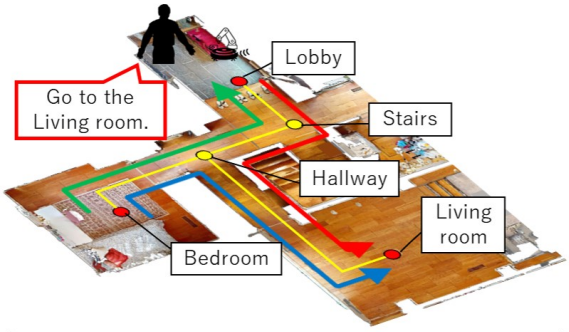
2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Language to Map: Topological map generation from natural language path instructions

Hideki Deguchi[1], Kazuki Shibata[1] and Shun Taguchi[1]

***Abstract*— In this paper, a method for generating a map from path information described using natural language (textual path) is proposed. In recent years, robotics research mainly focus on vision-and-language navigation (VLN), a navigation task based on images and textual paths. Although VLN is expected to facilitate user instructions to robots, its current implementation requires users to explain the details of the path for each navigation session, which results in high explanation costs for users. To solve this problem, we proposed a method that creates a map as a topological map from a textual path and automatically creates a new path using this map. We believe that large language models (LLMs) can be used to understand textual path. Therefore, we propose and evaluate two methods, one for storing implicit maps in LLMs, and the other for generating explicit maps using LLMs. The implicit map is in the LLM's memory. It is created using prompts. In the explicit map, a topological map composed of nodes and edges is constructed and the actions at each node are stored. This makes it possible to estimate the path and actions at waypoints on an undescribed path, if enough information is available. Experimental results on path instructions generated in a real environment demonstrate that generating explicit maps achieves significantly higher accuracy than storing implicit maps in the LLMs.**



Path1: Depart from the Bedroom to the Hallway, proceed to the Stairs and turn left. Then, proceed to the Lobby.

Path2: Depart from the Bedroom to the Hallway, turn right and proceed to the Living room.

Fig. 1. Image representation of task in this study. The above figure shows the map and the paths. The sentences in blue font underneath the figure describe the green and blue paths in the above figure. The task in this study was to input the detailed path as a bottom sentence and output the new path from the destination name.

I. INTRODUCTION

In recent years, owing to the growing use of robots by people in their living environments, research on vision-and-language navigation (VLN) has accelerated [1], [2]. VLN is a navigation system that uses a camera image and natural language instructions to reach a set destination [3]–[6]. The use of natural language for explanations is expected to make it easier for users to provide instructions to robots.

One of the problems with VLN is the high cost of instruction to explain the path detail. To illustrate, instructions, such as "Go straight out of the bedroom, turn right at the second corner and go down the stairs on the left...", must be provided for each navigation. From the user's perspective, the ideal situation is to take to the destination simply by specifying its name. While there are methods to search space by destination name [7]–[9], it would be more efficient to move to a destination if a path could be generated.

Therefore, a solution to this problem was proposed in this paper. The proposed solution is to create a map as a topological map from textual path instructions. Then, users get path from created map only they instruct the destination name. One way to create a map from natural language path instructions is to use large language models (LLMs) [10]–[12] with prompt [13]. However, although LLMs can facilitate the creation of implicit map through the use of an appropriate prompt, they have difficulties in processing long-term memory [14], which may prevent them from recognizing the spatial structure and creating a correct map from the textual path.

Then, we considered using LLMs to create an explicit map from textual maps for path description. The proposed explicit map was similar to a topological map [15]. Our explicit map feature contained actions extracted from the textual path in each node. The stored actions could be used to output a path in the reverse direction, generate a new path, and so on. This frees the user from having to explain previously described paths or paths that can be inferred from those paths. To the best of our knowledge, the proposed method is the first to map a space using solely natural language descriptions.

Figure 1 shows an example task from this study. The sentences in the figure show the textual paths of the blue and green arrows in the environment. The goal of this study was to create a map that, given the sentence in Figure 1, can output the path indicated by the red arrow above Figure 1 from the name of the destination in natural language.

The contributions of this study are as follows:

- Proposal of a method for memorizing textual path instructions and creating new ones that are not described.
- Proposal of a new map style that can be created from only natural language.

[1]They are work for Toyota central R&D Labs., Inc., Yokomichi41-1, Nagakute, Aichi, Japan.

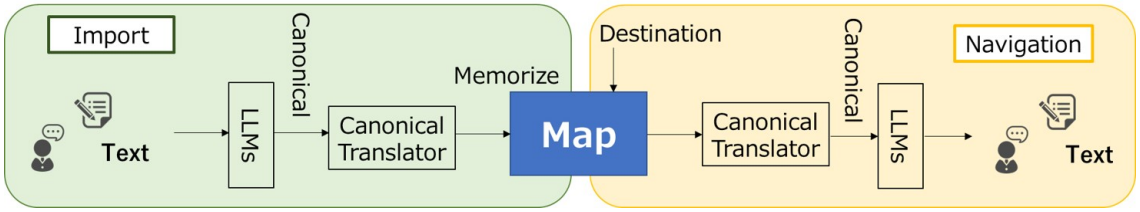


Fig. 2. System overview. The left side shows the system that generates a map from user's instruction. The right side shows the system that outputs the new path from destination name.

- Proposal of an algorithm for acquiring intermediate representations to convert natural languages into maps using LLMs.
- Confirmation by experiment that LLMs have poor spatial comprehension ability and that this ability can be improved by using explicit maps.

II. RELATED WORK

In recent years, there have been remarkable advances in LLMs that deal with natural language information [10], [11]. In particular, GPTs [16], [17], such as ChatGPT [12], are being used in many areas, and research into their capabilities is ongoing [18]. The use of LLMs is also advancing in the field of robotics, where they are being used for a variety of tasks [19]–[22].

In the field of planning, there are studies that use natural language information to predict the scene of a destination or to determine the next action. Li et al. [4] proposed a method for predicting the direction and scenery of a specified goal based on textual path instructions and 360-degree camera images using the results of learning various environmental data in advance. Zehao et al. [23] proposed a VLN method for path planning using geometric maps. Their method entailed comparing landmarks and actions in linguistic instructions and features of the geometric map to derive an appropriate path to the destination. Zhou et al. [24] proposed a navigation method that used LLMs with a camera image to estimate the next action of a robot on its trajectory. Shah et al. [25] proposed a method for planning a robot's path based on a topological map constructed from images and landmarks extracted from the textual path using LLMs.

In the field of mapping, natural language information processing by LLMs has been treated as auxiliary information for geometric mapping. For example, in their study on VLN, Georagakis et al. [26] improved the success rate of navigation by predicting the blind spots in a camera image using path descriptions, in contrast to the conventional method of creating semantic maps from camera images. Peihao et al. [27] proposed a method for describing object information on semantic maps using the object information on paths in a textual path, thus enabling detailed object classification (e.g., color), beyond what was possible using existing segmentation models. Chen et al. [28] proposed a method that used language path instructions to generate topological maps in unknown environments. This method is similar to our proposed method, but Chen et al. differ significantly from our method in that they create a topological map from image information and then use linguistic information as supplementary information.

Thus, many scholars use LLMs for planning and mapping. Most of them use pre-created geometric maps or camera images. In contrast, our method does not need to use them and creates a map using only natural language information.

III. METHOD

A. System Overview

To reduce the burden of path descriptions in VLN, we proposed a system that stores past path descriptions and automatically generates path descriptions when a new destination is obtained. In our system, multiple path descriptions are initially provided to the system. The names of the starting points and destination are entered as queries at runtime. The system generates a map from the first input path descriptions, searches for a path on the internal map when a query is provided, converts the obtained path into a natural language description, and outputs the map.

In this study, we propose two methods; the first uses implicit maps and the other uses explicit maps. In the first method, all these inputs are given to the LLMs, and the map is stored in the LLMs. When a query is provided, LLMs generate a path based on an internally stored map. By contrast, in the method using explicit maps, the input is converted into a canonical representation by the LLMs. The map is then generated from the canonical information. When a query is given, pathfinding and action estimation are performed on the map, which is then converted into a natural language and output. The next section describes the use of the explicit map.

B. Method using Explicit Map

Since it is not certain whether LLMs can store implicit maps, this paper also proposes a method of having explicit maps, which will be compared and evaluated. We show the overview of the proposed method using explicit map in Figure 2.

1) Map Construction: In this method, we construct topological map as a graph $G(N, E)$, consists of nodes $n \in N$ and edges $e \in E$. We also store actions at each nodes

$a_{n_i}(e_{ji}, e_{ik})$ defined as follows:

$$a_{n_i}(e_{ji}, e_{ik}) = \begin{cases} a_{\mathrm{F}} & \text{if} \quad -\theta & \leq \angle(e_{ji}, e_{ik}) & \leq \theta, \\ a_{\mathrm{L}} & \text{if} \quad \theta & < \angle(e_{ji}, e_{ik}) & < \pi - \theta, \\ a_{\mathrm{R}} & \text{if} \quad \pi + \theta & < \angle(e_{ji}, e_{ik}) & < -\theta, \\ a_{\mathrm{T}} & \text{if} \quad \pi - \theta & \leq \angle(e_{ji}, e_{ik}) & \leq \pi + \theta, \end{cases} \tag{1}$$

where $a_{n_i}(e_{ji}, e_{ik})$ denotes the action at the node n_i through from the edge e_{ji} to e_{ik}, $a_{\mathrm{F}}, a_{\mathrm{L}}, a_{\mathrm{R}}, a_{\mathrm{T}}$ represents the actions "Forward", "Turn Left", "Turn Right", "Turn Around", respectively. As indicated by the above definition, each action defined in the language was divided into four parts based on the range of angles between the edges defined in the Euclidean space. The θ is threshold of the angle. The action of each node, which can be described in any linguistic expression, was approximated and assigned to one of four actions.

The proposed system uses canonical representation that can be translated from both language instructions and map when it translates using LLMs. This is because it is known to be difficult to translate language instructions into formal information using LLMs, unlike vice versa [29]. In this study, canonical representation is a set of waypoints on the path W and the actions at those waypoints. For example of path 1 in Figure 1, the waypoints are

$$W = [\mathrm{Bedroom}, \mathrm{Hallway}, \mathrm{Stairs}, \mathrm{Lobby}], \tag{2}$$

and the actions are

$$a_{\mathrm{Hallway}}(e_{\mathrm{Bedroom,Hallway}}, e_{\mathrm{Hallway,Stairs}}) = a_{\mathrm{F}}, \tag{3}$$
$$a_{\mathrm{Stairs}}(e_{\mathrm{Hallway,Stairs}}, e_{\mathrm{Staris,Lobby}}) = a_{\mathrm{L}}, \tag{4}$$

in canonical representation. To construct a topological map, we added each waypoint as a node and each edge between waypoints as an edge and stored the actions at those waypoints.

2) Path Finding and Instruction Generation: To generate path instruction, we have to find the path between the start point and the destination of the query, and estimates action at each waypoints.

The path can be found on the topological map. One of the methods that can be used for finding the path is the use of the Dijkstra search algorithm. However, it is necessary to estimate the action at each node from stored actions because they cannot be obtained by path finding on the topological map. The target action, if present in the stored data, was used. The target action, if not defined, was inferred from multiple actions, as follows:

$$a_{n_i}(e_{ji}, e_{ik}) = a_{n_i}(e_{ji}, e_{im}) a_{\mathrm{T}} a_{n_i}(e_{mi}, e_{ik}). \tag{5}$$

This is a transformation that uses the fact that the estimated action at node n_i when moving from node, n_j, through n_i to n_k is equivalent to the action of moving from n_j through n_i to n_m, then turning around and going from n_m through n_i to n_k.

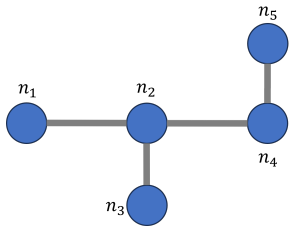


Fig. 3. The environment is shown in the figure below, where n_1 n_5 is the name of each node.

The following theorem, derived from (1) below, can be used to estimate the action.

$$a_{n_i}(e_{ji}, e_{ik})^{-1} = a_{n_i}(e_{ki}, e_{ij}), \tag{6}$$
$$a_{\mathrm{F}} a_{n_i}(e_{ji}, e_{ik}) = a_{n_i}(e_{ji}, e_{ik}), \tag{7}$$
$$a_{n_i}(e_{ji}, e_{ik}) a_{\mathrm{F}} = a_{n_i}(e_{ji}, e_{ik}), \tag{8}$$
$$a_{\mathrm{L}}^{-1} = a_{\mathrm{R}}, \tag{9}$$
$$a_{\mathrm{F}}^{-1} = a_{\mathrm{F}}, \tag{10}$$
$$a_{\mathrm{T}}^{-1} = a_{\mathrm{T}}, \tag{11}$$
$$a_{\mathrm{T}} a_{\mathrm{L}} = a_{\mathrm{R}}, \tag{12}$$
$$a_{\mathrm{T}} a_{\mathrm{R}} = a_{\mathrm{L}}, \tag{13}$$
$$a_{\mathrm{L}} a_{\mathrm{R}} = a_{\mathrm{F}}, \tag{14}$$
$$a_{\mathrm{L}} a_{\mathrm{L}} = a_{\mathrm{T}}, \tag{15}$$
$$a_{\mathrm{R}} a_{\mathrm{R}} = a_{\mathrm{T}}. \tag{16}$$

For clarity, we used the environment shown in Figure 3. When user instructs the path of n_1 to n_3 and n_1 to n_5, our method memorizes these two path as below:

$$W_1 = [n_1, n_2, n_3], \tag{17}$$
$$W_2 = [n_1, n_2, n_4, n_5]. \tag{18}$$

Subsequently, the actions were also memorized, as follows:

$$a_{n_2}(e_{12}, e_{23}) = a_{\mathrm{R}}, \tag{19}$$
$$a_{n_2}(e_{12}, e_{24}) = a_{\mathrm{F}}, \tag{20}$$
$$a_{n_4}(e_{24}, e_{45}) = a_{\mathrm{L}}, \tag{21}$$

here, the path from n_5 to n_3 is $[n_5, n_4, n_2, n_3]$ can be estimated using path finding method. The action of node n_4, and n_2 can be estimated as the following procedures:

$$\begin{aligned} a_{n_4}(e_{54}, e_{42}) &= a_{n_4}(e_{24}, e_{45})^{-1} \\ &= a_{\mathrm{L}}^{-1} \\ &= a_{\mathrm{R}}, \end{aligned} \tag{22}$$

$$\begin{aligned} a_{n_2}(e_{42}, e_{23}) &= a_{n_2}(e_{42}, e_{12}) a_{\mathrm{T}} a_{n_2}(e_{12}, e_{23}) \\ &= a_{n_2}(e_{12}, e_{24})^{-1} a_{\mathrm{T}} a_{n_2}(e_{12}, e_{23}) \\ &= a_{\mathrm{F}} a_{\mathrm{T}} a_{\mathrm{R}} \\ &= a_{\mathrm{L}}. \end{aligned} \tag{23}$$

Finally we obtained, the following canonical representation:

$$W = [n_5, n_4, n_2, n_3], \tag{24}$$
$$a_{n_4}(e_{54}, e_{42}) = a_{\mathrm{R}}, \tag{25}$$
$$a_{n_2}(e_{42}, e_{23}) = a_{\mathrm{L}}. \tag{26}$$

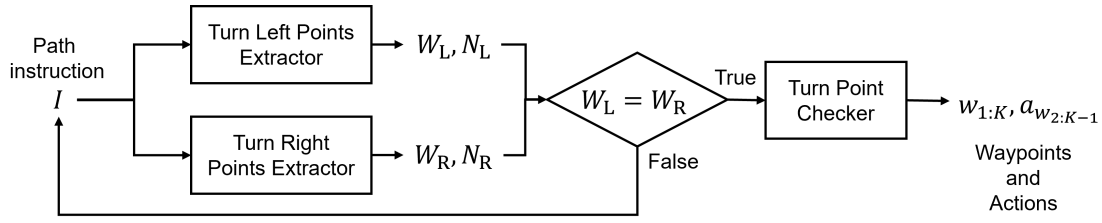


Fig. 4. Flowchart of LLMs' prompt in this study.

To output the path instruction, convert the canonical representations to natural language using LLMs. Finally, we obtain the following path instruction: "Depart from n_5 to n_4. Then, turn right and proceed to n_2. Then, turn left and proceed to n_3."

3) Translation to Canonical Representation using LLMs: As explained above, the proposed method entailed translating natural language path instructions to canonical representations using LLMs. The extraction procedure for the canonical information is shown in Figure 4.

First, path instructions input to LLMs using two different prompts. Their extract the whole nodes in the path and left and right turn points respectively. Our method is splitted into two parts, rather than prompting the LLM to output the left-turn points and right turn points simultaneously. It makes possible that reduces the difficulty of the task required for the LLM and increases the success rate. Moreover, it also makes it possible to check for errors by comparing the output node sequences, thus improving the robustness of the system. Specifically, if the output waypoints from each extractor are not equal, the process is repeated again.

Thenafter, the action of each waypoint is estimated. If the waypoint is donoted as a turnpoint by one of the extractors, assign an action of a_{L} or a_{R} according to the extractor. If neither, an action of a_{F} is assigned. If the waypoint is denoted as a turnpoint by both extractors, the direction of rotation is determined by checking again with the LLM to determine the direction of rotation.

Complete algorithm is shown in Algorithm 1, where TurnLeftPointExtractor, TurnRightPointExtractor are the modules to extract waypoints and left turn (right turn) points. TurnPointChecker is a module to check turn direction at point w_k. C is the maximum repetitive number or extractions.

Algorithm 1 Translation to Canonical Representation

Input: path instruction I
Output: waypoints $w_{1:K}$, actions $a_{w_{2:K-1}}$
$W_{\mathrm{L}}, N_{\mathrm{L}} \leftarrow \mathrm{TurnLeftPointsExtractor}(I)$
$W_{\mathrm{R}}, N_{\mathrm{R}} \leftarrow \mathrm{TurnRightPointsExtractor}(I)$
$c = 0$
while $W_{\mathrm{L}} \neq W_{\mathrm{R}} \cap c < C$ **do**
  $W_{\mathrm{L}}, N_{\mathrm{L}} \leftarrow \mathrm{TurnLeftPointsExtractor}(I)$
  $W_{\mathrm{R}}, N_{\mathrm{R}} \leftarrow \mathrm{TurnRightPointsExtractor}(I)$
  $c \leftarrow c + 1$
end while
$w_{1:K} \leftarrow W_{\mathrm{R}}$
for $w_k \in \{w_k | w_k \in w_{2:K-1}\}$ **do**
  if $w_k \in N_L \cap w_k \in N_R$ **then**
    $a_{w_k} \leftarrow \mathrm{TurnPointChecker}(I, w_k)$
  else if $w_k \in N_{\mathrm{L}}$ **then**
    $a_{w_k} \leftarrow a_{\mathrm{L}}$
  else if $w_k \in N_{\mathrm{R}}$ **then**
    $a_{w_k} \leftarrow a_{\mathrm{R}}$
  else
    $a_{w_k} \leftarrow a_{\mathrm{F}}$
  end if
end for
return $w_{1:K}, a_{w_{2:K-1}}$

IV. EXPERIMENT

A. Dataset

In this study, the effectiveness of implicit and explicit map storage methods for inferring spatial structure from path instructions using LLMs was evaluated. Graph maps were manually created for models in the Matterport3D dataset [30] and used as the evaluation environment. The path instructions were generated manually for these environments. Figure 5 shows an example of the environment used. We selected 10 environments and five nodes as the start or destination nodes in each environment. We then created 10 textual paths by selecting two pairs from these five nodes. In this study, we assumed the following for the user path description:

- The user has a name at each node of the topological map in mind.
- The paths are created such that they have the minimum nodes from start to destination.
- The user doesn't describe landmarks not related to the action.
- The user describe all the landmarks which related to the action in their path description.
- The user doesn't miss their instructions.

B. Evaluation

In this experiment, the following two evaluations were conducted:

Reverse path
: Create the reverse path of the input path.

Combined path
: 9 paths were selected and input from the 10 paths prepared for each environment. One excluded path was generated from the information on the start and goal names of the paths.

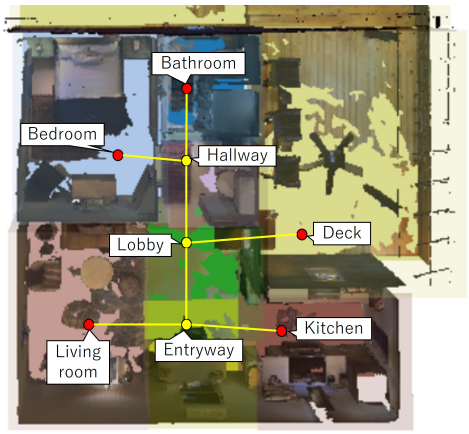


Fig. 5. Example of the map with the path generation. The red circles describe the start or destination nodes and the yellow circles describe the waypoint nodes. The yellow lines show the edge.

TABLE I

PROMPTS USED IN EXPERIMENT.

Types of Prompt		Prompt
Implicit method	Reverse path	Show the reverse path, reversing the start and goal of the following path.
	Combined path	Understand the spatial structure of path1-9 below and create the shortest path from the specified start to goal. However, be sure to indicate the action to be taken at each passing point.
Explicit method	Turn points extractor	Extract waypoints in the description of the navigation path. Then, extract the points which turn left/right.
	Turn points checker	For the following path, answer the action at specified point is turn right or left.

The reverse path was evaluated to assess the system's ability to grasp the spatial structure from the path input. The combined path was evaluated to assess the ability of the system to generate a new path from multiple input paths. When evaluating the combined path, we use two evaluation metrics, reachable path and shortest path. Reachable path allows the path not to be the shortest. Note that the shortest route here means the route with the smallest number of edges on the topological map. Perform the experiment with the 100 paths described in Section IV-A and derive the success rate.

C. Setting

Table I shows the prompts which use our method in the experiment. Implicit method for reverse path and combined path in Table I are simple prompts to evaluation. Turn points extractor and Turn points checker in Table I are the prompts at explicit map method. The turn points extractor is purposely implemented in two steps: the extraction of all waypoints and the extraction of left/right turn points. In this study, we use OpenAI API [31], [32] and the GPT function calling feature is used in the implementation.

D. Results

Tables II and III present the results of the representation reverse path and combined path using LLMs evaluation.

TABLE II

SUCCESS RATE OF REVERSE PATHS EVALUATION.

Method	LLM	Success rate
Implicit method	GPT-3.5-turbo	67%
	GPT-4	66%
Explicit method	GPT-3.5-turbo	83%
	GPT-4	**94%**

TABLE III

SUCCESS RATE OF COMBINED PATHS EVALUATION.

Method	LLM	Success rate	
		Reachable path	Shortest path
Implicit method	GPT-3.5-turbo	19%	8%
	GPT-4	44%	10%
Explicit method	GPT-3.5-turbo	77%	77%
	GPT-4	**92%**	**92%**

Implicit/Explicit method in tables indicates the method that uses an implicit/explicit map.

The result of Table II shows a 60% success rate for the simple task of deriving a path in the opposite direction of the described path. However, the success rate of the proposed method using explicit maps and GPT-4 [32] was $> 90\%$ for LLMs. Therefore, the proposed method with explicit map almost solved the inverse generation problem.

Based on the result of Table III, the proposed method using an implicit map could not create new paths from textual paths. However, the proposed method using an explicit map effectively clarified the spatial structure of paths described in natural language and generated new paths. The failed cases were errors in extracting waypoints and actions from natural language (Turn left/right extractor modules in fig.4).

An example of the input/output using the environment and path sentences shown in Figure 5 is shown in Figure 6. Verifying the results of Figure 6 against Figure 5 reveals that the proposed method using an implicit map output the wrong direction for the entry way turn, whereas the proposed method using an explicit map output the correct path to the destination. The Figure 7 shows the results of a qualitative evaluation conducted using LLMs [32] with various representations of textual paths as input. The result of Figure 7 indicates the explicit method can create the map and the new path from various representations.

V. DISCUSSION

The results in Table III show that the existing LLMs failed approximately 40%. One of the most common causes of LLMs failure is incorrect rotational position or direction in the path(96%). This suggests that LLMs lack the ability to grasp the spatial structures of language paths. This becomes clearer when we consider the evaluation results of the combined path. In the combined path evaluation, methods using only LLMs were almost unable to show the shortest path, while the method using only GPT-4 showed a success rate of about 40% for the reachable path. Therefore, it can be seen that LLMs have the ability to use the given path sentences as they are but do not have the ability to recombine them to generate appropriate paths. By contrast, the proposed method

Start point: “Living room”
Goal point: “Bathroom”

Path 1 Depart from the ”Living room” to the ”Entryway”, and proceed to the ”Kitchen”.

Path 2 Depart from the ”Kitchen” to the ”Entryway”, turn right and proceed to the ”Lobby”. Continue on to the ”Hallway” and proceed to the ”Bathroom”.

Implicit method

From the ”Living Room”, proceed to the ”Entryway” (Path 1).
From the ”Entryway”, turn right and head to the ”Lobby” (Path 2).
From the ”Lobby”, continue to the ”Hallway” (Path 2).
From the ”Hallway”, proceed to the ”Bathroom” (Path 2).

Explicit method

Depart from the living room and advance to the entryway, then turn left. From there, advance to the lobby and continue on to the hallway, then advance further to the bathroom where you will arrive.

Fig. 6. An example of the textual path in Figure 5 used for the quantitative evaluation. The green sentences show the textual paths, the blue sentence shows the output of the implicit map method and the red sentence shows the output of the explicit map method. This figure shows only the paths used for output from the 9 input paths. The notation “(Path 1)” or “(Path 2)” is itself contained in the raw output of the implicit method. These phrases indicate which textual paths were used to create the new path.

Start point: “Deck”
Goal point: “Kitchen”

Path 1 Commence on the “Deck”, walk to the ”Lobby”, swing left there and towards the ”Entryway”. Then, guide yourself right there and go to the “Living room”.

Path 2 Begin in the “Living room”, move through the ”Entryway”, and then head straight to the ”Kitchen”.

Implicit method

1. Start at the ”Deck”, go to the ”Lobby” (Path 1).
2. From the ”Lobby”, go to the ”Entryway” (Path 1).
3. From the ”Entryway”, go straight to the ”Kitchen” (Path 2).

Explicit method

Depart from the deck and advance to the lobby, then turn left. From the lobby, continue on to the entryway, again turning left. From there, advance to the kitchen where you’ll arrive.

Fig. 7. Paths converted with LLMs [32] to more diverse wording than the input path of Figure 5. The explicit method can output path descriptions more accurately than the implicit method, even if the expression of the instructions changes.

using an explicit map had a high success rate based on both evaluations and fully compensated for the weakness of LLMs in understanding the spatial structure. The method that uses an explicit map can always output the correct path if the intermediate representation is accurate. Hence, the accuracy can be improved by making the prompt to extract canonical information from the textual path more accurate.

The main limitation of this study is the assumption regarding the inputs. First, the proposed method does not deal with the diversity of instruction style. For example, the proposed method requires to describe path information from one starting point to another. However, users may describe the locations of multiple points from a single starting point in one time. In this case, we need to make a system that decomposes the instruction into multiple paths and it is future work.

Second, this study assumes that users do not miss instructions. However, user instructions sometimes contain errors, against which the proposed method must prove robust when used in a real environment. To address this issue, we plan to introduce a system that detects discrepancies in explanations from the generated explicit maps and asks the user whether the instructions are correct.

Finally, we evaluated a small environment, similar in size to a house, because this was sufficient to compare the implicit and explicit map methods. However, there are many large environments, such as shopping malls, which our explicit map method must address. In future, we will evaluate how large a space can be mapped using our proposed method.

VI. CONCLUSION

In this study, we focused on the task of creating a map from a path described in a natural language. To address this issue, we proposed two methods, one that implicitly memorize the paths in LLMs and another that explicitly create maps. The implicit map method uses the prompt and stores the spatial structure of the LLMs implicitly. The explicit map method uses a prompt to create a map that explicitly consists of waypoints and actions at each waypoint of the textual path. The experiments were conducted using paths generated on a graph map created in a real environment. The experimental results show that existing LLMs have a success rate of only approximately 65% for tasks that output the reverse of a given path, whereas the method using the map has a success rate of over 90%, even for difficult recombination tasks. In the future, we plan to improve the proposed method so that it can handle ambiguous user instructions and descriptions in various spaces, and extend our method to a path planning that can consider the dynamics of the robot.

REFERENCES

[1] J. Gu, E. Stefani, Q. Wu, J. Thomason, and X. E. Wang, "Vision-and-language navigation: A survey of tasks, methods, and future directions," *arXiv preprint arXiv:2203.12667*, 2022.

[2] M. Shridhar, J. Thomason, D. Gordon, Y. Bisk, W. Han, R. Mottaghi, L. Zettlemoyer, and D. Fox, "Alfred: A benchmark for interpreting grounded instructions for everyday tasks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 740–10 749.

[3] 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 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVIII 16*. Springer, 2020, pp. 104–120.

[4] M. Li, Z. Wang, T. Tuytelaars, and M.-F. Moens, "Layout-aware dreamer for embodied referring expression grounding," *arXiv preprint arXiv:2212.00171*, 2022.

[5] C. Huang, O. Mees, A. Zeng, and W. Burgard, "Visual language maps for robot navigation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 10 608–10 615.

[6] M. Zubair Irshad, N. Chowdhury Mithun, Z. Seymour, H.-P. Chiu, S. Samarasekera, and R. Kumar, "Sasra: Semantically-aware spatio-temporal reasoning agent for vision-and-language navigation in continuous environments," *arXiv e-prints*, pp. arXiv–2108, 2021.

[7] Y. Wu, Y. Wu, G. Gkioxari, and Y. Tian, "Building generalizable agents with a realistic and rich 3d environment," *arXiv preprint arXiv:1801.02209*, 2018.

[8] J. Ye, D. Batra, A. Das, and E. Wijmans, "Auxiliary tasks and exploration enable objectnav," *arXiv preprint arXiv:2104.04112*, 2021.

[9] D. S. Chaplot, D. P. Gandhi, A. Gupta, and R. R. Salakhutdinov, "Object goal navigation using goal-oriented semantic exploration," *Advances in Neural Information Processing Systems*, vol. 33, pp. 4247–4258, 2020.

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

[11] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[12] 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.

[13] J. Gu, Z. Han, S. Chen, A. Beirami, B. He, G. Zhang, R. Liao, Y. Qin, V. Tresp, and P. Torr, "A systematic survey of prompt engineering on vision-language foundation models," *arXiv preprint arXiv:2307.12980*, 2023.

[14] W. Zhong, L. Guo, Q. Gao, and Y. Wang, "Memorybank: Enhancing large language models with long-term memory," *arXiv preprint arXiv:2305.10250*, 2023.

[15] O. Booij, B. Terwijn, Z. Zivkovic, and B. Krose, "Navigation using an appearance based topological map," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, 2007, pp. 3927–3932.

[16] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, *et al.*, "Improving language understanding by generative pre-training," 2018.

[17] A. Radford, J. Wu, D. Amodei, D. Amodei, J. Clark, M. Brundage, and I. Sutskever, "Better language models and their implications," *OpenAI blog*, vol. 1, no. 2, 2019.

[18] Y. Liu, T. Han, S. Ma, J. Zhang, Y. Yang, J. Tian, H. He, A. Li, M. He, Z. Liu, *et al.*, "Summary of chatgpt-related research and perspective towards the future of large language models," *Meta-Radiology*, p. 100017, 2023.

[19] J. Pan, G. Chou, and D. Berenson, "Data-efficient learning of natural language to linear temporal logic translators for robot task specification," *arXiv preprint arXiv:2303.08006*, 2023.

[20] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. J. Ruano, K. Jeffrey, S. Jesmonth, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, M. Yan, and A. Zeng, "Do as i can and not as i say: Grounding language in robotic affordances," in *arXiv preprint arXiv:2204.01691*, 2022.

[21] D. Driess, F. Xia, M. S. M. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, W. Huang, Y. Chebotar, P. Sermanet, D. Duckworth, S. Levine, V. Vanhoucke, K. Hausman, M. Toussaint, K. Greff, A. Zeng, I. Mordatch, and P. Florence, "Palm-e: An embodied multimodal language model," in *arXiv preprint arXiv:2303.03378*, 2023.

[22] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 9493–9500.

[23] Z. Wang, M. Li, M. Wu, M.-F. Moens, and T. Tuytelaars, "Find a way forward: a language-guided semantic map navigator," *arXiv preprint arXiv:2203.03183*, 2022.

[24] G. Zhou, Y. Hong, and Q. Wu, "Navgpt: Explicit reasoning in vision-and-language navigation with large language models," *arXiv preprint arXiv:2305.16986*, 2023.

[25] D. Shah, B. Osiński, S. Levine, *et al.*, "Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action," in *Conference on Robot Learning*. PMLR, 2023, pp. 492–504.

[26] G. Georgakis, K. Schmeckpeper, K. Wanchoo, S. Dan, E. Miltsakaki, D. Roth, and K. Daniilidis, "Cross-modal map learning for vision and language navigation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 15 460–15 470.

[27] P. Chen, D. Ji, K. Lin, R. Zeng, T. Li, M. Tan, and C. Gan, "Weakly-supervised multi-granularity map learning for vision-and-language navigation," *Advances in Neural Information Processing Systems*, vol. 35, pp. 38 149–38 161, 2022.

[28] S. Chen, P.-L. Guhur, M. Tapaswi, C. Schmid, and I. Laptev, "Think global, act local: Dual-scale graph transformer for vision-and-language navigation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 16 537–16 547.

[29] J. Pan, G. Chou, and D. Berenson, "Data-efficient learning of natural language to linear temporal logic translators for robot task specification," *arXiv preprint arXiv:2303.08006*, 2023.

[30] 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 preprint arXiv:1709.06158*, 2017.

[31] OpenAI, "Gpt-3.5-turbo," *(June 13 version)*, 2023. [Online]. Available: https://chatgpt.pro/

[32] ——, "Gpt-4," *(June 13 version)*, 2023. [Online]. Available: https://chatgpt.pro/