Goal-oriented Vision-and-Dialog Navigation via Reinforcement Learning

Yan Cao[1], **Keting Lu**[2], **David DeFazio**[3], **Shiqi Zhang**[3]
[1]University of Science and Technology of China, [2]Baidu Inc, [3]Binghamton University
{caotian, ktlu}@mail.ustc.edu.cn; {ddefazi1, zhangs}@binghamton.edu

Abstract

Vision-and-dialog navigation is a recent benchmark for evaluating the AI capabilities of perception, interaction, and decision making. While existing methods developed for this benchmark have demonstrated great successes, they mostly rely on large datasets, where data collection can be a challenge, and the learned policies are not adaptive to domain changes. In this paper, we focus on a new problem, referred to as goal-oriented vision-and-dialog navigation (GVDN), where an agent uses reinforcement learning techniques to compute dialog-navigation policies from trial and error. A robot conducts visual navigation to locate target objects, and can talk to a remote human operator as needed. Our remote human is able to provide guidance on navigation only if the robot correctly conveys its location through dialog. Experiments have been conducted using photo-realistic simulation environments. Results suggest that, our agent outperforms competitive baselines in success rate.

1 Introduction

Embodied mobile robots in human spaces need navigation capabilities to move from one place to another, while at the same time interacting with people. While many sensory modalities have been applied to robot navigation, vision is particularly attractive due to the significant achievements (Mnih et al., 2015; Silver et al., 2016; Levine et al., 2016; Mnih et al., 2016a; Schulman et al., 2017). As a result, researchers have studied the problem of vision-based robot navigation to demonstrate and evaluate a robot's simultaneous capabilities of perception and decision making (Bonin-Font et al., 2008; Zhu et al., 2017). When we further incorporate people into the loop, dialog systems (Lu et al., 2019; Gašić et al., 2013; Young et al., 2013; Yin and Wang, 2021; Liang et al., 2020; Chen et al., 2018; Ni et al., 2021; Li et al., 2020; Cao et al., 2020; Zhang et al., 2022) become important for human-robot communication (Zhang and Stone, 2015; Amiri et al., 2019; Tellex et al., 2020). Given the importance of robot interaction with both people and the environment, researchers have developed the "**vision-and-dialog navigation**" (VDN) benchmark, where a robot visually perceives an environment, talks to people using natural language, and makes decisions for navigation (Thomason et al., 2020; Nguyen and Daumé III, 2019; Anderson et al., 2018; Gu et al., 2022).

Existing VDN research focuses on training robots to follow language instructions, which describe the unambiguous or ambiguous goal and how to reach the goal (MacMahon et al., 2006; Chen and Mooney, 2011; Blukis et al., 2018; Fried et al., 2018; Ma et al., 2019b,a). Current VDN tasks have been modeled as sequence-to-sequence translation problems, and tackled using different supervised learning methods in the literature, e.g., Wang et al. (2018, 2020); Hao et al. (2020). Those methods require massive amounts of labeled data of robots performing dialog and navigation tasks collected from crowd-sourcing platforms. First, data collection is very expensive. A second issue is that solutions computed using those datasets are usually domain-dependent, and tend to soon become less applicable or outdated due to unforeseen domain changes. As a result, there is a need of computing VDN policies from the experience of interacting with both people and the environments (Nguyen et al., 2019; Nguyen and Daumé III, 2019; Thomason et al., 2020). To this end, we propose the **Goal-oriented Vision-and-Dialog Navigation (GVDN)** task, as a new benchmark, where a robot learns policies for vision-based perception, control for navigation, and dialog-based interaction. This new benchmark (GVDN) is our **first contribution**.

A GVDN agent must clearly communicate its current location in dialog, so as to get useful guidance for navigation from a human operator. Some areas are more confusing than the others for navi-

Findings of the Association for Computational Linguistics: EMNLP 2022, pages 4473–4482

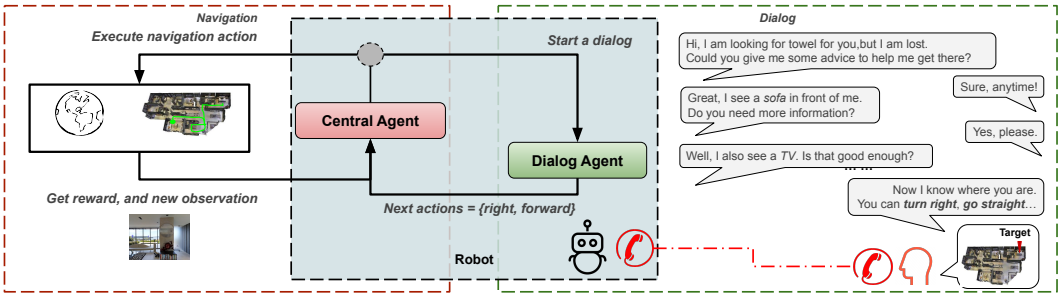


Figure 1: Our GVDN framework includes two agents for overall coordination and dialog respectively. A central agent learns navigation policies, and decides when to initiate a conversation. A dialog agent, as its name indicates, helps the robot making decisions on what dialog actions to take, and which object to tell human. After a dialog is concluded, the remote human provides the robot with a sequence of navigation actions, whose usefulness depends on whether the dialog is successfully performed or not.

gation, and the robot might find the guidance more useful when it is less experienced in navigation. Those factors together make it necessary for the robot to figure out when, where, and how a dialog is performed. Our **second contribution** is a novel GVDN approach that includes two reinforcement learning (RL) agents for overall coordination and dialog respectively, as shown in Figure 1.

We have implemented and evaluated our GVDN approach using Matterport3D (Chang et al., 2017). We evaluated our robot system based on its performance on the success rate and accumulative reward in GVDN tasks. Compared with visual navigation baselines (Zhu et al., 2017) with and without a dialog system, we demonstrated the superiority of our RL-based approach on GVDN tasks.

2 Methodology

In GVDN tasks, a robot needs to compute a policy to navigate to a predefined target object within an indoor environment. A remote operator can help the robot by suggesting navigation actions via natural language. The remote operator is helpful only if the robot can correctly communicate its current location; otherwise, the suggested navigation actions will be misleading due to the operator's wrong belief about the robot's location. At the beginning of each task, the robot is given as input a single target word token W_T. Given the robot's initial pose which includes the spatial position along with heading and elevation angles, the robot observes an initial RGB image, and the utterance of the operator. At each turn, the robot is expected to choose either a **locomotive** action to execute in the home environment or a **dialog** action to the operator for help to get closer to the target poses.

The locomotive action set $\mathcal{A}_{loc}$ consists of five actions corresponding to *left, right, up, down,* and *forward*. The forward action is defined to always move to the reachable viewpoint that is closest to the centre of the robot's visual field. The left, right, up and down actions are defined to move the camera by 30 degrees. The dialog actions $\mathcal{A}_{dial}$ include *inform, request, ok, unknown*, and *greeting*. The $inform$ action is to tell the operator objects which the robot has seen.

The challenge of GVDN tasks comes from the combination of two action spaces, referring to $\mathcal{A}_{loc} \cup \mathcal{A}_{dial}$. The large action space brings difficulties in naively learning from trial and error. To address this challenge, we exploit the main idea of hierarchical reinforcement learning for action space decomposition.

Algorithm Overview: Our GVDN framework includes two interdependent agents, namely a **central agent**, and a **dialog agent**. The central agent gives the commands to an action executor and the dialog agent. When the action executor receives locomotive actions in $\mathcal{A}_{loc}$, it will execute it in the environment. When the dialog agent receives a **dialog activation** action, it collects environment information and begins a dialog with the operator. The dialog agent aims to get commands (a sequence of navigation actions) from the operator which leads the robot to the target. To complete this sub-task, the dialog agent needs to learn to select a dialog action in $\mathcal{A}_{dial}$ in each turn of the dialog. Thus, the original action space is decomposed. The action space of the central agent is denoted as $\mathcal{A}_{cent}$ shown in Equation 1.

$$\mathcal{A}_{cent} = \mathcal{A}_{nav} \cup \{dialog\ activation\} \quad (1)$$

The $dialog\ activation$ action is dispatched to the dialog agent to execute. The action space of the dialog agent is inherited from $\mathcal{A}_{dial}$.

The central agent chooses locomotive actions to enable the robot to navigate towards the target

Algorithm 1 Our GVDN algorithm

1: Initialize $Q_{cent}(s, a; \theta_{Qc})$ of the central agent $agent^{Cent}$ and $Q_{dial}(s, a; \theta_{Qd})$ of the dialog agent $agent^{Dial}$
2: Initialize experience replay buffers B^C and B^D for the interaction of the central agent and the dialog agent
3: **while** a new target W_T arrives **do**
4: Initialize action sequence $Acts^h$ from the operator, and collect initial state, s
5: **while** $s \notin$ term **do** // Start an interaction with the 3D environment
6: Select $a \leftarrow CentralFunc(Q_{cent}, s, W_T, Acts^h)$ of the central agent
7: **if** a is *dialog* action **then**
8: $Acts^h \leftarrow DialogInteract^2(Q_{dial}, s, B^D)$ // Start a dialog interaction with the operator
9: **else**
10: Execute a in the 3D environment
11: **end if**
12: Collect next state s', and reward r, then add (s, a, r, s') to B^C
13: $s \leftarrow s'$
14: **end while**
15: Randomly sample a batch from B^C, and update $agent^{Cent}$ via DQN
16: Randomly sample a batch from B^D, and update $agent^{Dial}$ via DQN
17: **end while**

Algorithm 2 DialogInteract

Input: s_{cent}, state from the central agent; $Q_{dial}(\cdot)$ for the dialog agent; B^D replay buffer to store the dialog tuples
Output: $Acts$, locomotive action sequence to be executed

1: Initialize the next action sequences $Acts \leftarrow \emptyset$
2: Collect initial dialog state, s
3: **while** $s \notin$ term **do**
4: Select $a \leftarrow argmax_{a'} Q_{dialog}(s, a'; \theta_{Qd})$, and execute a
5: Collect next state s', and reward r
6: Add dialog turn $d = (s, a, r, s')$ to B^D
7: $s \leftarrow s'$
8: **end while**
9: Extract $Acts$ from the dialog history
10: **return** $Acts$

poses. Otherwise, the central agent queries the dialog agent and expects next suggested commands that help lead the robot from the current pose to the target. Before beginning an episode of dialog, the dialog agent collects necessary information of the environment from the central agent. The information contains the current objects around the robot, which may be requested by the operator. At the end of each dialog, the central agent will get the next commands from the operator if the dialog was successful. Otherwise, nothing is transferred from the dialog agent to the central agent. Once the central agent has received the suggested commands from the dialog agent, it dispatches them to the action executor to execute. The two agents collaborate as above, until the robot arrives at the target poses.

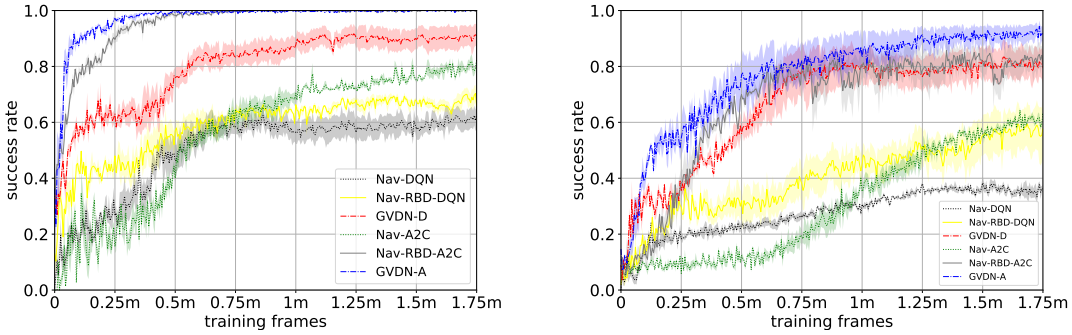
Our GVDN Algorithm: Our GVDN robot is mainly composed of the central agent and the dialog agent that are respectively learned from the interactions with the 3D environment and the operator. Algorithm 1 shows the learning process of our GVDN algorithm. Algorithm 1 starts with an initialization of the two agents' experience replay buffers (B^C and B^D respectively), the value function $Q_{cent}(s, a; \theta_{Qc})$ of the central agent, and the value function $Q_{dial}(s, a; \theta_{Qd})$ of the dialog agent. Before the start of each episode, action sequence $Acts^h$ is initialized as empty for storing the next actions from dialogs. In the *while* loop (starting in Line 5), the central agent chooses action a to execute, interacts with the 3D environment, and stores the real experience in B^C. If a is a *dialog activation* action, the dialog agent is activated to interact with the operator, and pass the next action sequence $Acts^h$ to the central agent (in Line 8). At the end of each episode, we use DQN to update both θ_{Qc} and θ_{Qd}

The Dialog Agent directly interacts with the operator, where the interaction (invoked in Algorithm 1) is presented in Algorithm 2. In addition to the value function $Q_{dial}(s, a; \theta_{Qd})$ and the replay buffer B^D for the dialog agent, there is parameter s_{cent} for navigation state. Algorithm 2 starts with an initialization of the next action sequence $Acts$. The dialog agent interacts with the operator while collecting and saving experience in B^D. At the end of each dialog episode, the dialog agent extracts the next action from the dialog history. The output of Algorithm 2, $Acts$, is returned to Algorithm 1 to guide the central agent's navigation actions.

3 Experiments

Experiments have been conducted in a realistic 3D indoor simulation platform, called Matterport3D (Anderson et al., 2018), which allows an embodied agent to virtually move throughout a scene by adopting poses coinciding with panoramic viewpoints. Each panoramic viewpoint is comprised of 18 images captured from a single 3D position. We manually add action noise into the simulated robot's navigation behaviors, i.e., there is 0.1 probability that the robot fails to execute selected actions and keeps still.

We conduct experiments in two house environments, referred to as **house-I** and **house-II**. House-I includes 20 viewpoints, 8 rooms, and 200 objects;



(a) House-I (b) House-II

Figure 2: Learning curves of success rate on the GVDN task in the house-I (left) and house-II (right). Six robots: Nav-DQN (One-step Q), Nav-RBD-DQN, GVDN-D, Nav-A2C, Nav-RBD-A2C, and GVDN-A.

House-II is bigger and includes 50 viewpoints, 15 rooms, and 274 objects.[1]

Task Specification: We on average select 4 objects as target sets and annotate the target locations and poses in each house environment. The start pose of the robot is randomly selected, which is no more than 15 steps from the target locations. If acting optimally, the average number of actions to reach the target from the starting position is 11. When the robot arrives at the target pose in < 50 steps, it will receive reward 10 (otherwise -5).

3.1 Case Illustration

Before presenting statistical results, we use an example GVDN task to illustrate the interaction between the robot and the environment, and the interaction between the robot and human, as shown in Figure 3. The goal of the GVDN task is to navigate to a predefined target object. In the example, the target object is a unique "statue". The beginning pose of the robot is shown in the first frame (in the top left corner of Figure 3). On the first step, the robot decided to turn left, and got a new observation of the second frame. Then, the robot made serial decisions and came downstairs at the pose shown in the sixth frame. Then, the robot decided to talk to the operator for advice. The dialog between the operator and the robot started with a request from the robot. Before the operator provides its suggested commands, the robot responded to the operator with what it had seen. At the end of the dialog, the robot successfully got the next commands. Following the suggested commands, the robot arrived at the pose shown in the ninth frame. The robot made decisions on its own until it arrived at the target pose and saw the object in the twelfth frame.

[1]The environment IDs of house-I and house-II are *8194nk5LbLH* and *JF19kD82Mey* respectively.

3.2 Baselines

We have selected four competitive baselines for GVDN tasks, which use either DQN or A2C. **Nav-DQN** has a DQN-based navigation policy, i.e., One-step Q (Zhu et al., 2017). **Nav-A2C** is the same as Nav-DQN, except that DQN is replaced by single-threaded A3C (Mnih et al., 2016b), also called A2C. **Nav-RBD-DQN** uses a DQN-based visual navigation policy and a rule-based dialog (RBD) approach for dialog management (Nguyen et al., 2019), where language-based assistance is always-on, and provided in a predefined pattern. **Nav-RBD-A2C** is the same as Nav-RBD-DQN, except that DQN is replaced by A2C.

There are two configurations of our proposed GVDN approach. **GVDN-D** uses a DQN-based navigation policy and a DQN-based dialog policy, and **GVDN-A** is the counterpart of GVDN-D that replaces DQN with A2C for policy learning.

3.3 Hypothesis and Result

For each of the six models, we have conducted three "runs", where each run includes 1,750,000 navigation frames. In each run, after every 5000 training frames, we let the robots interact with the 3D environment 100 times and compute the success rate over the 100 navigation interactions. Each data point in the figures is an average over the three success rates collected from the three runs.

Success Rate: Figure 2 presents the quantitative comparisons between both configurations of our GVDN method and the four baselines. Figure 2(a) shows that GVDN-A (the blue line) performed consistently better than the four baselines, which supports our key hypothesis. In addition, GVDN-D (the red line) and GVDN-A (the blue line) are respectively learning faster than Nav-RBD-DQN (the yellow line) and Nav-RBD-A2C (the grey solid

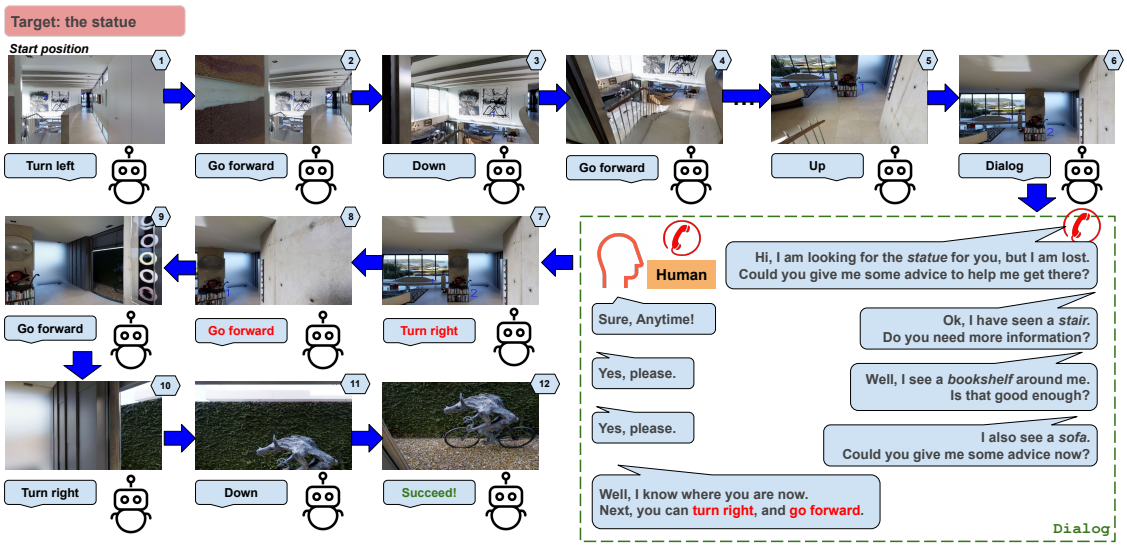


Figure 3: An example of a GVDN task: Given a target object (the statue), the robot starts to select actions from the start pose to the target pose. With the different heading and elevation of the robot, the robot can go forward to different poses, including upstairs and downstairs. The robot can also choose to talk to the human for help and receive the next commands to make progress, if it accesses them successfully through dialog (after the sixth frame). At the twelfth frame, the robot finally arrives and sees what it seeks for, the statue.

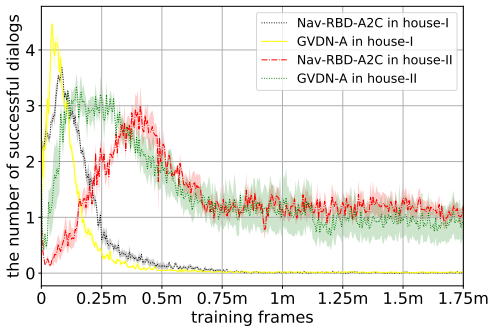


Figure 4: The average number of the successful dialogs robots made in the training phase

line). We conclude that the learned dialog policy accelerates the training process, when compared to the rule-based dialog policy.

To test the robustness of our method we use a bigger house environment, house-II, with the same settings, where we receive similar observations from Figure 2(b). We also observe that the learning rates of all methods in house-II are lower than the learning rates in house-I, which indicates that bigger domains are more difficult for GVDN agents.

Dialog Results: To further figure out to what degree the learned dialog policy influences the number of the successful dialogs the robot made, we conduct experiments with both rule-based and learned dialog policies in two houses of different sizes. Figure 4 shows the average number of successful dialogs during learning phases of Nav-RBD-A2C and GVDN-A in both house-I and house-II. At the early 125,000 training frames in house-I, GVDN-A made a few more successful dialogs than Nav-RBD-A2C. In house-II, during the first 100,000 training frames, GVDN-A succeeded in making more successful dialogs than Nav-RBD-A2C. Thus, the learned dialog policy can enable the robot to get more suggested commands to make progress than the rule-based dialog policy, in both houses. After 750,000 training frames, the successful dialogs decrease in all four experiments, which denotes the robots relied less on suggested commands from the human. During the later training phase, the robots in house-II make a few more successful dialogs than the ones in house-I, which denotes help from the human is still needed in bigger environments. We conclude that the learned dialog policies can enable the robot to get more suggested commands in both house-I and house-II.

4 Conclusion

In this work, we focus on goal-oriented vision-and-dialog navigation (GVDN), where an agent is not provided with any data, and needs to learn navigation-dialog policies from trial and error toward achieving long-term goals. Our GVDN algorithm enables robots to learn from the simultaneous experiences of interacting with the environment via navigation, and interacting with the human via dialog. Our GVDN algorithm outperforms competitive baselines that are incapable of performing dialog actions or use rule-based dialog systems in success rate and task completion efficiency.

Limitations

The current conversations are only about landmark objects and their locations. There is great potential of improving the work by introducing other types of dialog actions, such as describing the semantic information of room types and their functionalities. Our method works mostly for home environments, while other types of environments (such as streets, shopping malls, and hospitals) might introduce new challenges and opportunities.

Another direction for future work is to incorporate robot motion control into the loop, while currently our robot is teleported between viewpoints. The current conversations are only about landmark objects and their locations. There is great potential of improving the work by introducing other types of dialog actions, such as describing the semantic information of room types and their functionalities. The experiments were performed in home environments, while other types of environments (such as offices, shopping malls, and hospitals) might introduce new challenges and opportunities.

Ethics Statement

The described research intends to enable robots to learn from the simultaneous trial-and-error experiences of interacting with an environment via navigation behaviors, and interacting with a remote human via dialog behaviors. Data used to train the robot was collected from the simulation of a mobile robot interacting with a 3D indoor environment, and a remote human operator. There is the great potential of applying our developed approach to utilizing additional information extracted from human-robot dialog scenarios, and interacting with different types of human users (simulated or real ones). Finally, the observations and conclusion from this research are expected to generalize in other platforms and the real world.

References

Saeid Amiri, Sujay Bajracharya, Cihangir Goktolgal, Jesse Thomason, and Shiqi Zhang. 2019. Augmenting knowledge through statistical, goal-oriented human-robot dialog. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 744–750. IEEE.

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. 2018. Vision-and-Language Navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Valts Blukis, Dipendra Misra, Ross A. Knepper, and Yoav Artzi. 2018. Mapping navigation instructions to continuous control actions with position-visitation prediction. In *CoRL*.

Francisco Bonin-Font, Alberto Ortiz, and Gabriel Oliver. 2008. Visual navigation for mobile robots: A survey. *Journal of intelligent and robotic systems*.

Yan Cao, Keting Lu, Xiaoping Chen, and Shiqi Zhang. 2020. Adaptive dialog policy learning with hindsight and user modeling. In *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 329–338.

Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niebner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. 2017. Matterport3d: Learning from rgb-d data in indoor environments. In *2017 International Conference on 3D Vision (3DV)*. IEEE.

David Chen and Raymond Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Yun-Nung Chen, Asli Celikyilmaz, and Dilek Hakkani-Tur. 2018. Deep learning for dialogue systems. In *Proceedings of the 27th International Conference on Computational Linguistics: Tutorial Abstracts*, pages 25–31.

Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. 2018. Speaker-follower models for vision-and-language navigation. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*.

Milica Gašić, Catherine Breslin, Matthew Henderson, and et al. 2013. On-line policy optimisation of bayesian spoken dialogue systems via human interaction. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE.

Jing Gu, Eliana Stefani, Qi Wu, Jesse Thomason, and Xin Eric Wang. 2022. Vision-and-language navigation: A survey of tasks, methods, and future directions. In *Association for Computational Linguistics (ACL)*.

Weituo Hao, Chunyuan Li, Xiujun Li, Lawrence Carin, and Jianfeng Gao. 2020. Towards learning a generic agent for vision-and-language navigation via pre-training. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on*

computer vision and pattern recognition, pages 770–778.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. 2016. End-to-end training of deep visuomotor policies. *JMLR*.

Xiujun Li, Zachary C Lipton, Bhuwan Dhingra, Lihong Li, Jianfeng Gao, and Yun-Nung Chen. 2016. A user simulator for task-completion dialogues. *arXiv preprint arXiv:1612.05688*.

Xuijun Li, Yun-Nung Chen, Lihong Li, Jianfeng Gao, and Asli Celikyilmaz. 2017. End-to-end task-completion neural dialogue systems. In *Proceedings of The 8th International Joint Conference on Natural Language Processing*.

Ziming Li, Sungjin Lee, Baolin Peng, Jinchao Li, Julia Kiseleva, Maarten de Rijke, Shahin Shayandeh, and Jianfeng Gao. 2020. Guided dialogue policy learning without adversarial learning in the loop. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 2308–2317.

Weixin Liang, Youzhi Tian, Chengcai Chen, and Zhou Yu. 2020. Moss: End-to-end dialog system framework with modular supervision. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8327–8335.

Keting Lu, Shiqi Zhang, and Xiaoping Chen. 2019. Goal-oriented dialogue policy learning from failures. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, pages 2596–2603.

Chih-Yao Ma, Jiasen Lu, Zuxuan Wu, G. Al-Regib, Z. Kira, R. Socher, and Caiming Xiong. 2019a. Self-monitoring navigation agent via auxiliary progress estimation. *ICLR*.

Chih-Yao Ma, Zuxuan Wu, G. Al-Regib, Caiming Xiong, and Z. Kira. 2019b. The regretful agent: Heuristic-aided navigation through progress estimation. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

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

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Tim Harley, Timothy P. Lillicrap, David Silver, and Koray Kavukcuoglu. 2016a. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016b. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, and et al. 2015. Human-level control through deep reinforcement learning. *Nature*.

Khanh Nguyen and Hal Daumé III. 2019. Help, anna! visual navigation with natural multimodal assistance via retrospective curiosity-encouraging imitation learning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Khanh Nguyen, Debadeepta Dey, Chris Brockett, and W. Dolan. 2019. Vision-based navigation with language-based assistance via imitation learning with indirect intervention. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12519–12529.

Jinjie Ni, Tom Young, Vlad Pandelea, Fuzhao Xue, Vinay Adiga, and Erik Cambria. 2021. Recent advances in deep learning based dialogue systems: A systematic survey. *arXiv preprint arXiv:2105.04387*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms.

David Silver, Aja Huang, Chris J. Maddison, and et al. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature*.

Stefanie Tellex, Nakul Gopalan, Hadas Kress-Gazit, and Cynthia Matuszek. 2020. Robots that use language. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:25–55.

Jesse Thomason, Michael Murray, Maya Cakmak, and Luke Zettlemoyer. 2020. Vision-and-dialog navigation. In *Conference on Robot Learning*. PMLR.

Hanqing Wang, Wenguan Wang, Tianmin Shu, Wei Liang, and Jianbing Shen. 2020. Active visual information gathering for vision-language navigation. In *European Conference on Computer Vision*, pages 307–322. Springer.

Xin Wang, Wenhan Xiong, Hongmin Wang, and William Yang Wang. 2018. Look before you leap: Bridging model-free and model-based reinforcement learning for planned-ahead vision-and-language navigation. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Kai Yin and Zhenyu Wang. 2021. Weighted user goal sampling for dialog policy learning. In *Journal of Physics: Conference Series*, volume 1757, page 012078. IOP Publishing.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*.

Haodi Zhang, Zhichao Zeng, Keting Lu, Kaishun Wu, and Shiqi Zhang. 2022. Efficient dialog policy learning by reasoning with contextual knowledge. In *Proceedings of the Thirty-Sixth AAAI Conference on Artificial Intelligence*.

Shiqi Zhang and Peter Stone. 2015. Corpp: commonsense reasoning and probabilistic planning, as applied to dialog with a mobile robot. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 1394–1400.

Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, and et al. 2017. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE.

A Appendix

A.1 Algorithm Instantiation

For completeness, we include some details about the central agent, including state space, the reward design, and the policy learning.

State Space: To navigate to the target location, the central agent assumes it operates over a Markov Decision Process (MDP). In each step, the central agent observes image o taken by the robot's RGB camera in its first person view. For each image observation o, we use a ResNet-152 CNN pretrained on ImageNet to extract a mean-pooled feature vector. Given a target, we use a pre-trained *word2vector* model to encode the target word W_T. The image observation and word vector are then concatenated together to form the state:

$$s_{cent} = \textit{ResNet}(\ o\ ) \oplus \textit{Word2Vec}(\ W_T\ ) \quad (2)$$

where $\oplus$ denotes concatenation.

Reward: Only when the robot successfully locates the target object (i.e., standing in front of and facing the target), will the central agent receive a big bonus. It will receive a penalty when it fails in target search in the max number of steps. When executing actions in the 3D environment, there exists a small time penalty r_{nav} to encourage shorter trajectories. When the dialog agent correctly communicates its current location, a large bonus is provided to encourage the dialog agent to learn to improve its dialog management skills. When the central agent chooses *dialog activation* actions, the immediate reward depends on the performance of the dialog agent. A successful dialog produces a reward that is four times of that from an unsuccessful one. This setting is for encouraging the central agent to interact with the dialog agent when dialog quality is high. At the same time, it should be noted a rational central agent is able to find that interacting with a dialog agent is relatively less useful when it becomes better at selecting navigation actions.

Central Policy: Due to the state space being continuous, the approximated value function of the central agent is implemented using a three-layer fully connected neural network, $Q_{cent}(s_{cent}, a; \theta_{Qc})$, parameterized by θ_{Qc}. We improve the value function by adjusting θ_{Qc} to minimize the mean-squared loss function. Besides the value function to make decisions, the central agent deterministically follow the advised actions (when available) in the action sequence $\overrightarrow{a}$ from the central agent instead of the output of the value function Q_{cent}:

$$a \leftarrow \begin{cases} \textit{argmax}_{a'} Q_{cent}(s_{cent}, a'; \theta_{Qc}) & \overrightarrow{a} = \emptyset \\ \overrightarrow{a} & \overrightarrow{a} \neq \emptyset \end{cases} \quad (3)$$

Note that the advised actions in $Acts^h$ are not always optimal in terms of taking the shortest route towards the target. The quality of the dialog policy influences the correctness of the suggested action sequence. In some ways, this is a sort of exploration strategy of the central learning agent.

Dialog Policy: Before starting a dialog with the operator, the dialog agent should access environment information from the central agent, i.e. the visible object set $Objs$ around the robot. The dialog agent concatenates the one-hot vector of $Objs$ into the dialog state s. In each dialog turn, the dialog agent chooses a dialog action in $\mathcal{A}_{dial}$ to speak to the human, based on the current dialog state. At the end of the dialog, if the robot accesses the next action sequence, the dialog is considered successful. We model the dialog policy with a three-layer fully connected neural network, $Q_{dial}(s, a; \theta_{Qd})$, parameterized by θ_{Qd}. We improve the dialog policy θ_{Qd} by adjusting θ_c via DQN to minimize the mean-squared loss function.

A.2 Implementation Details

The central agent is implemented using Deep Q-Network (DQN) and Advantage Actor-Critic (A2C). We use a ResNet-152 (He et al., 2016) CNN pretrained on ImageNet to extract a mean-pooled feature vector with 2049 dimensions. We use GloVe (Pennington et al., 2014) model to get word representations with 50 dimensions. The DQN network of the central agent includes one hidden layer with 100 hidden nodes, and its output layer includes 6 units corresponding to 6 navigation actions. We set the discount factor $\gamma = 0.9$. The experience buffer size is 8000. The network parameters are updated every 20 navigation episodes, while the target network parameters are updated every 200 navigation episodes. The network architecture of the actor and critic models are MLPs with a hidden layer of 64 units. The input and output size of the network are the same as the DQN model. The number of forward steps in A2C is 5. The learning rate is set 0.00005. The number of suggested commands passed by the central agent

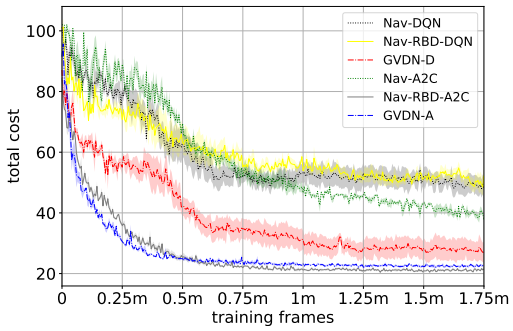


Figure 5: Total cost of navigation steps and dialog steps in house-I with the models: Nav-DQN (One-step Q), Nav-RBD-DQN, GVDN-D, Nav-A2C, Nav-RBD-A2C, and GVDN-A

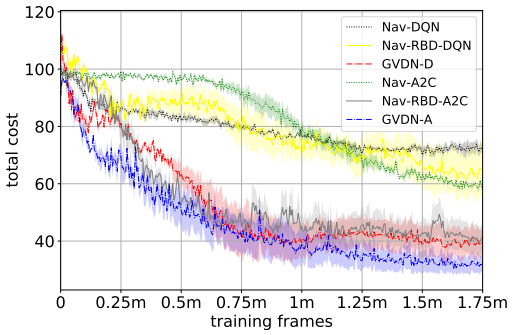


Figure 6: Total cost of navigation steps and dialog steps in house-II with the models: Nav-DQN (One-step Q), Nav-RBD-DQN, GVDN-D, Nav-A2C, Nav-RBD-A2C, and GVDN-A

is no more than 3. These tuples are also stored in a replay buffer for training.

The dialog agent is implemented using DQN. The dialog DQN includes one hidden layer with 60 hidden nodes and ReLU activation, and its output layer of 42 units corresponds to 42 feasible dialog actions. We set the discount factor $\gamma = 0.9$ The buffer size is 8000, and we use uniform sampling in experience replay. The target value function is updated after every 200 dialog episodes. In each epoch, the dialog policy is refined using one-step 16-tuple-minibatch updates. The learning rate is set to 0.0001. All neural network parameters are randomly initialized, and optimized using RMSProp. The experiments were performed using a desktop machine equipped with a RTX 3080 GPU and 32G memory.

Simulated Human Operator: In our GVDN setting, we construct a rule-based simulator to simulate the operator that responds to robot requests. The operator has the global map of the house environment, but does not know the location of the robot. Based on the objects around the robot, the operator can infer which room the robot is in through dialog, and give the next command to the robot. In each dialog episode, the operator keeps a candidate region list. After the robot tells the operator which object it has seen each turn, the operator will delete the region which excludes the object. Note that the number of regions in the candidate region list may be empty or more than one at the end of the dialog, in which case the operator won't plan the route. Once getting only one candidate region, the operator begins planning the shortest path to the target. Then the operator gives the next commands to the robot.

Dialog Simulation: We use TC-bot (Li et al., 2017, 2016) for human-robot dialog simulation, and set "language level" as semantic. A dialog action consists of 8 dialog acts and 43 slots. The dialog actions include *request_next_act, inform, thanks, greeting, OK, not_clear, closing*, and *you_just_said*. An inform dialog action of *sofa* can be like INFORM('sofa'). The max dialog turns are set 20. The dialog is considered successful only when the robot acquires the next commands from the operator, where the robot will receive a big bonus 40, otherwise, -20. In each dialog turn, the robot receives a small punishment, -1, so as to encourage shorter dialogs.

Total Cost: To further analyze the efficiency of our method in terms of total cost, we consider the weighted average of both navigation and dialog steps in each episode. When the robot remains still, we consider the cost to be one unit. Thus, one dialog interaction costs one unit. We consider a navigation step as costing 2 units, because we assume physical movements are more expensive than language actions. Figure 5 shows that GVDN-D (the red line) has a lower cost than Nav-DQN (the grey dotted line) and Nav-RBD-DQN (the yellow line). Similarly, GVDN-A (the blue line) and Nav-RBD-A2C (the grey solid line) has lower cost than Nav-A2C (the green line). We conclude that our GVDN algorithm enables robots to learn more efficiently than the baselines.

To test our GVDN method in bigger environments, we experimented it in house-II. Figure 6 displays similar results as house-I, which shows the robustness of our GVDN method. Note that the total cost of each method in Figure 6, which is a bigger house, is higher than those in Figure 5.