2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Visual Language Maps for Robot Navigation

Chenguang Huang[1], Oier Mees[1], Andy Zeng[2], Wolfram Burgard[3]

***Abstract*— Grounding language to the visual observations of a navigating agent can be performed using off-the-shelf visual-language models pretrained on Internet-scale data (e.g., image captions). While this is useful for matching images to natural language descriptions of object goals, it remains disjoint from the process of mapping the environment, so that it lacks the spatial precision of classic geometric maps. To address this problem, we propose VLMaps, a spatial map representation that directly fuses pretrained visual-language features with a 3D reconstruction of the physical world. VLMaps can be autonomously built from video feed on robots using standard exploration approaches and enables *natural language indexing of the map* without additional labeled data. Specifically, when combined with large language models (LLMs), VLMaps can be used to (i) translate natural language commands into a sequence of open-vocabulary navigation goals (which, beyond prior work, can be spatial by construction, e.g., "in between the sofa and the TV" or "three meters to the right of the chair") directly localized in the map, and (ii) can be shared among multiple robots with different embodiments to generate new obstacle maps on-the-fly (by using a list of obstacle categories). Extensive experiments carried out in simulated and real-world environments show that VLMaps enable navigation according to more complex language instructions than existing methods. Videos are available at https://vlmaps.github.io.**

I. INTRODUCTION

People are excellent navigators of the physical world – due in part to their remarkable ability to build cognitive maps [1] that form the basis of spatial memory [2], [3] to (i) localize landmarks at varying ontological levels, such as a book; on the shelf; in the living room, or to (ii) determine whether the layout permits navigation between two points. Classic methods for robot navigation [4], [5] build geometric maps for path planning and can parse goals from natural language commands [6], [7], but struggle to generalize to unseen instructions. Learning methods directly optimize for navigation policies grounded in language end-to-end (commands to actions) [8], [9], but require copious amounts of data.

Meanwhile, recent works show that visual-language models (VLMs) [10], [11] pretrained on Internet-scale data (e.g., image captions) can be used out-of-the-box to ground language to the visual observations of a navigating agent, without additional data collection or model fine-tuning. These models enable mobile robots to handle new instructions that specify unseen object goals and can be combined with exploration algorithms to search for the first instance of any object (CoW) [12] or traverse object-centric landmarks in graphs (LM-Nav) [13]. While promising, these methods predominantly use VLMs as critics to match image observations to object goal descriptions, but do so in ways that remain disjoint from the mapping of the environment. Without grounding language onto a spatial representation, these systems may struggle to (i) recognize correspondences that associate independent observations of the same object, to (ii) localize spatial goals e.g., "in between the sofa and the TV", or to (iii) build persistent representations that can be shared across different embodiments, e.g., mobile robots, drones. Existing VLM-based solutions generalize to new object goals, but lose the spatial precision of classic geometric maps – is it possible to get the best of both?

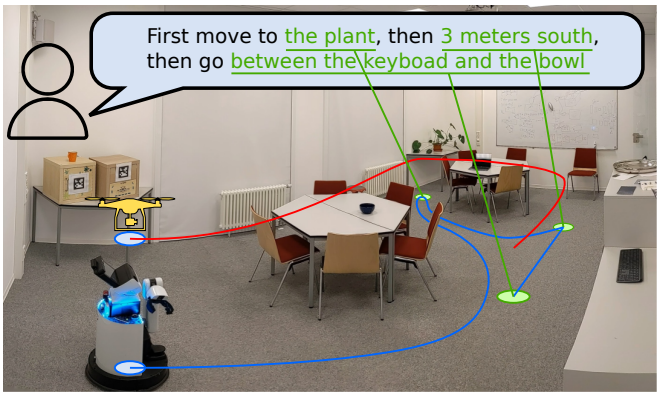


Fig. 1: **VLMaps** is a spatial map representation in which pretrained visual-language model features are fused into a 3D reconstruction of the physical world. Spatially anchoring visual language features enables *natural language indexing in the map*, which can be used to, e.g., localize landmarks or spatial references with respect to landmarks – enabling zero-shot spatial goal navigation without additional data collection or model finetuning.

In this work, we investigate the utility of a *spatial* visual-language map representation VLMaps, which fuses pretrained visual-language features from image observations directly with a 3D reconstruction of the physical world. VLMaps can be effectively built from video feed on robots using standard exploration algorithms. When paired with large language models (LLMs) in Socratic fashion [14], VLMaps can translate natural language instructions into a sequence of open-vocabulary goals, directly localized in the map. A key aspect of VLMaps is that they are spatial, which enables them to:

- Localize spatial goals beyond object-centric ones, e.g., "in between the TV and sofa" or "to the right of the chair" or "kitchen area" using code-writing LLMs, expanding beyond capabilities of CoW or LM-Nav.
- Generate new obstacle maps for new embodiments given natural language descriptions of landmark categories that they can or cannot traverse, e.g., "tables" are obstacles for a large mobile robot, but traversable for a drone.

Extensive experiments show that using VLMaps enables more effective long-horizon multi-object goal navigation than baseline alternatives, e.g., CoW [12] and LM-Nav [13], and, in particular, excels at enabling spatial open-vocabulary navigation tasks. We also provide ablations on different ways of constructing

[1]University of Freiburg, Germany.
[2]Google Research, USA.
[3]University of Technology Nuremberg, Germany.
This work has been supported partly by the German Federal Ministry of Education and Research under contract 01IS18040B-OML

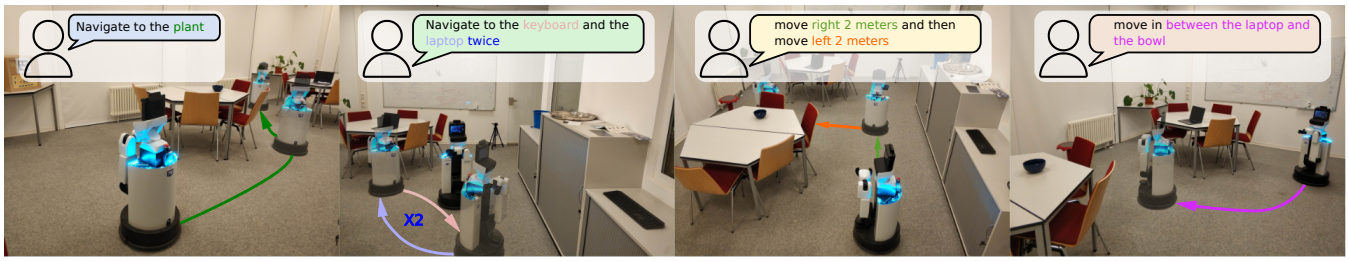


Fig. 2: VLMaps enables a robot to perform complex zero-shot spatial goal navigation tasks given natural language commands, without additional data collection or model finetuning.

VLMaps with different language models as well as a discussion on limitations, which point to areas for future work. Code and videos are available at https://vlmaps.github.io.

II. Related Work

Semantic Mapping. The maturity of traditional SLAM techniques together with the advancements in semantic understanding capabilities of convolutional neural networks has recently spurred considerable interest around augmenting 3D maps with semantic information [15], [16]. The literature has focused on either densely annotating 3D volumetric maps with 2D semantic segmentation CNNs [16] or object-oriented approaches [17], [18], [19], which build 3D maps around detected objects to enable object-level pose-graph optimization. Although progress has been made at generating more abstract maps, such as scene graphs [20], [21], current approaches are limited to a predefined set of semantic classes. In contrast to this, VLMaps are open-vocabulary semantic maps that, unlike prior work, enable *natural language indexing in the map*.

Vision and Language Navigation. Recently, also Vision-and-Language Navigation (VLN) has received increased attention [8], [22]. Further work has focused on learning end-to-end policies that can follow route-based instructions on topological graphs of simulated environments [8], [23], [24]. However, agents trained in this setting do not have low-level planning capabilities and rely heavily on the topological graph, limiting their real-world applicability [9]. Moreover, despite extensions to continuous state spaces [22], [25], [26], most of these learning-based methods are data-intensive.

Zero-shot Models. The recent success of large pretrained vision and language models [10], [27] has spurred a flurry of interest in applying their zero-sot capabilities to different domains including object detection and segmentation [28], [29], [11], robot manipulation [30], [31], [32], [33], and navigation [13], [12], [34]. Most related to our work is the approach denoted LM-Nav [13], which combines three pre-trained models to navigate via a topological graph in the real world. CoW [12] performs zero-shot language-based object navigation by combining CLIP-based [10] saliency maps and traditional exploration methods. However, both LM-Nav [13] and CoW [12] are limited to navigating to object landmarks and are less capable to understand finer-grained queries, such as "to the left of the chair" and "in between the TV and the sofa". In contrast, our method enables spatial language indexing beyond object-centric goals and can generate open-vocabulary obstacle maps. A concurrent work is NLMap [34], which demonstrates that VLMs can be used to build queryable scene representations to allow LLM robot planning [35] with new objects and locations.

III. Method

Our goal is to build a *spatial* visual-language map representation, in which landmarks ("the sofa") or spatial references ("between the sofa and the TV") can be directly localized using natural language. We propose VLMaps as one such representation, which can be constructed using off-the-shelf visual-language models (VLMs) and standard 3D reconstruction libraries. In the following subsections, we describe (i) how to build a VLMap (Sec. III-A), (ii) how to use these maps to localize open-vocabulary landmarks (Sec. III-B), (iii) how to build open-vocabulary obstacle maps from a list of obstacle categories for different robot embodiments (Sec. III-C), and (iv) how VLMaps can be used together with large language models (LLMs) for zero-shot spatial goal navigation on real robots from natural language commands (Sec. III-D), without additional data collection or model fine-tuning. Our pipeline is visualized in Fig. 3.

A. Building a Visual-Language Map

The key idea behind VLMaps is to fuse pretrained visual-language features with a 3D reconstruction. We achieve this by computing dense pixel-level embeddings from an existing visual-language model (over the video feed of the robot) and by back-projecting them onto the 3D surface of the environment (captured from depth data used for reconstruction with visual odometry).

In our work, we utilize LSeg [11] as the visual-language model, a language-driven semantic segmentation model that segments the RGB images based on a set of free-form language categories. The LSeg visual encoder maps an image such that the embedding of each pixel lies in the CLIP feature space. In our approach, wie fuse the LSeg pixel embeddings with their corresponding 3D map locations. In this way, without explicit manual segmentation labels, we incorporate a powerful language-driven semantic prior that inherits the generalization capabilities of VLMs. The only assumption we make is access to odometry, which is readily available from RGB-D SLAM systems and enables us to build a map from sequences of RGB-D images,

Formally, we define VLMap as $\mathcal{M} \in \mathbb{R}^{\bar{H} \times \bar{W} \times C}$, where $\bar{H}$ and $\bar{W}$ represent the size of the top-down grid map, and C represents the length of the VLM embedding vector for each grid cell. Together with the scale parameter s, a VLMap $\mathcal{M}$ represents an area with size $s\bar{H} \times s\bar{W}$ meters. To build the map, we, for each RGB-D frame, back-project all the depth pixels $\mathbf{u} = (u,v)$ to form a local depth point cloud that we transform to the world frame, $\mathbf{P}_k = D(\mathbf{u})K^{-1}\tilde{\mathbf{u}}$ and $\mathbf{P}_W = T_{Wk}\mathbf{P}_k$ where $\tilde{\mathbf{u}} = (u,v,1)$, K is the intrinsic matrix of the depth camera, $D(\mathbf{u}) \in \mathbb{R}$ is the depth value of the pixel $\mathbf{u}$, T_{Wk} is the transformation from the world coordinate frame to the k-th camera frame, $\mathbf{P}_k \in \mathbb{R}^3$ is the 3D point position in

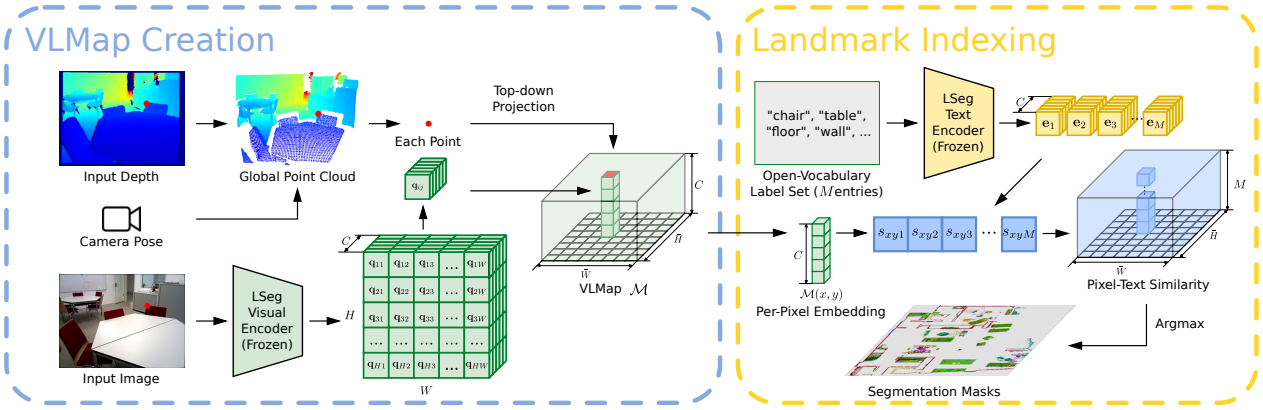


Fig. 3: System overview. A VLMap is created by fusing pretrained visual-language features into the reconstruction of the environment to enable visual-spatial-language-based reasoning. By providing a list of open-vocabulary labels, we retrieve segmentation masks for semantic classes required by downstream applications.

the k-th frame, and $\mathbf{P}_W \in \mathbb{R}^3$ is the 3D point position in the world coordinate frame. We then project the point $\mathbf{P}_W$ to the ground plane and get the pixel $\mathbf{u}$'s corresponding position on the grid map,

$$p^x_{map} = \left\lfloor \frac{\bar{H}}{2} + \frac{P^x_W}{s} + 0.5 \right\rfloor, p^y_{map} = \left\lfloor \frac{\bar{W}}{2} - \frac{P^z_W}{s} + 0.5 \right\rfloor \quad (1)$$

where p^x_{map} and p^y_{map} represent the coordinates of the projected point in the map $\mathcal{M}$.

Once we build the grid map, we apply LSeg's visual encoder $f(\mathcal{I}) : \mathbb{R}^{H\times W\times 3} \rightarrow \mathbb{R}^{H\times W\times C}$ to the RGB image $\mathcal{I}_k$ and generate the pixel-level embedding $\mathcal{F}_k \in \mathbb{R}^{H\times W\times C}$. Given the RGB-D registration, we project each image pixel $\mathbf{u}$'s embedding $\mathbf{q} = \mathcal{F}_k(\mathbf{u}) \in \mathbb{R}^C$ to its corresponding grid cell location (p^x_{map}, p^y_{map}) in the top-down grid map. Intuitively, there exist multiple 3D points projecting to the same grid location in the map. Thus, we average their embeddings, $\mathcal{M}(p^x_{map},\ p^y_{map}) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{q}_i$ where $\mathcal{M}(p^x_{map},\ p^y_{map}) \in \mathbb{R}^C$ represents the map features at the grid position (p^x_{map}, p^y_{map}), n represents the total number of points projecting to the grid location $(p^x_{map},\ p^y_{map})$, and $\mathbf{q}_i \in \mathbb{R}^C$ denotes the corresponding pixel embedding of each point. We note that these n points might not only come from a single frame, but also from points from multiple frames. Therefore, the resulting features contain the averaged embeddings from multiple views of the same object.

B. Localizing Open-Vocabulary Landmarks

We now describe how to localize landmarks in VLMaps with free-form natural language. Formally, we define the input language list as $\mathcal{L} = [\mathbf{l}_0, \mathbf{l}_1, ..., \mathbf{l}_M]$ where $\mathbf{l}_i$ represents the i-th category in text form, and M represents the number of categories defined by the user. Some examples of the input language list are ["chair", "sofa", "table", "other"] or ["furniture", "floor", "other"]. As Li *et al.* [11], we apply the pre-trained CLIP text encoder [10] to convert such list of texts into a list of vector embeddings $[\mathbf{e}_0, \mathbf{e}_1, ..., \mathbf{e}_M]$, $\mathbf{e} \in \mathbb{R}^C$, which are organized into an embedding matrix $E \in \mathbb{R}^{M\times C}$, where each row of the matrix represents the embedding of a category. The map embeddings $\mathcal{M}$ are also flattened into a matrix $Q \in \mathbb{R}^{\bar{H}\bar{W}\times C}$, where each row represents the embedding of a pixel in the top-down grid map. We then compute the pixel-to-category similarity matrix $S = Q \cdot E^T$, where $S \in \mathbb{R}^{\bar{H}\bar{W}\times M}$. Each element S_{ij} in the matrix stores the similarity value between a pixel and a text category, indicating how likely this pixel belongs to the class. By applying the argmax operator along the row direction to S and reshaping the resulting vector to shape $\bar{H}\times\bar{W}$, we get the final segmentation result $R \in \mathbb{R}^{\bar{H}\times\bar{W}}$. Each element R_{ij} represents the label index of the input language list $\mathcal{L}$ at the grid map location (i, j). With the final resulting matrix R, we compute the most related language-based category for every pixel in the grid map.

C. Generating Open-Vocabulary Obstacle Maps

Building a VLMap enables us to generate obstacle maps that inherit the open-vocabulary nature of the VLMs used (LSeg and CLIP). Specifically, given a list of obstacle categories described with natural language, we can localize those obstacles at runtime to generate a binary map for collision avoidance and/or shortest path planning. A prominent use case for this is sharing a VLMap of the same environment between different robots with different embodiments (i.e., cross-embodiment problem [36], [37]), which may be useful for multi-agent coordination [38]. For example, a large mobile robot may need to navigate around a table (or other large furniture), while a drone can directly fly over it. By simply providing two different lists of obstacle categories – one for the large mobile robot (that contains "table"), and another for the drone (that does not), we can generate two distinct obstacles maps for the two robots to use respectively, sourced on-the-fly from the same VLMap.

To do so, we first extract an obstacle map $\mathcal{O} \in \{0,1\}^{\bar{H}\times\bar{W}}$ where each projected position of the depth point cloud in the top-down map is assigned 1, and otherwise 0. To avoid points from the floor or the ceiling, points P_W are filtered out depending on their height,

$$\mathcal{O}_{ij} = \begin{cases} 1, & t_1 \le P^y_W \le t_2 \text{ and } p^x_{map} = i \text{ and } p^y_{map} = j \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

where $t_1, t_2 \in \mathbb{R}$ are the lower and upper thresholds for the y-component of the point P_W. Second, to obtain obstacle maps tailored to a certain embodiment, we define a list of potential obstacle categories $\mathcal{L}_{obs} = [\mathbf{l}_{obs0}, \mathbf{l}_{obs1}, \dots, \mathbf{l}_{obsM}]$, where $\mathbf{l}_{obsi}$ represents the i-th obstacle category in language, and M represents the total number of obstacle categories defined by the user. We

then apply the open-vocabulary landmark indexing introduced in Sec. III-B and obtain segmentation masks for all defined obstacles. For a specific embodiment k, we choose a subset of classes out of the whole potential obstacle list $\mathcal{L}_{obs}$ and take the union of their segmentation masks to get the obstacles mask $\tilde{\mathcal{O}}_{em_k}$. We ignore false predictions of obstacles on floor region in $\tilde{\mathcal{O}}_{em_k}$ by taking the intersection with $\mathcal{O}$ to get the final obstacle map $\mathcal{O}_{em_k}$.

D. Zero-Shot Spatial Goal Navigation from Language

In this section, we describe our approach to long-horizon (spatial) goal navigation, given a set of landmark descriptions specified by natural language instructions such as

```
move first to the left side of the counter, then
move between the sink and the oven, then move back
and forth to the sofa and the table twice
```

Notably different from prior work [12], [13], VLMaps allow us to reference precise spatial goals such as: "in between the sofa at the TV" or "three meters to the east of the chair." Specifically, we use a large language model (LLM) to interpret the input natural language commands and break them down into subgoals [35], [13], [14]. In contrast to prior work, which may reference these subgoals with language and map to low-level policies with semantic translation [39] or affordances [35], [40], [41], [42], we leverage the code-writing capabilities of LLMs to generate executable Python robot code [43], [33], [44], [27] that can (i) make precise calls to parameterized navigation primitives, and (ii) perform arithmetic when needed. The generated code can directly be executed on the robot with the built-in Python **exec** function.

Note that recent works [43], [33], [44], [27] have shown that code-writing language models (e.g., Codex [44]) trained on billions of lines of code from Github can be used to synthesize new simple Python programs from docstrings. In this work, we re-purpose these models for mobile robot planning, by priming them with several input examples of natural language commands (formatted as comments) paired with corresponding robot code (via few-shot prompting). The robot code can express functions or logic structures (if-then-else statements or for/while loops) and parameterize API calls (e.g., robot.move_to(target_name) or robot.turn(degrees). The full list is available in the Appendix, Sec. A) that map to spatial behaviors specified by the language commands. At test time, the models can subsequently take in new commands and autonomously re-compose API calls to generate new robot code respectively (prompt in gray, input task commands in green, and generated outputs are highlighted):

```
# move a bit to the right of the fridge
robot.move_to_right('refrigerator')
# face the toilet
robot.face('toilet')
# move to the west of the chair
robot.move_west('chair')
# turn right 20 degrees
robot.turn(20)
# move back and forth to the chair and table 3 times
pos1 = robot.get_pos('chair')
...
# move forward for 3 meters
robot.move_forward(3)
```

```
# move first to the left side of the counter, then
move between the sink and the oven, then move back and
forth to the sofa and the table twice
robot.move_to_left('counter')
robot.move_in_between('sink', 'oven')
pos1 = robot.get_pos('sofa')
pos2 = robot.get_pos('table')
for i in range(2):
   robot.move_to(pos1)
   robot.move_to(pos2)
# move 2 meters north of the laptop, then move 3
meters rightward
robot.move_north('laptop')
robot.face('laptop')
robot.turn(180)
robot.move_forward(2)
robot.turn(90)
robot.move_forward(3)
```

The code-writing LLM generates code that not only references the new landmarks mentioned in the language commands (as comments), but also can chain together new sequences of API calls to follow unseen instructions accordingly. The prompt has been truncated for brevity here. Please see the full prompt in the Appendix (Sec. B).

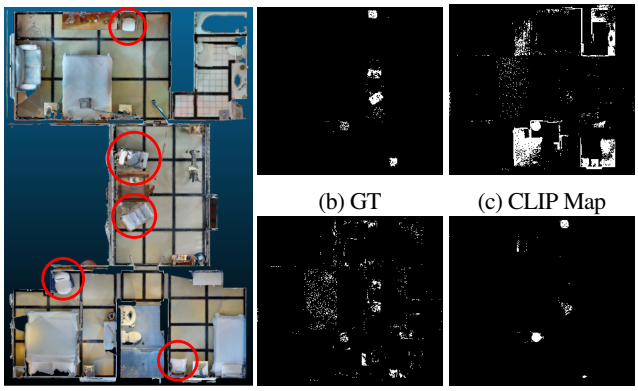
The navigation primitive functions being called by the language model (e.g., robot.move_to_left('counter')) use a pre-generated VLMap to localize the coordinates of the open-vocabulary landmarks ("counter") in the maps (described in Sec. III-B) modified with predefined scripted offsets (to define "left"). We then navigate to these coordinates using an off-the-shelf navigation stack that takes as input the embodiment-specific obstacle map (generated using the same VLMap, with the process described in Sec. III-C).

IV. Experiments

The goals of our experiments are four-fold: (i) to quantitatively evaluate our VLMaps approach against recent open-vocabulary navigation baselines on the standard task of multi-object goal navigation (Sec. IV-B), (ii) to investigate whether our method can better navigate to *spatial* goals specified by language commands versus alternative approaches (Sec. IV-C), (iii) to study whether VLMaps with their capacity to specify open-vocabulary obstacle maps can provide utility in improving the navigation efficiency of different robots with different embodiments (Sec. IV-D), and (iv) to demonstrate on real robots that VLMaps can enable zero-shot spatial goal navigation given unseen language instructions (Sec. IV-E).

A. Simulation Setup

Experimental setup. We use the Habitat simulator [45] with the Matterport3D dataset [46] for the evaluation of multi-object and spatial goal navigation tasks. The dataset contains a large set of realistic indoor scenes that help evaluate the generalization capabilities of navigating agents. To evaluate the creation of open-vocabulary multi-embodiment obstacle maps, we adopt the AI2THOR simulator due to its support of multiple agent types, such as LoCoBot and drone. In these two environments, the robot is required to navigate in a continuous environment with actions: **move forward 0.05 meters**, **turn left 1 degree**, **turn right 1 degree** and **stop**. For map creation in Habitat, we collect 12,096 RGB-D frames across ten dif-



(a) Top-Down Map (b) GT (c) CLIP Map (d) CoW (e) VLMaps

Fig. 4: Object mask for object type "chair". 4a shows the top-down map of the scene and the red circles specify the locations of type "chair". 4b shows the ground truth mask for type "chair" and 4c, 4d, 4e show the predicted masks by CLIP Map, CoW and VLMaps.

ferent scenes and record the camera pose of each frame. Similarly, we collect 1,826 RGB-D frames across ten rooms in AI2THOR.

Baselines. We evaluate VLMaps against three baseline methods, all of which utilize visual-language models and are capable of zero-shot language-based navigation:

- LM-Nav [13] creates a graph where image observations of an environment are stored as nodes while the proximity between images are represented as edges. By combining GPT-3 and CLIP, it parses language instructions into a list of landmarks and plans on the graph towards corresponding nodes.
- CLIP on Wheels (CoW) [12] achieves language-based object navigation by building a saliency map for the target category with CLIP and GradCAM [47]. By thresholding the saliency values, it retrieves a segmentation mask for the target object category and then plans the path on the map.
- CLIP-features-based map (CLIP Map) is an ablative baseline that generates a feature map for the environment in a similar way as ours. Instead of using LSeg visual features, it projects the CLIP visual features onto the map averaged across views. Object category masks are generated by thresholding the similarity between map features and the object category features.

For additional context and analysis, we also report results from a system that has access to a ground truth semantic map for navigation, to provide a systems-level upper bound on performance.

B. Multi-Object Navigation

We collect 91 sequences of tasks for the evaluation of object navigation. In each sequence, we randomly specify a starting position of the robot in one scene and then pick four among 30 object categories as subgoal object types. The robot is required to navigate to these four subgoals sequentially. In each sequence of subgoals, when the robot reaches one subgoal category, it should call the **stop** action to indicate its progress. We consider the navigation to one subgoal as success when the distance of stop position from the correct object is within one meter. To evaluate the long-horizon navigation capabilities of the agents, we compute the success rate (SR) of continuously reaching one to four subgoals in a sequence, shown in Tab. I. We also report the independent subgoal success rate, which indicates the total successful subgoals number divided by the total subgoals number (364 subgoals).

Tasks	No. Subgoals in a Row				Independent
	1	2	3	4	Subgoals
LM-Nav [13]	26	4	1	1	26
CoW [12]	42	15	7	3	36
CLIP Map	33	8	2	0	30
VLMaps (ours)	**59**	**34**	**22**	**15**	**59**
GT Map	91	78	71	67	85

TABLE I: The VLMaps-approach performs favorably over alternative open-vocabulary baselines on multi-object navigation (success rate [%]) and specifically excels on longer-horizon tasks with multiple sub-goals.

We observe that VLMaps performs consistently better compared to all baselines. LM-Nav has a weak performance as it is only able to navigate to locations represented by images stored in graph nodes. To obtain more insights into the map-based methods, we visualize the object masks generated by VLMaps, CoW, and CLIP Map, in comparison to GT, in Fig. 4. The masks generated by CoW (Fig. 4d) and CLIP (Fig. 4c) both contain considerable false positive predictions. Since the planning generates the path to the nearest masked target area, these predictions lead to planning towards wrong goals. In contrast, the predictions obtained with VLMaps shown in Fig. 4e are less noisy, which leads to higher success rates in object navigation.

C. Zero-Shot Spatial Goal Navigation from Language

In these experiments, we investigate the performance of VLMaps versus other baselines for zero-shot *spatial* goal navigation from language. Our benchmark consists of 21 trajectories in seven scenes, with manually specified corresponding language instructions for evaluation. Each trajectory contains four different spatial locations as subgoals. Examples of subgoals are "east of the table", "in between the chair and the sofa", or "move forward 3 meters". There are also instructions for the robot to realign itself in reference to nearby objects such as "with the counter on your right". We only consider a subgoal as having been achieved, when the robot reaches the subgoal location within a range of one meter. We compute the in-a-row success rate in the same way as in Sec. IV-B. For all map-based methods, including CoW, CLIP Map, ground truth semantic map and our method, we apply the code generation techniques introduced in Sec. III-D. For LM-Nav, we simply use the same parsing method in the original paper [13] to break down the language instruction into subgoals.

Tasks	No. Subgoals in a Row			
	1	2	3	4
LM-Nav [13]	5	5	0	0
CoW [12]	33	5	0	0
CLIP Map	19	0	0	0
VLMaps (ours)	**62**	**33**	**14**	**10**
GT Map	76	48	33	29

TABLE II: The VLMaps approach can navigate to spatial goals specified by natural language and outperforms other open-vocabulary zero-shot navigation baseline alternatives (success rate [%]) in this setting.

Tab. II summarizes the zero-shot spatial goal navigation success rates. Our method outperforms other baselines in this task.

Different from object navigation tasks where agents only need to approach a certain object type within a range disregarding the relative spatial shift to the object, the language-based spatial goal navigation tasks require the robot to accurately arrive at the described location in reference to the object. This poses a bigger challenge to the landmark localization ability of the method. The low localization ability of CoW and CLIP Map analyzed in the previous section (Sec. IV-B) leads to their high failure rates in this task.

D. Cross-Embodiment Navigation

We study the ability of VLMaps to improve navigation efficiency by retrieving different obstacle maps for navigation with different embodiments (given the same VLMap). We evaluate more than 100 sequences of subgoals as in Sec. IV-B in the AI2THOR simulator. We evaluate VLMaps on both a LoCoBot and a drone to test its capability of generating obstacle maps at runtime for multi-embodiment navigation. We apply the open-vocabulary obstacle map generation method in Sec. III-C to create an obstacle map for the drone (drone map) and one for the LoCoBot (ground map) by defining obstacles for them differently (see the prompts in Appendix Sec. E). We test the navigation ability of these embodiments with three setups: a LoCoBot with a ground map, a drone with a ground map, and a drone with a drone map.

We evaluate the Success Rate (SR) and the Success rate weighted by the (normalized inverse) Path Length (SPL) [48] 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 evaluated tasks, $S_i \in \{0,1\}$ is the binary indicator of success, l_i denotes the ground truth shortest path length, and p_i denotes the actual path length of the agent in navigation. This metric indicates how efficient the actual path is compared to the ground truth shortest path when the navigation task is achieved. In our three setups, the ground truth trajectories for the LoCoBot and the drone are planned on floor-level and on height level of 1.7 meters respectively.

Tasks	No. Subgoals in a Row								Independent Subgoals
	1		2		3		4		
	SR	SPL	SR	SPL	SR	SPL	SR	SPL	SR
LoCoBot (ground map)	53	**49.0**	28	**17.8**	14	6.7	6	**2.5**	52.3
Drone (ground map)	53	41.8	28	15.5	14	5.3	6	2.0	53.3
Drone (drone map)	**56**	45.4	**30**	16.3	**17**	**7.0**	**7**	**2.5**	**55.0**

TABLE III: VLMaps generate different obstacle maps for different robot embodiments, conditioned on a list of obstacle categories. This improves object navigation efficiency (Success [%] weighted by Path Length, SPL).

The results provided in Tab. III show that the average navigation success rates of the ground-map version of the LoCoBot and the drone are similar because the same obstacles map is used for planning. However, there is an obvious gap between their SPL values. This is because when the drone does not have access to a customized obstacle map, it fails to benefit from flying over ground objects to improve the navigation efficiency. In contrast, while achieving similar success rate compared to the drone with a ground map, the drone with a drone map manages to navigate with higher path efficiency, reflected by the increased SPL values. The comparable SPL values for the drone with the drone map and the LoCoBot with the ground map shows that VLMaps help to generalize the navigation efficiency among different embodiments. An example of the multi-embodiment object navigation task is shown in Fig. 5, where by defining a more efficient obstacles map, the drone flies over the sofa and reaches the laptop target directly, while the LoCoBot has to move aside first to avoid colliding with the sofa.

Fig. 5: VLMaps enable different embodiments to define their own obstacle maps for navigation. The left image shows the top-down view of an environment. The middle columns show the observations of agents during navigation. The images on the right demonstrate the obstacles maps generated for different embodiments and the corresponding navigation paths.

E. Real Robot Experiments

We also perform real-world experiments using the HSR mobile robot for indoor navigation given natural language commands. For map creation, we record 374 frames for the evaluated scene and use an off-the-shelf RGB-D SLAM solution, RTAB-Map [49] to estimate the camera poses. During inference, we also use the global localization module of RTAB-Map to initialize the robot pose. We test our VLMaps in a semantically rich indoor scene with more than ten different classes of objects. We define 20 different language-based spatial goals for testing purposes. Across different test runs, we initialize the robot at different locations.

The robot finishes ten navigation goals out of the 20. Among the successful trials, six of them are spatial goals like "move between the chair and the wooden box" or "move to the south of the table". three of them are goals relative to the current position of the robot like "move 3 meters right and then move 2 meters left". Another one is an instruction with repetition: "move between the keyboard and the laptop twice". We observe that failure cases are caused by: 1) inaccurate depth, which introduces noise during the map creation and decreases the landmark indexing accuracy and 2) action noise, which can negatively influence the navigation performance at test time. Overall, these results demonstrate the ability of VLMaps to index landmarks with natural language in the real world and, more importantly, its applicability to achieve a wide variety of open-vocabulary language-based spatial navigation goals.

V. Discussion and Limitations

In this work, we propose VLMaps, a spatial map representation enriched with pretrained visual-language features, which enables natural language indexing in the map. When combined with large language models, VLMaps can be applied in zero-shot spatial goal navigation and can be shared among multiple robots with different embodiments to generate new obstacles map in runtime. VLMaps are not without limitations. Notably, they remain sensitive to 3D reconstruction noise and odometry drift during navigation. They also cannot resolve object ambiguities during landmark indexing when the scene is cluttered with similar objects. In future work, we plan to improve VLMaps with better visual language models and to extend it to scenes with dynamic objects and moving humans.

References

[1] T. P. McNamara, J. K. Hardy, and S. C. Hirtle, "Subjective hierarchies in spatial memory." *Journal of Experimental Psychology: Learning, Memory, and Cognition*, vol. 15, no. 2, p. 211, 1989.

[2] M. M. Chun and Y. Jiang, "Contextual cueing: Implicit learning and memory of visual context guides spatial attention," *Cognitive psychology*, vol. 36, no. 1, pp. 28–71, 1998.

[3] E. L. Newman, J. B. Caplan, M. P. Kirschen, I. O. Korolev, R. Sekuler, and M. J. Kahana, "Learning your way around town: How virtual taxicab drivers learn to use both layout and landmark information," *Cognition*, vol. 104, no. 2, pp. 231–253, 2007.

[4] S. Thrun, W. Burgard, and D. Fox, "A probabilistic approach to concurrent mapping and localization for mobile robots," *Autonomous Robots*, vol. 5, no. 3, pp. 253–271, 1998.

[5] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard, "An evaluation of the rgb-d slam system," in *2012 IEEE international conference on robotics and automation*. IEEE, 2012, pp. 1691–1696.

[6] S. Tellex, T. Kollar, S. Dickerson, M. Walter, A. Banerjee, S. Teller, and N. Roy, "Understanding natural language commands for robotic navigation and mobile manipulation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 25, no. 1, 2011, pp. 1507–1514.

[7] M. MacMahon, B. Stankiewicz, and B. Kuipers, "Walk the talk: Connecting language, knowledge, and action in route instructions," *Def*, vol. 2, no. 6, p. 4, 2006.

[8] 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*, 2018, pp. 3674–3683.

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

[10] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International Conference on Machine Learning*. PMLR, 2021, pp. 8748–8763.

[11] B. Li, K. Q. Weinberger, S. Belongie, V. Koltun, and R. Ranftl, "Language-driven semantic segmentation," in *International Conference on Learning Representations*, 2021.

[12] S. Y. Gadre, M. Wortsman, G. Ilharco, L. Schmidt, and S. Song, "Clip on wheels: Zero-shot object navigation as object localization and exploration," *arXiv preprint arXiv:2203.10421*, 2022.

[13] D. Shah, B. Osinski, B. Ichter, and S. Levine, "Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action," *arXiv preprint arXiv:2207.04429*, 2022.

[14] A. Zeng, A. Wong, S. Welker, K. Choromanski, F. Tombari, A. Purohit, M. Ryoo, V. Sindhwani, J. Lee, V. Vanhoucke *et al.*, "Socratic models: Composing zero-shot multimodal reasoning with language," *arXiv preprint arXiv:2204.00598*, 2022.

[15] 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*, 2013, pp. 1352–1359.

[16] J. McCormac, A. Handa, A. Davison, and S. Leutenegger, "Semanticfusion: Dense 3d semantic mapping with convolutional neural networks," in *2017 IEEE International Conference on Robotics and automation (ICRA)*. IEEE, 2017, pp. 4628–4635.

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

[18] J. McCormac, R. Clark, M. Bloesch, A. Davison, and S. Leutenegger, "Fusion++: Volumetric object-level slam," in *2018 international conference on 3D vision (3DV)*. IEEE, 2018, pp. 32–41.

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

[20] N. Hughes, Y. Chang, and L. Carlone, "Hydra: a real-time spatial perception system for 3d scene graph construction and optimization," *Proceedings of Robotics: Science and Systems. New York City, NY, USA, http://dx. doi. org/10.15607/RSS*, 2022.

[21] S.-C. Wu, J. Wald, K. Tateno, N. Navab, and F. Tombari, "Scenegraphfusion: Incremental 3d scene graph prediction from rgb-d sequences," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 7515–7525.

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

[23] D. Fried, R. Hu, V. Cirik, A. Rohrbach, J. Andreas, L.-P. Morency, T. Berg-Kirkpatrick, K. Saenko, D. Klein, and T. Darrell, "Speaker-follower models for vision-and-language navigation," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[24] P.-L. Guhur, M. Tapaswi, S. Chen, I. Laptev, and C. Schmid, "Airbert: In-domain pretraining for vision-and-language navigation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 1634–1643.

[25] J. Krantz, A. Gokaslan, D. Batra, S. Lee, and O. Maksymets, "Waypoint models for instruction-guided navigation in continuous environments," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 15 162–15 171.

[26] Y. Hong, Z. Wang, Q. Wu, and S. 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*, 2022, pp. 15 439–15 449.

[27] 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.

[28] A. Kamath, M. Singh, Y. LeCun, G. Synnaeve, I. Misra, and N. Carion, "Mdetr-modulated detection for end-to-end multi-modal understanding," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 1780–1790.

[29] X. Gu, T.-Y. Lin, W. Kuo, and Y. Cui, "Open-vocabulary object detection via vision and language knowledge distillation," in *International Conference on Learning Representations*, 2021.

[30] M. Shridhar, L. Manuelli, and D. Fox, "Cliport: What and where pathways for robotic manipulation," in *Conference on Robot Learning*. PMLR, 2022, pp. 894–906.

[31] O. Mees, L. Hermann, E. Rosete-Beas, and W. Burgard, "Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks," *IEEE Robotics and Automation Letters (RA-L)*, vol. 7, no. 3, pp. 7327–7334, 2022.

[32] O. Mees, L. Hermann, and W. Burgard, "What matters in language conditioned robotic imitation learning over unstructured data," *IEEE Robotics and Automation Letters (RA-L)*, vol. 7, no. 4, pp. 11 205–11 212, 2022.

[33] O. Mees, J. Borja-Diaz, and W. Burgard, "Grounding language with visual affordances over unstructured data," *arXiv preprint arXiv:2210.01911*, 2022.

[34] B. Chen, F. Xia, B. Ichter, K. Rao, K. Gopalakrishnan, M. S. Ryoo, A. Stone, and D. Kappler, "Open-vocabulary queryable scene representations for real world planning," *arXiv preprint arXiv:2209.09874*, 2022.

[35] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," *arXiv preprint arXiv:2204.01691*, 2022.

[36] K. Zakka, A. Zeng, P. Florence, J. Tompson, J. Bohg, and D. Dwibedi, "Xirl: Cross-embodiment inverse reinforcement learning," in *Conference on Robot Learning*. PMLR, 2022, pp. 537–546.

[37] A. Ganapathi, P. Florence, J. Varley, K. Burns, K. Goldberg, and A. Zeng, "Implicit kinematic policies: Unifying joint and cartesian action spaces in end-to-end robot learning," *arXiv preprint arXiv:2203.01983*, 2022.

[38] J. Wu, X. Sun, A. Zeng, S. Song, S. Rusinkiewicz, and T. Funkhouser, "Spatial intention maps for multi-agent mobile manipulation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 8749–8756.

[39] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents," *arXiv:2201.07207*, 2022.

[40] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar *et al.*, "Inner monologue: Embodied reasoning through planning with language models," *arXiv preprint arXiv:2207.05608*, 2022.

[41] A. Zeng, "Learning visual affordances for robotic manipulation," Ph.D. dissertation, Princeton University, 2019.

[42] J. Borja-Diaz, O. Mees, G. Kalweit, L. Hermann, J. Boedecker, and W. Burgard, "Affordance learning from play for sample-efficient policy learning," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Philadelphia, USA, 2022.

[43] 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," *arXiv preprint arXiv:2209.07753*, 2022.

[44] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman *et al.*, "Evaluating large language models trained on code," *arXiv:2107.03374*, 2021.

[45] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra, "Habitat: A Platform for Embodied AI Research," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[46] 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," *International Conference on 3D Vision (3DV)*, 2017.

[47] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," *International Journal of Computer Vision*, vol. 128, no. 2, pp. 336–359, 2020.

[48] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva *et al.*, "On evaluation of embodied navigation agents," *arXiv preprint arXiv:1807.06757*, 2018.

[49] 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.

[50] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.