sensors

MDPI

Article

Multi-Objective Location and Mapping Based on Deep Learning and Visual Slam

Ying Sun [1,2,3], Jun Hu [1,2,*], Juntong Yun [1,2], Ying Liu [1,2], Dongxu Bai [1,2], Xin Liu [1,2], Guojun Zhao [1,2], Guozhang Jiang [1,2,3], Jianyi Kong [1,2,3] and Baojia Chen [4]

1 Key Laboratory of Metallurgical Equipment and Control Technology, Ministry of Education, Wuhan University of Science and Technology, Wuhan 430081, China
2 Research Center for Biomimetic Robot and Intelligent Measurement and Control, Wuhan University of Science and Technology, Wuhan 430081, China
3 Hubei Key Laboratory of Mechanical Transmission and Manufacturing Engineering, Wuhan University of Science and Technology, Wuhan 430081, China
4 Hubei Key Laboratory of Hydroelectric Machinery Design and Maintenance, China Three Gorges University, Yichang 443002, China
* Correspondence: hujun1996@wust.edu.cn

Abstract: Simultaneous localization and mapping (SLAM) technology can be used to locate and build maps in unknown environments, but the constructed maps often suffer from poor readability and interactivity, and the primary and secondary information in the map cannot be accurately grasped. For intelligent robots to interact in meaningful ways with their environment, they must understand both the geometric and semantic properties of the scene surrounding them. Our proposed method can not only reduce the absolute positional errors (APE) and improve the positioning performance of the system but also construct the object-oriented dense semantic point cloud map and output point cloud model of each object to reconstruct each object in the indoor scene. In fact, eight categories of objects are used for detection and semantic mapping using coco weights in our experiments, and most objects in the actual scene can be reconstructed in theory. Experiments show that the number of points in the point cloud is significantly reduced. The average positioning error of the eight categories of objects in Technical University of Munich (TUM) datasets is very small. The absolute positional error of the camera is also reduced with the introduction of semantic constraints, and the positioning performance of the system is improved. At the same time, our algorithm can segment the point cloud model of objects in the environment with high accuracy.

Keywords: deep learning; target tracking; visual SLAM; multi-objective location; semantic mapping

Citation: Sun, Y.; Hu, J.; Yun, J.; Liu, Y.; Bai, D.; Liu, X.; Zhao, G.; Jiang, G.; Kong, J.; Chen, B. Multi-Objective Location and Mapping Based on Deep Learning and Visual Slam. *Sensors* **2022**, *22*, 7576. https://doi.org/10.3390/s22197576

Academic Editors: Tao Peng and Yanhua Shih

Received: 12 September 2022
Accepted: 3 October 2022
Published: 6 October 2022

1. Introduction

With the development of artificial intelligence and neural network, the status of simultaneous localization and mapping technology has been growing, and now, it has become a research hotspot in the field of computer vision and robotics, which has made remarkable achievements in the scientific and academic fields [1–4]. SLAM has been applied in many aspects of our lives, such as medical services, virtual/augmented reality, drones, autonomous driving, etc. [5,6]. Due to the rapid development of computer vision and deep learning, the use of semantic information in SLAM, which is called semantic SLAM, has become a very promising and challenging topic [7–13]. Semantics and SLAM seem to be two independent modules, but they are not. They influence and help each other in many ways [14–16]. On the one hand, the mapping and positioning of the classical SLAM system are mostly based on the geometric matching at the pixel level, lacking a priori information to help the sensor locate the objects [17,18]. Data association can be improved from the traditional pixel level to the object level with the help of semantic information, which can help improve the positioning accuracy in complex scenes [19–21]. Compared

with the traditional physical geometric information, semantic information has consistency and stability and is less affected by environmental transformations. Additionally, it is suitable for image matching in both static and dynamic environments. On the other hand, the semantic information obtained by the semantic segmentation algorithm for the same object may be different at different times and from different angles, especially for objects with similar shapes. The SLAM system can provide constraints for the extraction of semantic information to improve the accuracy of semantic understanding [22]. In general, it is computationally cheaper to accomplish data association and camera pose estimation through objects than through feature points, as there are many fewer objects compared to the number of 3D points in a map.

Existing SLAM systems can already construct traditional two-dimensional (2D) maps of the environment, and many are capable of building 3D sparse point cloud maps and dense point cloud maps of the scene. With these maps, robots can carry out low-level tasks, such as localization and navigation [23,24]. However, sophisticated interactions between a robot and the surroundings require three-dimensional (3D) semantic information about the environments. The image processing methods based on deep learning, including target detection, semantic segmentation and instance segmentation, are important and promising steps to address this problem. Combined with semantic information, the SLAM system can help mobile robotics locate itself more precisely and significantly improve the ability to interact with the environment, which means that a task like "bring me an apple on the table" can be performed effectively [25]. The concept of semantic maps was first introduced at the beginning of the 21st century [26]. Most of the current semantic maps built by semantic SLAM are scenario-oriented semantic maps. The scenario-oriented semantic maps belong to pixel-level 3D semantic maps where each map point carries semantic information, making it easier for robots to understand the scene, such as background, wall, table, door and so on [27–29]. This way of building maps can construct a more expressive map of the environment to assist the robot to better understand the environment as a whole, but it is not good for the robot to recognize the individuals in the environment. Robots cannot interact with individuals in the environment because they cannot distinguish between different individuals in the environment, which limits the application of intelligent robots to a certain extent [30–32]. The main contributions of this paper are as follows:

(1) Propose a method of target tracking combined with instance segmentation, which can segment the objects in an image and track each object to establish data associations for different frames.
(2) Propose a method for constructing objected-oriented semantic maps.
(3) Propose a method for reconstructing multiple objects in 3D space.
(4) Propose a method for optimizing the camera pose using object constraints.

In the rest of the paper, the second part is related works, the third part is the method and theory, the fourth part is the experiment and analysis, and the last part is summary and expectations.

2. Related Work

2.1. Image Segmentation Networks

Semantic segmentation goes further on the basis of target detection and needs to distinguish each pixel point in the image to segment and distinguish different classes of objects. Instance segmentation pursues more precise segmentation based on semantic segmentation to distinguish each object of the same category. Starting from the proposal of fully convolutional networks (FCN) in 2015, the classical neural-network-based image segmentation algorithms start to appear. Long et al. propose FCN, which replaces the full convolutional layer with the full connection layer in the convolutional neural network (CNN) to build an end-to-end, pixel-to-pixel semantic segmentation network [33]. Wang et al. propose RDS Net, which introduces three modules, including the target frame auxiliary instance mask relationship module, the mask pruning module and the mask refinement target positioning module, and it designs a dual flow network to overcome the

low resolution of the instance mask to a great extent [34]. Cai et al. propose Cascade R-CNN, which adds a split branch for each cascade stage, extending the cascade architecture to instance segmentation tasks [35]. Jiang et al. introduce two branches on the basis of Mask R-CNN [36] and propose a post-processing refinement module based on the overall nested edge detection best possible recall (BPR) to improve the boundary quality of Mask R-CNN [37]. Instance segmentation tasks are also widely used in remote-sensing images, face detection and other scenes.

2.2. Visual SLAM

The research on visual SLAM includes the positioning of the camera and the construction of an environmental map. Kerl et al. propose DVO-SLAM, a dense visual SLAM method for RGB-D cameras that minimizes both the photometric and the depth error over all pixels [38]. Endres et al. propose RGBD-SLAM v2, which calculates the camera motion between two frames by feature matching and iterative closest point (ICP) methods in the front end and performs pose graph optimization with loop detection in the back end, and it can build 3D maps with high accuracy [39]. McCormac et al. propose Fusion++, which is built from modules for image-based instance segmentation, truncated signed distance function (TSDF) fusion and tracking and pose graph optimization, and create a long-term map, which focuses on the most important object elements of a scene [40]. Mur-Artal et al. propose ORB-SLAM2, which is the excellent successor of parallel tracking and mapping (PTAM) and achieves the first parallelization of three threads for real-time tracking, local bundle adjustment (BA) optimization and global loop detection, and it uses the Oriented FAST and Rotated BRIEF (ORB) features for feature point extraction, matching, loop detection and relocalization [41]. Qin et al. propose the monocular visual-inertial SLAM system called Vins-Mono, which belongs to the optimization-based tight-coupling method [42]. Whelan et al. propose Elasticfusion, which adopts the mapping method of continuously optimizing reconstruction to improve the accuracy of reconstruction, and the map is represented by the surfel model [43]. Dou et al. propose Fusion4d, which uses several depth cameras to reconstruct rapidly changing non-rigid bodies [44]. Muhammet et al. adopt visual-inertial odometry (VIO), a low-cost sensor fusion method that integrates inertial data with images using a deep-learning (DL) framework to predict the position of an unmanned aerial system (UAS) [45]. Cao et al. propose GVINS, which tightly integrate GNSS raw measurements with visual and inertial information to achieve real-time and drift-free state estimation [46]. In general, the research direction of SLAM is more focused on improving the positioning performance of the system.

2.3. Semantic Map

Some scholars proposed the concept of semantic maps at the beginning of the 21st century and elaborated that building environment maps with semantic information could enhance the perception and interaction capabilities of robots. In recent years, more and more researchers have combined deep-learning methods with SLAM technology to extract semantic information of the environment using target detection, semantic segmentation and other algorithms. Then, they are integrated into the environment map to construct a semantic map of the environment. Semantic maps can be broadly divided into scenario-oriented semantic maps and object-oriented semantic maps.

The scenario-oriented semantic maps divide the environment into several semantic parts in a holistic way. Li et al. combine the DeepLab network [47] with LSD-SLAM [48] to build semi-dense semantic maps through pixel-level semantic segmentation without any prior depth information. The semantic annotations are transferred into the 3D map and regularized with a CRF process. However, the initialization of the system still needs to be optimized [49]. McCorma et al. propose a method called SemanticFusion to construct pixel-wise semantic maps in indoor scenes. This method optimizes the semantic segmentation results of a single image based on the matching relationship of image feature points output by the SLAM algorithm, but the improvement is very limited and requires great

computational resources [50]. Brucker et al. propose a method to determine the room type based on information such as the type and distribution of objects within the environment and propose a method to calculate the degree of association between objects and rooms, which helps improve the execution efficiency of robot interaction tasks [51]. Chen et al. propose a probabilistic model to consider the semantic information of all frames to estimate the semantics of each map point. In addition, the authors use temporal motion information to argue whether a map point is dynamic or static [52]. Zhao et al. propose a pixel–voxel network to obtain global context information by using PixelNet to exploit the RGB image and, meanwhile, preserve accurate local shape information by using VoxelNet to exploit the corresponding 3D point cloud. Subsequently, each point on the 3D map is identified and semantically tagged [53]. CNN-SLAM naturally blends the dense depth map predicted by CNN with the depth measurements obtained by direct monocular SLAM to obtain an improved depth map, which is used to build a scene-oriented semantic map [54].

Object-oriented semantic maps are composed of individual instances, which allows the robot to manipulate and maintain each object in the map. Salas-Moreno et al. propose SLAM++ to create a map with semantically defined objects, but it relies on a predefined database and hand-crafted template models [55]. Hoang et al. propose Object-RPE to solve the problem of constructing semantic maps when there are occlusions and environmental clutter in the environment, and it has better robustness [56]. Sunderhauf et al. propose an object-oriented semantic mapping method based on the combination of SSD [57] and ORB-SLAM2, which maintains and updates the point cloud for each class of objects during the runtime. However, it is not possible to maintain and update each object in the map individually, which makes it difficult for the robot to distinguish between objects of the same category in practical applications [58]. Grinvald et al. propose an online volume-instance-aware semantic map building system based on RGB-D data, where the exact segmentation of objects is obtained by joint inference of geometric and semantic cues, and finally, the global object-level semantic map is obtained by fusing local information. However, the pose estimation errors accumulate during the operation of the system, leading to an increased impact on the map [59]. Li et al. directly use the mapping relationship between voxels and pixels to obtain the semantic categories of voxels, thus reducing the computational effort in semantic fusion and improving the real-time performance of the algorithm to a certain extent, but the accuracy of the semantic map is slightly poor [60]. Liu et al. propose to combine YOLO V4 [61] with RGB-DSLAM to perform dense mapping and object location, but the boundaries of the constructed map are not perfect [62]. Runz et al. propose MaskFusion, which is a method of semantic mapping based on instance segmentation in dynamic scenes. This method combines instance segmentation and geometric segmentation to solve the problem of rough edges, but the segmentation method used requires a large amount of computational resources [63]. Dorian et al. propose a monocular SLAM system for perceptible objects, which uses the binary bag of words (a database containing 500 3D object models) to complete target recognition and construct 3D semantic maps, but the system causes obvious errors in the maps when the initialization fails [64].

3. Method

Our method is divided into two main parts: semantic information extraction and semantic mapping. In the section of semantic information extraction, the classical target tracking algorithm Deepsort [65] and the instance segmentation algorithm Mask R-CNN [36] are combined to extract the 2D semantic information of the target object and establish data associations between the images, which provide a priori information for subsequent camera pose optimization and point cloud fusion. In the section of semantic mapping, the RGB-D version of ORB-SLAM2 [41], an open-source SLAM system, is used for tracking and mapping, which uses both RGB and depth information for sparse tracking. Semantic information is added to the newly created semantic mapping thread of ORB-SLAM2 to build object-oriented semantic maps, reconstruct the target objects in the maps and optimize the camera pose, which is critical and applicable in the field of human–computer interaction.

We are cautious about terming our method "semantic SLAM", as its usual task consists of two aspects: SLAM helping semantics and semantics helping SLAM. Our method contains one of the two directions—semantics helping SLAM—including localization and mapping.

3.1. Target Tracking Based on Instance Segmentation

Mask R-CNN [36], an instance segmentation algorithm, is used to obtain 2D semantic information in RGB images in our method, which is used as the observation data for the target tracking algorithm to track the target objects to establish the data association in a video sequence. The extracted 2D semantic information includes semantic label, tracked identification (ID) and 2D pixel coordinates, where the 2D pixel coordinate information of the objects will be recorded in txt files in a hierarchical manner according to the timestamp of the image, the semantic classification and tracking ID of the object. These pixel coordinates are extracted from the object's mask, and the two coordinates of a point occupy one line of the txt file. The frame diagram of Deepsort with Mask R-CNN is shown in Figure 1.

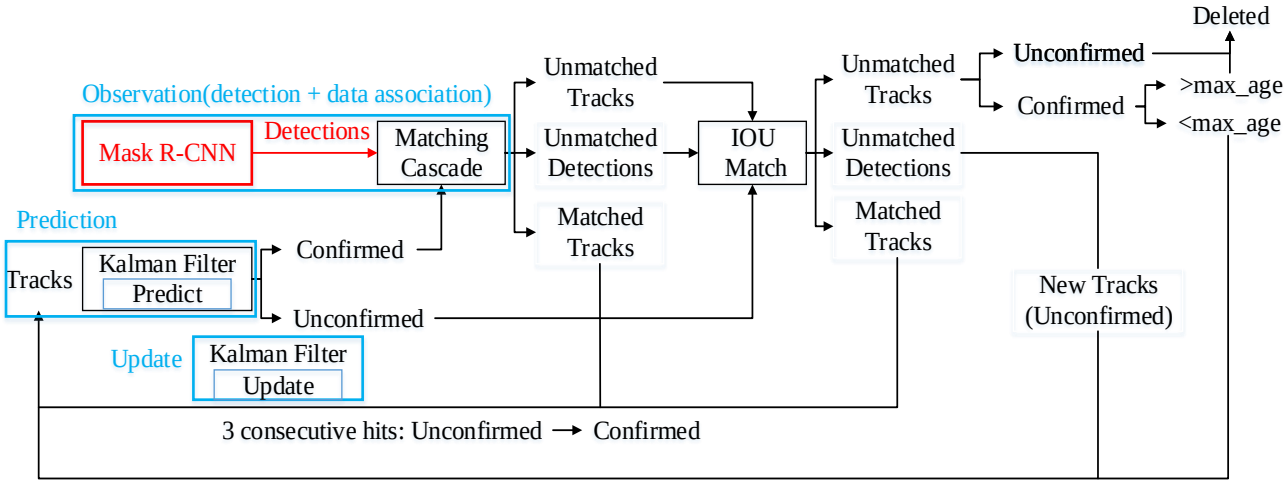


Figure 1. The frame diagram of Deepsort with Mask R-CNN.

Deepsort [65], a classical multi-objective tracking algorithm, is mainly used to track and predict vehicles and pedestrians in combination with target detection. The core processes of Deepsort include prediction, observation and update. The prediction thread performs prediction for the next frame to obtain the prediction box of the target objects. The observation thread performs target detection for the current frame to obtain the detection box of the target objects and establish data association with the objects in the previous frame. There is some error between the results generated by the above two threads and the real results. The update thread combines the results of the prediction thread and the observation thread to update the results, making the final error smaller than that of the single thread. The prediction thread and update thread of Deepsort can predict the position of tracks at the next time and update the predicted location based on detections by using Kalman filtering. Kalman filtering uses a linear homogeneous model, which means the movement of the frame, the change in the size and deformation of the frame are considered to change linearly and homogeneously. In Deepsort, the tasks of the prediction thread can be expressed as the following formula.

$$\overline{\hat{x}}_k = A\hat{x}_{k-1} \tag{1}$$

$$\overline{P}_k = AP_{k-1}A^T + Q \tag{2}$$

where Formula (1) is used to compute the mean vector $\overline{\hat{x}}_k$ of the prediction box at frame k; Formula (2) is used to compute the covariance matrix $\overline{P}_k$ of the prediction box at frame k; A represents the motion matrix and is used to predict the state of the system at the next time. The tasks of the update thread include computing the Kalman gain, updating the

estimate via measurement and updating the error covariance, which can be expressed as the following formula, respectively.

$$y = z_k - H\bar{\hat{x}}_k \tag{3}$$

$$S = H\overline{P}_k H^T + R \tag{4}$$

$$K_k = \overline{P}_k H^T S^{-1} \tag{5}$$

$$\hat{x}'_k = \hat{x}_k + K_k y \tag{6}$$

$$P_k = (I - K_k H)\overline{P}_k \tag{7}$$

where $z_k = [m, n, l, h]$ represents the mean vector of the detection box at frame k; the four parameters represent the horizontal and vertical coordinates of the upper left corner, length and width of the detection box; y represents innovation; H represents the measurement matrix; in Formula (4), R represents the noise matrix of the detector; in Formula (5), K_k represents the Kalman gain at frame k and is used to estimate the importance of the error; Formulas (6) and (7) are used to compute the updated mean vector and the covariance matrix. The observation thread includes the results detected by Mask R-CNN and data association. The results detected by Mask R-CNN, which include the box, mask, class and score of each object in the picture, are regarded as the observed data of Kalman filtering to match one by one with the prediction box in the previous frame. The match algorithm includes the intersection-over-union (IOU) match and the cascade match, and the cost function includes the Mahalanobis distance and cosine distance. If both distances are as small as possible, and the boxes and features are similar, it is considered that the two boxes are the same thing. The successful pairing of the detection box and the prediction box means that the previous frame and the next frame are successfully tracked. Whether the objects can be tracked successfully is greatly influenced by the detection results.

3.2. Improved Dense Mapping Based on ORB-SLAM2

ORB-SLAM2 [41], one of the very classic and practical systems in modern SLAM work, which represents a peak of the mainstream feature point SLAM, is used to construct semantic maps in our method. The whole system is computed around ORB features, with fast computation speed, rotational invariance and scale invariance. We propose to add a new point cloud mapping thread in ORB-SLAM2 to upgrade the original sparse map to a dense semantic map. The prerequisite for mapping is to calculate the camera's pose. Therefore, the camera pose estimation problem needs to be solved first. In SLAM, the pose of the camera can be initially estimated in the visual odometry, which may have some errors. In order to optimize the camera pose x, SLAM is transformed into an optimization problem. The equation of observation and motion in visual SLAM can be expressed as

$$\begin{cases} x_i = f(x_{i-1}, u_i) + \omega_i \\ z_{i,j} = h(y_j, x_i) + v_{i,j} \end{cases} \quad i = 1, 2, \ldots\ldots, N,\ j = 1, 2, \ldots\ldots, M \tag{8}$$

where x_i represents the camera pose at frame i; y_j represents the jth road sign; $z_{i,j}$ represents the data generated by the jth road sign observed at frame i; u_k represents the motion data at frame i; w_i and $v_{i,j}$ represent the motion noise and observation noise generated at frame i. If there are no motion data, the camera pose can also be optimized by observing the data. Considered in terms of least squares, then, the error of the observation can be expressed as

$$e_{ij} = z_{i,j} - h(\xi_i, y_j) \tag{9}$$

After considering the observed data at other moments, the overall cost function of the observed error to be optimized can be expressed as

$$\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}||e_{ij}||^2 = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}||z_{ij} - h(\xi_i, y_j)||^2 \tag{10}$$

where $z_{i,j}$ represents the observed data of the jth road sign y_j at frame i; ξ_i represents the Lie algebra corresponding to the camera pose at frame i. e_{ij} represents the observation error arising from the observation of road sign j at frame i. m denotes the number of images involved in the calculation. n denotes the number of road signs involved in the calculation. Solving this least-squares calculation is equivalent to making adjustments to both the camera pose ξ and the road signs y, which is also known as BA. The point cloud mapping thread converts the coordinates of all pixels with a depth value greater than 0.01 on the key frame to 3D coordinates using the information of the corresponding depth value, camera pose and camera internal parameter matrix. The transformation can be divided into four steps:

(1) World coordinates to camera coordinates

$$\vec{P_C} = \begin{bmatrix} X_C \\ Y_C \\ Z_C \end{bmatrix} = [R \quad t] \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix} \tag{11}$$

(2) Camera coordinates to the normalized coordinates P_C

$$P_C = [X_C/Z_C, Y_C/Z_C, 1]^T \tag{12}$$

(3) The points in the normalized plane are distorted, and the distortion parameter is

$$D = (k_1, k_2, k_3, p_1, p_2) \tag{13}$$

(4) Normalized coordinates to pixel coordinates

$$P_{uv} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} P_C \tag{14}$$

The total conversion formula is

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \frac{1}{Z_C} \begin{bmatrix} f_x & 0 & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & t \\ \vec{0} & 1 \end{bmatrix} \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix} \tag{15}$$

where f_x, f_y, u_0, v_0 represent the internal parameters of the camera; R represents the rotation matrix; t represents the translation matrix; Z_C represents the depth value; (u, v) represent the pixel coordinates; (X_W, Y_W, Z_W) represent the corresponding world coordinates. (X_C, Y_C, Z_C) represent the corresponding camera coordinates. The point cloud mapping thread can generate a dense point cloud of the environment using the key frame according to the transformation relationship between the world coordinate system, the camera coordinate system and the pixel coordinate system. The key frame is representative of a series of local ordinary frames and is responsible for recording local information. The mapping method based on the key frame can retain most of the mapping information, which compensates for the shortage of carrying too little information in the sparse map, and it can also avoid the problems of large amount of calculation and information redundancy caused by calculating each image.

3.3. Semantic Mapping Based on Semantic SLAM

The contents of Sections 3.1 and 3.2 can be combined to build object-oriented semantic maps. The frame diagram of the object-oriented semantic mapping system is shown in Figure 2.

The specific steps of the system can be divided into three steps. The first step is the extraction of semantic information carried out by Mask R-CNN and Deepsort. The input RGB images are detected and segmented by Mask R-CNN, and the detected results are

put into the Deepsort as observation data, including the box, class, mask and score of each object. Deepsort tracks each target and assigns an ID to each tracking target. The 2D pixel coordinates of each object on each image are recorded in txt files. These txt files contain information such as the timestamp, class names, tracked ID and the pixel coordinates corresponding to each ID and are put into ORB-SLAM2.

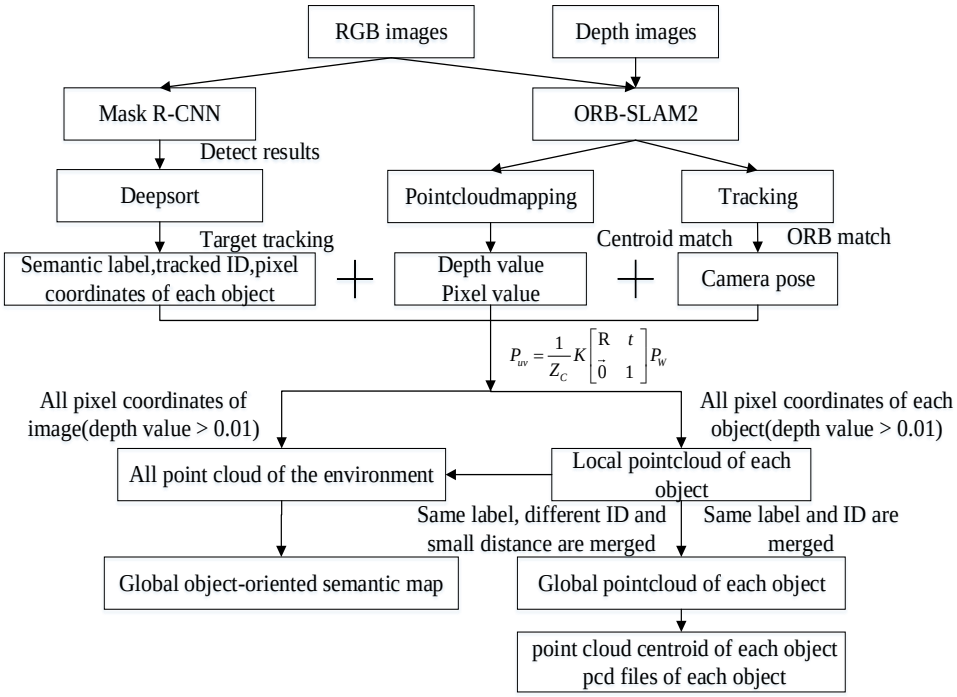


Figure 2. The frame diagram of the object-oriented semantic mapping system.

The second step is the construction of a semantic map using the extracted semantic information and ORB-SLAM2. When the ORB-SLAM2 system receives a key frame, the pixel coordinates in the corresponding txt files are converted by the point cloud mapping thread to 3D coordinates with semantic colors in the camera coordinate system, and the other pixel coordinates in the key frame are transformed into 3D coordinates with the original color. The map obtained in this way is a local semantic map that can be converted to a point cloud map in the world coordinate system based on the current camera pose. In order not to affect the running speed of SLAM, the RGB images, depth maps, local semantic maps and camera poses corresponding to key frames are stored in separate containers, and there is a separate calculation process of traversing information and building maps. After filtering, all local maps are transformed into point cloud maps in the world coordinate system according to the transformation matrix. Then, these maps are fused into a global semantic map.

The third step is the reconstruction of the target object. In order to operate and maintain each object in the map, we need to know the location of each object. A possible method is to convert the coordinates of each object extracted in the first step into a 3D point cloud and subsequently fuse the point clouds with the same ID at different moments to construct a point cloud model of the target object. An obvious problem, however, is that objects with different IDs may belong to the same object because the ID may change during the process of target tracking. To collect point cloud information from multiple angles of an object to reconstruct the object, the point cloud of the same object with different IDs needs to be fused according to the theory of multi-view geometry. In this paper, we calculate the centroids of the point clouds and calculate the Euclidean distance between the centroids of

the point clouds; then, the two point clouds are considered as the same point clouds if the distance of the two centroids is less than 0.1 m. The calculation formulas are as follows.

$$x_c, y_c, z_c = \sum_{i=1}^{n} x_i, \sum_{i=1}^{n} y_i, \sum_{i=1}^{n} z_i \tag{16}$$

$$d_{12} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} \tag{17}$$

where (x_c, y_c, z_c) is the centroid of the point cloud; (x_i, y_i, z_i) is the 3D coordinate of the ith point. n represents the number of points in the point cloud.d_{12} represents the Euclidean distance between two centroids.x_1, y_1, z_1 and x_2, y_2, z_2 represent the 3D coordinates of the two centroids. There will be outlier points in the actual calculation process. For example, the boundary information extracted by semantic segmentation is not perfect and contains some points that do not belong to the object, which are converted into 3D points with a large distance from the 3D points of the object and should be eliminated; otherwise, the calculated centroid is inaccurate. In this paper, the point clouds are filtered by using the filtering method of rejecting outlier points. The general idea is as follows:

(1) For each point in the point cloud, we calculate the average distance d from it to all the points in its K-neighborhood.
(2) The distances calculated above are assumed to obey a Gaussian distribution. The corresponding points with distance values outside the standard range (defined by the mean and variance of the sample) are defined as outliers and should be removed from the datasets.

The threshold for the outlier judgment is calculated as

$$t = m + k * s \tag{18}$$

where t is the judgement threshold, m represents the mean value of the distance sample, s represents the standard deviation of the distance sample. k represents the parameter used to adjust the size of the threshold. The smaller the k, the smaller the threshold and the more obvious the filtering effect. However, this does not mean that the smaller the k, the better the filtering effect. When k decreases, the boundary points of the objects will be filtered out, which is not the desired result. As the local maps of the objects are fused, a large number of duplicate or very close points will be added to the point cloud, which will eventually result in a very dense point cloud and take up a lot of memory space. Too dense point clouds are not meaningful for object reconstruction and need to be filtered. In this paper, the point clouds are filtered by using the voxel filtering method. The core idea is to divide the point cloud into multiple voxels (three-dimensional cubes) and use the points nearest the center of the voxel to represent all the point clouds within the voxel. This filtering method allows for a significant reduction in the number of 3D points within the point cloud while preserving the shape characteristics of the point cloud. In addition, the instance segmentation algorithm has a misdetection in the process of detection because the features of the same object from different angles are not the same. For example, in the detection process, teddy bears are sometimes detected as people, and mouses are sometimes detected as cups. To solve this problem, we consider that the object can be correctly detected when its corresponding ID appears at least 3 times, and we consider this object to be an object that needs to be reconstructed. The point clouds of the correctly detected objects are saved, and whenever the objects carrying the same ID or with a distance of less than 0.1 m from the centroids of the existing point clouds reappear, the newly transformed point clouds are added to the corresponding existing point clouds. Finally, we can obtain the point cloud models of all objects. The position of an object in a three-dimensional space is represented by the centroid of the corresponding point cloud.

3.4. Camera Pose Optimization Based on Semantic SLAM

In addition to using geometric features, our approach uses detected semantic objects as landmarks and tracks and matches objects in different images, which elevates the data association problem in SLAM from the traditional pixel level to the object level and provides object constraints for camera pose estimation, thus optimizing the camera pose. After obtaining the pixel coordinates of the object, the pixel coordinates are transformed into 3D coordinates under the camera coordinate system by the camera internal reference, and thus, the 3D point cloud of the object is obtained. The centroid of the object's point cloud is used to represent the object's position in the 3D space. In Section 3.1, we use the target tracking algorithm to establish data association between different images and assign tracking IDs to the objects, so that objects carrying the same ID in different images are considered as the same object and are used for object matching. Our approach uses the point cloud centroid of an object to represent its position in a three-dimensional space and is used for object matching. Compared with feature matching, object matching is less influenced by the environment and has better robustness. The 3D to 3D method of calculating the poses is the ICP algorithm. The goal remains to calculate the camera pose x. The camera pose can also be expressed using the Lie algebra ξ. Suppose there are a set of well-matched 3D point cloud centroids, which are as follows:

$$P = \{p_1, p_2 \cdots, p_n\}, \quad P' = \{{p'}_1, {p'}_2 \cdots, {p'}_n\} \tag{19}$$

When expressing the camera pose in terms of the Lie algebra ξ, the objective function can be written as follows:

$$f(\xi) = \frac{1}{2}\sum_{i=1}^{n} ||(p_i - \exp(\xi^{\wedge}){p_i}')||_2^2 \tag{20}$$

where $p_i, {p_i}'$ represent a pair of centroids matching the 3D coordinates; $\exp(\xi^{\wedge})$ represents the left multiplication perturbation model of the Lie algebra. n signifies that there are n pairs of matching points. The goal is to find the ξ that minimizes $f(\xi)$. Object matching can provide more stable observation data. Correct centroid matching can add constraints to the pose estimation and optimize the camera pose, while incorrect centroid matching can do the opposite, so more correct centroid matching is needed for pose optimization. This paper proposes a method to calculate the probability that two point clouds are point clouds of the same object, which is used to select the correct centroid matching.

Suppose there are two sets of point clouds A, B; first, transform the two point clouds into the same coordinate system by the estimated transformation matrix of the two images and build A as an octree; then, iterate through all points in B and query whether this point exists in the corresponding voxel of A (i.e., whether the point clouds of A and B exist in the same voxel); if it does exist, the point is the overlapping point cloud of B. The above method can obtain the overlapping point clouds in A and B. The degree of overlap of the two point clouds can be expressed as the following equation.

$$\mathrm{M} = \frac{\mathrm{AB}}{\mathrm{A} + \mathrm{B} - \mathrm{AB}} \tag{21}$$

where AB represents the overlap of point cloud A and point cloud B; M represents the degree of overlap. There are several ways to evaluate whether A and B are point clouds of the same object.

(1) Both have the same ID.
(2) The distance between the two centroids in the same coordinate system is less than 0.1 m.
(3) The degree of overlap between the two, M, is greater than 0.8.

In practice, although the target tracking algorithm provides a priori information for object matching, it still has limitations. Since the IDs will switch during target tracking,

objects with different IDs may belong to the same object. Likewise, objects with the same ID may not be the same object due to the matching error problem. Moreover, two point clouds with small centroid distance or high degree of overlap do not necessarily belong to the same object. Therefore, it is necessary to consider the above three conditions together. The formula to determine whether two point clouds belong to the same object is as follows.

$$P = w(P_{ID}, D, M)^T \tag{22}$$

where w represents the weight vector. P_{ID} represents whether two objects have the same ID, which equals 1 if they are the same and 0 if they are different. D represents the distance between the two centroids. M represents the degree of overlap of two point clouds. P represents the probability that two point clouds belong to the same object. The larger the P, the higher the probability that two point clouds belong to the same object. We choose a probability threshold K. When $P > K$, this indicates that the two point clouds belong to the same object. The centroids of the two matching point clouds are added to the calculation of pose estimation to optimize the camera pose.

4. Experiments and Analysis

All the experiments in this paper are completed under the Ubuntu20.04 system. The semantic information extraction part adopts Mask R-CNN and Deepsort based on the deep-learning framework of keras-2.1.5 and tensorflow_gpu-1.13.2, completed by python. The semantic mapping part adopts the ORB-SLAM2 system based on opencv-3.4.16, pcl-1.9.0, pangolin-0.6, eigen-3.3.8, vtk-7.1.1 and other dependency libraries, completed by C++. The computing power of the computer is 5.2. The experiments use cuda-10.0 and cudnn-7.4.1.5 for gpu acceleration, opencv for image data feature extraction and matching, pcl for mapping, point cloud segmentation, point cloud fusion, point cloud storage, pangolin for visual interface. Eigen provides a large number of linear algebras of matrices, matrices, vector operation functions, etc. G2o is a graph optimization solver, and the DBoW3 is used for camera relocalization and loop detection in the visual SLAM. The computer parameters of the experiments are shown in Table 1.

Table 1. The computer parameters of the experiments.

Name	Model	Remarks
Operating System	Ubuntu 20.04	/
Graphic Processing Unit (GPU)	NVIDIA Quadro M2000	4G
Central Processing Unit (CPU)	Intel® Xeon(R) CPU E5-1620 v4	3.5 GHz * 8
Random Access Memory (RAM)	DDR4	32G
Hard Disk	SSD256G + HDD 1000G	Samsung

The semantic mapping algorithm in this paper is applicable to indoor office scenes. In all SLAM datasets, the TUM dataset is a classic indoor SLAM dataset, including the static scene, dynamic scene, object reconstruction scene and other scenes [66]. This paper conducts experiments and tests on six datasets in TUM using the semantic mapping algorithm, where, some images in the freiburg2_dishes datasets are used for labeling and training to generate the self-training weight, which is used to predict and generate the mask and label of the objects. Meanwhile, the other five datasets use the coco weight trained in advance to predict on the images. The task of our method is to optimize the camera pose, construct object-oriented semantic maps and reconstruct the multi-objective objects detected by the instance segmentation algorithm in the indoor scene.

4.1. Experiments on Self-Training Weight

The RGBD datasets of freiburg2_dishes in TUM are selected as the datasets of the experiments. The datasets contain 3007 color images and 3005 depth images. The main detection objects are three types of tableware in the image, including two bowls, one cup

and one plate. The detection task is to complete the instance segmentation of four objects in the images, generate the detection box, semantic label and mask of each object, and obtain the pixel coordinates of the four objects according to the corresponding mask. In total, 750 images are selected from freiburg2_dishes for annotation to generate json files, which will be converted to voc format and put into Mask R-CNN for training. The specific parameters for training in Mask R-CNN are shown in Table 2.

Table 2. The specific parameters for training in Mask R-CNN.

Parameters	Quantity
backbone	Resnet101
Epoch	50
Number of pictures	750
Training ratio	0.8
Validation ratio	0.1
Test ratio	0.1
Bach size	1
Training time(s/epoch)	2917
class_names	bowl, plate, cup

In total, 750 images are selected for annotation and training in a dataset containing 3000 images. It is now common practice to divide the dataset into a training set, a validation set and a test set. Empirically, the training ratio is 0.8, the validation ratio is 0.1, and the test ratio is 0.1. In addition, the model terminates the experiment early if there is no decrease in validation loss for five consecutive epochs during training. The training set is used to train the parameters of the model, and the test set is used to evaluate the goodness of the model. The validation set is used to validate the model training effect in the training phase and is used to select the hyperparameters. Resnet101 is selected as the backbone of the model, and this network is used as the backbone in many instance segmentation networks due to its performance advantages, such as Mask R-CNN [36], Cascade Mask R-CNN [35], etc. The epoch and Bach size are set according to the performance of the computer. The training time for each epoch in the training is 2917 s. Overall, the parameters of our model are chosen empirically or with reference to the work of others and are not necessarily the best options. There is not a lot of difference in the overall performance and no major impact on subsequent tasks.

To ensure the accuracy of the pixel coordinates of the extracted objects, the detection performance of the Mask R-CNN needs to be verified. The mean average precision (mAP) is selected as the evaluation indicator to evaluate the detection performance of Mask R-CNN. The two important parameters of the instance segmentation network are precision and recall, and the calculation formula of the two parameters is

$$precision = \frac{TP}{TP + FP} \qquad recall = \frac{TP}{TP + FN} \tag{23}$$

where TP (true positives) signifies that the object is divided into positive samples and is correct. TN (true negatives) signifies that the object is divided into negative samples and is correct. FP (false positives) signifies that the object is divided into positive samples and is incorrect. FN (false negatives) signifies that the object is divided into negative sample and is incorrect. The average precision (AP) in fact refers to the area under the precision and recall curve. The mAP refers to the mean value of the AP for all classes. The higher the mAP, the better the detection performance of the network. The training results of Mask R-CNN on freiburg2_dishes are shown in Figure 3.

It can be seen that the value loss basically converges when the epoch approaches 50. The detection results of freiburg2_dishes using self-training weight are shown in Figure 4.

As shown in Figure 4, each object on the image has its own ID after the target tracking algorithm is added to Mask R-CNN. After instance segmentation and target tracking, the RGB images, depth images and the txt files with the pixel coordinates of each object on

each key frame are input to the ORB-SLAM2 system to obtain the semantic map of the environment and the dense point cloud model of each object. The results of dense mapping and dense semantic mapping on freiburg2_dishes using self-training weight are shown in Figure 5.

As shown in Figure 5, the algorithm works well for objects with obvious features, such as a bowl and a plate, but not for objects with no obvious features, such as a cup. The reason is that the measured depth of the edge of the object is usually inaccurate, and the positioning of orbslam2 also has errors, and the mask obtained by Mask R-CNN is not perfect, which causes the point cloud of each object to be not fine enough.

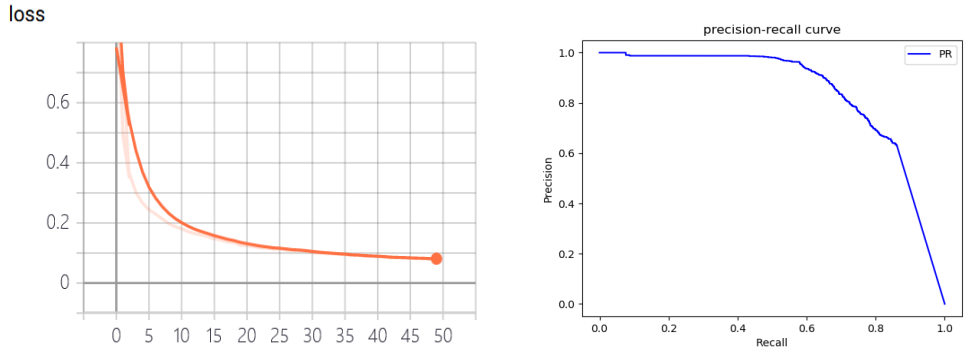


Figure 3. The training results of Mask R-CNN on freiburg2_dishes. The left image is the value loss diagram, and the right is the precision–recall diagram.

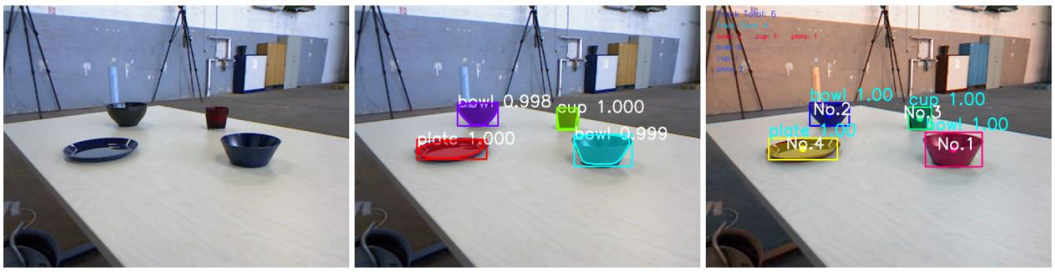
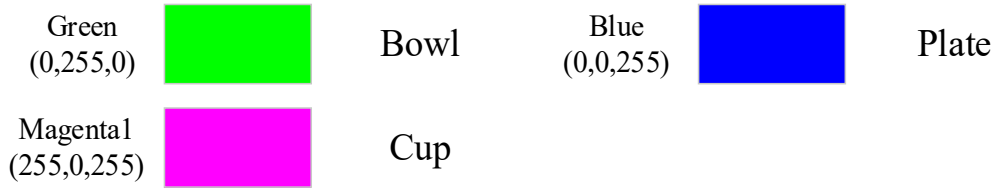
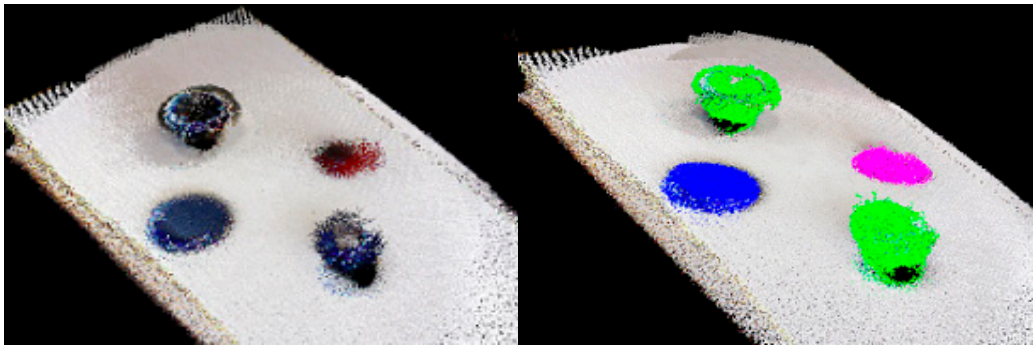


Figure 4. The detection results of freiburg2_dishes using self-training weight. From left to right, the first image is the original RGB image. The second is the detection result of Mask R-CNN. The third is the detection result of Mask R-CNN with Deepsort.

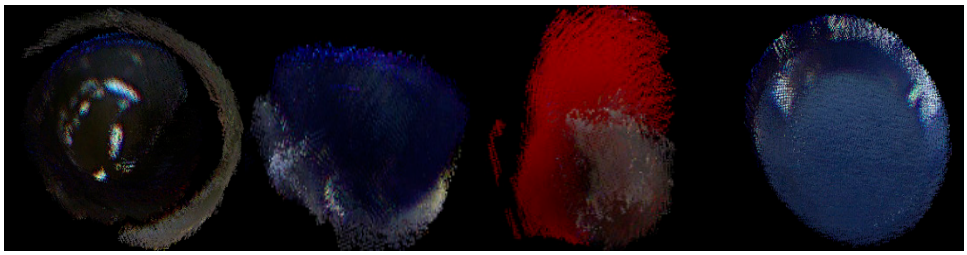


(**a**) The color information corresponding to three categories of objects in semantic map

Figure 5. *Cont.*



(**b**) The dense point cloud map (**left**) and the dense semantic point cloud map (**right**)



(**c**) The dense point cloud model of each object of freiburg2_dishes (bowl, bowl, cup, plate)

Figure 5. The results of semantic mapping of freiburg2_dishes using self-training weight. Figure (**a**) is the color information corresponding to three categories of objects in semantic map. Figure (**b**) is the dense point cloud map (**left**) and the dense semantic point cloud map (**right**). Figure (**c**) is the dense point cloud model of each object of freiburg2_dishes (bowl, bowl, cup, plate).

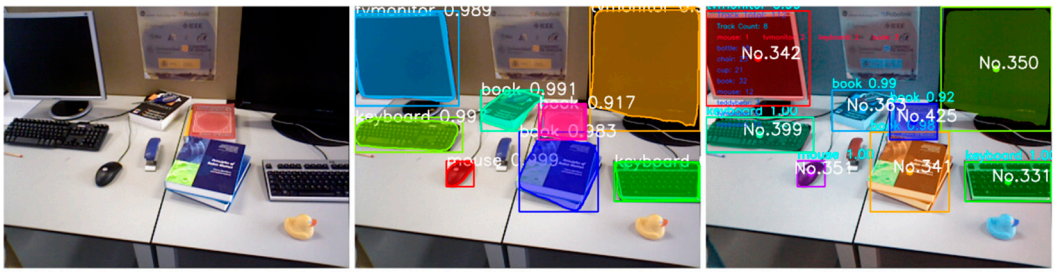
4.2. Experiments on Coco Weight

The experiments are conducted on five RGBD datasets in TUM using the coco weight, including freiburg1_desk, freiburg1_room, freiburg2_desk, freiburg3_office, freiburg3_teddy. The mask_rcnn_coco.h5, which was already trained on the coco datasets, is selected as the weight used for detection. Eight categories of objects in the coco datasets are selected for detection in our method, including chair, cup, TV monitor, teddy bear, bottle, book, keyboard and mouse. The detection task is to complete the instance segmentation in each image and target tracking of the objects in an image sequence and record the pixel coordinates of each object in the corresponding txt files. The results of target tracking and instance segmentation on TUM datasets using the coco weight are shown in Figure 6.

As shown in Figure 6, the detection result of Mask R-CNN includes the semantic label, score, detection box and mask. The detection result of Mask R-CNN with Deepsort includes the semantic label, score, detection box, mask and tracked ID. Multiple objects in the image are tracked to establish the data association between different images. It can be seen from (a) and (b) that only using Mask R-CNN for detection may result in partial missed detection, such as the book with ID 13 in (a) and the book with ID 49, the cup with ID 59 in (d). Mask R-CNN with Deepsort can predict the state of the system in the next frame according to the detection results of Mask R-CNN and the motion model of the system. Therefore, there will be less missed detection in the converged system, which will make the point cloud model of the object more complete in the subsequent point cloud mapping. Moreover, instance segmentation will produce some wrong results. For example, windows are detected as TV monitors, cups are detected as mice, which eventually leads to some errors in semantic mapping. The target tracking algorithm can reduce wrong results through data associations.

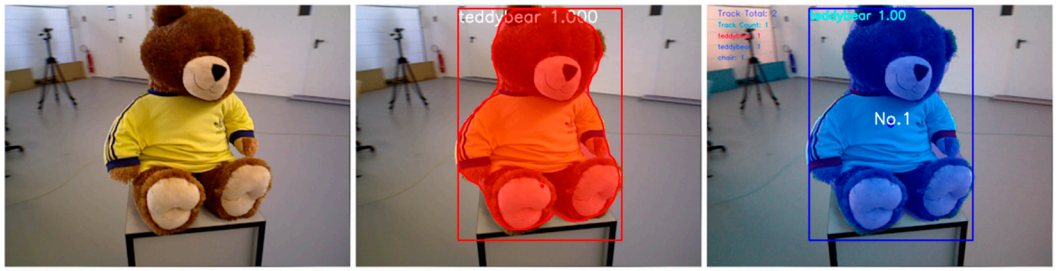
The above semantic information extracted by instance segmentation and target tracking are input to the ORB-SLAM2 system along with the RGB images and depth images to obtain the dense point cloud map, the dense semantic point cloud map of the environment and the dense point cloud model of each object. The results of dense point cloud mapping

and dense semantic point cloud mapping of the five datasets in TUM using the coco weight are shown in Figure 7.



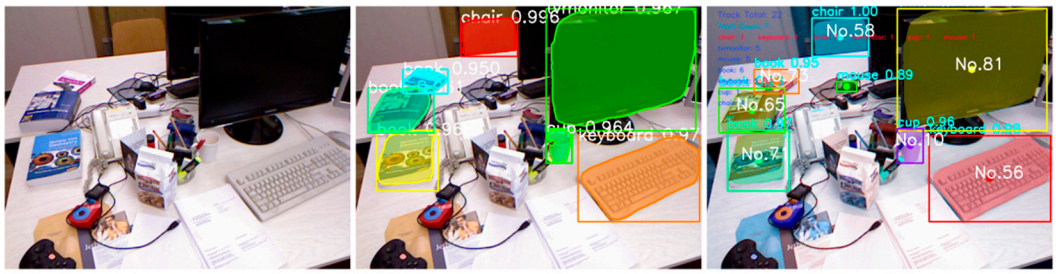
(**a**) The detection results of the back of freiburg3_office

(**b**) The detection results of the front of freiburg3_office



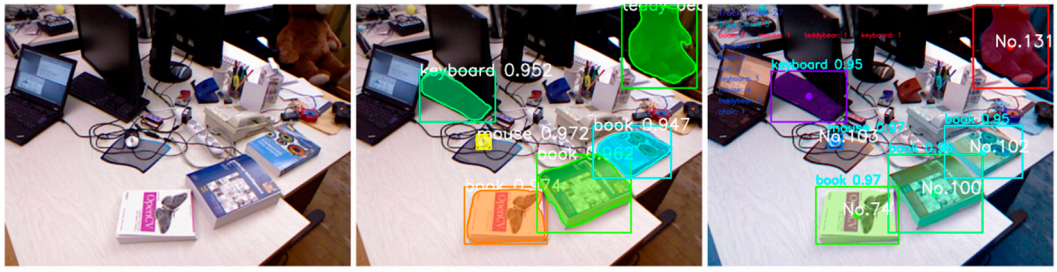
(**c**) The detection results of freiburg3_teddy

(**d**) The detection results of freiburg2_desk



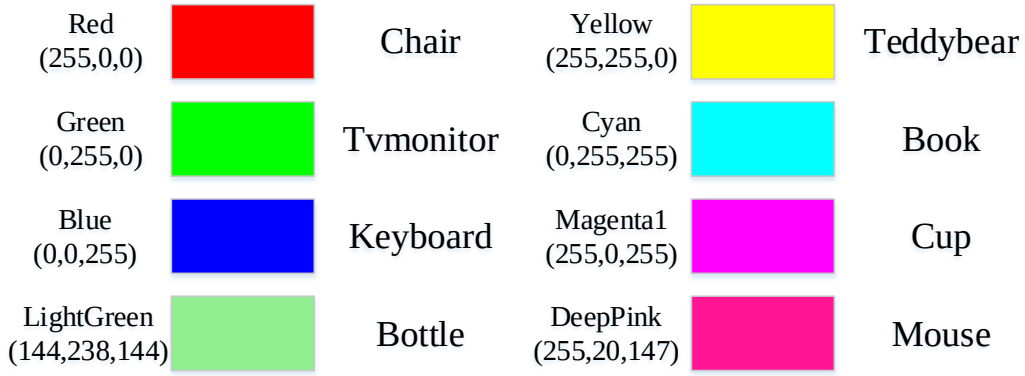
(**e**) The detection results of freiburg1_desk

Figure 6. *Cont.*

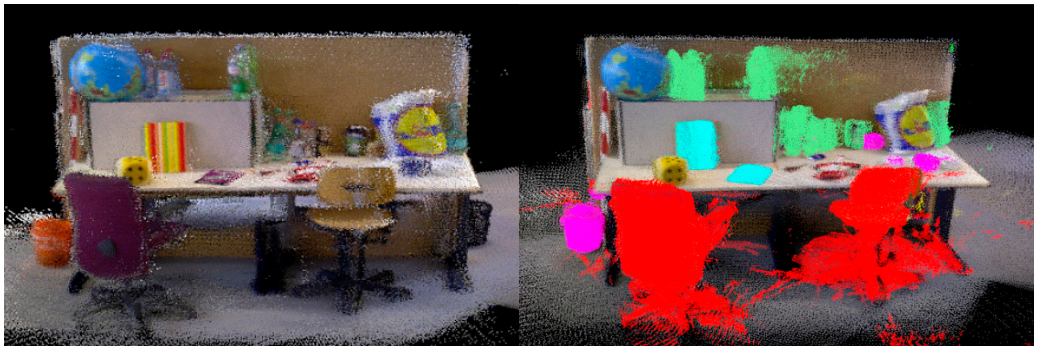


(**f**) The detection results of freiburg1_room

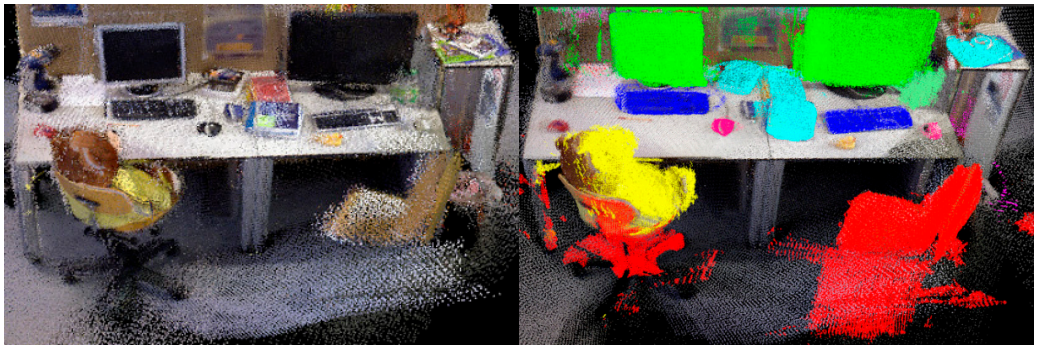
Figure 6. The results of target tracking and instance segmentation on TUM datasets using coco weight. From left to right, the first figure is the original RGB image. The second is the detection result of Mask R-CNN. The third is the detection result of Mask R-CNN with Deepsort. Figure (**a**–**f**) are the detection results of the back of freiburg3_office, the front of freiburg3_office, freiburg3_teddy, freiburg2_desk, freiburg1_desk and freiburg1_room respectively.



(**a**) The color information corresponding to eight categories of objects in semantic map

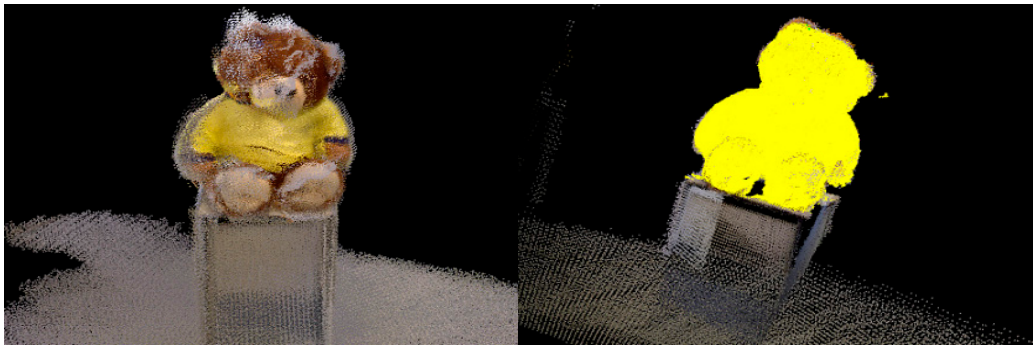


(**b**) The front of the dense semantic point cloud map of freiburg3_office

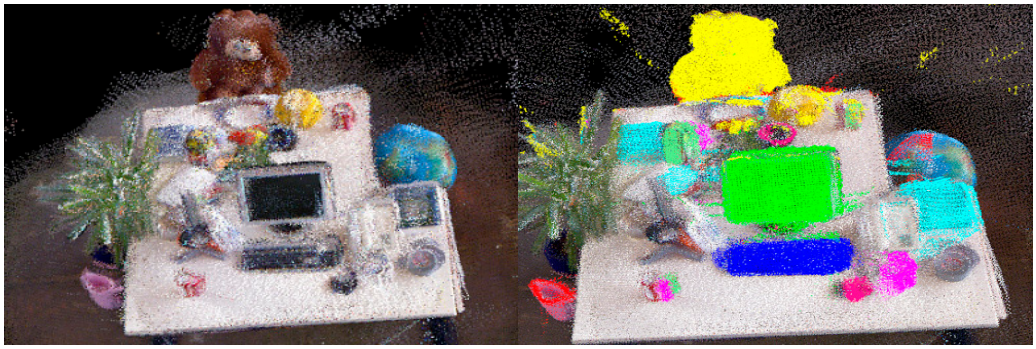


(**c**) The back of the dense semantic point cloud map of freiburg3_office

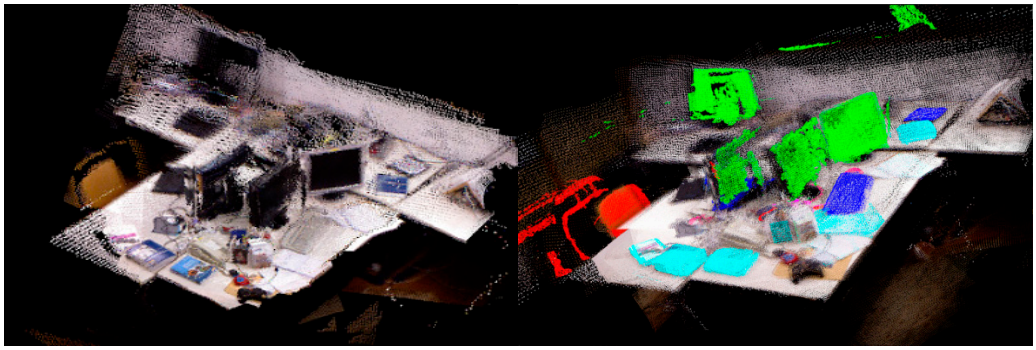
Figure 7. *Cont.*



(**d**) The dense semantic point cloud map of freiburg3_teddy



(**e**) The dense semantic point cloud map of freiburg2_desk



(**f**) The dense semantic point cloud map of freiburg1_desk

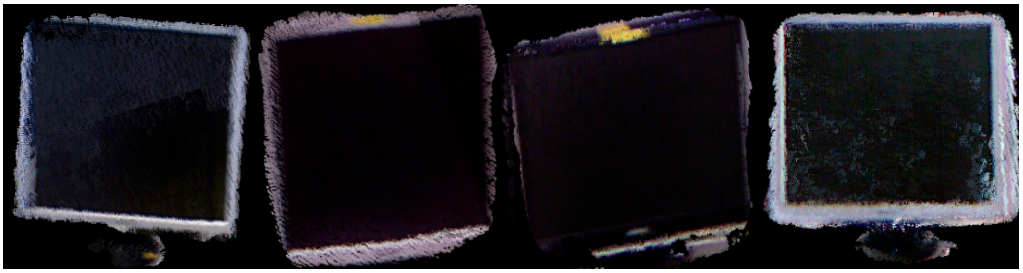
(**g**) The dense semantic point cloud map of freiburg1_room

Figure 7. The results of semantic mapping of five datasets in TUM using coco weight. Figure (**a**) shows the color corresponding to the objects. Figures (**b**–**g**) show the dense point cloud map (left) and the dense semantic point cloud map (right) of the 5 datasets.

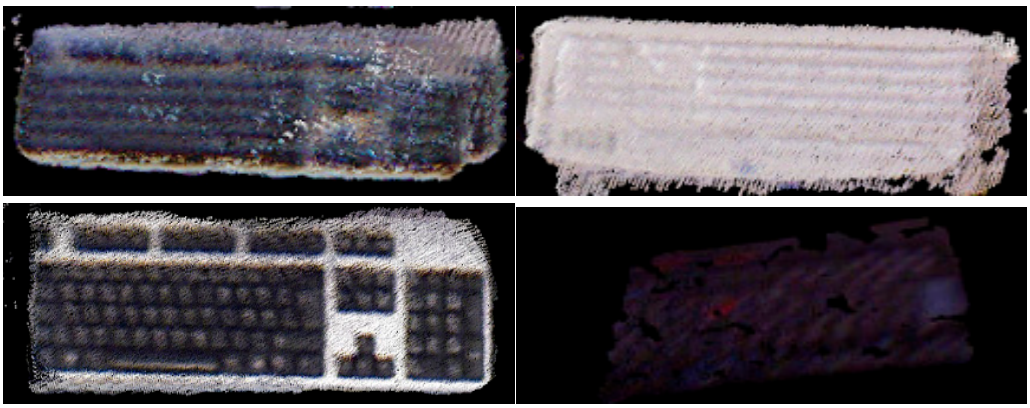
As shown in Figure 7, the dense point cloud map is added as a comparison to make the results of semantic mapping look clearer. There is a lot of noise in the boundary of the object when it is converted to 3D point cloud due to the imperfect boundary segmentation

of the object in semantic segmentation, which can be mitigated by adjusting the parameters of the outpoint filtering algorithm. It can be seen that the semantic maps do not change in the overall scene but add colors to each target object to distinguish and facilitate scene understanding. The object to be reconstructed is the object marked with a color. The dense point cloud model of the eight categories of objects obtained by matching and fusion between point clouds is shown in Figure 8.

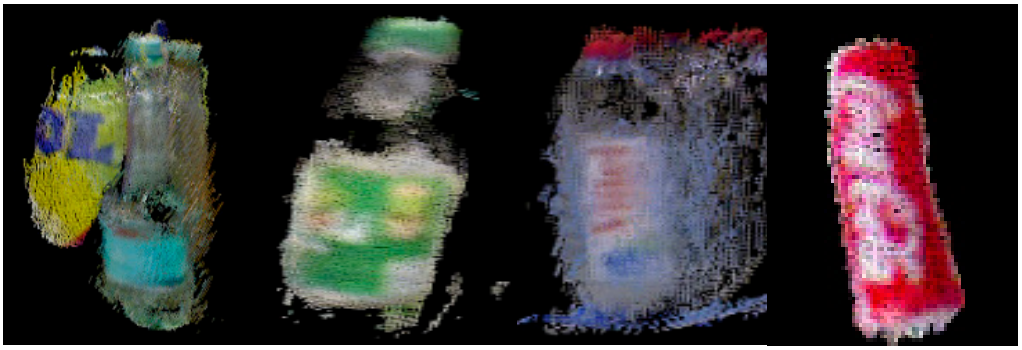
(**a**)



(**b**) TV monitor



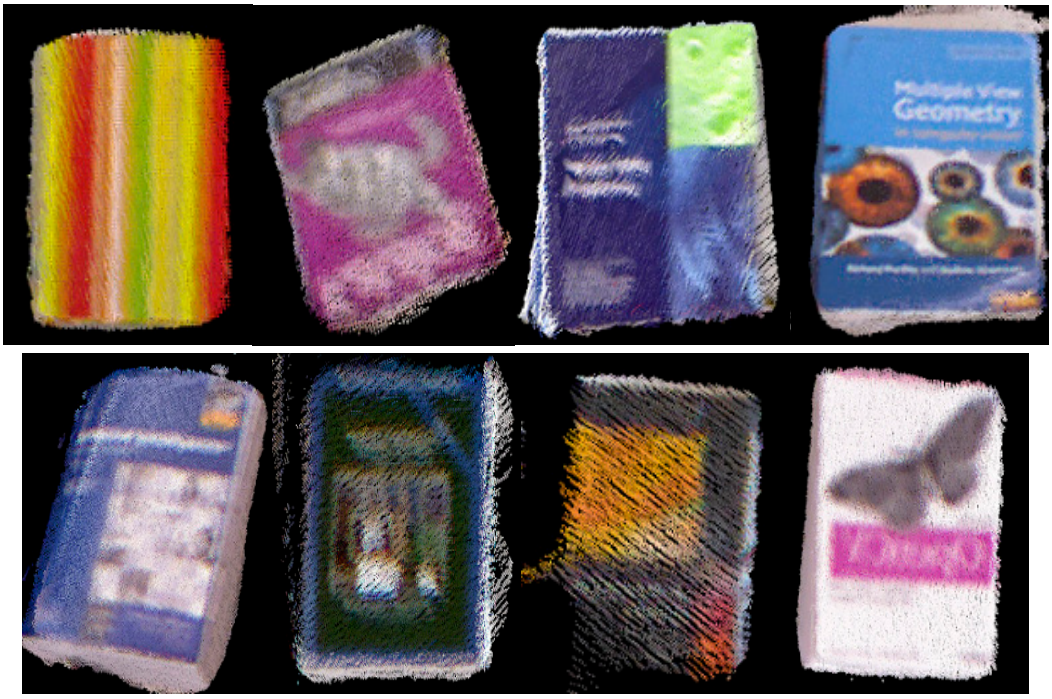
(**c**) keyboard



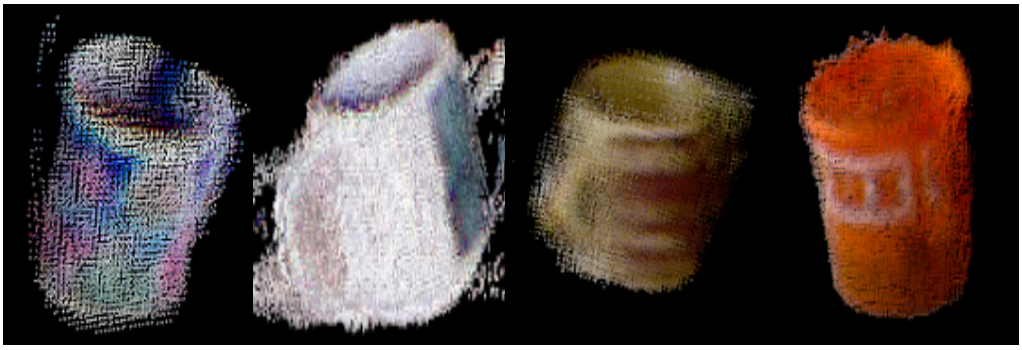
(**d**) bottle

Figure 8. *Cont.*

(**e**) teddy bear



(**f**) book



(**g**) cup

Figure 8. *Cont*.

(**h**) mouse

Figure 8. The dense point cloud model of the eight categories of objects. Figure (**a**–**h**) are the point clouds of the chair, TV monitor, keyboard, bottle, teddy bear, book, cup, mouse respectively.

As shown in Figures 7 and 8, the algorithm works well for objects with obvious features, such as a TV monitor, book, teddy bear and keyboard, but not for objects with no obvious features, such as a cup, bottle, mouse and chair. This is partly because the depth data of the object edge in the depth map collected by the RGB-D camera are inaccurate. Another reason is that there are errors in the camera pose estimated by ORB-SLAM2, which leads to ghosting of the map after fusion, such as the bottle, keyboard and cup shown in Figure 8c,d,g. In addition, the mask obtained by Mask R-CNN is not perfect. In the process of mapping, only when the camera rotates 360 degrees around the object can we obtain a complete point cloud model of the object. Otherwise, the point cloud model will be missing somewhere. For example, when we view the point cloud model from another angle, it may be missing a piece. Adding a target tracking algorithm to the process of extracting semantic information can provide a priori information for matching and fusing objects in a 3D space, thus optimizing the camera poses and constructing globally consistent point clouds of objects.

The centroid of the dense point cloud can be calculated after obtaining the point cloud model of each object. There are two ways to evaluate the accuracy and effectiveness of the algorithm. The first method is to compare the number of detected objects (the number of final point cloud models, expressed by D) with the number of actual objects (the ground truth value, expressed by GT) in the five datasets, and the results are shown in Table 3 (only the above eight categories of objects are calculated).

Table 3. The number of detected objects and actual objects.

Datasets	The Number of Detected Objects (D)	The Number of Actual Objects (GT)	Error
fr3_office	36	30	20.0%
fr3_teddy	1	1	0.0%
fr2_desk	18	16	12.5%
fr1_desk	16	15	6.7%
fr1_room	21	20	5.0%

The difference between D and GT can be used to express the accuracy of the algorithm, and the formula of the difference can be expressed as

$$\text{error} = \frac{\text{abs}(D - GT)}{GT} \times 100\% \tag{24}$$

The smaller the error, the more accurate the algorithm. It is worth mentioning that D is always greater than GT because the IDs of objects always switch in target tracking, despite the point cloud models with the same label and close distance being merged. As shown in Table 3, the number of detected objects (D) is greater than or equal to the number of actual objects in all datasets. This is because an object may correspond to multiple tracking

IDs, or some detection areas that are not target objects are detected as objects. The more objects in the environment, the greater the error of the algorithm. It can be seen from the results of the error that our algorithm can segment the point cloud model of objects in the environment with a high accuracy.

The second method is to compare the objects' centroids computed by the mapping algorithm with the ground truth of the objects' centroids. The TUM datasets do not provide the true position of each object but the ground truth pose, which is measured by an external (very advanced) motion capture device. The ground truth pose can be regarded as a standard answer to calculate the ground truth of the objects' centroids in a 3D space. The estimated camera pose takes the camera pose of the first frame as the origin of the world coordinate system, and the ground truth camera pose takes a fixed point in space as the origin of the world coordinate system. Therefore, the ground truth camera pose needs to be converted to the estimated world coordinate system. The objects' centroids calculated by the estimated pose and the ground truth pose are shown in Table 4 (take fr3_office, for example).

Table 4. The objects' centroids calculated by the estimated pose and the ground truth.

Objects	Tracked ID	Method	Centroids	Error (m)
chair	6	estimated ground truth	[0.585, 0.343, 2.32] [0.591, 0.352, 2.34]	0.0237
TV monitor	389	estimated ground truth	[0.592, −0.713, 3] [0.593, −0.705, 3]	0.00823
keyboard	513	estimated ground truth	[0.565, −0.614, 3.24] [0.576, −0.612, 3.24]	0.0116
bottle	3	estimated ground truth	[−0.513, −0.78, 2.4] [−0.511, −0.773, 2.4]	0.00828
teddy bear	188	estimated ground truth	[0.778, −0.966, 3.81] [0.776, −0.953, 3.81]	0.0131
book	11	estimated ground truth	[−0.203, −0.0962, 2.28] [−0.198, −0.0899, 2.29]	0.00997
cup	41	estimated ground truth	[−1.37, 0.143, 2.92] [−1.37, 0.147, 2.93]	0.0098
mouse	402	estimated ground truth	[0.252, −0.701, 3.41] [0.254, −0.7, 3.4]	0.00312

The error represents the distance of the estimated centroid and the ground truth centroid. The smaller the error, the higher the accuracy of the algorithm. It can be seen from Table 4 that the average error of the position of the object calculated for the eight categories of objects in the fr3_office dataset is 0.0109 m, which means that the positioning algorithm has a high accuracy. It can be seen from the table that the larger the volume of the object, the worse the positioning accuracy, which is consistent with our common sense. Moreover, the higher the positioning accuracy of the algorithm, the more accurate the positioning of the object. Fortunately, the positioning accuracy of orbslam2 is very high, and the absolute pose error is 0.0094 m in the fr3_office dataset. In order to further verify the effectiveness of the algorithm, the average positioning error of each dataset is calculated and is shown in Table 5.

The average positioning error of objects refers to the distance of the objects' three-dimensional centroids calculated by using the estimated and true values of the camera pose. The maximal average positioning error (m) of objects is 0.347 m and the minimal is 0.00375 m. Why can some objects have such large errors? The reason is that the observation angles of each object in each picture are different. Similarly, the pose errors of each different key frame are different, and each object with different angles will generate a point cloud. Finally, when the point cloud is fused, some redundancy may occur, resulting in a certain degree of offset of the object's centroid. However, it can be seen in general that, from the

above data, the algorithm has a high accuracy for small objects and a poor accuracy for large objects. Overall, the system has a high positioning accuracy.

Table 5. The average positioning error (m) of objects in five datasets.

Objects	fr3_Office	fr3_Teddy	fr2_Desk	fr1_Room	fr1_Desk
chair	0.0235	/	0.146	0.266	0.232
TV monitor	0.00927	/	0.266	**0.347**	0.0839
keyboard	0.0118	/	0.0908	0.347	0.123
bottle	0.0154	/	0.0557	0.0347	0.00387
teddy bear	0.0177	0.00186	0.133	0.189	/
book	0.00972	/	0.0238	0.161	0.00946
cup	0.0167	/	0.0199	0.0539	0.0366
mouse	0.00656	/	**0.00375**	0.0979	0.0846

If collision and path planning are not considered, other map information in the environment is redundant, except for the map information of objects. The number of point clouds of each dataset is shown in Table 6. The number of point clouds of each dataset is reduced by 45.3%, 59.8%, 57.6%, 61.5% and 54.3%, respectively, after segmenting the objects from the environment, which effectively reduces the system memory usage and removes the redundant information from the environment.

Table 6. The number of point clouds of five datasets.

Datasets	Number of Original Point Clouds	Number of Point Clouds with Objects Only	Reduction
fr3_office	4,814,372	2,635,597	45.3%
fr3_teddy	8,451,663	3,395,009	59.8%
fr2_desk	3,090,597	1,195,625	57.6%
fr1_room	2,581,204	993,299	61.5%
fr1_desk	364,226	166,546	54.3%

4.3. Comparison with Existing Methods

Quantitative comparison. In the study of SLAM systems, the absolute trajectory error (ATE) or relative pose error (RPE) are generally used to evaluate the merits of a SLAM system, and both evaluation criteria revolve around the pose estimation results of SLAM systems. In this paper, the absolute positional error is used to evaluate the performance of SLAM systems. The following Table 7 shows the comparison of our method with the existing methods.

Table 7. Comparison of the localization performance of our method with existing SLAM systems. RMSE (m).

Sequence	ORB-SLAM2 [41]	Elastic-Fusion [43]	DVO-SLAM [38]	Fusion++ [40]	Ours
fr3_office	0.0121	0.017	0.035	0.1082	**0.0115**
fr3_teddy	**0.0225**	0.048	0.046	0.1535	0.0259
fr2_desk	0.0095	0.071	0.017	0.1144	**0.0086**
fr1_room	0.0474	0.068	**0.043**	0.2356	0.0452
fr1_desk	0.0163	0.020	0.021	0.0499	**0.0153**

As can be seen from the above table, our method shows better performance in most of the datasets. Compared with the ORB-SLAM2 system, our approach has an overall performance improvement with the addition of semantic constraints. It is worth noting that our system is not yet up to the real-time requirement, which requires further tight integration of semantics and SLAM to further improve the performance of the system.

Functional comparison. There are no recognized reliable evaluation criteria for map construction effects. Some classical semantic slam systems are selected for comparison

with the proposed approach in this paper, mainly around the implemented functions. The comparison results are shown in Table 8.

Table 8. Comparison of the function of our method with existing semantic SLAM systems.

Semantic SLAM System	Semantics	Scenario-Oriented Semantic Maps	Object-Oriented Semantic Maps	Semantic Help SLAM Positioning	Objects Location	Objects Reconstruction
SLAM++ [55]	√		√			
Meaningful maps [58]	√		√			
CNN-SLAM [54]	√	√				
SemanticFusion [50]	√	√		√		
MaskFusion [63]	√		√	√		√
Ours	√		√	√	√	√

Most of the current semantic SLAM systems focus on semantic mapping and semantic-assisted localization while ignoring the individual entities in the scene. The method proposed in this paper constructs an object-oriented semantic map of the scene while completing the reconstruction and localization of multi-target objects and optimizing the camera pose with semantic constraints, which has certain implications for the subsequent development of semantic SLAM.

Specifically, a beautiful object-based semantic map is constructed by using prior information in SLAM++, but the 3D model of the object needs to be known in advance, and the real scene cannot be restored [55]. CNN-SLAM combines the depth map predicted by CNN with the depth map estimated by monocular SLAM to obtain a more accurate depth map and uses this to build a scenario-oriented semantic map, but the system is more concerned with depth estimation than with semantic maps [54]. Sunderhauf combines SSD with ORB-SLAM2 to construct object-based semantic maps, but the experiment only includes four objects, and the mapping results perform well only for square objects, such as monitors, books and keyboards [58]. SemanticFusion can construct a dense surfel reconstruction from a video sequence but not to the level of a single object [50]. Mask-Fusion combines Mask R-CNN with Elastic-fusion to construct a real-time, object-aware, semantic and dynamic RGB-D SLAM system, which can reconstruct each object in the environment [63]. However, two GPUs with very good performance are required to accelerate in the actual running process in Mask-Fusion, and even so, the running speed is still very slow, which is difficult to meet the real-time performance requirement, and the map effect is poor. In our system, the process of extracting semantic information is the process of pre-processing, and the main running tasks are optimizing the camera pose, constructing semantic maps and reconstructing multi-target objects. In fact, eight categories of objects are used for detection and semantic mapping using coco weights in the experiments, and most objects in the actual scene can be reconstructed. In addition, our system can construct the dense semantic point cloud map and output point cloud model of each object to reconstruct each object with a high accuracy and small positioning error.

Overall, the method proposed in the paper has clear advantages. First, with the introduction of semantic information, the camera poses of the traditional SLAM system are optimized, which indicates that the robot is positioned more accurately in the scene. Second, the introduced semantic information can convert the traditional point cloud map into an object-oriented point cloud map, which allows the robot to interact with individuals in the environment. For example, tasks such as asking the robot to pick up a book on a table can be achieved because the map clearly shows the location of each object, and the system has sufficient localization accuracy, except for individual errors. Again, our system is able to output point cloud maps of each object in the environment, and these point cloud maps can be used for subsequent 3D reconstruction tasks, such as using TSDF-based surface reconstruction, which can result in beautiful object models. Of course, our proposed

method also has some disadvantages. First, the system is currently not real time because it contains a pre-processing step for instance segmentation, and the mask R-CNN is slow in detection, although it has a high accuracy. In order to make the system run in real time, both detection accuracy and detection speed must be considered. Second, when optimizing the camera pose, we only track and match the center of the mass of the target point cloud without using the full semantic information. If the full semantic information could be used, the camera pose could be further optimized.

5. Conclusions

Inspired by image processing methods, such as deep-learning, target detection and instance segmentation algorithms, we propose a method for constructing object-oriented semantic maps that maintains individual objects as the key entities in the map, which builds on SLAM, instance segmentation and target tracking. Experiments show that the 3D semantic point cloud map, which only contains objects, reduces the number of points by about 50% compared with the dense point cloud map of the complete environment. The average positioning error of the eight categories of objects in TUM datasets is very small, which means that the positioning algorithm has a high accuracy. In the tests on the five TUM datasets, the absolute positional error of the camera is also reduced with the introduction of semantic constraints, and the positioning performance of the system is improved. At the same time, our algorithm can segment the point cloud model of objects in the environment with a high accuracy, which is of great practical importance in practical engineering applications.

Unfortunately, there are some shortcomings in the methodology of this paper, which need to be improved. Our method uses semantic information to improve the accuracy of SLAM localization and complete semantic map building but does not currently meet the real-time requirements, nor does it use data association to improve the accuracy of semantic information extraction. Our future work will focus on the above two issues.

Author Contributions: Conceptualization, Y.S. and J.Y.; methodology, Y.S.; software, Y.L.; validation, G.J., J.K. and B.C.; formal analysis, D.B.; investigation, J.Y.; resources, X.L.; data curation, J.H.; writing—original draft preparation, J.H.; writing—review and editing, J.H.; visualization, Jun Hu; supervision, J.Y.; project administration, Y.S.; funding acquisition, G.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data that support the findings of this study are available from the corresponding author upon reasonable request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Huang, L.; Chen, C.; Yun, J.; Sun, Y.; Tian, J.; Hao, Z.; Yu, H.; Ma, H. Multi-scale Feature Fusion Convolutional Neural Network for Indoor Small Target Detection. *Front. Neurorob.* **2022**, *16*, 881021. [CrossRef]
2. Huang, L.; Fu, Q.; He, M.; Jiang, D.; Hao, Z. Detection Algorithm of Safety Helmet Wearing Based on Deep Learning. *Concurr. Comput. Pract. Exp.* **2021**, *33*, e6234. [CrossRef]
3. Jiang, D.; Zheng, Z.; Li, G.; Sun, Y.; Kong, J.; Jiang, G.; Xiong, H.; Tao, B.; Xu, S.; Yu, H.; et al. Gesture Recognition Based on Binocular Vision. *Clust. Comput.* **2019**, *22* (Suppl. 6), 13261–13271. [CrossRef]
4. Jiang, D.; Li, G.; Sun, Y.; Hu, J.; Yun, J.; Liu, Y. Manipulator Grabbing Position Detection with Information Fusion of Color Image and Depth Image Using Deep Learning. *J. Ambient. Intell. Humaniz. Comput.* **2021**, *12*, 10809–10822. [CrossRef]
5. Li, G.; Xiao, F.; Zhang, X.; Tao, B.; Jiang, G. An Inverse Kinematics Method for Robots after Geometric Parameters Compensation. *Mech. Mach. Theory* **2022**, *174*, 104903. [CrossRef]
6. Sun, Y.; Zhao, Z.; Jiang, D.; Tong, X.; Tao, B.; Jiang, G.; Kong, J.; Yun, J.; Liu, Y.; Liu, X.; et al. Low-illumination Image Enhancement Algorithm Based on Improved Multi-scale Retinex and ABC Algorithm Optimization. *Front. Bioeng. Biotechnol.* **2022**, *10*, 865820. [CrossRef] [PubMed]

7. Bai, D.; Sun, Y.; Tao, B.; Tong, X.; Xu, M.; Jiang, G.; Chen, B.; Cao, Y.; Sun, N.; Li, Z. Improved Single Shot Multibox Detector Target Detection Method Based on Deep Feature Fusion. *Concurr. Comput. Pract. Exp.* **2022**, *34*, e6614. [CrossRef]
8. Liu, X.; Jiang, D.; Tao, B.; Jiang, G.; Sun, Y.; Kong, J.; Tong, X.; Zhao, G.; Chen, B. Genetic Algorithm-based Trajectory Optimization for Digital Twin robots. *Front. Bioeng. Biotechnol.* **2022**, *9*, 793782. [CrossRef] [PubMed]
9. Liu, Y.; Jiang, D.; Yun, J.; Sun, Y.; Li, C.; Jiang, G.; Kong, J.; Tao, B.; Fang, Z. Self-tuning Control of Manipulator Positioning Based on Fuzzy PID and PSO Algorithm. *Front. Bioeng. Biotechnol.* **2022**, *9*, 817723. [CrossRef]
10. Liu, Y.; Xiao, F.; Tong, X.; Tao, B.; Xu, M.; Jiang, G.; Chen, B.; Cao, Y.; Sun, N. Manipulator Trajectory Planning Based on Work Subspace Division. *Concurr. Comput. Pract. Exp.* **2022**, *34*, e6710. [CrossRef]
11. Liu, Y.; Li, C.; Jiang, D.; Chen, B.; Sun, N.; Cao, Y.; Tao, B.; Li, G. Wrist Angle Prediction Under Different Loads Based on GAELM Neural Network and sEMG. *Concurr. Comput. Pract. Exp.* **2022**, *34*, e6574. [CrossRef]
12. Liu, Y.; Jiang, D.; Tao, B.; Qi, J.; Jiang, G.; Yun, J.; Huang, L.; Tong, X.; Chen, B.; Li, G. Grasping Posture of Humanoid Manipulator Based on Target Shape Analysis and Force Closure. *Alex. Eng. J.* **2022**, *61*, 3959–3969. [CrossRef]
13. Liu, Y.; Jiang, D.; Xu, C.; Sun, Y.; Jiang, G.; Tao, B.; Tong, X.; Xu, M.; Li, G.; Yun, J. Deep Learning Based 3D Target Detection for Indoor Scenes. *Appl. Intell.* **2022**, 1–14. [CrossRef]
14. Wu, X.; Jiang, D.; Yun, J.; Liu, X.; Sun, Y.; Tao, B.; Tong, X.; Xu, M.; Kong, J.; Liu, Y.; et al. Attitude Stabilization Control of Autonomous Underwater Vehicle Based on Decoupling Algorithm and PSO-ADRC. *Front. Bioeng. Biotechnol.* **2022**, *10*, 843020. [CrossRef] [PubMed]
15. Zhao, G.; Jiang, D.; Liu, X.; Tong, X.; Sun, Y.; Tao, B.; Kong, J.; Yun, J.; Liu, Y.; Fang, Z. A Tandem Robotic Arm Inverse Kinematic Solution Based on an Improved Particle Swarm Algorithm. *Front. Bioeng. Biotechnol.* **2022**, *10*, 832829. [CrossRef] [PubMed]
16. Zhang, X.; Xiao, F.; Tong, X.; Yun, J.; Liu, Y.; Sun, Y.; Tao, B.; Kong, J.; Xu, M.; Chen, B. Time Optimal Trajectory Planing Based on Improved Sparrow Search Algorithm. *Front. Bioeng. Biotechnol.* **2022**, *10*, 852408. [CrossRef] [PubMed]
17. Yun, J.; Jiang, D.; Liu, Y.; Sun, Y.; Tao, B.; Kong, J.; Tian, J.; Tong, X.; Xu, M.; Fang, Z. Real-time Target Detection Method Based on Lightweight Convolutional Neural Network. *Front. Bioeng. Biotechnol.* **2022**, *10*, 861286. [CrossRef]
18. Yun, J.; Jiang, D.; Sun, Y.; Huang, L.; Tao, B.; Jiang, G.; Kong, J.; Weng, Y.; Li, G.; Fang, Z. Grasping Pose Detection for Loose Stacked Object Based on Convolutional Neural Network with Multiple Self-Powered Sensors Information. *IEEE Sens. J.* **2022**. [CrossRef]
19. Feng, Q.; Huang, L.; Sun, Y.; Tong, X.; Liu, X.; Xie, Y.; Li, J.; Fan, H.; Chen, B. Substation Instrumentation Target Detection Based on Multi-scale Feature Fusion. *Concurr. Comput. Pract. Experience* **2022**, e7177. [CrossRef]
20. Wang, S.; Huang, L.; Jiang, D.; Sun, Y.; Jiang, G.; Li, J.; Zou, C.; Fan, H.; Xie, Y.; Xiong, H.; et al. Improved Multi-Stream Convolutional Block Attention Module for sEMG-Based Gesture Recognition. *Front. Bioeng. Biotechnol.* **2022**, *10*, 909023. [CrossRef] [PubMed]
21. Shi, K.; Huang, L.; Jiang, D.; Sun, Y.; Tong, X.; Xie, Y.; Fang, Z. Path Planning Optimization of Intelligent Vehicle Based on Improved Genetic and Ant Colony Hybrid Algorithm. *Front. Bioeng. Biotechnol.* **2022**, *10*, 905983. [CrossRef] [PubMed]
22. Tao, B.; Wang, Y.; Qian, X.; Tong, X.; He, F.; Yao, W.; Chen, B.; Chen, B. Photoelastic Stress Field Recovery using Deep Convolutional Neural Network. *Front. Bioeng. Biotechnol.* **2022**, *34*, e7177. [CrossRef] [PubMed]
23. Tao, B.; Liu, Y.; Huang, L.; Chen, G.; Chen, B. 3D Reconstruction Based on Photo Elastic Fringes. *Concurr. Comput. Pract. Exp.* **2022**, *34*, e6481. [CrossRef]
24. Tao, B.; Huang, L.; Zhao, H.; Li, G.; Tong, X. A Time Sequence Images Matching Method Based on the Siamese Network. *Sensors* **2021**, *21*, 5900. [CrossRef] [PubMed]
25. Han, J.; Zhang, D.; Cheng, G.; Liu, N.; Xu, D. Advanced Deep-learning Techniques for Salient and Category-specific Object Detection: A Survey. *IEEE Signal Process. Mag.* **2018**, *35*, 84–100. [CrossRef]
26. Vasudevan, S.; Gächter, S.; Nguyen, V.; Siegwart, R. Cognitive Maps for Mobile Robots-an Object Based Approach. *Robot. Auton. Syst.* **2007**, *55*, 359. [CrossRef]
27. Li, T.; Wang, F.; Ru, C.; Jiang, Y.; Li, J. Keypoint-Based Robotic Grasp Detection Scheme in Multi-Object Scenes. *Sensors* **2021**, *21*, 2132. [CrossRef] [PubMed]
28. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
29. Andreas, P.; Marcus, G.; Kate, S.; Robert, P. Grasp Pose Detection in Point Clouds. *Int. J. Robot. Res.* **2017**, *36*, 1455–1473.
30. Asif, U.; Bennamoun, M.; Sohel, F. RGB-D Object Recognition and Grasp Detection using Hierarchical Cascaded Forests. *IEEE Trans. Robot.* **2017**, *33*, 547–564. [CrossRef]
31. Hao, Z.; Wang, Z.; Bai, D.; Tong, X. Surface Defect Segmentation Algorithm of Steel Plate Based on Geometric Median Filter Pruning. *Front. Bioeng. Biotechnol.* **2022**, *10*, 945248. [CrossRef]
32. Rogowski, A.; Bieliszczuk, K.; Rapcewicz, J. Integration of Industrially-oriented Human-robot Speech Communication and Vision-based Object Recognition. *Sensors* **2020**, *20*, 7287. [CrossRef] [PubMed]
33. Long, J.; Shelhamer, E.; Darrell, T. Fully Convolutional Net-works for Semantic Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *39*, 640–651.
34. Wang, S.; Gong, Y.; Xing, J.; Huang, L.; Huang, C.; Hu, W. RDSNet: A New Deep Architecture for Reciprocal Object Detection and Instance Segmentation. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 12208–12215.

35. Cai, Z.; Vasconcelos, N. Cascade R-CNN: High Quality Object Detection and Instance Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *43*, 1483–1498. [CrossRef] [PubMed]
36. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2961–2969.
37. Jiang, S.; Qi, S.; Wang, L.; Jia, H. Instance Segmentation Modal Based on Mask R-CNN and Multi-feature Fusion. *Comput. Technol. Dev.* **2020**, *30*, 65–70.
38. Kerl, C.; Sturm, J.; Cremers, D. Dense Visual SLAM for RGB-D Cameras. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; pp. 2100–2106.
39. Endres, F.; Hess, J.; Sturm, J.; Cremers, D.; Burgard, W. 3D Mapping with an RGB-D Camera. *IEEE Trans. Robot.* **2014**, *30*, 177–187. [CrossRef]
40. McCormac, J.; Clark, R.; Bloesch, M.; Davison, A.; Leutenegger, S. Fusion++: Volumetric Object-level Slam. In Proceedings of the 2018 International Conference on 3D Vision (3DV), Verona, Italy, 5–8 September 2018; pp. 32–41.
41. Mur-Artal, R.; Tardos, J.D. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Trans. Robot.* **2017**, *33*, 1255–1262. [CrossRef]
42. Qin, T.; Li, P.; Shen, S. VINS-Mono: A Robust and Versatile Monocular Visual- inertial State Estimator. *IEEE Trans. Robot.* **2018**, *34*, 1004–1020. [CrossRef]
43. Whelan, T.; Salas-Moreno, R.F.; Glocker, B.; Davison, A.J.; Leutenegger, S. ElasticFusion: Real-time Dense SLAM and Light Source Estimation. *Int. J. Robot. Res.* **2016**, *35*, 1697–1716. [CrossRef]
44. Dou, M.; Khamis, S.; Degtyarev, Y.; Davidson, P.; Fanello, S.R.; Kowdle, A.; Escolano, S.O.; Rhemann, C.; Kim, D.; Taylor, J.; et al. Fusion4d: Real-time Performance Capture of Challenging Scenes. *ACM Trans. Graph. (TOG)* **2016**, *35*, 114. [CrossRef]
45. Muhammet, F.A.; Akif, D.; Abdullah, Y.; Alper, Y. HVIOnet: A Deep Learning based Hybrid Visual-Inertial Odometry Approach for Unmanned Aerial System Position Estimation. *Neural Netw.* **2022**, *155*, 461–474.
46. Cao, S.; Lu, X.; Shen, S. GVINS: Tightly Coupled GNSS-Visual-Inertial Fusion for Smooth and Consistent State Estimation. In Proceedings of the IEEE Transactions on Robotics: A publication of the IEEE Robotics and Automation Society, Hongkong, China, 5 August 2022; pp. 38–40.
47. Chen, L.C.; Zhu, Y.; Papandreou, G.; Schroff, F.; Adam, H. Encoder-decoder with Atrous Separable Convolution for Semantic Image Segmentation. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 801–818.
48. Engel, J.; Schöps, T.; Cremers, D. LSD-SLAM: Large-scale Direct Monocular SLAM. In Proceedings of the European Conference on Computer Vision, Zurich, Switzerland, 6–12 September 2014; Springer: Cham, Switzerland, 2014; pp. 834–849.
49. Li, X.; Belaroussi, R. Semi-Dense 3D Semantic Mapping from Monocular SLAM. *arXiv* **2016**, arXiv:1611.04144.
50. McCormac, J.; Handa, A.; Davision, A.; Leutenegger, S. SemanticFusion: Dense 3D Semantic Mapping with Convolutional Neural Networks. In Proceedings of the IEEE International Conference on Robotics and Automation, Singapore, 29 May–3 June 2017; pp. 4628–4635.
51. Brucker, M.; Durner, M.; Ambrus, R.; Márton, Z.C.; Wendt, A.; Jensfelt, P.; Arras, K.O.; Triebel, R. Semantic Labeling of Indoor Environments from 3D RGB Maps. In Proceedings of the 2018 EEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May 2018; pp. 1871–1878.
52. Chen, W.; Fang, M.; Liu, Y.H.; Li, L. Monocular Semantic SLAM in Dynamic Street Scene based on Multiple Object Tracking. In Proceedings of the 2017 IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM), Ningbo, China, 19–21 November 2017; pp. 599–604.
53. Zhao, C.; Sun, L.; Purkait, P.; Duckett, T.; Stolkin, R. Dense RGB-D Semantic Mapping with Pixel-voxel Neural Network. *Sensors* **2018**, *18*, 3099. [CrossRef]
54. Tateno, K.; Tombari, F.; Laina, I.; Navab, N. Cnn-slam: Real-time Dense Monocular Slam with Learned Depth Prediction. In Proceedings of the IEEE conference on computer vision and pattern recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 6243–6252.
55. Salas-Moreno, R.; Newcombe, R.; Strasdat, H.; Kelly, P. SLAM++: Simultaneous Localization and Mapping at the Level of Objects. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, 23–28 June 2013; pp. 1352–1359.
56. Hoang, D.C.; Stoyanov, T.; Lilienthal, A.J. Object-rpe: Dense 3d Reconstruction and Pose Estimation with Convolutional Neural Networks for Warehouse Robots. In Proceedings of the 2019 European Conference on Mobile Robots (ECMR), Prague, Czech Republic, 4–6 September 2019; pp. 1–6.
57. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. Ssd: Single Shot Multibox Detector. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; Springer: Cham, Switzerland, 2016; pp. 21–37.
58. Sunderhauf, N.; Pham, T.; Latif, Y.; Milford, M.; Reid, I. Meaningful Maps with Object-oriented Semantic Mapping. In Proceedings of the 2017 IEEE/RSJ International Conference on Intellient Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 5079–5085.
59. Grinvald, M.; Furrer, F.; Novkovic, T.; Chung, J.J.; Cadena, C.; Siegwart, R.; Nieto, J. Volumetric Instance-aware Semantic Mapping and 3D object discovery. *IEEE Robot. Autom. Lett.* **2019**, *4*, 3037–3044. [CrossRef]

60. Li, W.; Gu, J.; Chen, B.; Han, J. Incremental Instance-oriented 3D Semantic Mapping via RGB-D cameras for unknown indoor scene. *Discret. Dyn. Nat. Soc.* **2020**, *2020*, 2528954. [CrossRef]
61. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. Yolov4: Optimal Speed and Accuracy of Object Detection. *arXiv* **2020**, arXiv:2004.10934.
62. Liu, Y.; Xu, M.; Jiang, G.; Tong, X.; Yun, J.; Liu, Y.; Chen, B.; Cao, Y.; Sun, N.; Liu, Z. Target Localization in Local Dense Mapping Using RGBD SLAM and Object Detection. *Concurr. Comput. Pract. Exp.* **2022**, *34*, e6655. [CrossRef]
63. Runz, M.; Buffier, M.; Agapito, L. MaskFusion: Real-time Recognition, Tracking and Reconstruction of Multiple Moving Objects. In Proceedings of the 2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), Munich, Germany, 16–20 October 2018; pp. 10–20.
64. Gálvez-López, D.; Salas, M.; Tardós, J.D.; Montiel, J.M.M. Real-time Monocular Object SLAM. *Robot. Auton. Syst.* **2015**, *75*, 435–449. [CrossRef]
65. Wojke, N.; Bewley, A.; Paulus, D. Simple Online and Realtime Tracking with a Deep Association Metric. In Proceedings of the 2017 IEEE International Conference on Image Processing (ICIP), Beijing, China, 17–20 September 2017; pp. 3645–3649.
66. Sturm, J.; Engelhard, N.; Endres, F.; Burgard, W.; Cremers, D. A Benchmark for the Evaluation of RGB-D SLAM systems. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Algarve, Portugal, 7–12 October 2012; pp. 573–580.