

**End If**

Through the prediction of camera motion, the calculation \( f_{k_i+1} \sim S_{i+1} f_{k_i}^t \) of landmark \( f_{k_i}^t \) at the next moment can be obtained. Then, the error objective function of the prediction model can be established as follows:

\[
\text{Incremental BA with fusion of segmentation information}
\]

In the back-end optimization process, ORB-SLAM2 uses covisibility graph, essential graph, and spanning tree. When inserting keyframes in the local mapping thread, keyframes with a certain number of covisibility landmarks will be established, and the spanning tree saves all nodes (i.e. keyframes) as a subset of the covisibility graph. After that, the loop closing thread will compute the similarity of the covisibility graph and optimize the camera pose according to the essential graph. This algorithm heavily relies on
covicibility landmarks to establish edge connections. At the same time, the setting and deletion strategies of redundant keyframes also increase computational complexity. The reduction of covicibility landmarks in scenes with large view transformation and a large number of invalid keyframes added frequently causes graph maintenance failure and tracking failure.

In this article, we still follow the tracking and optimization framework of ORB-SLAM2: (1) Optimize pose Euclidean transformation matrix in local BA (tracking thread) and (2) optimize pose similarity transformation matrix and global pose in global BA. But different from ORB_SLAM2, we use the factor graph to represent feature points and camera poses of keyframes. After that, we use the Bayes tree to run incremental optimization, which enables us to add the camera motion direction and angle factor to the optimization process, which consequently reduces both the dependence on covicibility landmarks and the computational complexity.

According to iSam and iSam2, the optimization process in SLAM can be expressed as a dynamic Bayesian network, and the motion and observation equations can express the conditional probability between state variables. As shown in equations (1) and (2), we add the observation equation and the direction of camera motion to the Bayes tree as factor graphs. In the Bayes tree, the nodes are classified by clique, and the k’th clique can be expressed as $C_k$. We define the conditional probability density $p(F_k|S_k)$ for each node. Among them, separator $S_k$ is the intersection of $C_k$ and its parent group $\Pi_k:C_k \cap \Pi_k$. Frontal variable $F_k$ is the set of remaining variables in $C_k$ except $S_k;F_k = C_k \setminus S_k$ and $(u_i, z_i) \in C_k$. Therefore, the joint probability density of the Bayes tree $p(X)$ can be expressed as

$$p(X) = \prod_k p(F_k|S_k)$$

In this article, an example of factor graph and Bayes tree is shown in Figure 3, where $f_i$ is the landmark, $x_i$ is the camera pose, and the segmented part $Seg_i \in x_{i+1}$ is based on the camera segmentation of camera motion direction $R = \begin{bmatrix} l_{1} & l_{2} & Seg_{1} & f_{1} & f_{2} & f_{3} \\ X & X & X & & & \\ X & X & X & & & \\ X & X & X & & & \\ X & X & & & & \\ \end{bmatrix}$

Then, we will introduce how to perform incremental optimization based on the matrix formed by the factor graph.

The incremental optimization algorithm is often used with IMU, so that the camera can obtain reliable pose information and improve the accuracy of the optimization algorithm. In this article, without IMU, we use an incremental BA method with fusion of segmentation information to optimize the camera pose, feature points, and the prediction model in the unstable state. Compared with the traditional method, it is able to improve the optimization speed and the robustness of the drawing thread.

In the case of incremental updating optimization, to optimize the new parameters, we update the nearest matrix to reuse the results of the existing measurements. A non-linear error optimization objective function can be obtained using a classical linearization method

$$\Delta^* = \text{argmin} \sum_i || A \Delta - b ||^2_2$$

where $\Delta$ is the state update vector, including camera pose, landmark position, predicted landmark, and predicted main motion direction in the prediction model; $A$ and $b$ represent the parameters of a sparse linear system with $\Delta$ as the solution. Through householder QR decomposition, the objective function is rewritten as

$$|| A \Delta - b ||^2_2 = || R \Delta - d ||^2_2 + c$$

where $c$ represents the sum of squares of the residuals of the least squares. When the new vectors need to be reoptimized and updated, the previous decomposition can be modified
The classical QR update is the result of adding a new state increment $\alpha^T, \beta$ to $[A|b]$. The classical QR update increases $\alpha^T$ in the former factor $R$ and a new scalar element $\beta$ in the right vector $d$. In this article, the state quantity $\alpha^T$ is sparse in the unstable tracking state, which is related to the motion direction, angle, and selected landmark of the scene segments. Then, we have

$$[A'|b'] = \begin{bmatrix} A & b \\ \alpha^T & \beta \end{bmatrix}$$

$$R_a = \begin{bmatrix} R \\ \alpha^T \end{bmatrix} = \begin{bmatrix} Q^T & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} A \\ \alpha^T \end{bmatrix} \quad (9)$$

After obtaining the state quantity update, we use Givens rotation to cancel the newly added row element on the left side and then change the new linear system into the upper triangle form in an incremental way. The specific method is shown in Figure 4

$$G \triangleq \begin{bmatrix} \cos \varphi & \sin \varphi \\ -\sin \varphi & \cos \varphi \end{bmatrix} \quad (10)$$

Therefore, the matrix of QR decomposition is multiplied by several Givens rotation matrices to the left and for the nonzero elements of the row state cell added by the increment, consequently, an upper triangular matrix containing the update factor $R'$ is generated from the left most nonzero element

$$G_{j_k} \cdots G_{j_2}G_{j_1}R_a = \begin{bmatrix} R' \\ 0 \end{bmatrix}$$

Similarly, using Givens rotation matrix to update the right state vector $d_a$ incrementally, we get $d'$. Then, the equivalent objective function after incrementally updating of the objective function is as follows
\[
\| R' \Delta - d' \|_2^2 = \| A \Delta - b \|_2^2 + \| \alpha \Delta - \beta \|_2^2 + c'
\]

where \( c' \) is the sum of residual squares after incremental updating.

In fact, there is still some work to do for the incremental optimization based on Bayes tree in GTSAM. Our work in this part is the same as iSam2, so it will not be repeated.

---

**Table 2.** RPE comparison of absolute trajectory error in the TUM RGB-D and NYUDv2 RGB-D benchmark.

|                      | ORB-SLAM2 | RKD-SLAM | ElasticFusion | OpenVSLAM | Our method |
|----------------------|-----------|-----------|---------------|------------|------------|
| fr1_desk             | 1.6 cm    | 2.5 cm    | 4.3 cm        | 6.9 cm     | 4.2 cm     |
| fr1_floor            | 117 cm    | 325.3 cm  | No final results\(^a\) | —         | 56.1 cm    |
| fr2_large_with_loop  | 210.5 cm  | 198.3 cm  | —             | —         | 95.5 cm    |
| fr2_xyz              | 0.4 cm    | 1.2 cm    | 1.1 cm        | 2.8 cm     | 1.5 cm     |
| fr3_nostructure_notexture_near_withloop | — | 2.9 cm | 1.6 cm | 1.8 cm | 1.8 cm |
| NYUDv2 lab1          | —         | 207.5 cm  | —             | 109 cm     | 75.9 cm    |

RPE: relative pose error.

\(^a\)No final results represents the failure of localization and mapping.
Figure 8. Running results of TUM datasets: (a) fr1_desk, fr1_floor based on ORB-SLAM2, (b) fr1_desk, fr1_floor based on this paper, (c) fr2_xyz, fr3_nostructure based on OpenVSLAM, and (d) f2_xyz, fr3_nostructure based on this article.
Through the incremental updating of error objective function, the incremental optimization results of camera pose and feature points can be obtained from the optimization of the prediction model in an unstable state so as to improve the optimization speed and the robustness of the mapping thread.

**Experiments**

By implementing the aforementioned algorithm, considering that ORB-SLAM2 uses g2o library based on graph optimization and OpenVSLAM\(^29\) reproduces it with high readability, we developed a V-SLAM system based on ORB-SLAM2 and OpenVSLAM. Its user interface is shown in Figure 5, where Figure 5(a) shows the mapping and tracking thread, and Figure 5(b) shows the mapping results.

**Test configuration**

To evaluate the performance of our algorithm, we choose TUM RGB-D dataset and NYUDv2 RGB-D dataset to run the comparative experiments. In the aforementioned datasets, scenes have sparse landmarks and large view transformation. The basic conditions of the selected datasets are as follows:

1. TUM fr1_desk and fr2_xyz are basic datasets with desktop and empty indoor space. The size of their environment space is \(5 \times 5 \times 0.5\) m\(^3\). And the size of datasets is 1–2 GB, and there are about 5000 pictures.
2. TUM fr1_floor and fr3_nostructure_notexture_near_withloop are scenes with sparse landmarks and large view transformation of camera motion. In the \(15 \times 5 \times 0.5\) m\(^3\) space, the floor and empty corridors are mainly included, and there are about 8000 pictures in the datasets. According to our experiments, ORB-SLAM, OpenVSLAM, and our

---

**Table 3. Configuration of the MYNT-eye RGB-D camera.**

| Feature                  | Value                                      |
|--------------------------|--------------------------------------------|
| Size                     | \(165 \times 31.5 \times 30.12\) mm\(^3\) |
| Frame rate               | Up to 60 fps                               |
| Resolution               | \(2560 \times 720; 1280 \times 480\)       |
| Depth resolution         | On chip \(1280 \times 720; 640 \times 480\) |
| Baseline                 | 120.0 mm                                   |
| Visual angle             | D:121° H:105° V:58°                       |
| Synchronization precision| <1 ms (up to 0.01 ms)                      |

---

![Figure 9. Visual-SLAM system based on Kirin 970 (a) HUWAEI-Hisilicon Kirin 970-HiAI, (b) Mynt-eye RGB-D camera, (c) Kirin 970 with Lebuntu system, and (d) hardware system based on Kirin 970. SLAM: simultaneous localization and mapping.](image-url)
method extract feature points within 200 in each frame of image in such scenes, so the probability of tracking failure and reinitialization is more than 70% in ORB-SLAM.

3. TUM fr2_large_with_loop and NYUDv2 lab1 are scenes with sparse landmarks in 10 × 50 × 0.5 m³ large-scale environment. The size of them is 4–5 GB and there are about 12,000 pictures. In these scenes, the number of feature points extracted in each frame of image is within 300, and about 20% of the images are with weak texture.

The hardware platform concludes with a desktop platform and an embedded development platform. The configurations of them are given in Table 1.

### Experimental results

We select the state-of-the-art V-SLAM systems, such as RKD-SLAM, ORB-SLAM2, ElasticFusion, OpenVSLAM, and LSD-SLAM, to compare with and evaluate our algorithm and system.

### Experimental results on desktop platform

As given in Table 2, the experimental results of relative pose error (RPE) between tum TUM RGB-D dataset and NYUDv2 RGB-D dataset are compared. It could be found that the average accuracy of our method is 51.7% higher than the average accuracy of the ORB-SLAM2 on the datasets fr1_floor and fr2_large_with_loop with serious landmark missing and large view transformation of the camera, and is 59.2%
higher than the RKD-SLAM in NYUDv2 lab1, respectively. As shown in Figure 6, the average processing time of frames in the dataset fr2_xyz is given. In our method, the average processing time of a single frame is 28 ms, which is 82.7% faster than ORB-SLAM2, while the absolute track accuracy is basically unchanged. Through the experiments of a large number of TUM RGB-D datasets, the average processing time of the keyframe of this algorithm is 10.2% higher than ORB-SLAM2 and its improved algorithm.

Figure 7(a) shows the scene segmentation result based on the RKD-SLAM on the dataset NYUDv2 lab1, and Figure 7(b) shows the scene segmentation result based on the method in this article, where the black track is the ground truth and the color track is the segmentation result. The absolute trajectory error of this algorithm is 62% higher than that of RKD-SLAM in this dataset. As shown in Figure 8, the left figure shows the results of the ORB-SLAM2 and OpenVSLAM in TUM datasets, and the right figure shows the results of the methods in this article in the sequences fr1_desk, fr1_floor, fr2_xyz, fr3_nostructure_notexture_near_withloop.

Experimental results on embedded development platform. To evaluate the performance of the proposed algorithm in the embedded development platform, we conducted a comparative experiment based on HUAWEI Kirin970. The configurations of the embedded platform and the camera are presented in Table 3. The V-SLAM system based on Kirin 970 is shown in Figure 9, and it could be found that our algorithm improves computational efficiency by 22.4% than LSD-SLAM and ORB-SLAM2, which is essential for the application of V-SLAM in embedded development platform.

As given in Table 4, the experimental results of the RPE and running time of the algorithm in this article with ORB-SLAM2, LSD-SLAM, and OpenVSLAM on the TUM dataset are listed. Experiments show that the average running speed of the method in this article is 19.5% higher than that of ORB-SLAM2 and 25.3% higher than that of RKD-SLAM.

At the same time, as shown in Figure 10, on the fr1_floor, fr2_large_with_loop, and fr3_nostructure_notexture datasets, our algorithm shows better tracking stability than both ORB-SLAM2 and LSD-SLAM. And in the actual test, we chose a 25 × 25 m² indoor scene. The scene is mainly empty
rooms and corridors. The number of feature points in each frame of image is not more than 300. Meanwhile, it is a three-story building with a height of 10 m that has large view transformation of camera motion in the process of going up and down. The closed loop error of our algorithm is 0.5 m, and the average trajectory error is 0.83 m. For comparison, the closed loop error of ORB-SLAM2 is 1.8 m, and the average track error is 2.1 m. As shown in Figure 11, our algorithm has better positioning accuracy and loopback accuracy than ORB-SLAM, and the tracking performance is more stable.

**Conclusion**

Considering the problem that existing V-SLAM systems are usually sensitive to the situation of sparse landmarks or large view transformation, this article proposes an improved V-SLAM method based on scene segmentation and incremental optimization strategy. And the experimental results demonstrate that our method improves the computational efficiency by 10.2% compared with state-of-the-art V-SLAMs on the desktop platform and by 22.4% on the embedded platform, respectively. Meanwhile, the robustness of our method is better than that of ORB-SLAM2 on the TUM RGB-D dataset. In general, our improved algorithm provides strong support for the effective application of V-SLAM under embedded platforms.

In future research, we plan to tightly integrate GNSS/BEIDOU information to further improve the SLAM accuracy in large scale and missing closed-loop scenarios.

**Acknowledgements**

The authors would like to acknowledge Jiangsu Overseas Visiting Scholar Program for University Prominent Young & Middle-aged Teachers and Presidents.

**Declaration of conflicting interests**

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

**Funding**

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by the National Natural Science Foundation of China [Grant Number: 62073078] and the Natural Science Foundation of Jiangsu Province of China [Grant Number: BK20160696].

**ORCID iD**

Xiaoguo Zhang https://orcid.org/0000-0001-9961-6443

**References**

1. Sturm J, Engelhard N, Endres F, et al. A benchmark for the evaluation of RGB-D SLAM systems. In: IEEE/RSJ international conference on intelligent robots and systems, Vilamoura, Algarve, 7–12 October 2012, pp. 573–580. Piscataway, NJ: IEEE.
2. Silberman N, Hoiem D, Kohli P, et al. Indoor segmentation and support inference from RGB images. In: European conference on computer vision, Berlin, Heidelberg, 2012, pp. 746–760. Berlin, German: Springer.
3. Mur-Artal R, Montiel JM, and Tardos JD. ORB-SLAM: a versatile and accurate monocular SLAM system. IEEE Trans Robot 2015; 31(5): 1147–1163.
4. Mur-Artal R and Tardos J D. ORB-SLAM2: an open-source slam system for monocular, stereo, and RGB-D cameras. IEEE Trans Robot 2017; 33(5): 1255–1262.
5. Hosseinzadeh M, Li K, Latif Y, et al. Real-time monocular object-model aware sparse SLAM. In: IEEE international conference on robotics and automation (ICRA), Montreal, Canada, 20–24 May 2019, pp. 7123–7129. Piscataway, NJ: IEEE.
6. Cadena C, Carlone L, Carrillo H, et al. Past, present, and future of simultaneous localization and mapping: toward the robust-perception age. IEEE Trans Robot 2016; 32(6): 1309–1332.
7. Song X, Wang P, Zhou D, et al. Apolloloc3d: a large 3d car instance understanding benchmark for autonomous driving. In: Proceedings of the IEEE conference on computer vision and pattern recognition, Long Beach, USA, 16–20 June 2019, pp. 5452–5462. Piscataway, NJ: IEEE.
8. Bowman S L, Atanasov N, Daniilidis K, et al. Probabilistic data association for semantic slam. In: IEEE international conference on robotics and automation (ICRA), Marina Bay Sands, Singapore, 29 May–3 June 2017, pp. 1722–1729. Piscataway, NJ: IEEE.
9. Lu Y and Song D. Robust RGB-D odometry using point and line features. In: Proceedings of the IEEE international conference on computer vision, Los Condes, Chile, 11–19 December 2015, pp. 3934–3942. Piscataway, NJ: IEEE.
10. Ila V, Porta JM, and Andrade-Cetto J. Information-based compact pose SLAM. IEEE Trans Robot 2009; 26(1): 78–93.
11. Forster C, Carlone L, Dellaert F, et al. On-manifold preintegration for real-time visual-inertial odometry. IEEE Trans Robot 2016; 33(1): 1–21.
12. Davison J, Reid ID, Molton ND, et al. MonoSLAM: real-time single camera SLAM. IEEE Trans Pattern Anal Mach Intell 2007; 29(6): 1052–1067.
13. Gu Z and Liu H. A survey of monocular simultaneous localization and mapping. CAAI Trans Intell Syst 2015; 10(4): 499–507.
14. Frost D, Prisacariu V, and Murray D. Recovering stable scale in monocular SLAM using object-supplemented bundle adjustment. IEEE Trans Robot 2018; 34(3): 736–747.
15. Klein G and Murray D. Parallel tracking and mapping for small AR workspaces. In: Proceedings of the 2007 6th IEEE and ACM international symposium on mixed and augmented reality, Nara, Japan, 13–16 November 2007, pp. 1–10. Piscataway, NJ: IEEE.
16. Engel J, Schöps T, and Cremers D. LSD-SLAM: large-scale direct monocular SLAM. In: European conference on
17. Engel J, Koltun V, and Cremers D. Direct sparse odometry. *IEEE Trans Pattern Anal* 2017; 40(3): 611–625.
18. Morel JM and Yu G. ASIFT: a new framework for fully affine invariant image comparison. *SLAM J Imag Sci* 2009; 2(2): 438–469.
19. Zeng QH, Chen Y, Wang YS, et al. A fast matching method for large viewpoint changes images based on ORB algorithm. *J Control Decision* 2017; 32: 2233–2239.
20. Han XS. A geometric information fused loop closing method for visual SLAM. *Inform Technol* 2018; 20: 135–138.
21. Pumarola A, Vakhitov A, Agudo A, et al. PL-SLAM: real-time monocular visual SLAM with points and lines. In: *IEEE international conference on robotics and automation (ICRA)*, Marina Bay Sands, Singapore, 29 May–3 June 2017, pp. 4503–4508. Piscataway, NJ: IEEE.
22. Korman S and Litman R. Latent RANSAC. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, Salt Lake City, USA, 18–23 June 2018, pp. 6693–6702. Piscataway, NJ: IEEE.
23. Zhang G, Lee JH, Lim J, et al. Building a 3-D line-based map using stereo SLAM. *IEEE Trans Robot* 2015; 31(6): 1364–1377.
24. Nerurkar ED, Wu KJ, and Roumeliotis SI. C-KLAM: constrained keyframe-based localization and mapping. In: *IEEE international conference on robotics and automation (ICRA)*, Hong Kong, China, 31 May–5 June 2014, pp. 3638–3643. Piscataway, NJ: IEEE.
25. Kaess M, Johannsson H, Roberts R, et al. iSAM2: incremental smoothing and mapping using the Bayes tree. *Int J Robot Res* 2012; 31(2): 216–235.
26. Zhang G, Liu H, Dong Z, et al. Efficient non-consecutive feature tracking for robust structure-from-motion. *IEEE Trans Image Process* 2016; 25(12): 5957–5970.
27. Liu H. Robust and efficient 3D registration and structure recovery for challenging environment. PhD Thesis, Zhejiang University, CN, 2017.
28. Negre PL, Bonin-Font F, and Oliver G. Cluster-based loop closing detection for underwater slam in feature-poor regions. In: *IEEE international conference on robotics and automation (ICRA)*, Stockholm, Sweden, 16–21 May 2016, pp. 2589–2595. Piscataway, NJ: IEEE.
29. Sumikura S, Shibuya M, and Sakurada K. OpenVSLAM: a versatile visual SLAM framework. In: *Proceedings of the 27th ACM international conference on multimedia*, Nice, France, 21–25 October 2019, pp. 2292–2295. New York, NY: ACM.
30. Whelan T, Leutenegger S, Salas-Moreno R, et al. Elastic Fusion: dense SLAM without a pose graph. In: *Robotics: science and systems*, Rome, Italy, 13–17 July 2015, pp. 301–330. Thousand Oaks, CA: Sage Publishing.