Real-time instance-aware semantic mapping

X C Xu, L L Liu, R Xiong and L Jiang
Zhejiang University, Hangzhou, 310027, China

Email: xch_xu@outlook.com

Abstract. The semantic information helps robots to understand its surroundings like human beings and enables robots to achieve human-robot interaction. In recent years, there have been many interests in semantic mapping. Numerous approaches manage to build a semantic map and achieve good accuracy, but the existing mapping methods which create the metric semantic map ignore the subsequent applications of the semantic map. However, the metric map with the simple semantic class label has no direct benefit to localization. In this paper, we propose an approach to construct an object-centric map with promising applications. Employing the traditional metric and deep learning methods, we can extract objects from the environment and along with semantics. This object representation of the semantic map can be useful in other applications of robots, our local map and global map framework can be useful for navigation. At last, we report on the quality and speed of our object-centric mapping method.

1. Introduction
The map is the basis of the mobile robot movement and its localization and navigation. On this basis, the semantic map provides the category information of objects in the surrounding environment for the intelligent mobile robot so that the robot can intuitively understand the scene. It is of great value to study the method of semantic mapping in robot localization, navigation, and human-robot interaction. Furthermore, for mobile robots, one crucial constraint is real-time demand. For the robot to interact with the environment quickly, the mapping module must not consume much time.

With this background, this paper proposed a real-time framework to construct an object map within the city road scene. The framework contains lightweight algorithms such as traditional point clustering and object detection methods. Our system also applies the 2.5D representation of the environment to reduce storage consumption because of the large-scale scene in the city. Our system uses a hash table as the structure of data storage to avoid dependence upon CPU memory. With all these methods, the framework can be running at ~10fps. Besides the real-time feature, the proposed method is also born with a feature that can be used in navigation. The local traversable map, along with the global semantic map, can lead the robot to a more semantic interaction with people.

The framework uses a lidar-IMU unified method for localization and a camera for object detection. The lidar-IMU unified localization estimates the motion of the robot and enables the multiple observation of the object and improves the accuracy of the object-instance recognition. The camera provides abundance textures of the environment, which can be used to detect objects.
2. Related Works

2.1. Semantic Mapping

Nüchter [1] first proposed the idea of semantic mapping and the method is first to train object contour classifiers to semantically label objects in the scene and uses a pre-built object point cloud database to train the contour classifier to identify objects. After recognition, the model and the corresponding objects in the database are subject to Iterative Closest Point (ICP) evaluation. Salas-Moreno built a database of repetitive objects based on prior knowledge to perform semantic mapping at the object level. The point pair feature (PPF) is used to match the point cloud of the scene constructed by the RGBD camera with the object model in the database. Once a match is found, the pre-stored object model is placed on the map to achieve the semantic labeling of the map.

Fiorao [2] proposed a joint prediction of object and camera pose. Different from Salas-Moreno for object matching directly at the point cloud level, this method matches the image obtained by the camera in the visual SLAM with the 2D or 3D features of the model in the pre-built database and adds the matching error to the map model and the camera. The poses are optimized together, and a beam adjustment method (Bundle Adjustment, BA) at the semantic level is implemented.

Unlike mapping methods based on the database, the segmentation-based semantic mapping methods mainly use image segmentation algorithms and point cloud segmentation algorithms to semantically segment a single frame of data, and then use SLAM's pose estimation to map the segmentation results to a three-dimensional space to implement semantic mapping. The reliability of data association in the method is a key issue.

Stuckler [3] proposed using a random decision forest algorithm to classify two-dimensional images at the pixel level. By introducing depth information during the training process, a scale-invariant classifier that seamlessly fuses shape and texture features can be obtained. A pixel-level semantically labeled image is completed after the classifier, and a semantic 3D voxel model of the scene is generated according to the Bayesian criterion.

Hermans [4] used the method of image segmentation to perform 2D segmentation to 3D semantic mapping in a progressive manner. In this paper, a random decision tree is used for rough segmentation of 2D images, and then fine segmentation is performed by a fully connected conditional random field (DenseCRF) algorithm. The obtained results are combined with the pose obtained by the visual odometry to perform the 3D reconstruction.

Sunderhauf [5] combined an unsupervised 3D point cloud segmentation method with a single-shot Multi-box Detector (SSD) target detection algorithm, which not only achieved pixel-level semantic segmentation but also stored the object's point cloud. As an element of the map. The data association between the observation object and the object database in this paper is similar to the ICP method. The Euclidean distance between the point clouds of the two objects is calculated to determine whether they are the same object.

Semantic mapping methods based on image segmentation mostly involve machine learning, which requires higher computing resources. The semantic maps constructed by this method are mostly 3D point cloud maps with semantic annotations. In the subsequent positioning stage, how to use the 3D point cloud maps with semantic annotations is an issue to be considered.

2.2. 2D Recognition

CNN models achieve great performance in object detection tasks and exploiting semantic information from an input image. The SOTA segmentation models including Pointnet[6] and object detectors like YOLO v3 [7], SSD [8]. Both deliver outstanding performance and some of them can be used in the real-time system.

3. Methods

In this section, detailed methods will be introduced. The methods mainly about four aspects: object detection, localization, geometry segmentation fusion, and mapping.
3.1. Object Detection
To get the semantic information from the streaming image use a YOLO v3 framework integrated into the ROS system, to detect objects during the movement of the car.

The YOLO v3 detector typically relies on a single view for casting and validating hypotheses, so it may produce fault semantic information in the mapping system. To overcome this, we use the detection result in a constraint way. Mainly, we only use the detection result with the probability higher than 90% and filtered those detections far away from the camera. This consideration is based on that the far-away object is too small for object detector to extract validate features for classification.

3.2. Localization
We adopt the online ICP [9] based lidar-IMU front-end to localize the robot position, the poses are updated under the Extended Kalman Filter paradigm. By apply the pre-integrate algorithm of IMU data, the output localization can be running at a high rate. We focus on semantic and map representation including local mapping and global mapping. The localization module is off-shelf methods, which we will not consider in our system performance.

3.3. Mapping
In the mapping module, we explore the vast case of the application of the map. The widely used metric map is not informational for a robot to interact with people. However, the metric map has its advantages. A metric map can be used for localization or navigation. In the previous work of the semantic mapping, the object is much highlighted, which might neglect the usage of the map.

We introduce a local mapping with a dense 2.5D grid representation [10] for robot navigation and a global mapping for semantic understanding of the environment.

The map contains calculated information that can be used for geometry segmentation. In the fusion step, the geometry segmentation will improve the accuracy of the map. The calculated information is the surface normal vector and neighbor height deviation.

3.3.1. Surface normal vector Using SVD method, the normal vector can be obtained from the noisy surface representation of the environment. The normal vector then assigned to a specific grid on our 2.5D grid map. A classical calculate method is to shift the origin to $p_i$ which contains three-dimensions information of $P_{(x,y)}$ and fit a plane $S_i = n_{ix}x + n_{iy}y + n_{iz}z$ with surface points $Q_i$ in the neighboring area. i.e, solve:

$$\min_{x} \| [Q_i - 1_kp_i^T]n_i \|_2$$

(1)

Where k represents the num of points in the neighboring region, $Q_i$ is a 3*N matrix with three-dimensions coordinates of K neighbor points and $n_i = [n_{ix},n_{iy},n_{iz}]$ is a surface normal vector. This optimization problem can be solved by computing the SVD of $Q_i - 1_kp_i^T$. The minimizer $n_i$ is the vector in V that corresponds to the smallest singular value in $\Sigma$; thus, the normal vector is derived.

3.3.2. Neighbor height deviation It's common to use the Neighbor height deviation in the evaluation of the roughness. The deviation can be derived using the formula below:

$$H_d = |h_{P_{(x,y)}} - \bar{h}|$$

(2)

Where $h_{P_{(x,y)}}$ is the height of the cell $P_{(x,y)}$ and $\bar{h}$ is the mean height of all the grids in an N*N window centered on the cell $P_{(x,y)}$.

3.3.3. Dynamic objects Any dynamic objects will be integrated into the map and will cause a disastrous fault. The mapping method should be careful with those dynamic obstacles.
situation when there are dynamic obstacles, such as pedestrians and moving vehicles leaving the original location, the original height of the cell still retains. It causes a terrible error because the drivable region will be miscalculated. Ray tracing can be used to deal with the problem of the error height value produced by the dynamic objects. An object can be observed by an important principle: there are no other obstacles between the sensor and the object. We use real-time lidar data to check if the existing cell in the elevation map blocking the light path, and if so, clear it. For a dense map with drivable regions, we only need to check the cell with a high score of the traversability.

3.4. Geo-semantic fusion
The raw object detection result is not accurate in the object edge, which will cause semantic information mixed in the map. So we come up with a simple but efficient way to improve the accuracy of the semantic map.

The base idea of the improvement is based on the point cloud acquired by lidar. The multi-modal information may help to improve each other. The point cloud representation contains much information about geometry. We use a clustering algorithm to segment the objects. With the traversability information get by the local mapping process, we have a raw segmented point cloud. The raw point cloud includes all objects’ semantic information.

4. Experiments
Our experiment is conducted on PC with Nvidia GPU RTX2060 and AMD 3700X. Also, we test the framework on TX2, a common platform for mobile robots. Our test dataset is KITTI [11] car datasets containing lidar, camera and IMU.

First, we evaluate our object detection module, we adopt YOLO v3 under the ROS framework and test it. The performance of YOLO is shown below and it can be running on real-time ~30fps on our experiment platform.

![Figure 1. Object detection by YOLO v3.](image)

Our mapping module process a 3D point cloud into a 2.5D grid map, it will cost about 50ms. The local mapping is shown in figure 2 with red and green grids. The red region is represented as an area that has a high traversable score, which can be passed for robots and cars, while the green area is obstacles. The local mapping module is implemented on the GPU, which can be running in real-time.
The global mapping method simply adds the outdated local map to a point cloud map with semantic information. The real-time visualization with trajectory is shown below. The global mapping module consumes little computation resources and time.

Figure 2. Local mapping in 2.5D grid map.

The point cloud segmentation is efficient in removing outliers in semantic point cloud acquired by the global mapping step. The different clusters are shown in a different color.

Figure 3. Local mapping in 2.5D point cloud map.
We evaluate the time consumption of all steps in our framework and shown in table 1.

**Table 1.** Time consumption of different steps in our system.

|                         | PC (ms) |
|-------------------------|---------|
| Object detection        | 30      |
| Point cloud clustering  | 23      |
| Local mapping           | 78      |
| Global mapping + Fusion | 34      |

The final semantic map combining the raw object detection result and point cloud segment can be very clear in the object, with this clear point cloud, we can also perform object pose estimation on it. Along with this object map, we can also get a global colored map.
5. Conclusion
In this article, we propose a real-time semantic object map method with additional information about drivable navigation information. The method is based on multi-modal data such as lidar, IMU and camera. With the real-time object detection and clustering pipeline, we fuse the semantic information into our global object-centric map. The output of our system can generate cost maps for classic planning algorithms, such as A* and RRT. Also, this map representation can be used in a more complex motion planning situation of quadruped robots.

References
[1] Nüchter A and Hertzberg J 2008 Robotics and Autonomous Systems, Elsevier B.V. 56 915–926.
[2] Fioraio N and Di Stefano L 2013 Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition 1538–45
[3] Ma L, Stuckler J and Kerl C 2017 IEEE International Conference on Intelligent Robots and Systems 598–605
[4] Hermans A, Floros G and Leibe B 2014 Proceedings - IEEE International Conference on Robotics and Automation 2631–8
[5] Sunder hauf N, Pham T T and Latif Y 2017 IEEE International Conference on Intelligent Robots and Systems 5079–85
[6] Qi C R, Su H, Mo K, et Al 2017 Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition
[7] Redmon J, Farhadi A 2017 Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition
[8] Liu W, Anguelov D and Erhan D 2016 Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 21–37
[9] Yang J, Li H and Campbell D 2016 IEEE Transactions on Pattern Analysis and Machine Intelligence 38 2241–54
[10] Souza A and Gonçalves L M G 2016 Robotica 34 2592–609
[11] Geiger A, Lenz P and Stiller C 2013 International Journal of Robotics Research