Semantic Mapping and Autonomous Navigation for Agile Production System*

Benchun Zhou[1], Jan-Felix Klein[1], Markus Hillemann[2] and Bo Wang[2]

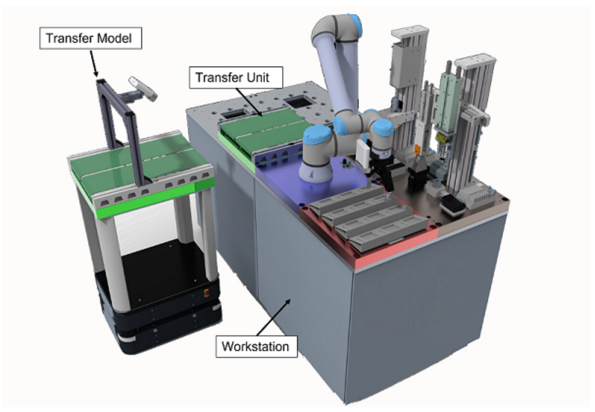
***Abstract*— A typical task for mobile robots in production logistics is to transport objects from one location to another. This requires the robots not only to locate the objects, but also to design a collision-free transport path. Currently, many mobile robots operate in an occupancy map, require a predefined goal pose as a destination, and lack the availability of high-level navigation. With the aid of RGB-D cameras, semantic objects can be detected and added to the map, providing more opportunities for scene understanding and flexible navigation. In this paper, we extend the current 2D mapping and navigation framework with object segmentation and fine position navigation to achieve better performance in task-level navigation. First, we propose a framework for creating and maintaining a hypermap by recognizing semantic objects in the environment and integrating them into an existing 2D occupancy map. Second, we present a coarse-to-fine navigation strategy on this hypermap. The coarse navigation receives object information from the map and designs a global path towards the destination, while the fine navigation utilizes the local information to ensure a precise docking to the workstation. A field experiment demonstrates that the proposed system can achieve high performance in autonomous navigation in a production logistics environment.**

Keywords: semantic mapping, hypermap, autonomous navigation, application

I. Introduction

Automated Ground Vehicles (AGVs) and Autonomous Mobile Robots (AMRs) are common solutions for automating the movements of goods between different zones in the warehouse. To autonomously navigate in the environment, these machines are usually equipped with various sensors, such as cameras or laser scanners, which enable them to perceive the surroundings and design a collision-free path towards the target. Furmans et al. [1] proposed that future material handling systems will focus on flexibility and should be easily adapted to new tasks or transferred to new locations.

The AgiProbot project (Agile PROduction system using mobile, learning roBOTs with multi-sensors for uncertain product specifications) [1] developed at the Karlsruhe Institute of Technology (KIT) includes a typical production logistics environment, where the robots are tasked to deliver goods between different workstations [2]. As shown in Fig. 1, the transport module (mobile robot) should recognize the transfer unit on the top of the workstation, design a collision-free path towards the destination, and perform a robust and accurate docking at the destination. To successfully transfer objects



(a) AgiProbot simulated environment.

(b) AgiProbot real environment.

Fig. 1: The AgiProbot project. The transfer module (mobile robot) is tasked to transfer goods among different workstations. Specifically, the robot should recognize the transfer unit (conveyor) on the top of the workstation, design a collision-free path towards the destination, and dock to the workstation precisely.

from the vehicle to the workstation, both the conveyors of the transport module and the transfer unit should be precisely aligned.

For planar navigation in an indoor environment, an occupancy map is well-studied and widely used in real-world applications [3]. It decomposes the space into a fixed-size grid, and each cell of the grid indicates that area is occupied or not. This idea is simple and practical to ensure safe navigation. However, this map contains only occupancy information and lacks semantics, which makes

*The AgiProbot project is funded by the Carl Zeiss Foundation

[1]Benchun Zhou and Jan-Felix Klein are with Institute for Material Handling and Logistics (IFL), Karlsruhe Institute of Technology, 76131, Germany. [2] Bo Wang and Markus Hillemann are with Institute of Photogrammetry and Remote Sensing (IPF), Karlsruhe Institute of Technology, 76131, Germany. benchun.zhou@kit.edu

[1]http://agiprobot.de/

scene understanding difficult. Moreover, the navigation goal can only be in a fixed position, which is not convenient and friendly for users. In the future, task-level navigation, such as "get me a coffee" seems more intuitive and convenient. For this purpose, important objects or places in the environment need to be detected and introduced to the map [4]. On the other hand, a fine navigation strategy is essential for real world applications, particularly when it comes to meeting docking requirements like those in a charging situation [5].

In this paper, we present an autonomous navigation pipeline that aims to realize semantic mapping and task-level navigation in a production logistics environment. For the mapping process, we first segment semantic objects from RGB-D images and project them onto the existing grid map. By introducing object information into the grid map, the robot can better understand the scene and achieve a versatile navigation. Then, we develop a coarse-to-fine navigation strategy to realize an accurate arriving. While the coarse navigation designs a fast and collision-free path towards the object in the map, the fine navigation uses local features and ensures an accurate docking process. Furthermore, we evaluate our system on a robotic platform to show the use-case in real applications. In summary, the contributions of this paper are as follows:

- A semantic mapping system to efficiently create a hypermap that appends semantic objects to the existing occupancy map.
- A coarse-to-fine navigation pipeline which uses this hypermap to ensure safe and precise navigation.
- A field experiment in a production logistics environment to demonstrate the effectiveness of the entire system.

II. Related Work

The related work is divided into semantic mapping and planar navigation. First, we review the main works that focus on creating 2D grid maps and integrating objects to the maps. Then, previous methods on object-based navigation are discussed.

A. Grid mapping with object information

A 2D occupancy grid map is a common solution for indoor planar navigation, where the environment is represented as a fixed, regular grid, and each cell indicates the probability that this area can be traversed or not. The 2D map can be automatically generated with laser scan data using Simultaneous Localization and Mapping (SLAM) algorithms, such as Gmapping [3], Hector SLAM [6], and Cartographer [7]. Although occupancy maps are convenient for navigation, the lack of semantic information limits their suitability for high-level tasks.

To address this limitation, additional information can be incorporated into the map to improve environmental understanding and navigational flexibility. Some researchers explore segmenting objects directly from the laser scan data. Leigh et al. [8] separated a scan into several connected groups and extracted hand-crafted features to label these points. By assigning different labels to the laser points, a meaningful map can be created. However, the 2D laser scanner is limited to detection in a 2D plane and cannot detect objects at different heights, such as a book on the table. In contrast, Himstedt et al. [9] presented a visual grid mapping method, where they detected objects from RGB-D images, converted the labeled point cloud into annotated scan data, and integrated it into a probabilistic SLAM framework to generate a semantic map. Although cameras provide an efficient way to detect objects, the map accuracy needs to be improved.

A multi-sensor system is developed to fuse laser data and camera information. For example, based on a pre-built 2D occupancy grid map, Sivananda et al. [10] detected objects from an RGB image, searched for a corresponding object in a database and added an approximate object model to the existing map. Pang et al. [11] attempted to segment objects from the laser points. They detected objects in the images, projected laser points into the images and assigned the corresponding classes to the laser points, if the projected points were within bounding boxes. Zaenker et al. [4] proposed to build a hypermap with multiple layers, such as the occupancy layer and semantic polygon layer, which can be generated separately and used for different purposes. On this basis, Dengler et al. [12] proposed an online mapping and object updating system, where the objects are represented with various 2D and 3D models.

The semantic mapping process in our paper is similar to [12]. The main difference is that we develop a new object geometry computation method and a new object association strategy to maintain the 2D representation. Besides, this 2D representation can directly benefit autonomous navigation.

B. Planar navigation with object information

A collision-free navigation in a 2D grid map can be guaranteed because the map contains occupancy information. However, one of the limitations is that we need to define the goal with fixed coordinates in the map (for example $x = 0.5$, $y = 1.0$), which is not flexible and user friendly. In contrast, mobile robots with task-level capabilities can navigate with object information. They can locate the position of an object and design a path to reach the object. For example, Dengler et al. [12] provided a demonstration of the task-level navigation. Upon receiving the command "get me the cup at the coffee machine", the mobile robot checks the position of the coffee machine, autonomously navigates to the destination and transports the cup back to the user. In addition, object-based maps can provide additional occupancy information to protect the robot from collisions. Sivananda et al. [10] built a map with objects, where the chair can provide more occupancy information than only legs. Given the same navigation goal, the mobile robot can design a better path to avoid the collision with the chairs.

In our paper, we aim to recognize semantic objects as the navigation goals and develop a coarse-to-fine strategy to realize task-level navigation. Apart from designing a global path to the objects, we focus on the fine-position docking

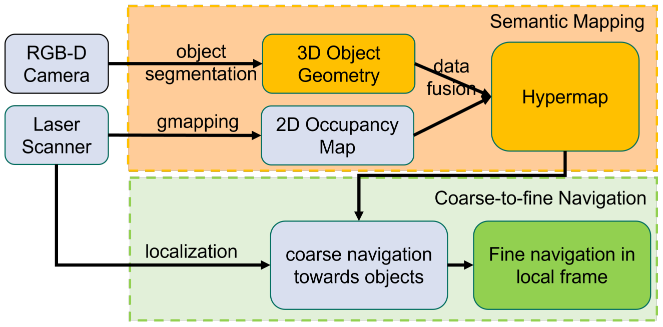


Fig. 2: Overview of our autonomous navigation pipeline. For semantic mapping, the laser scan data is used to generate a 2D occupancy map, while the RGB-D sequences are fed to the object segmentation framework to estimate semantic objects. For navigation part, the coarse navigation designs a collision-free path towards the object, while fine navigation ensures a precise dock.

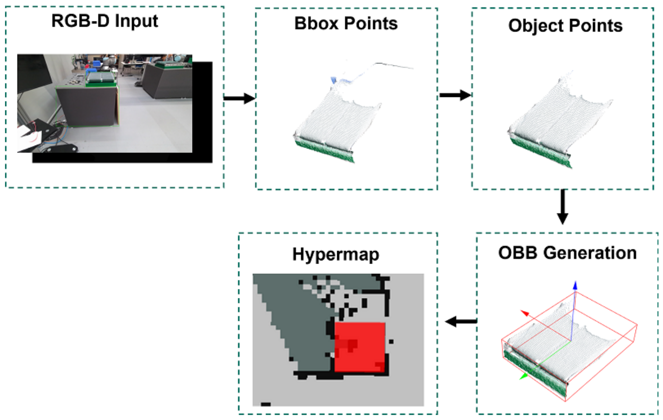


Fig. 3: Object segmentation process. Taking RGB-D images as input, we detect object bounding boxes and extract the points inside the bounding boxes, then, we utilized a robust outlier exclusion algorithm to remove points that does not belong to the objects, finally, the object is represented as geometric cuboid and project onto the existing map.

process, where we use local semantic features and slowly approach the goal with high precision.

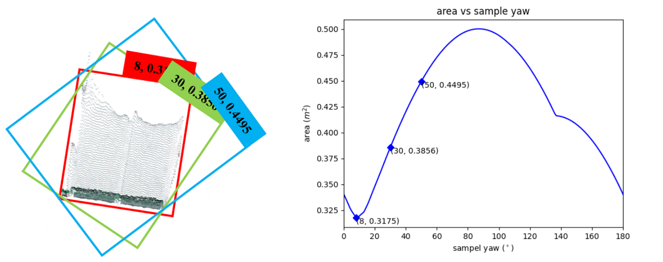
III. Method

The proposed navigation pipeline is illustrated in Fig. 2, which builds on the top of an off-the-shelf navigation system. First, the 2D occupancy map is generated by a laser scanner with a probabilistic SLAM system [3]. Next, 3D semantic objects are detected and segmented from RGB-D sequences, which are projected onto the existing occupancy map to generate a hypermap (Section III-A). Then, we use a coarse-to-fine navigation system to drive the robot to the object, where the coarse navigation can design a path to the destination without collision, and the fine navigation uses current local information to approach the docking position safely and precisely (Section III-B).

A. Semantic mapping system

The goal of the semantic mapping system is to generate a hypermap consisting of an occupancy and a semantic layer, where each cell of the space should be assigned with semantic and occupancy information. Given RGB-D sequences, we segment 3D objects with geometry representations and introduce them into the existing occupancy map. Looking at the AgiProbot project as an application [2], we aim to detect the transfer unit which has fixed size and can be observed completely. Its position and rotation can directly benefit task-level navigation.

1) Object segmentation: The $transferunit$ segmentation process is shown in Fig. 3. For each incoming RGB-D frame, a 2D object detector is used to predict object class categories along with their image bounding boxes (IBB). The point cloud inside the bounding box can be easily obtained from the depth image. Next, a robust outlier exclusion algorithm is applied to cluster object points. Since the $transferunit$ has a cuboid shape, it can be formulated by the surface planes. We estimate the planes from the object point cloud, check their spatial relationship, and collect surface planes points.



(a) Heading angle sampling. (b) Areas under different angles.

Fig. 4: Object geometry computation. (a) We perform a dense sample on heading angle from 0° to 180°. (b) The area of oriented bounding boxes under different heading angles.

Since there are many discrete outliers in the preliminary object model due to sensor noise and detection errors, we perform a Euclidean cluster extraction [13] to cluster object points. Thirdly, the object is represented as an oriented bounding box (OBB) in the 2D map (x, y, θ, w, l), where x, y represent the center of the object, and θ, w, l are the heading angle, width, and length. For this purpose, we project all clustered object points onto the map plane, and draw OBB to cover them. Here, we perform a discrete sampling of heading from 0° to 180° and record the areas of sampled OBB, as illustrated in Fig. 4. We observe that the smallest OBB matches the object points best, from which we can obtain the position, heading angle, and dimension (x, y, θ, w, l) to represent the object.

2) Object association and hypermap update: The estimation results from a single frame may not be accurate because the object can be partially observed. Therefore, we introduced an object association strategy to incrementally model the object through multiple frames. In this case, we treat the existing global objects as a database, find the correspondences between the frame-detected objects and stored objects, and decide whether to update the existing objects or add new objects to the database. In general, the detected object A can be associated with a mapped object B if they meet the following constraints:

First, for image detection, they should have the same class categories and their 2D bounding box in the image should have an Intersection over Union (IoU) higher than a threshold t. In our experiments, we choose $t = 0.5$.

$$IoU_{2D} = \frac{IBB_{2D}(A) \cap IBB_{2D}(B)}{IBB_{2D}(A) \cup IBB_{2D}(B)} \quad (1)$$

Second, we compare their appearance similarity, where we convert the object images to the HSV (hue, saturation, value) color space, generate normalized feature vectors $HSV(A)$, and $HSV(B)$, and calculate the correlation coefficient between these vectors. A higher correlation means a higher similarity, we set the threshold as 0.5.

$$Correlation = \frac{HSV(A) \times HSV(B)}{\sqrt{HSV(A)^2 + HSV(B)^2}} \quad (2)$$

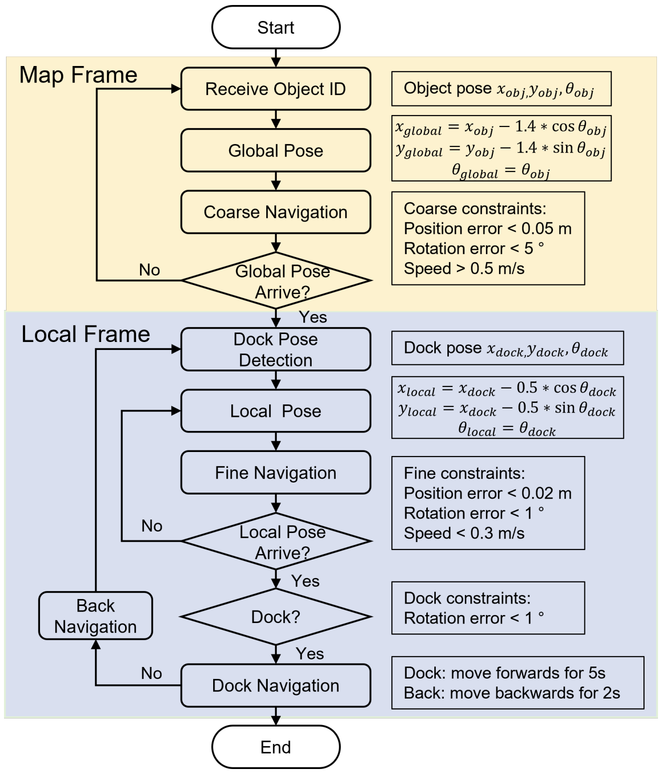
Thirdly, after projecting on the map, we calculate the IoU between the two projected oriented bounding boxes (OBB), where the threshold is set as 0.8 to search for a robust association.

$$IoU_{3D} = \frac{OBB_{2D}(A) \cap OBB_{2D}(B)}{OBB_{2D}(A) \cup OBB_{2D}(B)} \quad (3)$$

If there is no existing mapping object that matches the frame object, it should be initialized as a new object and added to the map, providing essential information such as $\{Frame\_ID$, $Mapped\_ID$, $class$, $2d\_bbox$, x, y, θ, w, l, HSV, $3D\ points$, ... $\}$. On the other hand, if the association is found, the frame object should be merged into the associated mapped object. The semantic information ($Frame\_ID$, $class$, $2d\_bbox$ and HSV) is replaced by new information, while the 3D points are accumulated, and the OBB parameters will be computed again by projecting and dense sampling as described before. Finally, the object geometry is matched with the occupancy information to create a hypermap, where the edge of the OBB should be aligned with the black grid in the map.

B. Coarse-to-fine navigation strategy

After having a map with occupancy information, the general navigation strategy [14] can design a path to the goal while taking collisions into account. This strategy focuses more on "position" than "orientation" and prefers to perform a "rotating-reaching-rotating" behavior when approaching the destination. However, it is not practical in AgiProbot project, because the robot cannot rotate when it is too close to the workstation. Therefore, the robot must rotate beforehand and subsequently dock in a linear motion. Since we have built a hypermap with occupancy and object information, a coarse-to-fine navigation strategy can be designed to reach the object autonomously. The workflow and schematic diagram is shown in Fig. 5 and 6, which will be described in the following:



Start
Map Frame
Receive Object ID
Object pose $x_{obj}, y_{obj}, \theta_{obj}$
Global Pose
$x_{global} = x_{obj} - 1.4 * \cos\theta_{obj}$
$y_{global} = y_{obj} - 1.4 * \sin\theta_{obj}$
$\theta_{global} = \theta_{obj}$
Coarse Navigation
Coarse constraints:
Position error < 0.05 m
Rotation error < 5 °
Speed > 0.5 m/s
No
Global Pose Arrive?
Yes
Local Frame
Dock Pose Detection
Dock pose $x_{dock}, y_{dock}, \theta_{dock}$
Local Pose
$x_{local} = x_{dock} - 0.5 * \cos\theta_{dock}$
$y_{local} = x_{dock} - 0.5 * \sin\theta_{dock}$
$\theta_{local} = \theta_{dock}$
Fine Navigation
Fine constraints:
Position error < 0.02 m
Rotation error < 1 °
Speed < 0.3 m/s
No
Local Pose Arrive?
Yes
Back Navigation
Dock?
Dock constraints:
Rotation error < 1 °
Yes
No
Dock Navigation
Dock: move forwards for 5s
Back: move backwards for 2s
End

Fig. 5: Workflow of our proposed coarse-to-fine navigation pipeline. the coarse navigation takes object information as input to design a global path, while the fine navigation define a dock pose in the local frame to ensure a precise dock.

1) Coarse navigation: To start, the robot will receive an "Object ID" to obtain the object geometry, and a "global pose" can be defined in front of the conveyor ($x_{global} = x_{obj} - 1.4 \times \sin(\theta_{obj})$, $y_{global} = y_{obj} - 1.4 \times \cos(\theta_{obj})$, $\theta_{global} = \theta_{obj}$), which leaves enough space for the robot to rotate safely. Then, a global path considering obstacle avoidance and relaxed constraints is performed to navigate the robot to the target.

2) Fine navigation: After reaching the predefined "global pose", the robot might have some position and rotation errors due to accumulated drift. Depending on the global map might be inaccurate, instead, we trust local information and estimate objects again in local frame. Here we choose laser data because the laser scanner has a wider view angle when compared to camera. For each scan, We segment it into several lines [15] and find the closest "workstation line" with conveyor. Under the setting that conveyors are aligned with workstation edges, we can define a dock pose ($x_{dock}, y_{dock}, \theta_{dock}$) on the "workstation line" as illustrated in Fig. 8, the dock position is the perpendicular point from the object center to the workstation line, while the heading angle comes from the line. After obtaining the final dock pose, we set a "local pose" in the front of it to ensure the precision: ($x_{local} = x_{dock} - 0.5 \times \sin(\theta_{dock})$, $y_{local} = y_{dock} - 0.5 \times \cos(\theta_{dock})$, $\theta_{local} = \theta_{dock}$). a local path considering a slower speed and strict constraints is designed

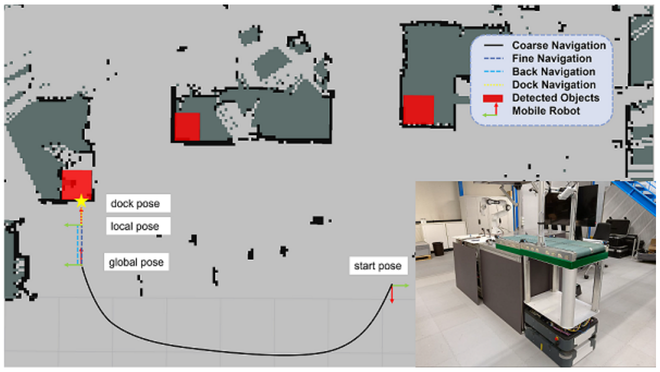


Fig. 6: Schematic diagram for autonomous navigation which illustrates the workflow in Fig. 5, the mobile robot is tasked to navigate to the left object, and it traverses from "start pose" to "dock pose".

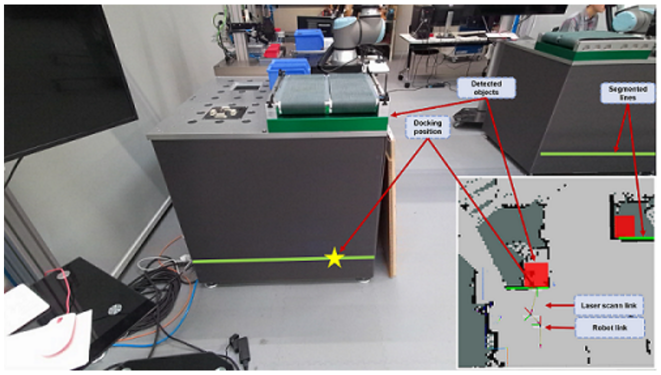


Fig. 7: An example for "dock pose" detection. The "work-station line" is close to the object in the map, and the "dock pose" lies on the "workstation line".

and drives the robot to the "local pose". Finally, we check the position and rotation again. In case the mobile robot pose does not match the "dock pose", we design a loop scheme to drive the robot backwards and activate the fine navigation again. Otherwise, we can enable the dock navigation to drive the robot forwards without any rotation. Since all information is in the local frame, there is no drift or delay during the process.

IV. Experiment

A. Experiment setting

We evaluate our proposed system on a real robotic platform (i5-7300U, 2.60GHz CPU, 8GB RAM, no GPU, Ubuntu 16.04), which is equipped with a 2D SICK laser scanner and a Microsoft Azure Kinect camera, as shown in Fig. 9.

The SICK laser scanner[2] is installed horizontally at a height of 0.3 m and in the north-west corner of the vehicle. It can measure a maximal distance as of 30 m with field of view of 270° and publishes the scan data with 30 Hz. The

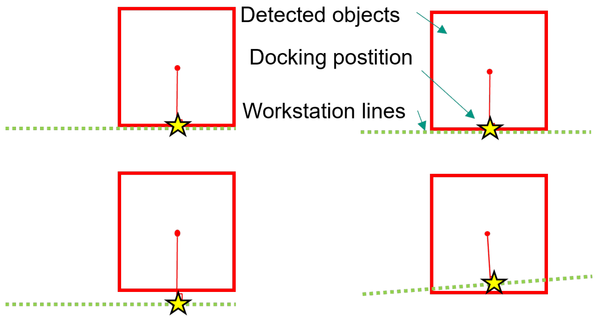
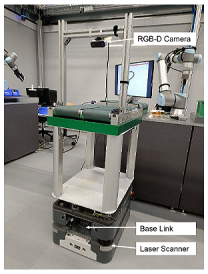
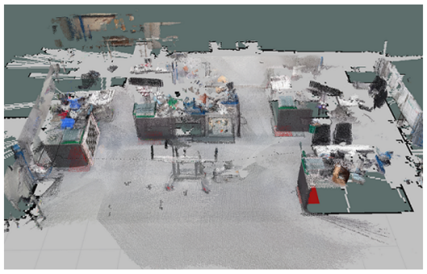


Fig. 8: More examples for "dock pose" detection. The position is the perpendicular point from the object center to the workstation line, while the heading angle comes from the line.



(a) The mobile robot.



(b) The point cloud map of the test environment.

Fig. 9: The Agiprobot environment.

Microsoft Azure Kinect camera[3] is mounted at a height of 1.4 m on the top of the vehicle center, facing downwards at 45°. It captures RGB and depth images in 720P resolution (1280x720) with a field of view of $90° \times 60°$ and has a frame rate of 15 Hz. Calibration[4] between laser scanner and camera is done with the method proposed by Zhang et al. [16].

B. Mapping result

To generate an offline hypermap, we use the joystick to drive the mobile robot around the whole environment $(6 * 12m)$ and record laser scan data and RGB-D images. The laser scan data are used to generate a 2D occupancy map by running Gmapping packages [3]. The RGB-D sequences are fed to the object segmentation framework to estimate semantic objects. The dependency of robot localization in 2D map is solved by ICP registration [17] on laser scan data, which achieves a better result than wheel odometry in robot poses tracking. On the basis, a point cloud map of the environment is visualized, from which we can manually annotate the object ground truth. For object detection, 100 images containing $conveyors$ are captured and manually annotated to train a Mask-RCNN [18], The trained Mask-RCNN is used to detect $conveyor$ with bounding boxes in every frame.

[2]https://www.alliedelec.com/product/sick/s30b-3011gb/70872466/

[3]https://learn.microsoft.com/en-us/azure/Kinect-dk/hardware-specification

[4]https://github.com/MegviiRobot/CamLaserCalibraTool

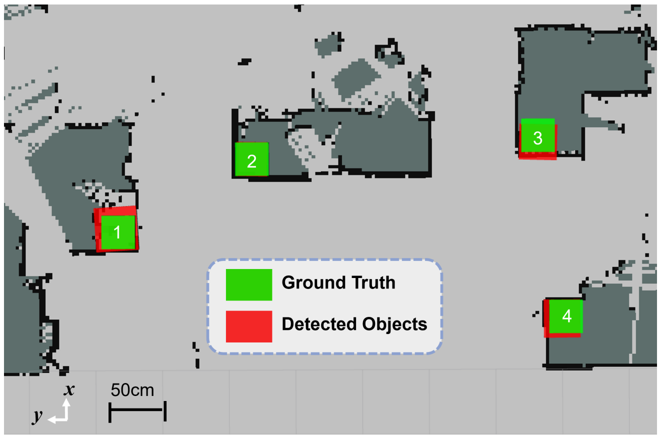


Fig. 10: The generated hypermap, where the truth objects and detected results are represented as green and red oriented bounding boxes.

TABLE I: The results of 3D object segmentation

Conveyor	IoU	translation error (m)	rotation error(°)
1	0.59	0.17	4
2	0.69	0.07	1
3	0.82	0.08	1
4	0.85	0.06	2
Average	0.7375	0.095	2

The reconstructed hypermap is shown in Fig. 10, it has a resolution of $5cm$ and contains an occupancy layer and an object layer that can be used for different purposes. 4 conveyors are detected and added to the map, which can act as navigation goals to enable task-level navigation. In addition, We also plot the ground truth on the map for a comparison. Further analysis on object segmentation is evaluated with following metrics:

Object IoU: The object IoU is calculated by comparing the OBB in the map between the detected objects and ground truth, which shows the general similarity of the two objects.

Translation Error: The translation error measures the Euclidean distance between the center of mass of two objects, indicating the displacement of the detected objects.

Rotation Error: The rotation error is computed by the heading angle difference, presenting the rotation inaccuracy of the observed objects compared to the ground truth.

As shown in Table I, the average object IoU is averagely 0.7375, and the translation error and angle error are $0.095m$ and $2°$ respectively. Two important factors may influence the results: One is sensor noise. We observe that when the robot is rotating, the 2D object detections and point cloud measurements are inaccurate and will deteriorate the segmentation result. We propose to remove these bad detections. The other is the object updating strategy. We simply accumulate object points and compute object geometry, it is hard to remove outliers. A better idea is to calculate and update the probabilities of all corresponding points.

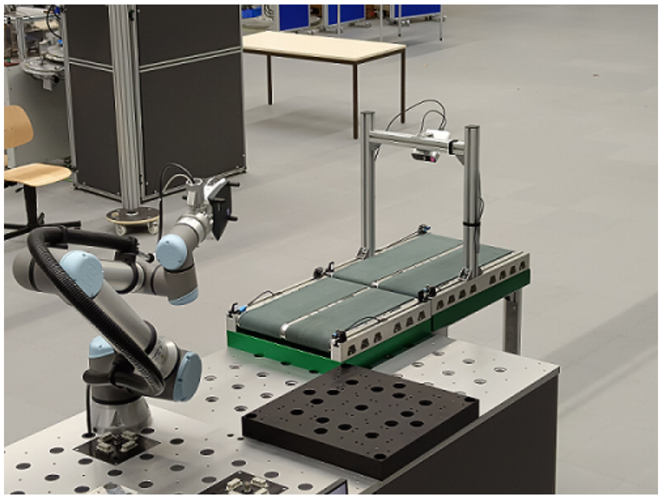


Fig. 11: An example of docking navigation in AgiProbot project, where two conveyors on the top of vehicle and workstation should be aligned precisely.

TABLE II: Results of task level navigation. We measure the result by comparing the distance error (in X, Y direction) and the angle difference.

Conveyor	Position	ROS Navigation (m, m, °)	Ours (m, m, °)
1	Left	(0.010, 0.010, 2)	(0.00, 0.010, 0)
2	Right	(0.010, 0.020, 3)	(0.00, 0.005, 1)
3	Left	(0.010, 0.010, 3)	(0.00, 0.010, 1)
4	Right	(0.020, 0.010, 2)	(0.02, 0.00, 0)

C. Navigation result

Taking the hypermap as input, the vehicle is able to achieve task-level autonomous navigation with high precision. We evaluate the task-level navigation by sending a "Conveyor ID" to the mobile robot, which indicates the navigation goal. We also implement ROS navigation [14] for comparison, since it doesn't receive "Conveyor ID", we manually define the "dock pose" in the map frame as the navigation goal to ensure they will arrive at the same place. For each "conveyor ID", we test 5 times and record the average errors in Table II. It can be observed that ROS navigation has an average error of $0.01\ m$ in translation and $3°$ in rotation. Although it has a good reaching position, but the angle error is larger than expected, making it difficult for docking application. Our coarse-to-fine strategy can significantly improve docking results. It reaches $0.01\ m$ in translation, and $1°$ in rotation. The recognition of $workstation$ from laser scan data defines the "dock pose" in the local frame, which eliminates the accumulated drift error and providing an accurate orientation of the goal. In addition, we also provide a video [5] to demonstrate the autonomous navigation with high docking accuracy in a real environment.

[5]https://github.com/benchun123/object-based-navigation.git

TABLE III: Average execution time of each processing module in our proposed system.

Module	Tasks	Runtime (Sec)
Mapping	Occupancy mapping	200
	Object mapping	135
Navigation	Coarse navigation	30
	Fine navigation	20

D. Runtime performance

We also evaluate the runtime performance of the system, the result is shown in Table III. The run-time experiment is conducted in a real environment ($6 * 12m$) with a robotic platform. Regarding different tasks, the occupancy mapping task that takes laser scan data to generate an occupancy map, requires $200s$, the object mapping task introduces objects to the existing map, which takes $135s$. The mapping process heavily depends on the size of the scenario, for sub-process, such as object detection, segmentation and updating, will take average $725ms$, $15ms$, and $15ms$ per frame respectively. For navigation, it requires average $30s$ and $20s$ to accomplish the coarse and fine navigation. Since we focus more on final dock performance, the time cost is acceptable.

V. Conclusion

In this paper, we presented an autonomous navigation pipeline for the purpose of object-based mapping and navigation. Firstly, an offline hypermap with an occupancy and object layer is created, where the occupancy layer is generated by off-the-shelf SLAM method using laser scan data, while the objects are segmented from RGB-D sequences and projected on the top of the 2D grid map. Then, a coarse-to-fine navigation strategy is designed to achieve task-level navigation. To this end, the coarse navigation takes object information as input to design a global path, while the fine navigation defines a dock pose in the local frame to ensure a precise dock. We evaluate our system on the AgiProbot project, and the results demonstrate that our system can successfully dock to the workstation with high performances and achieve a flexible material handling process.

Due to the specific intra-logistics environment, the proposed system is only suitable for AgiProbot project now. In the future, we focus on exploiting generating navigation strategies for more scenarios. Besides, during the mapping process, we first create a 2D occupancy map, and then add object information. It is worth exploring to simultaneously assign the grid space with occupancy and object information.

References

[1] K. Furmans, Z. Seibold, and A. Trenkle, "Future technologies in intralogistics and material handling," in *Operations, Logistics and Supply Chain Management*. Springer, 2019, pp. 545–574.

[2] J.-F. Klein, M. Wurster, N. Stricker, G. Lanza, and K. Furmans, "Towards ontology-based autonomous intralogistics for agile remanufacturing production systems," in *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2021, pp. 01–07.

[3] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[4] T. Zaenker, F. Verdoja, and V. Kyrki, "Hypermap mapping framework and its application to autonomous semantic exploration," in *2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. IEEE, 2020, pp. 133–139.

[5] X. Zhang, X. Li, and X. Zhang, "Automatic docking and charging of mobile robot based on laser measurement," in *2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, vol. 5. IEEE, 2021, pp. 2229–2234.

[6] S. Kohlbrecher, O. Von Stryk, J. Meyer, and U. Klingauf, "A flexible and scalable slam system with full 3d motion estimation," in *2011 IEEE international symposium on safety, security, and rescue robotics*. IEEE, 2011, pp. 155–160.

[7] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2d lidar slam," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 1271–1278.

[8] A. Leigh, J. Pineau, N. Olmedo, and H. Zhang, "Person tracking and following with 2d laser scanners," in *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2015, pp. 726–733.

[9] M. Himstedt and E. Maehle, "Online semantic mapping of logistic environments using rgb-d cameras," *International Journal of Advanced Robotic Systems*, vol. 14, no. 4, p. 1729881417720781, 2017.

[10] K. P. Sivananda, F. Verdoja, and V. Kyrki, "Augmented environment representations with complete object models," in *2022 31st IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2022, pp. 1123–1130.

[11] L. Pang, Z. Cao, J. Yu, S. Liang, X. Chen, and W. Zhang, "An efficient 3d pedestrian detector with calibrated rgb camera and 3d lidar," in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2019, pp. 2902–2907.

[12] N. Dengler, T. Zaenker, F. Verdoja, and M. Bennewitz, "Online object-oriented semantic mapping and map updating," in *2021 European Conference on Mobile Robots (ECMR)*. IEEE, 2021, pp. 1–7.

[13] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 1–4.

[14] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige, "The office marathon: Robust navigation in an indoor office environment," in *2010 IEEE international conference on robotics and automation*. IEEE, 2010, pp. 300–307.

[15] S. T. Pfister, S. I. Roumeliotis, and J. W. Burdick, "Weighted line fitting algorithms for mobile robot map building and efficient data representation," in *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, vol. 1. IEEE, 2003, pp. 1304–1311.

[16] Q. Zhang and R. Pless, "Extrinsic calibration of a camera and laser range finder (improves camera calibration)," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3. IEEE, 2004, pp. 2301–2306.

[17] A. Censi, "An icp variant using a point-to-line metric," in *2008 IEEE International Conference on Robotics and Automation*. Ieee, 2008, pp. 19–25.

[18] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.