Vision-and-Language Navigation in Real World using Foundation Models

Chengguang Xu **Hieu T. Nguyen** **Christopher Amato**

Lawson L.S. Wong
Khoury College of Computer Sciences
Northeastern University, MA 02115
{xu.cheng, nguyen.trungh, c.amato}@northeastern.edu, lsw@ccs.neu.edu

Abstract

When mobile robots become ubiquitous, they occasionally encounter unseen environments. Enhancing mobile robots with the ability to follow language instructions will improve decision-making efficiency in previously unseen scenarios. However, state-of-the-art (SOTA) vision-and-language navigation (VLN) methods are mainly evaluated in simulation, neglecting the complex real world. Directly transferring SOTA navigation policies learned in simulation to the real world is challenging due to the visual domain gap and the absence of prior knowledge about unseen environments. In this work, we propose a novel navigation framework to address the VLN task in the real world, utilizing the powerful foundation models. Specifically, the proposed framework includes four key components: (1) a large language models (LLMs) based instruction parser that converts a language instruction into a sequence of pre-defined macro-action descriptions, (2) an *online* visual-language mapper that builds a spatial and semantic map of the unseen environment using large visual-language models (VLMs), (3) a language indexing-based localizer that grounds each macro-action description to a waypoint location on the map, and (4) a pre-trained DD-PPO-based local controller that predicts the action. Evaluated on an Interbotix LoCoBot WX250 in an unseen lab environment, *without any fine-tuning*, our framework significantly outperforms the SOTA VLN baseline in the real world.

1 Introduction

Humans can efficiently navigate in familiar environments by creating mental maps that include both spatial and visual cues, such as landmarks [10, 12]. For instance, humans can easily plan a route to the coffee machine from anywhere in their houses because they possess not only a spatial but also a semantic understanding of their surroundings [12]. However, in unfamiliar environments, humans often rely on instructions in natural languages to navigate. Therefore, enhancing mobile robots with the ability to follow instructions in natural languages will facilitate decision-making in unseen scenarios, making them more robust and useful in everyday life.

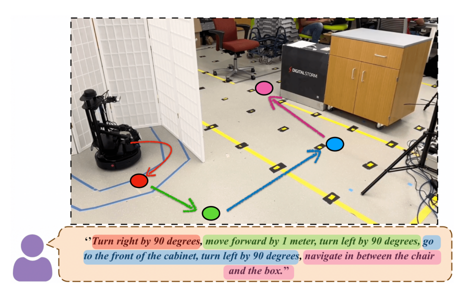

Figure 1: Vision-and-language Navigation in Continuous Environments

Foundation Models for Decision Making Workshop at the 37th Conference on Neural Information Processing Systems (NeurIPS 2023).

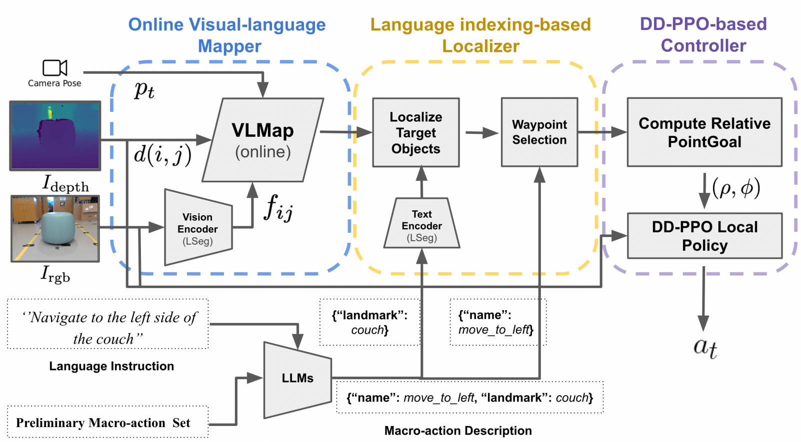

Figure 2: Framework Overview

The vision-and-language navigation (VLN) task [2] aims to benchmark this challenge. Depicted in Figure 1, a mobile robot uses visual inputs (i.e., RGB-D) to navigate in unseen environments by following unstructured natural language instructions. In particular, we consider the VLN task in continuous environments (VLN-CE) [24], where the robot moves continuously in physical space (i.e. SE(2)) by either taking primitive discrete actions [24] or by controlling the linear [25] and angular velocities [19]. Despite significant progress is made in VLN-CE, most recent methods [15, 16, 7, 20, 22, 24] are primarily evaluated in simulation, ignoring the complex real world.

Transferring a VLN-CE agent trained in simulation to the real world is challenging due to the visual domain gap and the absence of prior environment information [1]. To mitigate these challenges, fusing extra sensor information (e.g. laser scan) and employing domain randomization techniques [34] are recommended [1]. However, recent work [18, 33] demonstrates that using foundation models [38], such as LLMs and VLMs, can be beneficial for navigation in the real world. Specifically, LLMs are utilized to parse the instruction into landmarks or executable code, leveraging their powerful textual interpretation capabilities. VLMs are used for processing complex real-world observations and grounding parsed language instructions. Nevertheless, these methods still require prior mapping of the environment, which is not directly applicable to VLN-CE.

In this work, we propose a novel navigation framework to tackle the VLN-CE task in the real world, leveraging the powerful foundation models. Depicted in Figure 2, to ground the unstructured language instructions, we utilize an LLMs-based parser to convert the instruction into a sequence of pre-defined robot macro-action descriptions, which describe the robot's executable movements and associated landmarks. To handle the complex and noisy observations in unseen environments, we build an *online* visual-language map using VLMs. With the latest map and the parsed macro-action descriptions, a language indexing-based localizer grounds each macro-action description to a waypoint location on the map. Treating the waypoint as a point-goal, we adopt an off-the-shelf DD-PPO local policy to predict the next action. We conducted the experiments on an Interbotix LoCoBot WX250 in an unseen lab environment. In the examined instruction following tasks, *without any fine-tuning*, the proposed pipeline significantly outperforms the SOTA VLN-CE baseline in the real world.

2 Related work

Vision-and-language navigation In VLN, two main settings exist, namely discrete environments (VLN-DE) [2] and continuous environments (VLN-CE) [23]. In VLN-DE, due to the short horizon of an episode, the agent can store the visual memory for every step and efficiently reason the visual memory with language instruction using the attention mechanism [8, 28, 9, 17, 31]. In contrast, the long horizon of an episode in VLN-CE makes the metric map a more reasonable choice of visual memory where the observations at different steps can be fused together in the form of a map [7, 15, 20]. [1] first attempts to tackle the vision-and-language navigation in the real world by transferring the policy trained in the simulator to the real world. However, [1] concludes that such transferring will be challenging due to the visual domain gap and the absence of the environment priors. Unlike all these works that require training in the simulator, our approach requires *no training* in simulation and *no fine-tuning* in the real world. Instead, we use foundation models to enable generalization to the real world.

Navigation with online mapping Building maps during navigation achieves impressive performance in multiple embodied navigation tasks such as point-goal navigation, object-goal navigation, and image-goal navigation [5, 21, 6, 36]. However, these methods are designed for particular tasks that require the goal to be specified in a desired format (e.g. a pose, an object class label, or an image). In VLN-CE, only instructions in natural language are provided, requiring our method to implicitly reason about the goal from the instructions. Besides, the map built by our approach is a visual-language map. Unlike traditional occupancy maps and semantic maps, the built visual-language map stores both the spatial occupancy and the language-associated visual features computed from VLMs.

Navigation using foundation models Foundation models [38] have recently been used in navigation tasks. CoWs [14] adapts open-vocabulary models such as CLIP [32] and proposes a language-driven zero-shot object navigation (L-ZSON) task to benchmark object searching. LM-Nav [33] proposes a navigation framework that incorporates three types of foundation models (i.e., large language model (LLM), visual-language model (VLM), and visual navigation model (VNM)) and achieves long-distance navigation in outdoor environments. Our approach also uses off-the-shelf foundation models so that *no fine-tuning* is needed during inference. However, our approach is designed for the VLN-CE task, which is different from the L-ZSON task. Unlike LM-Nav, our method does not require prior data collected from the environment. VLMaps [18] is the most related work. But, unlike VLMaps which requires a pre-collected offline dataset to build the environment map beforehand, our method performs online visual-language mapping during navigation. Moreover, both LM-Nav and VLMaps are designed for multi-goal navigation tasks. In summary, our approach can be considered as an extension of VLMaps to tackle the VLN-CE task in the real world.

3 Problem statement

We consider the vision-and-language navigation task in continuous environment (VLN-CE) [23]. In particular, the continuous setting refers to the scenario where the robot has to take primitive actions to navigate to the desired goal in physical space (i.e., SE(2)) while following an instruction in natural language. This is in contrast to the discrete setting, where the robot selects discrete nodes from a pre-collected navigation graph, as seen in previous work [1, 16, 23].

Formally, at the beginning of each episode, an instruction in natural language $\mathcal{L} = \langle w_0, w_1, w_2, ..., w_L \rangle$ is given, where w_i is the token for a single word in the instruction. The robot also receives an initial front-view observation o_0 determined by the initial pose $s_0 = \langle x_0, y_0, \theta_0 \rangle$, which defines the robot's position and the heading. Following [23], at every time step t, the robot chooses one action a_t from a set of four discrete actions (i.e., *move_forward*, *turn_left*, *turn_right*, *stop*) to execute. Note that, concurrent work like [2] defines the action space as linear and angular velocities, which is different from the current setting. After taking the action a_t, the robot moves to a new pose s_{t+1} and observes a new o_{t+1}. Following the instruction $\mathcal{L}$, the episode terminates when the robot chooses the "stop" action or meets the timeout. The goal is to find a sequence of $\langle s_0, o_0, a_0, s_1, o_1, a_1, ..., s_T, o_t, a_t \rangle$ that aligns with the language instruction $\mathcal{L}$.

4 Method

In this section, we first explain how to convert the instruction into macro-action descriptions using LLMs in Sec. 4.1. Then, the online visual-language mapper using VLMs is explained in Sec. 4.2. Given the latest map and the parsed macro-action descriptions, we explain the language indexing-based localizer in Sec. 4.3. Finally, we explain the off-the-shelf DD-PPO-based local controller in Sec. 4.4.

4.1 LLMs-based instruction parser

We observe that the instruction in the VLN-CE task includes several sub-instructions. For instance, in the Room-to-Room (R2R) task [2], the robot is asked to move from one room to another adjacent room following the instruction. A typical instruction might read as follows "*Exit the bedroom and turn left. Walk straight passing the gray couch and stop near the rug.*". The entire instruction can be parsed into a sequence of sub-instructions such as ⟨*"Exit the bedroom"*, *"Turn left"*, *"Walk straight passing the gray couch"*, *"stop near the rug"* ⟩. Furthermore, we have noticed that each parsed sub-instruction describes either a pure robot movement (e.g., "turn left") or describes both the movement and associated landmarks. For instance, "Walk straight passing the gray couch" contains the movement "walk straight" and the landmark "gray couch". However, these parsed sub-instructions can not be directly executed by the robot.

To address this, we leverage the powerful textual interpretation abilities of LLMs (i.e., GPT 3.5 [29]) to parse and convert the instruction into a sequence of pre-defined robot macro-action descriptions. Specifically, inspired by [18], we define a set of macro-action descriptions serving as prior information about the robot's movements. Formally, we define 10 macro-action descriptions, each represented as a Python dictionary that includes the movement's name and associated parameters. For example, "Walk straight passing the gray couch" corresponds to "{"name": "move_to", "landmark": "gray couch"}". Following a similar approach to [18], we interact with ChatGPT through few-shot prompt engineering and parse each instruction before conducting navigation experiments.

4.2 Online visual-language Mapper

In VLN-CE, collecting data from the target environments is prohibitive because they are assumed to be unseen. Therefore, we extend VLMaps to the online setting and introduce an online mapper that progressively builds the visual-language map of the unseen environment.

In general, the visual-language map fuses the visual-language feature computed from VLMs with a 2-D occupancy grid [18]. These visual-language features enhance the representation of the 2-D occupancy map by incorporating richer semantic features compared to semantic labels. Furthermore, the visual-language map inherently benefits from the powerful generalization abilities of VLMs, which are promising to handle complex real-world observations and diverse language instructions. We adopt LSeg [26, 18], a large visual-language model renowned its dense pixel-wise semantic segmentation driven by flexible language labels. Specifically, LSeg's ViT-based [11] visual encoder aligns the pixel embedding with the text embedding of the corresponding semantic class [26]. Additionally, LSeg's CLIP-based [32] text encoder provides a flexible representation that generalizes well to previously unseen semantic classes during inference. Pretrained on large-scale image-text pairs, LSeg demonstrates significant potential for handling complex robot observations and unseen landmark objects in the real world.

Formally, the visual-language map takes the form of a grid map, denoted as $\mathcal{M} \in \mathbb{R}^{H \times W \times C}$, where H, W are the height and the width of the map, respectively, and C is the dimension of the stored visual-language feature in each grid cell. The resolution of the map is set at $\rho = 5\,\mathrm{cm}$, and each grid cell corresponds to a $25\,\mathrm{cm}^2$ region in the real world. In contrast to [18] that builds the map from an offline dataset, we update the map at every time step. Formally, at the time step t, the robot observes a new RGB image I_{rgb}, a new depth image I_{depth}, and a relative pose change $p_t = \langle x_t, y_t, \theta_t \rangle$ with respective to the initial pose. We assume the knowledge of the camera intrinsic matrix K. Consequently, we begin by back-projecting each pixel $(i, j) \in I_{depth}$ into a 3-D point $p_{cam} = (x, y, z) = K^{-1}(i \times d(i, j), j \times d(i, j), d(i, j))^T$ in the camera frame, where $z = d(i, j)$ is the depth value for pixel (i, j). Then, we project the 3-D points to the world frame $p_{world} = T^{-1} \times p_{cam}$, where T is the extrinsic matrix. In our world frame definition, the origin is positioned at the top-left corner of the map, the x axis extends to the right, and the y axis extends downward, following the conventions established in [5, 6]. On the map, the robot is consistently initialized in the middle and facing to the right $\langle \frac{H}{2}, \frac{W}{2}, 0.0 \rangle$. Finally, the 3-D points P_w are projected to the map plane as follows:

$$(p^x_{map}, p^y_{map}) = [\frac{p^y_{world}}{\rho}, \frac{p^x_{world}}{\rho}] \tag{1}$$

Meanwhile, we use the visual encoder of LSeg $E_{\text{ViT}} : \mathbb{R}^{h \times w \times 3} \rightarrow \mathbb{R}^{h \times w \times C}$ to compute the dense pixel-wise visual-language features. Following the same transformation above, we store the pixel-wise visual-language feature $f_{ij} = E_{ViT}(I_{rgb}[i, j])$ of pixel (i, j) at the corresponding grid (p^x_{map}, p^y_{map}). In this way, the new visual-language features are projected onto the map plane as $\overline{M}$. The global map

$\mathcal{M}_{t-1}$ gets updated as follows:

$$\mathcal{M}_t[u,v] = \begin{cases} \overline{M}[u,v], \text{ if } \mathcal{M}_{t-1}[u,v] = \text{None} \\ \frac{\overline{M}[u,v] + \mathcal{M}_{t-1}[u,v]}{n+1}, \text{otherwise} \end{cases} \tag{2}$$

where u, v are grid cell indices and n is the number of stored features. As [18], we average the features in each grid cell to handle the situation where the same object might be perceived from different views.

4.3 Language indexing-based localizer

The LLMs-based instruction parser converts the instruction into a sequence of macro-action descriptions. Since we use off-the-shelf DD-PPO as the local policy, we propose to ground each macro-action description to a waypoint location on the map and set it as an intermediate point-goal. Formally, suppose the current robot location on the map is $\langle p^x_{\text{map}}, p^y_{\text{map}}, \theta_{\text{map}} \rangle$.

For *pure movement* macro-action description such as "{"name": "move_forward", "dist": D}", the waypoint position is computed as $\langle p^x_{\text{map}} + D \times \cos(\theta_{\text{map}}), p^y_{\text{map}} + D \times \sin(\theta_{\text{map}}), \theta_{\text{map}} \rangle$. When there is no specified moving distance, we set the default moving distance to be 0.5 metre. A similar strategy is applied to rotations.

For *landmark-associated* macro-action descriptions such as "{"name": "move_to_left", "landmark": "couch"}" (See Figure 2), we first localize all target objects on the visual-language map through language indexing. Specifically, we construct a label list $[l_{\text{target}}, l^2_{\text{default}}, ..., l^k_{\text{default}}, \text{other}]$ where the first world is the target landmark label and the remaining are the default labels. Note that "other" is LSeg's default label to represent any out-of-range object classes. LSeg's text encoder takes in the label list and outputs a text embedding feature matrix $f_{\text{text}} \in \mathbb{R}^{C \times (K+1)}$. The similarity score for every label at every grid cell can be computed as $\mathcal{M}_t \times f_{\text{text}}$, where $\mathcal{M}_t \in \mathbb{R}^{H \times W \times C}$. With the similarity matrix, the localizer decides the label for each grid cell by selecting the label with the maximal similarity score. Therefore, at every time step, an open-vocabulary semantic map is generated. To tackle the landmark object ambiguity, we first apply density-based spatial clustering (DBSCAN) [13] to find the centers of all landmark labels. Next, we compute the orientation and Euclidean distance between the robot's current location and the centers on the map. We select the nearest label in front of the robot and use the corresponding center location as the waypoint. The design choice is under the assumption that instructions in VLN-CE are generated from the perspective of the robot's egocentric view. Using online mapping, we can mitigate the object ambiguity issue during navigation (See Figure 6). The waypoint is further computed in the local controller and is defined as a 2-D egocentric polar coordinate (ρ, ϕ), where ρ represents the relative distance between the waypoint and the robot's current pose and ϕ is the egocentric orientation towards the waypoint on the map.

4.4 DD-PPO-based local controller

To deal with the noisy observations in the real world, we use the DD-PPO navigation policy, pre-trained on a large-scale point-goal navigation task, as the local controller [30, 35]. Specifically, the controller takes in a front-view RGB-D observation $\{I_{rgb}, I_{\text{depth}}\}$ and a point-goal represented as a 2-D egocentric polar coordinate (ρ, ϕ) as inputs. The off-the-shelf local policy $\pi(a_t | I_{rgb}, I_{depth}, (\rho, \phi))$ predicts the next action a_t. Specifically, the action space is discrete and contains four primitive actions including a "stop" action to indicate termination or reaching the goal point. Empirically, we found that the DD-PPO local policy needs to be reset after reaching each waypoint.

5 Experiments

5.1 Mobile robot and environment setup

We conducted all experiments using an Interbotix LoCoBot WX250 equipped with an Intel RealSense D435 camera for capturing both depth and RGB images. The RGB image dimensions are $640 \times 480 \times 3$ and the depth dimensions are 640×480. The camera is mounted on a Kobuki base at a height of approximately $53\,\text{cm}$ with an elevation angle of -15.7 degrees. In our experiments, we disabled the robot's arms and exclusively controlled the Kobuki base. We implemented four primitive actions to align with the output of the DD-PPO local policy. Specifically, the 'move_forward' action advances

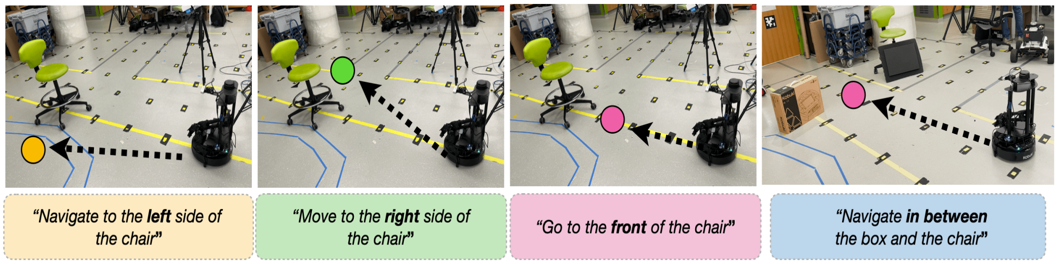

Figure 4: Single Instruction Following Task

the robot by $0.25\,\text{cm}$, while the 'turn_left' and 'turn_right' actions rotate the base by 15 degrees. A 'stop' action was also included for no movement. We conducted the entire experiment using ROS Noetic in an unstructured lab environment, which was unseen by both our pipeline and the baseline method. Figure 3 shows the robot used in all experiments.

5.2 Baseline

We compare our method against Cross-modal Map Learning (CM2) [15], a learning-based SOTA method to tackle the VLN-CE task. CM2 employs a strategy where it generates both global occupancy maps and global semantic maps by hallucinating information from local maps back-projected from depth and semantic observations, enhancing the spatial and semantic understanding of the unseen environments. Given an instruction, CM2 learns cross-modal map attention to ground the entire instruction into a sequence of waypoints on the map. The waypoint sequence will be predicted at every time step and the DD-PPO local policy is used to predict the next action. We select the best model provided by the author as our comparison [1]. To make the comparison fair, both CM2 and our method use the same front-view RGB-D observations and relative pose as inputs. We also use the same DD-PPO controller from [35, 30]. It's important to note that CM2 is extensively trained in simulation and does not undergo fine-tuning with real-world data. In contrast, our pipeline requires no training in the simulator and no fine-tuning in the real world, leveraging pre-trained foundation models.

Figure 3: Interbotix LoCoBot WX250

5.3 Instruction following tasks

In the context of VLN-CE, instructions often include several sub-instructions that the robot must follow. To evaluate the performance of our proposed pipeline, we designed two types of instruction-following tasks, ranging from easy to challenging. The first task, "**Single Instruction Following Task**", involves instructions that contain only one sub-instruction for the robot to execute. For instance, an instruction might read as "*Move forward by 2 meters.*" This task aims to assess the robot's ability to correctly infer the goal from the instruction and execute it accurately. It's important to note that in VLN, the goal location is implicitly encoded in the instruction. The second task, namely "**Complex Instruction Following Task**", is more demanding. Here, instructions include multiple sub-instructions that the robot must carry out. For instance, an instruction might read as "*Move to the left side of the chair. Then, turn left by 90 degrees.*" In addition to evaluating goal inference and execution accuracy, this task assesses each method's ability to ground complex instructions in the real world. Figure 4 and 5 show examples of the proposed tasks.

5.4 Results

Single Instruction Following - Pure Motion Task : We evaluate the accuracy of our method in executing instructions that involve pure movement. The tested straight distances

Table 1: Results of Pure Motion Task

Target Dist (m)	Actual Dist (m)	Est Dist (m)	Err Dist (m)
0.5	0.426	-	-
1.0	0.748	0.238	0.014
2.0	1.678	0.308	0.014

[1] https://github.com/ggeorgak11/CM2

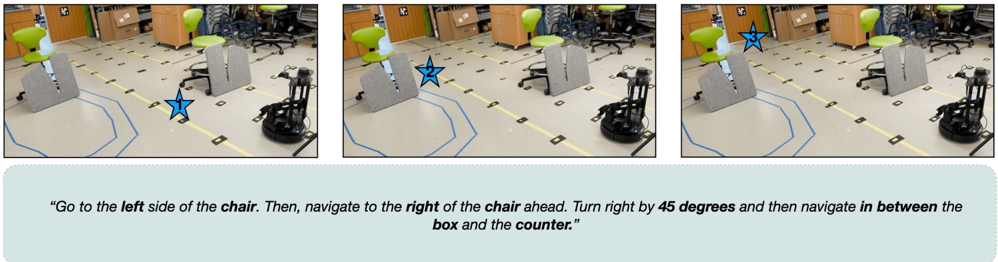

Figure 5: Complex Instruction Following Task

range from 0.5 to 2.0 meters, with each distance tested in 5 independent runs using different instructions. For example, instructions include “*Go forward by* 1.0 *meter*” or “*Navigate ahead by* 1.0 *meter.*” We provide two metrics: “*Actual Dist*”, representing the straight distance actually traversed by our pipeline, and “*Est Dist*”, an estimate of the distance between the stop position and the goal position on the map. Because the off-the-shelf DD-PPO controller outputs “STOP” action when it believes the goal is nearby. The movement error, “*Err Dist*” is computed by subtracting the “*Target Dist*” with the sum of “*Actual Dist*” and “*Est Dist*”. As shown in Table 1, the average movement error out of 15 runs in the real world is approximately $1.4\,\mathrm{cm}$. This result highlights the effectiveness of using DD-PPO as the local policy in real-world scenarios and the accuracy of the map built by our online mapper.

Single Instruction Following - Landmark-Associated Task : In this task, unlike the instructions provided, the instructions implicitly specifies spatial goals. We compare our method with the CM2 baseline using four different instructions, indicating four distinct spatial goals. For each instruction, we conduct 5 independent runs with varying initial robot locations. In Table 2, our approach significantly outperforms the CM2 baseline, achieving a much higher mean success rate of 95% compared to CM2’s 30% and substantially smaller distances to the goal location (Ours $0.2\,\mathrm{m}$ v.s. CM2’s $2.06\,\mathrm{m}$). The key to our method’s success lies in the powerful generalization ability of VLMs to real-world observations since the grounding a single instruction is not difficult. In contrast, CM2 struggles with generalization to the real world due to the visual domain gap, despite using the same DD-PPO local controller and being trained in visually realistic scenes in Matterport3D dataset [4], which consists of scenes reconstructed by real-world images captured from various indoor environments. The empirical results suggest that pre-trained VLMs would be powerful off-the-shelf visual encoders to tackle the unseen observations in the real world.

Table 2: Results of Landmark-associated Motion Task

Method	CM2 [15]	
Instruction	SR (%)	Dist to Goal (m)
“Navigate to the *left* side of the chair”	60	0.88
“Navigate to the *right* side of the chair”	40	0.97
“Navigate to the *front* of the chair”	20	1.37
“Move *in between* the box and the chair”	0	2.06
Average	30	1.32
Method	Ours	
“Navigate to the *left* side of the chair”	100	0.79
“Navigate to the *right* side of the chair”	100	0.83
“Navigate to the *front* of the chair”	100	0.81
“Move *in between* the box and the chair”	80	0.20
Average	**95**	**0.66**

Complex Instruction Following Task : In this task, we examine our pipeline with more complex instructions comprising multiple sub-instructions and landmark objects. The instruction used was: “*Go to the left side of the green chair. Then, navigate to the right of the red chair ahead. Turn right by 45 degrees and then navigate in between the box and the counter.*” We conduct the experiment with 5 independent runs from different initial robot locations and varying the description for every sub-motion. Table 3 shows the results. In summary, CM2 struggles to complete this complex instruction task. The reasons are two-fold. First, the visual domain gap still poses challenges. Second, such influence exaggerates when the instruction becomes more complex because the performance of grounding a complex instruction highly relies on the quality of the hallucinated map, which requires the method to be robust to unseen, complex, and noisy observations in the real world. In contrast, our method achieved a 100% success rate and

Table 3: Results of *Complex Instruction Following* Task

Method	SR (%)	Dist to Goal (m)	Time steps
CM2 [15]	0	4.9	203.4
Ours	**100**	**0.256**	**88.6**

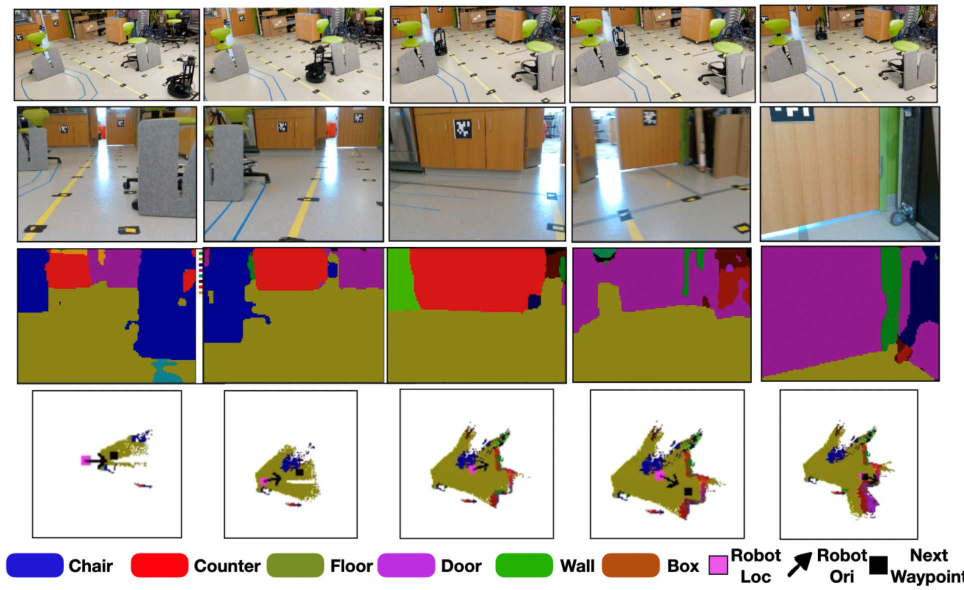

Figure 6: Trajectory visualization

stopped approximately 26 cm from the goal location, despite using the same DD-PPO controller as CM2. Unlike a simple language parser, we uses the LLMs to convert the instruction into a sequence of pre-defined macro-action descriptions, leveraging the powerful textual interpretation abilities of LLMs. Empirically, we found that LLMs is robust to different sub-instruction descriptions and can convert the sub-instruction to the desired format (e.g., a pre-defined macro-action description in our setting). Moreover, although CM2 achieves SOTA performance of VLN-CE in the simulation, it exhibits limited generalization to the VLN-CE task in the real world. As shown in Figure 6, we found that VLMs generalize surprisingly well to the observations in the real world even though they are not particularly trained for navigation tasks.

6 Limitations

While our pipeline demonstrates promising results, it does have certain limitations. Firstly, as instructions become more complex, the grounding performance of LLMs may be constrained by the scope of pre-defined macro-action descriptions. Complex instructions may include sub-instructions that are not easily convertible into pre-defined macro-action descriptions. Consequently, the design of an LLMs-based language grounding module remains an open question. Secondly, the localizer in our method prefers landmark objects or absolute movement for better waypoint proposals. However, objects can be ambiguous, and movement descriptions can be high-level. For example, there may be multiple similar objects on the map, and a movement description might simply be 'Exit the hallway,' with no specific landmark object or precise movement specified. Fortunately, recent approaches such as [37, 3] propose alternative design choices for LLMs-based instruction grounding, and recent advancements in VLMs [39, 27] shed light on language-referable object segmentation.

7 Conclusions

In this work, we propose a novel navigation framework to address the VLN-CE task in the real world scenarios. Leveraging three foundational models (LLMs, VLMs, and DD-PPO), and notably, *without any fine-tuning*, our method significantly outperforms the SOTA VLN-CE baseline. We have observed that the visual domain gap between simulation and the real world presents a significant challenge for transferring SOTA VLN-CE navigation policies from simulation to reality, even though the simulator contains real-world observations. Therefore, through our demonstrated instruction following tasks, we hope to provide insights into solving the VLN-CE task in real-world scenarios by harnessing the capabilities of these foundational models.

References

[1] Peter Anderson, Ayush Shrivastava, Joanne Truong, Arjun Majumdar, Devi Parikh, Dhruv Batra, and Stefan Lee. Sim-to-real transfer for vision-and-language navigation. In *Conference on Robot Learning*, pages 671–681. PMLR, 2021.

[2] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton 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*, pages 3674–3683, 2018.

[3] Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, et al. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on Robot Learning*, pages 287–318. PMLR, 2023.

[4] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niebner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. In *2017 International Conference on 3D Vision (3DV)*, pages 667–676. IEEE Computer Society, 2017.

[5] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural slam. In *International Conference on Learning Representations (ICLR)*, 2020.

[6] Devendra Singh Chaplot, Helen Jiang, Saurabh Gupta, and Abhinav Gupta. Semantic curiosity for active visual learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VI 16*, pages 309–326. Springer, 2020.

[7] Peihao Chen, Dongyu Ji, Kunyang Lin, Runhao Zeng, Thomas Li, Mingkui Tan, and Chuang Gan. Weakly-supervised multi-granularity map learning for vision-and-language navigation. *Advances in Neural Information Processing Systems*, 35:38149–38161, 2022.

[8] Shizhe Chen, Pierre-Louis Guhur, Cordelia Schmid, and Ivan Laptev. History aware multimodal transformer for vision-and-language navigation. *Advances in neural information processing systems*, 34:5834–5847, 2021.

[9] Shizhe Chen, Pierre-Louis Guhur, Makarand Tapaswi, Cordelia Schmid, and Ivan 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*, pages 16537–16547, 2022.

[10] Marvin M Chun and Yuhong Jiang. Contextual cueing: Implicit learning and memory of visual context guides spatial attention. *Cognitive psychology*, 36(1):28–71, 1998.

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[12] Russell A Epstein, Eva Zita Patai, Joshua B Julian, and Hugo J Spiers. The cognitive map in humans: spatial navigation and beyond. *Nature neuroscience*, 20(11):1504–1513, 2017.

[13] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.

[14] Samir Yitzhak Gadre, Mitchell Wortsman, Gabriel Ilharco, Ludwig Schmidt, and Shuran Song. Cows on pasture: Baselines and benchmarks for language-driven zero-shot object navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23171–23181, 2023.

[15] Georgios Georgakis, Karl Schmeckpeper, Karan Wanchoo, Soham Dan, Eleni Miltsakaki, Dan Roth, and Kostas Daniilidis. Cross-modal map learning for vision and language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15460–15470, 2022.

[16] Yicong Hong, Zun Wang, Qi Wu, and Stephen Gould. Bridging the gap between learning in discrete and continuous environments for vision-and-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15439–15449, 2022.

[17] Yicong Hong, Qi Wu, Yuankai Qi, Cristian Rodriguez-Opazo, and Stephen Gould. Vln bert: A recurrent vision-and-language bert for navigation. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 1643–1653, 2021.

[18] Chenguang Huang, Oier Mees, Andy Zeng, and Wolfram Burgard. Visual language maps for robot navigation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10608–10615. IEEE, 2023.

[19] Muhammad Zubair Irshad, Chih-Yao Ma, and Zsolt Kira. Hierarchical cross-modal agent for robotics vision-and-language navigation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13238–13246. IEEE, 2021.

[20] Muhammad Zubair Irshad, Niluthpol Chowdhury Mithun, Zachary Seymour, Han-Pang Chiu, Supun Samarasekera, and Rakesh Kumar. Semantically-aware spatio-temporal reasoning agent for vision-and-language navigation in continuous environments. In *2022 26th International Conference on Pattern Recognition (ICPR)*, pages 4065–4071. IEEE, 2022.

[21] Peter Karkus, Shaojun Cai, and David Hsu. Differentiable slam-net: Learning particle slam for visual navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2815–2825, 2021.

[22] Jacob Krantz, Aaron Gokaslan, Dhruv Batra, Stefan Lee, and Oleksandr Maksymets. Waypoint models for instruction-guided navigation in continuous environments. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15162–15171, 2021.

[23] Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan 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*, pages 104–120. Springer, 2020.

[24] Jacob Krantz, Erik Wijmans, Arjun Majundar, Dhruv Batra, and Stefan Lee. Beyond the nav-graph: Vision and language navigation in continuous environments. In *European Conference on Computer Vision (ECCV)*, 2020.

[25] Alexander Ku, Peter Anderson, Roma Patel, Eugene Ie, and Jason Baldridge. Room-across-room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4392–4412, 2020.

[26] Boyi Li, Kilian Q Weinberger, Serge Belongie, Vladlen Koltun, and Rene Ranftl. Language-driven semantic segmentation. In *International Conference on Learning Representations*, 2022.

[27] Feng Liang, Bichen Wu, Xiaoliang Dai, Kunpeng Li, Yinan Zhao, Hang Zhang, Peizhao Zhang, Peter Vajda, and Diana Marculescu. Open-vocabulary semantic segmentation with mask-adapted clip. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7061–7070, 2023.

[28] Abhinav Moudgil, Arjun Majumdar, Harsh Agrawal, Stefan Lee, and Dhruv Batra. Soat: A scene-and object-aware transformer for vision-and-language navigation. *Advances in Neural Information Processing Systems*, 34:7357–7367, 2021.

[29] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

[30] Ruslan Partsey, Erik Wijmans, Naoki Yokoyama, Oles Dobosevych, Dhruv Batra, and Oleksandr Maksymets. Is mapping necessary for realistic pointgoal navigation? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17232–17241, 2022.

[31] Yuankai Qi, Zizheng Pan, Yicong Hong, Ming-Hsuan Yang, Anton Van Den Hengel, and Qi Wu. The road to know-where: An object-and-room informed sequential bert for indoor vision-language navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1655–1664, 2021.

[32] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[33] Dhruv Shah, Błażej Osiński, Sergey Levine, et al. Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action. In *Conference on Robot Learning*, pages 492–504. PMLR, 2023.

[34] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

[35] Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames. *arXiv preprint arXiv:1911.00357*, 2019.

[36] Qiaoyun Wu, Jun Wang, Jing Liang, Xiaoxi Gong, and Dinesh Manocha. Image-goal navigation in complex environments via modular learning. *IEEE Robotics and Automation Letters*, 7(3):6902–6909, 2022.

[37] Andy Zeng, Maria Attarian, Krzysztof Marcin Choromanski, Adrian Wong, Stefan Welker, Federico Tombari, Aveek Purohit, Michael S Ryoo, Vikas Sindhwani, Johnny Lee, et al. Socratic models: Composing zero-shot multimodal reasoning with language. In *The Eleventh International Conference on Learning Representations*, 2022.

[38] Ce Zhou, Qian Li, Chen Li, Jun Yu, Yixin Liu, Guangjing Wang, Kai Zhang, Cheng Ji, Qiben Yan, Lifang He, et al. A comprehensive survey on pretrained foundation models: A history from bert to chatgpt. *arXiv preprint arXiv:2302.09419*, 2023.

[39] Xueyan Zou, Zi-Yi Dou, Jianwei Yang, Zhe Gan, Linjie Li, Chunyuan Li, Xiyang Dai, Harkirat Behl, Jianfeng Wang, Lu Yuan, et al. Generalized decoding for pixel, image, and language. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15116–15127, 2023.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] Please see Sec. 6
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...

(a) Did you state the full set of assumptions of all theoretical results? [N/A]
(b) Did you include complete proofs of all theoretical results? [N/A]

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] We will make the code, data, and instructions public soon
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [N/A] Our method does not require training.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A] Our pipeline requires no training.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]