Research on the Construction Method of 3D Dense Map Based on Visual Slam

Tianfang Sun*, Xiangbin Shi

College of Computer Science, Shenyang Aerospace University, Shenyang, Liaoning Province, 110136, China

*Corresponding author’s e-mail: 18210607403@email.sau.edu.cn

Abstract. Aiming at the problem that visual SLAM (Simultaneous Localization and Mapping) cannot satisfy both high-precision positioning and rapid mapping in large scenes, this paper proposes a map construction method. First, when calculating the pose, the method of combining EPNP and graph optimization is used to optimize the pose estimation to improve the accuracy of the pose. Then, the paper uses Lie algebra to define the vertices and edges of the pose graph, which improves the convergence speed of the model. Finally, when constructing a three-dimensional map, a divide-and-conquer map method is proposed, which uses multiple sub-maps that contain each other for point cloud registration and down-sampling filtering, so as to improve the accuracy of map construction. Experimental results show that the algorithm in this paper has good performance. In the TUM test set, the speed can reach 0.139s/frame, which is about 0.2s higher than the original method, and the accuracy error RMSE value is reduced to 0.055m. The feasibility of the algorithm is proved, and it can meet the accuracy and speed requirements of 3D dense map construction in large scenes.

1. Introduction

Visual SLAM is a system that uses a camera as a sensor to estimate the pose of a robot and build an environment map[1]. The three-dimensional dense map is a map formed by splicing the point cloud information of the key frames extracted by the visual SLAM. The three-dimensional dense map construction includes basic steps such as image feature point extraction, inter-frame matching, robot pose estimation, error optimization, loop detection, and three-dimensional point cloud information extraction. Three-dimensional dense maps have a wide range of applications in robot navigation, three-dimensional reconstruction, obstacle avoidance and other fields.

Pose estimation and optimization are the most basic and important links in visual slam. Endres et al. [2] proposed the RGBD-SLAM[3] (RGB-Depth SALM) algorithm to use iterative closest point ICP[4] (Iterative Closest Point) for pose estimation. Literature [5] proposed an RGB-D SLAM algorithm based on plane features. The use of flat point features is beneficial to improve the accuracy and robustness of traditional ICP, while maintaining a reasonable amount of calculation to ensure real-time performance. However, there are problems of serious resource consumption and unfavorable maintenance. ElasticFusion [6] realizes visual odometry by fusing ICP and dense photometric reprojection errors, and constructs a point cloud map with high accuracy. However, the need for GPU acceleration is very time-consuming, resulting in low real-time performance. BA can accurately optimize the position of each camera pose and feature points. However, in a large scene, there are a large number of feature points, which leads to too much calculation and too long processing time to be optimized in real time. The ORB-SLAM [7] system proposed by Mur-Artal et al. [8] supports three
modes of monocular, binocular, and RGB-D, while ensuring good rotation and scaling invariance. However, the system does not provide the functions of storing maps and relocating, and its output is sparse map points. If the robot builds a map in a larger scene, the cumulative error will gradually increase, resulting in a large drift phenomenon in the constructed map.

In view of the problems in the above literature, this paper proposes a method of fusing EPNP [9] with graph optimization when performing pose estimation, which improves the overall pose accuracy of SLAM. Then the paper uses Lie algebra to define the vertices and edges of the pose graph, which speeds up the calculation. Finally, when constructing a three-dimensional dense map, this paper proposes a divide-and-conquer map method, that is, the overall 3D map is split into several small 3D maps that contain each other, and each small 3D map is optimized, and then the sub-maps are spliced into the final 3D map by means of point cloud registration. Experiments show that this algorithm can construct a three-dimensional dense map for a long time and at a long distance, and while it has obtained better results under the standard RGBD data set.

2. Pose optimization

2.1. Pose estimation

Pose estimation is to calculate the movement pose of the robot. Pose is the position of the robot in space and its own orientation. The pose of the robot can be regarded as the transformation from the original position to the current position, which is described by the transformation matrix $T$. The transformation matrix $T$ is represented as follows:

$$T = \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix}$$

(1)

The upper left corner is the rotation matrix, the right side is the translation vector, the lower left corner is the 0 vector, and the lower right corner is the 1.

The ICP algorithm is usually used to estimate the pose, but the ICP algorithm has higher requirements for the initial value. If the initial value is incorrect, it will fall into a local optimum in the iteration, and the registration efficiency will be low. At the same time, the ICP algorithm will cause that the algorithm fails when there is noise or mismatch. This paper first uses the EPNP algorithm to solve the initial pose estimate $T$, and then uses graph optimization to optimize its results.

EPnP estimates the pose of the robot in the reference coordinate system based on the two-dimensional pixel coordinates of the feature points in the N pairs of images and their corresponding three-dimensional space point coordinates.

In this paper, the pose obtained by the EPNP method is used as the initial quantity, and the method of nonlinear optimization is adopted to construct the least square problem, and further optimize the obtained pose $T$. This paper uses graph optimization to solve the least squares problem. Graph optimization is to express the optimization problem in the form of a graph. A graph consists of several vertices and the edges connecting these vertices, The expression form is: $G=\{V, E\}$. In the pose estimation problem, $V$ represents the vertices of the graph and represents the optimization variable, that is, the rotation matrix $T$; $E$ represents the edge of the graph representing the error term, that is, the reprojection error. In graph optimization, the Levenberg-Marquadt [10] algorithm is used to solve the least square problem. Combining EPNP and graph optimization methods can calculate the camera pose more accurately.

2.2. Pose graph optimization

Since the number of feature points in BA is much greater than the number of pose nodes, The Hessian matrix $H$ is sparse and cannot achieve real-time calculation. At the same time, the calculation speed will be severely reduced in a larger scene, and the accumulated error will become larger and larger. Therefore, this paper adopts the optimization method of pose graph to solve this problem. In the optimization of the pose graph, the initial value of the edge is calculated from two key frames through feature matching. After obtaining the initial value, there is no need to optimize the position of the
landmark point, just to care about the connection between the camera poses. This way of only keeping
the key frame trajectory saves the calculation of optimized feature points. At the same time, in order to
speed up the convergence speed of the model and improve the accuracy, Lie algebra is used to define
vertices and edges. Lie algebra can directly perform the rotation transformation between two
coordinate systems without starting the transformation from the base coordinate system, thus speeding
up the model convergence speed. The movement ∆T_{ij} between the poses T_i and T_j is represented by
the Lie group:

\[ T_{ij} = T_i^{-1}T_j \] (2)

Move \( T_{ij} \) in formula (3) to the right side of the equation to construct the least square error \( e_{ij} \).

\[ T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} e_{ij} = \Delta \xi_{ij} \ln(T_{ij}^{-1}T_i^{-1}T_j)^{V} \] (3)

\( e_{ij} \) Derivative of optimization variables \( \xi_i \) and \( \xi_j \). Add a left disturbance on each side of the
optimization variable: \( \delta \xi_i \) and \( \delta \xi_j \).

\[ e_{ij} = \ln(T_{ij}^{-1}T_i^{-1})\exp((-\delta \xi_j)^{V})\exp((\delta \xi_i^{V})T_i)^{V} \] (4)

The disturbance is moved to the far right of the formula, and the right multiplication form is
derived.

\[ e_{ij} = \ln(T_{ij}^{-1}T_i^{-1})\exp((-\delta \xi_j)^{V})\exp(\delta \xi_i^{V})T_i^V \approx e_{ij} + \frac{\partial e_{ij}}{\partial \delta \xi_i} \delta \xi_i + \frac{\partial e_{ij}}{\partial \delta \xi_j} \delta \xi_j \] (5)

Find the Jacobian matrix of the error with respect to the two poses.

\[ T_i: \frac{\partial e_{ij}}{\partial \delta \xi_i} = -3T_i^{-1}(e_{ij})Ad(T_i) \] (6)

\[ T_j: \frac{\partial e_{ij}}{\partial \delta \xi_j} = 3T_i^{-1}(e_{ij})Ad(T_i^{-1}) \] (7)

Graph optimization is essentially a least squares problem, the optimization variable is the pose
of each vertex, and the edges are the pose observation constraints. \( \xi \) is the set of all edges, the overall
objective function.

\[ \min \frac{1}{2} \sum_{i,j \in \xi} e_{ij}^{T} \sum_{i,j}^{-1} e_{ij} \] (8)

After the original graph optimization algorithm was tested, it was found that the error was still
changing after 30 iterations. The overall error of the algorithm in this paper remains stable after 23
iterations, and the effect is significantly better than the original algorithm.

3. Divide and conquer to build a map
The map is the collection of all landmark points. After all landmark points are determined, the three-
dimensional point cloud information is extracted and stitched according to the estimated pose to form
a three-dimensional dense map. Due to the noise at the observation point, there will still be errors in
the optimized pose. The built-in map in a small space will produce a small amount of error, and when
it is expanded to a large space, the error will gradually accumulate and it will cause a larger error,
resulting in a larger drift and more overlap in the constructed map. When building a map, each
subsequent key frame will be stitched according to the estimated pose on the basis of the previous
frame. The estimated pose error for each previous frame will be accumulated for each subsequent
frame, which will make the entire map have a larger offset. When constructing a 3D dense map in a
large scene, because the constructed map is very large, it takes up a lot of computer memory, which
reduces the overall operating speed of the system. This paper further optimizes the landmark points in
the construction of the map, and proposes the divide and conquer map method.
3.1. Build ideas
First, the big map is split into several sub-maps that contain key frames. When the key frame contained in the sub-map reaches a certain number, the sub-map is automatically saved. Because the error of each sub-map is accumulated from 0, instead of the cumulative error sum of all previous estimated poses, the overall error is reduced. Then the paper uses the outer point removal filter for each submap to remove the external point cloud in a given range. Using a voxel grid downsampling filter (Voxel grid filter) can reduce the number of point clouds and improve the registration rate while maintaining the shape characteristics of the point cloud. Finally, point cloud registration is performed on each sub-map, and the second image is used as the reference coordinate system during the point cloud registration process. Using mutually contained key frames for point cloud registration, the error is reduced, and so on, the overall error is reduced, thereby also completing the optimization of the inconspicuous position of the feature point. This process improves the accuracy of the map.

3.2. Key frame extraction
The distance between frames is very close. If each frame is added to the map, the map will be updated frequently, wasting time and space, and will cause redundancy of 3D points. When selecting a key frame, this paper proposes a threshold key frame algorithm, that is, when the frame is within a certain threshold, it will be judged as a key frame and added to the map, otherwise the frame is discarded. 

\[ \text{min} \_\text{norm} \leq ||\Delta t|| + \min(2\pi - ||r||, ||r||) \leq \text{max} \_\text{norm} \]  

(9)

where the displacement of the camera translation vector \( t \) in the rotation matrix \( T \). \( r \) is the rotation angle \( R \) in the rotation matrix \( T \). The norm of displacement and rotation is added to describe the size of camera movement. \( \text{max} \_\text{norm} \) is the maximum motion between cameras. When the camera motion is greater than \( \text{max} \_\text{norm} \), the camera pose estimation error is considered and the frame is discarded. \( \text{min} \_\text{norm} \) is the minimum motion between cameras. When the camera motion is greater than \( \text{min} \_\text{norm} \) and less than \( \text{max} \_\text{norm} \), it is considered to be selected as a key frame. After debugging, it is believed that the optimal value of \( \text{min} \_\text{norm} \) is 0.3, and the optimal value of \( \text{max} \_\text{norm} \) is 5.

4. Experimental results and analysis

4.1. Experiment preparation
This paper is in the ubuntu16 LTS environment experiment, and the computer hardware configuration is Intel Core i7 processor, GTX1050 graphics card, 4G memory. The visual camera used is KINECT v1, and the standard data set provided by RGBD-SLAM is used to construct a dense map.

4.2. Compared with RGBD-SLAM algorithm
This article downloads 6 sets of data sets for experiments from the TUM data set \[11\] of the Technische Universität München. This data set provides the real pose of KINECT motion and the internal parameters of the cameras in each data set.

This paper compares the proposed algorithms with RGBD-SLAM-V1[12] and RGBD-SLAM-V2[13]. RGBD-SLAM-V1 is an RGBD-SLAM system completed by the University of Munich. The system does not include a closed-loop detection module. RGBD-SLAM-V2 is an improved system based on the original by the same author, including the use of GPU acceleration technology for acceleration, and the addition of a closed-loop detection module based on the bag-of-words model.

Figure 1 is a three-dimensional dense map which is constructed by RGBD-SLAM-V2 and the algorithm used in this paper, which applys room data set. It can be clearly seen that the algorithm used in this paper is obviously better than the three-dimensional dense map constructed by the RGBD-SLAM-V2 algorithm, with more complete details and less noise. Figure 2 shows the comparison between the real trajectory of the data set and the test trajectory under this algorithm. The black solid line is the real trajectory of the data set, the blue dashed line is the test trajectory, and the dashed line
formed by the red dots is the difference between the two trajectories. Table 1 shows the comparison of root mean square error (RMSE/m) and processing time per frame (Time/s) under 6 sets of data.

![Figure 1. The algorithm in this paper and the RGBD-SLAM algorithm are compared under the FR1 room data set.](image1)

![Figure 2. RGBD-SLAM algorithm](image2)

**Table 1.** Data set real trajectory and test Trajectory comparison.

| data set   | Processing time per frame/s | RMSE/m |
|------------|-----------------------------|--------|
|            | SLAM-V1 | SLAM-V2 | Algorithm | SLAM-V1 | SLAM-V2 | Algorithm |
| FR1_desk   | 0.447   | 0.346   | 0.164     | 0.102   | 0.049   | 0.053     |
| FR1_desk2  | 0.395   | 0.287   | 0.166     | 0.189   | 0.102   | 0.038     |
| FR1_floor  | 0.520   | 0.402   | 0.173     | 0.154   | 0.055   | 0.054     |
| FR1_plant  | 0.412   | 0.318   | 0.099     | 0.203   | 0.142   | 0.021     |
| FR1_room   | 0.412   | 0.318   | 0.099     | 0.321   | 0.219   | 0.049     |
| FR1_rpy    | 0.443   | 0.354   | 0.126     | 0.097   | 0.042   | 0.079     |
| average value | 0.477   | 0.348   | 0.139     | 0.178   | 0.101   | 0.055     |
It can be seen from Table 1 that the algorithm in this paper is significantly better than the RGBD-SLAM algorithm in terms of accuracy and speed. It can be seen from the time that the average processing time per frame of the 6 groups of RGBD-SLAM-V2 data sets is 0.348s, and the processing time of the algorithm in this paper is 0.052s, which is about 2.5 times that of RGBD-SLAM-V2. In terms of accuracy, the RMSE value of the algorithm in this paper is 0.055m, which is significantly lower than the RMSE value of RGBD-SLAM.

5. Conclusion
First, this paper proposed a fusion method of EPNP and graph optimization to improve the accuracy of pose estimation. Then, the Lie algebra was used to define the vertices and edges of the pose graph, which speeds up the convergence speed of the model. Finally, a divide-and-conquer map method was proposed, which improved the accuracy of the map and shortens the time to construct the map. The experimental data shows that the algorithm is better than RGBD-SLAM in accuracy and speed, and it is more robust. In the future, we will study the learning of semantic SLAM, IMU and other modules.

Acknowledgments
Thanks for the help of classmates and teachers in the laboratory. Thanks to the National Natural Science Foundation of China (61170185, 61602320), the Liaoning Provincial Doctoral Startup Fund Project (201601172) and the Liaoning Provincial Department of Education's Scientific Research Project (L201607) for funding.

References
[1] Gu, Z.P., Liu, H. (2016) A survey of monocular simultaneous localization and mapping. In:Journal of Computer-Aided Design and Computer Graphics. Chineses. pp. 855-868.
[2] ENDRES, F., HESS, J., STURM, J., et al. (2014) 3-D mapping with an RGBD camera. IEEE Transactions on Robotics, 30(1): 177 — 187.
[3] Bay, H. (2006) SURF: Speeded Up Robust Features. Computer Vision & Image Understanding, 110(3):404-417.
[4] Bergström, P., Edlund, O. (2016) Robust registration of surfaces using a refined iterative closest point algorithm with a trust region approach. Numerical Algorithms, 74(3): 1-25.
[5] GAO, X., ZHANG, T. (2015) Robust RGB-D simultaneous localization and mapping using planar point features. Robotics and Autonomous Systems, 2015, 72: 1-14.
[6] WHELAN, T., SALAS-MORENO, R.F., GLOCKER B, et al. (2016) ElasticFusion: real-time dense SLAM and light source estimation. The International Journal of Robotics Research. 35(14): 1697 -1716.
[7] MUR-ARTAL, R., MONTIEL, J.M.M., TARDOS S.J.D.,(2015) ORB-SLAM: a versatile and accurate monocular SLAM syste.m. IEEE Transactions on Robotics, 31(5): 1147 -1163.
[8] Mur-Artal, R., Montiel, J.M.M., Tardos, J.D.(2015) ORB-SLAM: A Versatile and Accurate Monocular SLAM System. Robotics IEEE Transactions on, 31(5):1147-1163.
[9] Lepetit, V., Moreno-Noguer, F., Fua, P.(2009)  EPnP: An Accurate O(n) Solution to the PnP Problem. International Journal of Computer Vision, 81(2):155-166.
[10] Cherrak, O., Ghennoui, H., Abarkan, E., et al. Levenberg-Marquardt Algorithm[J]. Tutorial on Lm Algorithm, 2004, 11(1):101-110.
[11] Jürgen, S., Engelhard, N., Endres, F., et al. (2012) A benchmark for the evaluation of RGB-D SLAM systems.IEEE/RSJ International Conference on Intelligent Robots & Systems. IEEE, 2012.
[12] Endres, F., Hess, J., Engelhard, N., et al. (2012) An evaluation of the RGB-D SLAM system. Proceedings IEEE International Conference on Robotics & Automation, 30(1):155-165.
[13] Endres, F., Hess, J., Sturm, J.,et al. (2014) 3-D mapping with an RGB-D camera. IEEE Transactions on Robotics, 30(1):177-187.