Mobile Robot Navigation Using Hand-Drawn Maps: A Vision Language Model Approach

Aaron Hao Tan, *IEEE Student Member*, Angus Fung, *IEEE Student Member*, Haitong Wang, and Goldie Nejat, *IEEE Member*

***Abstract*— Hand-drawn maps can be used to convey navigation instructions between humans and robots in a natural and efficient manner. However, these maps can often contain inaccuracies such as scale distortions and missing landmarks which present challenges for mobile robot navigation. This paper introduces a novel Hand-drawn Map Navigation (HAM-Nav) architecture that leverages pre-trained vision language models (VLMs) for robot navigation across diverse environments, hand-drawing styles, and robot embodiments, even in the presence of map inaccuracies. HAM-Nav integrates a unique Selective Visual Association Prompting approach for topological map-based position estimation and navigation planning as well as a Predictive Navigation Plan Parser to infer missing landmarks. Extensive experiments were conducted in photorealistic simulated environments, using both wheeled and legged robots, demonstrating the effectiveness of HAM-Nav in terms of navigation success rates and Success weighted by Path Length. Furthermore, a user study in real-world environments highlighted the practical utility of hand-drawn maps for robot navigation as well as successful navigation outcomes.**

Index Terms—Mobile robot navigation, vision language models, hand-drawn maps, robot planning

I. Introduction

Mobile robot navigation tasks may have to be performed in environments that can change, for example, due to structural instability in search and rescue scenarios [1], [2], construction progress during renovations or a new build [3], [4], and retail store reconfiguration [5]. In order for robots to navigate these environments they either use map-based [6]–[8] or map-less [9]–[11] methods. In map-based methods, accurate maps are generated prior to navigation using human teleoperation [1] or autonomous robot exploration [12]. However, map acquisition can be: 1) costly and time consuming [1], [13], and 2) requires expert knowledge [12]. On the other hand, map-less methods [9]–[11] can represent a robot's environment using real-time sensory data. However, existing

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC), and in part by the Canada Research Chairs program (CRC).The authors are with the Autonomous Systems and Biomechatronics Laboratory (ASBLab), Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, ON M5S 3G8, Canada (e-mail: aaronhao.tan@utoronto.ca;angus.fung@mail.utoronto.ca; haitong.wang@mail.utoronto.ca;nejat@mie.utoronto.ca). Corresponding author: Aaron Hao Tan.

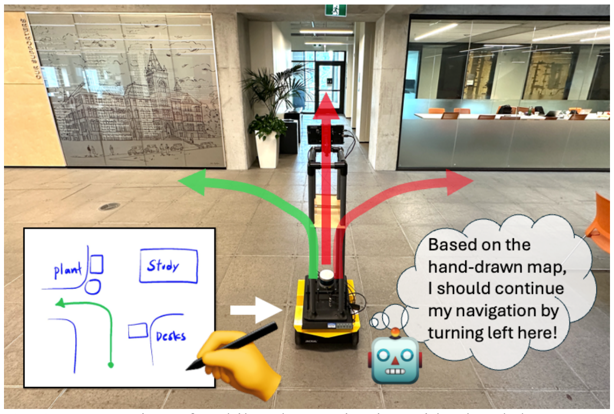

Plant
Study
Desks
Based on the hand-drawn map, I should continue my navigation by turning left here!

Fig. 1. An overview of mobile robot navigation with a hand-drawn map.

map-less methods require simultaneous exploration and navigation to reach the robot's goal position. This can affect navigation efficiency in terms of the total robot distance traveled due to the initial exploration stage [9].

Robot navigation using hand-drawn maps can provide an alternative approach to both map-based and map-less navigation methods, Fig. 1. Hand-drawn maps are freehand sketches generated by people based on their memory of an environmental layout to represent spatial relationships within the robot's environment [14]. Thus, hand-drawn maps can be used effectively for robot navigation without the need for a priori resource-intensive map acquisition [15], [16] or simultaneous exploration during navigation [17].

To date, existing robot navigation methods using hand-drawn maps can be classified as: 1) heuristic methods [15], [18]–[23], which recognize known landmarks for execution of predefined actions; and 2) probabilistic methods [12], [13], [17], [24]–[28], which match sensory data with map features for localization. However, these methods have been restricted to simple environments with hand-crafted landmarks, (e.g., boxes and cylinders), which do not represent complex real-world environments with realistic landmarks (e.g., furniture) [26] and multi-level floors. They have also require accurate hand-drawn maps with precise spatial layout representation [12], which are hard to obtain due to variations in human memory when sketching [26].

In this paper, we present a novel hand-drawn map robot navigation architecture, HAM-Nav, which uses pre-trained vision language models (VLMs) to interpret visual and textual cues from hand-drawn maps for robot navigation in unknown environments. HAM-Nav is the first method to generalize across diverse environments and varying hand-drawing styles without task-specific training. Our main research contributions are: 1) the introduction of a new adaptive visual prompting method, Selective Visual Association Prompting (*SVAP*), that places the robot view alongside a dynamically updated topological map overlaid on top of the hand-drawn map. Using *SVAP*, HAM-Nav can estimate the robot's position and select appropriate navigation actions in a zero-shot manner by enabling pre-trained VLMs to directly associate environmental features with corresponding elements in the hand-drawn map; and 2) the development of a Predictive Navigation Plan Parser (*PNPP*) to infer missing landmark information (e.g., class and location) using the common-sense knowledge of pre-trained VLMs to account for human errors in a hand-drawn map.

II. Related Works

We categorized the literature on robot navigation using hand-drawn maps into: 1) heuristic methods [15], [18]–[23], and 2) probabilistic methods [12], [13], [17], [24]–[28].

A. Heuristic Methods

Heuristic methods utilize pre-defined patterns such as spatial proximity to interpret landmark geometric features in hand-drawn maps and translate them into navigation commands for robots in simplistic environments. These methods include hand-crafted rule based [23], fuzzy control [15], [18]–[20], and optimization based [21], [22] approaches.

Hand-crafted rule based approaches compute objectness scores by combining image segmentation and fuzzy c-means algorithms to translate hand-drawn maps into navigation commands [23]. A robot updates its path using these objectness scores and sensory data during navigation.

Fuzzy control methods extract spatial descriptions from pixel coordinates of hand-drawn maps and categorize pixels into polygons (landmarks) and line segments (paths) [15], [18]–[20]. Polygon boundary forces determine the location of landmark relative to the robot. These locations are translated into actions (e.g. move forward) using fuzzy rules.

Optimization methods apply quadratic optimization to compute robot navigation waypoints by analyzing spatial relationships between landmarks and paths [21], [22]. Namely, hand-drawn waypoints are connected with virtual springs where quadratic programming is used to minimize potential energy to generate waypoints.

B. Probabilistic Methods

Probabilistic methods utilize statistical models to interpret hand-drawn maps and localize robots within them. These methods include Hidden Markov Models [24], Monte Carlo Localization (MCL) [13], [17], [25]–[27], Bayesian Filtering (BF) [12], and Supervised Learning (SL) [28].

In [24], a variable duration Hidden Markov Model (VDHMM) was trained using a dataset of hand-drawn maps to recognize strokes (lines) and inter-strokes (movements between lines) in order to generate robot movement vectors.

MCL methods determine a robot's position within a hand-drawn map using statistical sampling techniques such as particle filtering to either: 1) update a hand-drawn map to match the spatial layout of an environment [17], or 2) estimate map deformations [13], [25]–[27].

In [12], a BF method was used to generate a local occupancy grid map from panoramic RGB images obtained by a robot. The predicted grid map was aligned with a hand-drawn map by maintaining a belief over the robot's position. A similarity score between the predicted and observed local occupancy grid map was used to update the robot's estimated position. Using this updated belief, a grid search was used to generate a heuristic vector to guide the robot's navigation actions towards the goal.

In [28], a SL approach applied alpine-based registration to localize a robot within a hand-drawn map. A convolutional neural network (CNN) was used to predict the control points to guide the alignment of the robot's observation with the hand-drawn map. The A* algorithm for path planning was used for the robot to navigate towards the specified goal.

C. Summary of Limitations

Heuristic methods use reactive control to execute predetermined actions in controlled environments [15], [18]–[23]. They cannot adapt to changes in an environment and instead rely on recognizing only a fixed set of primitive landmarks. On the other hand, probabilistic methods require hand-drawn maps to accurately represent geometry and scale of an environment [12], [13], [17], [24]–[28]. Any deviation in the hand-drawn maps result in localization errors and imprecise navigation. Both methods have been applied to single-floor environments as they depend on 2D hand-drawn maps that directly correlate with the robot's sensor readings for localization. However, in multi-floor settings, a 2D hand-drawn sketch cannot represent vertically stacked floors without distorting spatial relationships by placing these floors side-by-side. Moreover, existing methods assume that all essential landmarks (shapes and locations) are correctly represented in the hand-drawn map which is challenging in real-world scenarios where human errors can introduce inaccuracies directly into the map [26].

To address the above limitations, we propose HAM-Nav, an architecture that uniquely leverages VLMs to: 1) detect realistic landmarks and enable zero-shot navigation in multi-floor environments by using adaptive visual prompting to align visual features from the environment with textual and spatial cues in the hand-drawn map, and 2) account for human errors in hand-drawn maps by using co-occurrence landmark patterns to predict missing landmarks.

III. Robot Navigation With Hand-Drawn Maps Problem Definition

The robot navigation problem using hand-drawn maps consists of requiring a mobile robot to autonomously navigate from a given starting position to a desired position within an unknown environment, utilizing navigation instructions conveyed through a free hand sketch. This sketch, referred to as a hand-drawn map, $\mathcal{M}_\text{h}$, is generated by a person based on their memory. $\mathcal{M}_\text{h} = (\mathcal{S}_\text{h}, \mathcal{L}_\text{h}, \mathcal{P}_\text{h})$ consists of three components, Fig. 2(a): 1) the spatial configuration, $\mathcal{S}_{\mathrm{h}}$, which represents the outer boundary and structural layout of the robot's environment; 2) the landmarks, $\mathcal{L}_{\mathrm{h}} = (\mathcal{L}_{\mathrm{h}}^{\mathrm{c}}, \mathcal{L}_{\mathrm{h}}^{\ell})$, which consist of text descriptions that depict a landmark's class (e.g., chair, desk), $\mathcal{L}_{\mathrm{h}}^{c}$, and its pixel location within $\mathcal{M}_{\mathrm{h}}$, $\mathcal{L}_{\mathrm{h}}^{\ell}$; and 3) a path, $\mathcal{P}_{\mathrm{h}}$. The path includes the initial robot position, $\mathrm{p}_0 = (x_0, y_0)$, Node 1 in Fig. 2(a), and the desired robot position $\mathrm{p}_{\mathrm{d}} = (x_d, y_d)$, Node 4 in Fig. 2(a). The true layout of the environment is denoted as $\mathcal{M}_{\mathrm{e}} = (\mathcal{S}_{\mathrm{e}}, \mathcal{L}_{\mathrm{e}})$, which represents an occupancy grid map that consists of two components: 1) the spatial configuration of the environment denoted as $\mathcal{S}_{\mathrm{e}}$, and 2) the landmarks within the environment, denoted as $\mathcal{L}_{\mathrm{e}} = (\mathcal{L}_{\mathrm{e}}^{\mathrm{c}}, \mathcal{L}_{\mathrm{e}}^{l})$, which includes the landmark class, $\mathcal{L}_{\mathrm{e}}^{\mathrm{c}}$, and location $\mathcal{L}_{\mathrm{e}}^{l}$. $\mathcal{M}_{\mathrm{h}}$ may differ from $\mathcal{M}_{\mathrm{e}}$ as a result of a person's recollection of the environment introducing possible errors in landmark positions, distances and scaling. A person may also misplace or omit landmarks in $\mathcal{M}_{\mathrm{h}}$, leading to an incomplete set for $\mathcal{L}_{\mathrm{h}}$ compared to $\mathcal{L}_{\mathrm{e}}$.

The mobile robot has an onboard RGB-D camera, $C(t)$, to capture both RGB, I_t^{RGB}, and depth images, I_t^{D} of its surroundings. The objective is to solve the following robot navigation problem: Given $\mathcal{M}_{\mathrm{h}}$ and $\mathcal{P}_{\mathrm{h}}$, a mobile robot must localize itself within $\mathcal{M}_{\mathrm{h}}$ and generate a sequence of actions $a(t)$ in order to navigate to p_{d} based on real-time observations $(I_t^{\mathrm{RGB}}, I_t^{\mathrm{D}})$ from $C(t)$. The sequence of actions $a(t)$ is determined by a robot action function $f(\cdot)$:

$$a(t) = f(\mathcal{M}_{\mathrm{h}}, \mathcal{P}_{\mathrm{h}}, C(t)) \text{ s.t. } \mathrm{p}(T) = \mathrm{p}_{\mathrm{d}}. \quad (1)$$

The goal is for the robot to select navigation actions in order to reach p_d at the final timestep, T.

IV. Hand-Drawn Map Navigation Architecture

The proposed Hand-Drawn Map Navigation Architecture (HAM-Nav) is presented in Fig. 2(b) and consists of the following four stages. **Stage 1: Prompt Engineering** consists of a *Topological Map Generator* (*TMG*), *Spatial Interpreter* (SI), *Visual Prompt Generator* (*VPG*), *Predictive Navigation Plan Parser* (*PNPP*), and an *Experience Manager* (*EM*) to extract navigation and environmental features from $\mathcal{M}_{\mathrm{h}}$ and I_t^{RGB} in order to structure and generate visual and textual prompts. **Stage 2: Position Estimation** uses a *Localization Engine* (*LE*) to estimate the robot's position p_t within $\mathcal{M}_{\mathrm{h}}$ based on the visual and textual prompts generated in Stage 1. **Stage 3: Action Selection** consists of a *Navigation Planning Engine* (*NPE*) to select an embodiment-agnostic discrete navigation action a based on p_t. Lastly, **Stage 4: Action Execution** uses a *Navigation Controller* (*NC*) to convert a into robot velocities to be executed in the environment. The following details the modules within each stage.

A. Topological Map Generator (TMG)

The *TMG* creates a topological map, $\mathcal{M}_{\mathrm{tp}}$, based on $\mathcal{M}_{\mathrm{h}}$ for robot localization and navigation planning. Namely, $\mathcal{M}_{\mathrm{h}}$ is first discretized into an occupancy grid, $\mathcal{M}_{\mathrm{g}}$, consisting of cells, g_i. $\mathcal{M}_{\mathrm{tp}} = (V, E)$ is a graph where $V = V_r \cup V_l$ represents the set of vertices comprising robot position nodes V_r, landmark nodes V_l, and a set of edges connecting these nodes, E, Fig. 2(a). V_r is generated by applying k-means clustering to the set of free cells in $\mathcal{M}_{\mathrm{g}}$ in order to obtain distinct clusters of free space, each corresponding to a potential robot position. The Google Cloud Vision API for optical character recognition (OCR) is used to generate V_l to identify and extract $\mathcal{L}_{\mathrm{h}}^{\mathrm{c}}$, and $\mathcal{L}_{\mathrm{h}}^{\ell}$ within $\mathcal{M}_{\mathrm{h}}$. Edges $e \in E$ connect V_r and V_l. $\mathcal{M}_{\mathrm{tp}}$ is provided to the following modules: 1) *SI* to perform landmark detection, 2) *VPG* for visual prompt generation, and 3) *PNPP* for path planning purposes.

B. Spatial Interpreter (SI)

At each timestep t: 1) a labeled image, $I_t^{\mathrm{b,RGB}}$, that includes the bounding boxes and object classes for the detected landmarks, and 2) a textual description, SD, of I_t^{RGB}. Two types of landmarks are detected within I_t^{RGB}: 1) object landmarks, L_{obj}, which include physical objects such as furniture and vehicles, and 2) structural landmarks, L_{str}, such as multi-way junctions (e.g., left and right turns).

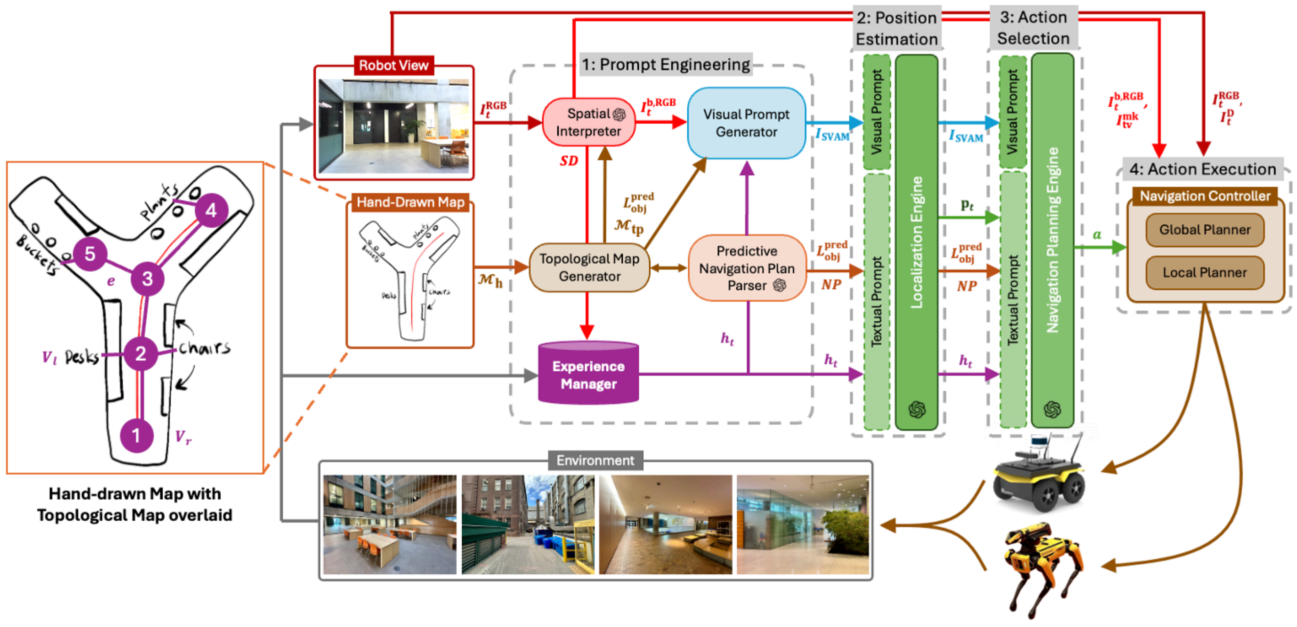
L_{obj} are detected using Grounding DINO (G-DINO) [29], an open vocabulary object detector . L_{str} are detected using our own three-stage approach, Fig. 3. Firstly, Grounded-Segment Anything Model (G-SAM) [30] is used to segment the traversable region within I_t^{RGB} by generating a pixel-level mask, $I_{\mathrm{tv}}^{\mathrm{mk}}$. In the second stage, edges are extracted from $I_{\mathrm{tv}}^{\mathrm{mk}}$ to generate I_{e} using the Hough Transform [31]. Edges in I_{e} are grouped into four different categories based on their orientation and length: horizontal $I_{\mathrm{e}}^{\mathrm{h}}$, vertical $I_{\mathrm{e}}^{\mathrm{v}}$, positive slope $I_{\mathrm{e}}^{\mathrm{p}}$, or negative slope $I_{\mathrm{e}}^{\mathrm{n}}$. Lastly, in the third stage, junction types are recognized by utilizing a decision function $f_{\mathrm{SL}}(I_{\mathrm{e}})$ to detect L_{str} based on the geometric relationships between $I_{\mathrm{e}}^{\mathrm{h}}$, $I_{\mathrm{e}}^{\mathrm{v}}$, $I_{\mathrm{e}}^{\mathrm{p}}$, and $I_{\mathrm{e}}^{\mathrm{n}}$:

$$f_{\mathrm{SL}}(I_{\mathrm{e}}) = \mathbb{I}\begin{Bmatrix}\text{Left if } d(I_{\mathrm{e}}^{\mathrm{h\cap v}}, I_{\mathrm{e}}^{\mathrm{v\cap p}}, I_{\mathrm{e}}^{\mathrm{p\cap h}}) \leq r \\ \text{Right if } d(I_{\mathrm{e}}^{\mathrm{h\cap v}}, I_{\mathrm{e}}^{\mathrm{v\cap n}}, I_{\mathrm{e}}^{\mathrm{n\cap h}}) \leq r\end{Bmatrix}, \quad (2)$$

where d is the Euclidean distance between the intersection points of each edge category. $\mathbb{I}$ denotes a Boolean indicator function that outputs a binary vector indicating the presence of a "left turn", and/or "right turn". In Fig. 3, the bounding boxes for both L_{obj} (pink) and L_{str} (green), are shown in $I_t^{\mathrm{b,RGB}}$. The bounding box coordinates for each landmark is denoted by L_{cd}.

The textual description of I_t^{RGB}, SD, at each t is obtained using a VLM. This process involves both a visual, $\sigma_{\mathrm{vis}}(I_t^{\mathrm{RGB}})$, and a textual, $\sigma_{\mathrm{text}}(L_{\mathrm{dict}})$, prompt. Specifically, L_{dict} describes the generalized landmark locations relative to the robot's perspective (e.g., left, front, right) within I_t^{RGB}. To obtain these generalized locations, we first divide I_t^{RGB} into three horizontal quadrants of equal widths: left quadrant Q_{l}, front quadrant Q_{f} and right quadrant Q_{r}. Each detected landmark is assigned to one of these quadrants based on the x-coordinate of the center of its bounding box, L_{cd}^{i}, in the pixel frame. The landmark quadrant assignments are then summarized into a dictionary, $L_{\mathrm{dict}} = \{L_{\mathrm{obj}}: Q\}$. The objective is to provide the VLM with generalized locations of landmarks in the textual prompt, following the textual format of "<L_{obj}> on your <Q>", to generate the final detailed textual description, SD of I_t^{RGB}. This process is described by:

$$SD = \mathrm{VLM}\left(\sigma_{\mathrm{vis}}(I_t^{\mathrm{RGB}}), \sigma_{\mathrm{text}}(L_{\mathrm{dict}})\right). \quad (3)$$



(a) (b)

Fig. 2. (a) Hand-drawn map with the spatial configuration $\mathcal{S}_h$ (black sketch), landmarks $\mathcal{L}_h$ (labeled with hand-written text), and hand-drawn path $\mathcal{P}_h$ (r line) overlaid with the topological map $\mathcal{M}_{tp}$ (purple line); and (b) the proposed HAM-Nav architecture. ֍ denotes a VLM.

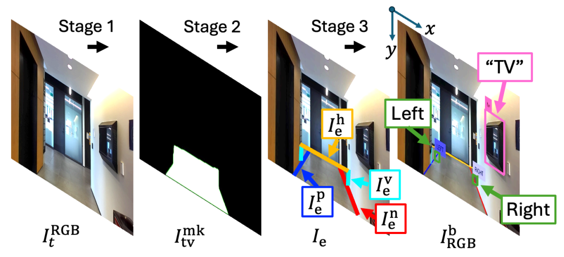


Fig. 3. The three stages of structural landmark detection.

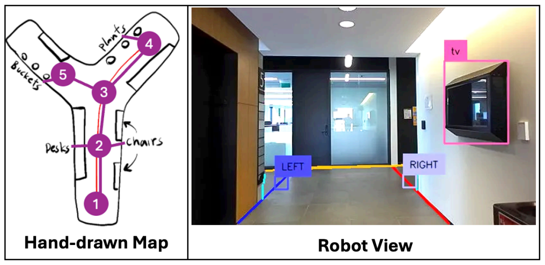


Fig. 4. A visual prompt, I_{SVAM}, that consists of a hand-drawn map with a pruned topological map, $\mathcal{M}'_{tp}$, and the robot view, $I_t^{b,RGB}$.

The $I_t^{b,RGB}$ is used by the *VPG* module for visual prompt generation and *NC* module for robot navigation, while *SD* is used by the *EM* module to retrieve relevant navigation experiences.

C. *Experience Manager (EM)*

The *EM* module collects and retrieves past navigation experiences to provide historical contextual navigation information using Retrieval Augmented Generation (RAG) [32]. The entire set of historical experiences, denoted as H, is stored onboard the robot, with each specific experience h_t, representing the navigation data at a particular t. Each h_t includes the prior observation SD'_t, estimated robot position p'_t, and executed action a'_t, at the corresponding t. To retrieve the most relevant h_t from H, the cosine similarity between the embedding of the current observation, E_c, and the embeddings of all past experiences E_p is calculated. The h_t with the highest cosine similarity score is provided as a textual prompt to the *LE* module for robot position estimation and the *NPE* module for navigation planning.

D. *Visual Prompt Generator (VPG)*

We developed the *VPG* module to enable Selective Visual Association Prompting by generating a visual prompt, I_{SVAM}, that determines the relationship between the features in I_t^{RGB} and $\mathcal{M}_h$, Fig. 4. Specifically, I_{SVAM} consists of an RGB image with two side-by-side components: 1) $I_t^{b,RGB}$, and 2) a pruned topological map, $\mathcal{M}'_{tp}$, overlaid on top of $\mathcal{M}_h$. Herein, $\mathcal{M}'_{tp}$ contains only the robot position node candidates with the highest likelihoods of representing the robot's true position in the environment. These likelihoods are determined using a probabilistic model that prunes position node candidates with low retention probability $\zeta(v_i)$. The $\zeta(v_i)$ for each $v_i \in V_r$ is determined by the following logistic function:

$$\zeta(v_i) = \frac{1}{1 + e^{\alpha\cdot(d(v_i,\mathrm{p}')-\beta)+\gamma\cdot\delta(v_i,\mathrm{p}',a')}}, \tag{4}$$

where α is a weighting factor that influences $\zeta(v_i)$ based on $d(v_i, \mathrm{p}')$, while β is a sensitivity parameter $\zeta(v_i)$ with respect to distance. The transition function $\delta(v_i, \mathrm{p}', a')$ determines the probability of the robot arriving at v_i after executing action a' at p'. γ is a weighting factor influencing $\zeta(v_i)$ based on δ. The topological map $\mathcal{M}'_{tp}$ includes only nodes where $\zeta(v_i)$ exceeds 0.5. We set $\beta = 2$ and both α and γ to 0.5. These values were selected through expert-guided empirical tuning to prune nodes with low likelihoods of representing the robot's position. I_{SVAM} is used by the *LE* and *NPE* modules for robot position estimation, and action selection, respectively.

E. Predictive Navigation Plan Parser (PNPP)

We uniquely propose a *PNPP* to provide: 1) predicted landmarks, $L_{\text{obj}}^{\text{pred}}$, and 2), a textual description of the navigation plan, NP. Specifically, the *PNPP* infers omitted landmarks using the VLM. The input to the VLM includes both a visual $\sigma_{\text{vis}}(\mathcal{M}_{\text{h}})$ and a textual $\sigma_{\text{text}}(V)$ prompt, which are conditioned on $\mathcal{M}_{\text{h}}$, and all V in $\mathcal{M}_{\text{tp}}$, respectively. The VLM uses this information to infer potential co-occurring landmarks based on the spatial relationships and proximity of nearby landmarks. The landmark prediction process for a node $v_i \in V$ is as follows:

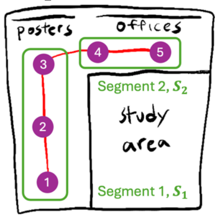
$$L_{\text{obj}}^{\text{pred}} = \text{VLM}(\sigma_{\text{vis}}(\mathcal{M}_{\text{h}}), \sigma_{\text{text}}(V)). \tag{5}$$

The $L_{\text{obj}}^{\text{pred}}$ for each v_i is incorporated into $\mathcal{M}_{\text{tp}}$ to be used by the *SI* module for landmark detection.

To generate NP, $\mathcal{M}_{\text{tp}}$ is segmented into local segments S_i by junction nodes $V_{\text{junc}} \subseteq V$, in the environment, shown in Fig. 5(a). For each S_i, the associated $L_{\text{str}}, L_{\text{obj}}, L_{\text{obj}}^{\text{pred}}$ are obtained from $\mathcal{M}_{\text{tp}}$. A descriptive sentence is generated for each S_i using:

$$NP_i = f_{\text{np}}\left(S_i, L_{\text{str}}, L_{\text{obj}}, L_{\text{obj}}^{\text{pred}}\right). \tag{6}$$

f_{np} maps each S_i and its landmarks ($L_{\text{str}}, L_{\text{obj}}, L_{\text{obj}}^{\text{pred}}$) to a local navigation plan, NP_i, following a fixed structure: "<navigation action> pass the <*landmarks*>, and <navigation action> when you see <*landmarks*>", as illustrated in Fig. 5(b). The collection of all local navigation plans $\{NP_i\}$ forms the global navigation plan NP, which is provided as textual prompt to both the *LE* and *NPE* modules.



(a)

1. **Go straight** pass the *desks, chairs, computers, whiteboards* and **Turn Right** when you see posters, *bulletin board, signage, clock.*
2. **Go straight** pass the *doors, nameplates,* and **Stop**.

(b)

Fig. 5. (a) $\mathcal{M}_{\text{tp}}$ overlaid on top of $\mathcal{M}_{\text{h}}$, with robot position nodes (purple) and landmark nodes (handwriting). The green boxes represent segments S_i; and (b) the local navigation plan, NP_i. Predicted landmarks $L_{\text{obj}}^{\text{pred}}$ are italicized.

F. Localization Engine (LE)

The *LE* module uses the VLM to estimate the robot's current position, p_t, by selecting a robot position node v_i in $\mathcal{M}'_{\text{tp}}$. The input to the VLM includes both visual, $\sigma_{\text{vis}}(I_{\text{SVAM}})$, and textual prompts, $\sigma_{\text{text}}(SD', \text{p}', a', NP)$. We utilize two prompting techniques for the textual prompt to estimate p_t. The first is chain of thought prompting (CoT) [33], to decompose the robot position estimation task into smaller explicit steps. This is achieved by asking the VLM to perform step-by-step reasoning by first identify visual landmarks and then relating these visual landmarks to the hand-drawn map before generating p_t. Score-based prompting (SB) [34] is used by asking the VLM to explicitly generate the probabilities of the robot position estimations. The position estimation process is:

$$\text{p}_t = \text{VLM}\big(\sigma_{\text{vis}}(I_{\text{SVAM}}), \sigma_{\text{text}}(SD', \text{p}', a')_{\text{CoT,SB}}\big). \tag{7}$$

The estimated p_t is used by the NPE module for robot navigation action selection.

G. Navigation Planning Engine (NPE)

The objective of the *NPE* is to generate embodiment-agnostic high-level actions such as "move forward", "turn right", "turn left", and "stop" using the VLM. The *NPE* module, like the *LE*, uses $\sigma_{\text{vis}}(I_{\text{SVAM}})$ and $\sigma_{\text{text}}(SD', \text{p}', a', NP, \text{p}_t)$ for zero-shot navigation decision making. CoT [33] and SB [34] prompting were used to guide the reasoning process. Specifically, CoT decomposes the navigation task into first understanding p_t, in $\mathcal{M}_{\text{h}}$, and then relating p_t to NP. CoT prompting is used to make this reasoning process explicit and sequential, while SB prompting is used to assign a probability score to each possible action, representing the likelihood of an action to successfully complete the navigation plan. The action with the highest probability score, a, is then selected and passed to the *NC* module for execution. The action selection process is:

$$a = \text{VLM}\big(\sigma_{\text{vis}}(I_{\text{SVAM}},), \sigma_{\text{text}}(SD', \text{p}', a', NP, \text{p}_t)_{\text{CoT,SB}}\big). \tag{8}$$

H. Navigation Controller (NC)

The *NC* module converts a into robot velocities (v, ω) for navigation execution. This is achieved in three stages using I_t^{RGB}, I_t^{D}, L_{cd} and $I_{\text{tv}}^{\text{mk}}$. In the first stage, the pixel coordinates $(h_{\text{cent}}, w_{\text{cent}})$ of the centroids of $I_{\text{tv}}^{\text{mk}}$ are used for the "move forward" action. For the "turn left" and "turn right" actions, the detected L_{str} coordinates $(h_{\text{L}}, w_{\text{L}})$ from L_{cd} are used. In the second stage, the 2D pixel coordinates, $\boldsymbol{\zeta} = (h, w)$, are projected into a 3D world coordinates, (x, y, z), to be passed to the robot path planners, using the pinhole camera model [35]. The third stage consists of global, Φ_{global}, and local Φ_{local}, path planners. Namely, Φ_{global}, is used to generate a sequence of waypoints, p_w, for the robot to follow. The local path planner, Φ_{local}, subsequently converts p_w into linear and angular velocities (v, ω), for robot navigation towards the desired position while avoiding dynamic and static obstacles.

V. Experiments

We conducted two sets of experiments to evaluate the performance of our novel HAM-Nav architecture: 1) an ablation study to assess the contributions of the specific design choices of HAM-Nav, and 2) a user study to investigate the feasibility and usability of HAM-Nav in real-world environments.

A. Ablation Study in Simulated Environments

Navigation performance was investigated using three metrics: 1) navigation time (NT), which measures the amount of time in seconds to reach the desired position $\mathbf{p_d}$, 2) success weighted by path length (SPL), to evaluate the robot's navigation path compared to the human hand-drawn path, and 3) success rate (SR), which represents the proportion of successful trials.

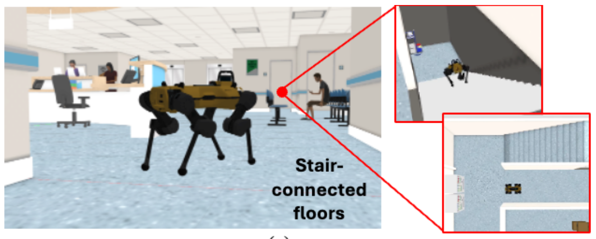
1) *Simulated Environments:* Two 3D photorealistic environments were generated in the Gazebo simulator, Fig. 6. The first environment was a structured indoor multi-floor workplace featuring rectilinear walls and stair-connected floors, Fig. 6(a). The second environment was an unstructured outdoor construction site with irregular navigation paths formed by randomly placed landmarks, Fig. 6(b). Examples of landmarks in these environments included pylons, boxes, dumpsters, chairs and computers, Fig. 6(c).

2) *Mobile Robots:* A Clearpath Jackal wheeled robot with a differential drive system, and a Boston Dynamic Spot quadruped robot were deployed. Both robots have an onboard RGB-D sensor. For the Jackal robot, the A* algorithm global planner [36] and the Timed Elastic Band Planner (TEB) local planner [37] were used. For the Spot robot, the Rapidly-exploring Random Trees based global planner (RRT) [38], and a non-linear model predictive controller (NMPC) local planner [39] were used. We used GPT4o as our VLM.

3) *Ablation Study Methods*: We compared HAM-Nav against the following configurations: (1) HAM-Nav without L_{dict} to assess the impact of generalized landmark locations in SD, (2) HAM-Nav without L_{obj}^{pred} to evaluate the effect of the predicted landmarks, (3) HAM-Nav without $\mathcal{M'}_{tp}$ to investigate the contribution of pruned topological maps, and (4) HAM-Nav without *EM* to determine the significance of historical navigation information.

4) *Procedure:* For each environment, two distinct hand-drawn maps were created for each robot platform with two random starting and desired positions. Five trials were conducted for each hand-drawn map across all ablation methods. To calculate SPL, $\mathcal{P}_h$ was converted into an equivalent path in the metric map, representing the optimal path. A successful trial is one where the desired position is reached.

5) *Results:* The average NT, SR and SPL for HAM-Nav and its variants are presented in Table I. Our complete HAM-Nav architecture had a higher overall performance across the different simulated environments, compared to all ablated methods. Specifically, HAM-Nav had the lowest NT (634s), the highest SR (80%), and the highest SPL (0.712).



(a)

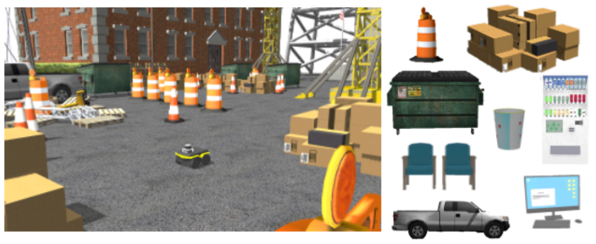

(b) (c)

Fig. 6. Simulated environment of: (a) an indoor multi-floor workplace ($25\ m \times 55\ m$), (b) an outdoor construction site ($40\ m \times 40\ m$), and (c) examples of photorealistic landmarks for both environments.

TABLE I: Ablation Study in Simulated Environments

Methods	Avg. NT (s) ↓	Avg. SR ↑	Avg. SPL ↑
HAM-Nav (ours)	634	80%	0.712
HAM-Nav w/o L_{dict}	743	45%	0.327
HAM-Nav w/o $L_{\text{obj}}^{\text{pred}}$	780	40%	0.287
HAM-Nav w/o $\mathcal{M'}_{\text{topo}}$	893	25%	0.134
HAM-Nav w/o EM	1583	5%	0.013

HAM-Nav without L_{dict} had the second best performance it terms of an average NT of 743 seconds, SR of 45%, and SPL of 0.327. The longer NT and lower SR and SPL were due to the lack of generalized landmark positions, which made this method prone to incorrectly estimate the robot's location during step-by-step reasoning. HAM-Nav without L_{obj}^{pred} relied solely on text from the hand-drawn map, which resulted in errors in landmark detection. HAM-Nav without $\mathcal{M'}_{\text{tp}}$ showed further degradation. Without pruning low likelihood robot positional nodes, I_{SVAM} became noisy, which led to an increase in incorrect robot position estimations. Lastly, HAM-Nav without the *EM* had the lowest performance overall due to VLM hallucinations. This resulted in repeated incorrect navigation actions without progress toward the goal. These results demonstrate the ability of HAM-Nav to execute robot navigation tasks based on hand-drawn maps across both single and multi-floor environments.

A. *User Study in Real-World Environments*

We conducted a user study to evaluate the feasibility and usability of HAM-Nav in real-world environments. We used the SR and SPL performance metrics. We also evaluated perceived usability of HAM-Nav using two standardized metrics: 1) the 5-point Likert System Usability Scale (SUS) [40], to provide an overall ease of use score, and 2) the Net Promoter Score (NPS) [41], to measure the likelihood of users recommending HAM-Nav based on their own experience.

1) *Real-World Environments:* Two structured indoor and one unstructured outdoor environments were used on the UofT campus, the: (1) Myhal Center for Engineering building (MH), Fig. 7(a), (2) Sandford Fleming building (SF), Fig. 7(b), and (3) Industrial Alley (IA), Fig. 7(c). The ground truth of MH is shown in Fig. 7(d).

2) *Mobile Robot:* A Jackal wheeled robot with a ZED Mini stereo camera was deployed. The A* algorithm [36] and the TEB planner [37] was used for global and local planning, respectively. We used GPT4o as our VLM.

3) *Procedure*: Ten engineering students, ages 22-42 (μ: 30.2, σ: 5.7) participated in the study. Each participant was given a 5-minute tour of each environment to observe the spatial layout. After the tour, two starting and desired robot positions were randomly chosen. Participants had 3 minutes to sketch an environment and the robot's path using an iPad and Apple Pencil. This simulated time-constrained scenarios. The robot executed two trials per environment. After all trials were completed for the environments, each participant rated the SUS questionnaire from 1 (strongly disagree) to 5 (strongly

TABLE II: Performance Metrics and Corresponding Results

SUS Questionnaire		**Median ($\tilde{x}$)**	***IQR***	**Frequency** **1**	**2**	**3**	**4**	**5**
S1	I think that I would like to use HAM-Nav frequently to provide navigation instructions to a mobile robot.	4	2	0	1	3	1	5
S2*	I found hand-drawn maps too complex for providing navigation instructions to a mobile robot.	2	0	2	7	1	0	0
S3	I thought HAM-Nav was easy to use.	4	0	0	0	0	7	3
S4*	I believe I would need help from a technical person to use HAM-Nav effectively.	1	1	6	1	2	1	0
S5	I thought the time provided for drawing the hand-drawn map was sufficient for me.	4	1	0	0	0	6	4
S6*	I thought the performance of HAM-Nav was inconsistent and did not meet my expectations based on my hand-drawn map.	2	1	4	6	0	0	0
S7	I would imagine that most people would learn to use HAM-Nav very quickly.	5	1	0	0	2	1	7
S8*	I found HAM-Nav to be very cumbersome to use.	2	1	3	5	1	1	0
S9	I felt confident using my memory to draw the map for navigation instructions for HAM-Nav.	4	1	1	1	1	5	2
S10*	I needed to learn many things before I could start using HAM-Nav.	1	1	6	2	1	1	0

** Statements are negatively worded.*

NPS Question: How likely is it that you would recommend HAM-Nav to a friend or colleague?

Participant #	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**		
SR	66.67%	83.33%	83.33%	100%	83.33%	100%	83.33%	83.33%	66.67%	33.33%	**Average**	78%
SPL	0.542	0.774	0.795	0.873	0.783	0.921	0.812	0.795	0.579	0.263	**Average**	0.714
SUS Score	70	77.50	70	70	70	95	92.50	95	70	60	**Average**	79.5
NPS Score	6	8	6	9	7	10	9	10	7	5	**Overall**	10

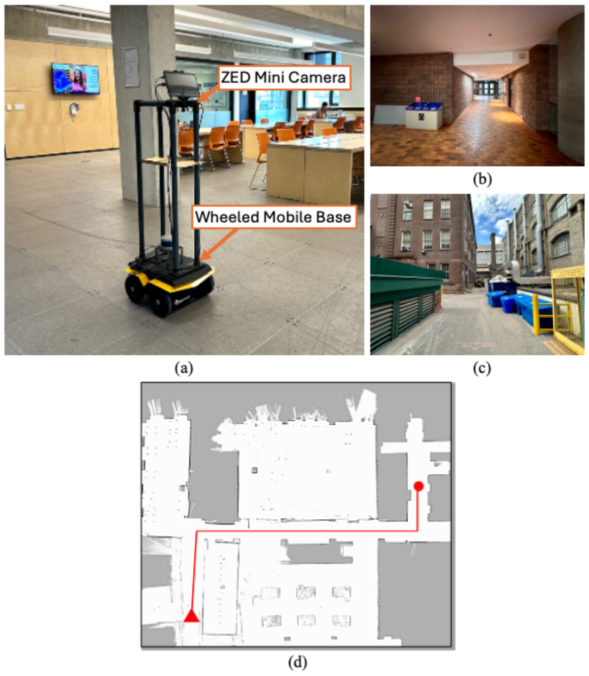


Fig. 7. (a) MH building ($40\,m \times 43\,m$), (b) SF building ($25\,m \times 15\,m$), (c) IA outdoor environment ($35\,m \times 25\,m$), (d) ground truth of MH with the starting (triangle) and desired (circle) robot positions.

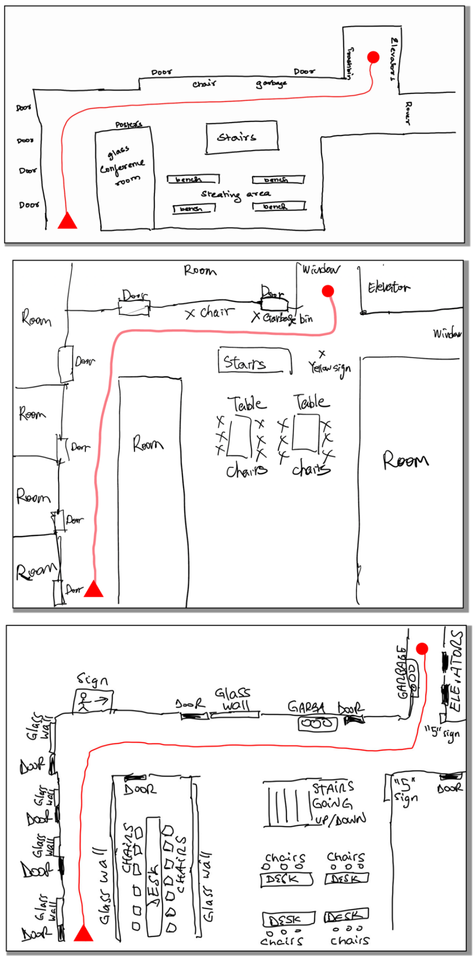


Fig. 8. Examples of hand-drawn maps in MH with low (top), medium (middle), and high (bottom) landmark densities. The starting and desired positions are denoted by red circles and triangles, respectively.

agree) and answered the NPS question from 1 (not at all likely), to 10 (extremely likely) to recommend HAM-Nav.

4) *Results:* Table II presents the SUS, NPS, SR, and SPL scores for the study. The real-world performance of HAM-Nav achieved an average SR of 78% and SPL of 0.714, which followed similar trends as in Section V.A. Additionally, HAM-Nav was able to generalize across diverse hand-drawing styles with varying landmark densities, Fig. 8, further demonstrating the robustness of our approach.

Overall, the average SUS score for HAM-Nav was 79.5. This score is defined as between "Good" and "Excellent" on the adjective rating scale [40]. Specifically, participants rated a strong willingness to use HAM-Nav frequently for providing navigation instructions (**S1**, $\tilde{x} = 4, IQR = 2$). They noted that HAM-Nav was not complex (**S2**, $\tilde{x} = 2, IQR = 0$), hard to use (**S3**, $\tilde{x} = 4, IQR = 0$), or cumbersome (**S8**, $\tilde{x} = 2$, $IQR = 1$). However, one participant believed they needed help from technical personnel (**S4**, $\tilde{x} = 1$, $IQR = 1$). Participants believed that most could learn to use HAM-Nav quickly (**S7**, $\tilde{x} = 5$, $IQR = 1$), felt confident using their memory to draw maps (**S9**, $\tilde{x} = 4$, $IQR = 1$), and that the time provided was sufficient (**S5**, $\tilde{x} = 4$, $IQR = 1$). Navigation performance of HAM-Nav also met participants' expectations (**S6**, $\tilde{x} = 2$, $IQR = 1$). The overall NPS score was 10 (within the range from -100 to 100), identifying HAM-Nav between "Good" and "Favorable" in terms of user recommendation likelihood [42]. A video of HAM-Nav is presented here: https://youtu.be/Pti9NnYSmeM.

VI. Conclusion

In this paper, we introduced the HAM-Nav architecture for mobile robot navigation using hand-drawn maps. Our approach uniquely leverages pre-trained VLMs for navigation. The novelty of HAM-Nav is in its robustness across varying environments and its ability to interpret diverse drawing styles without requiring the hand-drawn maps to be metrically accurate. The performance of HAM-NAV was validated through an ablation study as well as a user study. Results demonstrated that HAM-Nav can effectively navigate in both indoor and outdoor, single and multi-floor settings, with realistic landmarks. Future work will focus on extending HAM-Nav to support multi-robot systems.

References

[1] M. Skubic, D. Anderson, S. Blisard, D. Perzanowski, and A. Schultz, "Using a hand-drawn sketch to control a team of robots," *Auton. Robots*, vol. 22, no. 4, pp. 399–410, 2007.

[2] A. H. Tan, F. P. Bejarano, Y. Zhu, R. Ren, and G. Nejat, "Deep Reinforcement Learning for Decentralized Multi-Robot Exploration With Macro Actions," *IEEE Robot. Autom. Lett.*, vol. 8, no. 1, pp. 272–279, 2023.

[3] Y. Sun, I. Jeelani, and M. Gheisari, "Safe human-robot collaboration in construction: A conceptual perspective," *J. Safety Res.*, vol. 86, pp. 39–51, 2023.

[4] A. H. Tan, S. Narasimhan, and G. Nejat, "4CNet: A Diffusion Approach to Map Prediction for Decentralized Multi-robot Exploration," *arXiv*, pp. 1–16, 2024.

[5] I. Kramer, R. Memmesheimer, and D. Paulus, "Customer Interaction of a Future Convenience Store with a Mobile Manipulation Service Robot," *2021 IEEE Int. Conf. Omni-Layer Intell. Syst.*, pp. 1–7, 2021.

[6] H. Yang, C. Yao, C. Liu, and Q. Chen, "RMRL: Robot Navigation in Crowd Environments With Risk Map-Based Deep Reinforcement Learning," *IEEE Robot. Autom. Lett.*, vol. 8, no. 12, pp. 7930–7937, 2023.

[7] K. D. Katyal, A. Polevoy, J. Moore, C. Knuth, and K. M. Popek, "High-Speed Robot Navigation using Predicted Occupancy Maps," *IEEE Int. Conf. Robot. Autom.*, pp. 5476–5482, 2021.

[8] G. Chen *et al.*, "Robot Navigation with Map-Based Deep Reinforcement Learning," *IEEE Int. Conf. Networking, Sens. Control*, 2020.

[9] H. Wang, A. H. Tan, and G. Nejat, "NavFormer: A Transformer Architecture for Robot Target-Driven Navigation in Unknown and Dynamic Environments," *IEEE Robot. Autom. Lett.*, vol. 9, no. 8, pp. 1–8, 2024.

[10] L. Mezghan *et al.*, "Memory-Augmented Reinforcement Learning for Image-Goal Navigation," *IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, pp. 3316–3323, 2022.

[11] E. Wijmans *et al.*, "Dd-Ppo: Learning Near-Perfect Pointgoal Navigators From 2.5 Billion Frames," *Int. Conf. Learn. Represent.*, pp. 1–21, 2020.

[12] C. Xu, C. Amato, and L. L. S. Wong, "Robot Navigation in Unseen Environments using Coarse Maps," *IEEE Int. Conf. Robot. Autom.*, 2024.

[13] F. Boniardi, B. Behzadian, W. Burgard, and G. D. Tipaldi, "Robot navigation in hand-drawn sketched maps," *Eur. Conf. Mob. Robot.*, pp. 1–6, 2015.

[14] M. Mielle, M. Magnusson, and A. J. Lilienthal, "Using sketch-maps for robot navigation: Interpretation and matching," *Int. Symp. Safety, Secur. Rescue Robot.*, pp. 252–257, 2016.

[15] M. Skubic, P. Matsakis, B. Forrester, and G. Chronis, "Extracting navigation states from a hand-drawn map," *IEEE Int. Conf. Robot. Autom.*, vol. 1, pp. 259–264, 2001.

[16] J. Yun and J. Miura, "A Quantitative Navigability Measure of Rough Maps," *J. Robot. Mechatronics*, vol. 21, no. 1, pp. 95–103, 2009.

[17] K. Matsuo and J. Miura, "Outdoor visual localization with a hand-drawn line drawing map using FastSLAM with PSO-based mapping," *IEEE Int. Conf. Intell. Robot. Syst.*, pp. 202–207, 2012.

[18] G. Chronis and M. Skubic, "Sketch-based navigation for mobile robots," *IEEE Int. Conf. Fuzzy Syst.*, vol. 1, pp. 284–289, 2003.

[19] M. Skubic, C. Bailey, and G. Chronis, "A sketch interface for mobile robots," *Proc. IEEE Int. Conf. Syst. Man Cybern.*, vol. 1, pp. 919–924, 2003.

[20] M. Skubic, S. Blisard, C. Bailey, J. A. Adams, and P. Matsakis, "Qualitative Analysis of Sketched Route Maps: Translating a Sketch Into Linguistic Descriptions," *IEEE Trans. Syst. Man. Cybern.*, vol. 34, no. 2, pp. 1275–1282, 2004.

[21] D. C. Shah and M. E. Campbell, "A robust qualitative planner for mobile robot navigation using human-provided maps," *IEEE Int. Conf. Robot. Autom.*, pp. 2580–2585, 2011.

[22] D. C. Shah and M. E. Campbell, "A qualitative path planner for robot navigation using human-provided maps," *Int. J. Rob. Res.*, vol. 32, no. 13, pp. 1517–1535, 2013.

[23] J. Niu and K. Qian, "A hand-drawn map-based navigation method for mobile robots using objectness measure," *Int. J. Adv. Robot. Syst.*, vol. 16, no. 3, pp. 1–11, 2019.

[24] D. Shah, J. Schneider, and M. Campbell, "A sketch interface for robust and natural robot control," *Proc. IEEE*, vol. 100, no. 3, pp. 604–622, 2012.

[25] B. Behzadian, P. Agarwal, W. Burgard, and G. D. Tipaldi, "Monte Carlo localization in hand-drawn maps," *IEEE Int. Conf. Intell. Robot. Syst.*, pp. 4291–4296, 2015.

[26] F. Boniardi, A. Valada, W. Burgard, and G. D. Tipaldi, "Autonomous indoor robot navigation using a sketch interface for drawing maps and routes," *IEEE Int. Conf. Robot. Autom.*, vol. 2016-June, pp. 2896–2901, 2016.

[27] F. Foroughi, J. Wang, and Z. Chen, "Indoor robot localization in hand-drawn maps by using convolutional neural networks and Monte Carlo method," *ACM Int. Conf. Proceeding Ser.*, 2019.

[28] K. Chen, M. Vazquez, and S. Savarese, "Localizing against drawn maps via spline-based registration," *IEEE Int. Conf. Intell. Robot. Syst.*, pp. 8521–8526, 2020.

[29] S. Liu *et al.*, "Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection," *arXiv*, 2023.

[30] T. Ren *et al.*, "Grounded SAM: Assembling Open-World Models for Diverse Visual Tasks," *arXiv*, 2024.

[31] D. Duan, M. Xie, Q. Mo, Z. Han, and Y. Wan, "An improved Hough transform for line detection," *Int. Conf. Comput. Appl. Syst. Model. Proc.*, vol. 2, pp. 354–357, 2010.

[32] L. Caspari, K. G. Dastidar, S. Zerhoudi, J. Mitrovic, and M. Granitzer, "Beyond Benchmarks: Evaluating Embedding Model Similarity for Retrieval Augmented Generation Systems," *arXiv*, 2024.

[33] J. Wei *et al.*, "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," *Adv. Neural Inf. Process. Syst.*, vol. 35, pp. 1–14, 2022.

[34] D. Shah, M. Equi, B. Osinski, F. Xia, B. Ichter, and S. Levine, "Navigation with Large Language Models: Semantic Guesswork as a Heuristic for Planning," *Proc. Mach. Learn. Res.*, vol. 229, no. CoRL, pp. 1–17, 2023.

[35] A. H. Tan, A. Al-Shanoon, H. Lang, and M. El-Gindy, "Mobile Robot Regulartion with Image Based Visual Servoing," *Proc. ASME Int. Des. Eng. Tech. Conf. Comput. Inf. Eng. Conf.*, pp. 1–8, 2018.

[36] C. W. Warren, "Fast path planning using modified a method," *IEEE Int. Conf. Robot. Autom.*, vol. 2, pp. 662–667, 1993.

[37] C. Rosmann, W. Feiten, T. Wosch, F. Hoffmann, and T. Bertram, "Efficient trajectory optimization using a sparse model," *Proc. Eur. Conf. Mob. Robot.*, pp. 138–143, 2013.

[38] J. Norby and A. M. Johnson, "Fast global motion planning for dynamic legged robots," *IEEE Int. Conf. Intell. Robot. Syst.*, pp. 3829–3836, 2020.

[39] J. Norby *et al.*, "Quad-SDK," 2022. https://github.com/robomechanics/quad-software.

[40] J. R. Lewis, "The System Usability Scale: Past, Present, and Future," *Int. J. Hum. Comput. Interact.*, vol. 34, no. 7, pp. 577–590, 2018.

[41] S. Baehre, M. O'Dwyer, L. O'Malley, and N. Lee, "The use of Net Promoter Score (NPS) to predict sales growth: insights from an empirical investigation," *J. Acad. Mark. Sci.*, vol. 50, no. 1, pp. 67–84, 2022.

[42] A. Carpenter, "What is a good Net Promoter Score?," *qualtrics*. https://www.qualtrics.com/experience-management/customer/good-net-promoter-score/.