3D Semantic Mapping: A Registration Based Approach

Ryan Amaudruz, Dorian Bekaert, Milena Kapralova, Bogdan Palfi, Darie Petcu, Alexandru Turcu

Project Report - Master Artificial Intelligence

Informatics Institute
University of Amsterdam

Abstract

Vision-based scene understanding is a common task in the fields of robotics and computer vision in which visual systems attempt to understand their environment, both from a semantic and geometric perspective. This task was the base of the Robotic Vision Scene Understanding Challenge [8], which assesses the ability of a robotic vision system to comprehend semantic and geometric aspects of its surroundings. As such, this report proposes a 3D semantic mapping pipeline to solve the Object-based Semantic SLAM (Simultaneous Localization and Mapping) task presented in the challenge. The core of our method is the creation of a 3D point cloud map of a simulated environment with the use of fast global registration. The semantic map is then created using a pre-trained 3D object detection model that assigns bounding boxes around objects in the original 3D map. The results indicate that, while very accurate in terms of predicting the label of an object, our method is not robust regarding other aspects important to the resulting map, such as the number of objects detected and their spatial quality. Our pipeline is impaired by components related to the estimation of coordinate transformations and limitations of the fast global registration method.

1. Introduction

In the fields of computer vision and robotics, vision-based scene understanding involves extracting geometric and semantic information from the environment in order to solve various localization and reasoning tasks. Therefore, scene understanding goes beyond low-level image-processing tasks and focuses on a deeper understanding of the scene by detecting and identifying objects and relationships in the environment. A more specific case of scene understanding comes in the form of Semantic SLAM (*Simultaneous Localization and Mapping*), which is a common task in the fields of robotics and autonomous vehicles that involves mapping an agent's environment while also keeping track of the agent's location. The semantic component refers to the agent's ability to understand its environment beyond geometric features by detecting and identifying objects. The main advantage of this semantic approach is that it allows the agent to understand and reason about its environment, which not only leads to a higher quality SLAM but is also crucial when it comes to human-robot interactions [4].

The report focuses on the 3D semantic mapping aspect of Semantic SLAM in the context of the Robotic Vision Scene Understanding Challenge [8]. This challenge analyzes the performance of a robotic vision system in terms of mapping and understanding its environment. Specifically, the challenge involves the usage of a simulated environment, called Benchbot [20], which facilitates research in the field of semantic scene understanding by providing a realistic simulation of a robotic agent that can explore various indoor environments. Benchbot makes use of the NVIDIA Omniverse and Isaac Sim to provide physically accurate and photorealistic virtual environments, which act as synthetic data.

Out of the raw data returned by the robot, we make use of the RGB-D images and the intrinsic camera matrices. This information is made available on an incremental basis, following each movement of the robot. Therefore, we have designed an incremental pipeline for building a 3D representation of the target environment using the fast global registration algorithm [27]. The created 3D global map is then passed to the pre-trained 3D object detection algorithm, Votenet [15], to extract the information regarding the bounding boxes and labels of the objects. The results are slightly underwhelming, with the pipeline only achieving 2% of what is theoretically considered possible in the simulation.

2. Related Work

2.1. Traditional SLAM

Traditional SLAM methods employ two separate modules, one for estimating the pose in real-time and the other for updating the map [2], often referred to as the "front end" and "back end", respectively. Within this field, two SLAM approaches have been observed: direct SLAM, which does not pre-process collected images but instead relies only on pixel intensity; and indirect SLAM, which processes the available data to identify certain key points, then detects and matches them using the camera pose [1]. The trade-off between these two methods is that the direct method is faster, due to using less information, but less precise.

RGB-D SLAM and Point Clouds

In recent years, RGB-D cameras have been adopted in the SLAM domain. These sensors use cameras and depth sensors, the end result being RGB images with annotated depths. This approach has led to Visual SLAM (VSLAM), which only employs visual information to achieve the SLAM task. A common strategy in VSLAM is to convert the 2D visual information into 3D point clouds, which can then be used to create 3D maps of the environment. The pioneering work of [14] saw the first 3D reconstruction system using the data acquired from the RGB-D camera on a Kinect sensor.

Semantic SLAM

The previously discussed methods are concerned with learning the geometric features of the environment without incorporating semantic information. However, incorporating this semantic information allows the agent to not only understand *where* objects are placed around the environment, but also *what* the objects are, thus allowing for potential reasoning tasks. [10] discusses how acquiring semantic information about surrounding objects can provide useful information for optimizing the VSLAM pipeline: a higher-level understanding of the map can be inferred. A better map understanding can, in turn, lead to a more accurate SLAM application [3]. Semantic VSLAM relies on Deep Learning techniques for the accurate labeling of the detected objects. Representing state-of-the-art in various Computer Vision tasks such as object recognition and semantic understanding, Deep Learning is a crucial part of Semantic VSLAM [24].

2.2. The Robotic Vision Scene Understanding (RVSU) Challenge

The RVSU challenge was created by a group of researchers associated with the Australian Centre for Robotic Vision, with the goal of stimulating research in scene understanding for autonomous robotic agents. The semantic SLAM part of the challenge provides five simulated environments - House, Miniroom, Apartment, Company and Office - and three task types:

- Passive, Ground Truth: the robot follows a pre-defined path and ground-truth poses are available
- Active, Ground Truth: the user can input movement commands directly and ground-truth poses are available
- Active, Dead Reckoning: the user can input movement commands directly, but ground-truth poses are not available

The challenge also provides an evaluation metric for 3D cuboid object maps, named Object Map Quality (OMQ), and is discussed later.

Bosch Corporate Research Semantic SLAM Team

Bosch Corporate Research Semantic SLAM Team [21] is one of the teams that competed in the 2022 RSVU Challenge. Their method uses a semantic segmentation front-end combined with a custom foreground-background segmentation to create a point cloud per frame. They use two segmentation networks, Detectron2[1] and UPerNet Swin-B[2], which are applied to the RGB-D images collected from the robot. The output of both networks is fused to make the method more robust. Next, the point clouds are voxelized and the center of each voxel is determined by voting, resulting in a global point cloud. Bounding boxes are then extracted from this global point cloud by either top-down projection or clustering. Finally, size and confidence-based filtering are used to obtain the final detections. The described approach works for the passive control task, while this study focuses on active control. As such, their results couldn't be fairly compared to ours and were not included.

MSCLab

Another team that submitted to the 2022 RSVU challenge is the MSCLab team [12]. This team makes use of the 3D detector FCAF3D [17], which receives an RGB-D point cloud as input at each time step. However the network is not able to detect all object classes in the challenge. Therefore, they perform an additional instance segmentation step on the image using QueryInst [6]. If the object isn't detected by FCAF3D, QueryInst segments the instance of that object and fits a tight box from a set of predefined anchor boxes to it, simulating a bounding box. The result of this

[1] https://github.com/facebookresearch/detectron2
[2] https://github.com/open-mmlab/mmsegmentation

whole process is a list of detected objects, many of which have overlap between them. In order to account for this, the duplicates are merged if they pass a certain overlapping threshold, otherwise they are registered as new objects

2.3. VoteNet

VoteNet [15] is a 3D end-to-end object detection neural network that is built on the PointNet++ [16] backbone. As 3D data can be quite sparse, it creates proposals for the center of each object from the input point cloud. These proposals are then clustered and aggregated with the use of Hough Voting [9]. The output is comprised of bounding boxes with labels for the detected objects. The advantage of VoteNet over other 3D detection methods is that it handles 3D information directly, instead of converting the input point cloud to 2D before performing detection.

3. Methodology

The pipeline is comprised of multiple modules that work together to produce a semantically labeled 3D map. These modules were independently developed and used together to obtain the final results. In the following sections, we describe our approach towards implementing the components of the pipeline, as well as the difficulties we faced and the solutions we found to the encountered challenges. As a short summary, the pipeline first uses the depth image and the robot's intrinsic camera parameters to create a grayscale point cloud, where each pixel in the depth image is mapped to a 3D point in the point cloud. The 3D map is then created by combining the point clouds from all captured images. This process is done iteratively using fast global registration [27]. Votenet is then used for 3D object detection on the 3D map to identify objects and place bounding boxes around them. The code and installation instructions are publicly available [3].

3.1. BenchBot

Developing a solution to the challenge task is facilitated by the use of the BenchBot [20] software stack. The most important part of this software enterprise is the API, which allows the communication with the robot operating system (ROS) backend via high-level Python programming. Creating a solution in this context is a matter of overwriting an Agent class that restricts the input to the robot to two actions: forward and angular movement, while also offering RGB-D, LIDAR and pose information after each action. The result creation and evaluation methods are also implemented in the BenchBot stack.

[3] https://github.com/a-turcu/vslam-attempts

Robot movement

In our experiments, the robot is under active control in all environments. Specifically, the robot is first instructed to make a full rotation of $360°$, which allows for a full mapping of the smaller environment without generating significant artifacts. Afterward, the robot is moved around the environment in order to fully capture objects which might have been not been fully observed in the initial full rotation. Due to the limitations of the registration method, the robot only turns $10°$ at a time, either left or right, and moves forward 0.5 meters at a time.

3.2. Point Cloud Creation

The point cloud for each individual image is created directly from the depth map, using the intrinsic camera parameters. The z coordinate in the point cloud is given directly by the depth image while the x and y coordinates are calculated using the depth and the intrinsic camera parameters. Specifically, point coordinates in the point cloud are given by the formulas:

$$x = \frac{(u - p_x) * z}{f_x} \tag{1}$$

$$y = \frac{(v - p_y) * z}{f_y} \tag{2}$$

$$z = depth_{u,v} \tag{3}$$

where (u, v) are the image coordinates, z is the depth image value at point (u, v), f_x and f_y are the focal lengths while p_x and p_y are the coordinates of the principal point. The focal lengths and the principal point coordinates are taken from the camera intrinsic matrix, provided by the simulation. Using this method results in a grayscale point cloud. Colors are added subsequently by concatenating RGB information on top of the corresponding points in order to complete the point cloud.

3.3. 3D Mapping

3.3.1 Fast Global Registration

Stitching two point clouds at a time is done using the Fast Global Registration algorithm [27] implemented in the Open3D library [28]. The main idea behind this algorithm is to find a rigid body transformation between a source and a target point cloud. To this end, the first step involves extracting features from each point cloud using Fast Point Feature Histogram (FPFH) [18] feature descriptors. FPFH is a less computationally intensive implementation of Point Feature Histogram (PFH) [19], which encodes the geometrical properties of a point's neighborhood using the mean curvature around the point. Once the features are extracted, the algorithm matches the corresponding features based on

their descriptors and finds a rigid transformation that aligns the two point clouds. The main benefit of this algorithm over alternatives such as the Iterative Closest Point [26] registration algorithm is not only the speed, which is an order of magnitude faster than prior methods, but also the fact that it does not require an alignment for initialization [27].

3.3.2 Iterative registration

Our pipeline uses the Fast Global Registration algorithm iteratively to create the global point cloud in real-time. Making use of the point cloud of each RGB-D image, the 3D global map is created by iteratively applying Fast Global Registration to consecutive point clouds. Specifically, the algorithm starts by creating the point cloud of the initial RGB-D image seen by the robot. After each subsequent robot movement, a new point cloud is created from the new RGB-D image. Both the original and new point clouds are downsampled using a voxel size of $8 * 10^{-5}$. We found larger voxel sizes to result in overly sparse point clouds, while smaller voxel sizes required more computational time, without any visible improvements in the 3D map quality. No ablation experiments were conducted to test this value, but we generally found that values above 10^{-4} tend to produce degenerate 3D maps due to the increased sparseness while values below $5*10^{-5}$ require substantially more time with no apparent benefit. Fast Global Registration is then applied to the two downsampled point clouds, with the initial point cloud being the source and the new point cloud being the target. The resulting rigid-body transformation is then applied to the original point cloud before merging the two, not downsampled point clouds into a new combined point cloud. This process is applied iteratively, with the combined point cloud acting as the source while the new point cloud acting as the target. The pseudocode can be seen in Algorithm 1:

Algorithm 1 Iterative Fast Global Registration

```
pcd ← depth_to_pcd(RGB-D)
while robot is running do
    Get new RGB-D image
    new_pcd ← depth_to_pcd(new RGB-D)
    Downsample both point clouds
    Calculate FPFH geometric features
    transformation ← fgr(pcd, new_pcd, fpfh)
    Apply transformation to non-downsampled pcd
    pcd ← merge(pcd, new_pcd)
end while
```

3.4. Object Detection

The approach we have taken was to perform object detection on a global point cloud that is created by the iterative fast global registration method 3.3.2. As such, the need to remove duplicates is mitigated, as each object would only be detected once. This, however, forces the Votenet model to receive a global point cloud as input, which in our case has a considerable size that might affect the computational time. To account for this, the global point cloud was converted to grayscale and downsampled to 500000 points. The size was chosen empirically, as the point cloud needs to be large enough to accurately represent the room while also not to overwhelm the detection network. No ablation study was conducted since the 500000 value seemed to have an adequate performance.

3.5. Object Map Quality (OMQ) Evaluation

The OMQ, described in the RVSU challenge, uses the probability-based detection quality measure [7] to evaluate the quality of the generated labeled bounding boxes. These bounding boxes are comprised of their dimensions, location relative to the room and a list of probabilistic labels. From now on, we will refer to the bounding boxes as objects. The OMQ is composed of 4 other sub-metrics: pairwise object quality (pOQ), spatial quality (Q_{S_p}), label quality (Q_L) and false positives quality.

Spatial quality is calculated as the 3D Intersection of Union [25] score between the ground truth and the proposed object cuboids.

Label quality is computed by taking the probability assigned to the correct class. For example, if the proposed object has the following confidence scores: "chair":0.8, "table":0.15 and "toilet":0.05, while the ground truth object has the label "chair", then the label quality for the proposed object is 0.8.

Pairwise object quality is expressed as the geometric mean between the spatial and label quality of the proposed object (O_i) and the ground truth object (O_j). The ground-truth object with the highest non-zero pairwise quality is assigned to each object in the generated map. This assignment helps identify the "true positives," which are objects with a non-zero quality value and "false positives," which are objects in the generated map that don't have a matching pairwise quality value.

The false positive cost, a property of false positive objects, is the maximum confidence given to a non-background class. The **false positives quality** is a statistic that is equal to $1-$ false positives cost.

Finally, OMQ is computed as follows:

$$OMQ(M, \hat{M}) = \frac{\sum_{i=1}^{N_{TP}} q(i)}{N_{TP} + N_{FN} + \sum_{j=1}^{N_{FP}} c_{FP}(j)} \quad (4)$$

In the equation above, N_{TP} is the number of true positives (analogous for N_{FP} and N_{FN}), $q(i)$ is the pOQ of the i^{th} object and c_{FP} is the false positive cost.

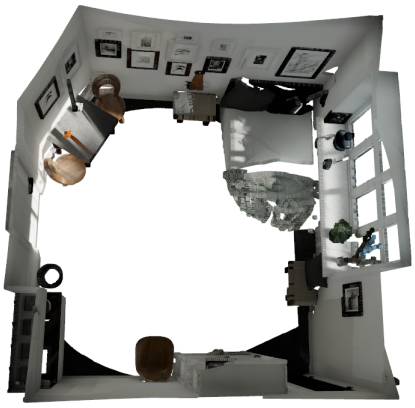

Figure 1. 3D reconstruction of the Miniroom environment.

3.6. Hardware

For development, we had access to 3 computers from the Intelligent Robotics Lab from the University of Amsterdam. Two of the machines were equipped with an Intel Core i7 9700 (8 cores/16 threads), 64 GB DDR4 and an NVidia RTX 2080 Ti (11 GB) with CUDA 10 and driver 415. The third machine was more powerful, being equipped with an Intel Core i9 10900KF (10 cores/20 threads), 64 GB DDR4 and an NVidia RTX 3090 (24 GB).

4. Results

The qualitative results consist of the point clouds generated with the fast global registration method. The point cloud of the Miniroom environment can be observed in Figure 1. The empty space on the floor is due to the fact that the camera is placed on top of the robot, such that the floor directly below is not visible to the robot. The artifacts that are visible are a result of errors in the computation of the rotations and translations between the captured images. It is worth mentioning that this specific point cloud has a resolution of approximately 51 million points. No voxel downsampling was used when visualizing this point cloud.

The point cloud that is used in the detection network and the resulting bounding boxes are displayed in Figure 2. The detection network does not need all of the detail shown in the point cloud from Figure 1, so it is downsampled to 500000 points and the color is removed. The displayed detections have a confidence score of over 90% and their labels are the following: bed, chair, table, bookshelf and nightstand. As a comparison, the ground truth bounding boxes applied over the same point cloud can be seen in Figure 3.

As for the quantitative results, the evaluation of the OMQ method and its sub-parts on the Miniroom environment can be seen in Table 1, alongside the results of the other teams (on all environments). The components of the OMQ are

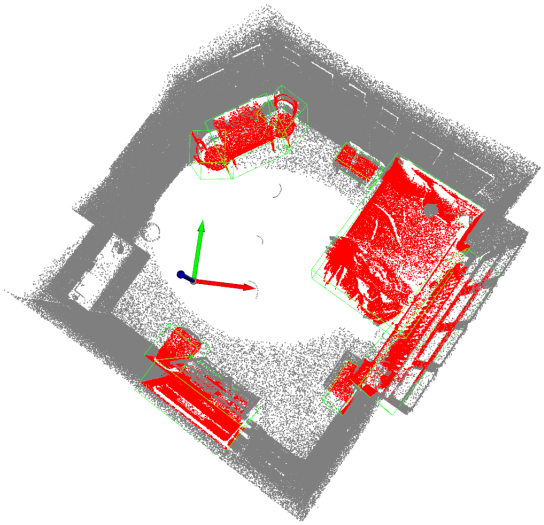

Figure 2. 3D segmentation of the Miniroom environment.

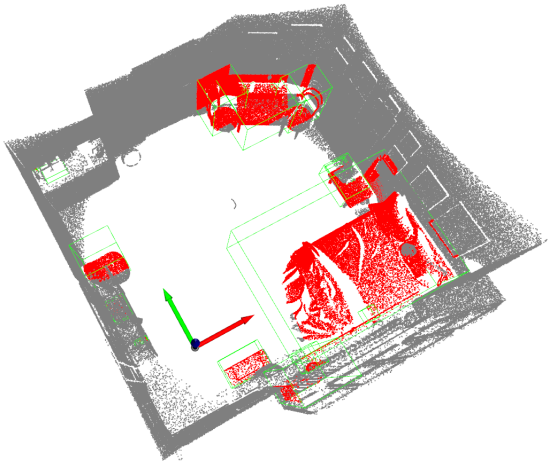

Figure 3. Ground truth bounding boxes on the Miniroom environment.

informative to our task and can be considered metrics on their own, so we have chosen to include them separately as well. A detailed analysis of the quantitative results can be found in Section 5.

5. Discussion

The obtained OMQ is 0.02, which means that we have achieved 2% of what is considered possible for this certain environment. The other sub-metrics hold more detailed information about our detection method:

- An average label quality of 0.99 means that the system correctly identifies the label of the proposed object 99% of the times, so it can be stated that the detection network accurately identifies the point cloud struc-

	Ours	SP	MSCLab	autoni
OMQ	0.02	**0.424**	0.298	0.242
Avg. label quality	0.99	0.895	0.997	**0.999**
Avg spatial quality	0.163	**0.575**	0.368	0.392
Avg. FP quality	**0.262**	0.146	0.020	0.001
Avg pairwise	0.402	**0.753**	0.575	0.591

Table 1. OMQ breakdown score of our approach compared to the teams from this year. Our approach runs in the Miniroom environment, but for the other teams, the score is a mean over all environments. All scores are collected in the (Active, GT) task.

tures.

- The average spatial quality of 16.3% indicates that the coordinates and the size of the predicted objects do not overlap with the ground truth objects most of the time.
- A low false positive score shows that the objects qualifying as false positives were assigned a high confidence, this being a negative indicator of our system. This is most likely caused by the fact that we assign the predicted objects only one class probability, which is usually very large ($\geq 90\%$). If the spatial or label quality of the object is sub-zero, this will cause the proposed object to be categorized as a false positive with a very high confidence, which destabilizes this metric. We believe that the poor result of this metric is caused by the combination of high confidence label and suboptimal spatial quality of the objects.

5.1. Challenges and difficulties

Creating the 3D Map

The first real challenge that we encountered was merging the individual point clouds in order to create the global 3D map. Multiple avenues were initially explored, both from a point cloud registration perspective as well as from a SLAM perspective. However, given the fact that the simulation already provides the ground truth values for the robot and camera poses, we chose to attempt the easier task of just building the map without taking localization into account. As such, two registration methods were implemented, multiway registration and fast global registration, the latter being the preferred choice.

While fast global registration was applied iteratively as explained in section 3.3.2, multiway registration is able to take multiple point clouds and merge them in a single pass, without the need for an iterative implementation. However, this implied collecting all observations from the simulation run before creating the map. Qualitatively, we could not observe any differences between the two created 3D maps. We instead opted for the iterative fast global registration approach, not only because it had a shorter computational time but also because it allowed for creating the map in real-time rather than at the end of the run.

The choice of creating the map via registration had consequences during the evaluation phase. Specifically, the coordinates of the created map were shifted compared to the global ground truth coordinates. This had no effect on detecting the objects themselves but rather on comparing our resulting bounding box coordinates with their ground truth counterpart. Our attempts to solve this issue are explained in the next section.

Camera to World coordinate matching

The bounding boxes output by the Votenet detection network have their coordinates centered around the camera that the global point cloud is seen from. The point cloud (camera) and the environment (world) coordinates are offset from each other, so it is not possible to directly use the coordinates from the point cloud system as the results. One of the hardest challenges we have faced involved transforming the coordinates of the bounding boxes such that they match the world coordinates.

It is important to note that the poses the robot goes through are available. By using those, it is possible to build a transformation matrix that should modify the points as to match the world environment. Thus, our first attempt was to convert the global point cloud to world coordinates before passing it to the detection network. However, this method did not work with the rest of our approach, as the detection network would not return any results on the transformed point cloud.

Secondly, we tried to transform the coordinates of the first point cloud used in the global registration to world coordinates. We had expected that, by concatenating the posterior point clouds, the final model would inherit the coordinates of the first frame. However, having the first frame and the subsequently combined point clouds as a target for the registration algorithm resulted in degenerate maps. Therefore, we had to continue using the new observed point cloud as the target.

Lastly, we have decided to convert every single point cloud that was used in the registration process to the world coordinates before stitching them together. This resulted in a global point cloud that, again, did not return any results when used as input for the detection network.

We have failed to come up with a feasible solution given the time constraint of the course, so we have decided on a hard-coded version of our first idea. Having observed that the path of our robot starts with a 360° turn to the left, we decided that we would empirically compose a transformation matrix that would revert this turn. This improvised method has enabled us to obtain some results, albeit not as accurate as we would have expected. The fact that a

global transformation was possible has led us to believe that a transformation matrix like the one we found empirically could be determined by pose calculations, however this remains a topic for further study.

Testing on larger environments

When trying to test our pipeline on larger environments, we realized that the robot controller could not connect to the simulation. This was the case for the house, apartment, and company environments while the office environment was working only when running on the more powerful machine using the RTX 3090 GPU. On machines using the RTX 2080 Ti GPU, we were only able to test the Miniroom environments. To be more specific, while the simulation would indeed start, the robot controller would fail to connect. We assume this might be the case because of longer waiting times required to load and run the simulation.

5.2. Limitations

Limited detected classes

The main limitation of the pipeline is the fact that the 3D object detection algorithm is unable to recognize most objects in the environment since it was trained on a different set of objects. As such, the algorithm can only detect 5 classes out of the 25 present in the ground truth list (chair, table, couch, bed and toilet). Additionally, the detection is only as good as the created point cloud map, which also depends on how much the robot was allowed to explore the environment and which path was taken. Therefore, some objects in the scene might not be recognizable due to their incomplete point clouds.

Stitching Point Clouds

A second limitation comes in the form of the registration algorithm, which is highly susceptible to mapping errors in case of large discrepancies between incremental camera poses. For this reason, the robot only moved forward a maximum of 0.5 meters and only turned $10°$ at a time. In addition, the algorithm is also highly dependent on the path the robot takes throughout the environment. For instance, scenes that are too similar but positioned differently might be combined thus resulting in a degenerate mapping. Therefore, the path had to be carefully selected, often through trial and error. However, even this carefully selected path was not enough to fully and correctly map larger maps such as the office, which has many similar objects and textures. The resulting partial point cloud can be seen in the Appendix, Figure 4 while a degenerate 3D map of the office can be seen in Figure **??**.

5.3. Future work

Given the limitations of our pipeline, two main avenues for improvements could be taken. Firstly, to improve the 3D maps, fast global registration could be replaced with deep global registration [5], which has shown state-of-the-art performance when it comes to point cloud registration. Alternatively, the registration method could be replaced altogether with a proper SLAM method such as ORB-SLAM2 [13]. Secondly, to ensure more objects are detected in the environments, the 3D object detection method could be replaced with a 2D object detection method, which was trained to recognize more classes. However, this would require transforming and keeping track of 2D bounding boxes in 3D coordinates. This would be the approach most similar to what other participants to the RVSU challenge had done, which has the potential advantage of the robot being aware of what it is looking at, which was shown to improve SLAM performance [11, 22, 23].

6. Conclusion

We propose an implementation and analysis of a 3D semantic mapping pipeline centered around fast global registration and 3D object detection in the context of the Robotic Vision Scene Understanding challenge. We show that our implementation is able to achieve an OMQ score of 0.02, while confined to a small environment and reliant on a predefined movement path. This low OMQ score is caused by issues regarding creating the 3D map, aligning the camera coordinates to the world coordinates, and the limited number of overlapping classes between the object detector and the environment. We discuss some potential solutions to these issues in the report, as we believe that with those improvements, our pipeline could represent a competent solution to the RVSU challenge.

Appendix

Fast global registration results in the office environment

Figure 4. Incomplete 3D map of the office environment.

References

[1] Rana Azzam, Tarek Taha, Shoudong Huang, and Yahya Zweiri. Feature-based visual simultaneous localization and mapping: A survey. *SN Applied Sciences*, 2:1–24, 2020. 2

[2] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics*, 32(6):1309–1332, 2016. 2

[3] Kaiqi Chen, Jianhua Zhang, Jialing Liu, Qiyi Tong, Ruyu Liu, and Shengyong Chen. Semantic visual simultaneous localization and mapping: A survey, 2022. 2

[4] Weifeng Chen, Guangtao Shang, Aihong Ji, Chengjun Zhou, Xiyang Wang, Chonghui Xu, Zhenxiong Li, and Kai Hu. An overview on visual slam: From tradition to semantic. *Remote Sensing*, 14(13):3010, 2022. 1

[5] Christopher Choy, Wei Dong, and Vladlen Koltun. Deep global registration, 2020. 7

[6] Yuxin Fang, Shusheng Yang, Xinggang Wang, Yu Li, Chen Fang, Ying Shan, Bin Feng, and Wenyu Liu. Instances as queries. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6910–6919, 2021. 2

[7] David Hall, Feras Dayoub, John Skinner, Haoyang Zhang, Dimity Miller, Peter Corke, Gustavo Carneiro, Anelia Angelova, and Niko Sünderhauf. Probabilistic object detection: Definition and evaluation, 2020. 4

[8] David Hall, Ben Talbot, Suman Raj Bista, Haoyang Zhang, Rohan Smith, Feras Dayoub, and Niko Sünderhauf. The robotic vision scene understanding challenge, 2020. 1

[9] Bastian Leibe, Aleš Leonardis, and Bernt Schiele. Robust object detection with interleaved categorization and segmentation. *International journal of computer vision*, 77:259–289, 2008. 3

[10] Ping Li, Guoqing Zhang, Jianluo Zhou, Ruolong Yao, and Xuexi Zhang. Study on slam algorithm based on object detection in dynamic scene. In *2019 international conference on advanced mechatronic systems (ICAMECHS)*, pages 363–367. IEEE, 2019. 2

[11] Ziwei Liao, Yutong Hu, Jiadong Zhang, Xianyu Qi, Xiaoyu Zhang, and Wei Wang. So-slam: Semantic object slam with scale proportional and symmetrical texture constraints. *IEEE Robotics and Automation Letters*, PP:1–1, 2021. 7

[12] MSCLab. Robotic vision scene understanding challenge: Msclab report. 2022. 2

[13] Raul Mur-Artal and Juan D. Tardos. ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, oct 2017. 7

[14] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE international symposium on mixed and augmented reality*, pages 127–136. Ieee, 2011. 2

[15] Charles R. Qi, Or Litany, Kaiming He, and Leonidas J. Guibas. Deep hough voting for 3d object detection in point clouds, 2019. 1, 3

[16] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 3

[17] Danila Rukhovich, Anna Vorontsova, and Anton Konushin. Fcaf3d: fully convolutional anchor-free 3d object detection. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part X*, pages 477–493. Springer, 2022. 2

[18] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE International Conference on Robotics and Automation*, pages 3212–3217, 2009. 3

[19] Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, and Michael Beetz. Learning informative point classes for the acquisition of object model maps. In *2008 10th International Conference on Control, Automation, Robotics and Vision*, pages 643–650. IEEE, 2008. 3

[20] Ben Talbot, David Hall, Haoyang Zhang, Suman Raj Bista, Rohan Smith, Feras Dayoub, and Niko Sünderhauf. Benchbot: Evaluating robotics research in photorealistic 3d simulation and on real robots, 2020. 1, 3

[21] Bosch Corporate Research Semantic SLAM Team. Panoptic hierarchical semantic slam. 2022. 2

[22] Han Wang, Jing Ying Ko, and Lihua Xie. Multi-modal semantic slam for complex dynamic environments. *ArXiv*, abs/2205.04300, 2022. 7

[23] Wenxin Wu, Liang Guo, Hongli Gao, Zhichao You, Yuekai Liu, and Zhiqiang Chen. Yolo-slam: A semantic slam system towards dynamic environment with geometric constraint. *Neural Computing and Applications*, 34:6011 – 6026, 2022. 7

[24] Dan Xu, Andrea Vedaldi, and João F Henriques. Moving slam: Fully unsupervised deep learning in non-rigid scenes. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4611–4617. IEEE, 2021. 2

[25] Jun Xu, Yanxin Ma, Songhua He, and Jiahua Zhu. 3d-giou: 3d generalized intersection over union for object detection in point cloud. *Sensors*, 19(19), 2019. 4

[26] Zhengyou Zhang. Iterative point matching for registration of free-form curves and surfaces. *International journal of computer vision*, 13(2):119–152, 1994. 4

[27] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast global registration. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pages 766–782. Springer, 2016. 1, 3, 4

[28] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3d: A modern library for 3d data processing. preprint arXiv:1801.09847, 2018. 3