Semantic Map Based Robot Navigation with Natural Language Input

Guang Yang, Xinchi Huang, and Yi Guo

***Abstract*— We present a new semantic map based robot navigation system in the paper. The system takes human voice input, processes multi-modal data including natural languages and RGB-D images, and generates semantic maps for robot navigation. Making use of recent development in image segmentation tools, we integrate robot mapping and localization with a customized real-time object detection model, so that the semantic and navigation layers are efficiently combined for robot navigation purpose. We demonstrate the performance of our developed algorithms in both simulation and real robot experiments. Compared with existing works, we demonstrate applicability to real robot systems and superior performance in terms of success rate.**

I. Introduction

There is an increasing demand for robots to assist humans in daily tasks [1], where the robot is expected to understand humans and respond to human instructions using natural languages. With the recent development in the field of large language models, it is now possible to integrate natural language input with semantic mapping, so that autonomous mobile robots can take verbal commands from humans directly and navigate in everyday environments.

Recent studies have targeted the problem of vision-and-language navigation (VLN), where visual and linguistic features are extracted and integrated through cross-modal attention mechanisms. Pioneering works in [2], [3] have demonstrated good performances incorporating vision and language modalities in simulation environments (e.g., the Matterport3D simulator [2]). Due to the limited dataset of the associated Room-to-Room environment, these algorithms are prone to overfitting and are overly sensitive to simulation-specific scenarios, which makes it difficult to adopt under complex real-world conditions. As described in [4], when these algorithms were tested in real-world environments, even environments simpler than the simulator, the success rate was around 50% only, underscoring the challenges in generalization. Another line of research develops a generalist agent (GA) using transformer models [5], [6]. While GAs demonstrated capabilities in addressing various problem types, their high failure rate in a single task makes it difficult to be used in robotic navigation tasks due to potential collisions that will damage robots. Challenges exist in extending simulation results to real-world navigation tasks in indoor environments. Next, We briefly review related methods addressing the challenges.

This work was partially supported by the US National Science Foundation under Grants CMMI-1825709 and IIS-1838799.

The authors are with the Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ 07030, USA. Emails: {gyang11, xhuang60, yguo1}@stevens.edu

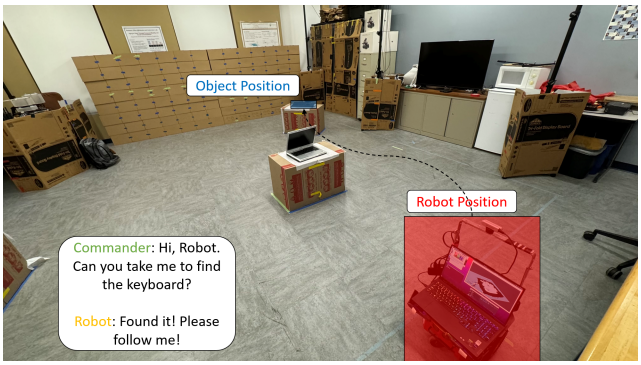


Fig. 1: The proposed semantic map based robot navigation system.

A. Related Work

Semantic Map: With the development of SLAM techniques and the application of convolutional neural networks (CNNs) for semantic understanding, semantic mapping has gained renewed interest. Recent works [7], [8], [9], and [10] focus on annotating 3D maps with dense semantic information using 2D semantic segmentation and adopting object-centric strategies. These strategies aim to build 3D maps centered around detected objects to facilitate pose-graph optimization at the object level. A significant limitation has been the extensive need for manual annotation of semantic labels. The introduction of You Only Look Once (YOLO) [11], trained on the Common Objects in Context (COCO) [12] dataset, addressed the challenge of manual labeling by automating object detection. However, relying solely on RGB images for semantic mapping can lead to inaccuracies in object location estimation and issues with perspective occlusion.

Large Language Model: Large Language Models (LLMs) are designed to estimate the probability $p(W)$ for a sequence $W = \{w_0, w_1, w_2, ..., w_n\}$, where each sequence is made up of strings $w_i, i = 1, \ldots, n$. This is typically done by breaking down the probability using the chain rule, leading to $p(W) = \Pi_{i=0}^{n} p(w_i|w_{<i})$, in which the next string is predicted from the preceding ones. The well-known models such as Transformers [13], BERT [14], T5 [15], GPT-3 [16], have demonstrated significant growth in their capacities. With billions of parameters in the neural network and extensive training on large text corpora, their capability to generalize over multiple tasks has been demonstrated. In our current work, we make use of the recent development and

apply BERT as an encoder for initial text preprocessing. We then utilize TextCnn [17] combined with Fully Connected Layer (FC) as the classification decoder to pull out specific goals from the natural language input, which facilitates the generation of high-level executable tasks.

Vision-and-Language Navigation (VLN): VLN represents a challenging yet pivotal research area in embodied artificial intelligence, as it involves developing agents capable of following instructions within real-world scenarios [18]. The development of large-scale datasets in Matterport3D [19], Habitat [20], and Gibson [21] has enabled performance evaluation of agent navigation in photorealistic environments. Using these simulation tools, researchers have provided various solutions focusing on cross-modal alignment and decision-making learning [3], [22]. Initial studies predominantly employed Long Short-Term Memory (LSTM) as the backbone [23]. With the development of BERT [14], recent approaches have shifted toward pre-trained vision-and-language BERT models [24], [25], which have demonstrated superior performance over traditional LSTM-based baselines.

Despite many VLN algorithms exhibiting satisfactory performance in simulators, they face significant limitations when applied to real-world tasks [26], [27]. The cross-modal attention mechanism does not always work well and the performance degradation has been observed when transiting from simulations to actual robotic navigation environments [26]. This discrepancy primarily arises due to the inherent difference between real-world navigation and simulated ones. Current simulation models predominantly utilize navigation graphs, which present limitations when applied to real-world environments, as these graphs assume precise robot localization, but actual robot navigation systems rely on sensor estimation with inherent sensing noises and localization uncertainties. Also, current VLN algorithms have limited generalization capabilities, as these algorithms trained in simulations do not handle unseen scenes, objects, or navigation instructions in everyday environments well [27]. Current VLN algorithms have yet to prove their efficacy in real-world environments.

B. Main Contribution

Addressing this gap, in this paper, we propose a robot navigation system that integrates both natural language processing and semantic maps, and demonstrates performances in both simulation environments and also in real-world settings. Specifically, we develop a novel method that effectively integrates vision and language for robot navigation. Our multi-modal robotic system combines a large language model, semantic Simultaneous Localization and Mapping (SLAM), and an automatic task executor to achieve autonomous navigation. This system ensures the safety of navigation without collision while accurately identifying desired goal objects and navigating to them in both simulation and real-world environments. Performance evaluation demonstrates our proposed method achieves a better success rate in real-world environments than existing VLN methods.

Our main contribution includes: 1) taking advantage of the recent development in natural language processing and image segmentation, our method integrates multi-modal data efficiently and enhances robot navigation capability; 2) comparing with classic robot navigation methods, where the task planner (i.e., choosing a goal position) and motion planning (i.e., planning a path to the given goal position) are separated in different design steps, our robot system takes human natural language as input, and automatically extracts goal positions and passes to the motion planner, thus the task planner and the motion planner are integrated seamlessly; and 3) comparing with existing VLN methods, our system can be deployed directly to real-world environments due to the adoption of the navigation map and semantic SLAM that can efficiently handle sensor uncertainties and localization errors in real-time.

II. The System and Problem Statement

We use a Pioneer-3DX mobile robot with an onboard RGB-D camera and a laptop (with CPU and GPU, detailed configuration given in Section V) that runs Robot Operating System (ROS) [28]. The input to the system is a voice command in natural language, L. The system output is the robot's velocity control $V = (v, \omega)$, where v represents the linear velocity and ω represents the angular velocity. Our navigation problem is defined as: *for a given voice input L with intent object expressed in natural language, the robot extracts human intent, plans a path, and generates velocity control V to the robot so that the robot navigates to the desired goal location.*

Fig. 1 shows an example of our system. A commander gives a voice input: "Hi, robot. Can you take me to find the keyboard?" The robot responds: "Found it! Please follow me", and starts to navigate to the location of the object – the keyboard, avoiding obstacles in the environment.

Our system needs to process multi-modal data that includes the voice input L and the RGB-D camera data, I. The system integrates a task generator and motion planner and the robot navigates to the desired location autonomously. We present our method in the next section.

III. Proposed Method

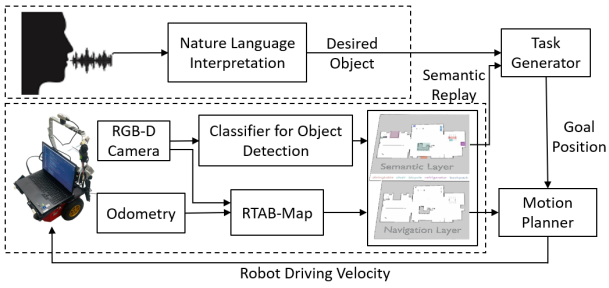


Fig. 2: Overall architecture of the system that takes human voice command as input, and outputs velocity control for robot autonomous navigation.

A. Overall Architecture

As shown in Fig. 2, our robot system receives the voice input L and generates velocity output V to drive the robot to the desired location. Our proposed method consists of three modules: 1) Semantic map generation that detects objects and generates a semantic map for localization and navigation; 2) Language processing for intent recognition, that is, processing the voice input and extracting the intent object as the goal location; (3) Navigation that includes task generation and motion planning for the robot to navigate to the goal location.

In the semantic map generation module, the robot makes use of its onboard odometry and the RGB-D sensor, and goes through a standard SLAM process using the existing RTAB-Map method [29], which generates a 2D grid map of the environment to be used as the "Navigation Layer" for navigation. Meanwhile, the RGB-D data is used for an object detection classifier that generates the "Semantic Layer". Combining the Semantic Layer and the Navigation Layer, the location and semantic label of objects are paired and stored in a look-up table D. When a desired object is extracted from the natural language interpretation module, the task generator generates a goal position by searching the look-up table D. Then, the motion planner plans a path using the 2D map generated to navigate to the desired goal position.

In the following, we describe our method in more detail and present the key algorithms.

B. Classifier for Object Detection

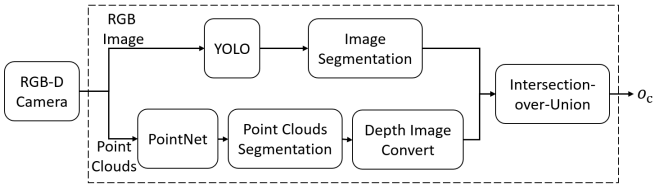


Fig. 3: The block diagram of the classifier for object detection.

In the semantic map generation module, a key block is the classifier for object detection. As shown in Fig. 3, the classifier receives input from the RGB-D camera, processes both the RGB images and the depth point clouds, goes through a few functional blocks, and outputs the object information in the camera frame, $o_c = (p_c, l)$, where $p_c = (x, y, z)$ denotes the object's position in the camera frame and l represents the semantic label of the object.

We integrated PointNet [30] and YOLO-v3 to detect objects and their position information. To enhance model performance, we use The Stanford 3D Indoor Spaces Dataset [31] to train PointNet and use the SUN Dataset [32] for fine-tuning the YOLO-v3 model, improving object detection accuracy in indoor environments.

PointNet is configured with a series of five shared MLP layers, utilizing global max pooling to process point clouds data effectively. YOLO-v3 employs a robust structure of 75 convolutional layers with local max pooling for object detection. Selected global max pooling and ReLU for PointNet, and local max pooling with Leaky ReLU for YOLO-v3 as pooling strategies and activation functions, ensuring efficient and synergistic feature extraction and mapping. Operational parameters such as batch sizes and learning rates are specifically adjusted to enhance training outcomes and system performance, and are provided in TABLE III in the Appendix.

For any given input RGB image I, the YOLO network generates a set of masks $M = \{m_1, m_2, ..., m_n\}$. Each mask m_i provides a pixel-wise delineation of a detected object, along with its associated class label l_i. Simultaneously, the point cloud data P inputs into PointNet for point clouds segmentation. PointNet processes this data to segment the point clouds input into distinct regions, denoted as $R = \{r_1, r_2, ..., r_m\}$. Each region r_j encompasses the point clouds data of the objects.

Matching segmentation of objects from 2D image data from YOLO to 3D point clouds data from PointNet, we transfer the 3D point clouds into 2D depth image first. We apply the Intersection-over-Union (IoU) align metric. For every mask m_i and depth region r_j, the IoU is calculated as:

$$\text{IoU}(m_i, r_j) = \frac{\text{PixelCount}(m_i \cap r_j)}{\text{PixelCount}(m_i \cup r_j)} \tag{1}$$

The IoU score is evaluated so that each object detected in the image aligns with its 3D point clouds. A mask and depth region are determined to be a match if their IoU score exceeds a threshold, $\tau = 0.8$.

Algorithm 1 details the classifier for object detection.

C. Coordinate Transformation to Global Map Frame

As o_c contains the object information in the camera frame, to generate a semantic map, we need to transform the object's coordinates from the camera frame to the global map frame. The object information in global frame representation is denoted as $o_g = (p_g, l)$, where $p_g = (x_g, y_g, z_g)$ represents the object coordinate in the global map and l represents its semantic label.

We use a transformation matrix, T_{cg}, from RTAB-Map, and T_{cg} is a 4 by 4 homogeneous matrix that includes the rotation and translation required to convert coordinates from the camera frame to the global map frame. It can be represented as follows:

$$T_{cg} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where r_{ij} are the elements of the rotation matrix, describing the orientation of the camera frame relative to the global

Algorithm 1 Classifier for Object Detection

Input: RGB Image I, Point Clouds P
Output: List of Objects o_c

Initialize YOLO-v3 model with trained weights
Initialize PointNet model with trained weights
Perform object detection on I to get bounding boxes and labels from YOLO-v3
For each detected object in I, extract segmentation masks $M = \{m_1, m_2, ..., m_n\}$
Use PointNet on P to segment into object regions $R = \{r_1, r_2, ..., r_m\}$
Project segmented point clouds to a 2D plane with a depth image
for each mask m_i in M **do**
  for each region r_j in R **do**
    Calculate

$$\text{IoU}(m_i, r_j) = \frac{\text{PixelCount}(m_i \cap r_j)}{\text{PixelCount}(m_i \cup r_j)}$$

  end for
  if $\text{IoU}(m_i, r_{j^*}) > \tau$ **then**
    Record the match of m_i with r_{j^*} and calculate the coordinates (x, y, z)
    Assign the corresponding label l
    Add (x, y, z, l) to o_c
  end if
end for
return o_c

frame, and (t_x, t_y, t_z) is the translation vector, indicating the position of the camera frame's origin in the global frame.

Algorithm 2 describes the details of the coordinate transformation.

D. Semantic Map Generation

We use the existing SLAM algorithm, RTAB-Map [29], to fuse the robot odometry data and the RGB-D camera data, which utilizes probabilistic graph optimization techniques to establish a map and perform real-time localization. The RTAB-Map module outputs a 2D grid map. We then generate the semantic map by integrating o_g and the 2D grid map. o_g is stored in a look-up table D, with the objects' coordinates and semantic labels. This information is used for the task generator to formulate navigation tasks, thereby creating a semantic map that not only represents physical space but also embeds semantic information of the objects in the space.

E. Nature Language Processing for Intent Extraction

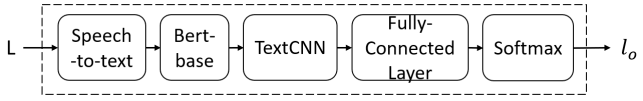


Fig. 4: Nature Language Interpretation to extract the intent.

Algorithm 2 Transforming Object Information from the Camera Frame to the Global Map Frame

Input: o_c
Output: o_g

Retrieve the current transformation matrix T_{cg} from RTAB-Map, representing the camera frame to the global frame transformation.
Represent p_c as a homogeneous coordinate $p_c^h = (x_c, y_c, z_c, 1)^T$.
Compute $p_g^h = T_{cg} \cdot p_c^h$ to transform the object position to the global frame.
Convert back from homogeneous coordinates p_g^h to Cartesian coordinates to get $p_g = (x_g, y_g, z_g)$.
$o_g \leftarrow (p_g, l)$
return o_g

When a robot receives a command, such as "Hi, robot, can you help me find my cellphone?", the robot needs to extract the intent, i.e., finding the desired object ("cellphone"). We design a neural network to process linguistic information to achieve this task. Upon receiving the voice command, the Google Cloud Speech-to-Text service transcribes the voice input into textual content. A BERT-based model is then designed as the encoder and TextCNN-Fully Connected layers-softmax is designed as the decoder for classification. Fig. 4 shows the functional blocks of this module.

The textual context is tokenized before input to the model. BERT generates a sequence of embedding for the text. It is designed as an encoder because of its ability to leverage contextual embeddings and deep bidirectional representations, thus capturing intricate linguistic patterns and dependencies. The pre-trained BERT-based fine-turned with ChatGPT generated textual data to enhance the performance in our classification problem. It consists of 12 transformer blocks, 12 self-attention heads, and a hidden size of 768. Given T as the tokenized representation of the input, the embedding E can be represented as: $E = BERT_{encoder}(T)$.

Using BERT only has limitations in overfitting on smaller datasets training. Using TextCNN-Fully Connected Layer-Softmax as a decoder can improve inference speed and provide an effective capture of local textual features like key phrases and patterns. It can also improve the model's ability to generalize to new data, potentially boosting the accuracy of intent recognition tasks without significantly compromising performance speed.

TextCNN architecture includes 256 filters across three parallel convolutional layers, utilizing kernel sizes of 2, 3, and 4 to capture n-gram features. Following the convolutional layers, the model integrates a fully connected layer comprising 256 units to further process the extracted features, with ReLU activation, followed by a dropout of 0.3 to reduce overfitting. The architecture concludes with an 80-unit fully connected layer, corresponding to the 80 categories within the COCO dataset. A softmax activation function is applied to this final layer, to compute a probability distribution over

the 80 classes, thereby enabling the model to predict the most likely category for a given input. TextCNN layers process the embedding: $G = TextCNN(E)$. Fully Connected Layer and Softmax designed to produce a probability distribution across the predefined intents and objects: $l_o =$ Softmax$[FC(G)]$.

Algorithm 3 Nature Language Interpretation

Input: Voice command L
Output: Object name l_o
1: $Text \leftarrow$ SpeechToText(L)
2: $T \leftarrow$ Tokenizer$(Text)$
3: $E \leftarrow BERT_{encoder}(T)$
4: $G \leftarrow TextCNN(E)$
5: $l_o \leftarrow$ Softmax$[FC(G)]$
6: **return** l_o

Algorithm 3 shows the procedure for interpreting natural language commands spoken to a robot. The voice input converts speech to text, then tokenizes to T, processes T through a BERT encoder, and utilizes the decoder to extract the desired object which is used to generate goal position in the task generator.

F. Robot Navigation

As described above, the generated semantic map consists of the look-up table D with the semantic label of the desired object l_o. We use `move_base` [33] as the motion planner, which is the open-source ROS navigation stack's package, designed for robot navigation tasks. The navigation employs a 2D grid map M, previously generated by RTAB-Map, as the navigation map. The navigation algorithm has D, l_o, M as input and the robot driving velocity V as the output.

Algorithm 4 details the navigation steps. The task generator searches l from the D for matching l_o in line 1. If a match is not found, the robot notifies the commander of the absence of the object using audio output in lines 2 and 3. Otherwise, the corresponding coordinates are set as the goal position and published to the motion planner in lines 5 and 6. The robot starts to plan navigation once receiving the goal position. The real-time odometry and camera scan data are input into the RTAB-Map package for localization in lines from 8 to 10. `Move_base` is then used to output the velocity control V that drives the robot toward the goal in line 11. A voice command is announced by the robot to ask the user to follow it in line 12. Meanwhile, the classifier of object detection keeps updating the semantic map and the look-up table D by updating o_g in lines 13 and 14.

Algorithm 4 Robot Navigation

Input: Desired object l_o, Look-up table D, 2D-grid map M
Output: V
1: $l \leftarrow$ Search(D, l_o)
2: **if** l is not found **then**
3: Inform the commander that the object is not found via audio output
4: **else**
5: $goal\_position \leftarrow$ GetCoordinate(l)
6: Publish $goal\_position$ to `move_base`
7: **while** navigation is in progress **do**
8: $odom \leftarrow$ odometry
9: $s \leftarrow$ camera scan
10: Robot's localization using RTAB-Map input $(M, odom, s)$
11: $V \leftarrow$ `move_base`(M)
12: Announce and ask the commander to follow the robot via audio
13: $o_g \leftarrow ClassifierOfObjectDetection(s)$
14: Update semantic map and D by o_g
15: **end while**
16: **end if**
17: **return** V

IV. Simulation Results

In this section, we present the simulation setup and results and evaluate the performance of our proposed algorithms.

A. Simulation Environment

We use the ROS Gazebo simulator and modified AWS Robomaker World [34] small home and bookstore as our home and library testing environment. To better mimic a real-world setting, we added and placed the additional objects such as apples, cups, laptop, backpack, and 11 other models in the environments. Fig. 5 shows a residential setting with a bedroom, living room, and a kitchen-dining area. The space is filled with standard home furniture, including sofas, televisions, dining tables, and chairs, to mimic a realistic home-like environment for algorithm testing. Fig. 6 shows a library environment that includes tables, chairs, books, laptop, clock, etc. The robot is randomly placed in the simulation environment, a commander gives a voice instruction such as "Hi Robot, can you take me to the refrigerator?" The robot extracts the commander's desired object from the voice input and autonomously plans and executes the navigation task. We randomly select objects as the desired goal based on the items available in the simulation environment. For each test scenario, we conducted 100 simulation trials to evaluate the performance.

B. Performance Metrics

We use the following metrics to evaluate the performance:

- **Navigation Error (NE):** Measuring the distance between the robot's actual end position and the desired goal position, expressed as: $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$, where (x_1, y_1) is the desired goal position and (x_2, y_2) is the robot's actual end position.
- **Success Rate (SR):** It is defined as the percentage of navigation attempts in which the robot successfully reaches the desired object indicated by the commander's

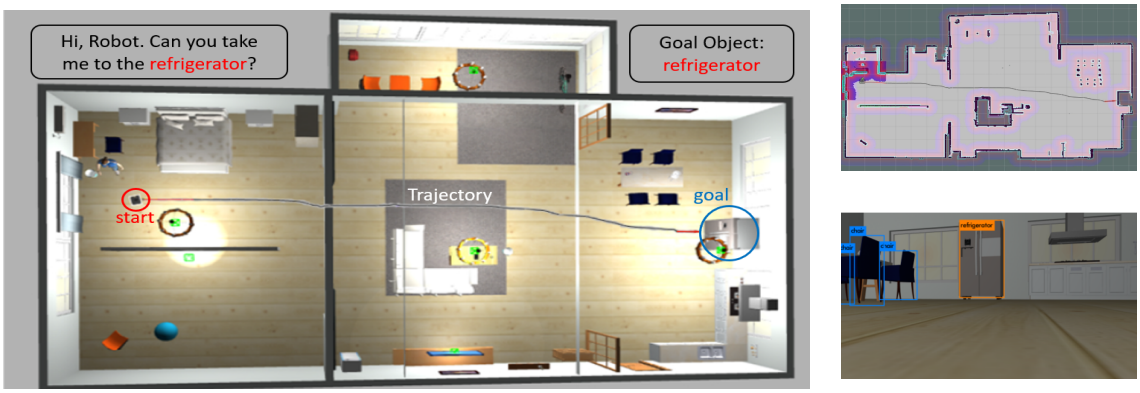


Fig. 5: ROS Gazebo simulation: a home environment. The robot receives voice input from the commander to navigate to the desired object. The home environment (left) with visualization of the navigation map (right top) and onboard camera view with objection detection (right bottom).

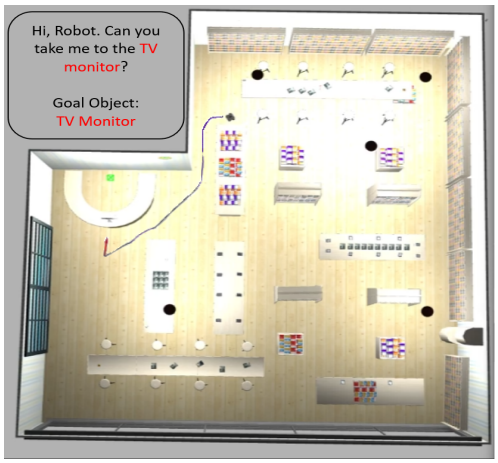


Fig. 6: ROS Gazebo simulation: a library environment.

voice input without any collisions. It is calculated as: $SR = \frac{\text{Number of successful attempts}}{\text{Total number of attempts}} \times 100\%$.

- **Success Weighted by Path Length (SPL)**: We follow the same performance metric as defined in [35], and use SPL that combines the success rate with the efficiency of the path taken. It is defined as: $SPL = \frac{1}{N}\sum_{i=1}^{N} S_i \frac{l_i}{\max(p_i, l_i)}$, where N is the total number of attempts, S_i is a binary indicator of success in the i-th attempt, l_i is the shortest path distance from the robot start position to the goal, and p_i is the actual path length traveled by the robot. SPL measures the efficiency of the actual path against the shortest possible path when the navigation task is completed.

C. Results

TABLE I: Simulation results.

Environment	Mean NE (SD)	SR	SPL
Home	0.16m (0.08m)	92%	0.74
Library	0.20m (0.13m)	93%	0.75

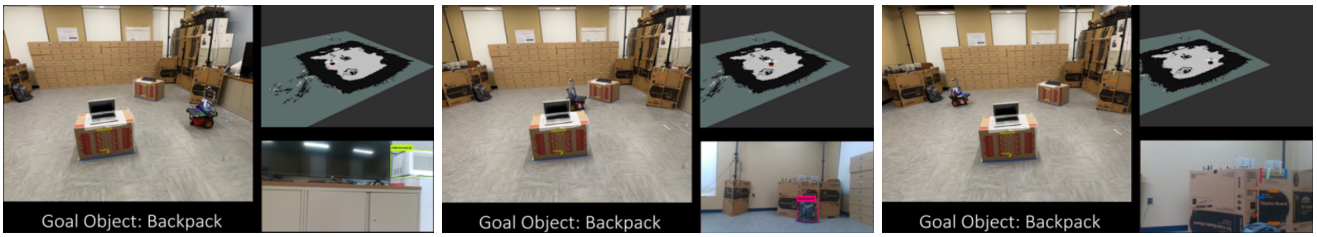
Our simulation results are shown in Table I using the defined performance metrics, where the mean value and the standard deviation (SD) of NE, SR, and SPL are provided in each of the two testing environments. In the home environment, the mean value of NE is 0.16m with a standard deviation (SD) of 0.08m. The success rate for this setting is 92%, and SPL is 0.74. Among those unsuccessful cases, voice recognition inaccuracies led to task failure in 3% of the trials, navigational challenges in constrained environments accounted for failures in 3% of the cases, and 2% of misidentification for object detection resulted in incorrect object navigation.

The library environment has more complex spatial layouts, thus the robot has narrower space to maneuver than in the home environment. It has a slightly higher mean value of NE at 0.20m with an increased SD of 0.13m, suggesting a slightly lower accuracy in navigation performance compared to the home environment. The success rate is slightly higher at 93%, with the SPL at 0.75. Among those unsuccessful cases were 2 failed trials due to voice recognition error, 3 failed trials where the robot stopped and got stuck in a narrow space during navigation, and 2 failed trials due to incorrect object detection. With SPL around 0.75 in both of our scenarios, it indicates our algorithm is an effective navigation strategy, that not only prioritizes completing the task but also does so with consideration for path efficiency.

Overall, our algorithm has demonstrated satisfactory performances in two indoor environments. The success rate exceeds 90% in both environments. In the existing work [3], [4], [22]–[26], transformer-based VLN methods were developed and tested using Room-to-Room datasets, and they achieved SR up to 76% in robot navigation tasks based on voice command input. Our method obtained a better success rate compared to the existing work.

V. Real Robot Experiments

We conducted real robot experiments in a laboratory environment. The space has standard lab equipment: laptop, TV monitor, keyboard, mouse, cell phone, books, table,



(a) t=0s (b) t=14s (c) t=22s

Fig. 7: Snapshots of real robot experiments in a laboratory environment. The navigation map is shown on the top right, and the current camera view is shown on the bottom right of each sub-figure. (a) The robot receives voice command of the goal object "Backpack" at t=0s; (b) The robot is navigating to the goal at t=14s; (c) The robot reaches the goal object at t=22s.

chairs, refrigerator, microwave oven, bottle, cup, and apple. The dimensions of the test area are 6.68m by 7.24m.

we used a Pioneer-3DX robot with a max 0.35 m/s linear velocity and max 1.0 rad/s angular velocity. The robot is equipped with an odometry sensor and an onboard Realsense D435i RGB-D camera. The RGB-D camera was used for both SLAM and object detection. A laptop is placed on the robot with ROS Noetic 20.04 installed on the Ubuntu platform. The laptop is equipped with an Intel 13700k CPU and an NVIDIA 4080 GPU.

In real-world experiments, we initialized the robot's starting position for the localization accuracy. The robot starts to receive the command in 8 seconds after it asks: "What can I do for you?" Then, the robot leads the user to navigate to the desired object. Due to the space limitations of the testing area and the field of view of the camera, some objects cannot be reached at the exact location (e.g., the keyboard is on a box), we consider the robot reached its destination when it is within 0.8m of the target object. We performed the experiments 50 times, each time randomly choosing one of the objects mentioned above as the desired object and asking the robot to find and navigate to it.

Fig. 7 shows the robot navigation process from our real robot experiments conducted in a lab setting. At the beginning (t=0s), the robot starts receiving a voice command to find a "Backpack," showing both the navigation map and the robot's camera view. At t=14s, the robot is on its way, navigating based on updates from the navigation map and its camera view. At t=22s, the robot successfully arrives at the goal with both the final location visible on the navigation map and the robot's camera view confirming the presence of the goal object - the backpack. Videos of the experiments are included in the multimedia file submitted with the paper.

We use the same performance metrics as defined in the simulation section. In real robot testing, the robot has motion delays due to the computational load onboard of the robot and uncertainties in sensing. In addition to the performance metrics used in the simulations, we add the **Responding Time (RT)**, which is defined as the time it takes the robot from receiving the command to starting the navigation.

During the 50 times laboratory tests, the robot completed the navigation task 41 times, resulting in a success rate of 82%. Failures cases include three instances due to localization and navigation issues: two localization failures, one collision, and two navigation planning failures. Three times failures were due to a language recognition error that led to a navigation error to an incorrect goal.

TABLE II: Real robot experimental results.

Environment	Mean NE (SD)	SR	SPL	Mean RT (SD)
Real-Robot	0.21m (0.02m)	82%	0.66	5.54s (2.21)

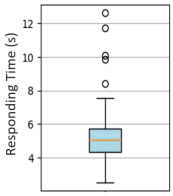


Fig. 8: The boxplot of the responding time (RT).

Table II presents the quantified testing results. We can see that the mean value of NE is 0.21m, the SR is 82%, the SPL is 0.66, and the mean value of the RT is 5.54s. We can see that the performances downgraded compared to the simulation results. This is reasonable, as the real robot experiments are subjected to real-world environmental complexity, sensor noises, and uncertainties, which lead to less optimal performance than in simulations. Similar conclusions were drawn in [4], where they reported a 50% success rate (10 out of 20 tasks completed) in real-robot lab experiments.

Fig. 8 shows the box plot of the RT. We can see that the average RT for is 5.54s, the median is 5.05s, and the 75% quartile value is 5.75s. These results have demonstrated that our algorithm can be applied to real-world environments with satisfactory performances.

VI. Conclusion and Future Work

We present a multi-modal robotic navigation system that integrates natural language processing and semantic map based navigation. This innovative approach allows our robot to autonomously generate and execute navigation tasks based on inputs in human natural language, facilitating human-robot interaction. We demonstrate the performance of our

system through validation in both simulated and real-world environments, where satisfactory performances have been observed. Our method outperforms existing VLN methods, and achieves significant improvement in success rates in simulations and real robot experiments under real-world conditions. In the future, we plan to expand our robotic system's capabilities, such as adding robotic arms for assisting humans in a wide range of daily activities.

APPENDIX

TABLE III: Parameters for PointNet and YOLO-v3.

Parameter	PointNet	YOLO-v3
Conv. Layers	5 (Shared) MLP	75 layers
Pooling Strategy	Max Pooling (Global)	Max Pooling (Local)
Batch Size	32	64
Learning Rate	1×10^{-3}	1×10^{-4}
Activation Function	ReLU	Leaky ReLU

REFERENCES

[1] K. Yamazaki, R. Ueda, S. Nozawa, M. Kojima, K. Okada, K. Matsumoto, M. Ishikawa, I. Shimoyama, and M. Inaba, "Home-assistant robot for an aging society," *Proceedings of the IEEE*, vol. 100, no. 8, pp. 2429–2441, 2012.

[2] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. Van Den Hengel, "Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3674–3683, 2018.

[3] X. Wang, Q. Huang, A. Celikyilmaz, J. Gao, D. Shen, Y.-F. Wang, W. Y. Wang, and L. Zhang, "Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6629–6638, 2019.

[4] C. Huang, O. Mees, A. Zeng, and W. Burgard, "Visual language maps for robot navigation," in *IEEE International Conference on Robotics and Automation*, pp. 10608–10615, 2023.

[5] S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-Maron, M. Gimenez, Y. Sulsky, J. Kay, J. T. Springenberg, *et al.*, "A generalist agent," *arXiv:2205.06175*, 2022.

[6] M. Shridhar, L. Manuelli, and D. Fox, "Perceiver-actor: A multi-task transformer for robotic manipulation," in *Conference on Robot Learning*, pp. 785–799, PMLR, 2023.

[7] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly, and A. J. Davison, "SLAM++: Simultaneous localisation and mapping at the level of objects," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1352–1359, 2013.

[8] J. McCormac, A. Handa, A. Davison, and S. Leutenegger, "Semanticfusion: Dense 3D semantic mapping with convolutional neural networks," in *IEEE International Conference on Robotics and Automation*, pp. 4628–4635, 2017.

[9] M. Runz, M. Buffier, and L. Agapito, "Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects," in *IEEE International Symposium on Mixed and Augmented Reality*, pp. 10–20, 2018.

[10] B. Xu, W. Li, D. Tzoumanikas, M. Bloesch, A. Davison, and S. Leutenegger, "Mid-fusion: Octree-based object-level multi-instance dynamic SLAM," in *IEEE International Conference on Robotics and Automation*, pp. 5231–5237, 2019.

[11] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," *arXiv:1804.02767*, 2018.

[12] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *Computer Vision–ECCV: 13th European Conference*, pp. 740–755, 2014.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *arXiv:1810.04805*, 2018.

[15] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 5485–5551, 2020.

[16] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020.

[17] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv:1408.5882*, 2014.

[18] J. Duan, S. Yu, H. L. Tan, H. Zhu, and C. Tan, "A survey of embodied ai: From simulators to research tasks," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 6, no. 2, pp. 230–244, 2022.

[19] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang, "MatterPort3D: Learning from rgb-d data in indoor environments," *arXiv:1709.06158*, 2017.

[20] M. Savva, A. Kadian, O. Maksymets, *et al.*, "Habitat: A platform for embodied ai research," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9339–9347, 2019.

[21] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese, "Gibson Env: Real-world perception for embodied agents," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9068–9079, 2018.

[22] G. Georgakis, K. Schmeckpeper, K. Wanchoo, *et al.*, "Cross-modal map learning for vision and language navigation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15460–15470, 2022.

[23] D. Fried, R. Hu, V. Cirik, *et al.*, "Speaker-follower models for vision-and-language navigation," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[24] S. Chen, P.-L. Guhur, C. Schmid, and I. Laptev, "History aware multimodal transformer for vision-and-language navigation," *Advances in Neural Information Processing Systems*, vol. 34, pp. 5834–5847, 2021.

[25] Y. Hong, Q. Wu, Y. Qi, C. Rodriguez-Opazo, and S. Gould, "VLN BERT: A recurrent vision-and-language BERT for navigation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1643–1653, 2021.

[26] P. Anderson, A. Shrivastava, J. Truong, A. Majumdar, D. Parikh, and D. Batra, "Sim-to-real transfer for vision-and-language navigation," in *Conference on Robot Learning*, pp. 671–681, PMLR, 2021.

[27] J. Krantz, E. Wijmans, A. Majumdar, D. Batra, and S. Lee, "Beyond the nav-graph: Vision-and-language navigation in continuous environments," in *Computer Vision–ECCV: 16th European Conference*, pp. 104–120, 2020.

[28] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, *et al.*, "ROS: an open-source Robot Operating System," in *ICRA Workshop on open source software*, vol. 3, p. 5, Kobe, Japan, 2009.

[29] M. Labbé and F. Michaud, "RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation," *Journal of Field Robotics*, vol. 36, no. 2, pp. 416–446, 2019.

[30] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 652–660, 2017.

[31] I. Armeni, S. Sax, A. R. Zamir, and S. Savarese, "Joint 2D-3D-semantic data for indoor scene understanding," *arXiv:1702.01105*, 2017.

[32] J. Xiao, K. A. Ehinger, J. Hays, A. Torralba, and A. Oliva, "SUN database: Exploring a large collection of scene categories," *International Journal of Computer Vision*, vol. 119, pp. 3–22, 2016.

[33] E. Marder-Eppstein, "Move_base: The ROS navigation stack's path planning and navigation package." `http://wiki.ros.org/move_base`, 2016.

[34] "AWS RoboMaker World." `https://github.com/aws-robotics`.

[35] P. Anderson, A. Chang, D. S. Chaplot, *et al.*, "On evaluation of embodied navigation agents," *arXiv:1807.06757*, 2018.