One Step at a Time: Long-Horizon Vision-and-Language Navigation with Milestones

Chan Hee Song[1] Jihyung Kil[1] Tai-Yu Pan[1] Brian M. Sadler[2] Wei-Lun Chao[1]
Yu Su[1]
[1]The Ohio State University [2]U.S. Army Research Laboratory
{song.1855, kil.5, pan.667, chao.209, su.809}@osu.edu
brian.m.sadler6.civ@army.mil

Abstract

*We study the problem of developing autonomous agents that can follow human instructions to infer and perform a sequence of actions to complete the underlying task. Significant progress has been made in recent years, especially for tasks with short horizons. However, when it comes to long-horizon tasks with extended sequences of actions, an agent can easily ignore some instructions or get stuck in the middle of the long instructions and eventually fail the task. To address this challenge, we propose a model-agnostic milestone-based task tracker (*M-TRACK*) to guide the agent and monitor its progress. Specifically, we propose a milestone builder that tags the instructions with navigation and interaction milestones which the agent needs to complete step by step, and a milestone checker that systemically checks the agent's progress in its current milestone and determines when to proceed to the next. On the challenging ALFRED dataset, our* M-TRACK *leads to a notable* 33% *and* 52% *relative improvement in unseen success rate over two competitive base models.*

1. Introduction

As autonomous agents (*e.g.*, robots) become more integrated into our daily life, it is increasingly important to develop autonomous agents that can understand natural language commands and carry out the corresponding tasks. To facilitate such a goal, various benchmarks have been proposed in the realm of robot instruction following such as vision-and-language navigation (VLN) [1, 3, 4, 9, 13, 28, 29, 38, 44], together with a number of novel algorithms that consistently push forward the state of the art [20, 21, 34, 37]. Specifically, to succeed in VLN, an agent must comprehend the language instruction, ground it into the partially-observable environment with only visual perception, and plan and perform navigation and interaction actions in the environment to complete the task.

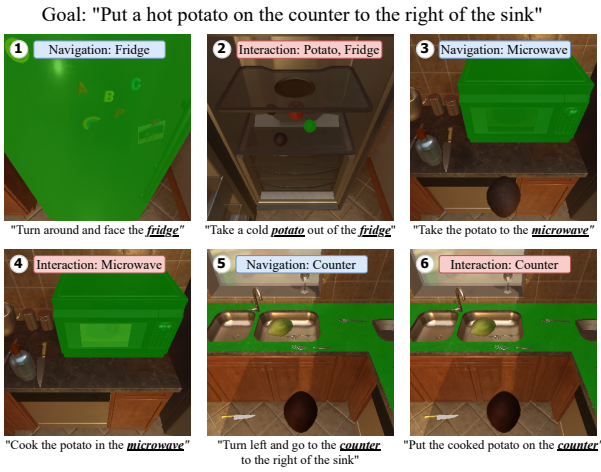

Figure 1. **Illustration of our M-TRACK approach**. We show an ALFRED task [30], which consists of an overall goal (text on the top) and six subtasks (text below each image). The blue/red text box within each image is our extracted navigation/interaction milestones from the subtask instructions. An agent needs to reach the milestone of the current subtask (*e.g.*, reaching proximity to the target object for navigation milestones, or having interacted with the target objects for interaction milestone; green masks for target objects) before it can proceed to the next subtask.

One critical challenge in VLN arises when the task horizon becomes substantially longer [30]. That is, a task is so complex that it essentially consists of multiple "subtasks" that need to be completed *sequentially* to fulfill the whole task. For example, in Figure 1 the task *"put a hot potato on the counter to the right of the sink"* can be decomposed into six subtasks. Moreover, the subtask *"heat the potato"* must be carried out before the subtask *"put the potato on the counter"*; otherwise, the final task is doomed to fail no matter how accurate the subsequent planning is. Such a sequential dependency requires the agent to closely monitor its progress and ensure it is staying on the right track when carrying out a long-horizon task.

At first glance, this challenge may seem trivial if the language instruction is detailed enough (like in Figure 1), such that it already defines the subtasks and their order. However, as shown in the literature [2, 14, 20, 31, 37, 42] and our experiments, an agent fed with detailed instructions still frequently skips subtasks, or wanders around within a subtask even when it is already completed. In essence, what an agent truly struggles with is the lack of awareness of *where it currently is in the long subtask sequence and how much progress it has made within a subtask*.

To address this issue, we propose to equip VLN agents with an *explicit task tracker*, which keeps track of the agent's progress within a subtask and guides it for when to move on to the next. Concretely, we propose the concept of *milestone*, which renders the necessary condition of completing a subtask. Namely, for a subtask to be considered as completed, the milestone must be reached. Take the subtask *"take a cold potato out of the fridge"* in Figure 1 as example. To complete it, the necessary condition is that the agent must see the potato and the fridge, be close enough to them, and perform an interaction action with the potato. We argue that by explicitly extracting such milestones from the instructions and *grounding* them to the environment state, we can systematically determine if the agent should continue working on the current subtask or proceed to the next.

To this end, we propose the *milestone-based task tracker* (M-TRACK), which consists of two components: *milestone builder* and *milestone checker*. The milestone builder extracts the milestone (*i.e.*, the necessary completion condition) of each subtask from the corresponding language instruction. We model it as a named entity recognition problem and train a BERT-CRF tagger [6, 32] to accurately extract both the target objects and their action type (*i.e.*, navigation or interaction). The milestone checker then tries to *ground* (*i.e.*, identify and localize) the extracted target objects in the perceived environment using an object detection model [10] and checks if the agent is close enough to them and/or is about to interact with them — to decide if the agent is completing the current subtask and ready to move on. It is worth noting that our M-TRACK only needs to access the language instructions, the visual input to the agent, and the agent's action, not any internal states of the agent. Thus, it is *model-agnostic* and can be easily integrated with any agent model with minimal changes.

How can M-TRACK *interacts with the agent to affect its action (*e.g.*, to not skip a subtask)?* We propose two simple yet effective ways. First, at any time step, we feed the agent with only the part of the instructions that corresponds to the current subtask determined by the milestone tracker. This explicitly guides the agent to focus on the current subtask. Second, and more importantly, we apply the milestone checker *proactively* — before the agent executes its predicted action — to reject actions that will lead to subtask failures. For instance, we reject the action of taking a *"sponge"* if the milestone object is *"fork"* (Figure 2).

We validate M-TRACK on ALFRED [30], a recently released large-scale VLN dataset for common household tasks. The tasks in ALFRED are considered long-horizon because on average each task needs 50 actions to complete. In contrast, another popular dataset R2R [1] needs only 5. We integrate M-TRACK into two baseline VLN models LSTM [30] and VLN⟳BERT [12], and demonstrate notable and consistent performance gains. When tested in seen environments, M-TRACK leads to 16%–57% relative improvement in success rate. In more challenging unseen environments, the relative gain increases to 33%–52%. Our ablation studies and qualitative results further verify that the improvement indeed comes from agents able to better follow the sequence of subtasks and stay on the right track.

2. Related Work

VLN datasets. Significant efforts have been devoted to creating simulated environments and datasets for VLN, where a virtual agent has egocentric perception of the environment and takes actions to navigate in it [1,3,4,9,13,28,29,38,44]. However, most datasets do not consider interaction actions with objects, significantly limiting the complexity of tasks that an agent can perform. The recent ALFRED dataset [30] is among the first to provide tasks that involve both navigation and interaction actions, providing a more challenging benchmark with much longer task horizons.

VLN models. Most early VLN models follow an LSTM-based sequence-to-sequence architecture, taking language and visual sequences as input and predicting a sequence of actions [1, 8, 16, 20, 30, 34]. Because of the recent success of the Transformer [35] in vision tasks, Transformer-based models are increasingly adopted for VLN [12,17,22,27,33]. Our M-TRACK is model-agnostic and is compatible with models of both types (cf. §4.3).

Natural language instructions. ALFRED provides each task with both a high-level (*i.e.*, goal) instruction and more detailed low-level instructions. Most previous studies train the agent with the whole instruction (*i.e.*, concatenation of the high-level and low-level instructions) at each time step [15, 27, 30, 31, 33]. However, for long-horizon tasks like those in ALFRED, the low-level instructions can be quite long (six sentences on average). An agent fed with the whole instruction thus could have difficulty digesting the long instruction and easily lose track of the progress. M-TRACK helps agents focus their attention on the most pertinent instruction and reduce distraction.

Step-by-step language guidance. To address the issues with long instructions, learning low-level instructions step by step has been explored in several prior studies [5, 11, 23, 42, 43]. BabyWalk [43] learns the low-level instruction step

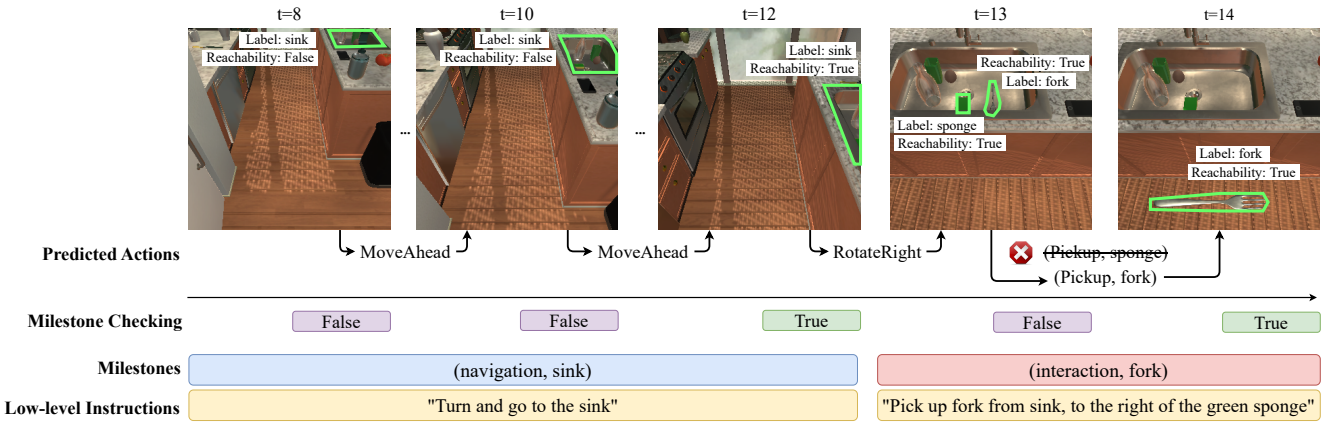

Figure 2. **Overview of the milestone checking process**. Milestones are extracted from the current low-level instruction by our milestone builder (§4.1.1). After an action is predicted, our milestone checker (§4.1.2) examines, based on objects with reachability information (text in images) from its object detector, if the resultant state satisfies the milestone. Only when the milestone is satisfied, the next low-level instruction is provided to the agent. The agent is prevented from picking up a wrong object (sponge) by our proactive checking (§4.2).

by step using curriculum learning. HiTUT [42] decomposes the whole instruction into hierarchical sub-problems and learns sequentially with a hierarchical task network. Concurrent to this work, FILM [23] decomposes the instruction into subtasks and learns them sequentially with the help of a semantic map. M-TRACK shares a similar rationale. However, M-TRACK is notably different from existing methods, especially regarding when to feed the next low-level instruction during *test time*. First, M-TRACK explicitly and systematically checks the agent's progress, during both training and test time, by 1) defining the completion condition, *i.e.*, the milestone, of each subtask and 2) verifying the milestone by grounding it into the environment via a visual object detector. In contrast, existing methods either train a binary classifier to determine subtask completion [4], or simply set an upper bound for the number of actions to execute within each subtask [42, 43], or only checks if the agent needs to stop using a separate module [40]. As will be seen in §5, M-TRACK notably outperforms these methods in tracking the agent's progress and feeding the right instruction. Second, M-TRACK also proactively guides the agent for better action prediction, creating another gain in performance (cf. §4.2). Finally, M-TRACK is not embedded in any specific VLN model; it is model-agnostic and can be easily integrated into different VLN models (cf. §4.3).

3. VLN Background

A VLN task is generally defined as follows: given a language instruction I, an agent needs to infer and perform a sequence of actions $\{a_0, a_1, \cdots, a_t, \cdots\}$ in the environment E to complete the task. In datasets like ALFRED [30], the instruction I is composed of a high-level instruction I_H and a list of low-level instructions I_L, as exemplified in Figure 1. A VLN task can thus be represented by a tuple (I, E, G), in which G is the goal test of the task.

For an agent to perform the task, it will be placed in the environment E and have a certain pose at time step t, from which it can receive a visual input v_t. Based on v_t and the instruction I, the agent then predicts an action a_t, which can either be a navigation action that changes the agent's pose (*e.g.*, MoveAhead) or an interaction action that interacts with the environment (*e.g.*, PickupObject). The agent also needs to predict a binary mask for the target object if it predicts an interaction action. Both types of actions can potentially change the visual input v_{t+1} of the next time step. The agent will stop when it believes the task has been completed. The final state of the environment is then compared with the goal state G to determine task completion.

Following ALFRED, we discretize an agent's action space into 5 navigation action (MoveAhead, RotateRight, RotateLeft, LookUp, and LookDown), 7 interaction actions (PickupObject, PutObject, OpenObject, CloseObject, ToggleOnObject, ToggleOffObject, and SliceObject), and 1 stop action (Stop).

Agent model. Without loss of generality, we define an agent model as $a_t = f(v_t, I_t, h_t)$, where h_t is the memory from the previous time steps (*e.g.*, the hidden state of an LSTM). a_t is a tuple (action, object mask); the mask is null for stop and navigation actions. I_t is the instruction input at time t, which can be the entire I or a portion of it.

4. Milestone-based Task Tracker (M-TRACK)

For long-horizon VLN tasks, an agent needs to complete multiple subtasks, usually in a specific order, to complete the whole task. More specifically, each low-level instruction in I_L can be seen as a subtask. Agents then have to decide, often implicitly, which subtask it is doing at each time step and when to move on to the next subtask, which itself is a challenging problem for the agent. To address that, we introduce an auxiliary module, *milestone-based task tracker*

(M-TRACK), to explicitly and interactively guide the agent to make such decisions (see Figure 2 for an overview). Next, we first introduce the design of M-TRACK (§4.1), followed by how to integrate it with agent models (§4.2). We then introduce two base agent models (§4.3) and how to train the base models with reinforcement learning (§4.4).

4.1. Design of M-TRACK

The core functionality of M-TRACK is to decide when an agent should move on to the next subtask. On the surface, this may be done simply by training a (binary) classifier, which takes all the language/visual signals as input. Doing so, however, does not exploit the fact that the (sub)tasks are compositional, composed of entities (*e.g.*, objects) that are identifiable and localizable both in the environment and in the instruction. Leveraging the compositional nature of the (sub)tasks has multiple advantages. First, it reduces the input space for making the decision from the space of language/visual signals to that of discrete entities. Second, it makes the decision rule systematic and explainable: we can make the decision by directly comparing the entities detected in both modalities. Both of them could improve the generalizability of the decision function.

We design M-TRACK to explicitly consider the compositional nature of (sub)tasks. Specifically, we introduce the concept of *milestone*, which is the *necessary condition for completing a subtask*, *i.e.*, an agent must reach the milestone in order for the corresponding subtask to be considered as completed. For example, if the subtask is *"move to the mug"*, then the agent must **navigate** to the **mug**, **see** it, and be **close enough** to it. If the subtask is *"pick up the mug"*, then the agent must **see** the **mug**, be **close enough** to it so that it can then **interact** with it. These two examples render the key ingredients of a milestone, which are its **target entities** and its **type** (navigation or interaction). Meanwhile, we say an agent has reached a milestone only when it can perceive (see) the target entities, is already close to them, and is doing the right type of action with them.

To this end, we represent a milestone by a tuple (type, target), and decompose our M-TRACK into two components: 1) a *milestone builder* which constructs milestones from the low-level instructions I_L, and 2) a *milestone checker* that checks if a milestone has been reached by an agent.

4.1.1 Milestone Builder

We generate the milestone of a subtask according to its corresponding low-level instruction in I_L using named entity recognition [6]. For example, given an instruction *"Turn to the left and face the toilet"*, the milestone builder should output the tag (navigation, toilet). For the instruction *"Pick the soap up from the back of the toilet"*, the milestone builder should output (interaction, soap).

Target Type	Val Seen	Val Unseen
Navigation	90.16	90.62
Interaction	96.85	97.17

Table 1. **F1 score of milestone builder on ALFRED validation.**

For an interaction milestone, it should contain the target objects that the agent is going to newly interact with in the current subtask. For instance, if the subtask is *"Put down the potato on the counter"* (Figure 1), the agent is supposed to already be holding a potato (from previous subtasks). Thus, *"potato"* is not a milestone target for the current subtask but *"counter"* should be. For a subtask that has multiple objects to be interacted with, the builder is designed to tag all of them. For instance, in the subtask *"Grab a potato from the fridge"* (Figure 1), the agent needs to 1) open the fridge, 2) pick up the potato, and 3) close the fridge. In this case, the builder tags both the potato and the fridge as the targets for an interaction milestone. In cases that the builder does not extract any target from the current subtask, it will merge the current subtask with the next one and use the milestone extracted from the next subtask.

Without loss of generality, we adopt a BERT-CRF model [6, 32] for the milestone builder, and train it with data derived from the ALFRED training data. Training data is prepared using the metadata from the ALFRED simulator. More details are in the supplementary materials. We show that our milestone builder reaches a fairly high F1 score (see Table 1). More analysis will be discussed in §5.3.2.

4.1.2 Milestone Checker

We introduce a milestone checker that determines if an agent has reached a milestone (see Figure 2). Specifically, we design it to be *explicit*: we directly estimate the state of the agent/environment from the visual input and compare it with the milestone. A navigation milestone is reached if the target object is detected in the visual input and located within a reachable distance to the agent (1.5 meters in ALFRED). An interaction milestone is reached with an extra condition: the agent has to interact with the target.

State estimation. We train an object detector using data from the ALFRED simulator that can not only localize and identify all 116 ALFRED object classes but also estimate their reachability (*i.e.*, within 1.5m or not). We build upon the Mask R-CNN model [10] and introduce an additional binary classification head for the reachability of each detected object. The ground-truth labels for reachability are obtained from the ALFRED simulator for training.

Milestone checking. As mentioned earlier, to reach either a navigation or an interaction milestone, the target objects must be detected and located within a reachable distance. To check this, we compare the target object names, which

are extracted from the language instruction (*e.g.*, *"kitchen island"*), to the class labels (*e.g.*, countertop) of the objects detected by Mask R-CNN, essentially a symbol grounding task. We only consider the detected objects that are estimated to be reachable. We apply an off-the-shelf word similarity tool based on Wordnet [7] with WUP [39] similarity from NLTK [19] to match the target names with the object labels. The reachable object whose label has the highest similarity (above a threshold) to a milestone target is considered as the *grounded instance* of that target; the target is then marked as a success.

For interaction milestones, we need to further check if the agent is interacting/has interacted with the target. As defined in §3, an interaction action is a tuple of (action, object mask); the object mask is simply a binary map over the input image. To determine if the agent's action is for the milestone target, we calculate the intersection-over-union (IoU) score between the object mask and the milestone target (provided by Mask R-CNN): if the IoU score is over a certain threshold (0.5), it is considered matched with the target object of the milestone. For an interaction milestone with multiple targets, the agent has to perform multiple interaction actions to interact with all of them. We keep a checklist of all the milestone targets. The milestone is reached after all the targets have been interacted with.

4.2. Planning with M-TRACK

The discussion of M-TRACK so far is detached from the agent. The next question is, *how can* M-TRACK *affect an agent's actions,* e.g.*, to prevent it from skipping a subtask?* We propose two simple yet effective ways. First, at any time step, we feed the agent with only the instruction of the current subtask determined by M-TRACK. This explicitly guides the agent to focus on the current subtask. Specifically in ALFRED, we feed the concatenation of I_H and the one sentence in I_L for the current subtask as opposed to the entire I_L. We do so starting from the beginning of a task, when the first sentence of I_L is guaranteed to be the first subtask. We then proceed to the next sentence only after the current subtask is marked as completed by M-TRACK. The use of M-TRACK frees the agent from solely relying on its internal mechanism like attention and hidden states to decide subtask switching.

Second, we apply the milestone checker *proactively* for interaction milestones — before the agent executes its predicted action. This can prevent an agent from interacting with a wrong object, as opposed to trying to correct the mistake after it has happened. For example, if the milestone is (interaction, fork) but the agent's predicted binary mask for interaction does not overlap with the grounded instance of fork in the image, M-TRACK will reject the agent's action by asking it to select another (action, object mask) tuple (Figure 2). This saves the agent from having to generate an action sequence for recovery, for example, to put the incorrectly picked-up object back down.

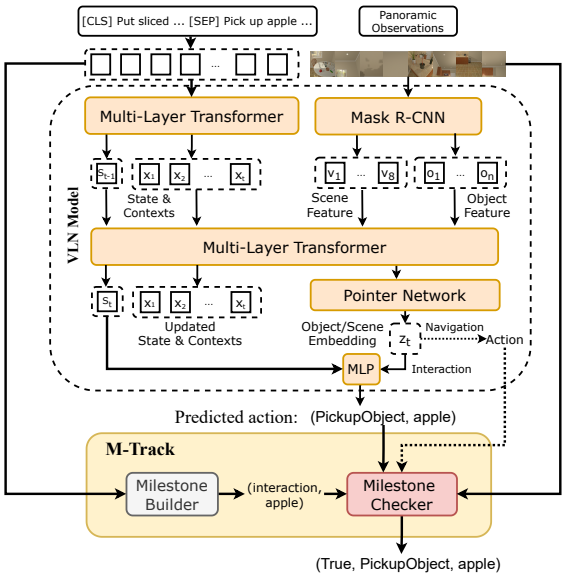

Figure 3. **Architecture of VLN⟳BERT with M-TRACK.**

In our implementation, if the first interaction action is rejected, we move on to the next action in the agent's top N list (*e.g.*, from a softmax classifier). We iterate over the N actions until we find an interaction action whose mask matches with the milestone target or we find a navigation action instead (*e.g.*, when the right object is not in sight). If none of those happens, the agent will take its top ranked navigation action. We set N to be 5 in the experiments.

4.3. Agent Models

4.3.1 VLN⟳BERT Baseline

Recently, Transformer-based models are becoming increasingly popular for VLN tasks [27, 27, 33, 42]. Following this line of work, we build upon the VLN⟳BERT [12] model which introduces the concept of recurrent state vector into the Transformer architecture. Since VLN⟳BERT was designed for the R2R dataset, which contains mostly short-horizon navigation tasks, we adapt it for ALFRED with a series of modifications. Input-wise, we utilize a pre-trained vision encoder[1] to extract a scene feature from 8 panoramic views and also object features from each view as our visual input. For action prediction, unlike VLN⟳BERT that only deals with navigation actions, we employ a pointer network [36] to choose between navigation, interaction, and stop actions: if the pointer network chooses a scene feature, agent outputs the navigation action needed to navigate to that scene; if it chooses an object feature, agent outputs the mask for that object, and additionally use an MLP to pre-

[1]For simplicity, we use the same Mask R-CNN model that is used in our milestone checker, but it is not necessary.

dict the interaction action type; if it chooses a stop feature (added to the list of visual features as an all-zero vector), agent outputs Stop. The MLP takes the concatenation of the chosen object feature and the updated state embedding as input. The architecture as well as its integration with M-TRACK is illustrated in Figure 3, and more implementation details are provided in the supplementary materials.

4.3.2 LSTM Baseline

To further show the model-agnostic nature of our M-TRACK, we use the LSTM baseline introduced in ALFRED [30], and extend the architecture with the same pre-trained vision encoder used in VLN↻BERT. Furthermore, to leverage the power of the pre-trained vision encoder, we follow [27, 31] and ask our agent to select an object from the detected objects instead of directly predicting a binary mask. The corresponding pixel mask is retrieved from the selected object. Refer to supplementary materials for details.

4.4. Learning

As shown in the ALFRED paper [30], base models like the LSTM performs rather poorly on ALFRED when simply trained with behavior cloning. Prior studies on other VLN tasks have demonstrated the importance of reinforcement learning (RL) [12, 34, 43], but its effectiveness has not been validated on ALFRED. We train the models with a combination of behavior cloning (using the cross-entropy loss between the predicted action sequence and the ground truth), object feature selection loss (for interaction actions), and RL. We apply the A2C algorithm [24] which, at time t, samples an action a_t according to agent's predicted log probability distribution $\log(p_t^a)$, and measures the advantage for that action adv_t with a critic network and a reward. We consider four different types of reward: 1) the straight line distance between the agent and the current navigation/interaction target, 2) the interaction action matching with the ground-truth interaction action which we can compute from the environment state, 3) the visibility of the target, *i.e.*, whether the target is reachable (within 1.5m in ALFRED) and is in sight by the agent, and 4) the final task success. Following VLNBERT [12], we combine behavior cloning loss, object feature selection loss for object selection and RL loss during all training iterations.

5. Evaluation

5.1. Experimental Setup

ALFRED. We validate our approach on the ALFRED [30] dataset which evaluates an agent's language-guided navigation and interaction abilities for common household tasks. ALFRED consists of 8055 expert demonstrations annotated with 25 743 natural language instructions. The standard training/validation/test splits contain 21 023/1641/3062 examples, respectively. The validation and test sets are further split into 1) a *seen* set where the environments have been seen during training and 2) an *unseen* set that contains new environments. The validation/test sets include 820/1533 seen and 821/1529 unseen examples, respectively.

Evaluation metrics. We report the three main metrics used by the ALFRED leaderboard. Success Rate (SR): a binary indicator of whether all subtasks were completed. Path-Length Weighted Success Rate (PLWSR): SR weighted by (expert demonstration path length)/(agent path length). Goal-Condition Success Rate (GC): ratio of completed goal-conditions.[2] We note that success rate on the unseen test set is considered the primary metric for ranking because models are prone to memorizing the seen environments and often fail to generalize to unseen environments.

Models for comparison. We denote the base models described in §4.3.1 and §4.3.2 as VLN↻BERT and LSTM, respectively. To improve their competence on ALFRED, we further augment them with 1) pre-training their vision encoder on ALFRED images, and 2) reinforcement learning (§4.4). We denote the enhanced models as VLN↻BERT-L and LSTM-L, indicating their improved capability for long-horizon tasks. Finally, we integrate each of them with M-TRACK. Even though the focus of our evaluation is to test the effectiveness of M-TRACK on improving different base models, we still compare our results with other methods that are already published. Please refer to the supplementary materials for implementation details.

5.2. Main Results

We summarize the main results on the ALFRED test set in Table 2. First of all, the results show that both of our base models are highly competitive, performing on par or better than many recent VLN models such as E.T., LWIT, and HiTUT. On top of that, M-TRACK is highly effective in improving both base models: it improves the unseen SR of LSTM-L and VLN↻BERT-L by **4.6%** and **4.1%** absolute (**53%** and **33%** relative). Finally, VLN↻BERT-L + M-TRACK performed as much as the best published method (HLSM) on unseen SR (main metric), better on seen and unseen PLWSR, similarly on seen SR. The higher seen and unseen PLWSR indicate that our method successfully reduced the path length by focusing on the current subtask to complete the task.

5.3. Fine-grained Analyses

5.3.1 When to Apply M-TRACK?

The flexibility of M-TRACK makes it possible to be applied at training time, test time, or both. In Table 3, we show that

[2] For example in Fig. 1, there are 3 goal conditions: a potato is heated, a potato is on the counterop, and a heated potato is on the counter.

Model	Test Unseen			Test Seen		
	SR	PLWSR	GC	SR	PLWSR	GC
MOCA [31]	5.30	2.72	14.28	22.05	15.10	28.29
LAV [26]	6.38	3.12	17.27	13.35	6.31	23.21
EmBERT [33]	7.52	3.58	16.33	31.77	23.41	39.27
E.T. [27]	8.57	4.10	18.56	38.42	**27.78**	45.44
LWIT [25]	9.42	5.60	20.91	30.92	25.90	40.53
HiTUT [42]	13.87	5.86	20.31	21.27	11.10	29.97
ABP [15]	15.43	1.08	24.76	**44.55**	3.88	**51.13**
HLSM [2]	**16.29**	4.34	**27.24**	25.11	6.69	35.79
LSTM-L	8.70	4.05	16.97	14.04	7.20	21.73
LSTM-L + M-TRACK	13.28	6.25	20.20	22.05	12.83	30.48
VLN⟳BERT-L	12.23	5.60	19.64	21.46	11.56	28.99
VLN⟳BERT-L + M-TRACK	**16.29**	**7.66**	22.60	24.79	13.88	33.35

Table 2. **Performance on the ALFRED test set.** We evaluate M-TRACK on LSTM-L and VLN⟳BERT-L. M-TRACK notably improves all evaluation metrics on both test unseen and seen splits. Note that M-TRACK using VLN⟳BERT-L (or LSTM-L) achieves comparable gains to other existing methods. **Bold** refers to the highest score and underline refers to the second highest score.

	LSTM-L		VLN⟳BERT-L	
Train/Test	−	+	−	+
−	9.37	10.48	10.35	13.29
+	12.20	**15.83**	13.17	**17.29**

Table 3. **Unseen SR on ALFRED validation set with (+) or without (−) M-TRACK during training and/or test.** For example, the 17.29 cell indicates when M-TRACK is integrated into VLN⟳BERT-L during both training and test.

M-TRACK is already beneficial when applied only during training or test time, but the gain is most significant when it is applied during both training and test, suggesting that M-TRACK may be helping the base models in different ways during different phases.

5.3.2 Ablation Studies

Table 4 shows the effectiveness of different components.

Reinforcement learning. We show that RL, with our reward design, dramatically improves the performance of both base models, especially in unseen environments. This clearly demonstrates the importance of RL for long-horizon VLN tasks. While similar findings have been discussed for tasks with shorter horizons like R2R [12, 34, 43], we are among the first to validate its importance on ALFRED.

Pre-training object detector on ALFRED. The default Mask R-CNN model is pre-trained on COCO [18]. We continue pre-training it on ALFRED (ALFRED-OD), which further improves the performance.

Different milestone checking strategies. For milestone checking, we compare the passive checking and proactive checking strategies discussed in §4.1.2. As shown in Table 4, proactive checking performs better than passive checking, suggesting that *preventing a wrong action from happening is more preferable than correcting the mistake afterwards*. In contrast to our milestone checking, prior work [4] has proposed a binary classifier to check the completeness of current instruction. To compare with that, we implement a binary classifier using an MLP conditioned on the hidden state (LSTM-L) or the state encoding (VLN⟳BERT-L) that predicts if the current milestone has been reached. While it also helps, our milestone checking strategy is still advantageous by a large margin. Finally, we also estimate an upper bound for M-TRACK using ground-truth milestones from the environment instead of our milestone builder. While the results still show a decent room for improvement, the gap is not dramatic, indicating that our milestone builder is reasonably accurate, echoing Table 1.

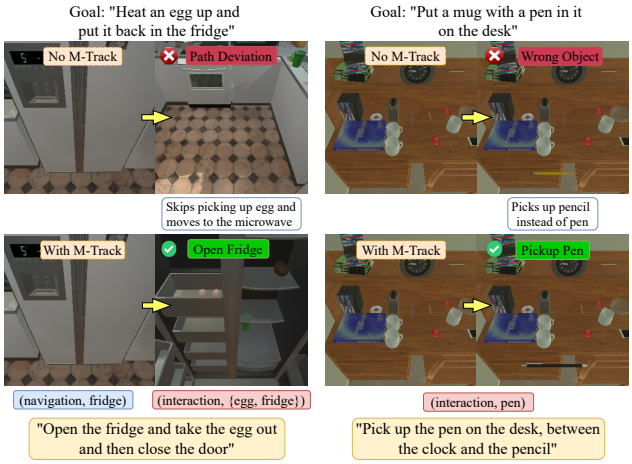

Figure 4. **Case studies for M-TRACK.**

5.4. Case Studies

We compare VLN⟳BERT-L (top) with VLN⟳BERT-L + M-TRACK (bottom) on two validation examples (left and right) to show the importance of M-TRACK (see Figure 4). First, VLN⟳BERT-L (top-left) skips the current in-

Model	Component						Val Unseen		Val Seen	
	RL	ALFRED-OD	Binary	Passive	Proactive	GT	SR	GC	SR	GC
LSTM							1.82	3.09	9.26	11.09
LSTM-L	✓						8.03	9.83	11.88	17.75
	✓	✓					9.37	12.56	15.00	18.37
	✓	✓	✓				10.83	13.39	17.68	20.46
	✓	✓		✓			15.22	18.88	20.97	24.73
LSTM-L + M-TRACK	✓	✓			✓		**15.83**	**20.34**	**21.70**	**25.45**
	✓	✓				✓	20.36	30.79	25.12	31.41
VLN⟳BERT							3.66	7.19	14.51	20.11
VLN⟳BERT-L	✓						9.37	16.42	16.83	23.12
	✓	✓					10.35	18.94	21.32	25.67
	✓	✓	✓				14.85	22.13	22.92	28.90
	✓	✓		✓			17.05	27.37	25.48	32.07
VLN⟳BERT-L + M-TRACK	✓	✓			✓		**17.29**	**28.98**	**26.70**	**33.21**
	✓	✓				✓	24.38	39.34	31.95	46.27

Table 4. **Ablation studies on the validation set. RL**: Reinforcement learning. **ALFRED-OD**: Mask R-CNN object detector pre-trained on ALFRED training images. **Binary**: Binary milestone classifier. **Passive**: Milestone checking *after* an action is executed. **Proactive**: Milestone checking *before* an action is executed. **GT**: M-TRACK with ground-truth milestones (an upper bound).

struction without completing it (*"take the egg out from the fridge"*), showing its limitation on the long-horizon task. In contrast, VLN⟳BERT-L + M-TRACK (bottom-left) completes all subtasks and eventually completes the whole task. Second, VLN⟳BERT-L (top-right) chooses the wrong object pencil instead of the correct object pen. The agent may be confused between the two objects since *"pencil"* also appears in the current instruction and indeed is semantically/visually similar to a pen. In contrast, VLN⟳BERT-L + M-TRACK (bottom-right) correctly performs the interaction task because of proactive milestone checking.

6. Discussion and Conclusions

We introduce a novel milestone-based task tracker (M-TRACK) for vision-and-language navigation (VLN) and show that explicit milestone detection and checking significantly benefits long-horizon VLN tasks such as those in ALFRED [30]. Our empirical results show the effectiveness of M-TRACK with two strong baseline models. In summary, this work clearly demonstrates the importance of *explicit progress monitoring* (as opposed to, *e.g.*, resorting to a single policy network for both planning and implicit progress monitoring), especially for long-horizon tasks. To make the point, we propose one instantiation with reference to the conditions in ALFRED, and different (or more generic) instantiations for different conditions can be explored in the future. We note the following limitations of the current design that warrant further development:

Assumptions in milestone builder. Our current instantiation assumes divisible language instructions corresponding to subtasks. It is worth mentioning that prior work (*e.g.*, BabyWalk [43]) does try a similar task decomposition idea on the R2R dataset and shows promising results, and we believe M-Track could be adapted similarly, though it is a less interesting setting for our purpose because of the short horizons. Nonetheless, in more general, realistic settings, accurate milestone building will likely be more challenging, especially when milestones are implicit (*e.g.*, *"fetch a cold beer"*). One interesting direction is to discover milestones by inductive reasoning over training instances instead of solely from language instructions. Event process mining techniques [41] could potentially be leveraged to discover that *"fetch a cold X"* generally entails going to a fridge and fetching X (*e.g.*, *"beer"*) from it.

Assumptions in milestone checker. To date, most VLN tasks are *declarative*. Milestone/goal checking thus can be done by checking solely against the environment state. For *procedural* instructions (*e.g.*, *"turn around twice"*) the milestone checker may need to check against the agent's action history, though such instructions are rare in existing datasets.

Non-unique golden trajectories. Though uncommon in ALFRED, in more complex tasks and/or environments, there could exist multiple viable trajectories (*e.g.*, different execution orders of subtasks leading to the same goal state) to complete a task. Currently milestones are assumed to be hard constraints that an agent has to achieve in order to proceed. It may be helpful to (learn to) soften the constraints imposed by milestones to provide more flexibility.

Acknowledgement

The authors would like to thank the colleagues from the OSU NLP group for their thoughtful comments. This research was supported in part by NSF OAC 2118240 and NSF OAC 2112606. J. Kil, T. Pan, and W. Chao are also partially supported by NSF IIS 2107077, the OSU GI Development fund, and the OSU CCTS Pilot grant.

References

[1] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *CVPR*, 2018. 1, 2

[2] Valts Blukis, Chris Paxton, Dieter Fox, Animesh Garg, and Yoav Artzi. A persistent spatial semantic representation for high-level natural language instruction execution. *arXiv preprint arXiv:2107.05612*, 2021. 2, 7

[3] Howard Chen, Alane Suhr, Dipendra Misra, Noah Snavely, and Yoav Artzi. Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *CVPR*, 2019. 1, 2

[4] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied question answering. In *CVPR*, 2018. 1, 2, 3, 7

[5] Abhishek Das, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Neural modular control for embodied question answering. In *Conference on Robot Learning*, 2018. 2

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 2, 4

[7] Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998. 5

[8] Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. Speaker-follower models for vision-and-language navigation. *arXiv preprint arXiv:1806.02724*, 2018. 2

[9] Daniel Gordon, Aniruddha Kembhavi, Mohammad Rastegari, Joseph Redmon, Dieter Fox, and Ali Farhadi. Iqa: Visual question answering in interactive environments. In *CVPR*, 2018. 1, 2

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017. 2, 4

[11] Yicong Hong, Cristian Rodriguez-Opazo, Qi Wu, and Stephen Gould. Sub-instruction aware vision-and-language navigation. *arXiv preprint arXiv:2004.02707*, 2020. 2

[12] Yicong Hong, Qi Wu, Yuankai Qi, Cristian Rodriguez-Opazo, and Stephen Gould. A recurrent vision-and-language bert for navigation. In *CVPR*, 2021. 2, 5, 6, 7

[13] Vihan Jain, Gabriel Magalhaes, Alexander Ku, Ashish Vaswani, Eugene Ie, and Jason Baldridge. Stay on the path: Instruction fidelity in vision-and-language navigation. *arXiv preprint arXiv:1905.12255*, 2019. 1, 2

[14] Liyiming Ke, Xiujun Li, Yonatan Bisk, Ari Holtzman, Zhe Gan, JJ (Jingjing) Liu, Jianfeng Gao, Yejin Choi, and Siddhartha Srinivasa. Tactical rewind: Self-correction via backtracking in vision-and-language navigation. In *CVPR*, 2019. 2

[15] Byeonghwi Kim, Suvaansh Bhambri, Kunal Pratap Singh, Roozbeh Mottaghi, and Jonghyun Choi. Agent with the big picture: Perceiving surroundings for interactive instruction following. In *Embodied AI Workshop CVPR*, 2021. 2, 7

[16] Shuhei Kurita and Kyunghyun Cho. Generative language-grounded policy in vision-and-language navigation with bayes' rule. *arXiv preprint arXiv:2009.07783*, 2020. 2

[17] Xiujun Li, Chunyuan Li, Qiaolin Xia, Yonatan Bisk, Asli Celikyilmaz, Jianfeng Gao, Noah Smith, and Yejin Choi. Robust navigation with language pretraining and stochastic sampling. *arXiv preprint arXiv:1909.02244*, 2019. 2

[18] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 7

[19] Edward Loper and Steven Bird. Nltk: The natural language toolkit. In *Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics. Philadelphia: Association for Computational Linguistics*, 2002. 5

[20] Chih-Yao Ma, Jiasen Lu, Zuxuan Wu, Ghassan AlRegib, Zsolt Kira, Richard Socher, and Caiming Xiong. Self-monitoring navigation agent via auxiliary progress estimation. *arXiv preprint arXiv:1901.03035*, 2019. 1, 2

[21] Chih-Yao Ma, Zuxuan Wu, Ghassan AlRegib, Caiming Xiong, and Zsolt Kira. The regretful agent: Heuristic-aided navigation through progress estimation. In *CVPR*, 2019. 1

[22] Arjun Majumdar, Ayush Shrivastava, Stefan Lee, Peter Anderson, Devi Parikh, and Dhruv Batra. Improving vision-and-language navigation with image-text pairs from the web. In *ECCV*, 2020. 2

[23] So Yeon Min, Devendra Singh Chaplot, Pradeep Ravikumar, Yonatan Bisk, and Ruslan Salakhutdinov. Film: Following instructions in language with modular methods. *arXiv preprint arXiv:2110.07342*, 2021. 2, 3

[24] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *ICML*, 2016. 6

[25] Van-Quang Nguyen, Masanori Suganuma, and Takayuki Okatani. Look wide and interpret twice: Improving performance on interactive instruction-following tasks. *arXiv preprint arXiv:2106.00596*, 2021. 7

[26] Kolby Nottingham, Litian Liang, Daeyun Shin, Charless C Fowlkes, Roy Fox, and Sameer Singh. Modular framework for visuomotor language grounding. *arXiv preprint arXiv:2109.02161*, 2021. 7

[27] Alexander Pashevich, Cordelia Schmid, and Chen Sun. Episodic transformer for vision-and-language navigation. *arXiv preprint arXiv:2105.06453*, 2021. 2, 5, 6, 7

[28] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *CVPR*, 2018. 1, 2

[29] Michaela Regneri, Marcus Rohrbach, Dominikus Wetzel, Stefan Thater, Bernt Schiele, and Manfred Pinkal. Grounding action descriptions in videos. *In TACL*, 2013. 1, 2

[30] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. In *CVPR*, 2020. 1, 2, 3, 6, 8

[31] Kunal Pratap Singh, Suvaansh Bhambri, Byeonghwi Kim, Roozbeh Mottaghi, and Jonghyun Choi. Factorizing perception and policy for interactive instruction following. In *CVPR*, 2021. 2, 6, 7

[32] Fábio Souza, Rodrigo Nogueira, and Roberto de Alencar Lotufo. Portuguese named entity recognition using bert-crf. *ArXiv*, abs/1909.10649, 2019. 2, 4

[33] Alessandro Suglia, Qiaozi Gao, Jesse Thomason, Govind Thattai, and Gaurav Sukhatme. Embodied bert: A transformer model for embodied, language-guided visual task completion. *arXiv preprint arXiv:2108.04927*, 2021. 2, 5, 7

[34] Hao Tan, Licheng Yu, and Mohit Bansal. Learning to navigate unseen environments: Back translation with environmental dropout. *arXiv preprint arXiv:1904.04195*, 2019. 1, 2, 6, 7

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 2

[36] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *NeurIPS*, 2015. 5

[37] Hanqing Wang, Wenguan Wang, Tianmin Shu, Wei Liang, and Jianbing Shen. Active visual information gathering for vision-language navigation. In *ECCV*, 2020. 1, 2

[38] Erik Wijmans, Samyak Datta, Oleksandr Maksymets, Abhishek Das, Georgia Gkioxari, Stefan Lee, Irfan Essa, Devi Parikh, and Dhruv Batra. Embodied question answering in photorealistic environments with point cloud perception. In *CVPR*, 2019. 1, 2

[39] Zhibiao Wu and Martha Palmer. Verb semantics and lexical selection. In *ACL*, 1994. 5

[40] Jiannan Xiang, Xin Wang, and William Yang Wang. Learning to stop: A simple yet effective approach to urban vision-language navigation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 699–707, Online, Nov. 2020. Association for Computational Linguistics. 3

[41] Hongming Zhang and et al. Analogous process structure induction for sub-event sequence prediction. In *EMNLP*, 2020. 8

[42] Yichi Zhang and Joyce Chai. Hierarchical task learning from language instructions with unified transformers and self-monitoring. *arXiv preprint arXiv:2106.03427*, 2021. 2, 3, 5, 7

[43] Wang Zhu, Hexiang Hu, Jiacheng Chen, Zhiwei Deng, Vihan Jain, Eugene Ie, and Fei Sha. Babywalk: Going farther in vision-and-language navigation by taking baby steps. *arXiv preprint arXiv:2005.04625*, 2020. 2, 3, 6, 7, 8

[44] Yuke Zhu, Daniel Gordon, Eric Kolve, Dieter Fox, Li Fei-Fei, Abhinav Gupta, Roozbeh Mottaghi, and Ali Farhadi. Visual semantic planning using deep successor representations. In *ICCV*, 2017. 1, 2