Deep Learning Methods for Semantic Segmentation of Dense 3D SLAM Maps

Pei Yingjian
MIST, Kyushu Institute of Technology, 680-4 Kawazu
Iizuka-shi, Fukuoka 820-8502, Japan

Sakmongkon Chumkamon
MIST, Kyushu Institute of Technology, 680-4 Kawazu
Iizuka-shi, Fukuoka 820-8502, Japan

Eiji Hayashi
MIST, Hayashi Lab, 680-4 Kawazu
Iizuka-shi, Fukuoka 820-8502, Japan
E-mail: yingjian.pei801@mail.kyutech.jp, m-san@mmcs.mse.kyutech.ac.jp, haya@mse.kyutech.ac.jp
www.kyutech.ac.jp

Abstract

Most real-time SLAM systems can only achieve semi-dense mapping, and the robot lacks specific knowledge of the mapping results, so it can only achieve simple positioning and obstacle avoidance, which may be used as an obstacle in the face of the target object to be grasped, thus affecting the realization of motion planning. The use of semantic segmentation in dense SLAM maps allows the robot to better understand the map information, distinguish the meaning of different blocks in the map by semantic labels, and achieve fast feature matching and Loop Closure Detection based on the relationship between semantic labels in the scene. There are many semantic segmentation datasets based on street scenes and indoor scenes available for use, and these datasets have some common tags. Based on these training data, we can derive a semantic segmentation model based on RGB images by using the Pytorch platform for training.

Keywords: 3D SLAM, Semantic Segmentation, Point Cloud, ROS

1.Introduction

Active obstacle avoidance motion planning is an important element in the autonomous motion planning of the robot arm. Active obstacle avoidance can not only avoid collision damage during the operation of the robot, but also improve the robot's ability to sense the environment and avoid causing safety accidents. In active obstacle avoidance motion planning, one of the conditions is the level of the robot's perception of the workspace. Effective obstacle avoidance motion planning can only be achieved if the robot has a prior awareness of the obstacles in the workspace and is able to update environmental information at any time during the work. The advantage of 3D SLAM is that the robot can get the complete spatial information of the current environment, and get the abstract modeling of the real environment in the virtual environment through the octomap, and through the Rviz plugin of ROS platform, the obstacle avoidance motion planning can be realized with the octomap as the reference.

For semantic segmentation of dense maps, we can either use a direct segmentation method on 3D point cloud data or a semantic segmentation method based on 2D RGB images, and we use the second method due to the convenience of training data. We use the second method for the convenience of training data. There are many semantic segmentation datasets based on street

© The 2021 International Conference on Artificial Life and Robotics (ICAROB2021), January 21 to 24, 2021

scenes and indoor scenes available for use, and these datasets have some common tags. Based on these training data, we can derive a semantic segmentation model based on RGB images by using the PyTorch platform for training.

There has been some progress in scene marking research based on this approach, and our project has now applied some of this technology to detect target types in 3D point cloud data. Combining this progress with obstacle avoidance motion planning, the robot will be able to accurately distinguish between grasping and obstacle avoidance based on an understanding of the meaning of the scene, and it will be easier to integrate the two systems into the same framework.

2.Segmentation Model Training

We refer to the work of Xuan Zhang et al. in selecting the semantic segmentation model and improve it to some extent based on our system. Our study was mainly trained using the ade20k dataset, a training set that includes a large number of images and labels of common household items and can be applied to most scenarios. For the laboratory scenario we will use, we used annotation software to additionally annotate a portion of the images and add them to the dataset for training. Figure.1 shows Example image of our own data.

In terms of neural network selection, we chose PSPNet as the basis, which is very suitable for semantic segmentation work such as scene parsing, and demonstrated a certain degree of accuracy when tested in real scenes using pre-trained models. We used the PyTorch platform to initially train the dataset with PSPNet50. Due to the limited GPU computing resources and the complex environment during the initial training, the semantic segmentation effect was not perfect, but a clear segmentation boundary could be demonstrated under sufficient lighting conditions.

2.1. *Segmentation Models Comparison*

In choosing the model, we mainly examined the training accuracy of different neural network models on the ade20k dataset, and the indoor part of the ade20k dataset was chosen for testing, mainly because our project is based on indoor scenes, and the part of the dataset chosen cannot accurately evaluate the accuracy of some labels, but it can greatly accelerate the training.

Figure.1 represents the real-time display of 2D semantic segmentation under different models, from top to bottom, the original RGB image, PSPnet_50, PSPnet_50 with Bayesian filter on, and the unmodified ResNet_50 image. From the comparison graphs, it can be seen that Bayesian filter does not contribute much to the semantic segmentation accuracy of indoor scenes, while PSPnet, as an improved network based on ResNet, reflects better results on ade20k dataset. It can be seen from the figure that PSPnet accurately identifies the screen that appear only partially in the scene (green part), while the original ResNet is more ambiguous.

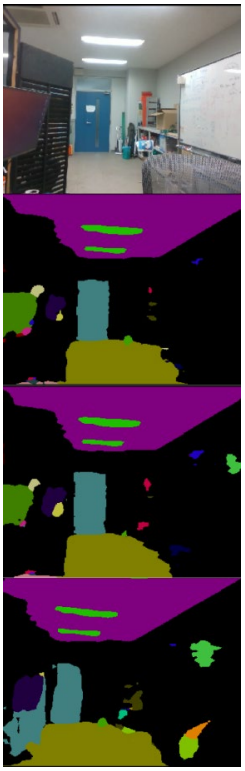

Figure. 1 Comparison between different models

Table.1 Comparison of segmentation speed (FPS)

Model	FPS (Average)
PSPnet_50	20
PSPnet_50 (Bayesian filter)	15
BasicResNet_50	23

2.2. *Segmentation Speed Comparison*

Since SLAM pursues real-time, accuracy and robustness, the frame rate should be used as the reference value for speed determination. The comparisons in Table.1 are all made under Ubuntu 18.04, ROS Melodic environment using CPU calculations, and the relevant point cloud

generation module is not activated to save system resources during the frame rate comparison.

2.3. *Conclusion of Model Selection*

It can be seen from the table that turning on the Bayesian filter will have a significant impact on the frame rate, and the original ResNet is better than PSPnet in terms of fluency. Taking all factors into consideration, the PSPnet semantic segmentation model without bayesian filter has achieved a good balance between accuracy and speed, and we will use this scheme in the actual test.

3. Overall System Construction

In this section, we will discuss the setting up of the experimental environment and camera selection. The first section will give a general overview of the experimental environment and the experimental format, and the second section will present our considerations when selecting a depth camera.

3.1. *Environment Construction*

In order to allow flexible observation of the workspace by depth cameras, we placed the depth camera for SLAM at the manipulator of the robot arm, and to compensate for blind spots in the observation, we also placed a plurality of depth cameras at other locations in the workspace. In the following sections, we will mainly demonstrate and illustrate the main camera fixed on the manipulator.

After our depth camera acquires the semantic segmentation image, it publishes and projects the 2D image to the point cloud, and generates Octomap using the point cloud image.

Figure.2 shows how the system looks like in a rviz UI in ROS, you can see Semantic Image on the corner and a octomap view in the main window.

Figure.3 shows the overall view of system frame.

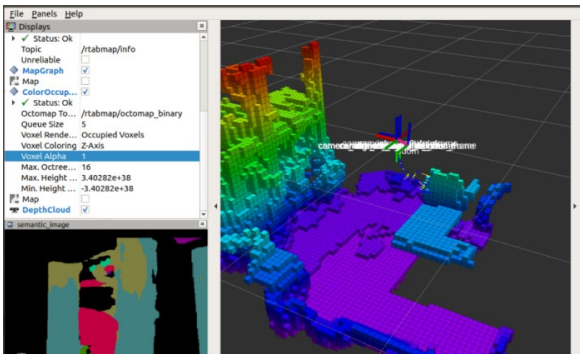

Figure. 2 Rviz scene demo

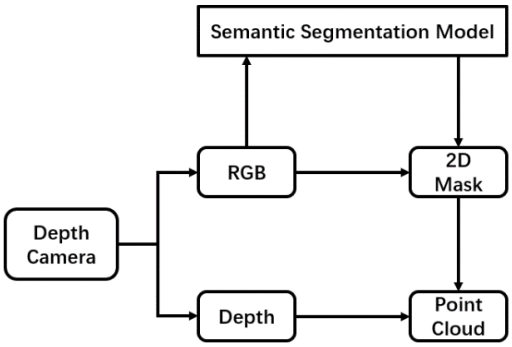

Figure. 3 Conceptual Frame

3.2. Camera Selection

In terms of camera selection, Intel Realsense D400 series was chosen for this study. The D435i with IMU module and the D435 without IMU module were used in practice and they both performed consistently in the tests. The reason for using D435 series is that when building 3D SLAM maps, it is necessary to provide accurate depth data and clearer images for feature recognition and reference, and the overall size of the camera has to be taken into account as well, so the camera should not be too heavy or too big to be mounted on the operation end of the robot arm. The D435i, which comes with an IMU module, is useful for 3D SLAM mapping and offers more possibilities for optimization in subsequent development.

4. Mapping

The platform we use is ROS Melodic based on Ubuntu 18.04, and the graphical interface is ROS Rviz, which is also used to facilitate the subsequent obstacle avoidance motion planning.

Due to the relatively small workspace used in this study, some of the parameters of the official launch file are not applicable to the realities of this study, so we have adjusted some of the parameters.

Through the actual test, the overall map building resolution is relatively high and the map building is nearly perfect. The semantic octomap built can be successfully saved through the relevant functions of octomap_server, and can be published directly in octomap_server with .bt format into Marker Array Topic under the condition that the actual working scene of the robot remains unchanged.

After completing the scene mapping, the obstacle avoidance motion planning is performed using the RRTconnect algorithm in MoveIt! and output directly to

the Motoman robot arm. In the Gazebo simulation environment, the obstacle avoidance planning effect on octomap is good.

In practice, due to the problem of hand-eye calibration and coordinate conversion, we can use static coordinate conversion to fix the coordinates of the camera and robot model to realize the map building and obstacle avoidance function in MoveIt! It is also possible to save and publish octomap at the same time to realize real-time map building and obstacle avoidance planning in different Rviz terminals.

Since the actual runtime is based on the Python 2 platform, and the latest version of PyTorch no longer supports Python 2, we use the older version 0.4.0, which is only supported by the CPU, and therefore runs with a certain degree of lag and has some shortcomings in terms of time efficiency.

5. Conclusion

After simulation and practical testing, the deep learning method can be integrated with our system to a certain extent, which is of great help in the subsequent development of intelligent control for Motoman robot. Semantic 3D SLAM's mapping results are in line with expectations, the obstacle avoidance function works normally, and the constructed map can be reused, which eliminates the need to model the scene in a virtual environment. This module will continue to be refined and developed as part of the project.

6. Discussion & Future Works

In practical tests, we also found the following problems:
1) Inefficient map building. When using the deep learning method described in this paper, because the feature point matching method not use the semantic data to match the point, and the rate at which the camera extracts keyframes from the video stream depends on the processing speed of the computer and the camera, in the case where the robot arm moves too fast, the camera will collect two pictures with almost no similar features, and thus cannot form a closed loop to build a map. In the actual test we have to use slower movement speed to get the complete map, which affects the overall map building efficiency.
2) Due to the existence of multiple coordinate nodes in Motoman's virtual model, multiple coordinate conversions are required when binding the camera coordinates, which will result in the camera coordinates being incorrectly bound to the world coordinate system or the overhang position. We use a compromise between opening two terminals to create the final effect, and using fixed coordinates is also a last resort.
3) We placed more than one depth camera in the workspace, but we only used one of them in practice, and we hope that in the future we will be able to integrate the depth data from all the cameras into a complete map.

Our next plan is to replace the modules with models trained by other deep learning methods to improve the efficiency of map building. We plan to use Siamese Neural Network-based semantic feature recognition matching algorithms and implement the object-level building, and we refer to the methods used in Ref. 4 and Ref. 5 for this work.

References

1. Labbe, Mathieu & Michaud, François. (2013). Appearance-Based Loop Closure Detection for Online Large-Scale and Long-Term Operation. Robotics, IEEE Transactions on. 29. 734-745. 10.1109/TRO.2013.2242375. (IEEE Xplore)
2. Zhao, H., Shi, J., Qi, X., Wang, X., & Jia, J. (2017). Pyramid Scene Parsing Network. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 6230-6239.
3. Xuan, Zhang and David, Filliat, Real-time voxel based 3D semantic mapping with a hand held RGB-D camera, 2018, GitHub, GitHub repository, \url{https://github.com/floatlazer/semantic_slam
4. Hu, L., Xu, W., Huang, K., & Kneip, L. (2019). Deep-SLAM++: Object-level RGBD SLAM based on class-specific deep shape priors. ArXiv, abs/1907.09691.
5. Puligilla, Sai Shubodh, Satyajit Tourani, Tushar Vaidya, Udit Singh Parihar, Ravi Kiran Sarvadevabhatla and K. Krishna. “Topological Mapping for Manhattan-like Repetitive Environments.” 2020 IEEE International Conference on Robotics and Automation (ICRA) (2020): 6268-6274.