A Persistent Spatial Semantic Representation for High-level Natural Language Instruction Execution

Anonymous Author(s)
Affiliation
Address
email

Abstract: Natural language provides an accessible and expressive interface to specify long-term tasks for robotic agents. However, non-experts are likely to specify such tasks with high-level instructions, which abstract over specific robot actions through several layers of abstraction. We propose that key to bridging this gap between language and robot actions over long execution horizons are persistent representations. We propose a persistent spatial semantic representation method, and show how it enables building an agent that performs hierarchical reasoning to effectively execute long-term tasks. We evaluate our approach, the Hierarchical Language-conditioned Spatial Model (HLSM) on the ALFRED benchmark and achieve state-of-the-art results, despite completely avoiding the commonly used step-by-step instructions. https://hlsm-alfred.github.io/

Keywords: vision and language, spatial representations

1 Introduction

Mobile manipulation in a home environment requires addressing multiple challenges, including making long-term inferences about what actions to perform, where to explore, and which manipulations are necessary to enable others in the future. In addition to reasoning, robot agents also require an accessible yet sufficiently expressive interface to to specify the full diversity of possible tasks. Natural Language provides an intuitive mechanism for task specification, and coupled with advances in language understanding, is increasingly applied to embodied agents [e.g., 1–10].

However, robust reasoning about manipulation goals from unrestricted natural language instruction has a variety of open challenges. For instance, consider a object retrieval task in a home (Figure 1) with the instruction *secure two discs in a bedroom safe*. The robot must first locate the specific *safe* in the *bedroom*. It then needs to distribute the actions entailed by *secure* to two separate objects (*two discs*), each requiring a distinct sequence of actions, but targeting the same *safe*. It is also required to map between *secure* and its action space. In parallel, the robot must still address core mobile manipulation challenges, and often can only identify what actions are required as it observes and manipulates the world (e.g., the *safe* only needs to be opened if closed).

In this paper, we study the problem of learning a model to map high-level natural language instructions to low-level mobile manipulation actions in an interactive 3D environment [11] without step-by-step *low-level* instructions. Existing work largely studies language tightly aligned to the robot actions, either using single-sentence instructions [e.g., 1, 2, 5, 9] or sequences of instructions [12–14]. In contrast, we focus on high-level instructions, which require implicit reasoning across layers of abstraction to generate actions not explicit in the instruction. Key to our approach is hierarchical reasoning about high-level subgoals and their underlying low-level motion and manipulation capabilities. In our example (Figure 1), the sequence of subgoals is ⟨pick up a CD, open the safe, put the CD in the safe, . . . ⟩ and each is aligned with the corresponding actions the system takes to accomplish it. Generating the sequences of subgoals and actions requires global and consistent reasoning about the world state, for example to repeatedly return to the same *safe*.

We propose to construct & continually update a *spatial semantic representation* of the world from sequential robot observations, as it explores and acts (Figure 2). This persistent representation allows us to select subgoals that use previously observed objects currently outside of the agent's view, or to decide about needed exploration. Similar to widely used map representations [15], we retain the

Submitted to the 5th Conference on Robot Learning (CoRL 2021). Do not distribute.

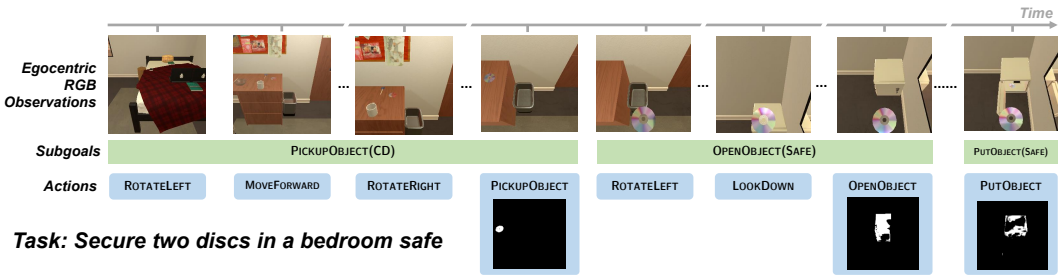

Figure 1: Illustration of the task and our hierarchical formulation. The agent receives a high-level goal specified in a natural language instruction. The goal of the agent is to map a sequence of RGB images to a sequence of navigation and manipulation actions that achieve the specified goal.

spatial properties of the environment, allowing the robot to effectively navigate and reason about the relations between objects, as expressed in the language or as required to accomplish its task. We show that our spatial representation is particularly suitable for hierarchical reasoning, which alternates between selecting the next subgoal and generating the actions to accomplish it.

We evaluate our approach, the Hierarchical Language-conditioned Spatial Model (HLSM) on the ALFRED [11] benchmark, showing the significant benefit of combining our spatial representation with hierarchical reasoning, allowing us to achieve state-of-the-art results without low-level instructions used by previous work [12–14]. This paper makes *three key contributions*: (a) the first demonstration of a modular representation learning approach for the problem of mapping high-level natural language task descriptions to actions in a 3D environment; (b) a method for utilizing a spatial semantic representation within a hierarchical model for solving mobile manipulation tasks; and (c) state-of-the-art performance on the ALFRED benchmark with only high-level instructions, instead of the more common step-by-step instructions.

2 Related Work

Natural language has been extensively studied in robotics research, including with focus on language grounding [1, 16–18], question generation [19], and dialogue [4, 20, 21]. Most work in this area has considered either synthetic instructions of relatively simple goals [7, 22–24], or natural language instructions where all intermediate steps are explained in detail [5, 11, 25–30]. In contrast, we focus on high-level instructions, which are more likely in home environments [31].

Representation of world state, action history, and language semantics plays a central role in robot systems and their algorithm design. Symbolic representations have been extensively studied for instruction following agents [1–4, 17, 31–39]. While they simplify the symbol grounding problem, they are laborious to scale to new environments and complex language. Representation learning presents an alternative by learning to map raw observations and language directly actions [5, 8, 9, 25]. World state and language semantics are typically represented with vectors [25] or by memorizing past observations [13]. Modelling improvements have enabled representation learning approaches to achieve good performance on complex language-conditioned navigation tasks [7, 9, 10, 25, 28, 29], but this success has not yet translated to mobile manipulation [11, 40, 41]. We propose integrating a spatial representation in form of a semantic voxel map within a hierarchical representation learning system. Semantic 2D maps have been successfully used in grounded language navigation tasks [7, 8, 42, 43]. We show that 3D semantic maps effectively facilitate following high-level natural language mobile manipulation instructions as well.

3 Problem Definition

Let $\mathcal{A}$ be the set of agent actions, and $\mathcal{S}$ the set of world states. Given a high-level natural language instruction L and an initial state $s_0 \in \mathcal{S}$, the agent's goal is to generate an execution $\Xi = \langle s_0, a_0, s_1, a_1, \ldots, s_T, a_T \rangle$, where $a_t \in \mathcal{A}$ is an action taken by the agent at time t, $s_t \in \mathcal{S}$ is the state before taking a_t, and $s_{t+1} = \mathcal{T}(s_t, a_t)$ under environment dynamics $\mathcal{T} : \mathcal{S} \rightarrow \mathcal{S}$. The state s_t is defined by the environment layout and the poses and states of all objects and the agent. The agent does not have access to the state s_t, but only to an observation o_t. An observation $o_t = (I_t, P_t, v_t^S, L)$ includes a first-person RGB camera image I_t, the agent's pose P_t, a one-hot

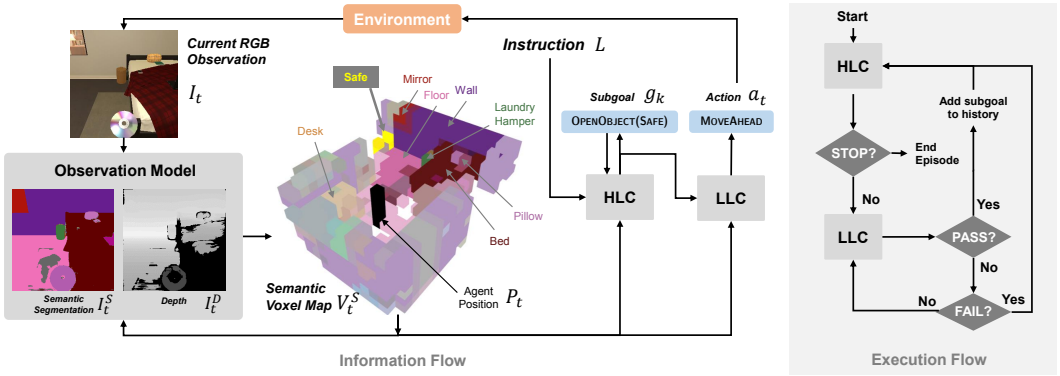

Figure 2: Model architecture illustration. Our approach consists of an observation model, high-level controller (HLC), and low-level controller (LLC). The execution flow is shown on the right, and the information flow between the modules is on the left. At each timestep, observation model takes as input the RGB image and updates the semantic voxel map representing the world state. The HLC uses the voxel map to predict the next subgoal given the instruction. The LLC outputs a sequence of actions that achieve the subgoal. The semantic voxel map is visualized in the middle. The agent position is shown as a black pillar. The current sugoal argument mask is bright-yellow. Other colors represent different segmentation classes. The more saturated blocks are observed in the current timestep, the washed-out blocks are remembered from previous timesteps.

encoding of the object class the agent is holding v_t^S, and the instruction L.[1] The task is considered successful if all goal-conditions corresponding to the task L are true at the final state s_T. Partial success is measured as the fraction of goal-conditions that have been achieved.

The ALFRED dataset includes sets of seen and unseen environments. The set of actions $\mathcal{A}$ includes parameter-free navigation actions (MoveAhead, RotateLeft, RotateRight) and interaction actions (e.g. PickupObject, ToggleObjectOn, ...) parameterized by a binary mask that identifies the argument of the interaction in the agent's current first-person view.

4 Hierarchical Model with a Persistent Spatial Semantic Representation

We model the agent behavior with a policy π that maps an instruction L and the observation o_t at time t to an action a_t. The policy π is made of an *observation model* F and two controllers: a high-level controller π^H and a low-level controller π^L. The policy maintains a spatial *state representation* $\hat{s}_t$ that captures its cumulative knowledge of the world at time t. Figure 2 illustrates the policy.

The high-level controller π^H computes a probability over *subgoals* g. A subgoal g is a tuple $(\texttt{type}, \texttt{arg}^C, \texttt{arg}^M)$, where type is an interaction type (e.g., OpenObject, PickupObject), $\texttt{arg}^C$ is the semantic class of the interaction argument (e.g., Safe, CD), and $\texttt{arg}^M$ is a 3D mask identifying the location of the argument instance. When predicting the k-th subgoal at time t, π^H considers the instruction L, the current state representation $\hat{s}_t$, and the sequence of past subgoals $\langle g_i, \rangle_{i<k}$. During inference, we sample from π^H. Unlike $\arg\max$, sampling allows the agent to re-try different subgoals if one fails (e.g., if a Mug was not found, pick up a Cup).

The low-level controller π^L is given the subgoal g_k as its goal specification at time t. At every timestep $j > t$, π^L maps the state representation $\hat{s}_j$ and subgoal g_k to an action a_j, until it outputs one of the stop actions: a_{PASS} or a_{FAIL} to indicate successful or failed subgoal completion.

Figure 2 (right) illustrates the execution flow. Execution starts at time $t = 0$ by sampling g_0 from π^H. Thereafter, the low-level controller π^L passes the control back to π^H through two actions: (a) a_{PASS}, which adds the subgoal g_k to the subgoal history; or (b) a_{FAIL}, which doesn't. π^H then computes a new distribution over subgoals to sample the $(k+1)$-th subgoal in case of a_{PASS}, or to resample k-th subgoal in case of a_{FAIL}. This process continues until π^H outputs the stop subgoal g_{STOP} or the maximum time horizon is exceeded.

4.1 State Representation

The state representation $\hat{s}_t$ at time t captures the agent's current understanding of the state of the world, including the locations of objects observed, and the agent's relation to them. The state repre-

[1] We compute P_t and v_t^S using dead-reckoning from RGB observations and actions.

sentation is a tuple $(V_t^S, V_t^O, v_t^S, P_t)$. The semantic map $V_t^S \in [0,1]^{W \times L \times H \times C}$ is a 3D voxel map that for every position indicates which of the $c \in [1, C]$ object classes are present in the voxel. The observability map $V_t^O \in \{0,1\}^{W \times L \times H}$ is a 3D voxel map that indicates whether the corresponding position has been observed. The inventory vector $v_t^S \in \{0,1\}^C$ indicates which of the C object classes the agent is currently holding. The agent pose $P_t = (x, y, \omega_p, \omega_y)$ is specified by the 2D position (x, y), pitch angle ω_p, and yaw angle ω_y.

We additionally compute 2D *state features* $\textsc{Afford}(\hat{s}_t) \in [0,1]^{7 \times W \times L}$ in a top-down view that represent each position with one or more of seven affordance classes {`pickable`, `receptacle`, `togglable`, `openable`, `ground`, `obstacle`, `observed`}. Each $[\textsc{Afford}(\hat{s}_t)]_{(\tau,x,y)} = 1.0$ if at least one of the voxels at position (x, y) has affordance class τ, otherwise it is zero.[2]

4.2 Observation Model

The observation model $F(\hat{s}_{t-1}, o_t, g_k)$ updates the state representation with new observations. It considers the current subgoal g_k to actively acquire information relevant to g_k. The computation of F consists of three steps: perception, projection, accumulation.

Perception Step We predict semantic segmentation I_t^S and depth map I_t^D from the RGB observation I_t. We use neural networks pre-trained in the ALFRED environment. The semantic segmentation $[I_t^S]_{(u,v)}$ is a distribution over C object classes at pixel (u, v). The depth map $[I_t^D]_{(u,v)}$ is a binned distribution.[3] We also heuristically compute a binary mask M_t^D that indicates which pixels have confident depth readings. We allow more confidence slack in pixels that correspond to the current subgoal argument $\texttt{arg}_t^C$ according to I_t^S. Appendix 9.1 provides further details. The perception step models are designed to be trained with the limited data available in ALFRED, and can be replaced with more general models given more perception data [44, 45].

Projection Step We use a pinhole camera model to convert depth I_t^D and segmentation I_t^S to a point cloud that represents each image pixel (u, v) with a 3D position $(x, y, z) \in \mathbb{R}^{W \times L \times H}$ and a semantic distribution $[I_t^S]_{(u,v)}$. We use $\arg\max_B(I_t^D)$ to compute the 3D positions, and discard points at pixels (u, v) when the binary mask value is $[M_t^D]_{(u,v)} = 0$. We construct a discrete semantic voxel map $\hat{V^S}_t \in [0,1]^{W \times L \times H \times C}$, where W, L, and H are the width, height, and length. The value at each voxel $[\hat{V^S}_t]_{(x,y,z)}$ is the element-wise maximum of the segmentation distributions across all points within the voxel. We additionally compute a binary observability map $\hat{V^O}_t \in \{0,1\}^{W \times L \times H}$ that indicates the voxels observed at time t. A voxel is observed if it contains any points, or if a ray cast from the camera through the voxel centroid has expected depth greater than the distance from the camera to the centroid. Free-space voxels seen through are observed.

Accumulation Step We integrate $\hat{V^S}_t$ and $\hat{V^O}_t$ into the persistent state representation:

$$V_t^S = \hat{V^S}_t \times \hat{V^O}_t + V_{t-1}^S \times (1 - \hat{V^O}_t) \qquad V_t^O = \max(V_{t-1}^O, \hat{V^O}_t) \ . \tag{1}$$

This operation updates each voxel with the most recent semantic distribution, while retaining the values of all previously observed voxels not visible at time t. The output of the observation model is the spatial state representation $\hat{s}_t = (V_t^S, V_t^O, v_t^S, P_t)$. The inventory v_t^S and pose P_t are taken directly from the observation o_t.

4.3 High-level Controller (π^H)

At timestep t, when invoked for the k-th time, the input to π^H is the instruction L, the sequence of past subgoals $\langle g_i \rangle_{i<k}$, and the current state representation $\hat{s}_t$. The output is the next subgoal $g_k = (\texttt{type}_k, \texttt{arg}_k^C, \texttt{arg}_k^M)$. The controller model encodes the three inputs and predicts the subgoal via sampling. Figure 3 illustrates the high-level controller architecture.

Input Encoding We encode the text L using a pre-trained BERT [46] model,[4]. We use the CLS token embedding as the task embedding ϕ^L. We encode the state representation $\hat{s}_t$ to account for classes of all observed objects, and the class of the object that the agent is holding:

[2] We assume a known mapping between object semantic classes and affordance classes.

[3] We use B uniformly spaced depth bins $\{0, \Delta_D, 2\Delta_D, \ldots, (B-1)\Delta_D\}$, where Δ_D is a depth resolution. We suggest Δ_D should be less than 50% of the voxel size.

[4] During learning, we fine-tune BERT.

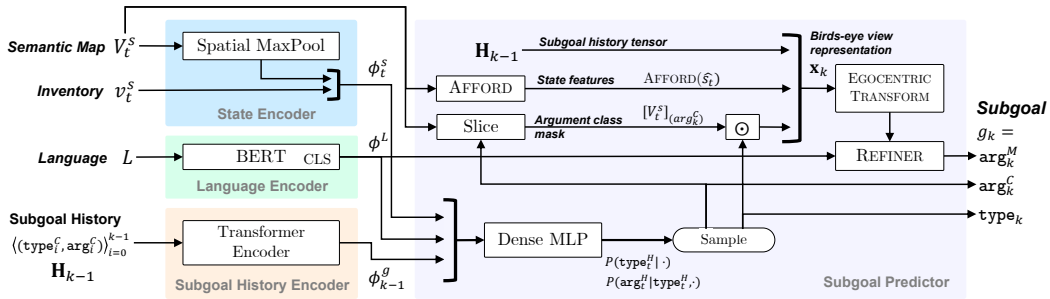

Figure 3: Illustration of the high-level controller π^H (Section 4.3).

$\phi^s(\hat{s}_t) = [v_t^S; \max_{(x,y,z)}(V_t^S)]$, where $\max_{(x,y,z)}$ is a max-pooling operation over spatial dimensions and $[\cdot;\cdot]$ denotes concatenation. We compute the representations of previous subgoals as $\langle \textsc{Repr}(g_i)\rangle_{i=0}^{k-1}$, where $\textsc{Repr}(g_i)$ is the sum of a sinusoidal positional encoding [47] of index i and learned embeddings for $\texttt{type}_i$ and $\texttt{arg}_i^C$. We process this sequence with a two-layer transformer autoregressive encoder [47] to compute $\langle \phi_i^g \rangle_{i=0}^{k-1}$. We take ϕ_{k-1}^g as the subgoal history embedding vector. We additionally encode the argument mask information $\texttt{arg}_i^M$ from the subgoal history in an integer-valued subgoal history tensor $\mathbf{H}_{k-1} \in \mathbb{N}^{K\times W\times L}$ where $[\mathbf{H}_{k-1}]_{(\tau,x,y)}$ is the number of times an interaction action type τ was performed at 2D position (x, y) in the birds-eye view:

$$[\mathbf{H}_{k-1}]_{(\tau,x,y)} = \sum_{\substack{i=0\ldots k-1 \\ \texttt{arg}_i^C=\tau}} \max_z([\texttt{arg}_i^M]_{(x,y,z)}) \tag{2}$$

Subgoal Prediction We concatenate the three representations $\mathbf{h}_{(t,k)} = [\phi^L; \phi_t^s; \phi_{k-1}^g]$. We use a densely connected multi-layer perceptron [48] to predict two distributions $P(\texttt{type}_k \mid \mathbf{h}_{(t,k)})$ and $P(\texttt{arg}_k^C \mid \texttt{type}_k, \mathbf{h}_{(t,k)})$, from which we sample a subgoal type $\texttt{type}_k$ and argument class $\texttt{arg}_k^C$.

The remaining component of the subgoal is the action argument mask $\texttt{arg}_k^M$. $[V_t^S]_{(\texttt{arg}_k^C)}$ is a voxel map that identifies all objects of class $\texttt{arg}_k^C$ in the semantic map V_t^S. We refine it to identify a single object instance. We compute a birds-eye view representation:

$$\mathbf{x}_t = [\textsc{Afford}(\hat{s}_t);\ \mathbf{H}_{k-1};\ \max_z([V_t^S]_{(\texttt{arg}_k^C)}) \odot \mathbb{1}_{\texttt{type}_k}] \tag{3}$$

where $\textsc{Afford}(\hat{s}_t)$ is a birds-eye view state feature map (Section 4.1) and $\mathbb{1}_{\texttt{type}_k}$ is a one-hot encoding of $\texttt{type}_k$.[5] Finally, we compute the 3D argument mask $\texttt{arg}_k^M \in [0,1]^{W\times L\times H}$:

$$\texttt{arg}_k^M = \textsc{Refiner}(\textsc{EgoTransform}(\mathbf{x}_t, P_t), \phi^L)\ , \tag{4}$$

where $\textsc{EgoTransform}(\mathbf{x}, P_t)$ transforms the map $\mathbf{x}$ to the agent egocentric pose P_t, $\textsc{Refiner}$ is a neural network based on the LingUNet architecture [27] , and ϕ^L is the language embedding. The refined $\texttt{arg}_k^M$ is a $[0,1]$-valued 3D mask that identifies the instance of the interaction argument object. If the object is believed to be unobserved, then $\texttt{arg}_k^M$ contains all zeroes. The controller output is the subgoal $g_k = (\texttt{type}_k, \texttt{arg}_k^C, \texttt{arg}_k^M)$.

4.4 Low-level Controller (π^L)

The low-level controller π^L is conditioned on the most recent subgoal $g_k = (\texttt{type}_k, \texttt{arg}_k^C, \texttt{arg}_k^M)$. At time t, it maps the state representation $\hat{s}_t$ to an action a_t. The low-level controller combines engineered and learned components. Appendix 9.3 provides the implementation details. The controller π^L invokes a set of procedures across multiple time steps t. The procedures are `NavigateTo`, `SampleExplorationPosition`, `SampleInteractionPose`, and `InteractMask`. Their invokation follows a pre-specified execution flow graph across multiple timesteps. First, we perform a 360° rotation to identify nearby voxels with an object of type $\texttt{arg}_k^C$, the action argument. If none are observed, we explore the environment by sampling a position $(x, y) =$

[5] $\odot$ denotes multiplication of a $X \times Y$ tensor with a K-dimensional vector to obtain a $K \times X \times Y$ tensor. $[\cdot;\cdot;\cdot]$ denotes channel-wise concatenation.

$\texttt{SampleExplorationPosition}(\hat{s}_t)$, navigating there using the procedure $\texttt{NavigateTo}(x, y, \hat{s}_t)$, and performing a 360° rotation. We repeat exploration until a voxel in V_t^S contains the class $\texttt{arg}_k^C$ with >50% probability. To interact with an object, we sample an interaction pose $(x, y, \omega_y, \omega_p) = \texttt{SampleInteractionPose}(\hat{s}_t, g_k)$, invoke $\texttt{NavigateTo}(x, y, \hat{s}_t)$ to reach the position (x,y), and then rotate according to yaw and pitch angles (ω_y, ω_p). Finally, we generate the egocentric interaction action mask $\texttt{mask}_t = \texttt{InteractMask}(\hat{s}_t, \texttt{arg}_k^M)$, and output the interaction action $(\texttt{type}_k, \texttt{mask}_t)$.

All procedures use the spatial representation $\hat{s}_t$. `NavigateTo` navigates to a goal position using a value iteration network (VIN) [49] that reasons over obstacle and observability maps from $\hat{s}_t$. `SampleExplorationPosition` samples positions on the boundary of observed space in $\hat{s}_t$. `SampleInteractionPose` uses a learned neural network NAVMODEL to predict a distributon of poses from which the interaction g_k will likely succeed. `InteractMask` uses the segmentation image I_t^S and the 3D argument mask $\texttt{arg}_t^M$ to compute the first-person mask of the target object.

5 Learning

Our policy contains four learned models: the segmentation and depth networks, the high-level controller π^H, and the navigation model NAVMODEL used by the low-level controller. We train all networks using supervised learning. We assume access to a training dataset $\mathcal{D} = \{(L^{(j)}, \Xi^{(j)})\}_{j=1}^{N_D}$ of natural language instructions $L^{(j)}$ paired with demonstration execution $\Xi^{(j)}$ in a set of seen environments. Each execution $\Xi^{(j)}$ is a sequence of states and actions $\langle s_0^{(j)}, a_0^{(j)}, \ldots, s_T^{(j)}, a_T^{(j)} \rangle$. We denote N_P the total number of states in $\mathcal{D}$, and N_G the total number of subgoals.

We process $\mathcal{D}$ into three datasets. The perception dataset $\mathcal{D}^P = \{([I]^{(i)}, [I^D]^{(i)}, [I^S]^{(i)}\}_{i=1}^{N_P}$ includes RGB images $[I]^{(i)}$ with ground truth depth $[I^D]^{(i)}$ and segmentation $[I^S]^{(i)}$. The subgoal dataset $\mathcal{D}^g = \{(L^{(i)}, \hat{s}_t^{(i)}, \langle g_j^{(i)} \rangle_{j=0}^k)\}_{i=1}^{N_G}$ contains natural language instructions $L^{(i)}$, state representations $\hat{s}_t^{(i)}$ at the start of k-th subgoal execution, and sequences of the first k subgoals $\langle g_j^{(i)} \rangle_{j=0}^k$. The navigation dataset $\mathcal{D}^N = \{(\hat{s}^{(i)}, g^{(i)}, P^{(i)})\}_{i=1}^{N_P}$ consists of state representations $\hat{s}^{(i)}$, subgoals $g^{(i)}$, and agent poses $P^{(i)}$ at the time of taking the interaction action corresponding to subgoal $g^{(i)}$. The state representations $\hat{s}^{(\cdot)}$ in datasets $\mathcal{D}^g$ and $\mathcal{D}^N$ are constructed using the observation model (Section 4.2), but using ground-truth depth and segmentation images.

We train the perception models on $\mathcal{D}^P$ and the π^H on $\mathcal{D}^g$ to predict the k-th subgoal by optimizing cross-entropy losses. We use $\mathcal{D}^N$ to train the navigation model NAVMODEL by optimizing a cross-entropy loss for positions and yaw angles, and an L2 loss for the pitch angle.

6 Experimental Setup

Environment, Data, and Evaluation We evaluate our approach on the ALFRED [11] benchmark of indoor scenes. The scenes are split to 108 training scenes, 88/4 validation seen/unseen scenes, and 107/8 test seen/unseen scenes. There are 21,023 training tasks, 820/821 validation seen/unseen tasks, and 1533/1529 test seen/unseen tasks. Each task is specified with a high-level natural language instruction. The goal of the agent is to map raw RGB observations to actions to complete the task. ALFRED also provides detailed low-level step-by-step instructions, which simplify the reasoning process. We do not use these instructions for training or evaluation. We collect a training dataset of language-demonstration pairs for learning (Section 5).

Completing a task requires satisfying a number of measurable goal conditions. Following the common evaluation [50, 51], we report two evaluation metrics. *Success rate* (SR) is the fraction of tasks for which all goal conditions were satisfied. *Goal condition rate* (GC) is the fraction of goal-conditions satisfied across all tasks.

Systems We compare our approach, the Hierarchical Language-conditioned Spatial Model (HLSM) to others on the ALFRED leaderboard that only use the high-level instructions. At the time of writing, the only such published approach is HiTUT [41], a hierarchical approach with a subgoal / action task decomposition, but that directly reasons over first-person images instead of using a spatial representation. We additionally compare to approaches that use the sequential step-by-step instructions, even though this evaluation is not fair to our method.

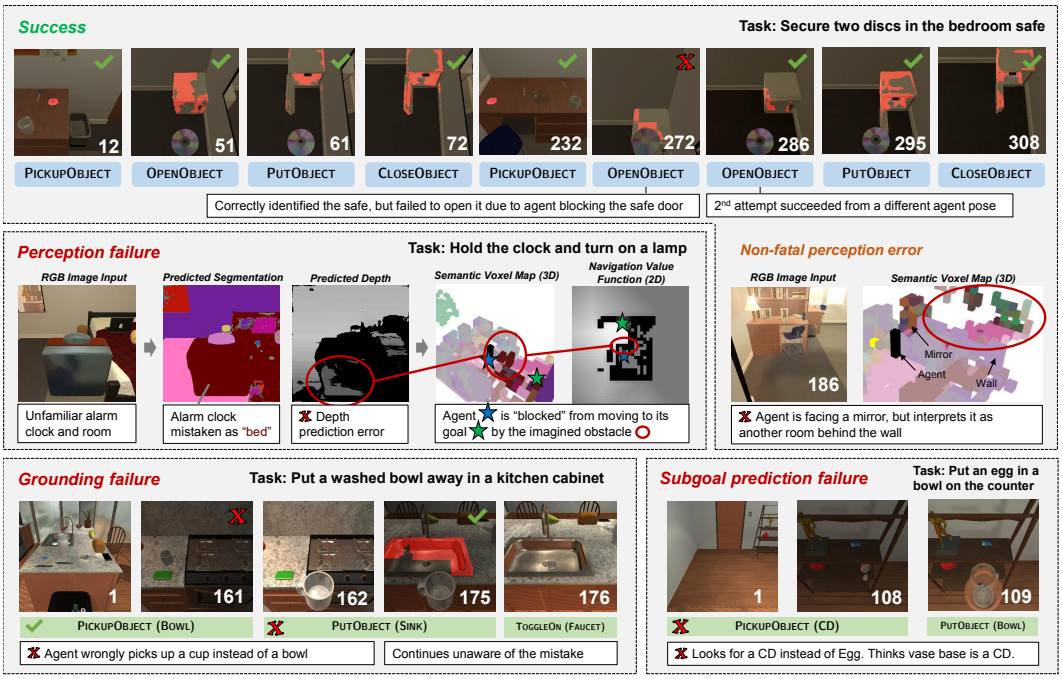

Figure 4: Qualitative results showcasing successes and failures of our approach. **Top row:** Snapshots of every interaction action taken during a successful task. Action argument masks are overlaid in red over the RGB images. The white numbers are timesteps. **Middle-right**: Illustration of a non-fatal perception error. **Middle-left**: Illustration of a fatal perception error. The agent incorrectly interprets the reflection on the alarm clock as an obstacle. The obstacle is represented in the semantic map, causing the agent (blue star) to believe that the path to the goal (green star) is blocked off. This is reflected in the *navigation value function* computed by the Value-Iteration Network (VIN) [49], where black cells are obstacles with value -1. White cell is the goal with value 1. **Bottom-left:** Grounding failure. The agent wrongly picks up the cup instead of a bowl. Predicted subgoals are shown in green. **Bottom-right:** High-level controller and percepton failure. π^H predicts the wrong subgoal argument class (CD instead of EGG). The segmentation model then mistakes the vase for a CD.

We perform ablations and study sensory oracles to understand areas of improvement. To study the observation model, we compare to using sensory oracles for ground truth depth, ground truth segmentation, and both. We report high-level controller ablations that remove the subgoal encoder, state representation encoder, and language encoder. We also study a low-level controller ablation that removes the exploration procedure.

7 Results

Table 1 shows the quantitative results. Our approach achieves state-of-the-art performance across both seen and unseen environments in the setting with only high-level instructions. We achieve 6.06% absolute (59.2% relative) improvement in SR on the *test unseen* split, and 6.7% absolute (36.4% relative) improvement in SR on the *test seen* split compared to HiTUT G-only.

Our approach performs competitively even when compared to approaches that also use the low-level step-by-step instructions, an evaluation setting not fair to our approach. We achieve 0.86% absolute (5.6% relative) improvement in SR on the *test unseen* split compared to ABP [12] that has not yet published their details. On the *test seen* split, our approach performs reasonably well, however ABP [12], E.T. [13], and LWIT [14] perform better.

Tables 2 and 3 show development results. Using ground truth depth (+ *gt depth*) provides 3.2%/1% absolute improvement in seen/unseen environments respectively. Using ground truth segmentation (+ *gt seg*) provides 16.3%/19.3% absolute improvement. Using both ground truth depth and segmentation (+ *gt depth, gt seg*) provides 19.1%/30.2% absolute improvement, and also narrows the gap between seen and unseen performance from 13.2% to 2.1%. We conclude that perception is the biggest generalization bottleneck that could be addressed with better depth and segmentation.

We report high-level controller π^H input encoder ablations. The poor performance without the *language encoder* reflects task difficulty. Sampling and executing random subgoals without language input seldom solves the task. Zeroing the input to the *subgoal history encoder* (but keeping position

Method	Seq. instr.	Test				Validation			
		Seen		*Unseen*		*Seen*		*Unseen*	
		SR	GC	SR	GC	SR	GC	SR	GC
SEQ2SEQ [11]	✓	3.98	9.42	0.39	7.03	3.70	10.00	0.00	6.90
MOCA [40]	✓	22.05	28.29	5.30	14.28	19.15	28.5	3.78	13.4
E.T. [13]	✓	28.77	36.47	5.04	15.01	33.78	42.48	3.17	13.12
E.T. [13] + *synth. data*	✓	38.42	45.44	8.57	18.6	46.59	52.82	7.32	20.87
LWIT [52]	✓	30.92	45.44	9.42	20.91	33.70	43.10	9.70	23.10
HITUT[41]	✓	21.27	29.97	13.87	20.31	25.24	34.85	12.44	23.71
ABP [12]	✓	44.55	51.13	15.43	24.76	-	-	-	-
HITUT G-only[41]	x	18.41	25.27	10.23	20.27	13.63	21.11	11.12	17.89
LAV [53][2]	x	13.35	23.21	6.38	17.27	-	-	-	-
HLSM (Ours)	x	**25.11**	**35.79**	**16.29**	**27.24**	**25.00**	**36.37**	**11.80**	**24.70**

Table 1: Test results. Test seen/unseen and validation seen/unseen splits. Top section approaches use sequential step-by-step instructions. The bottom section uses only high-level instructions.

Method	Validation			
	Seen		*Unseen*	
	SR	GC	SR	GC
HLSM	**25.0**	**36.4**	11.8	24.7
+ *gt depth*	28.2	36.7	12.8	26.2
+ *gt depth, gt seg.*	44.1	54.5	42.0	53.2
+ *gt seg.*	41.3	51.6	31.1	43.2
w/o language enc.	0.6	8.3	0.2	7.6
w/o subg. hist. enc.	23.3	31.8	10.0	22.8
w/o state repr enc.	23.8	34.3	**12.9**	**25.7**
w/o exploration	24.9	35.9	11.3	23.7

Table 2: Development results on validation split. Performance of our full approach, with perception oracles, a perception ablation, π^H ablations, and π^L ablations

Task Type	Validation			
	Seen		*Unseen*	
	SR	GC	SR	GC
Examine	37.2	55.9	22.5	45.7
Pick & Place	42.3	42.3	28.0	28.0
Stack & Place	8.7	23.1	1.8	7.0
Clean & Place	25.0	43.8	10.6	25.3
Cool & Place	11.9	29.9	3.7	22.9
Heat & Place	12.1	29.1	0.0	14.1
Pick 2 & Place	35.5	50.8	12.3	29.2
Overall	**25.0**	**36.4**	**11.8**	**24.7**

Table 3: Performance breakdown per task type on the validation split.

encodings) reduces performance by only 1.7%/1.8%, showing that knowing the index of the current subgoal in addition to the state representation is usually sufficient. Removing *state representation encoder* reduces SR by 1.7% in seen scenes, but it *increases* SR by 1.1% in unseen scenes. In fact, this ablation achieves state-of-the-art validation-unseen performance of SR 12.9%, even comparing to approaches in the sequential instruction setting. This is likely explained by perception errors that affect state representations in unseen environments more than seen. Removing the *exploration* behavior from the low-level controller reduces SR by 0.1%/0.5% on seen/unseen scenes. Therefore the random exploration is not very effective.

Finally, we illustrate model behavior with qualitative results in Figure 4, showing both successes and common failures. The main failures in *valid unseen* scenes are due to (1) perception errors that result in missing or extraneous obstacles or picking up wrong objects, (2) insufficiency of random exploration (e.g. not searching inside cabinets), (3) navigation model errors (e.g. blocking objects from opening), (4) subgoal prediction errors (e.g. picking up wrong objects), and (5) lack of state-aware multi-step planning and backtracking. More qualitative results are available in Appendix 9.6.

8 Discussion

We have shown that a persistent spatial semantic representation enables a hierarchical model to achieve state-of-the-art performance on a challenging virtual mobile manipulation instruction-following task. Our focus is on the representation and the hierarchical formulation, not on the specific low-level controller design. Improving the low-level controller is an important direction for future work emphasized by the advantage our formulation shows. For example, learning exploration or navigation models instead of using hand-coded primitives could improve robustness. In terms of perception, better segmentation and depth prediction in unseen environments is an area of possible improvement. Finally, incorporating reinforcement learning to train the high-level controller, low-level controller, and observation model jointly could improve robustness of the approach.

References

[1] S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. Gopal Banerjee, S. Teller, and N. Roy. "Approaching the Symbol Grounding Problem with Probabilistic Graphical Models. *AI Magazine*, 2011.

[2] C. Matuszek, E. Herbst, L. Zettlemoyer, and D. Fox. Learning to parse natural language commands to a robot control system. In *ISER*, 2012.

[3] D. K. Misra, J. Sung, K. Lee, and A. Saxena. Tell me dave: Context-sensitive grounding of natural language to mobile manipulation instructions. In *RSS*, 2014.

[4] J. Thomason, S. Zhang, R. J. Mooney, and P. Stone. Learning to interpret natural language commands through human-robot dialog. In *IJCAI*, 2015.

[5] D. Misra, J. Langford, and Y. Artzi. Mapping instructions and visual observations to actions with reinforcement learning. In *EMNLP*, 2017.

[6] D. Nyga, S. Roy, R. Paul, D. Park, M. Pomarlan, M. Beetz, and N. Roy. Grounding robot plans from natural language instructions with incomplete world knowledge. In *CoRL*, 2018.

[7] V. Blukis, N. Brukhim, A. Bennet, R. Knepper, and Y. Artzi. Following high-level navigation instructions on a simulated quadcopter with imitation learning. In *RSS*, 2018.

[8] V. Blukis, D. Misra, R. A. Knepper, and Y. Artzi. Mapping navigation instructions to continuous control actions with position-visitation prediction. In *CoRL*, 2018.

[9] V. Blukis, Y. Terme, E. Niklasson, R. A. Knepper, and Y. Artzi. Learning to map natural language instructions to physical quadcopter control using simulated flight. In *CoRL*, 2019.

[10] V. Blukis, R. A. Knepper, and Y. Artzi. Few-shot object grounding and mapping for natural language robot instruction following. In *Proceedings of the Conference on Robot Learning*, 2020.

[11] M. Shridhar, J. Thomason, D. Gordon, Y. Bisk, W. Han, R. Mottaghi, L. Zettlemoyer, and D. Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *CVPR*, 2020.

[12] B. Kim, S. Bhambri, K. P. Singh, R. Mottaghi, and J. Choi. Abp, alfred leaderboard, may 10th 2021. https://leaderboard.allenai.org/alfred/submission/c2t70j37q4q5ci4so89g. Accessed: June 16th, 2021.

[13] A. Pashevich, C. Schmid, and C. Sun. Episodic Transformer for Vision-and-Language Navigation, 2021.

[14] Anonymous. Lwit, alfred leaderboard, may 10th 2021. https://leaderboard.allenai.org/alfred/submission/bvppcin94ro4j7j0jqlg. Accessed: May 25th, 2021.

[15] I. Kostavelis and A. Gasteratos. Semantic mapping for mobile robotics tasks: A survey. *Robotics and Autonomous Systems*, 66:86–103, 2015.

[16] T. Kollar, S. Tellex, D. Roy, and N. Roy. Toward understanding natural language directions. In *HRI*, 2010.

[17] C. Matuszek, N. FitzGerald, L. Zettlemoyer, L. Bo, and D. Fox. A Joint Model of Language and Perception for Grounded Attribute Learning. In *ICML*, 2012.

[18] D. Arumugam, S. Karamcheti, N. Gopalan, L. L. Wong, and S. Tellex. Accurately and efficiently interpreting human-robot instructions of varying granularities. In *RSS*, 2017.

[19] R. A. Knepper, S. Tellex, A. Li, N. Roy, and D. Rus. Recovering from Failure by Asking for Help. *Autonomous Robots*, 2015.

[20] T. Brick and M. Scheutz. Incremental natural language processing for hri. In *HRI*, 2007.

[21] A. Suhr, C. Yan, J. Schluger, S. Yu, H. Khader, M. Mouallem, I. Zhang, and Y. Artzi. Executing instructions in situated collaborative interactions. In *EMNLP*, 2019.

[22] K. M. Hermann, F. Hill, S. Green, F. Wang, R. Faulkner, H. Soyer, D. Szepesvari, W. Czarnecki, M. Jaderberg, D. Teplyashin, et al. Grounded language learning in a simulated 3d world. *arXiv preprint arXiv:1706.06551*, 2017.

[23] D. S. Chaplot, K. M. Sathyendra, R. K. Pasumarthi, D. Rajagopal, and R. Salakhutdinov. Gated-attention architectures for task-oriented language grounding. *AAAI*, 2018.

[24] C. Paxton, Y. Bisk, J. Thomason, A. Byravan, and D. Foxl. Prospection: Interpretable plans from language by predicting the future. In *ICRA*, pages 6942–6948. IEEE, 2019.

[25] 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 *CVPR*, 2018.

[26] 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. In *NeurIPS*, 2018.

[27] D. Misra, A. Bennett, V. Blukis, E. Niklasson, M. Shatkin, and Y. Artzi. Mapping instructions to actions in 3D environments with visual goal prediction. In *EMNLP*, 2018.

[28] C.-Y. Ma, J. Lu, Z. Wu, G. AlRegib, Z. Kira, R. Socher, and C. Xiong. Self-monitoring navigation agent via auxiliary progress estimation. In *ICLR*, 2019.

[29] H. Tan, L. Yu, and M. Bansal. Learning to navigate unseen environments: Back translation with environmental dropout. In *NAACL-HLT*, 2019.

[30] H. Chen, A. Suhr, D. Misra, and Y. Artzi. Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *CVPR*, 2019.

[31] D. K. Misra, K. Tao, P. Liang, and A. Saxena. Environment-driven lexicon induction for high-level instructions. In *ACL*, 2015.

[32] M. MacMahon, B. Stankiewicz, and B. Kuipers. Walk the talk: Connecting language, knowledge, and action in route instructions. In *AAAI*, 2006.

[33] S. R. K. Branavan, L. S. Zettlemoyer, and R. Barzilay. Reading between the lines: Learning to map high-level instructions to commands. In *ACL*, 2010.

[34] F. Duvallet, T. Kollar, and A. Stentz. Imitation learning for natural language direction following through unknown environments. In *ICRA*, 2013.

[35] M. R. Walter, S. Hemachandra, B. Homberg, S. Tellex, and S. Teller. Learning Semantic Maps from Natural Language Descriptions. In *RSS*, 2013.

[36] Y. Artzi and L. Zettlemoyer. Weakly supervised learning of semantic parsers for mapping instructions to actions. *TACL*, 2013.

[37] S. Hemachandra, F. Duvallet, T. M. Howard, N. Roy, A. Stentz, and M. R. Walter. Learning models for following natural language directions in unknown environments. In *ICRA*, 2015.

[38] N. Gopalan, D. Arumugam, L. L. Wong, and S. Tellex. Sequence-to-sequence language grounding of non-markovian task specifications. In *RSS*, 2018.

[39] E. C. Williams, N. Gopalan, M. Rhee, and S. Tellex. Learning to parse natural language to grounded reward functions with weak supervision. In *ICRA*, 2018.

[40] K. P. Singh, S. Bhambri, B. Kim, R. Mottaghi, and J. Choi. Moca: A modular object-centric approach for interactive instruction following. *arXiv preprint arXiv:2012.03208*, 2020.

[41] Y. Zhang and J. Chai. Hierarchical task learning from language instructions with unified transformers and self-monitoring. *ACL Findings*, 2021.

[42] D. Gordon, A. Kembhavi, M. Rastegari, J. Redmon, D. Fox, and A. Farhadi. Iqa: Visual question answering in interactive environments. In *CVPR*, 2018.

[43] P. Anderson, A. Shrivastava, D. Parikh, D. Batra, and S. Lee. Chasing ghosts: Instruction following as bayesian state tracking. In *NeurIPS*, 2019.

[44] J. McCormac, A. Handa, S. Leutenegger, and A. J. Davison. Scenenet rgb-d: Can 5m synthetic images beat generic imagenet pre-training on indoor segmentation? In *ICCV*, 2017.

[45] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.

[46] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

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

[48] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

[49] A. Tamar, Y. Wu, G. Thomas, S. Levine, and P. Abbeel. Value iteration networks. In *NeurIPS*, 2016.

[50] M. Shridhar and D. Hsu. Interactive visual grounding of referring expressions for human-robot interaction. In *RSS*, 2018.

[51] M. Shridhar, J. Thomason, D. Gordon, Y. Bisk, W. Han, R. Mottaghi, L. Zettlemoyer, and D. Fox. Alfred leaderboard. https://leaderboard.allenai.org/alfred/. Accessed: June 16th, 2021.

[52] V.-Q. Nguyen, M. Suganuma, and T. Okatani. Look wide and interpret twice: Improving performance on interactive instruction-following tasks. In *IJCAI*, 2021.

[53] K. Nottingham, L. Liang, D. Shin, C. C. Fowlkes, R. Fox, and S. Singh. Lav, alfred leaderboard, may 10th 2021. https://leaderboard.allenai.org/alfred/submission/c2cm7eranqs9puf9uvjg. Accessed: May 25th, 2021.

[54] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, 2015.