Appendices

In this supplementary material, we provide details omitted in the main text. Additionally, we release our code at https://github.com/chanhee-luke/M-Track.

- Appendix A: More related work
- Appendix B: M-Track implementation details
- Appendix C: Model implementation details
- Appendix D: Additional experiments

A. More Related Work

Due to the space constraint, we only include the most related works in the main text. Here, we add some extra related works to show recent trends in VLN.

Auxiliary information in VLN. M-Track can be viewed as an auxiliary information to the agent. A variety of auxiliary information has indeed been explored to improve the VLN models [2–5, 19, 20, 23, 25, 31, 38, 40, 41, 43]. Several prior works proposed to build a semantic map that encodes the spatial semantic information to bridge the gap between instructions and visual observations [2, 4, 23, 25]. Other studies suggested a topological map that memorizes previous actions and locations to facilitate planning [5, 31, 38]. M-Track is different from them by its simplicity, functionality, and compatibility — it is completely detached from VLN models and thus model-agnostic.

Data Augmentation in VLN. A number of prior studies investigate data augmentation to increase generalizability in unseen environments. One stream of works focuses on generating synthetic language instructions [7, 12, 22, 28, 39]. E.T. [28] constructs synthetic instructions using the expert path planner in ALFRED. Speaker-Follower [7] generates human-like textual instructions based on a VLN model trained on ground-truth routes. The other stream considers augmenting visual observations. Most of these works include surrounding views to enlarge an agent's field of view and thus enhance its navigation ability [4, 8, 11, 15, 27, 35]. In line with these studies, we augment visual observations using panoramic views with different angles and headings.

Learning Strategies in VLN. Several studies train VLN models with imitation learning [1, 17, 21, 28, 32, 33, 40] while some other works apply reinforcement learning [20, 26, 41]. To balance exploration and exploitation in navigation, some recent works leverage both imitation learning and reinforcement learning [11, 13, 14, 19, 36, 44]. Following the recent studies, M-Track exploits both of them and shows notable improvement on ALFRED.

Visual Input. There has been significant recent progress in learning visual representations of views. Several studies take image features encoded by ResNet [10] as visual input [1, 7, 32, 44]. VLN↻BERT [22] take as input object features from Faster R-CNN [29] to encode objects' semantic information. Some other studies leverage both image and object features to learn better visual representations [11, 13]. Recently, several papers use a pre-trained segmentation model (*e.g.*, Mask R-CNN [9]) to obtain more accurate object information [15, 25, 28, 33, 35, 42]. Following the recent trend in VLN, M-Track exploits Mask R-CNN to detect objects for milestone checking and encode their visual representations to facilitate interaction tasks.

B. M-Track Implementation Details

B.1. Milestone Builder

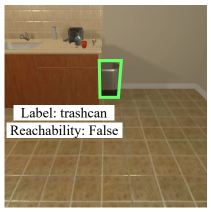
To estimate an upper bound of M-Track, we first build a ground-truth dataset using ground-truth tags derived from the ALFRED [32] expert demonstrations. The ALFRED expert demonstrations are encoded in Planning Domain Definition Language (PDDL) [24] rules. PDDL annotations include task-specific goal conditions for each low-level instruction. Each low-level instruction in PDDL language is defined by (d, i, p), where d = (action, argument) is a discrete action tuple containing the description of the action and its argument (object), i is the index of the low-level instruction, p = (action, location/ObjectID) is a planner action which is an action tuple directly applicable to the simulator. We use the discrete action tuple to tag the low-level instruction with the ground-truth object labels. For example, *"Go to the trash can on the far side of the kitchen"*, is labeled with the discrete action (GotoLocation, trashcan), based on which we automatically tag the instruction as (O, O, O, B-Nav, I-Nav, O, O, O, O, O, O, O). We apply BIO tagging format[1] to turn the object labels into tags. Every ALFRED low-level instruction has annotated labels, enabling us to build the ground-truth milestone training data easily. After tagging the ALFRED training and validation data with the ground truth object labels in this way, we train a BERT-CRF [6, 34] tagger on the training data to predict the milestone tags in the instruction. We choose BERT to utilize its powerful context encoding capability and add a CRF [18] layer on top of BERT to better model the interdependence of tag predictions. BERT-CRF is trained end to end with training data generated from the training split of ALFRED and validated with validation set generated with validation seen and unseen annotations for ALFRED. The model that has the highest F1 on the validation set is chosen.

During the main model (*e.g.*, VLN↻BERT-L + M-Track) execution, BERT-CRF outputs tags for a given low-level instruction. For instance, in the instruction, *"Turn*

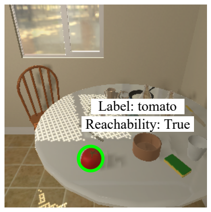
[1] B- prefix indicates the tag is the beginning of an object label, I- prefix indicates the tag is inside/end of the object label, and O refers to all non-tagged words.

Label: sink
Reachability: False

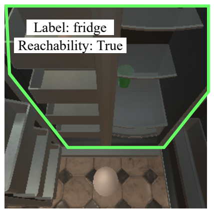
Walk around the island to the ***sink***.
Milestone: (navigation, sink)



Go to the ***trashcan*** across the room next to the counter
Milestone: (navigation, trashcan)



Pick up the ***tomato*** at the front of the table.
Milestone: (interaction, tomato)



Place the ***egg*** in the ***fridge***
Milestone: (interaction, {fridge, egg})

Figure 1. **Examples of milestones.**

and go to the sink”, BERT-CRF outputs (O, O, O, O, O, B-Nav). The (tag, word) pair is our predicted milestone for the instruction. We freeze BERT-CRF during the main model training.

B.2. Milestone Checking

Here we elaborate on the design decisions for some corner cases during milestone checking.

- **What if an instruction contains multiple milestones?**
  1. Navigation + Navigation: Navigate to the first milestone then to the next.
  2. Navigation + Interaction: Navigate to the navigation milestone first and navigate/interact with the interaction milestone.
  3. Interaction + Interaction: Interact with both objects without specifying a fixed order.
- **What if an instruction contains no milestone?** There are only less than 1% of such cases in the validation unseen environments in ALFRED (*i.e.*, 40 out of 5,140 instructions). For those cases, we concatenate the current instruction with the next instruction that has milestones detected.
- **What happens if the milestone builder makes a mistake, *e.g.*, missing a milestone / extracting an unnecessary milestone / detecting a wrong object?** Generally the milestone builder is pretty accurate (*c.f.* Table 1 in the main paper), so mistakes are not common. However, in the cases when a mistake does occur, we currently have the agent skip the milestone if the checking fails 15 consecutive times. On the other hand, if the object detector fails to detect the object, usually the failure is recovered later when the agent is in a different pose to take a new view of the object.

C. Model Implementation Details

C.1. VLN↻BERT

Our modified VLN↻BERT [13] consists of four modules: a language encoder, a vision encoder, an action decoder, and a pointer network with a multi-layer perceptron. Given a time step t, the language encoder takes current instruction (*i.e.*, the concatenation of the high-level instruction and the current low-level instruction) as input and outputs the contextualized token embeddings. Following VLN↻BERT, we consider the [CLS] embedding as a state embedding s_t representing an agent's current state and denotes other textual token embedding as x_i[2]. For the vision encoder, we leverage Mask R-CNN to obtain two types of visual features: 1) a scene feature v_j representing a view, and 2) an object feature o_k indicating an object in the view. In total, we extract 8 scene features from panoramic views (4 headings of 90° and 2 elevation angles of $\pm 30°$) and 20 highest scoring object features from all scenes. The action decoder then performs a grounded language learning by taking four inputs: a previous state embedding s_{t-1}, a sequence of textual embeddings $\{x_i\}$, a sequence of scene features $\{v_j\}$, and a sequence of object features $\{o_k\}$.

$$\begin{aligned} s_t, \{x'_i\}, \{v'_j\}, \{o'_k\} = \\ \text{VLN↻BERT}(s_{t-1}, \{x_i\}, \{v_j\}, \{o_k\}) \end{aligned} \tag{1}$$

Unlike VLN↻BERT, we employ a pointer network [37] to let the agent choose between navigation and interaction action. The pointer network predicts an action for the time step t by (2), (3), and (4).

$$\begin{aligned} u_n &= z^\top \tanh(W_1 e_n + W_2 s_t), \\ e_n &\in \{v'_j\} \cup \{o'_k\}, n \in (1, \cdots, J+K) \end{aligned} \tag{2}$$

where z, W_1, W_2 are learnable parameters, e_n is either the scene feature or the object feature, and s_t is the updated state embedding.

$$\hat{n} = \underset{n \in (1,\cdots,J+K)}{\text{argmax}} \; \sigma(u)_n \tag{3}$$

where σ is the softmax normalizing the vector u. If $\hat{n}$ represents a scene, the agent should navigate to the scene $\hat{n}$ at the time step t. In contrast, if $\hat{n}$ indicates an object, the agent should interact with the object $\hat{n}$ at the time step t by the corresponding interaction action $\hat{a}$

$$\hat{a} = \underset{a \in \text{IA}}{\text{argmax}} \, \sigma(W_3[s_t; o'_{\hat{n}}])_a \tag{4}$$

[2]We omit t on other variables (*e.g.*, textual token embeddings, scene/object features, etc.) for simplicity except for the state embedding.

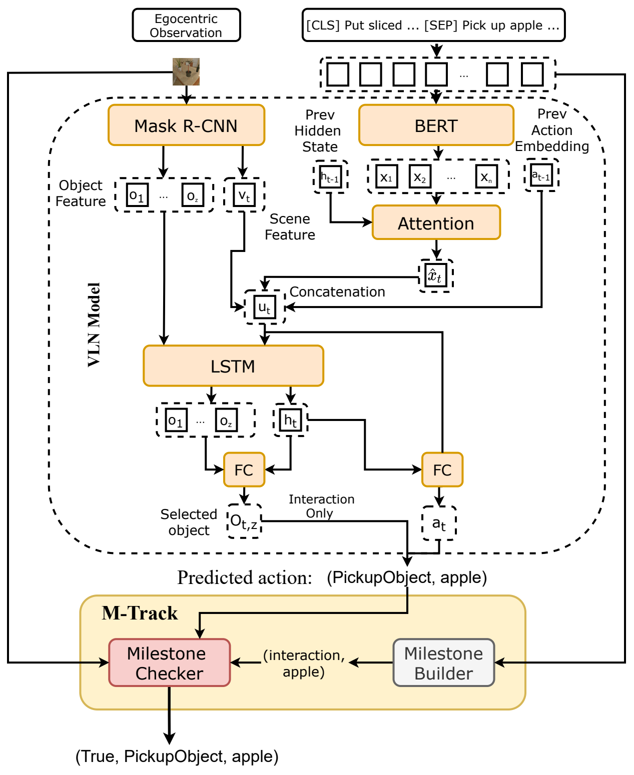


Figure 2. **Architecture of LSTM-L with M-TRACK.**

where W_3 is the learnable parameter, $o'_{\hat{n}}$ is the feature of object $\hat{n}$, and IA is the set of 7 interaction actions.

C.2. LSTM

We modify the CNN-LSTM model presented in ALFRED [32] with support for a pre-trained object encoder inspired by common VLN models' approach [7, 36, 38] in choosing between multiple scene features. Given a high level goal instruction G and step-by-step instructions $S = \{s_1, s_2, \cdots s_n\}$ of n instruction sentences, we concatenate the goal instruction with only the relevant step-by-step instruction such as $L = \{G, \text{<SEP>}, s_n\}$ with <SEP> token indicating the difference between the goal and step-by-step instruction. Then we perform a soft-attention on the language feature generated from BERT [6] to compute the attention distribution conditioned on the previous hidden state of the LSTM:

$$\begin{aligned} \alpha_t &= \text{softmax}((W_x h_{t-1})^\top) x_t \\ \hat{x_t} &= \alpha_t^\top x_t \end{aligned} \tag{5}$$

where W_x is the learnable parameter, h_{t-1} is the previous LSTM hidden state, x_t is the current language feature, and $\hat{x_t}$ is the weighted sum of x_t over the attention distribution α_t. Furthermore, each visual observation of the agent's view is encoded with a pre-trained Mask R-CNN [9], where we take the scene feature from its ResNet-50-FPN backbone. At each time step t, the LSTM takes in the object feature $\{o_{t,z}\}$, which are 20 highest-scoring object features from the vision encoder concatenated with spatial and reachability encoding as in VLN◯BERT, scene feature v_t, language feature $\hat{x_t}$, previous action embedding a_{t-1}, and outputs a new hidden state h_t:

$$h_t = \text{LSTM}([\{o_{t,z}\}; v_t; \hat{x_t}; a_{t-1}], h_{t-1}) \tag{6}$$

The agent interacts with the environment by choosing an action and providing a binary mask (if the action is an interaction). To leverage the power of the pre-trained vision encoder, we follow [28, 33] and ask our agent to choose an object $o_{t,z}$. The corresponding pixel mask is retrieved from the predicted object. We formulate object choosing in a same fashion as choosing navigable directions in common VLN models [7, 36, 38]. Action and object are generated from two different networks:

$$\begin{aligned} a_t &= \text{argmax}(W_a[h_t; u_t]) \\ p(o_{t,z}) &= \text{softmax}_z(o_{t,z} W_o h_t) \\ \hat{o}_{t,z} &= \operatorname*{argmax}_z p(o_{t,z}) \end{aligned} \tag{7}$$

where W_o and W_a are learnable parameters, and $u_t = [v_t; \hat{x_t}; a_{t-1}]$. Action prediction is trained with the ground-truth expert actions and reinforcement learning. The object feature is learned end-to-end with the ground-truth object information.

C.3. Vision Encoder

For the vision encoder for M-TRACK, we train an instance segmentation model, Mask R-CNN [9], as our vision encoder with training data generated from the expert demonstration images from ALFRED and ground-truth segmentation information from Ai2Thor simulator. Mask R-CNN is a two-stage detector, which the first stage proposes region of interests (RoI) by Region Proposal Network (RPN) [29] and the second stage extracts RoI features from the feature map by RoI Align [9] and makes predictions with three heads, box classification, box regression, and mask head. The box heads share the same RoI features extracted with proposals, while the mask head extracts with the predictions by the box regression head.

In the milestone checking, we not only are interested in objects and their locations but also care about whether they are reachable by the agent. Following the idea, we further implement the fourth head, using the same manner as the mask head, to predict availability for each object. We simply define objects within 1.5 meters as available for the binary classification by using the distance information from Ai2Thor simulator. Finally, the overall loss of pre-training our vision encoder is the summation of losses from four heads,

$$\mathcal{L} = \mathcal{L}_{cls} + \mathcal{L}_{reg} + \mathcal{L}_{mask} + \mathcal{L}_{avail}. \tag{8}$$

With a pre-trained vision encoder, we use its ResNet [10] backbone to encode scene features from environments and top-k RoI features from the mask head as object features to attend with the language model. For milestone checking, it also provides object labels and their reachability information.

C.4. Learning

C.4.1 Reward Shaping

In addition to the progress (navigation) and stop rewards defined in VLN↻BERT [13], we apply the interaction action matching as an additional reward to guide the agent to perform interaction action when needed. Furthermore, we introduce a visibility reward to make the agent learn to face the correct direction during the interaction.

Navigation Reward. Following VLN↻BERT, navigation reward acts as a strong supervision for directing our agent to the target object. Formally, D_t is a distance from the agent to the target object at time t, and $\Delta D_t = D_{t-1} - D_t$ is a change of distance by an action a_t. Reward for each a_t is defined as:

$$r_t^D = \begin{cases} +1, & \Delta D_t > 0 \\ -1, & \text{otherwise} \end{cases} \tag{9}$$

Stop Reward. When the agent decides to stop $a_t ==$ stop, we give the agent the final reward depending on if the task is successful or not:

$$r_{final} = \begin{cases} +3, & \text{Task} == \text{Success} \\ -3, & \text{otherwise} \end{cases} \tag{10}$$

where the task success means that the agent has completed all subtasks for the task.

Interaction Reward. Since VLN↻BERT trains the agent in a navigation-only dataset, we need to define an additional interaction reward in order for the agent to learn to choose between navigation and interaction actions. Formally, at a given time step t, the agent will be rewarded if the interaction action matches the ground-truth interaction action, which we retrieve from the environment state. The reward for each a_t is defined as:

$$r_t^I = \begin{cases} +1, & a_t == a_t^* \\ -1, & \text{otherwise} \end{cases} \tag{11}$$

Visibility Reward. We define an additional visibility reward to ensure the agent learns to face the correct direction of the object to be interacted with before predicting an interaction action to that object. This reward is paired with the interaction reward, so the total reward that an agent can get from interacting with the right object is $+2$.

$$r_t^V = \begin{cases} +1, & o_t^* \text{ is reachable} \\ -1, & \text{otherwise} \end{cases} \tag{12}$$

C.4.2 Behavior Cloning

Following the prevalent approach in training VLN models [13, 36], we combine reinforcement learning and imitation learning (*i.e.*, behavior cloning) to train our model. At each time step, the model is expected to produce the ground-truth action and interaction mask (for interaction actions). We apply cross-entropy loss between the predicted actions and the ground-truth action and add it to our reinforcement learning loss to ensure that the current trajectory is favored toward the expert demonstration trajectory. While we can adapt the expert demonstration directly on the LSTM-L + M-Track , it is not straightforward in VLN↻BERT-L + M-Track because it requires a panoramic input. Since ALFRED does not provide panoramic expert demonstration, we generated panoramic ground-truth trajectory information from ALFRED expert demonstration using its trajectory augmentation tool[3].

C.5. Training Details

For the LSTM, we use a pre-trained BERT as the language encoder and randomly initialize the rest of the model. For VLN↻BERT, we use its pre-trained weights on R2R [1] to initialize the model. For the vision encoder in the milestone builder, we use a ResNet-50-FPN [9] as the backbone for Mask R-CNN and finetune on ALFRED expert demonstrations from the Ai2Thor simulator [16] with batch size 16. We finetune the Mask R-CNN pretrained on ImageNet [30] on 4 Nvidia A6000 GPUs for 270k iterations, with a learning rate of 0.02, which is decreased by 10 at the 210k and 250k iteration. A weight decay of 0.0001 and momentum of 0.9 are applied. For all the experiments with M-Track and baseline models, we use a single Nvidia 2080TI GPU and AdamW optimizer is applied with a fixed learning rate of 10^{-5} for VLN↻BERT and 10^{-4} for LSTM. The batch size is set to 4, and the agent is trained for 20 epochs maximum. Early stopping is applied when the model shows no improvement on 3 consecutive epochs, and the model that shows the highest SR on the validation unseen split is adopted for testing. For all model training, only training split was used for training and validation split was held out.

D. Additional Experiments

Agent frequently skips subtask As mentioned in the main paper, we perform an experiment that shows that the agent frequently skips a subtask and tries to execute the next subtask. In Table 1, we perform an analysis of the first fatal

[3]https://github.com/askforalfred/alfred/tree/master/gen

Error Types		**L**	**L+M-TRACK**	**V**	**V+M-TRACK**
No error		76	129	83	134
Interaction Failure		231	255	178	199
	Collision	87	99	100	121
	Interact with other object	31	23	31	54
	Wander endlessly	66	130	22	24
	Navigate to next subtask location	47	0	33	0
Navigation Failure		513	436	559	487
	Collision	205	288	245	296
	Interact with other object	24	18	33	13
	Wander endlessly	183	130	203	172
	Navigate to next subtask location	101	0	78	0

Table 1. **Error cases on unseen validation. Interaction Failure:** Agent gets close to the target object but fails to interact with it. **Navigation Failure:** Agent does not navigate to the target object at all. **Next Action:** Next action that happens after the failure has happened. **L** stands for LSTM-L and **V** stands for VLN↻BERT-L.

Task Type	**Valid Unseen**	**Valid Seen**
Pick & Place	48	63
Stack & Place	28	25
Place Two	32	38
Examine	50	69
Heat & Place	19	18
Cool & Place	32	29
Clean & Place	27	29

Table 2. **Validation completion rate (%) by task type for VLN↻BERT-L + M-TRACK .**

error that the agent encounters for the validation unseen split. We first categorize the failure type by *interaction failure*, which the agent gets close to the interaction target but fails to perform an interaction action, and *navigation failure*, where the agent does not get close to a target object at all. Then, we also retrieve the next sequence of actions after the failure and sort them into four categories:

- **Collision**: Agent gets stuck in the environment and can not get out
- **Interaction with other object**: Agent interact with a wrong object or interacts when interaction is not needed
- **Wander endlessly**: Agent endlessly repeats a certain sequence of actions
- **Navigate to the next subtask location**: Agent navigates to the location of the next target object in the next subtask

We show that M-TRACK performs better on most error cases in both interaction and navigation. Specifically, M-TRACK notably prevents the agent from performing the next subtask without completing the current one (*e.g.*, 101 vs. 0), reflecting M-TRACK's idea.

M-TRACK completion rate by task type We further analyze M-TRACK's completion rate by task type in Table 2. We can see that M-TRACK excels at completing relatively simple tasks such as *"Pick & Place"* or *"Examine"* effectively, while suffers in complex tasks that require multiple interactions (*e.g.*, *"Heat & Place"*). We attribute this phenomenon due to the fact that the current M-TRACK does not effectively learn the multi-interaction reasoning. However, by seeing the notable improvement on the simple tasks, M-TRACK opens up the possibility of leveraging more fine-grained milestone construction to improve the agent's task learning in VLN.

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, 4

[2] Valts Blukis, Nataly Brukhim, Andrew Bennett, Ross A Knepper, and Yoav Artzi. Following high-level navigation instructions on a simulated quadcopter with imitation learning. *arXiv preprint arXiv:1806.00047*, 2018. 1

[3] Valts Blukis, Ross A Knepper, and Yoav Artzi. Few-shot object grounding and mapping for natural language robot instruction following. *arXiv preprint arXiv:2011.07384*, 2020. 1

[4] 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. 1

[5] Kevin Chen, Juan Pablo de Vicente, Gabriel Sepulveda, Fei Xia, Alvaro Soto, Marynel Vázquez, and Silvio Savarese. A behavioral approach to visual navigation with graph local-

ization networks. *arXiv preprint arXiv:1903.00445*, 2019. 1

[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. 1, 3

[7] 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. 1, 3

[8] Weituo Hao, Chunyuan Li, Xiujun Li, Lawrence Carin, and Jianfeng Gao. Towards learning a generic agent for vision-and-language navigation via pre-training. In *CVPR*, 2020. 1

[9] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017. 1, 3, 4

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 4

[11] Yicong Hong, Cristian Rodriguez-Opazo, Yuankai Qi, Qi Wu, and Stephen Gould. Language and visual entity relationship graph for agent navigation. *arXiv preprint arXiv:2010.09304*, 2020. 1

[12] Yicong Hong, Cristian Rodriguez-Opazo, Qi Wu, and Stephen Gould. Sub-instruction aware vision-and-language navigation. *arXiv preprint arXiv:2004.02707*, 2020. 1

[13] Yicong Hong, Qi Wu, Yuankai Qi, Cristian Rodriguez-Opazo, and Stephen Gould. A recurrent vision-and-language bert for navigation. In *CVPR*, 2021. 1, 2, 4

[14] 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

[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. 1

[16] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv*, 2017. 4

[17] Shuhei Kurita and Kyunghyun Cho. Generative language-grounded policy in vision-and-language navigation with bayes' rule. *arXiv preprint arXiv:2009.07783*, 2020. 1

[18] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, page 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. 1

[19] Jialu Li, Hao Tan, and Mohit Bansal. Improving cross-modal alignment in vision language navigation via syntactic information. *arXiv preprint arXiv:2104.09580*, 2021. 1

[20] Juncheng Li, Xin Wang, Siliang Tang, Haizhou Shi, Fei Wu, Yueting Zhuang, and William Yang Wang. Unsupervised reinforcement learning of transferable meta-skills for embodied navigation. In *CVPR*, 2020. 1

[21] 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. 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. 1

[23] Bar Mayo, Tamir Hazan, and Ayellet Tal. Visual navigation with spatial attention. In *CVPR*, 2021. 1

[24] Drew McDermott, Malik Ghallab, Adele E. Howe, Craig A. Knoblock, Ashwin Ram, Manuela M. Veloso, Daniel S. Weld, and David E. Wilkins. Pddl-the planning domain definition language. 1998. 1

[25] 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. 1

[26] Dipendra Misra, Andrew Bennett, Valts Blukis, Eyvind Niklasson, Max Shatkhin, and Yoav Artzi. Mapping instructions to actions in 3d environments with visual goal prediction. *arXiv preprint arXiv:1809.00786*, 2018. 1

[27] 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. 1

[28] Alexander Pashevich, Cordelia Schmid, and Chen Sun. Episodic transformer for vision-and-language navigation. *arXiv preprint arXiv:2105.06453*, 2021. 1, 3

[29] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *NeurIPS*, 2015. 1, 3

[30] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015. 4

[31] Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation. *arXiv preprint arXiv:1803.00653*, 2018. 1

[32] 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, 3

[33] Kunal Pratap Singh, Suvaansh Bhambri, Byeonghwi Kim, Roozbeh Mottaghi, and Jonghyun Choi. Factorizing perception and policy for interactive instruction following. In *CVPR*, 2021. 1, 3

[34] Fábio Souza, Rodrigo Nogueira, and Roberto de Alencar Lotufo. Portuguese named entity recognition using bert-crf. *ArXiv*, abs/1909.10649, 2019. 1

[35] 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. 1

[36] Hao Tan, Licheng Yu, and Mohit Bansal. Learning to navigate unseen environments: Back translation with environmental dropout. *arXiv preprint arXiv:1904.04195*, 2019. 1, 3, 4

[37] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *NeurIPS*, 2015. 2

[38] Hanqing Wang, Wenguan Wang, Wei Liang, Caiming Xiong, and Jianbing Shen. Structured scene memory for vision-language navigation. In *CVPR*, 2021. 1, 3

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

[40] Jiannan Xiang, Xin Eric Wang, and William Yang Wang. Learning to stop: A simple yet effective approach to urban vision-language navigation. *arXiv preprint arXiv:2009.13112*, 2020. 1

[41] Wei Yang, Xiaolong Wang, Ali Farhadi, Abhinav Gupta, and Roozbeh Mottaghi. Visual semantic navigation using scene priors. *arXiv preprint arXiv:1810.06543*, 2018. 1

[42] Yichi Zhang and Joyce Chai. Hierarchical task learning from language instructions with unified transformers and self-monitoring. *arXiv preprint arXiv:2106.03427*, 2021. 1

[43] Fengda Zhu, Yi Zhu, Xiaojun Chang, and Xiaodan Liang. Vision-language navigation with self-supervised auxiliary reasoning tasks. In *CVPR*, 2020. 1

[44] 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. 1