Research on SLAM Drift Reduction Mechanism Based on Point Cloud Segmentation Semantic Information

Junqin Lin¹*, Zhihong Chen¹, Yanbo Wang¹ and Yu Liao¹

¹ Beijing Research Institute of Precise Mechatronics and Controls, Beijing, 100076, China

*Corresponding author’s e-mail: linjunqin2010@163.com

Abstract. This paper combines the semantic segmentation of scenes with Simultaneous localization and Mapping (SLAM) technology to build a three-dimensional semantic map. The input sequence is selected by ORB-SLAM for key frame selection, and the scene's semantic segmentation is performed in the corresponding point cloud data. We use a new 3D segmentation framework, which can effectively simulate the local structure of point cloud. A drift reduction mechanism based on semantic constraints and Bundle Adjustment (BA) constraints was proposed. This mechanism considers the three-dimensional objects, feature points and camera pose for semantic recognition in the scene, and integrates them into the back-end BA to optimize them. The final experimental results show that compared with the current popular ORB-SLAM, this mechanism can reduce the system's translation drift error by 18.8%.

1. Introduction

Simultaneous localization and mapping (SLAM) refers to the process by which a robot estimates its pose and builds an environmental map in a strange environment using only sensors it carries. It is a prerequisite for many robot application scenarios, such as path planning, Collision-free navigation, environmental awareness. Obtaining accurate environmental information is the key link for mobile robots to perform tasks autonomously. When the robot is in motion, it can describe its surrounding environment through SLAM technology, that is, the environment map. However, the traditional SLAM composition only considers geometric data, and cannot obtain the category information of the objects in the map. The information provided is insufficient and the features are weakly distinguishable. Semantic information includes object categories, target detection, and semantic segmentation. It can understand the content of the scene and help robots perform target-oriented tasks. Therefore, the combination of the SLAM and Semantic is an inevitable need.

A long time ago, researchers tried to integrate object information into SLAM for semantic mapping. Mozos et al. [1] used a hidden Markov model to segment a map built with distance sensors into functional spaces (rooms, corridors, doorways, etc.). Nuchter et al. [2] combined the work of 3D indoor reconstruction labeling into SLAM, emphasizing the importance of semantic maps, and added a step of semantic segmentation in a thread. Civera et al. [3] proposed two parallel threads of EKF monocular SLAM and target recognition. Among them, target recognition is achieved through SURF feature point matching and geometric commonality verification. When a target is successfully identified, it is inserted into SLAM. In the map, its position information will be continuously optimized by subsequent frame sequence images. But it needs to manually obtain a small number of images of the target through the camera to generate the target model.
Since 2015, based on machine learning or deep learning, methods for combining image semantic understanding with SLAM have been proposed [4-6]. But most of the image segmentation uses full convolutional neural network (FCN) [7], the effect is not ideal. The SLAM ++ system proposed by Salas-Moreno et al. [8] is currently a better semantic mapping method. It compares the features of the point cloud with a database of prepared objects. If a matching object is found, it will correspond to the point cloud. Then insert it in the map. However, SLAM ++ can only map predefined objects, and its features used for template matching are manually extracted. Hermans et al. [9] considered that the dense semantic segmentation of 3D point clouds is difficult, but the segmentation effect is better on 2D pictures. Therefore, 2D-3D label conversion based on Bayesian update and dense conditional random field is used to generate 3 Semantic maps in dimensional space. Concha et al. [10] merged semantic segmentation with semi-dense Large-Scale Direct method (LSD) monocular SLAM, obtained the planes in the image through super pixel segmentation, and used the Large-Scale Direct method to obtain more prominent features such as edges. The fusion is performed to obtain denser real-time mapping results. The disadvantage is that the accuracy of the plane is not ideal.

This paper uses ORB-SLAM [11] to perform key frame filtering and inter-frame relative pose estimation on 2D images, and performs semantic segmentation on the corresponding point clouds of key frame images. A new framework for 3D point cloud segmentation is proposed, which can effectively Geo-modeling local structures in point clouds. A drift reduction mechanism based on semantic constraints and BA constraints was proposed. This mechanism considers the semantically recognized 3D objects, feature points and camera pose in the scene, and integrates them into the back-end BA to optimize together. It uses the spatial 3D positions of feature points and semantic information in the scene to reduce system drift errors.

2. system structure

In general, the classic methods in SLAM track visual geometric features, such as points (ORB-SLAM), lines, and planes across frames, and then use BA constraints to minimize projection or photometric errors. In this paper, the 3D semantic information detected in the map can not only play the role of the above several geometric features, but also provide additional semantic and geometric constraints to improve the pose of the camera.

![Figure 1. SLAM flowchart of our system](image-url)

The SLAM process of this system is shown in Figure 1: The input sequence is selected by ORB-SLAM for key frame selection, and the scene's semantic segmentation is performed in the corresponding point cloud data. This SLAM system is based on the ORB-SLAM2 framework based on feature points. It includes front-end camera tracking and back-end BA (Bundle Adjustment) optimization. The main part we changed is to consider the 3D objects, feature points and camera poses, and optimize them together in the back-end BA.
3. 3D semantic segmentation

Point cloud is a data format that effectively represents three-dimensional information, but the existing three-dimensional segmentation methods for point clouds either do not model local dependency relationships or require increased computation. We use a new 3D segmentation framework that can efficiently model local structures in point clouds. Our proposed method mainly includes three modules, namely the pooling layer of slices, the recurrent neural network (RNN) layer, and the pooling layer combination on slices. The slice pooling layer is designed to project the features of unordered points onto an ordered sequence of feature vectors, so that the network can learn end-to-end. The network framework is shown in Figure 2. The 3D point cloud segmentation network we use mainly includes four parts: feature extraction module, segmentation pooling layer, RNN layer and segmentation upper pooling layer.

![Figure 2. 3D point cloud segmentation network](image)

3.1 Feature extraction module

We use two feature extraction modules, corresponding to the input feature extraction module and the output feature extraction module. In both modules, we use a $1 \times 1$ convolutional layer to extract independent features for each point on the point cloud.

3.2 Dividing the pooling layer

The main purpose of this layer is to transform unordered point cloud data input features into ordered feature vector sequences. First, the point cloud data is segmented in three directions: x, y, and z. Taking the z axis as an example, according to the coordinate values on the z axis of the 3D point cloud data, the data points are evenly divided into N parts, and the resolution of each slice is controlled by the hyperparameter $r$. During the experiment, we set $r = 0.2 \text{cm}$, so we get 50 slices in three directions:
x, y, and z. For the features inside each slice, we use the maximum pooling method to get the global features of the final output. The feature vectors obtained by the above operations are ordered and structured, and local related information can be extracted in subsequent operations. The schematic diagram of the split pooling layer is shown in Figure 3.

3.3 RNN layer
RNN can be used to process sequence data, including time series and space series. We used bidirectional RNN (bidirectional RNN) to update features during the experiment. Bidirectional RNN considers the features of adjacent slices, thus achieving the purpose of extracting local correlation features. Here, 6 layers of RNN layers are used in the branches of each slice, and the number of channels is 256, 128, 64, 64, 128, 256 in turn.

3.4 The upper pooling layer
In the segmentation pooling layer, the features of multiple points in a slice are mapped into a global feature vector. Therefore, in the segmentation and pooling layer, the inverse process of mapping needs to be completed, as shown in Figure 4 below. The feature of a point corresponds to the global feature vector of the slice it belongs to.

During the experiment, we validated our method using the S3DIS dataset. The detection results obtained are shown in Table 1 for the 3D point cloud segmentation results. Figure 5 shows the 3D point cloud segmentation results.

| IoU  | ceiling | floor | wall | chair | table | sofa |
|------|---------|-------|------|-------|-------|------|
| 76.12% | 93.34% | 98.36% | 79.18% | 65.52% | 67.87% | 52.45% |
4. Drift Reduction Mechanism

After the semantic information in SLAM has been mentioned, we can optimize the unified semantic information, feature point information, and pose information. In the process, the detected three-dimensional objects, feature points and camera poses are taken into consideration, and the BAs integrated into the back end are optimized together.

4.1 Backend BA optimization cost function

BA is a process of jointly optimizing different map components, such as camera pose and feature points. Feature points are also used in most of our experiments because individual objects usually cannot completely limit the camera’s pose. Represent the set of camera poses, 3D cuboids, and feature points as $C = \{c_i\}$, $O = \{o_j\}$, and $P = \{p_k\}$, then BA can be expressed as a nonlinear least squares optimization problem:

$$C^*, O^*, P^* = \arg \max_{C, O, P} \sum_{i,j} e(c_i, o_j)^2 + \sum_{i,k} e(c_i, p_k)^2 + \sum_{j,k} e(o_j, p_k)^2$$

(1)

Among them, $e(c_i, o_j)$, $e(c_i, p_k)$, and $e(o_j, p_k)$ respectively represent the measurement errors of camera-object, camera-feature point, and object-feature point. $\Sigma$ is the covariance matrix of different measurement errors. Then use Gauss-Newton algorithm or Levenberg-Marquardt algorithm to solve the optimization problem of formula (1). This algorithm can be implemented using many libraries such as g2o or iSAM.

The camera pose is represented by $T_c \in SE(3)$, the points are represented by $P \in \mathbb{R}^3$, and the cuboid object is modeled as 9 DoF parameters: $O = \{T_o, d\}$, where $T_o = [R, t] \in SE(3)$ is a pose of 6
DoF, and \(d \in \mathbb{R}^3\) is the length of the three sides of the cuboid. In some cases (such as KITTI) we can also use the length of the provided object, so there is no need to optimize \(d\), and the subscript \(m\) represents the measured value. The coordinate system of this system is shown in Figure 6.

\[
6
\]

Figure 6. The coordinate system of this system and the measurement errors between the camera, feature points and objects

4.2 Measurement error

4.2.1 Camera-object measurement: Two measurement errors between the object and the camera are proposed.

\(a\) Three-dimensional measurement: First, when the detection result of the three-dimensional object is accurate, such as using the RGBD sensor for three-dimensional measurement. The object target pose \(O = (T_{om}, d_m)\) detected in the semantic segmentation in the previous section is used as the measurement amount of the object target in the camera coordinate system. In order to calculate its measurement error, we convert the corresponding landmark object to the camera coordinate system and then compare it with the measurement result:

\[
\begin{align*}
  e_{o_3} &= \log((T_{c}^{-1}T_{o})^\top_{se3} d - d_m) \\
  e_{o_3} &\in \mathbb{R}^9
\end{align*}
\]

Among them, \(\log\) maps the SE3 error into the tangent vector space of 6DoF, so \(e_{o_3} \in \mathbb{R}^9\). Huber’s robust cost function is applied to all measurement errors to improve robustness.

\(b\) Two-dimensional measurement: In two-dimensional measurement, we projected the cuboid landmark onto the image plane to obtain the red rectangular two-dimensional bounding box shown in Figure 4, and compared it with the two-dimensional bounded box detected by blue. The specific method is to project 8 corners onto the image, find the minimum and maximum values of the x and y coordinates of the projected pixels, and form a rectangle:

\[
\begin{align*}
  [u, v]_{\text{min}} &= \min\{\pi(R(\pm dx, \pm dy, \pm dz)/2 + t)\} \\
  [u, v]_{\text{max}} &= \max\{\pi(R(\pm dx, \pm dy, \pm dz)/2 + t)\} \\
  c &= ([u, v]_{\text{min}} + [u, v]_{\text{max}})/2 \\
  s &= [u, v]_{\text{max}} - [u, v]_{\text{min}}
\end{align*}
\]

\([u, v]_{\text{min}, \text{max}}\) is the minimum / maximum xy coordinates of the eight projection angles, that is, the upper left and lower right corners of the projection rectangle. \(c\) and \(s\) are the center and size of the 2D box. They are two-dimensional vectors, so \([c, s] \in \mathbb{R}^4\). Then define the four-dimensional rectangle error as:

\[
e_{o_2} = [c, s] - [c_m, s_m]
\]

Compared with the 3D error in formula (2), the uncertainty of this measurement error is much smaller, because 2D target detection is usually more accurate than 3D detection. This is similar to projecting a map point onto an image to form a reprojection error.
4.2.2 Object-feature point measurement: Points and objects can provide constraints to each other. If the point P belongs to an object on the object, it should be located in a three-dimensional cuboid. So we first transform the points into the coordinate system of the cuboid, and then compare it with the size of the cuboid to get the three-dimensional error:

\[ e_{op} = \max(|T_0^{-1}p| - d_m, 0) \]

We use \( \max \) because we only require the points to be inside the box, not exactly on the surface.

4.2.3 Camera-feature point measurement: We use standard 3D point reprojection error in feature-based SLAM.

\[ e_{cp} = \pi(T_c^{-1}P) - z_m \]

Where \( z_m \) is the coordinate on the image of the three-dimensional point \( P \).

5. Experiments
Each figure should have a brief caption describing it and, if necessary, a key to interpret the various lines and symbols on the figure.

Finally, we evaluated the performance of the improved SLAM in the public data set. The root mean square error (RMSE) and the KITTI translation error are used to evaluate the camera pose. We selected 10 KITTI raw sequences, of which the most detail labeled ground truth is the "2011_0926_00xx" fragment. Real camera pose is provided by GPS / INS from the car. The experiments were performed on an Intel i7-4790 CPU with a 4.0 GHz processor. The running interface is shown in the figure.

Figure 7. Our SLAM operation interface
Figure 8. ORB-SLAM operation interface
Figure 7 is our semantic SLAM operation interface. The left side is the detected image. The box represents the detected car and its frame. After adding these semantic information to SLAM's BA optimization, the map information can be obtained. Figure 8 is the running interface of the mainstream ORB algorithm.

![Figure 9. Path comparison between semantic-based SLAM and ORB-SLAM](image)

The comparison of the two paths is shown in Figure 9. Red is the reference position provided by the vehicle's GPS / INS, green is the ORB-SLAM path, and blue is the path generated by our SLAM based on semantic information. The drift error compared with the current mainstream ORB-SLAM algorithm is shown in Table 2.

| Sequence | 22 | 23 | 36 | 39 | 61 | 64 | 93 | 95 | 96 | 117 | Average |
|----------|----|----|----|----|----|----|----|----|----|-----|---------|
| **Translation error(%)** | ORB-SLAM | 13.0 | 1.17 | 7.08 | 6.67 | 1.06 | 7.07 | 4.40 | **0.86** | 3.96 | 4.10 | 4.95 |
| OUR-SLAM | 11.68 | 1.72 | **5.93** | **4.61** | 1.24 | **5.93** | **3.60** | 1.49 | **1.81** | 2.21 | **4.02** |

As shown in Table 2, after the data association and BA optimization of most sequences, in the camera pose estimation, the SLAM based on semantic constraints in this paper can provide more geometric constraints to reduce the monocular SLAM drift. The translation error was reduced from 4.95 to 4.02, a decrease of 18.8%.

6. Conclusion and discussion
This paper proposes a BA-based drift reduction mechanism based on semantic information. This mechanism takes into account the three-dimensional objects, feature points, and camera poses that are semantically recognized in the scene, and optimizes them into the back-end BA to reduce the translation drift error of SLAM by 18.8%. However, there are some sequence results, compared with the ORB-SLAM method, our SLAM method has lower accuracy than the latter (such as sequences 23, 61, 95). The reason for this result may be the error of the detection of the three-dimensional box of the object, which leads to more noise information in the joint BA optimization in the later stage. Therefore, the optimization result is not as good as the BA optimization using only the feature points. In the future, more work can be done to improve the accuracy of object recognition to improve the results.
References

[1] Mozos O M, Triebel R, Jensfelt P, et al. Supervised semantic labeling of places using information extracted from sensor data[J]. Robotics and Autonomous Systems, 2007, 55(5): 391-402.

[2] Nüchter A, Hertzberg J. Towards semantic maps for mobile robots[J]. Robotics and Autonomous Systems, 2008, 56(11): 915-926.

[3] Civera J, Gálvez-López D, Riazuelo L, et al. Towards semantic SLAM using a monocular camera[C]//2011 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2011: 1277-1284.

[4] Vineet V, Miksik O, Lidegaard M, et al. Incremental dense semantic stereo fusion for large-scale semantic scene reconstruction[C]//2015 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2015: 75-82.

[5] McCormac J, Handa A, Davison A, et al. Semanticfusion: Dense 3d semantic mapping with convolutional neural networks[C]//2017 IEEE International Conference on Robotics and automation (ICRA). IEEE, 2017: 4628-4635.

[6] Yu J S, Wu H, Tian G H, et al. Semantic database design and semantic map construction of robots based on the cloud[J]. Robot, 2016, 38(4): 410-419.

[7] Long J, Shelhamer E, Darrell T. Fully convolutional networks for semantic segmentation[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 3431-3440.

[8] Salas-Moreno R F, Newcombe R A, Strasdat H, et al. Slam++: Simultaneous localisation and mapping at the level of objects[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2013: 1352-1359.

[9] Hermans A, Floros G, Leibe B. Dense 3d semantic mapping of indoor scenes from rgb-d images[C]//2014 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2014: 2631-2638.

[10] Concha A, Civera J. DPPTAM: Dense piecewise planar tracking and mapping from a monocular sequence[C]//2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2015: 5686-5693.

[11] Mur-Artal R, Tardós J D. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras[J]. IEEE Transactions on Robotics, 2017, 33(5): 1255-1262.