FSD-SLAM: a fast semi-direct SLAM algorithm

Xiang Dong¹ · Long Cheng^{2,3} · Hu Peng¹ · Teng Li¹

Received: 31 December 2020 / Accepted: 1 March 2021 / Published online: 15 March 2021
© The Author(s) 2021

Abstract

Current visual-based simultaneous localization and mapping (SLAM) system suffers from feature loss caused by fast motion and unstructured scene in complex environments. Addressing this problem, a fast semi-direct SLAM algorithm is proposed in this paper. The main idea of this method is to combine the feature point method with the direct method in order to improve the robustness of the system in the environment of scarce visual features and low texture. First, the feature enhancement module based on subgraph is developed to extract image feature points more stably. Second, an apparent shape weighted fusion method is proposed for camera pose estimation, which can still work robustly in the absence of feature points. Third, an incremental dynamic covariance scaling algorithm is studied for optimizing the error of camera pose estimation. Finally, based on the optimized camera pose, a face element model is designed to estimate and fuse the point cloud pose, and obtain an ideal three-dimensional point cloud map. The proposed algorithm has been tested extensively on the benchmark TUM dataset and the real environment. The results show that the algorithm has better performance than existing visual based SLAM algorithms.

Keywords SLAM · Feature enhancement · Pose estimation · Incremental dynamic covariance scaling · Point cloud integration

Introduction

Simultaneous localization and mapping (SLAM) is a key technology for mobile robots used in detection, navigation, and rescue in unknown environments. The robot uses the information acquired by the sensors to reconstruct the environment, and estimates the real-time position of itself according to the reconstructed part. With the development of depth cameras, many studies on visual SLAM (VSLAM) systems adopt RGB-D cameras as sensors [1].

Current VSLAM methods mainly have two directions: the feature point methods based on feature match and the direct method without the need of feature points [2–10]. Due to

the need to extract feature points and calculate descriptors, feature method is not only time-consuming, but also cannot be applied to the environment with low texture and other features missing. Compared with the feature method, the direct method estimates the camera pose directly according to the brightness difference of pixels. Since there is no need to calculate feature points, the direct method runs very fast. However, the direct method is so sensitive to the camera's internal parameters and exposure that the camera's pose is easily lost when the camera moves quickly, resulting in the direct method is less accurate than the feature point method in the camera's pose estimation.

To address the above-mentioned problems, this paper proposes a fast semi-direct SLAM algorithm (FSD-SLAM). The contributions are as follows:

1. A feature enhancement module is designed for stable feature extraction. With this module, the subgraph which is the image with feature enhancement is used to improve the efficiency of feature extraction.
2. An apparent shape weighted fusion method is proposed for accurate camera pose estimation. Based on the apparent shape weighted fusion method, a camera pose estimation algorithm is designed to achieve accurate cam-

✉ Long Cheng
long.cheng@ia.ac.cn

✉ Teng Li
liteng@ahu.edu.cn

¹ School of Electrical Engineering and Automation, Anhui University, Hefei 230601, China

² State Key laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China

³ School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China

era pose estimation in hard scenes with fast motion, no structure, and missing features.

3. An incremental dynamic covariance scaling algorithm is designed for efficient global optimization. Addressing the problems of high computational load of traditional back-end optimization algorithms, the incremental dynamic covariance scaling algorithm is designed for global optimization, which improves the efficiency of the optimization.
4. A face element model is proposed for robust point cloud fusion. In the process of map construction, there are often staggered layers during the superposition process, which cannot be perfectly superimposed. To solve this problem, the surface element model is studied for point cloud fusion to improve the success rate of the map construction.

The rest of this paper is organized as follows. In the second section discusses the related works. In the third section introduces the proposed SLAM algorithm in detail. In the fourth section analyzes the results of algorithm evaluation. Finally, in the Conclusion section, we give the conclusion and the potential application scenarios.

The related works

In the field of SLAM, we summarize the related works in three directions: feature point-based method, direct method, and hybrid method.

The feature point-based method

Henry et al. [11] at the University of Washington first used RGB-D camera to implement SLAM algorithm. They used the SIFT algorithm [12,13] to extract features, and used known depth data to generate 3D–3D feature point pairs, and adopted the ICP algorithm to iteratively solve the optimal camera pose. Endres et al. [14] implemented the RGB-D SLAM system. The system uses the SURF algorithm [13,15] for feature point detection and descriptor calculation, and uses an improved ICP method to estimate the camera pose. In 2014, Artaf and Tardos et al. [16] designed ORB-SLAM system. Later, on the basis of ORB-SLAM, ORB-SLAM2 [17] was proposed. ORB-SLAM absorbed the advantages of PTAM [18] and adopted ORB Algorithm [19] to extract feature points and calculate descriptors. It is a typical representative of SLAM system based on feature point method. However, ORB-SLAM also has obvious shortcomings, such as a large amount of calculation, the effect of the reconstructed map is not ideal when facing the missing area of the feature point, and so on.

The direct method

Engel et al. [20] proposed LSD-SLAM through baseline stereo registration between a large number of pixel pairs, which was performed for pixels and realized the reconstruction of a semi-dense map on CPU. However, in the phase of camera pose optimization, it is still necessary to use feature points for closed-loop detection, so this method does not completely get rid of the dependence on feature points. In terms of accuracy, LSD-SLAM still has a big gap compared to SLAM methods based on feature points such as RGB-D SLAM and ORB-SLAM.

The hybrid method

Forster et al. proposed a semi-direct visual odometer named SVO [21]. In SVO, the author uses a direct method to track sparse feature points to complete the camera motion estimation. The high speed of SVO is its biggest advantage, which is far ahead of other SLAM systems. However, SVO is designed for the UAV's downward-looking camera, which cannot adapt to complex robot movements, and has high requirements on the environment. Moreover, SVO adopts monocular camera, which cannot recover the real pose information and scene scale. In addition, SVO is a visual odometer, which cannot achieve the closed-loop function to improve the consistency of positioning and mapping. The work done by Krombach et al. [22] also combined the direct method SLAM (LSD SLAM) and the feature method VO (LIB-VISO2) [23], and use the feature method to obtain camera pose initialization of the direct method. This algorithm can obtain semi-dense depth map, but the accuracy of the algorithm's visual odometer needs to be improved.

Although the previous proposed VSLAM algorithms succeeded, they may fail in hard environments. In our method, using the feature accumulation module combined with the feature point information and photometric information of the image, the stable and accurate camera pose estimation can be achieved in various complex scenes. More importantly, our system has improved the camera pose optimization and the entire map reconstruction process to achieve better performance.

The proposed fast semi-direct SLAM

The main idea of the FSD-SLAM system is to combine the feature method and the direct method at the same time. The system framework is shown in Fig. 1. There are three main stages: initialization and feature enhancement, pose estimation and optimization, and 3D mapping. The following is a detailed description of the three stages divided into four small parts.

Fig. 1 The overall framework of the proposed method

Fig. 2 The concept of feature enhancement module based on subgraphs

Initialization and feature enhancement

First, the entire system is initialized. With the input of video frames, the image features of two adjacent frames are calculated and matched. When there are enough feature matching points, the depth of the map point associated with the feature and the 3D coordinates of the map point are solved to complete the initialization.

Second, it is a challenge for VSLAM system to extract the key points of motion blur and textureless environment. FAST key points [24] belong to a kind of feature points that are particularly fast in calculation. The feature enhancement module based on subgraph can enhance features of FAST key points. According to the reliability of robot ranging, it can generate subgraphs through the enhancement of three-dimensional feature points, which is convenient for the system to extract image feature points stably. After the initialization of the system, the video frames are put into the fixed timeline Q_t queue. The feature enhancement module of

the subgraph acts on the Q_t header, outputs the RGB frame with FAST feature point enhancement, and updates the output results back to the timeline Q_t queue. Figure 2 shows how this module works in an inadequate feature environment. It is realized by enhancing three-dimensional feature points in an environment with insufficient features. The enhanced 3D point l_i of the feature L in the i th subgraph is shown as follows:

$$L_{l_i} = \sum_{j=v \in l_i}^{m \in l_i} H_0^{-1} H_j F_D^G v_j. \quad (1)$$

Here, v_j represents the 3D feature point in the j th node, F_D^G is the coordinate transformation from the camera to the robot, and H_j represents the transformation expression of the j th ranging. v and m represent the first and last node index of the subgraph, respectively.

Camera pose estimation based on weighted fusion

To calculate the camera pose more accurately, this paper proposes to fuse the reprojection information of the matching feature point pairs and the corresponding pixel photometric information to estimate the camera pose. This method can overcome the problem that the feature point method is difficult to accurately solve the camera pose in the environment of lacking texture feature information. High-quality camera pose estimation can directly improve the real-time and accuracy of whole SLAM system.

Fig. 3 Reprojection error of feature points

The camera pose estimation method based on the reprojection errors of feature point pair is shown in Fig. 3 below.

Among them, c_{13} is the new camera pose estimated by the motion model, c_{12} is the actual camera positions, $\mathbf{R}$ and $\mathbf{t}$, respectively, represent the rotation and translation matrix of the current frame relative to the previous frame. This pixel coordinates of the spatial point P_1 on the two frames of images are p_{11} and p_{13} , respectively. If the camera pose is accurate, the real positions p_{12} and p_{13} should be the same point. If the camera pose is not correct, there will be a reprojection error e between the projection points p_{13} and p_{12} . c_{13} is used to reproject the visible map points of the previous frame onto the new image, data association is carried out in a search window s_w around the projection points, and then the reprojection error e is minimized by using euclidean transformation parameters to calculate the camera pose $\mathbf{T}$ to be estimated. Without loss of generality, consider any spatial point P_{1i} , e is defined as follows:

$$e = p_{1i} - \frac{1}{Z_{1i}} \mathbf{K} \mathbf{T} P_{1i}. \quad (2)$$

Considering all the feature points, the total reprojection error is as follows:

$$E = \frac{1}{2} \sum_{i=1}^N \left\| p_{1i} - \frac{1}{Z_{1i}} \mathbf{K} \mathbf{T} P_{1i} \right\|^2. \quad (3)$$

In Eqs. (2) and (3), p_{1i} is the two-dimensional pixel point corresponding to the spatial point P_{1i} on the second frame image; $\mathbf{T}$ represents camera pose to be estimated; $\mathbf{K}$ represents the camera's internal parameters; Z_{1i} represents the coordinates of the relative Z axis after the projection of P_{1i} onto the camera frame; N represent the sum of feature matches.

Fig. 4 Photometric error

To strengthen the optimization effect of the corresponding feature point errors, the photometric errors of matching feature point pairs are considered on the basis of reprojection error optimization. The camera pose is optimized and calculated by fusing optimized reprojection and photometric errors. The photometric error is shown in Fig. 4.

Suppose that the spatial point corresponding to a pixel p_{21} in the previous frame image is P_2 , and the pixel position after projecting onto the second frame image using the camera pose is p_{22} . If the camera pose is not accurate, there is an error in the photometric between the two positions. By optimizing iterative solution, the photometric error between them can be continuously reduced, and finally the optimal camera pose can be obtained. According to the above description, the photometric error of a pixel can be defined as:

$$\hat{e} = I_1(p_{2j}) - I_2 \left(\frac{1}{Z_{2j}} \mathbf{K} \mathbf{T} P_{2j} \right). \quad (4)$$

Then, the sum of the photometric error functions of multiple pixel pairs is:

$$\hat{E} = \frac{1}{2} \sum_{j=1}^W \left\| I_1(p_{2j}) - I_2 \left(\frac{1}{Z_{2j}} \mathbf{K} \mathbf{T} P_{2j} \right) \right\|. \quad (5)$$

In Eqs. (4) and (5), p_{2j} represents the pixel coordinates in the first frame; P_{2j} represents the spatial point coordinate of the point p_{2j} ; Z_{2j} represents the coordinates of the relative Z axis after the projection of P_{2j} onto the camera frame; $I_1()$ and $I_2()$ represent the photometric information of the pixels corresponding to the previous frame image and the current frame image; W represents the total number of pixels.

By combining the reprojection error of matching feature points with the photometric error of pixel pairs, the camera pose estimation value can be solved as follows:

$$T^* = \arg \min_T \left(\sum_{i=1}^N \left\| p_i - \frac{1}{Z_i} K T P_i \right\|^2 + \lambda \sum_{j=1}^W \left\| I_1(p_j) - I_2 \left(\frac{1}{Z_j} K T P_j \right) \right\|^2 \right). \quad (6)$$

In the formula, λ is a weight coefficient, which is used to linearly adjust the role of reprojection error and photometric error in camera pose optimization. This method improves the camera pose optimization effect by fusing and optimizing the matching errors of the sampled data due to changes in the baseline and illumination conditions, respectively. Among them, the weight λ takes the empirical value of 0.5.

Global optimization of incremental dynamic covariance scaling algorithm

The system estimates the camera pose from the images between adjacent frames, and connects the poses of all adjacent moments to form the robot's motion trajectory. Since there will be a certain error in the estimation of camera pose every time, the cumulative error will be formed with the increase of the input image. To address this problem, two techniques are required: close loop detection and global optimization. The close loop detection adopts a fast detection method based on visual word bags to judge whether the robot has been to this place. To obtain consistent trajectories and maps, the global optimization needs to optimize the close loop detection information and the error of camera pose estimation at different moments.

The representative global optimization algorithm is SC (Switchable constraint) [25] algorithm. This algorithm proposed in the paper is an improvement to SC algorithm. This SC algorithm expresses the goal function of the SLAM optimization problem as follows:

$$Y^*, G^* = \arg \min_{Y, G} \underbrace{\sum_i \|y(a_i, a_{i+1}) - x_{i,i+1}\|_{\Sigma_i}^2}_{\text{pose estimation constraints}} + \underbrace{\sum_{ij} \|\Psi(g_{ij})(y(a_i, a_j) - x_{ij})\|_{\Lambda_{ij}}^2}_{\text{closed-loop constraints with added switching variables}} + \underbrace{\sum_{ij} \|1 - g_{ij}\|_{\Xi_{ij}}^2}_{\text{switching variable constraints}}. \quad (7)$$

Equation (7) shows that the solution of SLAM optimization problem includes node state variable Y^* and switch variable G^* . The right side of the equation is decomposed into three parts, which are sequentially expressed as: pose estimation constraints, closed-loop constraints with added switching

variables, and switching variable constraints. Among them, g_{ij} is the switch variable, $\Psi(g_{ij})$ is the zoom function, Ξ_{ij} is a switch prior, a_i is the robot pose at variable i , x_{ij} represents the conversion between camera poses at adjacent moments, and $y(a_i, a_j)$ is the measurement value function that predicts between a_i and a_j . The covariance of odometer and sensors measurement is Σ and Λ .

It can be seen from Eq. (7) that the SC algorithm needs to solve the joint optimization of the switch variable, pose and closed loop at the same time, so the calculation amount is large and the real time is not strong. Based on these problems, we improve it.

To get a better solution, we consider the adjacent node v and u and the edges between them. The error can be divided into two parts: including the edge except vu and considering only the edges containing vu . The objective function can be redefined:

$$Y^*, G^* = \arg \min \sum_i \|f(a_i, a_{i+1}) - x_{i,i+1}\|_{\Sigma_i}^2 + \sum_{ij \neq vu} \|g_{ij}(y(a_i, a_j) - x_{ij})\|_{\Lambda_{ij}}^2 + \sum_{ij \neq vu} \|1 - g_{ij}\|_{\Xi_{ij}}^2 + \|1 - g_{vu}\|_{\Xi_{vu}}^2 + \|g_{vu}(f(a_v, a_u) - x_{vu})\|_{\Lambda_{vu}}^2. \quad (8)$$

Set $q = Y^*, G^*$, and arrange Eq. (8) to get:

$$q = \arg \min_{Y, G} \underbrace{s(Y_{ij \neq uv}, G_{ij \neq uv})}_{s(\cdot)} + \underbrace{g_{uv}^2 \chi_{uv}^2 + (1 - g_{uv})^2 \Phi}_{h(\cdot)}. \quad (9)$$

Here, $s(\cdot)$ represents all the errors except those containing edge uv , and $h(\cdot)$ means the errors containing edges uv . When optimizers converge to a certain degree, it takes partial derivative of each variable in the entire function, and the result is zero. Therefore, the partial derivative of the switch variable g_{uv} in Eq. (9) can be obtained as follows:

$$\nabla q = \left[\frac{\partial q}{\partial g} \right] = \left[2g\chi_v^2 - 2(1-g)\Phi \right] = [0]. \quad (10)$$

Using the equality of both sides of the equation, formula (10) can be simplified as follows:

$$g = \frac{\Phi}{\chi_v^2 + \Phi}. \quad (11)$$

By substituting the obtained g into $h(\cdot)$, we can get:

$$\hat{h} = \frac{\Phi^2 \chi_v^2}{(\chi_v^2 + \Phi)} + \Phi - \frac{2\Phi^2}{\chi_v^2 + \Phi} + \frac{\Phi^3}{(\chi_v^2 + \Phi)^2}, \quad (12)$$

where $\hat{h}$ is the value obtained when the derivative of the function h is taken as 0, and finding the maximum value

of this function is equivalent to obtaining the upper limit of χ_v^2 , so as to obtain all possible solutions calculated by the optimization program. The derivative with respect to $\hat{h}$ is:

$$\frac{d\hat{h}}{d\chi_v} = \frac{2\chi_v\Phi^2}{(\Phi + \chi_v^2)^2}. \quad (13)$$

It can be seen from Eq. (13) that when χ_v takes 0, the derivative is equal to 0, and the function h takes the minimum value at this time. When χ_v takes 0 or tends to positive and negative infinity:

$$\lim_{\chi_v \rightarrow \pm\infty} \hat{h} = \Phi + 0 - 0 + 0 = \Phi; \quad \hat{h}_{\chi_v=0} = \Phi - 2\Phi + \Phi + 0 = 0. \quad (14)$$

Then, it is easy to know $0 \leq h \leq \Phi$. We need to consider all the edges between nodes u and v , that is to say, for all switching variables, the constraint $0 \leq h \leq \Phi$ is satisfied. So there are:

$$g(g(\Phi + \chi_v^2) - 2\Phi) \leq 0. \quad (15)$$

From equation (15), we can get:

$$0 \leq g \leq \frac{2\Phi}{\Phi + \chi_v^2}. \quad (16)$$

At this point, the optimal value range of the switching variable is found. In this way, we not only retain the advantages of the original SC algorithm, but also effectively reduce the value range of the switch variable and increase the accuracy of global optimization.

3D mapping

After obtaining the high-precision camera pose through the optimized processing, the point cloud data are obtained by back-projecting each image pixel into a three-dimensional space, thereby performing point cloud fusion to obtain a high precision dense three-dimensional map.

In this paper, the surface element model is used as the minimum processing unit. For each surface element, the location information (X, Y, Z) , vertex vector $\mathbf{n}$, color informations (R, G, B) , surface element radius r , weight w , and point acquisition time information are t are stored. The global optimization is achieved by constructing the deformation graph. The deformation graph is composed of many sides and nodes. Each node $\mathbf{g}^n$ stores position information $\mathbf{g}_g$, rotation matrix $\mathbf{g}_R$, translation matrix $\mathbf{g}_t$ and time information $\mathbf{g}_{t_0}$. The rotation matrix and translation matrix are parameters to be optimized.

Any three-dimensional structure in the space can be composed of a surface element model, and any surface element $\mathbf{M}^S$ is controlled by the node in deformed map $C(\mathbf{M}^S, \mathbf{g})$. When the new positions parameter are obtained by node optimization, it will affect the surface element, including the change of the normal vector and the position change of the surface element. After the node is optimized, the affected surface element location becomes:

$$\hat{\mathbf{M}}_P^S = \sum_{n \in I(\mathbf{M}^S, \mathbf{g})} w^n(\mathbf{M}^S) [\mathbf{g}_R^n(\mathbf{M}_P^S - \mathbf{g}_g^n) + \mathbf{g}_g^n + \mathbf{g}_t^n]. \quad (17)$$

Among them, $\mathbf{M}_P^S$ represents the coordinates before deformation, $\hat{\mathbf{M}}_P^S$ represents the coordinates after surface element position deformation, and $\mathbf{g}_R^n$ and $\mathbf{g}_t^n$ are the positions and postures after node optimization. The surface element normal vector becomes:

$$\hat{\mathbf{M}}_n^S = \sum_{n \in I(\mathbf{M}^S)} w^n(\mathbf{M}^S) \mathbf{g}_R^{n-1} \mathbf{U} \mathbf{M}_n^S. \quad (18)$$

Among them, $\mathbf{M}_n^S$ represents the normal vector of the surface element before deformation, $\hat{\mathbf{M}}_n^S$ represents the normal vector of the surface element after deformation, and $\mathbf{U}$ represents the transformation relationship between the surface element before and after deformation. $w^n(\mathbf{M}_n^S)$ represents the weight of node $\mathbf{g}^n$'s influence on the surface element model, and the weights of the node that have influence on the current face are added together to 1:

$$w^n(\mathbf{M}^S) = 1 - \frac{\|\mathbf{M}_P^S - \mathbf{g}_g^n\|_2}{d_{\max}}. \quad (19)$$

Among them, $d_{\max}$ represents the peak value of the Euclidean distance between surface element and all nodes in the neighborhood.

After the construction of the deformed map is completed, the pose parameters are further optimized through the graph optimization. For the two point clouds to be fused, there are many corresponding point pairs, from which some point pairs are uniformly extracted to establish constraints. The established constraints are as follows:

$$\begin{aligned} \mathbf{Q}^P &= (\mathbf{T}_d P(u, D_t^a), \mathbf{T}_S P(u, D_t^a), \mathbf{Q}_{d_t}^P, \mathbf{Q}_{S_t}^P) \\ &= (\mathbf{Q}_d^P, \mathbf{Q}_S^P, \mathbf{Q}_{d_t}^P, \mathbf{Q}_{S_t}^P). \end{aligned} \quad (20)$$

Among them, $\mathbf{Q}_d^P$ represents the position of target point cloud, $\mathbf{T}_S$ represents the pose of camera in current frame, $\mathbf{T}_d$ represents the camera pose in which the loopback frame is detected, $P(u, D_t^a)$ represents the point obtained from this

model projection through the pose of previous frame, and $\mathbf{Q}_S^P$ represents the position in the frame point cloud, $\mathbf{Q}_{d_t}^P$ and $\mathbf{Q}_{S_t}^P$ represent the timestamp corresponding to $\mathbf{Q}_d^P$ and $\mathbf{Q}_S^P$. The graph optimization is to optimize the points at these closed loops, using the following four cost function optimizations.

First, the Forbenius norm is used to optimize the pose of each node to ensure that the pose matrix $\mathbf{R}$ is orthogonal matrix, and the square error function is:

$$E_{\text{rot}} = \sum_l \left\| \mathbf{g}_R^{lU} \mathbf{g}_R^l - \mathbf{C} \right\|_F^2. \quad (21)$$

Second, it is necessary to normalize the nodes so that the pose parameters of the two neighbor nodes are uninterrupted, and the corresponding square error function is:

$$E_{\text{reg}} = \sum_l \sum_{n \in N(g^l)} \left\| \mathbf{g}_R^l (\mathbf{g}_g^n - \mathbf{g}_g^l) + \mathbf{g}_g^l + \mathbf{g}_t^l - (\mathbf{g}_g^n - \mathbf{g}_t^l) \right\|_2^2. \quad (22)$$

Combine Eq. (17) to solve the deformed coordinates $\hat{\mathbf{Q}}_d^P$ and $\hat{\mathbf{Q}}_S^P$ of $\mathbf{Q}_d^P$ and $\mathbf{Q}_S^P$, and align and optimize the point clouds in the two adjacent frames into target point cloud, and the cost function is as follows:

$$E_{\text{con}} = \sum_P \left\| \hat{\mathbf{Q}}_S^P - \hat{\mathbf{Q}}_d^P \right\|_2^2. \quad (23)$$

Finally, because the target point cloud was previously generated, it is necessary to optimize the coordinates of the target point cloud. This square error function is as follows:

$$E_{\text{pin}} = \sum_P \left\| \hat{\mathbf{Q}}_d^P - \mathbf{Q}_d^P \right\|_2^2. \quad (24)$$

Based on the above formulas (21)–(24), the final total optimization function can be constructed to optimize the solution deformation graph, and the total cost function is as follows:

$$E_{\text{def}} = w_{\text{rot}} E_{\text{rot}} + w_{\text{reg}} E_{\text{reg}} + w_{\text{con}} E_{\text{con}} + w_{\text{pin}} E_{\text{pin}}. \quad (25)$$

Among them, the empirical value of each weight is:

$$w_{\text{rot}} = 1, w_{\text{reg}} = 10, w_{\text{con}} = w_{\text{pin}} = 100. \quad (26)$$

The optimized camera pose is provided to the map construction program, and the point cloud is fused and optimized through the surface element model to obtain the final 3D reconstruction map.

Fig. 5 Feature extraction efficiency compared with other methods (A/seconds)

Table 1 Table of contrast experiments based on TUM RGB-D datasets

ATE/m v.s. h/ms	FOVIS	DVO	FSD-SLAM
fr1/xyz	0.043/142	0.036/186	0.029/86
fr1/rpy	0.063/153	0.042/196	0.038/95
fr1/desk	0.073/187	0.064/175	0.056/99
fr3/struct_text_far	0.025/137	0.018/183	0.014/82
fr3/struct_notext_far	0.127/185	0.130/197	0.086/81

Experiments

We evaluated the proposed FSD-SLAM system in the public TUM dataset [26] of the Technical University of Munich and in the real environment. The experimental platform in the real environment is turtlebot 2, and the equipped sensor is KinectV1. The operating system is Ubuntu Linux 18.04 LTS and ROS Melodic.

Evaluation standard

The indicators commonly used to evaluate VSLAM are absolute trajectory error (ATE) [27] and relative pose error (RPE) [28]. In addition, to provide a measure independent of the scene length, indicators usually need to evaluate the average value of the evaluation (Mean), median (Median), minimum (Min), maximum (Max) and root mean square error (RMSE). Given the estimated trajectory with translation component $\hat{x} = [\hat{x}_1, \dots, \hat{x}_n]$, and the real ground trajectory with translation component $\hat{x} = [x_1, \dots, x_n]$, RMSE is given by Eq. (27).

$$\text{RMSE} = \left(\frac{1}{n} \sum_{i=1}^n \left\| \hat{x}_i - x_i \right\|^2 \right)^{\frac{1}{2}}. \quad (27)$$

Evaluation on TUM RGB-D dataset

From the earliest Moravec, to Harris, to SIFT, SUSAN, and SURF algorithms, it can be said that feature extraction

Fig. 6 Figure of contrast experiments based on the fr1/desk dataset

Fig. 7 Freiburg1_room sequence tests ATE after optimization

Fig. 8 Freiburg1_room sequence tests RPE after optimization

Table 2 Freiburg1_room sequence tests ATE after optimization

Algorithm	RMSE	Mean	Median	Min	Max
RGB-D SLAM	0.063425 m	0.053672 m	0.046759 m	0.009736 m	0.182376 m
FSD-SLAM	0.045632 m	0.0426578 m	0.034562 m	0.006743 m	0.154378 m

Table 3 Freiburg1_room sequence was tested for the translation error of relative pose after optimization

Algorithm	RMSE	Mean	Median	Min	Max
RGB-D SLAM	0.042637 m	0.043789 m	0.051879 m	0.012576 m	0.195483 m
FSD-SLAM	0.035627 m	0.025674 m	0.025364 m	0.002364 m	0.178543 m

Table 4 Freiburg1_room sequence was tested for the rotation error of relative pose after optimization

Algorithm	RMSE	Mean	Median	Min	Max
RGB-D SLAM	1.825637°	1.764839°	1.648932°	0.356278°	5.835269°
FSD-SLAM	1.724893°	1.634672°	1.568742°	0.157234°	3.673892°

Table 5 Comparison results of robustness between FSD-SLAM and ORB-SLAM2

TUM sequence	ORB-SLAM2	FSD-SLAM
fr3_nostructure_far	20	0
fr3_nostructure_notexture_near	20	3
fr3_structure_notexture_far	12	0
fr3_nostructure_texture_withloop	0	0
fr3_nostructure_texture_far	20	0
fr3_nostructure_texture_near	20	0
fr3_structure_texture_near	0	0
fr3_structure_texture_far	0	0

algorithms are endless. The features extracted by the above algorithms such as SIFT and SURF are also excellent, but time consumption is very large. This makes the real-time performance bad and reduces the performance of the system. After research, Edward Rosten and Tom Drummond proposed the FAST algorithm [24]. The FAST detector is very fast because it does not involve complex operations such as scale, gradient and so on.

To verify the effect of the feature enhancement module in the system of this paper, for each image of the fr3_long_office sequence in the TUM dataset, the feature extraction algorithm mentioned above and FSD-SLAM are used for feature extraction, and the feature extraction efficiency of each image is calculated. Then we take Min, Max, Mean and RMSE for the feature extraction efficiency of each algorithm in the whole sequence, and Fig. 5 shows this result. It can be seen from the experimental result that FSD-SLAM can extract more feature points in a short time and has better feature extraction efficiency.

To qualitatively analyze the accuracy of FSD-SLAM for camera motion estimation in complex scenes, experiments are carried out on the TUM dataset with FSD-SLAM, FOVIS [29] based on the feature point method, and DVO [30] based on the direct method. Table 1 is the comparison result of the method in this paper with the other two methods in terms of registration speed and accuracy. h represents the average data feature extraction and registration processing time per frame. Figure 6 is a comparison diagram of the pose estimation results of the method in this paper and the other two methods on fr1/desk sequence. The fr1/desk sequence is image data recorded by KinectV2 at full speed (30HZ). The resolution of the sensor is 640×480 . It was shot in a typical office environment.

Through comparative experiments, it can be seen that the three algorithms can correctly track the camera's trajectory.

However, it can be seen from Table 1 that FSD-SLAM is significantly better than DVO and FOVIS in terms of registration speed, and can reach more than ten frames per second.

The evaluation of the accuracy of the algorithm is carried out in this paper, the optimized ATE and RPE are used as the evaluation criteria, and the freiburg1_room sequence in the TUM dataset is selected to test the optimization work effect, and compared with the representative RGB-D SLAM [14] optimization effect. The ATE result after trajectory optimization during the test is shown in Fig. 7 and the RPE results is shown in Fig. 8. It can be seen from the experimental result that, under the same video sequence freiburg1_room, the RPE and ATE of FSD-SLAM algorithm are only about 10% compared with the RGB-D SLAM algorithm.

The results of quantitative evaluation of the optimized ATE are shown in Table 2. Quantitative evaluation of optimized RPE is divided into translation errors and rotation errors. The results are shown in Tables 3 and 4, respectively. According to the result in Tables 3 and 4, the positioning accuracy after FSD-SLAM optimization reaches 0.472%, and the positioning accuracy after RGB-D SLAM optimization is 0.341%, which means that the positioning accuracy after FSD-SLAM optimization is 38% higher than that after RGB-D SLAM optimization.

Further, we verify the robustness of the system in the environment of visual feature scarcity and low texture. Verification experiments are carried out on the image sequence with or without texture and structure in the TUM dataset, and compared with the ORB-SLAM2 [17] system. The evaluation standard is to run the system 20 times, and the number of times the system is lost. The experimental results are shown in Table 5. The bold results indicate less lost times. It can be seen that the ORB-SLAM2 system is lost in the image sequence of fr3_nostructure_far, fr3_nostructure_noture_near, fr3_nostructure_texture_far and

Table 6 Comparison results of tracking time between FSD-SLAM and ORB-SLAM2 single frame image (s)

TUM sequence	ORB-MONO	FSD-MONO	ORB-STEREO	FSD-STEREO	ORB-RGB-D	FSD-RGB-D
R1_360	0.018	0.017	X	X	0.026	0.024
fr1_desk	0.021	0.016	X	X	0.032	0.027
fr1_desk2	0.028	0.027	X	X	0.037	0.028
fr1_floor	0.017	0.014	X	X	0.023	0.022
fr1_room	0.024	0.024	X	X	0.031	0.024
fr1_xyz	0.031	0.012	X	X	0.030	0.020
fr2_360_hemi	0.014	0.011	X	X	0.026	0.018
fr2_360_kidnap	0.016	0.010	X	X	0.020	0.019
fr2_desk	0.022	0.013	X	X	0.034	0.016
fr2_large_withloop	0.017	0.018	X	X	0.018	0.021
fr2_large_no_loop	0.018	0.018	X	X	0.028	0.017
fr2_pioneer_360	0.027	0.029	X	X	0.023	0.022
fr2_pioneer_slam	0.021	0.018	X	X	0.025	0.026
fr2_pioneer_slam2	0.022	0.025	X	X	0.024	0.023
fr2_pionner_slam3	0.019	0.027	X	X	0.024	0.024
fr2_rpy	0.023	0.015	X	X	0.025	0.017
fr2_xyz	0.027	0.017	X	X	0.027	0.017

fr3_nostructure_texture_near, and our system can track these sequences well. Overall, our system has less lost times than the ORB-SLAM2 system that only uses the feature method in most environments without texture and structure.

Another purpose of the algorithm proposed in the paper is to improve the real-time performance of SLAM system. To evaluate this system more effectively, the time of tracking a single frame image is taken as the evaluation standard in the dataset of TUM, and compared with ORB-SLAM2 system. Table 6 shows the tracking time of a single frame image under the same parameters for the system in this article and the ORB-SLAM2 system.

In Table 6, MONO, STEREO and RGB-D, respectively, represent three different SLAM processing modes of monocular, binocular and RGB-D [17]. The result in bold indicates faster tracking time. It can be seen that the FSD-SLAM takes less time to track a single image in most cases, which improves real-time performance of SLAM system.

Evaluation in real environment

From the start of the robot to the end of the work, the corresponding duration of continuous movement is 48.90 s, and the actual track length of the ground is 3.986 m, the average translation speed of 0.432 m/s and the average angular speed of 29.882 rad/s. The test compares the ATE and RPE of the proposed method and several representative SLAM methods in camera pose estimation. These SLAM methods are RGB-D SLAM, HOOFR SLAM [3], ORB-SLAM, BAD SLAM [31] and DOOR-SLAM [32]. The results are shown

Table 7 ATE and RPE evaluations compared to other methods

Method	ATE (m)	RPE (m)
FSD-SLAM	0.018793	0.028753
RGBD-SLAM	0.025415	0.038691
HOOFR SLAM	0.029326	0.048293
ORB-SALM	0.035612	0.031973
BAD SLAM	0.029569	0.029427
DOOR-SLAM	0.022371	0.047628

Fig. 9 The initialized point cloud map and three-dimensional point cloud map generated by FSD-SLAM

in Table 7 and the bold item is the best. Figure 9 shows the initialized point cloud map and global dense point cloud map obtained by the system.

The experimental results in Table 7 show that this work outperforms the real-time SLAM method and achieves a lower error rate. At the same time, we can see from Fig. 9

that the system of this paper can reconstruct the dense global point cloud map of the indoor environment.

Conclusion

The FSD-SLAM algorithm proposed in the paper uses the feature enhancement module based on subgraph in the initialization and feature enhancement part, which improves the efficiency of feature extraction. Combining the feature point information and photometric information of the image at the same time in the pose estimation part of the camera can realize stable and accurate pose estimation in the environment with scarce visual features and low texture. The incremental dynamic covariance scaling algorithm proposed in the optimization part optimizes the estimated camera pose globally, which simplifies the optimization goal, avoids a large number of calculations, and improves the positioning accuracy. Finally, a 3D point cloud map is constructed through map fusion. Both the TUM data set and the actual scene online experiment show that FSD-SLAM maintains a high accuracy rate and positioning accuracy. The validation test in the data set shows that FSD-SLAM has a small number of lost track times in most untextured and unstructured environments, and has high real-time performance.

With the advantages of excellent precision, high reliability, good efficiency and robustness, the FSD-SLAM algorithm could be applied to the navigation of service robots in noisy environment of large shopping malls, the navigation and positioning of AGV in automatic warehouses, the automatic parking service of automatic parking system and the rapid patrol of commercial buildings. Our next work is to combine the semantic mapping method, so that the algorithm can be applied to indoor and outdoor scenes.

Acknowledgements This work was supported in part by the National Key Research and Development Project Intelligent Robot Special Program under (Grant 2018YFB1305804), the National Natural Science Foundation of China (Grants 62025307, 61873268 and U1913209), the Beijing Natural Science Foundation (Grant JQ19020) and the Anhui University Cooperative Innovation Project (Grant GXXT-2019-003).

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Endres F, Hess J, Engelhard N, Sturm J, Cremers D, Burgard W (2016) An evaluation of the rgb-d slam system. In: 2012 IEEE international conference on robotics and automation, pp 1691–1696
- Bavle H, De La Puente P, How JP, Campoy P (2020) Vps-slam: visual planar semantic slam for aerial robotic systems. *IEEE Access* 8:60704–60718
- Nguyen D, Elouardi A, Rodriguez Florez SA, Bouaziz S (2019) Hoofr slam system: an embedded vision slam algorithm and its hardware-software mapping-based intelligent vehicles applications. *IEEE Trans Intell Transp Syst* 20(11):4103–4118
- Gomez-Ojeda R, Moreno F, Zuñiga-Noël D, Scaramuzza D, Gonzalez-Jimenez J (2019) Pl-slam: a stereo slam system through the combination of points and line segments. *IEEE Trans Rob* 35(3):734–746
- Kim H, Song S, Hyun J, Hong SH, Myung H (2018) Rgb-d and magnetic sequence-based graph slam with kidnap recovery. In: 2018 18th international conference on control, automation and systems (ICCAS), pp 1440–1443
- Liu D-X, Jing X, Chen C, Long X, Xinyu W (2019) Vision-assisted autonomous lower-limb exoskeleton robot. *IEEE Trans Syst Man Cybernet Syst PP*(99):1–12
- Fu Qiang Y, Hongshan LL, Jingwen W, Mingui S (2019) A robust rgb-d slam system with point and line for low texture indoor environment. *IEEE Sensor J PP*(99):1
- Wei HU, Xing-yu LIU (2019) A monocular slam image matching method based on improved sift algorithm. *Electron Opt Control* 2019:5
- Belter D, Nowicki M, Skrzypczyński P (2016) Improving accuracy of feature-based rgb-d slam by model spatial uncertainty of point feature. In: 2016 IEEE international conference on robotics and automation (ICRA), pp 1279–1284
- Meng C, Guo B, Liu X (2016) Simultaneous localization and mapping using monocular direct method. In: 2016 14th international conference on control, automation, robotics and vision (ICARCV), pp 1–5
- Zhang W, Yan Z, Xiao G, Feng A, Zuo W (2019) Real-time direct monocular slam with learning-based confidence estimation. *IEEE Access PP*(99):1
- Juan L, Gwun O (2009) A comparison of sift, pcasift and surf. *Int J Image Process* 3(4):143–152
- Lowe DG (2004) Distinctive image features from scale-invariant keypoints. *Int J Comput Vis* 60(2):91–110
- Henry P, Krainin M, Herbst E, Cheng M, Ren X, Fox D (2013) Rgb-d mapping: using depth cameras for dense 3d modeling of indoor environments. *Int J Robot Res* 31(5):647–663
- Endres F, Hess J, Sturm J, Cremers D, Burgard W (2014) 3-d mapping with an rgb-d camera. *IEEE Trans Rob* 30(1):177–187
- Bay H (2006) Surf: speeded up robust features. *Comput Vision Image Understand* 110(3):404–417
- Mur-Artal R, Tardós JD (2017) Orb-slam2: an open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Trans Rob* 33(5):1255–1262
- Klein G, Murray D (2007) Parallel tracking and mapping for small ar workspaces. In: 2007 6th IEEE and ACM international symposium on mixed and augmented reality, pp 225–234
- Mur-Artal R, Montiel JMM, Tardós JD (2015) Orb-slam: a versatile and accurate monocular slam system. *IEEE Trans Rob* 31(5):1147–1163
- Engel J, Schps T, Cremers D (2014) Lsd-slam: large-scale direct monocular slam. In: European conference on computer vision
- Forster C, Pizzoli M, Scaramuzza D (2014) Svo fast semi-direct monocular visual odometry. In: 2014 IEEE international conference on robotics and automation (ICRA), pp 15–22

22. Rublee E, Rabaud V, Konolige K, Bradski G (2016) Orb: An efficient alternative to sift or surf. In: 2011 international conference on computer vision, pp 2564–2571
23. Krombach N, Droschel D, Behnke S (2016) Combining feature-based and direct methods for semi-dense real-time stereo visual odometry
24. Ma Y, Shi L (2017) A modified multiple self-adaptive thresholds fast feature points extraction algorithm based on image gray clustering. In: 2017 international applied computational electromagnetics society symposium (ACES), pp 1–5
25. Sünderhauf N, Protzel P (2012) Switchable constraints for robust pose graph slam. In: 2012 IEEE/RSJ international conference on intelligent robots and systems, pp 1879–1884
26. Keimel C, Redl A, Diepold K (2018) The tum high definition video datasets. In: 2012 Fourth international workshop on quality of multimedia experience, pp 97–102
27. Sturm J, Engelhard N, Endres F, Burgard W, Cremers D (2016) A benchmark for the evaluation of rgb-d slam system. In: 2012 IEEE/RSJ international conference on intelligent robots and systems, pp 573–580
28. Runz M, Buffier M, Agapito L (2018) Maskfusion: real-time recognition, tracking and reconstruction of multiple moving objects. In: 2018 IEEE international symposium on mixed and augmented reality (ISMAR), pp 10–20
29. Huang AS, Bachrach A, Henry P, Krainin M, Maturana D, Fox D, Roy N (2017) Visual odometry and mapping for autonomous flight using an rgb-d camera. In: IEEE international symposium on circuits & systems
30. Kerl C, Sturm J, Cremers D (2013) Robust odometry estimation for rgb-d cameras. In: 2013 IEEE international conference on robotics and automation, pp 3748–3754
31. Schöps T, Sattler T, Pollefeys M (2019) Bad slam: Bundle adjusted direct rgb-d slam. In: 2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 134–144
32. Lajoie P, Ramtoula B, Chang Y, Carlone L, Beltrame G (2020) Door-slam: distributed, online, and outlier resilient slam for robotic teams. *IEEE Robot Autom Lett* 5(2):1656–1663

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.