Approaches for Interactions in Robotics Applications

Junha Roh

A dissertation

submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

University of Washington
2022

Reading Committee:

Dieter Fox, Chair
Ali Farhadi, Chair
Maya Cakmak

Program Authorized to Offer Degree:
Computer Science and Engineering

University of Washington

Abstract

Approaches for Interactions in Robotics Applications

Junha Roh

Co-chairs of the Supervisory Committee:

Dieter Fox

Computer Science and Engineering

Ali Farhadi

Computer Science and Engineering

As increasing numbers of robots for helping humans are developed and deployed, it is important to have effective interaction with humans and other agents. Robots cooperating with humans are asked to be robust for safety and often desired to be explainable. On top of the challenges that general robots have, communication with humans needs the ability to understand the human direction, conditioning robots' behavior as well as their internal states or observations. Additionally, training a model for robotics applications requires costly environments for collecting data or active simulations to generate data while recent approaches with large models collect the data from the web. It makes the training process expensive, especially for the models with the

capability of using language in specific contexts.

In this thesis, we propose various methods that produce interpretable results using composable sub-models for interactions in robotics applications.

In the first part of the thesis, we propose models for interaction in driving. We develop a model that enables humans to control a vehicle with language instructions such as "turn left and then turn right." The model consists of two sub-models: a high-level policy to translate the language instruction to a sequence of sub-tasks and a low-level policy to control the vehicle to accomplish each sub-task. We also propose a model that predicts future trajectories of agents on a four-way intersection where it tackles another important form of interaction for autonomous vehicles. The first sub-model predicts destinations and topologically invariant description of the order of executions from reference trajectories. Given the abstract description of the scene, the second sub-model predicts multiple future trajectories.

In the second part, we propose visual grounding models in 3D pointcloud and RGBD images as essential tasks for robot navigation and human-robot interaction. The task is to identify the referred object given language description either from a reconstructed 3D scene or a pair of RGBD images. The model for 3D visual grounding extends a large language model to a spatial-language model for identifying the target object. The model for RGBD visual grounding combines a pre-trained 2D visual grounding model and a 3D bounding-box proposal model. They can leverage the high generalization performance of large models, achieve comparable numbers to state-of-the-art methods, and produce interpretable intermediate results.

Acknowledgements

First, I would like to thank my advisors Ali Farhadi and Dieter Fox for their support and guidance throughout my Ph.D. They helped me learn how to conduct research and communicate ideas with other people. Even with the project without progress for a long time, they encouraged me to keep up and sincerely helped me go through hard times. Additionally, I want to acknowledge my committee members, Maya Cakmak and Fei Xia.

I would also like to thank my lab mates who shared research questions and ideas and emotionally supported me throughout my time at the University of Washington. Everyone in our lab was inspiring and has a lot of things to learn. I really enjoyed a coffee chat with my lab mates.

Next, I would like to thank the collaborators I have worked with at the University of Washington and elsewhere for providing perspectives and helping me grow as a collaborator. Thanks to Karthik Desingh, Christoforos Mavrogiannis, Chris Paxton, and Andrzej Pronobis for collaboration and mentorship.

Finally, I would like to thank my family, my friends, and my cat for their support and encouragement. A special thanks to Jinnie who considerately supported me and helped me during my Ph.D.

DEDICATION

To my family and Jinnie.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Challenges of robots with human

Despite the effort to bring robots outside of industrial environments, the adoption of robots at home or office is not quite common and it is due to plenty of challenges in building general-purpose robots.

Generalization is one important aspect of almost every robotics task. For instance, in a problem of perception, the robot has to deal with new object instances in various environments. Many vision tasks rely on static datasets (Deng et al., 2009; Lin et al., 2014; Cordts et al., 2016; Geiger et al., 2012) but models trained from one dataset consistently suffer from degenerated performance on another dataset (Taori et al., 2020). Even if we ship a robot with a near-perfect perception capability on specific datasets, it will often fail in recognizing new objects in new environments. For instance, standard (indoor) 3D detection tasks usually assume 10-30 classes for detection (Dai et al., 2017; Song et al., 2015; Armeni et al., 2016). We cannot expect the model trained from those datasets will perform well anywhere in the real world. In addition, general robots might be asked to do some tasks that they have never learned before. It has to learn to accomplish a new task from its learned policies and it is a task of continual learning (Parisi et al., 2019). We still need to learn how to deal with catastrophic forgetting when it comes to learning a new task without keeping the training data from previous tasks (Kirkpatrick

et al., 2017).

Besides dealing with distributional shifts in data and tasks, robots working with humans need to deal with various interactions at different levels.

First, safety in human-robot interaction requires understanding and prediction of human behavior. Robots cooperating with humans have to defensively predict potential modes of human behavior. Evaluation of the cooperative robot behavior must be different from that of the robot in a restricted and controlled environment as well. Autonomous vehicles are popular examples that need robustness in motion prediction. One of the challenges for autonomous vehicles is to correctly predict the future trajectories of agents on the road including other vehicles, pedestrians, cyclists, and so on. State estimation of the agents is already challenging, but dealing with extremely diverse interactions of the agents to safely respond in driving is a core technology for autonomous vehicles. We can find similar examples such as navigating through crowds easily. Additionally, communicating through behavior should be done on top of understanding. Suppose an autonomous vehicle meets another car in front of a narrow street. Either one of two vehicles should pass the street first and it needs implicit communication by understanding or assuming intentions among possible scenarios.

Second, robots should be capable of explicit communication. Robots should be general enough to handle various forms of communication and the form can vary depending on the task and the context. Suppose we are asking a robot to find and bring a chair for us. If we are at home and the robot already knows everything about the house, the most efficient way of communication can be language, e.g. "bring me a chair." However, if we are in a completely new house, some people may prefer pointing to a location on the map if it exists. For each type of communication, robots should be able to handle multi-modal inputs. It has to understand the relationship between observations and the conditional command from the human and connect the intention of the command to planning and actions. For instance, a language command "bring me a chair" implies a sequence of actions of finding a chair, potentially the shortest one in terms of time, navigation to the chair implicitly chosen, picking it up, and bringing it back to the human. It also

assumes that the robot manages the map of the environment and remembers the configuration of objects. In another example of pointing to a map, the robot has to constantly build another map during exploration, localize itself in the map, and consistently find correspondences between its internal map and the given map from the human. Therefore, it should be able to take various types of commands and connect those to both its internal states and policies.

In the thesis, we focus on tasks arising from various interactions both in implicit and explicit forms.

1.2 What makes the interaction tasks difficult?

As the interaction is an additional task of conditioning on existing tasks or combinations of tasks, it still has to deal with the aforementioned challenges such as generalization.

On top of those challenges, one of the difficulties in interactive tasks is the high cost of data collection. Recent advances in deep learning highly rely on large datasets such as (Thomee et al., 2016; Changpinyo et al., 2021; Foundation; Kuznetsova et al., 2020). CLIP (Radford et al., 2021) showed that larger models trained with largely crawled pairs of text and image can perform and generalize better than other models in vision and NLP various tasks. Various large models (Ramesh et al., 2022; Alayrac et al., 2022; Saharia et al., 2022) for text-image generation have shown realistic image generation ability recently. Unified vision-language models (Wang et al., 2022a; Diao et al., 2022; Wang et al., 2021) were able to solve multiple vision-language tasks from a single, pre-trained model. Large datasets are usually collected from the web or specific web services and refined as statics collection of data. Many of them are already on the web as a free resource and collectors may focus on consistently curating data and labeling.

On the contrary, many interactive tasks cannot simply leverage data on the web but they need to generate data from environments. The data generated from an environment are often dependent on the specific environment. Let us consider a vision-language navigation (Anderson et al., 2018d) task as an example. It was proposed to navigate in a virtual indoor environment

by following language instructions. For data collection, they asked human annotators to annotate language commands by navigating some predefined paths from the virtual indoor environment. By looking at a sequence of images from given paths, annotators may mention some objects and their configuration (e.g. "chairs next to the black table"), spaces, or directions related to objects or rooms (e.g. "go through the kitchen"), or actions directly (e.g. "turn left"). All the language directions are tightly coupled with the specific path and the environment and hard to augment annotations automatically. In addition, directions can be dependent on the agent's configuration in the environment such as state space and action space. If the robot can only turn by 90 degrees, people may refrain from using language instruction to turn 45 degrees or other behaviors that the robot cannot simply do. The dependency on the specific configuration of the environment makes it hard to transfer the data to another environment.

It also makes learning hard. In the example of vision-language navigation, language instructions that are path-dependent can make the robot confused when it deviates from the original trajectory. A slight deviation from the training data distribution makes it hard to learn and it is hard to provide augmented data which is common in vision tasks.

The data generation process is also expensive. Interactions involving humans and the agent in the data generation loop can be time-consuming and costly. For instance, collecting interactions of agents on the road costs many drivers with fully-equipped vehicles. Interactive trajectories are time-sensitive and scene-dependent as well. On the other hand, embodied AI tasks have been proposed along with simulated environments for the generation of interactive data (Anderson et al., 2018a; Shen et al., 2021; Shridhar et al., 2020a; Dei, 2020; Kol, 2017). Assets for realistic environments, gathered by human design or 3D reconstruction, are also expensive and hard to scale up to the level of the web-scale diversity. Adding human intervention to the tasks for interaction further limits scalability.

1.3 Dissertation Overview

This dissertation explores a few aspects of interactions in robotics applications. Models proposed in various tasks do not share one common backbone or architecture but they are all ***interpretable*** as they are producing interpretable intermediate results from composable sub-models. We believe interpretability from our models can provide analysis of the errors in the tasks and help develop better models for general-purpose robots. It is of importance, especially in interactive applications where safety is the top priority. The modular architecture is also reducing the complexity of the task. It is useful when the interaction happens within a long time horizon.

1.3.1 Part 1: interactions in driving

In the first part of the thesis, we develop models for interactions in driving. Specifically, in Chapter 2, we propose a model to predict the trajectories of agents in driving. We focus on a 4-way intersection without traffic lights to study the extreme case of interaction between agents; implicit communication between agents is encouraged without traffic lights and the complexity of interaction becomes high when a multi-way road structure is provided. Instead of directly predicting trajectories, we propose to predict the concise description of interaction first. We call the description as *mode* which tells you which agent crosses the road first and where its destination is. It provides a topologically invariant scene description that is not sensitive to perturbations of trajectories. Then, from the reference trajectories and the predicted modes, the future trajectories are predicted. The model employ GraphNet (Battaglia et al., 2018) for effective modeling of interactions of agents and VAE (Higgins et al., 2017) for producing multiple candidates at inference. Two sub-models are in charge of predicting modes and trajectories and each task becomes simpler than the original task. Furthermore, modes are interpretable; it abstractly describes the predicted scene interactions and it enables tracking down the source of error.

In Chapter 3, we deal with a different type of interaction in driving: language grounding. Language provides an effective way to take high-level control of an autonomous vehicle. For instance, “turn left and then right”

will make the car take a left turn at the first intersection and a right turn at the second. Given language instructions, the model produces a translated sequence of sub-tasks as intermediate representations. Another module within the model then controls the car to accomplish each sub-task until it finishes the original task. Similar to the *modes* in the trajectory prediction model, a predicted sequence of sub-tasks provides an interpretation of how the model understands the instruction and generates a plan, reduces the overall complexity of the task, and debugs the error. For instance, by comparing sub-tasks to the instruction, we can track whether the high-level model or the low-level model failed.

1.3.2 Part 2: language grounding in indoor navigation

In the second part of the thesis, we focus on exploring language grounding in indoor navigation. Specifically, we study the visual grounding task in 3D for robot navigation. Referring object plays an important role in various indoor tasks since many tasks involve interaction with objects in the environment and the key step is to find the object being referred to (Das et al., 2018a). For interactive robotics applications, 3D visual grounding is an important task. ReferIt3D (Achlioptas et al., 2020) proposed a benchmark of measuring ability of identifying 3D object from grounding language and a reconstructed 3D scene based on (Dai et al., 2017). The task can be challenging since the model has to understand various language expressions that cannot be completely covered by utterances in the training data. In comparison to the recent datasets built for vision or language tasks, 3D datasets (Dai et al., 2017; Song et al., 2015; Armeni et al., 2016) are small and as well as the size of the language annotation on the 3d datasets (Achlioptas et al., 2020; Chen et al., 2020a). Instead of training an end-to-end model from a small dataset, we again explore compositions of sub-models which can connect models trained with web-scale datasets to other models for the tasks of interest.

In Chapter 4, we propose a model to identify the mentioned object from the language grounding given a reconstructed scene in 3D with object bounding boxes. Our proposed model consists of two sub-models: an object classification model and a spatial-language model for reference. First, the object classification

model predicts a class label from pointcloud in each bounding box. It decomposes the perception error from the overall task error. Then the predicted class labels augmented with the spatial information and the language description are passed to the large language model. It becomes a spatial-language model for referred object identification; it does not only attend to the words in class labels given language description, but it also leverages spatial information from each object bounding box to specify the target object. The modular architecture provides the benefits of interpretability, and lower complexity as it does in part 1. By using a pre-trained language model as a backbone, our model can handle vocabularies that were not in the training data.

In Chapter 5, we propose a model for the visual grounding in RGBD images. While visual grounding in fully reconstructed 3D scenes can be useful for robots in known environment, it is not suitable for partially observed scenes or newly visited, unknown environment. In addition, view-dependent utterances for fully observed scenes may not occur in real-time robot navigation. They often confuse the model in training since the original viewpoint information is missing in the dataset. Visual grounding in RGBD images can resolve the confusion and be more suitable for robot navigation. Therefore, we propose a model for visual grounding in RGBD images. Similarly to the model composition in Chapter 4, our model is composed of a 2D visual grounding model and a 3D bounding-box estimation model. We employ the 2D visual grounding model from (Wang et al., 2022a) and fine-tune the model with dataset in (Liu et al., 2021a). It can leverage the generalization benefit of the large model with huge datasets and does not get affected by quality of depth sensors. Then we trained 3D bounding-box proposal model based on (Ma et al., 2022); it takes a pointcloud cropped by the predicted 2D bounding-box from the 2D visual grounding model and predicts a 3D bounding-box in class-agnostic fashion. While we achieve higher accuracy than (Liu et al., 2021a), out model can analyze the effect of each part on the error.

Finally, we discuss future directions of the research and summarize the thesis in the last chapter.

Chapter 2

Multimodal Trajectory Prediction via Topological Invariance for Navigation at Uncontrolled Intersections

2.1 Introduction

The widespread interest in autonomous driving technology in recent years (Baltic et al., 2019) has motivated extensive research in multiagent navigation in driving domains. One of the most challenging driving domains (US Department of Transportation, Federal Highway Administration, 2018) is the *uncontrolled* intersection, i.e., a street intersection that features no traffic signs or signals. Within this domain, we focus on scenarios in which agents do not communicate explicitly or implicitly through e.g., turn signals. This model setup gives rise to challenging multi-vehicle encounters that mimic real-world situations (arising due to human distraction, violation of traffic rules or special emergencies) that result in fatal accidents (US Department of Transportation, Federal Highway Administration, 2018). The frequency and severity of such situations has motivated vivid research interest in uncontrolled intersections (Isele et al., 2018; Mavrogiannis et al., 2020; Fridovich-Keil et al., 2020).

In the absence of explicit traffic signs, signals, rules or explicit communication among agents, avoiding collisions at intersections relies on the ability of agents to predict the dynamics of interaction amongst themselves. One prevalent way to model multiagent dynamics is via trajectory prediction. However, multistep multiagent trajectory prediction is NP-hard (Cooper, 1990), whereas the sample complexity of existing learning algorithms effectively prohibits the extraction of practical models. Our key insight is that the geometric structure of the intersection and the incentive of agents to move efficiently and avoid collisions with each other (rationality) compress the space of possible multiagent trajectories, effectively simplifying inference.

We explicitly account for this structure by collapsing the space of multiagent trajectories at an intersection into a finite set of *modes* of joint behavior. We represent modes using a notion of topological invariance (Berger, 2001): all trajectories belonging to the same mode leave the same topological signature (Fig. 2.2a depicts an example). Leveraging the flexibility and expressiveness of Graph Neural Networks (Battaglia et al., 2018), we present a trajectory-prediction architecture that biases inference towards high-likelihood modes. We show that our architecture, entitled *Multiple Topologies Prediction* (MTP), outperforms a state-of-the-art baseline (MFP (Tang and Salakhutdinov, 2019)) by 78.24% in terms of reconstruction accuracy on a dataset of challenging intersection crossing tasks. Based on MTP, we design MTPnav, an optimization-based planning framework (see Fig. 2.1) which achieves time-efficient, collision-free navigation across a variety of challenging multiagent intersection-crossing scenarios in the CARLA simulation environment (Dosovitskiy et al., 2017).

2.2 Related Work

In recent years, the end goal of autonomous driving has motivated extensive research on prediction and planning for navigation in multiagent domains. Considerable attention focuses on the task of multiagent trajectory prediction. Earlier works in the area employed physics-based models (Ammoun and Nashashibi, 2009), hidden Markov models (Firl et al., 2012), and dynamic Bayesian networks (Gindele et al., 2015). With the advent of deep learning

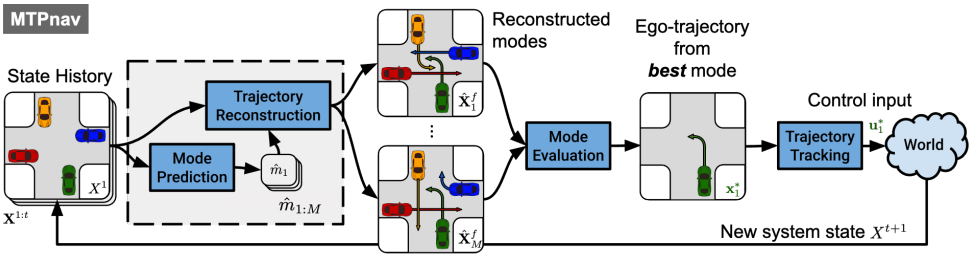


Figure 2.1: Overview of our decentralized navigation planner, MTPnav: The ego-vehicle (shown in green) invokes MTP (Multiple Topologies Prediction), our trajectory prediction mechanism, which determines M *modes* of highly likely future multiagent behavior, and reconstructs trajectory representatives for them. These modes are then evaluated based on the quality of their representatives incorporating considerations such as efficiency, clearance, likelihood, and smoothness. The ego-trajectory from the best mode is then passed to a controller which tracks the first waypoint.

and the availability of several public trajectory datasets (Chang et al., 2019; Houston et al., 2020), many recent works are using Recurrent Neural Networks (Gupta et al., 2018; Deo and Trivedi, 2018; Tang and Salakhutdinov, 2019) and graph-based models (Li et al., 2019; Chandra et al., 2019; Li et al., 2020). For instance, Deo and Trivedi (2018) use social pooling layers along with maneuver-based trajectory generation by labeling trajectories with maneuvers, which leads to multimodal predictions. Li et al. (2019) make use of graph-convolutional networks combined with an LSTM-based encoder-decoder to account for inter-vehicle interactions. A recent line of work employs models for *multimodal* trajectory prediction in an effort to account for the uncertainty in multiagent navigation (Schmerling et al., 2018; Mavrogiannis and Knepper, 2020; Tang and Salakhutdinov, 2019; Liang et al., 2019; Monti et al., 2020). For example, Tang and Salakhutdinov (2019) propose the Multiple Futures Prediction (MFP) method for multimodal multiagent trajectory predictions by capturing different behaviors present in the data using discrete latent variables in an unsupervised manner. Our work also embraces the virtues of multimodality but differs by incorporating a mathematically sound, compact and interpretable formalism for representing multimodality using concepts from topology to drive the learning process.

In parallel, relevant works on planning and control are focusing on integrating models of multiagent dynamics in the decision making. Some works employ implicit models of interaction, such as risk level sets (Pierson et al., 2018), deep reinforcement learning (Isele et al., 2018), or imitation learning (Rhinehart et al., 2018). While such approaches may yield desirable performance in interesting driving scenes, they abstract away the richness of interaction that unfolds in driving domains, thus showing limited applicability, and they are often further constrained by data dependencies. In an effort to explicitly model interaction, some works model multiagent dynamics as *games* by employing tools from game theory (Cleac'h et al., 2020; Fridovich-Keil et al., 2020). While game-theoretic approaches elegantly capture the richness of interactions in driving domains, the challenge of computing (Nash) equilibria and the intractability of scaling to large numbers of agents limits their applicability. Other works have employed topological abstractions such as braids (Mavrogiannis et al., 2020; Artin, 1947) to enable agents to reason over multiagent collision-avoidance strategies. Topological braids capture critical events in multiagent navigation, such as the order of passing maneuvers, in a compact and interpretable fashion (Mavrogiannis and Knepper, 2019; Mavrogiannis et al., 2017); however, their lack of analytic descriptions complicates the construction of generalizable prediction models and so limits their applicability. Instead of braids, our work leverages a notion of topological invariance (Berger, 2001) with analytic descriptions that enable the use of data-driven methods. This allows us to retain the benefits of topological reasoning while achieving state-of-the-art performance in challenging and realistic multiagent intersection-crossing tasks.

2.3 Problem Statement

Consider a four-way uncontrolled intersection (see Fig. 2.2a), where no explicit rules (e.g., right-turn priority) or signals (e.g., traffic lights) are set in place to regulate traffic. Assume that $1 < n \leq 4$ non-communicating agents with *simple-car* kinematics are navigating. Denote by $x_i \in \mathcal{X} \subseteq SE(2)$ the state of agent $i \in \mathcal{N} = \{1, \dots, n\}$ with respect to (wrt) a fixed reference frame. Agent i

is tracking a path $\tau_i : I \to \mathcal{X}$, for which it holds that $\tau_i(0) = s_i$ and $\tau_i(1) = d_i$, where $s_i, d_i \in \mathcal{X}$ are states lying at different sides of the intersection (i.e., right, left, up, down) and $I = [0, 1]$ is a path parameterization. At timestep t (assume a fixed time discretization dt), agent i executes a policy $\pi_i : \mathcal{X} \to \mathcal{U}_i$ that generates controls $u_i \in \mathcal{U}_i$ (speed and steering angle) contributing progress along τ_i while avoiding collisions with $j \neq i$. Agent i is not aware of the intended path τ_j, the destination d_j, or the policy π_j of agent $j \neq i \in \mathcal{N}$.

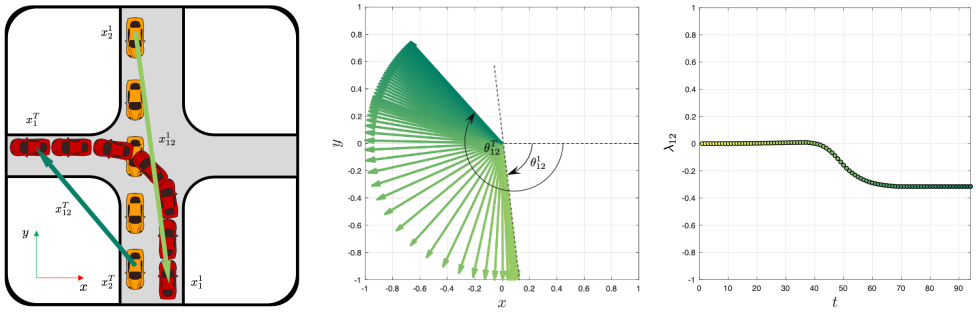
Let agent #1 be the *ego* agent. The ego agent perfectly observes the current system state $X^t = (x_1^t, \ldots, x_n^t) \in \mathcal{X}^n$ and has access to the complete system state history $\mathbf{X}^{1:t} = (X^1, \ldots, X^t)$. Our goal is to design a policy π_1 that enables the ego agent to follow collision-free and time-efficient intersection crossings despite the uncertainty resulting from the constraint of no communication.

2.4 Navigation with Multiple Topologies Prediction

We describe a policy π_1 for decentralized navigation at uncontrolled intersections. Our policy is based on a data-driven mechanism for multimodal multiagent trajectory prediction. Our mechanism leverages a topological formalism that effectively compresses the space of possible multiagent trajectories into a set of *modes* of multiagent behavior. This allows us to guide trajectory prediction towards high-likelihood modes. At planning time, our policy evaluates a set of high-likelihood multiagent trajectory alternatives and follows the ego trajectory from the one with minimum cost.

2.4.1 A Topological Formalism of Modes for Navigation at Intersections

Consider two agents navigating a four-way intersection. Denote by $x_{12}^t = x_1^t - x_2^t$ a vector–referred henceforth as the *winding vector*– expressing the relative location of agents and by θ_{12}^t its angle with respect to a global frame at timestep $t \geq 1$ (see Fig. 2.2a). From time t to $t+1$, agents' displacement, $\Delta x_{12}^t = x_{12}^{t+1} - x_{12}^t$, results in a rotation $\Delta\theta_{12}^t = \theta_{12}^{t+1} - \theta_{12}^t$. Taking the sum of these rotations from the beginning of the execution ($t = 1$) until the end ($t = T$), we derive a *winding*



(a) Top view of agents' trajectories.

(b) Unit winding vector trajectory.

(c) Winding number convergence.

Figure 2.2: Identifying modes of intersection crossing: (a) top view of agents' trajectories, indicating the initial and terminal winding vectors; (b) unit winding vector trajectory (darkness increases with time); and (c) winding number convergence. Any execution of this scenario in which the orange agent passes before the red agent converges to a negative winding number.

number (Berger, 2001) characterizing the total relative rotation of both agents:

$$\lambda_{12} = \frac{1}{2\pi} \sum_{t=1}^{T-1} \Delta\theta_{12}^{t}. \tag{2.1}$$

The value λ_{12} tracks the signed number of times agents $1, 2$ "revolved" around each other from time $t = 1$ to $t = T$.

The winding number is a *topological invariant*: fixing agents' endpoints s_1, s_2, d_1, d_2, any topology-preserving deformations of agents' trajectories (continuous trajectory deformations not involving penetrations or cuts) would be identified using the same winding-number value λ_{12}. Importantly, the sign of the winding number $w_{12} = \text{sgn}(\lambda_{12})$ indicates the side on which the two agents pass each other. A right-side passing corresponds to a clockwise rotation of the winding vector x_{12}, yielding a positive winding number $\lambda_{12} > 0$; a left-side passing corresponds to a counterclockwise rotation of the winding vector x_{12}, yielding a negative winding number $\lambda_{12} < 0$. Fig. 2.2 illustrates the machinery of the winding number at an intersection example with two agents.

Extending this idea to a scene with n agents with endpoints $S = (s_1, \ldots, s_n)$

and $D = (d_1, \ldots, d_n)$, we define the *topology* of an intersection-crossing task as:

$$W = \left(\ldots, w_{ij}, \ldots\right), ij \in \mathcal{N}_e, \tag{2.2}$$

where $\mathcal{N}_e$ is the set of all pairs formed among n agents. Further, we define a *mode* of intersection crossing as a tuple $m = (S, D, W)$. At an intersection-crossing task, a mode encodes *where* agents are heading (destination D) and *how* they are getting there (topology W) from their initial state, S.

2.4.2 Multiple Topologies Prediction

Leveraging the formalism of modes, we build *Multiple Topologies Prediction* (MTP), a multimodal trajectory prediction framework. MTP biases trajectory inferences towards a set of high-likelihood modes. Given a recent state history $\mathbf{X}^p = \mathbf{X}^{t_p+1:t}$ of $h_p = t - t_p$ timesteps in the past, MTP: 1) determines a set of highly likely modes of future multiagent behavior $\hat{\mathbf{m}} = \{\hat{m}_1, \cdots, \hat{m}_M\}$; 2) predicts a corresponding set of trajectory representatives $\hat{\boldsymbol{\mathcal{X}}} = \{\hat{\mathbf{X}}_1^f, \ldots, \hat{\mathbf{X}}_M^f\}$ where $\hat{\mathbf{X}}_l^f = \mathbf{X}_l^{t+1:t_f}$, $l = 1, \ldots, M$, and $h_f = t_f - t$ is a horizon into the future. We construct models for the two stages of prediction using a GraphNet architecture (Battaglia et al., 2018). We first describe how to reconstruct a trajectory for a desired mode m and then describe a technique for generating high-likelihood modes.

Mode-Conditioned Trajectory Reconstruction

Our model represents the world state at time t, X^t as a directed graph g^t (see Fig. 2.3). The graph g^t consists of a set of node attributes $V = \left\{\mathbf{v}_i^t : i \in \mathcal{N}\right\}$, a set of edge attributes and corresponding node indices $E = \left\{\left(\mathbf{e}_k^t, s_k, r_k\right) : k \in \mathcal{N}_e, s_k, r_k \in \mathcal{N}\right\}$, and a global attribute $\mathbf{u}^t$. We set the position and velocity of agent i at time t to be a node attribute, i.e., $\mathbf{v}_i^t = \left(x_i^t, \Delta x_i^t\right)$, and the relative position of the k-th pair of agents at time t to be an edge attribute, i.e., $\mathbf{e}_k^t = \Delta x_{s_k r_k}^t$. We initialize the global attribute with a zero vector since the model does not have a dynamic global context. However, we leave the global attribute for the model to have ability to keep aggregated information in the

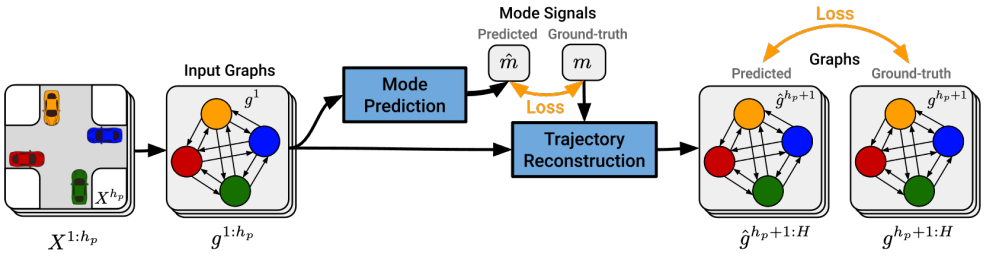


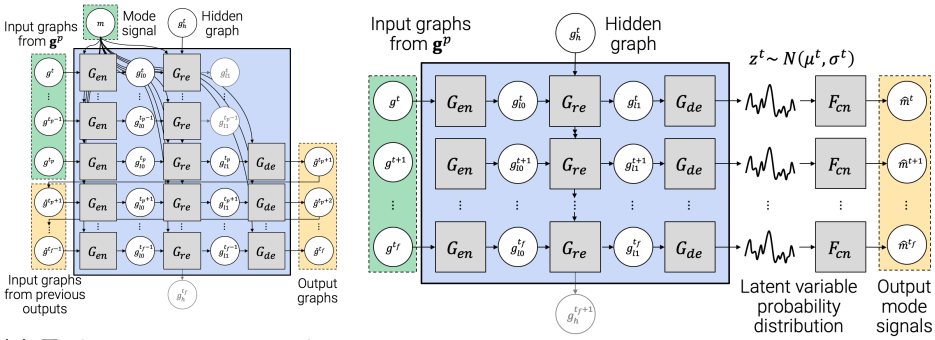
Figure 2.3: Overview of model training. We train a mode prediction and a trajectory reconstruction model, using the GraphNet (Battaglia et al., 2018) framework. Encoding the system state as a graph, we supervise predicted mode signals and reconstructed trajectories.

attribute.

Our model takes as input the world state g^t and outputs a predicted world state $\hat{g}^{t+1}$. Computation takes place within a network of interconnected blocks: an encoder block G_{en}, a decoder block G_{de}, and a recurrent block G_{re} (see Fig. 2.4a), introduced to handle temporal data effectively. The block G_{en} encodes the graph g^t into a *latent* graph g^t_{l0}. Then, the recurrent block G_{re} modifies it into g^t_{l1} and passes it to G_{de}, which returns an updated graph $\hat{g}^{t+1}$ that represents the predicted state of the world in Cartesian coordinates. Within each block, the following computations take place (we set $g = g^t$, $g' = g^{t+1}$, and assume $i \in \mathcal{N}$, $k \in \mathcal{N}_e$ for brevity):

$$
\begin{aligned}
\mathbf{e}'_k &= \mathbf{e}_k + \phi_e\left(\mathbf{e}_k, \mathbf{v}_{s_k}, \mathbf{v}_{r_k}, \mathbf{u}, \mathbf{w}_k\right) & \bar{\mathbf{e}}'_i &= \rho_{e\to v}\left(E'_i\right) & E'_i &= \left\{\left(\mathbf{e}'_k, s_k, r_k\right), r_k = i\right\} \\
\mathbf{v}'_i &= \mathbf{v}_i + \phi_v\left(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}, \mathbf{d}_i\right) & \bar{\mathbf{e}}' &= \rho_{e\to u}\left(E'\right) & V' &= \left\{\mathbf{v}'_i\right\} \\
\mathbf{u}' &= \mathbf{u} + \phi_u\left(\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u}\right) & \bar{\mathbf{v}}' &= \rho_{v\to u}\left(V'\right) & E' &= \left\{\left(\mathbf{e}'_k, s_k, r_k\right)\right\},
\end{aligned}
\tag{2.3}
$$

where ϕ and ρ are respectively update and aggregation functions, $\mathbf{d}_i \in \{0,1\}^4$, $\mathbf{w}_k \in \{0,1\}^2$ are one-hot encoded vectors of the destination of agent i and of the sign of the winding number of edge k respectively. We model update functions ϕ_e, ϕ_v, ϕ_u using fully connected layers with ReLU and layer normalization and aggregation functions $\rho_{e\to v}$, $\rho_{e\to u}$, $\rho_{v\to u}$ as means over the updates of their corresponding sets. In the recurrent block, G_{re}, we use a single-layer GRU (Gated Recurrent Unit) (Cho et al., 2014) to efficiently infer the sequence of vehicle positions.



(a) Trajectory reconstruction model.

(b) Mode prediction model.

Figure 2.4: Internal topology of trajectory reconstruction (a) and mode prediction (b) models. At time t, the trajectory reconstruction model takes as input the graph g^t, a hidden graph g_h^t, and a mode m and generates an output graph $\hat{g}^{t+1}$. This procedure is repeated up to timestep t_f, yielding a trajectory in the form of a sequence of graphs $\hat{\mathbf{g}}^f = \hat{g}^{t+1:t_f}$. The mode prediction model takes as input the graph g^t and a hidden graph g_h^t and produces a probability distribution over a latent variable $\mathbf{z}^t$. A sample is drawn from the distribution and transformed into a predicted mode $\hat{m}^t$. Variables in green shades indicate input from the ground-truth in training, variables in yellow shades are the outputs, and grey variables are discarded. At inference, graphs up to current timestep t, $g^{t_p:t}$, are fed to the mode-prediction model to estimate modes, $\{\hat{m}_{1:M}\}$. Given estimated modes and the graphs, the trajectory-reconstruction model predicts future trajectories.

Sampling High-Likelihood Modes

The trajectory reconstruction model takes a mode m as input. The number of modes is exponential in the number of agents, and not all modes are equally likely. To address this issue, we sample high-likelihood modes by adapting our reconstruction architecture (see Fig. 2.4a) into a mode prediction model (see Fig. 2.4b). The model fits a Gaussian distribution over a latent variable $\mathbf{z}^t \sim \mathcal{N}\left(\mu^t, \sigma^t\right)$. This variable is transformed into a mode via fully connected layers, i.e., $\hat{m}^t = F_{cn}(\mathbf{z}^t)$. This process resembles a variational auto-encoder (Kingma and Welling, 2014); however, our model is supervised to generate conditional signals (modes) instead of reconstructing original input signals (configurations).

Following High-Quality Modes

Based on MTP, we build MTPnav, a cost-based planner (see Fig. 2.1). At planning time t, MTPnav invokes MTP, which returns a set of M high-likelihood trajectory predictions $\mathcal{X}$, corresponding to high-likelihood modes $\mathbf{m}$ of multiagent behavior. These predictions are then evaluated with respect to a trajectory cost $J : \mathcal{X}^n \to \mathbb{R}$:

$$J(\mathbf{X}) = w_{sm}J_{sm}(\mathbf{X}) + w_{ref}J_{ref}(\mathbf{X}) + w_{col}J_{col}(\mathbf{X}) + w_pJ_p(\mathbf{X}), \tag{2.4}$$

where J_{sm} penalizes non-smooth trajectories, J_{ref} penalizes deviations from the ego-vehicle reference trajectory, J_{col} penalizes collisions, J_p penalizes low-likelihood trajectories, and $w_{sm}, w_{ref}, w_{col}, w_p$ are corresponding weights (refer to Appendix, Sec. A.1 for detailed formulation). The trajectory of minimum cost $\mathbf{X}_*$ is selected and passed to a tracking controller, which returns a control input u^* for the agent to execute.

2.5 Evaluation

We evaluate the performance of the proposed framework in two ways. First, we verify the ability of MTP to produce high-quality, mode-conditioned trajectory reconstruction. Next, we illustrate the value of MTPnav for tasks involving

Scenario	Two Agents		Three Agents		Four Agents	
Metric	minADE (m)↓	minFDE (m)↓	minADE (m)↓	minFDE (m)↓	minADE (m)↓	minFDE (m)↓
MTP	**0.301 ± 0.346**	**0.611 ± 0.893**	**0.304 ± 0.459**	**0.717 ± 1.366**	**0.264 ± 0.410**	**0.588 ± 1.067**
MTP-fc	0.523 ± 0.440	1.063 ± 1.020	0.456 ± 0.612	1.024 ± 1.743	0.334 ± 0.342	0.704 ± 0.996
GRU	0.929 ± 0.566	1.578 ± 1.159	1.176 ± 0.727	2.119 ± 1.870	0.958 ± 0.551	1.760 ± 1.408
MFP	0.834 ± 0.969	0.825 ± 1.318	1.603 ± 0.981	1.504 ± 1.857	1.556 ± 0.707	1.444 ± 1.758

Table 2.1: Prediction accuracy (mean, standard deviation) measured as minADE and minFDE (average and final displacement error) for models trained on datasets with two, three, and four agents. MTP performs significantly better than other models (Wilcoxon signed-rank test, $p < 0.001$).

navigation at uncontrolled intersections through a simulated case study on a series of challenging intersection-crossing scenarios.

2.5.1 Datasets

Existing open datasets (Chang et al., 2019; Houston et al., 2020) do not contain organized and sufficient amounts of interactions at uncontrolled intersections. For this reason, we generated a series of simulated datasets of challenging intersection crossings. Our data-generation pipeline consists of: (1) extracting realistic path references τ_i, $i \in \mathcal{N}$ following non-holonomic car kinematics using CARLA (Dosovitskiy et al., 2017), and (2) executing these references under a variety of different behaviors and scenarios using a custom, computationally efficient, purely kinematic simulator. Our simulator controls vehicles in a centralized fashion using a priority system that provides acceleration/deceleration signals based on a time-to-collision heuristic and according to a set of parameters representing agents' behavioral models (desired speed, acceleration, deceleration, etc.). The vehicles track their paths using PID-based steering and speed controllers. This pipeline lets us combine the realism of CARLA with the ability to generate data efficiently and control scenarios and agent behaviors. This strategy also provides the ability to deploy our models directly to CARLA without re-training or fine-tuning to evaluate navigation performance (Sec. 2.5.3). Overall, we generate three diverse datasets of the form $\mathcal{D}_n = \{\mathbf{X}_1^n, \ldots, \mathbf{X}_{|\mathcal{D}_n|}^n\}$, containing $|\mathcal{D}_2| = 116k$, $|\mathcal{D}_3| = 1,180M$, and $|\mathcal{D}_4| = 466k$ multiagent trajectories involving $n = 2, 3, 4$ agents, respectively, by methodically varying: (1) agents' destinations D

(spanning the set of combinations), (b) their target speeds ($3-12\text{m/s}$), and (c) their acceleration/deceleration ($1-5\text{m/s}^2$).

2.5.2 Model Training

We train different trajectory-reconstruction and mode-prediction models on each of the datasets $\mathcal{D}_2$, $\mathcal{D}_3$, $\mathcal{D}_4$, following 9:1 train/test splits. We partition the datasets into a set of subtrajectories with a length of $H = h_p + h_f$ where $h_p = 15$, and $h_f = 25$. We transform each subtrajectory into a sequence of graphs $\mathbf{g} = \{g^1, \cdots, g^H\}$ and split it into $\mathbf{g}^p = \{g^1, \cdots, g^{h_p}\}$ and $\mathbf{g}^f = \{g^{h_p+1}, \cdots, g^H\}$. We also compute a corresponding mode signal m for each sequence $\mathbf{g}$. Then we train the trajectory reconstruction model with student-forcing, providing the ground-truth mode signal m and a sequence of history graphs $\mathbf{g}^p$ to produce $\hat{\mathbf{g}}^f$. The mode prediction model is trained with the full ground-truth sequence $\mathbf{g}$ and generates a mode prediction $\hat{m}$. Fig. 2.3 depicts an overview of the training process. We use a mean squared error (MSE) loss for training the trajectory reconstruction model. For training the mode prediction model, we use a loss similar to the one used by β-VAE (Higgins et al., 2017) (setting $\beta = 8$), featuring a cross-entropy component for reconstructing the mode signal given the latent variable and a KL-divergence component keeping the latent distribution close to a Gaussian. Note that at inference time, we only take the predicted mode signal $\hat{m}^{h_p}$ from the mode-prediction model and feed it to the trajectory-reconstruction model to collect $\hat{\mathbf{g}}^f$. See Sec. A.2.2 for mode details on the training process.

2.5.3 Navigation Performance

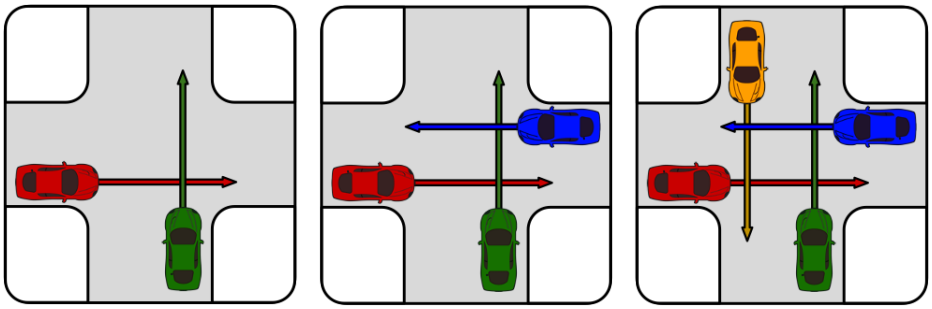
We further evaluate the quality of the MTP predictions through a simulated case study on a decentralized navigation task at a four-way uncontrolled intersection of size 60×60 m on CARLA (Dosovitskiy et al., 2017). We consider three different intersection-crossing scenarios involving $n = 2, 3, 4$ agents, specifically selected to elicit challenging vehicle interactions, as shown in Fig. 2.5. During execution, the ego agent (agent #1, the bottom agent in Fig. 2.5) is running our framework (MTPnav), while other agents are running

the CARLA (Dosovitskiy et al., 2017) Autopilot controller.[1] We consider two different conditions: *Easy* and *Hard*. *Easy* consists of *Cautious* and/or *Aggressive* agents with target speeds sampled uniformly from $[10, 25]$ and $[30, 45]$ (km/h), respectively. *Hard* consists of *Aggressive* and/or *Normal* agents with target speeds sampled from $[30, 45]$ and $[20, 40]$ (km/h), respectively, where *Cautious*, *Normal* and *Aggressive* are behavioral agents provided by CARLA. Uniformly sampling from these ranges, we construct 50 random experiments per scenario and condition.

We compare the performance of MTPnav to two baselines: the Autopilot (homogeneous case, serving as a reference for the difficulty of the task) and a purely reactive Model Predictive Controller (MPC) that treats other agents as obstacles. We employ using two metrics: (1) collision frequency $\mathcal{C}(\%)$ and (2) the time the ego-agent took to reach its destination $\mathcal{T}$ (s). If the ego-agent collides, we mark the trial as *in-collision* and assign it a time penalty of $30s$ corresponding to the time-to-destination for a *Cautious* agent with constant speed of $10km/h$. Trials that did not terminate by the $30s$ mark are also marked as in-collision.

Table 2.2 shows the performance of MTPnav. We observe that MTPnav outperforms the baselines across all scenarios and interaction settings in terms of both time and collision frequency. In particular, its collision frequency is close to zero for scenarios with two and three agents, and significantly lower in scenarios involving four agents. We also see that it is significantly more time efficient, especially in the *Hard* scenarios. Overall, we attribute MTPnav's performance to its ability to exploit the domain structure: MTPnav agents are capable of rapidly adapting to the dynamic environment by foreseeing the multiagent interaction dynamics. We see a significant increase in collisions for scenarios involving four agents. This could be attributed to the steep increase in domain complexity (for reference, the number of possible modes is 162) but also possibly to the relatively small four-agent dataset that we employed.

[1] Autopilot is a widely available baseline that is part of the CARLA simulator (Dosovitskiy et al., 2017).



(a) Two-agent scenario. **(b)** Three-agent scenario. **(c)** Four-agent scenario.

Figure 2.5: Navigation scenarios considered in our evaluation on CARLA.

Scenario		Two Agents		Three Agents		Four Agents	
Interaction		Easy	Hard	Easy	Hard	Easy	Hard
Autopilot	$\mathcal{T}(s)$	23.37 ± 6.01	26.00 ± 7.52	23.01 ± 5.78	28.69 ± 3.96	22.16 ± 7.56	28.41 ± 4.31
	$\mathcal{C}(\%)$	28.00 ± 6.35	78.00 ± 5.86	30.00 ± 6.48	86.00 ± 4.91	40.00 ± 6.93	88.00 ± 4.60
MPC	$\mathcal{T}(s)$	22.37 ± 6.26	26.98 ± 6.43	21.34 ± 6.26	27.78 ± 5.54	21.28 ± 7.98	28.25 ± 4.74
	$\mathcal{C}(\%)$	30.00 ± 6.48	82.0 ± 5.43	28.00 ± 6.35	86.00 ± 4.91	40.0 ± 6.93	88.0 ± 4.60
MTPnav	$\mathcal{T}(s)$	**19.13 ± 3.78**	**14.81 ± 1.56**	**20.85 ± 3.71**	**18.70 ± 0.61**	**19.34 ± 8.03**	**21.36 ± 5.91**
	$\mathcal{C}(\%)$	**2.00 ± 1.98**	**0**	**2.00 ± 1.98**	**0**	**28.00 ± 6.35**	**30.00 ± 6.48**

Table 2.2: Navigation performance measured with respect to collision frequency ($\mathcal{C}$) and time to destination for ego-agent ($\mathcal{T}$). Each entry contains a mean and a standard deviation over 50 trials. red, green and blue entries indicate scenarios in which MTPnav outperformed other methods at significance levels $p < 0.001$, $p < 0.01$, $p < 0.05$ respectively (Wilcoxon signed-rank test).

2.6 Discussion

We introduced a prediction model (MTP) and a planner (MTPnav) for navigation at uncontrolled intersections, leveraging a mathematical insight about the topological structure of intersection crossings. MTP, trained on datasets acquired using a custom-built, lightweight simulator, outperformed a state-of-the-art, trajectory-prediction baseline (Tang and Salakhutdinov, 2019) and enabled MTPnav to exhibit safe and time-efficient behavior across a series of challenging intersection-crossing tasks (without additional fine-tuning or retraining) that were conducted using the high-fidelity simulation engine CARLA. Our findings illustrate the value of reasoning about topological modes as a strategy to guide prediction towards a small representative set of outcomes. Ongoing work involves real-world experiments on a miniature racecar (Srinivasa et al., 2019) to highlight the virtues of our approach for real-world applications.

Our work is not without limitations. Our data generation and evaluation pipelines are constrained by our behavior parametrizations and the mechanics of our simulators. Further, our evaluation experiments are limited by the computational load of running expensive CARLA experiments. Future work will focus on leveraging our empirical insights towards expanding our simulator to capture richer spaces of behaviors for data generation and testing. Finally, the MTPnav framework is proposed as an example, non-optimized incorporation of MTP on a cost-based planner; alternative use cases could apply reinforcement learning or model predictive control frameworks.

Chapter 3

Conditional Driving from Natural Language Instructions

3.1 Introduction

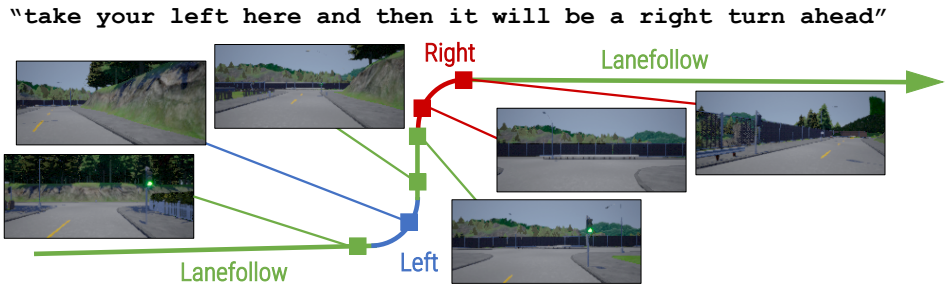


Figure 3.1: Natural language control of self-driving vehicles. The user provides a high-level instruction; the vehicle must then (a) translate natural language into the correct sequence of high-level sub-tasks and (b) correctly execute these, by steering and applying the throttle as appropriate.

Passengers of self-driving cars will expect to interact with their vehicles in the same way as they do with human ride-share drivers. This includes providing specific directions to the precise drop-off locations, preferences about the chosen route, or clarifications in case of ambiguities. Furthermore, a car equipped with a skill to interpret natural-language, able to rely on the help of its user, will be more robust to navigation errors resulting from poor

map coverage and inaccurate information about dynamic road conditions.

As shown in Fig. 3.1, our goal is to learn to understand language instructions, such as "you are going to go a little bit further for one block and make a left at the intersection," and use them to condition a policy that will drive a car safely using only image observations. The problem of end-to-end policy learning for self-driving cars is often formulated as imitation learning (Bojarski et al., 2016; Xu et al., 2017; Codevilla et al., 2018; Müller et al., 2018). We take a hierarchical approach. We use conditional imitation learning to learn policies for steering and throttle control, as in prior work (Codevilla et al., 2018; Müller et al., 2018). However, we also learn a high-level policy, which predicts high-level actions based on language instructions. This leads to a solution able to translate language into actions executed over long time horizons, including navigating multiple streets and turns before reaching the destination.

At the same time, we need to ensure that safety is not compromised, even in presence of incorrect and misleading instructions. A self-driving car will be used by non-experts, who might instruct the car to execute maneuvers that are not safe given the state of the world (e.g. to turn left when no left turn is possible). This is a known problem in language-to-control (Paxton et al., 2019). Moreover, artificial systems often struggle with understanding the intricacies of realistic human language, in particular for such a dynamic task as driving. We provide two pathways to mitigate the harm this can cause: first, our policy is designed to ensure that only safe actions are taken, even when invalid input is given by the passenger. Second, the agent will complete maneuvers, such as driving through an intersection, even if new instructions from the user interrupt the current high-level plan.

We validate our proposed approach using CARLA (Dosovitskiy et al., 2017), an open-source driving simulator. We perform a set of quantitative transfer experiments, showing that our hierarchical models navigate correctly and safely, and can generalize between different environments. Furthermore, we perform a series of ablation tests to study the properties of our model. Finally, we design an interactive experiment, with users instructing the car in real-time with randomly timed and misleading instructions.

To summarize, our core contributions are: (1) an end-to-end policy control-

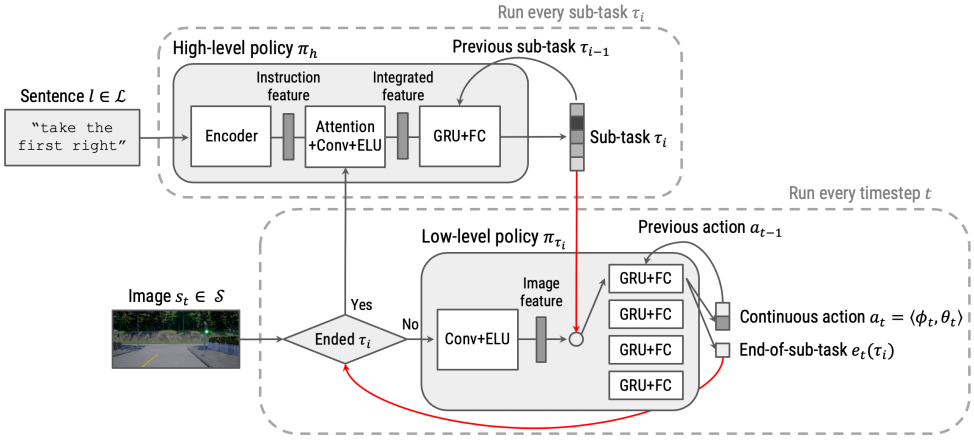


Figure 3.2: The proposed model for language-grounded driving. The model takes an image from the dashboard-mounted camera and a natural language instruction and generates steering and throttle values for control. Gray and red arrows represent flows of tensors and control switching signals, respectively.

ling a self-driving car from language and images; (2) a hierarchical architecture reasoning about both short and long time horizons as well as both high-level inputs and low-level continuous states and actions; (3) an implementation of interactive language-grounded driving robust to misleading user instructions.

3.2 Related Work

Our approach is closely related to work on Visual Question Answering (VQA), a growing area of research in which an agent is trained to move about in a home environment and find the answers to specific questions (Das et al., 2018b; Chung* et al., 2016; Ma et al., 2019b; Ke et al., 2019b; Shah et al., 2018; Anderson et al., 2018d). In particular Das et al. (2018b) proposed Neural Modular Control (NMC), which used a multi-level model to predict a "program" of actions that need to be taken. While closely related, our method uses continuous state and action spaces with a realistic car model, while the vision-language navigation problem is mainly focused on dealing with complicated language expressions with relatively limited discrete state and action spaces.

Other recent work has explored learning driving policies from images. Liang et al. (2018) use a mixture of imitation learning and reinforcement

learning via DDPG (Lillicrap et al., 2015). Codevilla et al. (2018) proposed a system for conditional end-to-end driving. Müller et al. (2018) extends this work, adding a segmentation model which improves generalization performance, but largely keeping the same structure from Codevilla et al. (2018). Paxton et al. (2017) also learn hierarchical policies for driving through intersections, but focus on interacting with other vehicles and do not use images.

Some work has also looked at learning representations based on images and language, but not for driving. Chaplot et al. (2018) proposed a Gated Attention model for learning navigation policies based on images and language, an approach we borrow for our high-level model. Paxton et al. (2019) learned to generate task plans and execute pick-and-place tasks. Blukis et al. (2018) learn a semantic map for navigation and demonstrate on a simulated quadrotor, which can be used to follow natural language instructions.

While we provide a manual decomposition of the task when training models as seen in some previous work (Paxton et al., 2019; Das et al., 2018b), some prior work weakens these assumptions. Shiarlis et al. (2018) propose TACO, which learns to break tasks up based only on a policy sketch. Krishnan et al. (2017) also propose a method for discovery of continuous actions from demonstrations, which could be applied to our problem in the future. Andreas et al. (2017) uses policy sketches together with curriculum reinforcement learning.

3.3 Hierarchical Language-Grounded Driving Model

The goal of our agent is to drive safely, following given language directions and a stream of images from a single camera. Driving requires a very long time horizon, with high-frequency controls but low-frequency decisions. This makes it difficult to directly apply a sequence-to-sequence approach to the problem. Instead, we introduce a hierarchical driving model where a high-level policy π_h chooses a series of sub-tasks $\{\tau_0, \ldots, \tau_{N-1}\}$ to achieve a specified task, and low-level policies π_{τ_i} generate the controls necessary to achieve each sub-task τ_i in sequence. This decomposes the problem into tractable

sub-problems and enables efficient use of short data sequences for training complex, language-conditioned control policies. This approach is similar to that taken in the vision-language navigation problem (Das et al., 2018b), but with increased complexity of low-level controls.

Algorithm 1 shows the pseudo code for execution of our language-grounded driving model. Figure 3.2 shows the architecture of our model. Consider the world $W : \mathscr{S} \times \mathscr{A} \rightarrow \mathscr{S}$ with a continuous state observation $s \in \mathscr{S}$ and a continuous action $a = \langle \phi, \theta \rangle \in \mathscr{A}$, where $\phi \in [0, 1]$ is the normalized throttle control and $\theta \in [-1, 1]$ is the normalized steering angle for the vehicle. We assume that our state observation s_t consists of an image from a dashboard-mounted camera at time t. Then the directions for the language-grounded driving are specified by a natural-language input $l \in \mathscr{L}$, such as "take the next right" or "go straight through this intersection, then turn left." Finally, our problem is defined by learning a policy $\pi : \mathscr{S} \times \mathscr{L} \rightarrow \mathscr{A}$.

We break up the original problem into a two-level hierarchy by introducing a sub-task $\tau \in \mathscr{T}$ where the set of possible sub-tasks $\mathscr{T} = \{\texttt{left}, \texttt{right}, \texttt{straight}, \texttt{lanefollow}\}$. The sub-task `straight` represents the case of going straight through an intersection while `lanefollow` corresponds to a policy ensuring safe lane following outside intersections. We extend $\mathscr{T}$ to include a `finish` token, indicating the end of the entire task: $\hat{\mathscr{T}} = \mathscr{T} \cup \{\texttt{finish}\}$.

With the hierarchical model, our problem is to learn a high level policy $\pi_h : \mathscr{S} \times \mathscr{L} \rightarrow \hat{\mathscr{T}}$, and a corresponding low-level policy $\pi_\tau : \mathscr{S} \times \mathscr{T} \rightarrow \mathscr{A}$. Also, our model detects when the sub-task is achieved. This determines whether the high or the low-level policy takes control. We define the end-of-sub-task signal as $e_t(\tau_i) \in \mathscr{E} = \{\texttt{True}, \texttt{False}\}$ which is an indicator that the current sub-task τ_i is finished and the high-level policy should regain control to generate the next sub-task. Therefore, the revised low-level policy can be expressed as $\pi_\tau : \mathscr{S} \times \mathscr{T} \rightarrow \mathscr{A} \times \mathscr{E}$.

3.3.1 High-level policy

The high-level policy consists of an encoder, the Gated Attention (GA) unit (Chaplot et al., 2018), and a recurrent unit. First, we generate a list

of 50-dimensional GloVe (Pennington et al., 2014) embedding vectors from words in the language instruction. Then the embedding vectors are fed to a single-layer GRU (Chung et al., 2014) and the attention mechanism introduced in (Luong et al., 2015) combines hidden states from the GRU to generate a single instruction feature vector. The image is fed to a series of convolution and ELU (Clevert et al., 2015) layers to generate an image feature vector.

Then the GA takes the instruction and image feature vectors and generates an integrated feature vector. The GA computes attention weights from the instruction to focus on the essential part of the image feature vector. Finally, the integrated feature vector is concatenated with another feature vector from the previous sub-task vector and fed into another single-layer GRU and a fully connected layer. We use a softmax function to generate 5-dimensional sub-task probability distribution for $\hat{\mathscr{T}}$.

3.3.2 Low-level policy

Once the sub-task τ_i is determined, the low-level policy takes control from the high-level policy and generates actions required to achieve the sub-task. First, it converts the input image to an image feature vector by applying a series of convolutional and ELU layers. Then, sub-task probabilities determined by the high-level policy are used to select one of a few sub-task-specific GRU layers. The activated GRU layer combined with an FC layer generates the final 2-dimensional control vector a_t and the end-of-sub-task signal $e_t(\tau_i)$.

The low-level policy remains in control until the end of the sub-task indicated by the $e_t(\tau_i)$ value. In order to make this transition robust to noisy predictions, we require that at least two out of the three recent predictions of $e_t(\tau_i)$ indicate the end of sub-task. In our implementation, we rely on two different low-level policies for predicting a_t and $e_t(\tau_i)$.

3.4 Training and Environment

We used CARLA (Dosovitskiy et al., 2017) to generate data for training and run experiments. CARLA provides road annotations and an auto-pilot function. We relied on the auto-pilot during training.

Algorithm 1: Hierarchical policies for language-grounded conditional driving.

Input: Initial state $s_0 \in \mathcal{S}$
Input: Language direction $l \in \mathcal{L}$
Input: World $W : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$
Input: High-level policy $\pi_h : \mathcal{S} \times \mathcal{L} \rightarrow \hat{\mathcal{T}}$
Input: Low-level policies $\pi_\tau : \mathcal{S} \times \mathcal{T} \rightarrow \mathcal{A} \times \mathcal{E}$

$i, t \leftarrow 0, 0$
$\tau_0 \leftarrow \pi_h(s_0, l)$
$e_0(\tau_0) \leftarrow$ `False`
while $\tau_i \neq$ `finish` **do**
  while $e_t(\tau_i) ==$ `False` **do**
    $a_t, e_t(\tau_i) \leftarrow \pi_{\tau_i}(s_t)$
    $s_{t+1} \leftarrow W(s_t, a_t)$
    $t \leftarrow t + 1$
  $i \leftarrow i + 1$
  $\tau_i \leftarrow \pi_h(s_t, l)$

In the environments provided in CARLA, all the roads are annotated and an auto-pilot function is implemented. First, we deployed a roaming agent, which randomly decided a direction at each intersection, and recorded observations from the agent at 10 Hz. We use two towns provided by the simulator, Town1 and Town2, as in previous work (Codevilla et al., 2018; Müller et al., 2018). Second, we partitioned the trajectory into a set of trajectory snippets corresponding to different sub-tasks. State observation, action, sub-task and end-of-sub-task values, $\langle s_t, a_t, \tau_t, e_t \rangle$, were generated for the snippets. Finally, we combined language data gathered from human users

(a) `rgb` **(b)** `ps` **(c)** `gs`

Figure 3.3: Examples of input dashboard images used in experiments: we compare performance on raw color images (`rgb`) with images trained on predicted (`ps`) or ground-truth segmentation (`gs`) from CARLA (Dosovitskiy et al., 2017).

single	double	ordinal
turn left	turn left at first and then right	you re going to take your second left up here
make a left turn	take a left here and then you re going to take a another right turn	you re going to go a little bit further for one block and make a left at the intersection
left	take your left here and then it will be a right turn ahead	take the second left

Table 3.1: Examples of generated sentences for left turns based on data gathered from realistic interactions. Examples are grouped into three categories, depending on time horizon and complexity of the instruction.

with the information about the snippets to generate realistic natural language instructions corresponding to our environment. In following subsections, we give details of the environment, data generation, and training.

3.4.1 Language generation

We collected language data by designing a two-player driving game with human subjects. In the game, one player was tasked with navigating the vehicle to a goal without any knowledge about the map of the world. The second player was tasked with instructing the first player about directions to goal using only natural languages. From this data, we designed templates to generate language instructions for training the high-level policy that matched our test environments. For more detailed procedure of the language generation, please see Appendix B.1.

We generated instructions of varying complexity that would be typical for interactions between a human and an autonomous vehicle. We grouped them into three categories: `single`, `double`, `ordinal`(see Table 3.1 for examples). The first category (`single`) contains instructions that tell the car how the behave at the next intersection. The second category (`double`) contains instructions about the behavior at the two upcoming intersections. Finally, the third category (`ordinal`) contains instructions including ordinal expressions relating to any of two upcoming intersections.

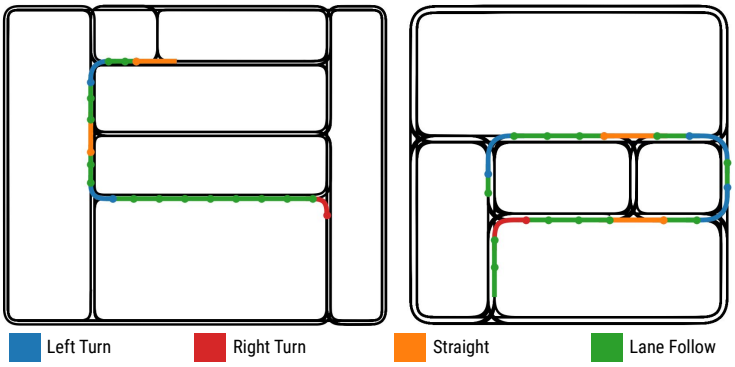


Figure 3.4: Top-down view showing trajectory segments with sub-task annotations.

3.4.2 Training and Trajectory Generation

We collected expert trajectories in simulation and trained each model with supervision analogous to prior work (Müller et al., 2018; Das et al., 2018b). First, we collected an expert trajectory, $d = \{p_t : t \in T\}$, by releasing a randomly roaming expert with a global planner and PID controller similar to (Müller et al., 2018), and obtained training sub-task labels based on the annotated road structure, where $p_t = \langle s_t, a_t, \tau_t \rangle$, $s_t \in \mathscr{S} = \mathbb{R}^{3\times200\times88}$ is an image used as a state observation, $a_t \in \mathscr{A}$ is an action, and $\tau_t \in \mathscr{T}$ is a sub-task label at time t.

Then, we partitioned the trajectories into a set of trajectory snippets $D = \{d_i\}$ for training both low-level and high-level policies based on the sub-task labels where $d_i = \{p_t : t \in R_i\}$ and $R_i = [p_i, q_i]$ such that $p_i \leq q_i \wedge p_i \in T \wedge q_i \in T \wedge \tau_a = \tau_b \; \forall a, b \in [p_i, q_i]$. Snippets around intersections were segmented according to which turn was taken into the `left`, `right`, and `straight` policies, corresponding to each of the three possible choices. Intersections were connected by the sub-task `lanefollow`.

In order to make the low-level policy robust, we added a margin before and after each snippet, so that each low-level policy also learns to follow the lane. We used L^1 loss for training the control model and binary cross entropy loss for training the end-of-sub-task model.

For high-level policies, we used three or five snippets as a longer segment which included one or two intersections and neighboring `lanefollow`

snippets. Within the segment, we drew data points from boundary regions between sub-tasks for training. We use cross entropy loss for training the high-level model. Fig. 3.4 shows a couple of examples of road segments with sub-tasks annotations.

In addition to one front-facing camera, we put two additional cameras rotated about 14 degrees to the left and to the right, to simulate the images from drifted states. We used three types of images to examine the effects of input modality on generalization from one town to the next, shown in see Fig. 3.3. These are: (1) `rgb`: color images from the dash-mounted camera, (2) `gs`: ground-truth binary road segmentation images from the simulator; and (3) `ps`: predicted binary road segmentation images from DeepLabv3+ (Chen et al., 2018) with Mobilenetv2 (Sandler et al., 2018) pretrained with COCO (Lin et al., 2014) and fine-tuned on the Cityscapes dataset (Cordts et al., 2016).

3.5 Experiments and Results

We perform a comprehensive evaluation of different properties of our model, and compare it to established baselines. We begin with a quantitative evaluation of generalization abilities of the model for different types of observations. We follow with ablation experiments and comparisons to previously published baselines. Then, we demonstrate robustness to misleading instructions and randomly timed commands. Finally, we show how our model can be used in interactive, real-time scenarios. In the following evaluations, we trained the model on Town1 and tested on both Town1 and Town2. In the training procedure, we draw fixed-length trajectories from the trajectory snippets.

3.5.1 Input comparison

One challenge with training policies on unstructured input such as images and language is transferring models to new environments. We explored the effects of different input modalities on performance and generalization inspired by the previous work (Müller et al., 2018) which has shown that segmentation-based policies transfer well between environments. The results can be seen in Table 3.2.

Input Modality →	rgb		gs		ps	
Language Type ↓	train	test	train	test	train	test
single	1.000	1.000	1.000	1.000	0.982	1.000
double	0.809	0.439	1.000	0.986	0.976	0.874
ordinal	0.813	0.333	1.000	0.938	1.000	0.938
all	0.880	0.613	1.000	0.970	0.982	0.926

Table 3.2: Comparison of results for three different input modalities: ground-truth segmentation `gs`, predicted segmentation `ps`, and raw color images `rgb`.

We performed language-grounded driving by starting from the beginning of the trajectory, given a randomly sampled sentence, and measure the rate of successful episodes. Overall, the model achieved almost perfect results for all levels of language complexity as long as ground truth segmentation was used as observations, even when generalizing across different environments. The performance dropped slightly, when predicted segmentation was used. Finally, raw color images resulted in largest performance drop when the model was transferred across environments, despite good performance on the training environment. Table 3.2 shows the quantitative evaluation result. The average performance drop from Town1 to Town2 with `rgb` is about 30.13% while the averages performance drops of `ps` and `gs` are about 5.038% and 1.301%. This reaffirms good generalization performance of semantic segmentation. This trend was primarily noticeable for language instructions of highest complexity (`ordinal`). A full comparison with model ablations and input modalities is provided in Table B.3 in Appendix.

3.5.2 Model Comparison

We compared our model (H_{ih}) with three variants of the model and two baseline models, a single policy and a Neural Modular Control (NMC) (Das et al., 2018b), given ground-truth road segmentation as input (`gs`). A single policy was implemented by extending a high-level policy to directly predict actions. We implemented NMC without attention for post-navigation question answering. The first variant ($H_{\emptyset}$) is a hierarchical baseline model with a high-level policy which only takes language instruction as its input. A high-level policy in the second variant (H_i) takes the image along with the language

Language Type → **Model ↓**	`single`		`double`		`ordinal`		`all`	
	train	test	train	test	train	test	train	test
Single policy	0.72	0.58	0.03	0.00	0.03	0.00	0.29	0.20
Single policy w/ history	0.68	0.67	0.08	0.00	0.14	0.00	0.31	0.23
NMC (Das et al., 2018b)	0.22	0.21	0.00	0.00	0.00	0.00	0.08	0.07
$H_\emptyset$: hierarchical baseline	**1.00**	**1.00**	**1.00**	**1.00**	0.97	0.91	1.00	0.99
H_i: $H_\emptyset$ w/ images	**1.00**	**1.00**	0.98	0.98	**1.00**	0.81	0.99	**0.99**
H_{ih}: H_i w/ history **(ours)**	**1.00**	**1.00**	**1.00**	0.96	**1.00**	**0.94**	**1.00**	0.97
H_{ihg}: H_{ih} w/ gated attention	0.98	**1.00**	0.98	0.94	0.97	0.89	0.98	0.95

Table 3.3: Comparison of our method to both a single policy and a Neural Modular Control (NMC) (Das et al., 2018b) baseline, and ablation of several different key components, given ground-truth road segmentation as input (`gs`.) Models used in ablation, $H_\emptyset$, H_i, and H_{ih}, are explained in Section 3.5.2. H_{ihg} replaces the original low-level model with Gated Attention model (Chaplot et al., 2018).

instruction but it does not use the sub-task history. Our model (H_{ih}) uses the language instruction, the image, and the sub-task history in the high-level policy. The last variant (H_{ihg}) uses the same high-level model as our model but it replaces the original low-level policy using a few sub-task-specific GRU layers by a new low-level policy which is conditioned by a sub-task label using GA. A single policy was implemented by extending a high-level policy to directly predict actions. Table 3.3 shows that all variants of our model outperformed the baselines.

We see that both baselines struggled to interpret more complex commands (`double` or `ordinal`). The single policy could learn a turning behavior from fixed-length sub-trajectories but failed to learn to distinguish multiple turns and plan a series of turns. This is understandable, given the length of the trajectories (hundreds of frames), which do not fit in a single recurrent unit. The hierarchical decomposition of the task in our model reduces the complexity of the problem and makes the model trainable with long-time horizon data. The NMC baseline performed poorly even for simple directions (`single`). Lack of an attention mechanisms resulted in poor performance of end-of-sub-task and sub-task prediction.

Among our model and two hierarchical variants of the models, H_{ih}, $H_\emptyset$ and

Language Type → / **Model ↓**	`single` train	`single` test	`double` train	`double` test	`all` train	`all` test
$H_\emptyset$: hierarchical baseline	**1.000**	**1.000**	0.758	0.652	0.828	0.742
H_i: $H_\emptyset$ with images	**1.000**	**1.000**	**1.000**	**0.957**	**1.000**	**0.968**
H_{ih}: H_i with sub-task history **(full model)**	**1.000**	**1.000**	**1.000**	0.928	**1.000**	0.946

Table 3.4: Evaluation of our approach for misleading language instructions (e.g. "go straight" when no straight road exists). We used ground-truth segmentation images `gs` as input.

H_i, we could not see a huge performance gap. Transition between sub-tasks is highly dependent on the end-of-sub-task value from low-level policy and that gives the model with a simple high-level policy high performance.

In addition, we replace the original low-level model with the gated-attention model, H_{ihg}, which takes sub-task values as a conditional input. Though this conditional low-level model performs slightly worse than the original model, H_{ih}, its performance is comparable to other ablation models. It implies that switching between a fixed number of special layers is not necessarily needed; for higher-level tasks in future, we can generalize the intermediate representation not restricted to a fixed number of sub-tasks.

3.5.3 Misleading Instructions

Realistic natural language instructions are often inconsistent and ambiguous. For an autonomous system, it is paramount to handle such instructions and generate only safe behavior. To evaluate our model in such conditions, we generated misleading language instructions containing directions not possible to execute given the current world map (e.g. "go straight" for a T-shaped intersection). In such cases, we expect a safe behavior and choose to train the model to stop for impossible straight directions or go straight for impossible turns. We evaluated the resulting model only on misleading instructions.

Table 3.4 shows the quantitative evaluation of our model for misleading language directions using ground-truth segmentation images as input. Usage of the image in the high-level policy seemed to be an important factor on performance. Performance of the model $H_\emptyset$ on complicated language directions is degraded in comparison to other models which use images in the high-level

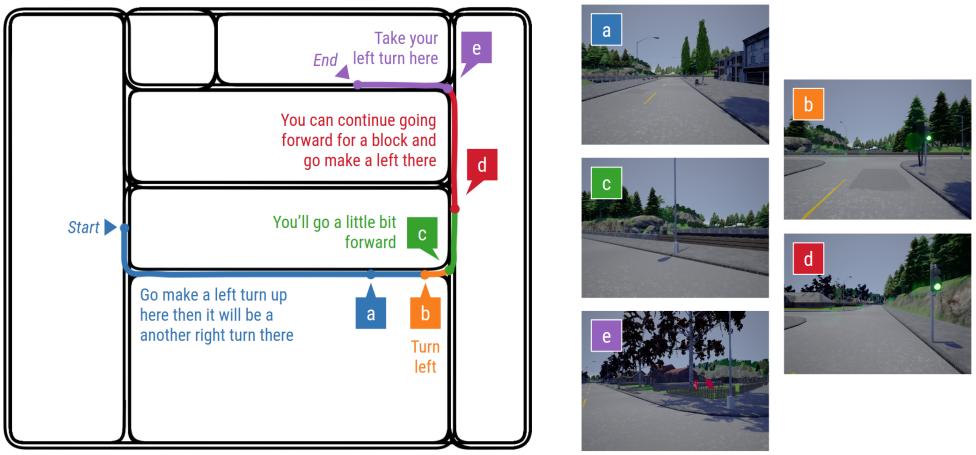


Figure 3.5: An example of interactive driving. The trajectory and the instructions provided by the user are shown on the left. The right side shows images corresponding to the indicated points along the trajectory.

policy. This demonstrates the importance of using image in decision making when it has to deal with misleading language instructions. Our model shows the robustness to misleading instructions.

3.5.4 Interactive Driving

In realistic settings, a self-driving car will receive instructions from the user at different moments in time, even if the car is currently executing a previous command. To illustrate the robustness of our model to imperfectly timed commands as well as random interruptions, we designed an interactive, real-time driving protocol, where users could provide natural language instructions at any moment in time. The agent interrupts the current plan whenever a new instruction s received. Here, we present and analyse an example of such experiment (see Fig. 3.5).

The experiment began with the instruction “Go make a left turn up here then it will be a another right turn there." Then the user interrupted the execution of the commands three times at random moments, often while a sub-task such as left turn is currently being executed. This interactive driving example shows that our agent can be used to drive continuously according to user directions, even when frequently interrupted or when given inconsistent

commands. Thanks to its hierarchical structure, our model is less sensitive to timing issues; it will complete the current maneuver before executing the next command.

3.6 Conclusion

We showed a system for linguistic control of a self-driving vehicle from images, and provide an ablation analysis of which components of the network are important for providing the best performance including generalization to new environments. In particular, we showed our model improves on related prior work for visual question answering (Das et al., 2018b) and extends work in driving using conditional policies. Our future work will focus on even more complex language expressions, with emphasis on objects in the environment.

Chapter 4

LanguageRefer: Spatial-Language Model for 3D Visual Grounding

4.1 Introduction

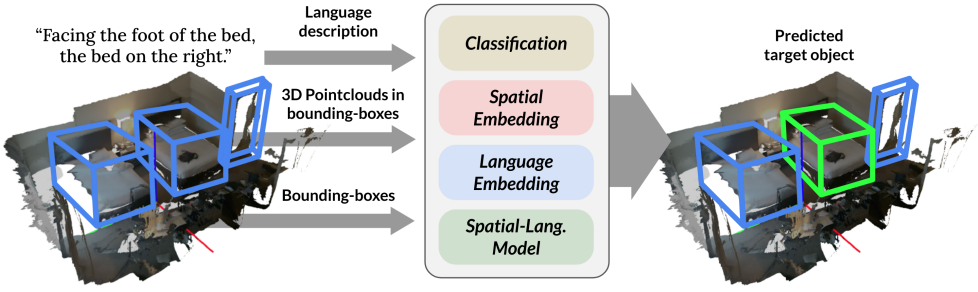


Figure 4.1: Simplified overview of LanguageRefer. The **LanguageRefer** model takes as input a grounding language description of a single object in the scene, a 3D point cloud of a scene, and bounding boxes of objects in the scene and predicts the target object. Its four modules include: a classifier, a spatial embedder, a language embedder, and a spatial-language model.

For robots to communicate seamlessly with humans to perform meaningful tasks in indoor environments, they must understand natural language utterances and ground them to real-world elements. Several recent advances

have combined language and visual elements, producing methods for tasks such as visual question and answering (VQA) involving spatio-temporal reasoning tasks (Johnson et al., 2017; Yi et al., 2020; Girdhar and Ramanan, 2020), embodied QA (Das et al., 2018a), and pre-training for visual recognition tasks with language descriptions (Radford et al., 2021). Further, embodied agents can follow visually grounded language instructions to perform embodied tasks (Shridhar et al., 2020b; Kolve et al., 2017). However, for real robots to intelligently perform these tasks, we need 3D representations from raw sensor data; ReferIt3D (Achlioptas et al., 2020) proposes a benchmarking dataset of language utterances referring to objects in 3D scenes from the ScanNet (Dai et al., 2017) dataset.

Our long-term goal is to enable robots to visually navigate indoor environments based on referential instructions. In this paper we take a step towards this goal by leveraging the ReferIt3D dataset to build a model that can identify the 3D object referred to in a language utterance.

Referential language to identify an object in real-world 3D scenes poses a challenging problem. Consider the sample utterance*"Facing the foot of the bed, the bed on the right"* along with a 3D scene as shown in Figure 4.1. Humans can easily follow the language clues, infer the point of view of the speaker, locate all the referenced elements, and spatially reason to locate the *bed* in the scene despite two instances of *beds*. However, viewpoint prediction, object identification, and spatial reasoning remain open-ended research problems in robotics and vision.

The 3D reference task proposed by ReferIt3D is difficult because: (1) reconstructed 3D scenes of the real world in the ScanNet dataset are noisy and lack fine details compared to 2D images or rendered 3D scenes, (2) fine-grained class labels and expressions in natural language utterances are diverse and not exactly matched, (3) view-dependent utterances often require guessing the original viewpoints, which deters the model from properly learning spatial concepts, and (4) the combined complexity of multiple challenges complicates efforts to analyze what the model learns or understands.

Inspired by the success of the methods in (Ding et al., 2020; Kamath et al., 2021) on CLEVR and CLEVRER domains for the spatial reasoning task, we

hypothesized that for the 3D reference task, decoupling the spatial-reasoning from the perceptual task of identifying the objects in the 3D scene would improve performance and clearly track the role of perception noise in performance. More specifically, instead of developing an integrated multi-modal perception system, we assumed a pre-trained instance classification model or ground-truth classes for the objects that informed the spatial reasoning task. We focused on how the language model with spatial information could handle the reference task.

The ReferIt3D dataset includes two sub-datasets containing natural and synthetic language utterances, namely Nr3D and Sr3D, respectively. In our experiments, our model achieved comparable scores on both with predicted instance class labels. We observed high accuracy with ground-truth labels in Sr3D, which indicates that our model better understood template-based language data. Since our pipeline is modular and features multiple models (perceptual, spatial embedding, pre-trained language embedding, and spatial-language), our approach is flexible and adaptable to different environments and object entities.

Another aspect of the 3D reference task is viewpoint prediction. The ReferIt3D dataset contains utterances that can be grouped into view-independent (VI) and view-dependent (VD) categories. An example of a VI utterance is *"The lamp closer to the white armchair"* and a VD utterance is *"The lamp on the right in-between the beds"*. The VD utterance requires viewpoint prediction. This distinction is crucial for robotics applications where the agent must infer the viewpoint to which the speaker is referring. In the ReferIt3D dataset, some VD utterances lack information to guess the valid orientation of the agent (who utters the language description), which prevents a model from understanding the significance of spatial relationships such as 'left of.' This is due to the annotation process of ReferIt3D datasets, where both a speaker and a listener can freely rotate the scene to infer an ill-defined orientation in the utterance. Human annotators who are aware of spatial concepts tend to validate otherwise arbitrary orientations. However, a data-driven model will suffer from degraded learning when it cannot verify orientations as well as human annotators. Viewpoint-free annotations may suit most robotics

applications; nonetheless, predicting or verifying whether a given statement and the viewpoint match remains important. To better train and investigate the agent in view dependencies, we provide an extra collection of orientation annotations for VD utterances and compare the models with and without viewpoint correction from them.

To summarize, the paper contributes: (1) a novel transformer-based spatial-language model in a modular pipeline that better understands spatial relationships in a 3D visual grounding task. We show that our model achieves comparable performance with state-of-the-art methods on the Nr3D and Sr3D datasets. (2) analysis of the ReferIt3D dataset with viewpoint orientation annotations to remove potential artifacts from implicit orientations. (3) ablation and additional experiments with ground-truth classes that decouple the impact of perception noise in the spatial reasoning task.

4.2 Related Work

4.2.1 Vision-and-Language Navigation and Robot Navigation

Vision-and-language navigation (VLN) has been extensively studied and made remarkable progress over the last few years ((Anderson et al., 2018c; Ku et al., 2020; Qi et al., 2020; Chen et al., 2020b; Savva et al., 2019; Ma et al., 2019a; Ke et al., 2019a; Anderson et al., 2018b; Fried et al., 2018; Hong et al., 2020; Shrivastava et al., 2021)). Given a language instruction in the simulation environment, the goal is for an agent to reach the desired node on the pre-defined traversal graph using images as input. The literature offers several extensions. ALFRED (Shridhar et al., 2020b) proposes an extended VLN task that grounds a sequence of sub-tasks to achieve a higher level task in the AI2Thor environment (Kolve et al., 2017). ALFRED navigation sequences have (implicit) goals that are often close to objects of interest in the subsequent sub-tasks, e.g., when an agent is asked to move in front of the sink because it is going to clean a cup there. With high-level semantic tasks, object-centric spatial understanding is of even greater importance.

In another direction, recent approaches have relaxed the constraint of discrete traversal in VLN into continuous space ((Anderson et al., 2020; Krantz

et al., 2020; Blukis et al., 2020; Roh et al., 2019; Irshad et al., 2021)). Here, an agent encounters more complex tasks involving time and space. Thus, expanding the space representation to 3D can be an effective solution. In the context of VLN, we consider the 3D visual grounding task as a proxy for 3D indoor navigation that includes the full observability assumption and goal-oriented language descriptions. In particular, our approach focuses on understanding spatial relationships among objects, which plays a key role in VLN and robot navigation.

4.2.2 2D and 3D Visual Grounding

The 2D visual grounding task localizes an object or region in an image given a language description about the object or region ((Plummer et al., 2015; Kazemzadeh et al., 2014; Mao et al., 2016)). Most methods use two-stage approaches: they first generate proposals and then compare the proposals to the language description to choose the grounded proposal ((Yu et al., 2018; Yang et al., 2019; Yu et al., 2016; Liu et al., 2020; Ye et al., 2019)).

The 3D visual grounding task localizes a 3D bounding box from the point cloud of a scene given a language description. Recently, ReferIt3D (Achlioptas et al., 2020) and ScanRefer (Chen et al., 2020a) were proposed as datasets for 3D visual grounding tasks, with language annotation on the ScanNet (Dai et al., 2017) dataset. Most 3D grounding approaches ((Achlioptas et al., 2020; Chen et al., 2020a; Yuan et al., 2021; Feng et al., 2021; Yang et al., 2021)) follow a two-stage schemes similar to many 2D visual grounding tasks. First, multiple bounding boxes are proposed or the ground-truth bounding boxes are used, and then features from the proposals are combined or compared with features from the language description. InstanceRefer (Yuan et al., 2021) extracts attribute features both from point clouds and utterances and compares them to select the object that best matches. FFL-3DOG (Feng et al., 2021) matches features from language and point clouds with guidance from language and a visual scene graph. These methods rely on specific designs, e.g., bird-eye-view mappings with different types of operations or intensive language pre-processing for graph generation. *In contrast, our approach leverages the language embedding space from the pre-trained language model.* Following the

pipeline and general architecture of the language model therefore requires minimal manual design compared to previous works. SAT (Yang et al., 2021), like our approach, relies on transformer (Vaswani et al., 2017) models; it learns a fused embedding of multi-modal inputs and uses auxiliary 2D images. In contrast, our approach uses a semantic classifier to predict object class labels and takes these labels as input. It has a marginal cost of learning fused embeddings compared to training multiple BERT models (Devlin et al., 2018) from scratch in SAT (Yang et al., 2021). Even given only semantic information from point clouds, our model still achieved comparable performance with state-of-the-art methods on Nr3D and Sr3D datasets. In addition, the decoupled perception module makes our approach modular and thus transferable to different data.

4.3 Problem Statement and Methodology

Given a 3D scene S with a list of objects $(O_1, \cdots, O_M)$ and a language utterance U, the problem is to predict the target object $O_{\mathscr{T}}, \mathscr{T} \in \mathscr{I}_M = \{1, \cdots, M\}$ referred to in the language. A single object O_i consists of a bounding box $B_i \in \mathscr{B} = \mathbb{R}^6$ and corresponding point cloud $P_i \in \mathscr{P} = \mathbb{R}^{N_i \times 6}$ (xyz positions and RGB values) in the bounding box with N_i number of points.

We propose an approach based on language models, called **LanguageRefer**, to solve a 3D visual grounding task. Our model focuses on understanding spatial relationships between objects from language descriptions and 3D bounding box information. We chose this approach due to (1) the high dependency on spatial relationship descriptions in language, and (2) the holistic nature of spatial relationship information, which differs from unary attribute information such as color and shape. As shown in Figure. 4.2, we use a two-stage approach. First, we determine the class labels of objects in the scene. Second, we use spatial-language embedding to identify the referenced object. The following subsections describe these steps in detail.

Semantic Classification Model and Tokenization. For semantic classification of the point cloud in a bounding box, we employed PointNet++ (Qi et al.,

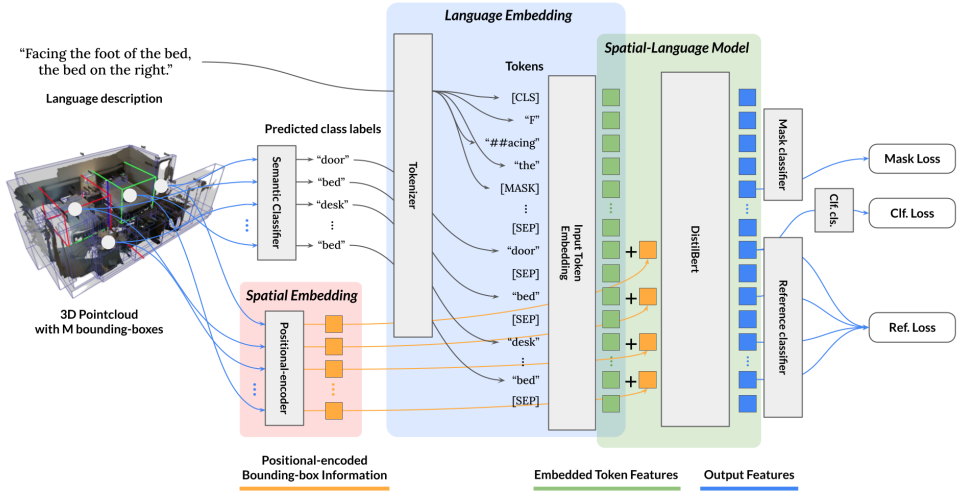


Figure 4.2: Detailed overview of LanguageRefer. A semantic classifier predicts class labels from a 3D point cloud in each bounding box (using color and xyz positions). The language description or utterance (e.g., "*Facing the foot of the bed, the bed on the right*") is transformed into a sequence of tokens. The input token embedding in DistilBert (Sanh et al., 2019) converts the tokens into embedded feature vectors (green squares). Bounding box position and size information are positional-encoded to form encoded vectors using techniques from (Vaswani et al., 2017) (orange squares); they are added to the corresponding embedded feature vectors (green squares). After the addition, our reference model processes the modified features and feeds them to multiple tasks. The main task is a reference task, i.e., it chooses the referred object from the object features. The instance classification task is a binary classification, i.e., it determines whether the given object feature belongs to the target class. Finally, the masking task, commonly used in language modeling, recovers the original token from a randomly replaced token in the utterance.

2017), which achieved 69% accuracy on average in the test dataset. In training and inference, we use a sampled point cloud $P'_i \in \mathbb{R}^{1024\times 6}$ and PointNet++ predicts the semantic label $\hat{l} \in \mathscr{L}$, where $\mathscr{L}$ is a set of class labels in text. Each scene has pairs of predicted class labels and bounding box values $((\hat{l}_i, B_i) : \hat{l}_i \in \mathscr{L}, B_i \in \mathscr{B})$ from objects. Predicted labels are concatenated to the utterance U with a separator [SEP] and then split by a tokenizer into a list of indices of tokens: U becomes $(u_1, \cdots, u_t)$, and each predicted class label $\hat{l}_i$ becomes $(o^i_1, \cdots, o^i_{n_1})$, where each token index is in $\mathscr{I}_D$ and D is the size of

the dictionary.

Language Model and Token Embedding Generation. Our model uses a pre-trained language model, DistilBert, for the reference task. Transformers consider relationships among all pairs of elements through attention, and they can be effectively leveraged to explain spatial relationships between objects as discrete entities. In our formulation, predicted class labels are considered to be sentences, so they are concatenated to the utterance with separation by [SEP]. Therefore, the final sequence of token indices would be V = ([CLS], $u_1, \cdots, u_t$, [SEP], $o_1^1, \cdots, o_{n_1}^1$, [SEP], $\cdots$, [SEP], $o_1^M, \cdots, o_{n_M}^M$, [SEP]). Though the token index sequence complies with the specification of DistilBert, it violates the number of sentences. Then, we transform V into the token embedding sequence $W = (w_1, \cdots, w_T), w_i \in \mathbb{R}^{768}$ using DistilBert's word embeddings. For concise notation, we define the indices of utterance tokens in V or W as a mask $M_U = (2, \cdots, t+1)$ and the indices of first tokens from objects $(o_1^1, \cdots, o_1^M)$ as a mask M_O. We also define a mask operator $[\cdot]$ to manipulate specific elements in the sequence; for instance, $W[M_U] = 0$ empties all utterance embeddings in W.

Spatial-Language Model and Spatial Embeddings.. To combine spatial information from raw bounding box values into the token embedding W, we employ sinusoidal positional encoding PE$(\cdot)$ from (Vaswani et al., 2017) to transform the bounding box vector (center position and size) $B_i \in \mathscr{B} \subset \mathbb{R}^6$ to $b_i = \text{PE}(B_i) \in \mathbb{R}^{768}$, which is then added to $W[M_O]$.[1] The token embedding W is then combined with spatial information and finally transformed into the output embedding $X = (x_1, \cdots, x_T), x_i \in \mathbb{R}^{768}$ by the reference model, which is fine-tuned from the pre-trained DistilBert. The final reference task is performed by the reference classifier from $X[M_O]$, as explained in the following subsection.

Loss Functions. We use three tasks for training and corresponding loss values. First, we use the reference loss, $\mathscr{L}_{ref}$, following the original proposal

[1]Each value in b_i is extended to 128-dimensions by PE and then concatenated to form a 768-dimensional vector.

in (Achlioptas et al., 2020). We ask the model to choose one object as the target instance from M candidates. We collect scalar values from $X[M_O]$ by a linear layer and take the argmax on those values to choose the target instance.

Second, we add a binary target classification loss $\mathscr{L}_{clf}$ on $X[M_O]$ to determine whether a given object belongs to the target class.

Last, we employ mask loss from language model pre-training, $\mathscr{L}_{mask}$. We randomly replace the tokens from nouns in the utterance with a probability of 15 %. The noun token is replaced by [MASK] with an 80 % chance, by a random token with a 10 % chance, or it remains the same with a probability of 10 %. Then, the model is asked to recover the original token index. We expect the model to fill in the replaced tokens in the utterance by understanding the relationship between objects. We use cross entropy loss for all tasks and compute the final loss as

$$\mathscr{L} = \mathscr{L}_{ref} + 0.5\mathscr{L}_{clf} + 0.5\mathscr{L}_{mask}. \tag{4.1}$$

At inference, we followed the approach of InstanceRefer (Yuan et al., 2021) to filter out objects that do not belong to the predicted target class. We used an extra DistilBert-based target classification model of 94 % accuracy that takes the language utterance as input to predict the target class. To reduce the chance of removing the true target instance in the filtering process, the top-k class predictions (from the semantic classifier) for each object are compared to the predicted target class. We use $k = 4$ throughout the experiments. For masking loss computation, we extracted nouns in the utterance. We used flair (Akbik et al., 2019) for part-of-speech (POS) tagging.

4.4 Experiments

4.4.1 Datasets

We evaluated our model on the reference task from ReferIt3D (Achlioptas et al., 2020) with two datasets, Nr3D (Natural reference in 3D) and Sr3D (Spatial reference in 3D, which contains spatial descriptions only). Both datasets augment ScanNet (Dai et al., 2017), a reconstructed 3D indoor scene dataset

with language descriptions. Nr3D has 41,503 natural language utterances, and Sr3D contains 83,572 template-based utterances, on 707 scenes following the official splits of ScanNet (Dai et al., 2017). The datasets have 76 target classes and are designed to have multiple same-class distractors in the scene.

4.4.2 Experiment Settings

ReferIt3D (Achlioptas et al., 2020) provides the ground-truth bounding boxes of objects in the scene, point clouds of the scene, utterances and corresponding target objects. We measured the accuracy of the model by comparing the object selected from M candidates to the ground-truth target object. When the number of same-class distractors exceeded two, we classified the instance as "hard" according to (Achlioptas et al., 2020). The other cases were classified as "easy."

We trained models with a learning rate of 0.0001 using AdamW optimization, warm-up and linear scheduling. Our model was initialized with a pre-trained model of the cased Distilbert base (Sanh et al., 2019) from the Hugging Face implementation (Wolf et al., 2020).

We compared the performance of our model with state-of-the-art methods on ReferIt3D ((Achlioptas et al., 2020; Yang et al., 2021; Chen et al., 2020a; Yuan et al., 2021; Feng et al., 2021)) based on the reported numbers on the challenge website (Achlioptas) and corresponding papers. Since SAT (Yang et al., 2021) uses an extra 2D image dataset in their training, we separated it from non-SAT, their baseline model that is not trained with the extra dataset.

4.4.3 Evaluation on ReferIt3D (Achlioptas et al., 2020)

Table 4.1 shows the accuracy on Nr3D and Sr3D. Our model outperformed other models (without extra training datasets) on both Nr3D and Sr3D. It achieved 43.9% on Nr3D with an +8.3% improvement over the baseline method of ReferIt3D (Achlioptas et al., 2020) and a 2.2% increase over the best accuracy from other models in Nr3D. For Sr3D, our model achieved 56.0%, which is a 15.2% and 8.0% increase over the baseline and the best accuracy in the Sr3D section, respectively. It is also comparable to the accuracy of SAT (with a 1.9%

Dataset	Method	Overall	Easy	Hard	View-dep.	View-indep.
Nr3D	ReferIt3D (Achlioptas et al., 2020)	35.6 %	43.6 %	27.9 %	32.5 %	37.1 %
	ScanRefer (Chen et al., 2020a)	34.2 %	41.0 %	23.5 %	29.9 %	35.4 %
	InstanceRefer (Yuan et al., 2021)	38.8 %	46.0 %	31.8 %	34.5 %	41.9 %
	FFL-3DOG (Feng et al., 2021)	41.7 %	48.2 %	35.0 %	37.1 %	44.7 %
	non-SAT (Yang et al., 2021)	37.7 %	44.5 %	31.2 %	34.1 %	39.5 %
	Ours	**43.9** %	**51.0** %	**36.6** %	**41.7** %	**45.0** %
Nr3D w/ 2D images	SAT (Yang et al., 2021)	49.2 %	56.3 %	42.4 %	46.9 %	50.4 %
Sr3D	ReferIt3D (Achlioptas et al., 2020)	40.8 %	44.7 %	31.5 %	39.2 %	40.8 %
	InstanceRefer (Yuan et al., 2021)	48.0 %	51.1 %	40.5 %	45.4 %	48.1 %
	non-SAT (Yang et al., 2021)	47.4 %	N/A	N/A	N/A	N/A
	Ours	**56.0** %	**58.9** %	**49.3** %	**49.2** %	**56.3** %
Sr3D w/ 2D images	SAT (Yang et al., 2021)	57.9 %	61.2 %	50.0 %	49.2 %	58.3 %

Table 4.1: Accuracy on ReferIt3D (Achlioptas et al., 2020). Our model outperformed state-of-the-art models on both Nr3D and Sr3D except for models with additional training data (SAT with 2D images). The average performance gap between ours and other models on Sr3D (10.6%) is larger than that on Nr3D (6.3%) since our model uses only spatial reasoning for the reference task.

difference), which was trained with additional 2D image training data. These results prove that our model can accurately reason about spatial relationships from spatial-language embeddings and outperforms other models on both datasets despite the loss of information in appearance. In addition, less diverse language expressions and utterances only about spatial reasoning can explain our model's strong performance on Sr3D.

4.4.4 Evaluation with Ground-Truth Class Labels

We further investigated the model's spatial reasoning ability by removing noise in class labels. We replaced predicted class labels with ground-truth class labels, which was possible due to the explicit usage of class labels in our model.

Table 4.2 shows the result with and without ground-truth class labels. The first and second columns demonstrate the type of dataset in training and evaluation, respectively. For instance, the second row shows the evaluation result of the model trained with the Nr3D dataset with ground-truth class labels and evaluated on the Nr3D dataset with predicted class labels. When we trained and evaluated our model with ground-truth labels on Nr3D and Sr3D

(row 4 and 8), we achieved 54.3% and 91.1% accuracy, respectively. Compared to the accuracy of models trained and evaluated with noisy labels, we realized an improvement is 10.4% and 35.1%, respectively, for each dataset. When we evaluated with ground-truth labels the models trained with noisy labels, their performance increase was 9.7% and 24.2%, respectively. Multiple reasons account for the difference in performance on Nr3D and Sr3D: diversity in the natural language dataset, a higher portion of view-dependent utterances or view-dependent utterances with no mention of orientation, and descriptions other than spatial relationships, such as color or appearance.

In addition, we examined the accuracy given switched datasets, namely, when we train a model on Nr3D and evaluate it using Sr3D, and vice versa. The last two rows in Table 4.2 show two switched evaluation results: 40.0% and 37.6% on Sr3D and Nr3D, respectively. The 37.6% accuracy on Nr3D from the model trained on Sr3D is comparable to the accuracy of non-SAT (Yang et al., 2021) and InstanceRefer (Yuan et al., 2021). It shows our model's generalizability of spatial reasoning and potential to transfer to different perception modules and datasets.

4.4.5 Ablation on Loss Terms

We trained models with different combinations of loss terms, and Table 4.3 shows the results. In addition to three tasks, we examined the effect of a text classification task that predicts the target class label $l \in \mathscr{L}$ from the tokens from utterance $X[M_U]$.

We found that only the binary classification loss shows its effect clearly (+3.5% from Ref. to Ref.-Clf., +3.9% from Ref.-Mask to Ref.-Mask-Clf., +1.0% from Ref.-Text to Ref.-Text-Clf.). The mask loss was not effective, and the text classification loss degraded accuracy. One hypothesis here is that the text classification loss does not help the model to transform the initial language embedding to a spatial-language embedding. The model may not require strong language constraints because (1) it already has a good initial language embedding, and (2) the text classification loss is independent of the scene's spatial configuration. Due to the performance drop, we chose our model without the text classification loss.

Dataset		All	Easy	Hard	View-dep.	View-ind.
Training	Evaluation					
Nr3D-pred	Nr3D-pred	43.9 %	51.0 %	36.6 %	41.7 %	45.0 %
Nr3D-gt	Nr3D-pred	41.4 %	48.0 %	34.6 %	38.1 %	43.0 %
Nr3D-pred	Nr3D-gt	53.6 %	64.4 %	42.5 %	50.7 %	55.0 %
Nr3D-gt	Nr3D-gt	54.3 %	65.5 %	42.8 %	49.1 %	56.8 %
Sr3D-pred	Sr3D-pred	56.0 %	58.9 %	49.3 %	49.2 %	56.3 %
Sr3D-gt	Sr3D-pred	52.1 %	53.8 %	48.2 %	43.9 %	52.5 %
Sr3D-pred	Sr3D-gt	80.2 %	83.2 %	73.1 %	62.5 %	81.0 %
Sr3D-gt	Sr3D-gt	91.1 %	93.1 %	86.2 %	67.0 %	92.1 %
Nr3D-pred	Sr3D-pred	40.0 %	43.6 %	31.5 %	41.2 %	39.9 %
Sr3D-pred	Nr3D-pred	37.6 %	45.3 %	29.6 %	34.5 %	39.0 %

Table 4.2: Ablation with ground-truth class labels. First and second columns show types of data used in training and evaluation. The high overall accuracy (91.1% at row 8) of the model both trained and evaluated with ground-truth class labels on Sr3D shows its spatial reasoning ability besides the perception noise. Accuracy gaps between ground-truth and predicted class labels on Nr3D and Sr3D (9.7%, 24.2%, respectively) indirectly tell us about language complexity and information loss due to classification. Transferring the model trained with Sr3D to Nr3D evaluation shows an overall number (37.6% at row 10) comparable to those from other methods. Our model can easily accommodate different classification models or datasets.

4.4.6 Viewpoint Annotation

View-dependent utterances without information about original viewpoint make the reference task in ReferIt3D (Achlioptas et al., 2020) more challenging. For instance, utterances such as "*The door is wood with the handle on the left side.*" assume specific orientations of the agent, and it is impossible to recover the true orientation without knowing the referred object; this differs from view-dependent utterances with explicit viewpoint information, such as "*Facing the foot of the bed.*" However, the original dataset of ReferIt3D (Achlioptas et al., 2020) does not distinguish the utterances without orientation information from those with it. Therefore, we split the view-dependent (VD) utterance category into two subcategories, VD-explicit and VD-implicit, where VD-explicit has explicit viewpoint information in the utterance. We then collected orientations

Ref.	Clf.	Mask	Text	All	Easy	Hard	View-dep.	View-ind.
✓	-	-	-	40.6 %	48.2 %	32.8 %	38.5 %	41.6 %
✓	✓	-	-	44.1 %	51.3 %	36.7 %	41.0 %	45.6 %
✓	-	✓	-	40.0 %	47.2 %	32.5 %	36.8 %	41.5 %
✓	-	-	✓	40.0 %	48.7 %	30.8 %	37.8 %	40.9 %
✓	✓	✓	-	43.9 %	51.0 %	36.6 %	41.7 %	45.0 %
✓	✓	-	✓	41.0 %	49.7 %	31.9 %	39.3 %	41.8 %
✓	✓	✓	✓	40.0 %	48.2 %	31.5 %	38.9 %	40.6 %

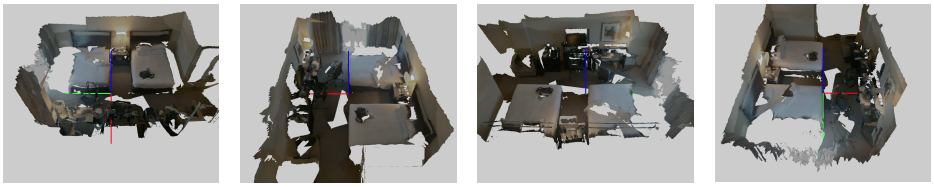
Table 4.3: Ablation of loss terms on Nr3D. The classification loss was effective, the mask loss did not significantly affect accuracy, and the text loss degraded accuracy. We chose the model without text losses (fifth row, in blue).

Correction		All	Easy	Hard	View-dep.	View-ind.
Training	Evaluation					
✓	-	43.5 %	50.7 %	36.0 %	37.0 %	46.6 %
✓	✓	49.0 %	56.0 %	41.8 %	54.4 %	46.4 %
-	✓	43.1 %	50.3 %	35.6 %	40.4 %	44.4 %
-	-	43.9 %	51.0 %	36.6 %	41.7 %	45.0 %

Table 4.4: Comparison of accuracy with and without corrected orientations on Nr3D.

from human annotators that validated the utterances. We set four standard orientations assuming the agent is in the room (around the center of the scene) and asked annotators to select all orientations that could be considered valid from the utterance. Figure 4.3 shows examples of the four orientations. We found that four orientations were sufficient to recover the original viewpoints of the speakers. In total, 12,680 view-dependent utterances of the Nr3D dataset were annotated; from these, 5,942 utterances were classified as VD-explicit. For train and test split, 10,206 and 2,474 utterances were annotated, respectively.

From the orientation for view-dependent utterances, we revised the dataset with the corrected orientation; we rotated the scene with respect to the annotation so all scenes remained valid at the canonical orientation. For view-independent utterances, we randomly rotated the scenes since they were valid in any direction. Table 4.4 shows the accuracy values for models



(a) An example of standard orientation 1.

(b) An example of standard orientation 2.

(c) An example of standard orientation 3.

(d) An example of standard orientation 4.

Figure 4.3: Examples of standard orientations for viewpoint annotation on Nr3D (a-d). We assume that the robot is always inside the room except for cases specified by utterances.

trained with and without corrected orientations. At inference, we evaluated each model with and without corrected orientations, as well. The second row shows the accuracy of the model that was trained and evaluated with corrected orientations. Its overall accuracy was improved by +5.1% from the final model without the correction that was used in Table 4.1 (the last row in Table 4.4). Note that the improvement on view-independent utterances was marginal (+1.4%), but the improvement on view-dependent ones was significant (+12.7%). The first row shows the accuracy of a model trained with corrected orientations and evaluated on the test data without correction. This model achieved accuracy comparable to the final model (−0.4%). This implies the correction helped the model to accurately interpret the view-dependent scene when the orientation was consistently aligned, introducing no unwanted bias.

4.5 Conclusion

We proposed LanguageRefer, a spatial-language model for 3D visual grounding in a reference task. LanguageRefer combines language embeddings from utterances and class labels with positional-encoded spatial information for efficient learning of the spatial-language embedding space without different modules for individual modality. Experimental results show that LanguageRefer outperformed state-of-the-art models on ReferIt3D with no additional training data. Analysis and ablations we performed demonstrate the effects

of 1) noisy class labels, 2) arbitrary viewpoints in view-dependent utterances, 3) and the transfer of our model to different datasets for future robotics applications.

Chapter 5

Visual Grounding on RGBD Images

5.1 Introduction

In Chapter 4, we propose a model that extends the pre-trained language model to a spatial-language model based on (Sanh et al., 2019) for 3D visual grounding tasks with an additional pointcloud classification model. We annotated the estimated valid orientations at the time of the original data collection which makes the view-dependent utterances valid and examined the effect of the annotation in accuracy. Assumption of the fully-observed scene without viewpoint constraints makes the model confused. It also makes the task detached from real-time navigation scenarios where the language utterance is coupled with the camera pose or the robot pose and the scene is partially-observed. Additionally, by using reconstructed 3D scenes, the quality of the 3D pointcloud gets degraded due to artifacts or failures in reconstructions from thin structures or reflective surfaces. Though the dataset was built from sequences of RGBD images, the model needs to rely on low-quality reconstructions instead of the raw data. In 4.3, we indirectly observe the effect of the perception error by replacing the predicted class labels with the ground-truth class labels and the quality of the 3D dataset may contribute to the perception error.

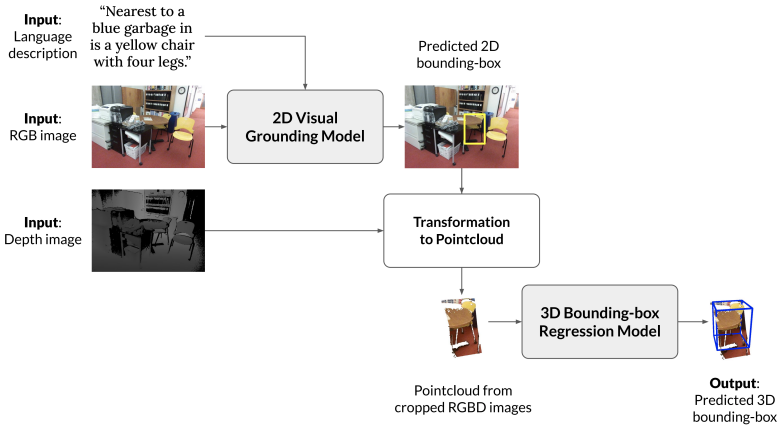


Figure 5.1: Simplified overview of the proposed model. The model takes a pair of RGBD images and a grounding language that refers to an object in the images. It estimates a 3D bounding-box of the referred object. Our model has two sub-models: a 2D visual grounding model and a 3D bounding-box regressor. The 2D visual grounding model takes the utterance and the color image to produce a 2D bounding-box of the target object on the image. Then the RGBD images are cropped with respect to the 2D bounding-box to create a pointcloud and it is fed to the 3D bounding-box regressor. At last, the regressor estimates 3D bounding-box from the cropped pointcloud.

In this chapter, we extend the proposed approach in Chapter 4 to a visual grounding task in RGBD images. (Liu et al., 2021a) proposed a visual grounding task in RGBD images which can alleviate the issues of (Achlioptas et al., 2020). By directly using RGBD images and language descriptions based on them, we no longer need the viewpoint estimation. It simulates realistic robot navigation scenarios of partially observed scenes while directly accessing high-quality observation. As following the strategy of leveraging a large pre-trained model and compositing modules for interpretability, we propose the model with two modules: a 2D visual grounding model and a 3D bounding-box regression model. Our model achieves better accuracy than the model in (Liu et al., 2021a).

SUNRefer dataset was a dataset proposed in (Liu et al., 2021a) for the visual grounding task, on top of the SUN-RGBD dataset (Song et al., 2015). It uses a subset of RGBD images of SUN-RGBD dataset (Song et al., 2015) (7,699 images) and provides five language descriptions for the specified object

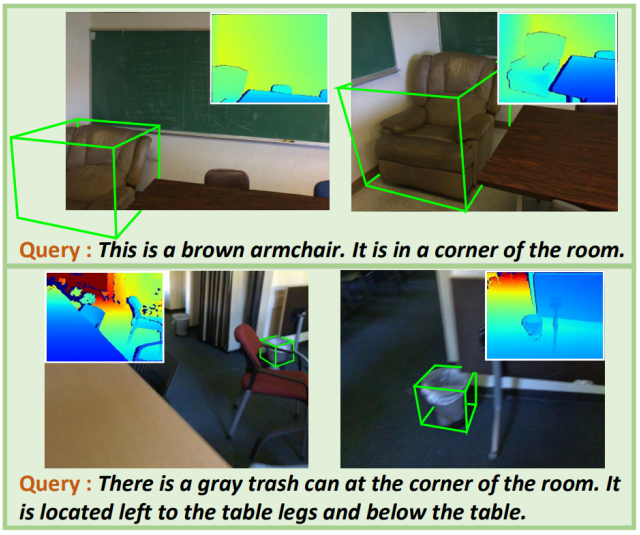


Figure 5.2: Examples of SUNRefer dataset (Liu et al., 2021a) (borrowed from the paper.) The figure shows the example descriptions and corresponding RGBD images. The task is to estimate a 3D bounding box given a pair of RGBD images and the corresponding language description that refers to an object in the images. Some objects can be partially observed as shown in the top left image or occluded as shown in the bottom left image.

for each image. Given a pair of RGBD images and a language description, a model has to output a 3D bounding-box of the referred object. Figure 5.2 and 5.3 show examples of the SUNRefer dataset (Liu et al., 2021a). As shown in Figure 5.2, some objects can be partially observed or occluded. A model needs to understand diverse language expressions as described in Figure 5.3.

Understanding various descriptions is one of the challenges due to the limited size of the language data. Another challenge is to estimate the complete 3D bounding-box from the partial observation. Figure 5.4 and 5.5 shows an example of an RGB image overlaid with 2D ground-truth bounding-boxes and a visualization of the 3D ground-truth bounding-boxes of the same scene with a pointcloud generated from the cropped area of the target object, respectively. In the example image, the target object is the leftmost dresser. The RGBD images are cropped from the target dresser bounding-box and transformed to form a pointcloud. Then it is clustered by DBSCAN (Ester et al., 1996) with Open3D (Zhou et al., 2018) and all clusters are rendered with different colors

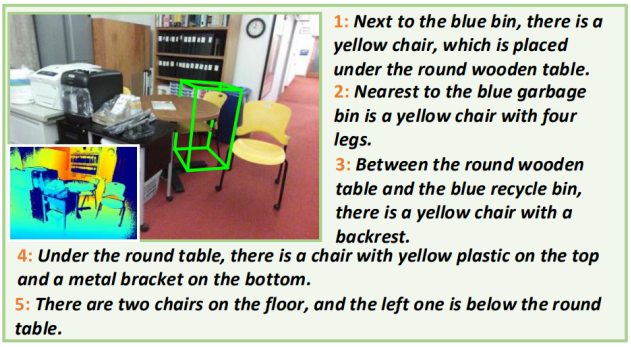


Figure 5.3: Example language descriptions in SUNRefer (Liu et al., 2021a) (borrowed from the paper.) The figure illustrates examples of five different language descriptions for a pair of RGBD images.

in Figure 5.5. In Figure 5.5, green box indicates the target 3D bounding-box and red boxes indicate bounding-boxes of other objects. A cyan box shows a 3D bounding-box that minimally covers the largest cluster from the pointcloud. When we compare the ground-truth target bounding-box and the heuristically generated bounding-box, they differ largely by their volumes. Recovering the original bounding-box is challenging because the depth image only captured the surface of the objects in the scene and the object is often partially observed due to the camera pose and the field of view. Points outside of the object such as those from the floor or wall can make the prediction more difficult.

We propose a model that consists of a 2D visual grounding model and a 3D bounding-box regression model. The 2D visual grounding model takes the RGB image and the language description to produce a 2D bounding-box in the image space. We fine-tuned OFA (Wang et al., 2022a) to the SUNRefer (Liu et al., 2021a) dataset. Once the 2D bounding-box is predicted, the RGBD images cropped by the bounding-box are transformed into a pointcloud. Then the 3D bounding-box regression model takes the pointcloud to regress a 3D bounding-box, the final output of the model. From the decomposition, we can leverage the vision-language model pre-trained with various datasets for better referring accuracy, track the source of error, and use existing 3D detection models without considering the language grounding part. The proposed model achieved state-of-the-art accuracy on the dataset.

Figure 5.4: Example image with the ground-truth 2D bounding-boxes. The figure shows an example RGB image with visualized ground-truth 2D bounding-boxes. The leftmost dresser is the target object described in language.

5.2 Related Work

5.2.1 Vision-Language Models

In recent years, transformer-based models (Vaswani et al., 2017) has been extensively proposed in various tasks, especially after the BERT model (Devlin et al., 2018). While BERT was proposed for natural language processing (NLP) tasks, the paradigm of fine-tuning to the downstream tasks from a pre-trained large model has been applied to different domains including multi-modal tasks such as VQA, Image Captioning, Image Retrieval, or Visual Reasoning (Young et al., 2014; Johnson et al., 2017; Yu et al., 2016; Antol et al., 2015). In addition, unifying a single architecture for multiple vision, language, and vision-language tasks at once has been proposed (Li et al., 2019b; Lu et al., 2019; Su et al., 2019; Chen et al., 2020c; Wang et al., 2021; 2022a; Lu et al., 2022a). From shared tasks and datasets, larger models can be trained and they generalize well. In the proposed method, we use a pre-trained vision-language model for the 2D visual grounding task as a solution for the sub-task of our problem.

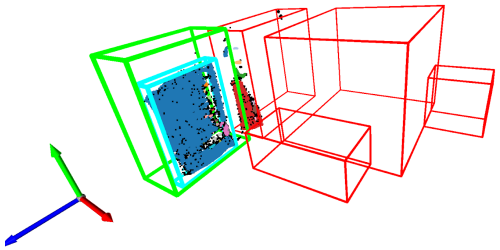


Figure 5.5: Visualization of 3D bounding-boxes and a cropped pointcloud from 2D bounding-box. The figure visualizes 3D ground-truth bounding-boxes from the same scene in Figure 5.4. The green box is indicating the target object ground-truth bounding-box while red boxes are the ground-truth bounding-boxes of other objects. Pointcloud is generated from an RGBD image pair and the 2D bounding-box of the target object is clustered and rendered with different colors. A cyan box is a minimal 3D bounding-box to the largest cluster for visualization purposes. As the depth image only represents the surface of objects, heuristically generated bounding-boxes cannot restore the original volume of the desired bounding-box. Points outside of the actual object area can also make a heuristic procedure challenging to apply.

5.2.2 3D Detection Models

The proposed method needs the ability to lift the RGBD images from a 2D bounding-box to a 3D bounding-box. During the process, the RGBD images are transformed into 6D pointclouds. For pointcloud classification and segmentation tasks, (Qi et al., 2017) has been extensively used as a fundamental backbone along with various approaches using graph models, convolutions, and transformers (Li et al., 2019a; Xu et al., 2021; Wang et al., 2019; Ma et al., 2022; Zhao et al., 2021; Qian et al., 2022). Datasets such as (Song et al., 2015; Dai et al., 2017) are collected from indoor scenes and propose 3D detection tasks. From the 3D pointcloud of a whole scene, the task is to propose multiple 3D bounding-boxes with corresponding class labels similar to object detection in 2D. For the detection task, multiple approaches have been proposed based on various architectures including transformers, sparse convolutions (Wang et al., 2022b; Ran et al., 2022; Vu et al., 2022; Liu et al., 2022; Rukhovich et al., 2021; Liu et al., 2021b; Misra et al., 2021; Graham et al., 2018; Vaswani et al., 2017).

In our model, we use pointmlp and FCAF3D (Ma et al., 2022; Rukhovich et al., 2021) for estimating a 3D bounding-box from the given pointcloud. Note that our model does not depend on a specific selection of sub-models thanks to its modularity.

5.2.3 Refer-it-in-RGBD

(Liu et al., 2021a) proposed the SUNRefer dataset and a model for visual grounding in RGBD. The proposed model has two stages: voxel-level matching for coarse localization and object-level matching for fine grounding. The first part of the model computes a heatmap in the voxel space to localize the relevant area to the language description. They used U-net with sparse convolution (Choy et al., 2019) with skip connections to generate a heatmap on voxels and concatenate language features from (Pennington et al., 2014) to the encoded voxel feature before the decoder. Then M seeds are sampled with a variant of farthest point sampling that uses heatmap values of the voxels to "ensure that a point with a higher heat value will have a higher chance to be chosen." After this step, the model has seeds that are sampled around the highly relevant regions to the referred object. The second part of the model employs PointNet++ (Qi et al., 2017) to aggregate features from M seeds and voting mechanism (Qi et al., 2019) to create K clusters of features and they are finally used to produce bounding-box proposals and corresponding scores. While they proposed two staged approaches, both modules are trained from scratch solely by the given dataset. As the fundamental detection process is done by the second module, we may not take the advantage of having interpretable representations such as a heatmap. A heatmap helps better initialization of seeds for voting, but not clear as it is a continuous distribution over voxels and it does not provide an analytic advantage of the detection process. On the other hand, our approach has a distinction between language grounding and detection processes which can benefit the understanding of the bottleneck. Furthermore, each module can leverage extra sources of data such as vision-language datasets or 3D detection datasets.

5.3 Problem Statement and Proposed Method

5.3.1 Problem Statement

SUNRefer is a dataset for visual grounding in RGBD images, proposed in (Liu et al., 2021a) based on SUNRGBD dataset (Song et al., 2015). From each RGBD image pair, one target object is selected and five language descriptions that identify the target object are provided. In total, 3989 image pairs are annotated along with 19945 language descriptions. The dataset provides class labels for the referred target objects and they are fine-grained. The total number of classes used in the dataset is 295 and 234 classes were used in the test dataset. SUNRGBD dataset (Song et al., 2015) provides 2D and 3D bounding-box annotations of objects in RGBD image pairs but not all of the objects are annotated or has corresponding bounding-boxes from 2D to 3D or vice versa. Given a pair of RGBD image and a language description, the task is to propose a 3D bounding-box of the referred object. It is similar to the task of ReferIt3D (Achlioptas et al., 2020) but it additionally requires 3D bounding-box proposal where ReferIt3D provided bounding-boxes as input.

5.3.2 Proposed Method

We propose a model for visual grounding in RGBD images consisting of two modules: a 2D visual grounding model and a 3D bounding-box regression model. Figure 5.1 shows the overview of the proposed model. First, a 2D visual grounding model takes an RGB image and the corresponding language description to estimate a 2D bounding-box of the referred object. From the figure, a yellow box on the image represents the predicted object state. We employ (Wang et al., 2022a) as the 2D visual grounding model and fine-tune the model with the dataset of interest. Then the RGBD images are cropped by the predicted 2D bounding-box and transformed into a pointcloud. In the figure, the first image in the bottom row shows the pointcloud generated from the cropped images. As seen in the image, even with a near-perfect 2D bounding-box, it is inevitable that the pointcloud contains points from outside of the target object. At last, the 3D bounding-box regression model takes the pointcloud to predict a 3D bounding-box. Note that the regression model

does not take the language description as input; standard 3D detection models can be used as the 3D bounding-box regression model. We choose FCAF3D (Rukhovich et al., 2021) pre-trained with SUNRGBD (Song et al., 2015) as the 3D bounding-box regression model.

Combining Multiple Regression Models

The modular architecture of the proposed model allows trying different 3D bounding-box models without re-training of the whole model. Unfortunately, 3D detection models are often trained on a few standard datasets with a limited number of class labels. For instance, (Dai et al., 2017) has 18 classes for the detection task while (Song et al., 2015) has 10 classes for the same task. Only eight classes (`'bed'`, `'table'`, `'sofa'`, `'chair'`, `'toilet'`, `'desk'`, `'bookshelf'`, `'bathtub'`) are shared among two sets of classes. Meanwhile, SUNRefer dataset (Liu et al., 2021a) uses 295 classes in language descriptions in total (234 classes in the test dataset.) About 37.8 % of the instances in the test dataset do not belong to the 20 detection classes combined from (Dai et al., 2017) and (Song et al., 2015) and 43.0 % of the test instances are outside of the 10 detection classes of SUNRGBD (Song et al., 2015).

As training a 3D detection model with a few hundred class labels on a small dataset is not feasible, we take an alternative approach. We trained another 3D bounding-box regression model based on (Ma et al., 2022). A new model is class-agnostic; it only regresses a 3D bounding-box without taking or predicting class labels. We use the second model for the instances outside of SUNRGBD detection classes. Instead of using a single 3D detection model as a 3D bounding-box regression model, we keep two different models based on the target class labels. Additionally, we train a binary target classification model using (Sanh et al., 2019). The language description determines whether the instance belongs to SUNRGBD detection classes or not. This works as a switch to 3D bounding-box regression models and achieves more than 96 % accuracy. Figure 5.6 shows a revised architecture with hybrid regression models.

We modified the original pointmlp (Ma et al., 2022) to regress 6-dimensional output for axis-aligned 3D bounding-box. The model computes 25 % and 75

% percentiles of each dimension from the points to get an initial estimation of the bounding-box. Then it scales and translates the initial estimate to get the finalized bounding-box. For training, we use three loss terms for center (L_{center}), size (L_{size}), and generalized IoU (L_{gIoU}) (Rezatofighi et al., 2019). Size and center vectors are penalized by smooth L1 loss. Generalized IoU provides a training signal even when two bounding-boxes are not overlapping each other. The overall loss L can be expressed as $L_{center} + L_{size} + \lambda L_{gIoU}$ and we set $\lambda = 0.1$.

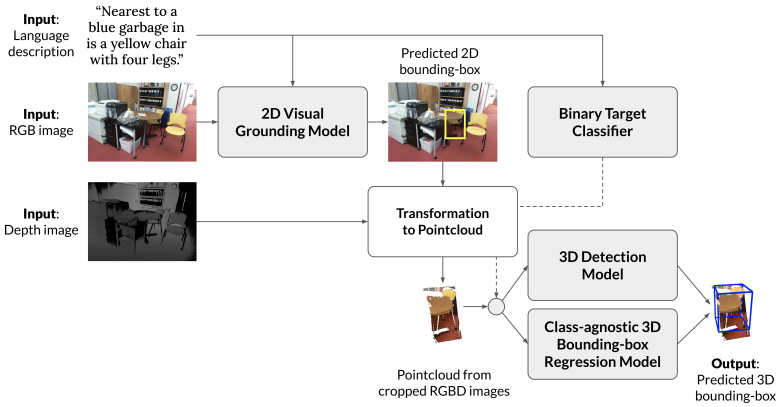


Figure 5.6: Overview of the proposed model with hybrid regression models. A binary target classification model is determining whether the instance belongs to the SUNRGBD (Song et al., 2015) detection classes or not. Based on the classification, the 3D bounding-box is predicted either by a pre-trained 3D detection model (Rukhovich et al., 2021) or a 3D bounding-box regression model (Ma et al., 2022). A dotted line represents a conditional signal that switches the flow for 3D bounding-box regression.

5.4 Experiments

5.4.1 Evaluation on SUNRefer

We evaluated the proposed method with various 3D regression models on SUNRefer dataset (Liu et al., 2021a). Refer-it-in-RGBD, the model from the original paper was used as a baseline. We tested four regression models: heuristics, pointmlp (Ma et al., 2022), FCAF3D (Rukhovich et al., 2021), and

the hybrid approach of combining pointmlp and FCAF3D based on the binary target classification. The heuristics model creates clusters from the pointcloud by DBSCAN (Ester et al., 1996), removes outlier clusters determined by the size of the clusters, and computes the minimum bounding-box of the inlier points. For FCAF3D (Rukhovich et al., 2021), we utilize all the proposed boxes and scores to determine the final 3D bounding-box. We compute the score $S = RC/V^{1/3}$ of each proposal from its confidence C, inlier point ratio R, and the volume of the bounding-box V. Then we choose the bounding-box with the largest score to be the final output.

Method	acc@0.25	acc@0.5
Refer-it-in-RGBD (Liu et al., 2021a)	49.6 %	<u>35.9</u> %
Ours with Heuristics	53.2 %	21.9 %
Ours with Pointmlp (Ma et al., 2022)	**60.9** %	32.3 %
Ours with FCAF3D (Rukhovich et al., 2021)	49.2 %	35.4 %
Ours with Hybrid	<u>58.6</u> %	**39.0** %

Table 5.1: Accuracy on SUNRefer (Liu et al., 2021a). Our model with the hybrid regression model outperformed the reported accuracies from (Liu et al., 2021a) by 10.0 % (acc@0.25) and 3.1 % (acc@0.50). The best and the second best numbers are highlighted and underlined, respectively. Note that the proposed model with FCAF3D (Rukhovich et al., 2021) as a regression model achieved comparable accuracies to the Refer-it-in-RGBD (Liu et al., 2021a) without extra steps for visual grounding.

For training of pointmlp (Ma et al., 2022), we used AdamW optimizer (Loshchilov and Hutter, 2017), learning rate 0.003, weight decay 0.01, embedding dimension 192, epochs 80, number of point samples 1000. For FCAF3D (Rukhovich et al., 2021), we used the pre-trained model on SUNRGBD (Song et al., 2015) provided by the authors.

Table 5.1 shows the accuracy of the proposed model with different 3D regression models and the model proposed in Refer-it-in-RGBD (Liu et al., 2021a). We follow the evaluation metric of accuracy values with thresholds of 0.25 and 0.50 intersection over union (IoU). In terms of accuracy with IoU threshold of 0.25, acc@0.25, our methods show higher values than the accuracy that was reported from Refer-it-in-RGBD (Liu et al., 2021a). However, when we compare acc@0.50, only the hybrid approach showed a higher value

than the reported accuracy. The proposed model with the heuristics-based regression shows poor acc@0.5 and it shows that it is hard to reliably estimate the complete bounding-box from the observation. Note that the proposed model with FCAF3D (Rukhovich et al., 2021) shows comparable accuracy values to the originally reported numbers while the only training part for the model was fine-tuning of the 2D visual grounding model.

5.4.2 Qualitative analysis

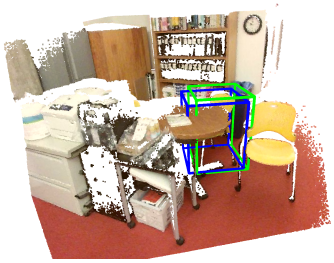
Figure 5.7 shows the pointcloud of an example scene and Figure 5.8 and Figure 5.9 show the predicted bounding-boxes with the whole pointcloud and the partial pointcloud cropped by the 2D bounding-box prediction result respectively. The green cuboids represent the ground-truth bounding-box and the blue cuboids represent the predicted bounding-boxes from the proposed model. While Figure 5.9a shows a successful example of the 3D bounding-box prediction, Figure 5.9b illustrates a failure case due to incorrect 2D bounding-box prediction. Note that the 3D bounding-box prediction is done from the cropped pointcloud and the pointcloud in Figure 5.8 is rendered only for visualization. For the example scene, four out of five utterances were succeeded in prediction with the threshold IoU of 0.5 and the failure case was instructed by the utterance: "under the round table, there is a chair with yellow plastic on the top and a metal bracket on the bottom."

Figure 5.10 illustrates a failure case where the 2D bounding-box prediction is correct but the 3D bounding-box prediction model fails to produce the bounding-box that represents the whole dresser from the partial observation. Figure 5.11 shows a successful example where both 2D and 3D bounding-box predictions are correct.

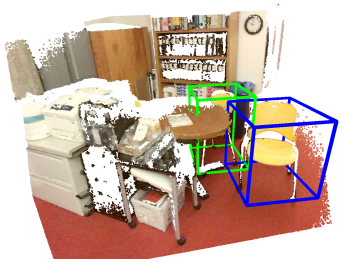
5.4.3 Evaluation with Ground-Truth 2D Bounding-Boxes

We investigate the effect of 3D bounding-box regression model by replacing the predicted 2D bounding-boxes with the ground-truth 2D bounding-boxes. Table 5.2 shows the accuracy values for variations of the proposed model. For the hybrid approach, the gain of the acc@0.25 and acc@0.50 from the 2D

Figure 5.7: A pointcloud of an example scene.



(a) An example of successfully predicted bounding-box. The pointcloud is rendered for visualization.



(b) An example of failed prediction of bounding-box. The pointcloud is rendered for visualization.

Figure 5.8: Examples of bounding-box predictions. The pointcloud is rendered for visualization.

(a) An example of successfully predicted bounding-box with the cropped pointcloud.

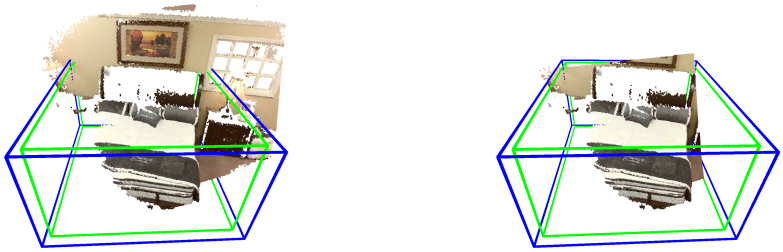
(b) An example of failed prediction of bounding-box from incorrectly cropped pointcloud.

Figure 5.9: Examples of bounding-box predictions with cropped pointclouds. While Figure 5.9a shows a successful example of prediction, Figure 5.9b fails due to a failure in 2D bounding-box prediction.

(a) An example of predicted bounding-box with the whole pointcloud.

(b) An example of predicted bounding-box from the cropped pointcloud.

Figure 5.10: A bounding-box prediction example with a small IoU. The 2D bounding-box prediction result correctly cropped the target object area as seen in Figure 5.10b but the 3D bounding-box prediction model failed to estimate the full-sized dresser from partial observation.



(a) An example of predicted bounding-box with the whole pointcloud.

(b) An example of predicted bounding-box from the cropped pointcloud.

Figure 5.11: A successful bounding-box prediction example. Both 2D and 3D bounding-box prediction results are correct.

bounding-box replacement was 13.7 % and 8 %. The highest acc@0.50 with ground-truth 2D bounding-boxes is 47 % achieved by the hybrid model. Due to the limitation of the 3D regression model, more than half of the instances failed in prediction with 0.5 IoU. A perfect 2D visual grounding contributes 17 % of the accuracy with the current 3D regression model.

Method	acc@0.25	acc@0.5
Refer-it-in-RGBD (Liu et al., 2021a)	49.6 %	35.9 %
Ours with Pointmlp (Ma et al., 2022)	60.9 %	32.3 %
Ours with Pointmlp (Ma et al., 2022) + 2D ground-truth	82.2 %	44.3 %
Ours with FCAF3D (Rukhovich et al., 2021)	49.2 %	35.4 %
Ours with FCAF3D (Rukhovich et al., 2021) + 2D ground-truth	55.8 %	39.6 %
Ours with Hybrid	58.6 %	39.0 %
Ours with Hybrid + 2D ground-truth	72.3 %	47.0 %

Table 5.2: Accuracy of models with and without ground-truth information on SUNRefer (Liu et al., 2021a). We replaced predicted 2D bounding-boxes with ground-truth information and computed accuracies. This eliminates the visual grounding error from the overall accuracy and measures the performance of 3D regression. Even with a perfect visual grounding in 2D images, our model still suffered from accurately estimating 3D bounding-boxes from pointclouds (up to the accuracy of 47.0 %.)

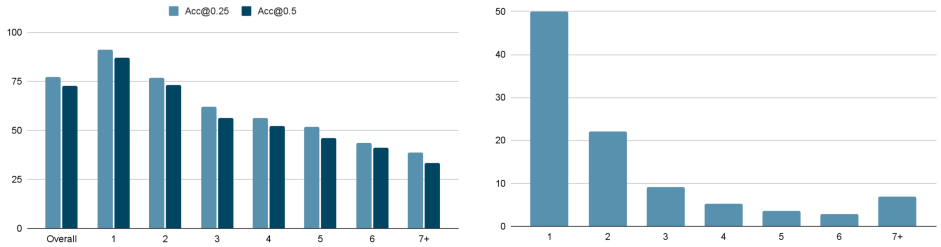
5.4.4 Analysis of 2D Visual Grounding Model Accuracy

The 2D visual grounding accuracy from OFA (Wang et al., 2022a) is shown in Table 5.3. We evaluated the model with and without fine-tuning. Even the zero-shot model was able to achieve around 70 % acc@0.25 and the predictions are stable as the difference between acc@0.25 and acc@0.5 from the models were 6.3 % and 4.4 % (without and with fine-tuning). The difference between acc@0.25 and acc@0.5 from the proposed model in Table 5.1 is 19.6 % which is 4.5 times larger than the same value from the 2D visual grounding task in the same dataset. Smaller accuracy difference between two IoU thresholds implies that the 2D visual grounding model localizes the target object well with the high IoU and it may imply that the main cause of the error can be identifying the correct object. In a realistic scenario of visual grounding with continuous observations and actions, it can be a useful property; once the 2D visual grounding is valid, we can keep track of the referred object from the image stream and reconstruct the object in 3D while actively exploring the unobserved part of the object.

Method	acc@0.25	acc@0.5
OFA (Wang et al., 2022a) zero-shot	69.6 %	63.3 %
OFA (Wang et al., 2022a) fine-tuned	77.0 %	72.6 %

Table 5.3: Accuracy of 2D visual grounding with (Wang et al., 2022a). We compare the accuracies of the 2D visual grounding model with and without fine-tuning. The fine-tuned model achieved better accuracy values (+7.4 % acc@0.25, +9.3 % acc@0.5) than those from the pre-trained model.

Figure 5.12 (a) illustrates accuracy of the 2D visual grounding model with respect to the number of objects in the image and Figure 5.12 (b) shows the proportions of the number of objects in the dataset. Note that about 50 % of the dataset contains only one target object in the scene and the 2D visual grounding can be considered as detection in this scenario. Also, more than 20 % of the dataset contains two target objects in images. For those single or double target object cases, the 2D visual grounding model achieves relatively high accuracy according to Figure 5.12 (a). Therefore, while increasing the overall accuracy in 2D visual grounding, the model may need to focus on



(a) Accuracy with respect to the number of target objects in the image.

(b) Proportion of the number of target objects from the dataset.

Figure 5.12: Statistics on the number of target objects in the image. Values on the x-axis show the number of target objects in the image. The graph in Figure 5.12 (a) shows the 2D visual grounding model accuracy values with respect to the number of target objects in images. The graph in Figure 5.12 (b) shows the proportion of the images with different numbers of target objects in the dataset.

multi-object cases where the number of target objects in the image is three or more.

Figure 5.13 show some example results on the image with multiple target objects. In the example image, there are four sofas on the left wall of the corridor. While the model successfully identified the left-most sofa, it failed to identify the right-most sofa. However, with another language expression, "far away," the model succeeded to identify the same object. We suspect that the model prefers certain forms of language expressions and they may have a higher chance to be correct. Additionally, the model may not recognize all the objects even though each looks the same. In this scenario, many language expressions relying on counting or anchoring multiple objects such as in between will fail. Providing structure to leverage the similarity between objects for multiple object scenes can be an interesting extension.

5.4.5 Analysis of 3D Bounding-Box Regression Model Accuracy in and out of Detection Classes

As shown in Table 5.1, The hybrid 3D bounding-box regression model achieved the best accuracy by taking the best of the both models, pointmlp (Ma et al., 2022) and FCAF3D (Rukhovich et al., 2021). For analysis of the accuracy,

(a) Example with "A red sofa left most."

(b) Example with "A red sofa right most."

(c) Example with "A red sofa far away."

Figure 5.13: Examples of 2D visual grounding results with multiple object in the image: (a) correctly identified the referred object, (b) failed to refer to the right most sofa as intended, (c) successfully detected the sofa from a different language expression.

we divide the dataset into two sets, in-class and out-class, based on the membership of the target class in the 10 classes for 3D detection on SUNRGBD dataset (Song et al., 2015) and evaluate the models on both subsets of the dataset. Table 5.4 shows accuracy values of 3D bounding-box regression models from two subsets of the dataset. We evaluated pointmlp (Ma et al., 2022), FCAF3D (Rukhovich et al., 2021), and FCAF3D pre-trained with ScanNet (Dai et al., 2017).

Method	Class	acc@0.25	acc@0.5
Ours with Pointmlp (Ma et al., 2022)	Overall	60.9 %	32.3 %
Ours with Pointmlp (Ma et al., 2022)	In-class	62.7 %	37.1 %
Ours with Pointmlp (Ma et al., 2022)	Out-class	58.2 %	24.9 %
Ours with FCAF3D (Rukhovich et al., 2021)	Overall	49.2 %	35.4 %
Ours with FCAF3D (Rukhovich et al., 2021)	In-class	58.9 %	48.2 %
Ours with FCAF3D (Rukhovich et al., 2021)	Out-class	34.4 %	15.9 %
Ours with FCAF3D (Rukhovich et al., 2021), ScanNet (Dai et al., 2017)	Out-class	42.7 %	19.1 %
Ours with Hybrid (FCAF3D + Pointmlp)	Overall	58.6 %	39.0 %
Ours with Hybrid (FCAF3D + FCAF3D on ScanNet)	Overall	52.5 %	36.7 %

Table 5.4: Accuracy of 3D bounding-box regressions models on in-class and out-class subsets of the dataset.

The pointmlp model was trained in a class-agnostic way while FCAF3D has classification loss and head. As a consequence, the acc@0.5 value of pointmlp in the out-class dataset (24.9 %) is relatively higher than that of FCAF3D (15.9 %). Note that the model with the same architecture but trained on ScanNet (Dai et al., 2017) with 18 classes shows better performance than

the FCAF3D model trained on SUNRGBD (Song et al., 2015) when we evaluate them on out-class subset of the dataset: (42.7 % vs. 34.4 %), (19.1 % vs. 15.9 %). Though their class labels share 8 classes, the model trained on ScanNet was explicitly trained with 10 extra classes and it might help the model have better performance on the out-class evaluation.

We found that the combination of the pre-trained detection models (from the last row of Table 5.4) can achieve a better accuracy than the reported numbers from Refer-it-in-RGBD (Liu et al., 2021a), it shows a necessity of general 3D detection models that can cover much larger sets of classes or support. In 2D detection, open world detection, large-vocabulary detection, or class-agnostic detection has been explored (Maaz et al., 2022; Joseph et al., 2021; Gupta et al., 2019) and we believe that open-vocabulary 3D detection or relevant tasks are of importance (Lu et al., 2022b).

5.5 Conclusion

We propose a framework for visual grounding in RGBD images by composing two sub-models: a 2D visual grounding model and a 3D bounding-box regression model. We employed a large, pre-trained vision-language model (Wang et al., 2022a) as a 2D visual grounding model to leverage the benefit of accessing web-scale image and language datasets. For the 3D bounding-box regression model, we combined a pre-trained 3D detection model on SUNRGBD (Rukhovich et al., 2021; Song et al., 2015) and a PointNet++-based model (Ma et al., 2022) as a class-agnostic bounding-box regressor to deal with long tail distribution of the referred object classes. Our method achieved the state-of-the-art result on the SUNRefer dataset (Liu et al., 2021a). It also produces interpretable 2D visual grounding results that provide analysis of the prediction error.

Chapter 6

Discussion

In this dissertation, we explore multiple tasks and methods for interactions in robotics applications. Throughout the models, we keep the framework that is composed of sub-models and produces interpretable intermediate results either useful in the error analysis or understanding the behavior of the model. We focus on two areas of applications: interactions in driving and visual grounding for robot navigation.

6.1 Summary

From various interactions that can occur in robotics applications, we first focus on the driving scenario specifically autonomous vehicles. Two models proposed for autonomous vehicles contribute to tasks involving different forms of interactions and are able to translate high-level goals into a sequence of controls.

In Chapter 2, we propose a model for the task of trajectory prediction, an essential problem of interaction between agents in the scene. The model divides the task into two sub-tasks by introducing a mode that describes the interaction from trajectories in an abstract and topologically invariant way. Mode values succinctly describe the interaction by the destination and the order of crossings between agents in the intersection. Our model predicts modes from trajectories as essential information of interaction and then future

trajectories that are consistent with the modes are predicted. Modes play a crucial role in reducing the complexity of the task and providing explainable clues for the outcome. In addition, mode prediction can be directly applied to navigation tasks since a single mode for the scene describes a particular form of interaction.

In Chapter 3, we focus on another type of interaction for autonomous driving, instruction following. It provides an efficient way of controlling a vehicle by language commands while the previous approach asked humans for decision-making and real-time control in discrete action space. The model introduces a hierarchy of sub-models to reduce the complexity of the task. The high-level sub-model translates the language command into a sequence of sub-tasks and the low-level sub-model executes actions in order to complete the sub-task. It produces a sequence of sub-tasks as an explainable result and it is robust to invalid or interrupting language commands thanks to the hierarchical structure.

For the second part, two models are proposed for the task of visual grounding for robot navigation. Object-oriented navigation is essential for many indoor robotics applications and visual grounding in indoor scenes is one interesting topic for navigation.

In Chapter 4, we propose a model for visual grounding in 3D and evaluate it on ReferIt3D dataset (Achlioptas et al., 2020). In a fully observed 3D scene with bounding-boxes and given a language description of an object in the scene, the model has to identify the referred object. We extend the pre-trained language model by augmenting spatial information of bounding-boxes, and the model becomes able to translate spatial relationships from language to geometry. In addition, we employ a 3D pointcloud classification model to predict class labels from bounding-boxes. This transforms a noisy pointcloud from a bounding-box into abstract information of class labels and 6D spatial vector and it makes the framework modular. The model achieves comparable accuracy to state-of-the-art models and provides an analysis of the error.

In Chapter 5, we further investigate the task of visual grounding in RGBD images. Visual grounding in 3D pointcloud has limitations that may not be suitable for real-time robot navigation. It does not provide a valid

viewpoint while the view-dependent utterance is given and relies on low-quality 3D reconstruction. Visual grounding in RGBD images does not have those limitations which can be useful for practice. We propose a model that combines a 2D visual grounding model and a 3D bounding-box regression model. The 2D bounding-box from the 2D visual grounding model provides a tool to localize the source of the error and a way of leveraging recent advances in large, pre-trained vision-language models. It reduces the complexity of the later stage; the active volume of the poincloud is restricted by the predicted 2D bounding-box. The 3D bounding-box regression model takes the pointcloud and predicts a 3D bounding-box. By separating from 2D visual grounding, either a 3D detection model or a pointcloud classification model without language can be used. Our model achieves the state-of-the-art result on the SUNRefer dataset (Liu et al., 2021a).

6.2 Future Directions

Although the proposed models have shown progress in various tasks with interactions, they are still far from the level that can be useful to our lives. Research focusing on a few directions may accelerate the realization of interactions in robotics applications.

6.2.1 General Description of Pairwise Interaction

Although the proposed models have shown progress in various tasks with interactions, they are still far from the level that can be useful to our lives. Research focusing on a few directions may accelerate the realization of interactions in robotics applications. In Chapter 2, we propose a winding number that describes a relative rotation of a pair of agents and use it as a description of the order of crossing at an unsignalized intersection. We only utilize the sign of the winding number since the order of crossing can be described in a discrete form. The magnitude of the winding number may represent the geometry of the trajectory. However, we were not able to find an interpretation of the magnitude of the winding number. In addition, scenarios such as lane changes may not cause negation of the sign of the winding number. Developing a

generalized topologically invariant representation can be applicable to wider applications. It can also be used as an auxiliary loss to existing trajectory prediction models; it can guide the model to focus on embedding the key abstract information of the scene-level interaction.

6.2.2 Bounding-Box-Level Augmentation for 3D Visual Grounding

LanguageRefer decomposes the classification part from the visual grounding and leverages the learned embedding space of the pre-trained language model with augmented spatial information for grounding. It enables various ablation studies such as replacing the predicted class labels with ground-truth class labels to get rid of the perception error from the model. From the abstracted input to the model, we may use low-cost data augmentation for 3D visual grounding. Recently, (Deitke et al., 2022) proposed a procedural creation of indoor scenes for multiple embodied AI tasks. Larger models can be used as a large set of randomly created scenes are provided and achieved state-of-the-art performance in various tasks. We may randomly alter the bounding-boxes or create bounding-boxes in the scene unless it harms the validity of the utterance.

6.2.3 Open-Vocabulary Object Detection in 3D

As discussed in Chapter 5, we suspect that the bottleneck of the model comes from the 3D bounding-box regression. Open-vocabulary or class-agnostic object detection in images has been actively studied but not much effort has been found in 3D. Although we lack a large collection of 3D datasets in comparison to 2D image datasets, we can still use 2D images and image features to shape the 3D feature space (Yang et al., 2021; Lu et al., 2022b). Also, models that can handle or predict depth images have been proposed recently (Girdhar et al., 2022; Lu et al., 2022a). We may see the model integrates 3D detection among vision-language tasks as vision-language models have been extended from language models.

6.2.4 Detection and Visual Grounding from RGBD Videos

In the direction of realistic robot navigation, it is natural to think of a visual grounding model from RGBD videos. We found that the prediction results of the 2D visual grounding model were stable in terms of IoU. However, even with the ground-truth 2D bounding-box of the referred object, we could not achieve 50 % accuracy with the threshold IoU of 0.5. While open-vocabulary 3D object detection may alleviate the issue, we may integrate the information from sequences to get a better estimate of the 3D bounding-box. If possible, we may end up integrating grounding into the 3D reconstruction pipeline.

6.2.5 Final Thoughts

In this thesis, we have presented a few methods for improving interactions in driving and indoor navigation. However, those tasks are preliminary steps toward the practical, complicated tasks and we have a long way to the end goal of general, interactive agents. Meanwhile, we are excited to witness rapid advances in deep learning and it will be interesting to see how these methods change the abilities of robots including the topics and methods of our work.

Bibliography

AI2-THOR: An Interactive 3D Environment for Visual AI. *ArXiv*, abs/1712.05474, 2017.

RoboTHOR: An Open Simulation-to-Real Embodied AI Platform. In *CVPR*, 2020.

P. Achlioptas. Referit3d benchmarks. URL `https://referit3d.github.io/benchmarks.html`.

P. Achlioptas, A. Abdelreheem, F. Xia, M. Elhoseiny, and L. Guibas. Referit3d: Neural listeners for fine-grained 3d object identification in real-world scenes. *16th European Conference on Computer Vision (ECCV)*, 2020.

A. Akbik, T. Bergmann, D. Blythe, K. Rasul, S. Schweter, and R. Vollgraf. Flair: An easy-to-use framework for state-of-the-art nlp. In *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59, 2019.

J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds, et al. Flamingo: a visual language model for few-shot learning. *arXiv preprint arXiv:2204.14198*, 2022.

S. Ammoun and F. Nashashibi. Real time trajectory prediction for collision risk estimation between vehicles. In *IEEE International Conference on Intelligent Computer Communication and Processing*, pages 417–422, 2009.

P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun,

J. Kosecka, J. Malik, R. Mottaghi, M. Savva, et al. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018a.

P. Anderson, A. X. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, and A. Zamir. On evaluation of embodied navigation agents. *ArXiv*, abs/1807.06757, 2018b.

P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. V. Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3674–3683, 2018c.

P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3674–3683, 2018d.

P. Anderson, A. Shrivastava, J. Truong, A. Majumdar, D. Parikh, D. Batra, and S. Lee. Sim-to-real transfer for vision-and-language navigation. *arXiv preprint arXiv:2011.03807*, 2020.

J. Andreas, D. Klein, and S. Levine. Modular multitask reinforcement learning with policy sketches. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 166–175. JMLR. org, 2017.

S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh. VQA: Visual Question Answering. In *International Conference on Computer Vision (ICCV)*, 2015.

I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

E. Artin. Theory of braids. *Annals of Mathematics*, 48(1):pp. 101–126, 1947.

T. Baltic, R. Hensley, and J. Salazar. *The trends transforming mobility's future*. McKinsey & Company, March 2019.

P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv e-prints*, art. arXiv:1806.01261, 2018.

M. A. Berger. Topological invariants in braid theory. *Letters in Mathematical Physics*, 55(3):181–192, 2001.

V. Blukis, N. Brukhim, A. Bennett, R. A. Knepper, and Y. Artzi. Following high-level navigation instructions on a simulated quadcopter with imitation learning. *arXiv preprint arXiv:1806.00047*, 2018.

V. Blukis, R. A. Knepper, and Y. Artzi. Few-shot object grounding and mapping for natural language robot instruction following. *ArXiv*, abs/2011.07384, 2020.

M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

R. Chandra, T. Guan, S. Panuganti, T. Mittal, U. Bhattacharya, A. Bera, and D. Manocha. Forecasting Trajectory and Behavior of Road-Agents Using Spectral Clustering in Graph-LSTMs. *arXiv e-prints*, art. arXiv:1912.01118, 2019.

M.-F. Chang, J. W. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, and J. Hays. Argoverse: 3d tracking and forecasting with rich maps. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

S. Changpinyo, P. Sharma, N. Ding, and R. Soricut. Conceptual 12m: Pushing web-scale image-text pre-training to recognize long-tail visual concepts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3558–3568, 2021.

D. S. Chaplot, K. M. Sathyendra, R. K. Pasumarthi, D. Rajagopal, and R. Salakhutdinov. Gated-attention architectures for task-oriented language grounding. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

D. Z. Chen, A. X. Chang, and M. Nießner. Scanrefer: 3d object localization in rgb-d scans using natural language. *16th European Conference on Computer Vision (ECCV)*, 2020a.

K. Chen, J. K. Chen, J. Chuang, M. V'azquez, and S. Savarese. Topological planning with transformers for vision-and-language navigation. *ArXiv*, abs/2012.05292, 2020b.

L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018.

Y.-C. Chen, L. Li, L. Yu, A. El Kholy, F. Ahmed, Z. Gan, Y. Cheng, and J. Liu. Uniter: Universal image-text representation learning. In *European conference on computer vision*, pages 104–120. Springer, 2020c.

K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In *Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, 2014.

C. Choy, J. Gwak, and S. Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019.

J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

M. J.-Y. Chung*, A. Pronobis*, M. Cakmak, D. Fox, and R. P. N. Rao. Autonomous question answering with mobile robots in human-populated environments. In *Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

S. L. Cleac'h, M. Schwager, and Z. Manchester. ALGAMES: A Fast Solver for Constrained Dynamic Games. In *Robotics: Science and Systems*, 2020.

D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

F. Codevilla, M. Miiller, A. López, V. Koltun, and A. Dosovitskiy. End-to-end driving via conditional imitation learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9. IEEE, 2018.

G. F. Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence*, 42(2):393–405, 1990.

M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The Cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5828–5839, 2017.

A. Das, S. Datta, G. Gkioxari, S. Lee, D. Parikh, and D. Batra. Embodied question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–10, 2018a.

A. Das, G. Gkioxari, S. Lee, D. Parikh, and D. Batra. Neural modular control for embodied question answering. *arXiv preprint arXiv:1810.11181*, 2018b.

M. Deitke, E. VanderBilt, A. Herrasti, L. Weihs, J. Salvador, K. Ehsani, W. Han, E. Kolve, A. Farhadi, A. Kembhavi, et al. Procthor: Large-scale embodied ai using procedural generation. *arXiv preprint arXiv:2206.06994*, 2022.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

N. Deo and M. M. Trivedi. Convolutional social pooling for vehicle trajectory prediction. In *IEEE Conference on Computer Vision and Pattern Recognition CVPR Workshops*, pages 1468–1476, 2018.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

S. Diao, W. Zhou, X. Zhang, and J. Wang. Prefix language models are unified modal learners. *arXiv preprint arXiv:2206.07699*, 2022.

D. Ding, F. Hill, A. Santoro, and M. Botvinick. Object-based attention for spatio-temporal reasoning: Outperforming neuro-symbolic models with flexible distributed architectures. *arXiv preprint arXiv:2012.08508*, 2020.

A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. of 2nd International Conference on Knowledge Discovery and*, pages 226–231, 1996.

M. Feng, Z. Li, Q. Li, L. Zhang, X. Zhang, G. Zhu, H. Zhang, Y. Wang, and A. Mian. Free-form description guided 3d visual graph network for object grounding in point cloud, 2021.

J. Firl, H. Stübing, S. A. Huss, and C. Stiller. Predictive maneuver evaluation for enhancement of car-to-x mobility data. In *IEEE Intelligent Vehicles Symposium*, pages 558–564, 2012.

W. Foundation. Wikimedia downloads. URL `https://dumps.wikimedia.org`.

D. Fridovich-Keil, E. Ratner, L. Peters, A. D. Dragan, and C. J. Tomlin. Efficient iterative linear-quadratic approximations for nonlinear multi-player general-sum differential games. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1475–1481, 2020.

D. Fried, R. Hu, V. Cirik, A. Rohrbach, J. Andreas, L.-P. Morency, T. Berg-Kirkpatrick, K. Saenko, D. Klein, and T. Darrell. Speaker-follower models for vision-and-language navigation. In *NeurIPS*, 2018.

A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

T. Gindele, S. Brechtel, and R. Dillmann. Learning driver behavior models from traffic observations for decision making and planning. *IEEE Intelligent Transportation Systems Magazine*, 7(1):69–79, 2015.

R. Girdhar and D. Ramanan. CATER: A diagnostic dataset for Compositional Actions and TEmporal Reasoning. In *ICLR*, 2020.

R. Girdhar, M. Singh, N. Ravi, L. van der Maaten, A. Joulin, and I. Misra. Omnivore: A Single Model for Many Visual Modalities. In *CVPR*, 2022.

B. Graham, M. Engelcke, and L. van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. *CVPR*, 2018.

A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2255–2264, 2018.

A. Gupta, P. Dollar, and R. Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5356–5364, 2019.

I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. M. Botvinick, S. Mohamed, and A. Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations (ICLR)*, 2017.

Y. Hong, Q. Wu, Y. Qi, C. Rodriguez-Opazo, and S. Gould. A recurrent vision-and-language bert for navigation. *ArXiv*, abs/2011.13922, 2020.

J. Houston, G. Zuidhof, L. Bergamini, Y. Ye, A. Jain, S. Omari, V. Iglovikov, and P. Ondruska. One Thousand and One Hours: Self-driving Motion Prediction Dataset. *arXiv e-prints*, art. arXiv:2006.14480, 2020.

M. Z. Irshad, C.-Y. Ma, and Z. Kira. Hierarchical cross-modal agent for robotics vision-and-language navigation. *ArXiv*, abs/2104.10674, 2021.

D. Isele, R. Rahimi, A. Cosgun, K. Subramanian, and K. Fujimura. Navigating occluded intersections with autonomous vehicles using deep reinforcement learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2034–2039, 2018.

J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. L. Zitnick, and R. Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*, 2017.

K. Joseph, S. Khan, F. S. Khan, and V. N. Balasubramanian. Towards open world object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5830–5840, 2021.

A. Kamath, M. Singh, Y. LeCun, I. Misra, G. Synnaeve, and N. Carion. Mdetr–modulated detection for end-to-end multi-modal understanding. *arXiv preprint arXiv:2104.12763*, 2021.

S. Kazemzadeh, V. Ordonez, M. Matten, and T. L. Berg. Referitgame: Referring to objects in photographs of natural scenes. In *EMNLP*, 2014.

L. Ke, X. Li, Y. Bisk, A. Holtzman, Z. Gan, J. Liu, J. Gao, Y. Choi, and S. Srinivasa. Tactical rewind: Self-correction via backtracking in vision-and-language navigation. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6734–6742, 2019a.

L. Ke, X. Li, Y. Bisk, A. Holtzman, Z. Gan, J. Liu, J. Gao, Y. Choi, and S. Srinivasa. Tactical rewind: Self-correction via backtracking in vision-and-language navigation. *arXiv preprint arXiv:1903.02547*, 2019b.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13): 3521–3526, 2017. doi: 10.1073/pnas.1611835114. URL `https://www.pnas.org/doi/abs/10.1073/pnas.1611835114`.

E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi. AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv*, 2017.

J. Krantz, E. Wijmans, A. Majumdar, D. Batra, and S. Lee. Beyond the nav-graph: Vision-and-language navigation in continuous environments. In *European Conference on Computer Vision*, pages 104–120. Springer, 2020.

S. Krishnan, R. Fox, I. Stoica, and K. Goldberg. Ddco: Discovery of deep continuous options for robot learning from demonstrations. *arXiv preprint arXiv:1710.05421*, 2017.

A. Ku, P. Anderson, R. Patel, E. Ie, and J. Baldridge. Room-across-room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. In *EMNLP*, 2020.

A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, A. Kolesnikov, et al. The open images dataset v4. *International Journal of Computer Vision*, 128(7):1956–1981, 2020.

G. Li, M. Muller, A. Thabet, and B. Ghanem. Deepgcns: Can gcns go as deep as cnns? In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9267–9276, 2019a.

J. Li, F. Yang, M. Tomizuka, and C. Choi. EvolveGraph: Multi-Agent Trajectory Prediction with Dynamic Relational Reasoning. *arXiv e-prints*, art. arXiv:2003.13924, 2020.

L. H. Li, M. Yatskar, D. Yin, C.-J. Hsieh, and K.-W. Chang. Visualbert: A simple and performant baseline for vision and language. *arXiv preprint arXiv:1908.03557*, 2019b.

X. Li, X. Ying, and M. C. Chuah. Grip: Graph-based interaction-aware trajectory prediction. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 3960–3966, 2019.

J. Liang, L. Jiang, K. Murphy, T. Yu, and A. Hauptmann. The Garden of Forking Paths: Towards Multi-Future Trajectory Prediction. *arXiv e-prints*, art. arXiv:1912.06445, 2019.

X. Liang, T. Wang, L. Yang, and E. Xing. CIRL: Controllable imitative reinforcement learning for vision-based self-driving. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 584–599, 2018.

T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

H. Liu, A. Lin, X. Han, L. Yang, Y. Yu, and S. Cui. Refer-it-in-rgbd: A bottom-up approach for 3d visual grounding in RGBD images. *CoRR*, abs/2103.07894, 2021a. URL `https://arxiv.org/abs/2103.07894`.

H. Liu, J. Zhang, K. Yang, X. Hu, and R. Stiefelhagen. Cmx: Cross-modal fusion for rgb-x semantic segmentation with transformers. *arXiv preprint arXiv:2203.04838*, 2022.

Y. Liu, B. Wan, X.-D. Zhu, and X. He. Learning cross-modal context graph for visual grounding. *ArXiv*, abs/1911.09042, 2020.

Z. Liu, Z. Zhang, Y. Cao, H. Hu, and X. Tong. Group-free 3d object detection via transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2949–2958, 2021b.

I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

J. Lu, D. Batra, D. Parikh, and S. Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *Advances in neural information processing systems*, 32, 2019.

J. Lu, C. Clark, R. Zellers, R. Mottaghi, and A. Kembhavi. Unified-io: A unified model for vision, language, and multi-modal tasks. *arXiv preprint arXiv:2206.08916*, 2022a.

Y. Lu, C. Xu, X. Wei, X. Xie, M. Tomizuka, K. Keutzer, and S. Zhang. Open-vocabulary 3d detection via image-level class and debiased cross-modal contrastive learning. *arXiv preprint arXiv:2207.01987*, 2022b.

M.-T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

C.-Y. Ma, J. Lu, Z. Wu, G. Al-Regib, Z. Kira, R. Socher, and C. Xiong. Self-monitoring navigation agent via auxiliary progress estimation. *ArXiv*, abs/1901.03035, 2019a.

C.-Y. Ma, J. Lu, Z. Wu, G. AlRegib, Z. Kira, R. Socher, and C. Xiong. Self-monitoring navigation agent via auxiliary progress estimation. *arXiv preprint arXiv:1901.03035*, 2019b.

X. Ma, C. Qin, H. You, H. Ran, and Y. Fu. Rethinking network design and local geometry in point cloud: A simple residual mlp framework. *arXiv preprint arXiv:2202.07123*, 2022.

M. Maaz, H. Rasheed, S. Khan, F. S. Khan, R. M. Anwer, and M.-H. Yang. Class-agnostic object detection with multi-modal transformer. In *17th European Conference on Computer Vision*. Springer, 2022.

J. Mao, J. Huang, A. Toshev, O.-M. Camburu, A. Yuille, and K. Murphy. Generation and comprehension of unambiguous object descriptions. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11–20, 2016.

C. Mavrogiannis, J. A. DeCastro, and S. S. Srinivasa. Implicit Multiagent Coordination at Unsignalized Intersections via Multimodal Inference Enabled by Topological Braids. *arXiv e-prints*, art. arXiv:2004.05205, 2020.

C. I. Mavrogiannis and R. A. Knepper. Multi-agent path topology in support of socially competent navigation planning. *The International Journal of Robotics Research*, 38(2-3):338–356, 2019.

C. I. Mavrogiannis and R. A. Knepper. Multi-agent trajectory prediction and generation with topological invariants enforced by hamiltonian dynamics. In *Algorithmic Foundations of Robotics XIII*, pages 744–761, Cham, 2020. Springer International Publishing.

C. I. Mavrogiannis, V. Blukis, and R. A. Knepper. Socially competent navigation planning by deep learning of multi-agent path topologies. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6817–6824, 2017.

I. Misra, R. Girdhar, and A. Joulin. An end-to-end transformer model for 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2906–2917, 2021.

A. Monti, A. Bertugli, S. Calderara, and R. Cucchiara. DAG-Net: Double Attentive Graph Neural Network for Trajectory Forecasting. *arXiv e-prints*, art. arXiv:2005.12661, 2020.

M. Müller, A. Dosovitskiy, B. Ghanem, and V. Koltun. Driving policy transfer via modularity and abstraction. *arXiv preprint arXiv:1804.09364*, 2018.

G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

C. Paxton, V. Raman, G. D. Hager, and M. Kobilarov. Combining neural networks and tree search for task and motion planning in challenging environments. *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, 2017.

C. Paxton, Y. Bisk, J. Thomason, A. Byravan, and D. Fox. Prospection: Interpretable plans from language by predicting the future. *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

J. Pennington, R. Socher, and C. Manning. GloVe: global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

A. Pierson, W. Schwarting, S. Karaman, and D. Rus. Navigating congested environments with risk level sets. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5712–5719, 2018.

B. A. Plummer, L. Wang, C. M. Cervantes, J. C. Caicedo, J. Hockenmaier, and S. Lazebnik. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. *International Journal of Computer Vision*, 123:74–93, 2015.

C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NIPS*, 2017.

C. R. Qi, O. Litany, K. He, and L. J. Guibas. Deep hough voting for 3d object detection in point clouds. In *proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9277–9286, 2019.

Y. Qi, Q. Wu, P. Anderson, X. Wang, W. Wang, C. Shen, and A. V. Hengel. Reverie: Remote embodied visual referring expression in real indoor environments. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9979–9988, 2020.

G. Qian, Y. Li, H. Peng, J. Mai, H. A. A. K. Hammoud, M. Elhoseiny, and B. Ghanem. Pointnext: Revisiting pointnet++ with improved training and scaling strategies. *arXiv preprint arXiv:2206.04670*, 2022.

A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021.

A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

H. Ran, J. Liu, and C. Wang. Surface representation for point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18942–18952, 2022.

H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese. Generalized intersection over union. June 2019.

N. Rhinehart, R. McAllister, and S. Levine. Deep Imitative Models for Flexible Inference, Planning, and Control. *arXiv e-prints*, art. arXiv:1810.06544, 2018.

J. Roh, C. Paxton, A. Pronobis, A. Farhadi, and D. Fox. Conditional driving from natural language instructions. *ArXiv*, abs/1910.07615, 2019.

D. Rukhovich, A. Vorontsova, and A. Konushin. Fcaf3d: Fully convolutional anchor-free 3d object detection. *arXiv preprint arXiv:2112.00322*, 2021.

C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. Denton, S. K. S. Ghasemipour, B. K. Ayan, S. S. Mahdavi, R. G. Lopes, et al. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*, 2022.

M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018.

V. Sanh, L. Debut, J. Chaumond, and T. Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra. Habitat: A platform for embodied ai research. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9338–9346, 2019.

E. Schmerling, K. Leung, W. Vollprecht, and M. Pavone. Multimodal probabilistic model-based planning for human-robot interaction. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

P. Shah, M. Fiser, A. Faust, J. C. Kew, and D. Hakkani-Tur. Followne: Robot navigation by following natural language directions with deep reinforcement learning. *arXiv preprint arXiv:1805.06150*, 2018.

B. Shen, F. Xia, C. Li, R. Martín-Martín, L. Fan, G. Wang, C. Pérez-D'Arpino, S. Buch, S. Srivastava, L. Tchapmi, et al. igibson 1.0: a simulation environment for interactive tasks in large realistic scenes. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7520–7527. IEEE, 2021.

K. Shiarlis, M. Wulfmeier, S. Salter, S. Whiteson, and I. Posner. TACO: Learning task decomposition via temporal alignment for control. *arXiv preprint arXiv:1803.01840*, 2018.

M. Shridhar, J. Thomason, D. Gordon, Y. Bisk, W. Han, R. Mottaghi, L. Zettlemoyer, and D. Fox. ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020a. URL `https://arxiv.org/abs/1912.01734`.

M. Shridhar, J. Thomason, D. Gordon, Y. Bisk, W. Han, R. Mottaghi, L. Zettlemoyer, and D. Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10737–10746, 2020b.

A. Shrivastava, K. Gopalakrishnan, Y. Liu, R. Piramuthu, G. Tur, D. Parikh, and D. Hakkani-Tur. Visitron: Visual semantics-aligned interactively trained object-navigator. *ArXiv*, abs/2105.11589, 2021.

S. Song, S. P. Lichtenberg, and J. Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 567–576, 2015.

S. S. Srinivasa, P. Lancaster, J. Michalove, M. Schmittle, C. Summers, M. Rockett, J. R. Smith, S. Choudhury, C. Mavrogiannis, and F. Sadeghi. MuSHR: A Low-Cost, Open-Source Robotic Racecar for Education and Research. *arXiv e-prints*, art. arXiv:1908.08031, 2019.

W. Su, X. Zhu, Y. Cao, B. Li, L. Lu, F. Wei, and J. Dai. Vl-bert: Pre-training of generic visual-linguistic representations. *arXiv preprint arXiv:1908.08530*, 2019.

C. Tang and R. R. Salakhutdinov. Multiple futures prediction. In *Advances in Neural Information Processing Systems*, pages 15398–15408, 2019.

R. Taori, A. Dave, V. Shankar, N. Carlini, B. Recht, and L. Schmidt. Measuring robustness to natural distribution shifts in image classification. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. URL `https://arxiv.org/abs/2007.00644`.

B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li. Yfcc100m: The new data in multimedia research. *Communications of the ACM*, 59(2):64–73, 2016.

US Department of Transportation, Federal Highway Administration. Unsignalized intersections, 2018. URL `https://safety.fhwa.dot.gov/intersection/conventional/unsignalized/`.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

T. Vu, K. Kim, T. M. Luu, T. Nguyen, and C. D. Yoo. Softgroup for 3d instance segmentation on point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2708–2717, 2022.

P. Wang, A. Yang, R. Men, J. Lin, S. Bai, Z. Li, J. Ma, C. Zhou, J. Zhou, and H. Yang. Unifying architectures, tasks, and modalities through a simple sequence-to-sequence learning framework. *arXiv preprint arXiv:2202.03052*, 2022a.

Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019.

Y. Wang, X. Chen, L. Cao, W. Huang, F. Sun, and Y. Wang. Multimodal token fusion for vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12186–12195, 2022b.

Z. Wang, J. Yu, A. W. Yu, Z. Dai, Y. Tsvetkov, and Y. Cao. Simvlm: Simple visual language model pretraining with weak supervision. *arXiv preprint arXiv:2108.10904*, 2021.

T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush. Huggingface's transformers: State-of-the-art natural language processing, 2020.

H. Xu, Y. Gao, F. Yu, and T. Darrell. End-to-end learning of driving models from large-scale video datasets. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2174–2182, 2017.

M. Xu, R. Ding, H. Zhao, and X. Qi. Paconv: Position adaptive convolution with dynamic kernel assembling on point clouds. In *CVPR*, 2021.

S. Yang, G. Li, and Y. Yu. Dynamic graph attention for referring expression comprehension. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4643–4652, 2019.

Z. Yang, S. Zhang, L. Wang, and J. Luo. SAT: 2d semantics assisted training for 3d visual grounding. *CoRR*, abs/2105.11450, 2021. URL `https://arxiv.org/abs/2105.11450`.

L. Ye, M. Rochan, Z. Liu, and Y. Wang. Cross-modal self-attention network for referring image segmentation, 2019.

K. Yi, C. Gan, Y. Li, P. Kohli, J. Wu, A. Torralba, and J. B. Tenenbaum. CLEVRER: collision events for video representation and reasoning. In *ICLR*, 2020.

P. Young, A. Lai, M. Hodosh, and J. Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78, 2014. doi: 10.1162/tacl_a_00166. URL https://aclanthology.org/Q14-1006.

L. Yu, P. Poirson, S. Yang, A. Berg, and T. L. Berg. Modeling context in referring expressions. *ArXiv*, abs/1608.00272, 2016.

L. Yu, Z. L. Lin, X. Shen, J. Yang, X. Lu, M. Bansal, and T. L. Berg. Mattnet: Modular attention network for referring expression comprehension. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1307–1315, 2018.

Z. Yuan, X. Yan, Y. Liao, R. Zhang, Z. Li, and S. Cui. Instancerefer: Cooperative holistic understanding for visual grounding on point clouds through instance multi-level contextual referring, 2021.

H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16259–16268, 2021.

Q.-Y. Zhou, J. Park, and V. Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.

Appendix A

Additional Materials for Multimodal Trajectory Prediction via Topological Invariance for Navigation at Uncontrolled Intersections

A.1 Cost functions definition

We now provide the formulation for individual cost functions used in Eq. 2.4:

Smoothness cost: $J_{sm}(\mathbf{X}) = \sum_{t=0}^{T-1} \gamma^t ||v_{t+1} - v_t||$,

Cross-track error: $J_{ref}(\mathbf{X}) = dist(\tau^t, X^t) * sin(\psi_{\tau^t} - \psi_{X^t})$,

Collision cost: $J_{col}(\mathbf{X}) = \sum_{t=0}^{T} \gamma^t c_t(X^t)$, where $c_t(X^t) = \begin{cases} 0 & \text{if } D_i(X^t) \geq d_{min} \\ 1 & \text{else} \end{cases}$,

Likelihood cost: $J_p(\mathbf{X}) = \frac{1}{p_{m_i}}$

where v_t is the velocity at timestep t. $dist(\tau^t, X^t)$ is the euclidean distance between agent position and reference waypoint at timestep t. ψ_{τ^t} is the heading angle on reference waypoint and ψ_{X^t} is the heading angle of the agent, at timestep t. p_{m_i} is the likelihood for mode m_i. D_i is a distance function and d_{min} is the distance threshold for collision. To reduce the computational burden of collision-checking, we use the circle-based collision checking method by reducing the car footprint to a set of three circles covering the car volume. Any object is thus in collision with the vehicle, if its distance from the center is less than the specified threshold.

A.2 Implementation Details

In this section, we provide more details about the data generation pipeline, the training process and the navigation experiments.

A.2.1 Data generation

We generated three different datasets using our custom simulator by varying three simulation parameters: a) agents' configurations (combinations of starting locations and intended destinations); b) agents' target speed; c) agents' acceleration/deceleration.

- Two agents
  - Number of configurations: $N_{d2} = 3^3 = 27$ (taking all possible configurations)
  - Number of speeds: $N_{s2} = 7^2$ (sampled from $[2.8 - 12.5]$ m/s with step size 1.4 m/s)
  - Number of accelerations: $N_{a2} = 10^2$ (sampled uniformly from $[1 - 5]$) m/s
  - Number of total episodes: $N_{e2} = N_{d2} \times N_{s2} \times N_{a2} \approx 132K$
- Three agents
  - Number of configurations: $N_{d3} = N_{d2} \times 2 \times 3 = 162$

  - Number of speeds: $N_{s3} = 7^3$ (sampled from $[2.8 - 12.5]$ m/s, step size 2.8 m/s)
  - Number of accelerations: $N_{a3} = 5^3$ (sampled uniformly from $[1 - 5]$) m/s
  - Number of total episodes: $N_{e3} = N_{d3} \times N_{s3} \times N_{a3} \approx 1,29M$

- Four agents
  - Number of configurations: $N_{d4} = 3^4$
  - Number of speeds: $N_{s4} = 3^4$ (sampled from $[2.8 - 9.7]$ m/s, step size 2.8 m/s)
  - Number of accelerations: $N_{a4} = 3^4$ (sampled uniformly from $[1 - 5]$) m/s
  - Number of total episodes: $N_{e4} = N_{d4} \times N_{s4} \times N_{a4} \approx 531K$

The final datasets were extracted upon pruning out episodes including collisions (see main paper for dataset sizes).

A.2.2 Training

Dataset split. We split the graph sequence $\mathbf{g} = g^{1:H}$ into a sequence of history graphs $\mathbf{g}^p = g^{1:h_p}$ and a sequence of target graphs $\mathbf{g}^f = g^{h_p+1:H}$.

Trajectory-reconstruction model training. The trajectory-reconstruction model takes a graph g^t and a hidden graph g_h^t and predicts a graph at next timestep $\hat{g}^{t+1}$ along with an updated hidden graph g_h^{t+1} (see Fig. 2.4a). Training procedure has two stages: reading history sequence and predicting future sequence. In the first stage where $t < h_p$, the model takes g^t from the history graphs $\mathbf{g}^p$ and the output graph in this stage is discarded. Note that the hidden graph remains updated to keep the information propagated through time. In the second stage where $t \geq h_p$, the input graph is taken from the output graph at the previous timestep $t - 1$, $\hat{g}^t$. The output graphs are collected to form the future trajectory. This process is repeated until the full predicted graph sequence $\hat{\mathbf{g}}^f = \hat{g}^{h_p+1:H}$ is produced. Loss is computed on $\hat{\mathbf{g}}^f$ and $\mathbf{g}^f$ by a

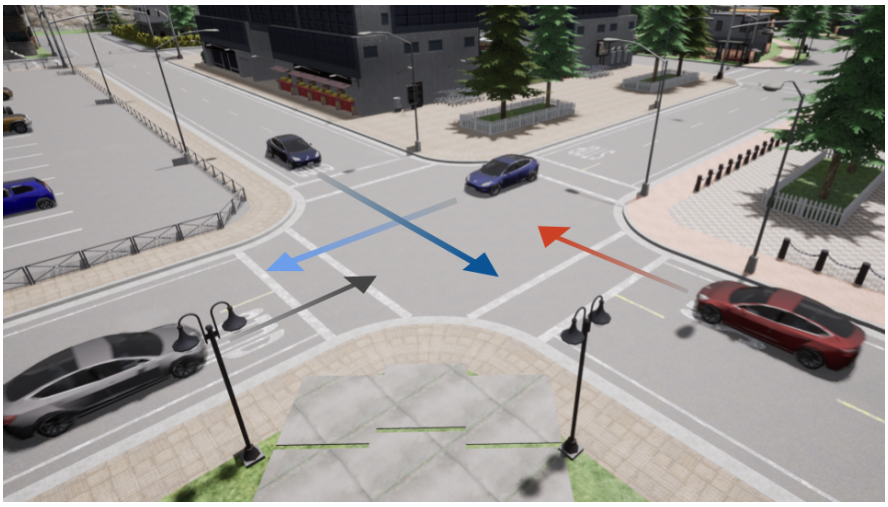


Figure A.1: An example of the 4-agent scenario at an uncontrolled intersection in CARLA

mean squared error function. In the training, we provide ground-truth mode signals.

Mode-prediction model training. The mode-prediction model shares a similar architecture (see Fig. 2.4b), with the difference that we provide the ground-truth sequence of graphs $\mathbf{g}$ to the model and collect mode signals $\hat{m}^t$ for $t \geq h_p$.

Hyperparameters. We trained the models considered (both MTP and MFP) using PyTorch. For **MTP**, we set the following options – (Optimizer: Adam; Learning rate: 1×10^{-3}; Gradient clip: 1.0; Hidden unit size: 30. For **MFP**, we made use of the official implementation (`https://github.com/apple/ml-multiple-futures-prediction`) setting the following options – (Number of modes: 3; Subsampling: 2; Encoder size: 64; Decoder size: 128; Neighbor encoding size: 8; Neighbor attention embedding size: 20; Remove y mean: False; Use forcing: classmate forcing).

A.2.3 Experiment Setup in CARLA

The navigation experiments are conducted at an uncontrolled intersection within the `Town04` map of CARLA. For each experiment, agents are spawned on a specified side of the intersection and headed towards a destination waypoint corresponding to an intended direction (left, right, forward). At the beginning, all the agents are controlled using the Autopilot provided by CARLA. The ego agent controller switches to our method (MTPnav) or one of the baselines when the ego-agent reaches a distance of 25m from the center of the intersection, and switches back to Autopilot after crossing the intersection. The weights used in the cost function (see main paper) are set to the following values: $w_{sm} = 50$, $w_{ref} = 100$, $w_{col} = 10000$, $w_p = 1$.

Appendix B

Additional Materials for Conditional Driving from Natural Language Instructions

B.1 Language Generation

To create the language dataset, we originally conducted a two-player game with human subjects to collect speech signals of commands and corresponding driving controls. In the game, two players are asked to collaboratively drive a car to reach three randomly spawned goals. While one player drives a car without knowing where the destination is, the other player reads a map and gives direction to the driver. After transcribing the collected audio data, we removed the sentences with actions that cannot be taken in the current environment and removed expressions mentioning objects or structures. Then we divide expressions into prefix, body, and suffix and cluster those expressions to transform the sentence into templates. Finally, we generated sentences for each combination of sub-tasks with the templates. In the implementation, we used a keyword to represent each type of combination.

We counted the number of expressions in the raw dataset for each. Table B.1 shows the number of sentences for each keyword. The keyword '`other`' and '`extra`' represents the sentences contain the actions that cannot be taken

Keywords	Sources	
	game	templates
`left`	1,093	150
`right`	1,016	150
`straight`	1,199	454
`left,left`	3	269,550
`left,right`	20	135,000
`left,straight`	22	408,600
`right,left`	28	135,000
`right,right`	2	269,550
`right,straight`	14	408,600
`straight,straight`	1	85
`first,left`	9	102,150
`first,right`	4	102,150
`second,left`	97	105,450
`second,right`	89	105,450
`other`	913	N/A
`extra`	379	N/A

Table B.1: Number of sentences collected from the preliminary two-player driving game (game) and the templates for training (templates). From the game, we also classified sentences which are out of actions defined in the environment we used in the training as `other` and sentences which do not contain meaningful commands as `extra`.

in the current environment and the sentences that do not have any meaningful commands, respectively. The total of the counted expressions is 4889 and 4600 sentences have single command '`left`', '`right`', '`straight`', '`other`', '`extra`'.

This high percentage of single command is due to the nature of the language in the driving setting where the reactive instruction should be given within a short amount of time. This shows that concentrating on instructive sentences is a reasonable approach in the context of driving. Another point worth noting on the dataset is that people make a lot of mistakes in commanding or driving. Sometimes a commander repeats the same command until the driver finishes that action or cancels previous actions by adding a new command. Manual pruning was necessary to make the dataset feasible to train on. As a trade-off, the distribution of sentences can be made more realistic than that coming from pre-recorded driving trajectories.

As a result of this dataset imbalance, we augment natural-language phrases according to a couple of simple rules. For the sentences with two commands, we concatenated expressions from a single keyword. The dictionary shows the 14 keywords we used in the paper and the corresponding number of sentences is shown in Table B.1.

When we use these sentences in the training, we draw a sentence from these lists with uniform distribution. For ordinary keywords, two groups of lists were used: one from the direct combination of two sentences of single keywords and the other one from the replacement of the keyword, such as replacing '`left`' with '`second left`'. In the training, for those with multiple groups, the group is first drawn and then the sentence is drawn from the group.

B.2 Language Examples

We show two examples of transcribed speech data from the preliminary two-player game experiments. Note that certain types of behavior such as going backward, reaching the target, and slowing down were excluded from the training dataset.

“oh there’s a map all right go straight", “and you’re going to turn right", “that’s good keep going straight", “and take your first left", “and slow down", “all right can you see the green square", “great", “okay so now you want to go straight", “and you’ll take a left at the first building", “that’s good that’s good keep going straight", “and take a left", “and take a right", “now straight", “and take a left", “went a little too far so reverse and back it up", “all right you doing good", “go a little bit forward", “yep there it is", “you got it", “okay so now you’re going to want to turn around", “you’re going to back it up a little bit", “looking good no collisions so far", “all right now you’ll take a right", “yep", “now go straight", “now take a left", “take a right", “go straight as fast as you can", “and you’ll take a left", “now right", “and the exit is right up here", “congratulations".
“go straight", “slow down a little bit", “make a right turn", “it’s going to be a narrow street so go straight", “and then you’re going to make a left turn when you see the first", “go straight", “and make a left turn here", “make a left turn", “and go straight", “and do you see the green spot", “park there", “okay", “go straight", “turn left turn here", “and another left turn", “and you’re going to make a right turn here", “and make another left turn", “go straight", “just go straight", “and make another left turn", “left turn", “make another right turn right turn", “go straight", “skip this", “and then make a left turn here left turn", “left turn", “left", “and park there", “wait for me", “can you go back", “reverse", “and then left turn", “go little more little more", “go back back", “back it out a little more", “good job", “okay go straight", “to your left side to your left side", “go straight", “keep going go straight", “pass the street intersection and then go", “go straight", “yeah can you go little faster", “and then make a left turn here", “okay try your best", “make a left turn", “left", “and you’re going to make another right turn right turn here right right", “okay", “go straight just keep going", “pass this", “okay slow down a little bit", “and you going to make a left turn okay", “go straight", “and then make a left turn", “left here and then left", “make a right turn right away", “right here right here", “and then another right", “right slow down slow down", “okay go straight", “and then the green will be on your left side left side", “cool we are done".

Table B.2: Language from two instances of the preliminary two-player driving game.

Model	Language Type	Input Modality					
		rgb		gs		ps	
		train	test	train	test	train	test
$H_\emptyset$: hierarchical baseline	single	1.000	0.958	1.000	1.000	1.000	1.000
	double	0.763	0.437	1.000	1.000	0.893	0.904
	ordinal	0.813	0.490	0.969	0.906	0.938	0.813
	all	0.858	**0.621**	0.996	0.986	0.939	0.923
H_i: $H_\emptyset$ with image	single	1.000	0.958	1.000	1.000	1.000	1.000
	double	0.821	0.411	0.979	0.982	0.945	0.856
	ordinal	0.674	0.344	1.000	1.000	1.000	0.813
	all	0.867	0.586	0.990	**0.991**	0.973	0.898
H_{ih} **(full model)**	single	1.000	1.000	1.000	1.000	0.982	1.000
	double	0.809	0.439	1.000	0.986	0.976	0.874
	ordinal	0.813	0.333	1.000	0.938	1.000	0.938
	all	**0.880**	0.613	**1.000**	0.970	**0.982**	**0.926**

Table B.3: Comparison of three different input modalities: ground-truth segmentation `gs`, predicted segmentation `ps`, and raw color images `rgb`. The highest values from `all` language type are highlighted. Models used in ablation, $H_\emptyset$, H_i and H_{ih}, are described in Section 3.5.2.

Appendix C

Additional Materials for LanguageRefer: Spatial-Language Model for 3D Visual Grounding

C.1 Examples of ReferIt3D Dataset

ReferIt3D (Achlioptas et al., 2020) proposed a 3D visual grounding task with the dataset with referring language annotation on a 3D reconstructed indoor scene dataset, ScanNet (Dai et al., 2017). Following the official splits of ScanNet (Dai et al., 2017), ReferIt3D (Achlioptas et al., 2020) provided natural language annotation of referring one of 76 target classes which is called Nr3D. It also provides template-based referring language on spatial relationship, Sr3D. Based on the datasets, it proposed the 3D visual grounding task. From a scene, ground-truth bounding-boxes, corresponding pointclouds, and the language description of the target object are given. The goal of the task is to choose one bounding-box from candidates.

We visualized three example scenes (scene0011_00, scene0231_00, scene0141_00) with bounding-boxes, utterances, and corresponding target bounding-boxes in Figure C.1-C.3. Subfigures a show all bounding-boxes (red) in the example

scenes. Bounding-boxes of selected target classes are highlighted in blue. For instance, in Figure C.1 (b-c) two bounding-boxes of tables are in blue since we choose table as the target class for visualization. Rest of figures (Figure C.1 (b-c), C.2 (b-e), C.3 (b-d)) show the individual bounding-box of objects (yellow) in the target classes. We add virtual robot paths for the application of robot navigation. Red dots indicate the random starting positions and yellow dots demonstrate paths to reach the target object. Each caption contains an example utterance for the corresponding target object.

C.2 LanguageRefer at Inference

Figure C.4 shows the inference stage of the proposed method. At inference, we followed the approach of InstanceRefer (Yuan et al., 2021) to filter out objects that do not belong to the predicted target class. A target classifier takes the language utterance as input and predicts the target class. Filtering masks are generated by comparing the predicted target class to predicted class labels from the semantic classifier. In order to reduce the chance of removing the true target instance in the filtering process, top-k class predictions (from the semantic classifier) for each object are compared to the predicted target class (not shown in the figure). Filtering masks are applied to the output embeddings of the spatial-language model to refine objects only related to the predicted target class.

C.3 Qualitative Results of LanguageRefer

Figure C.5 shows the qualitative prediction result of LanguageRefer on the first example scene (scene0011_00) with natural language utterances. In Figure C.5 (a) and (b), the model correctly chooses the target object given utterances in the test dataset as well as the custom utterances such as "***a smaller table***." Figure C.5 (c) shows the failure case where the custom utterance "***table without any chairs around***." is given but the model selected another table at the center of the room. Expressions such as without seem to be rare; our model correctly predicts all referred tables from corresponding utterances in the dataset:

"***this is the large conference table with many chairs***", "***the desk directly below the board on the wall***", "***the biggest table in room***", "***the large table in the middle of the room***", "***the thin wooden table underneath the television and immediately to the left of the trash can***", "***smaller table against the wall***", "***select the long table in the middle of the room***", "***choose the table that is up against the wall***", "***the long conference table in the middle of the room***", "***choose the table that sits against the wall***", "***the largest table in the room***", "***select the table underneath the tv***", "***a small table below the television on the wall***", "***the very big dining table in the center of the room***."

In Figure C.6, we asked the model to choose one of the three stacked boxes with natural language utterances. The model failed to select the top box (in yellow) and selected the box in the middle (in red) in Figure C.6 (a). When we replace predicted class labels from PointNet++ (Qi et al., 2017) by the ground-truth class labels, the model was able to choose the correct box on top in C.6 (b). However, the attempts to select the box in the middle failed with or without ground-truth class labels in Figure C.6 (c). Figure C.6 (d) shows the successful reference of the bottom box by the model with predicted class labels. Now the robot paths are visualized with the color of prediction; green if correct, red otherwise.

We examined the predicted class labels of three boxes: microwave, box, box (from top to bottom). This caused an incorrect reference of the top box: 0/7. After fixing the incorrect class label by the ground-truth class label, the accuracy of the reference task of the top box got higher: 6/7. However, the ground-truth class label did not improve the accuracy of the reference task of the box in the middle: 0/6. The reference task of the bottom box was 6/6 before fixing the label. By providing ground-truth labels, we were able to disentangle reference errors from perception errors. From the three-box example, we found the model was not able to refer to the box in the middle of the vertical stack.

In Figure C.7, we evaluated the accuracy of the reference task of kitchen cabinets with natural language utterances. Figure C.7 (a) and (b) show successful examples of reference. Figure C.7 (c) shows a failure case.

Note that the top-5 class predictions of the kitchen cabinets are noisy:

- **top-box**: ['cabinet', 'cabinets', 'kitchen cabinets', 'bathroom cabinet', 'kitchen cabinet'],
- **middle-box**: ['cabinet', 'bathroom cabinet', 'cabinets', 'kitchen cabinet', 'kitchen cabinets'],
- **bottom-box**: ['kitchen cabinets', 'cabinet', 'cabinets', 'kitchen cabinet', 'bathroom cabinet'].

It shows that our model is robust with subtle changes in class labels. Even without a unified format of similar classes including plurals, our model was able to accurately refer to the correct objects. We do not preprocess utterances except for tokenization; preprocessing of language expressions or transforming the utterance into a fixed form is not used (Feng et al., 2021).

C.4 Orientation Annotation for View-Dependent Utterances

In addition to the proposed model, we have collected orientations of the view-dependent utterances. View-dependent utterances without information about the original viewpoint make the reference task challenging. For instance, utterances such as "The door is wood with the handle on the left side." assume specific orientations of the agent and it is impossible to recover the true orientation without knowing the referred object, not like view-dependent utterances with explicit view-point information such as "Facing the foot of the bed." However, the original dataset of ReferIt3D (Achlioptas et al., 2020) does not distinguish the utterances without orientation information from those with orientation information. Therefore, we split the view-dependent (VD) utterance category into two subcategories, ***VD-explicit*** and ***VD-implicit***, where VD-explicit has explicit view-point information in the utterance. Then we collected orientations that make the utterances valid from human annotators.

We set four standard orientations assuming the agent is in the room (around the center of the scene) and ask the annotators to select all of the orientations that can be considered valid from the utterance. Figure 10 shows

examples of four orientations. With the assumption of the agent being inside of the room, we found that four orientations are good enough to recover the original viewpoints of the speakers. In total, 12,680 view-dependent utterances of the Nr3D dataset were annotated. From those, 5,942 utterances are classified as VD-explicit. For train and test split, 10,206 and 2,474 utterances were annotated.

We also provide a link to the orientation annotation webpage (Orientation Annotation Webpage). It is recommended to use a computer or a laptop to view the page. The first page shows the links to all the scenes for annotation. If you click one scene, you can see the list of utterances with some flags. By clicking a single utterance, all the bounding-boxes with highlighted target bounding-boxes are rendered. A green bounding-box is the ground-truth target and the red bounding-boxes are distractors that belong to the target class but not the target object instance. Then press any number in {1, 2, 3, 4} on the keyboard. You can see the scene with the canonical orientation of your selection. You can zoom in/out, translate, rotate with your mouse. We collect annotations on utterances only with correct guesses from humans and mention the target class. Figure C.8 shows examples of four standard orientations and Figure C.9 shows the annotation interface.

Note that, in the process of ReferIt3D (Achlioptas et al., 2020) annotation, the ground-truth target class and the distinguished bounding-boxes of the target class are provided to the annotators. We also provide those information to the annotators. However, in the actual task of ReferIt3D (Achlioptas et al., 2020), the model does not have access to the target class and many other bounding-boxes from other classes are given as you can see in Figure C.1 (a). If some utterances are assuming the shared view or orientation of speakers, the ambiguity can be easily resolved by human listeners since they have extra information and they can manipulate orientation as well. However, the same assumption can make the reference task even more challenging because the model needs to verify some hypothetical orientations with uncertainty of classes among multiple candidates.

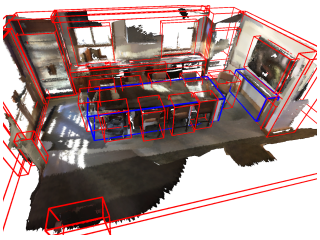
C.5 Ablation of Positional Encoding and DistilBert

In order to examine the effect of the spatial encoding (sinusoidal positional encoding function) (Vaswani et al., 2017) and the base language model (DistilBert (Sanh et al., 2019)), we have trained ablations models.

In the first ablation model, we replaced the sinusoidal positional encoding function with a linear layer to transform a 6-dimensional input vector of bounding-box information to a 768-dimensional embedding vector. In the second ablation model, we keep the spatial encoding to the positional encoding function but replaced DistilBert (Sanh et al., 2019) with BERT (Devlin et al., 2018).

The first ablation model of the positional encoding achieved 37.8 % accuracy on Nr3D while our final model achieved 43.9 % on Nr3D. It shows that selection of an effective spatial encoding scheme such as the sinusoidal is important.

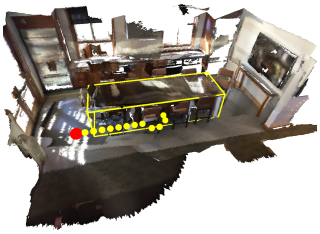
The second ablation model with BERT as the base language model achieved 45.3 % accuracy on Nr3D which is slightly higher than the accuracy of the model with DistilBert (Sanh et al., 2019) (43.9 %). However, when it was trained on Sr3D, it only achieved 49.3 % while the original model achieved 56.0 %. We observed instability in training with the BERT (Devlin et al., 2018) model. We also observed that the successfully trained model on Nr3D converged faster than the DistilBert-based model. During the training, the total loss of the model on Sr3D surged in the middle and did not recover. While these are some observations during our ablation, the second study with BERT (Devlin et al., 2018) model is inconclusive. We chose DistilBert-based model in our framework is because it is lightweight and easy to train. Our goal is to develop a modular approach that can be easily modified based on the advancements in the area of learning embeddings, especially in NLP and computer vision.



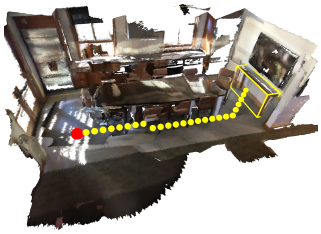
(a) Two bounding-boxes of tables are shown in blue while the bounding-boxes of other classes in red.

(b) A Yellow bounding-box shows the ground-truth bounding-box with the utterance of "***this is the large conference table with many chairs.***."

(c) A Yellow bounding-box shows the ground-truth bounding-box with the utterance of "***the desk directly below the board on the wall***."

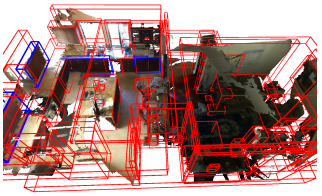


(d) A virtual robot path from a random position in the scene to the target object from (b) is shown in yellow dots.

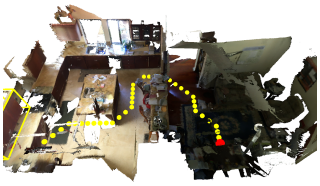


(e) A virtual robot path from a random position in the scene to the target object from (c) is shown in yellow dots.

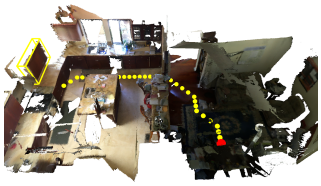
Figure C.1: Examples of ReferIt3D (Achlioptas et al., 2020) dataset. Figure (a) shows all the bounding-boxes (red) in an example scene with two highlighted bounding-boxes (blue) of the target class. Figure (b-c) show two utterance examples and corresponding target object bounding-boxes. Figure (d-e) visualize virtual robot paths generated by A* from a random position to two target objects. Yellow dots indicate the path and red dots indicate the (random) initial positions.



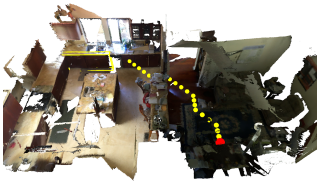
(a) Four bounding-boxes of kitchen cabinets are shown in blue and the other bounding-boxes are shown in red.



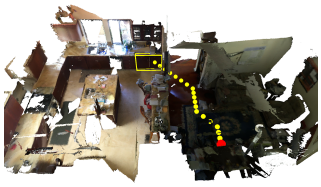
(b) A Yellow bounding-box shows one of the bounding-box of kitchen cabinets with the utterance of "***Kitchen cabinet to the left of the stove***."



(c) A Yellow bounding-box shows one of the bounding-box of kitchen cabinets with the utterance of "***This upper cabinet is between the stove and sinks***."

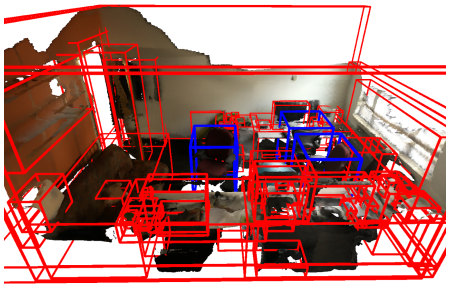


(d) A Yellow bounding-box shows one of the bounding-box of kitchen cabinets with the utterance of "***The cabinets under the sink***."



(e) A Yellow bounding-box shows one of the bounding-box of kitchen cabinets with the utterance of "***Of the two bottom cabinets, choose the one on the right***."

Figure C.2: Examples of ReferIt3D (Achlioptas et al., 2020) dataset. Figure (a) shows all the bounding-boxes (red) in an example scene with four highlighted bounding-boxes (blue) of kitchen cabinets. Figure (b-e) show four utterance examples, corresponding target object bounding-boxes, and robot paths to reach the objects. Yellow dots indicate the path and red dots indicate the (random) initial positions.



(a) Three bounding-boxes of chairs are shown in blue and the other bounding-boxes are shown in red.

(b) A Yellow bounding-box shows one of the bounding-box of chairs with the utterance of "***Chair closest to the door***."

(c) A Yellow bounding-box shows one of the bounding-box of chairs with the utterance of "***select the chair pushed in at the desk***."

(d) A Yellow bounding-box shows one of the bounding-box of chairs with the utterance of "***The chair is the red one closest to the window facing towards the blue ball and away from the desk***."

Figure C.3: Examples of ReferIt3D (Achlioptas et al., 2020) dataset. Figure (a) shows all the bounding-boxes (red) in an example scene with three highlighted bounding-boxes (blue) of chairs. Figure (b-d) show three utterance examples, corresponding target object bounding-boxes, and robot paths to reach the objects. Yellow dots indicate the path and red dots indicate the (random) initial positions.

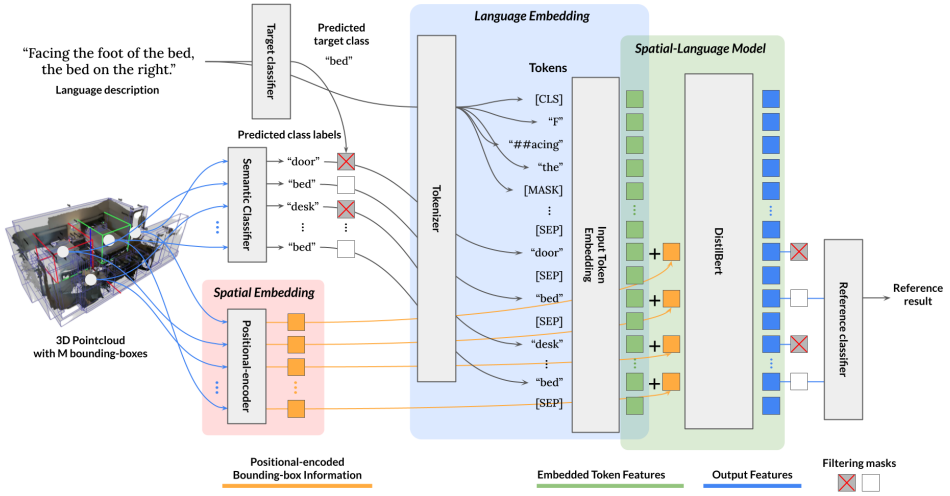


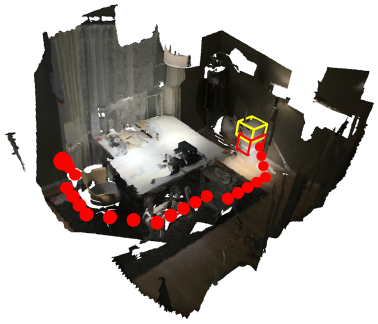
Figure C.4: Detailed overview of LanguageRefer at the inference stage. At inference, an extra target classifier is employed to exclude objects that are not related to the target class in the reference task. Please refer to the details in the section C.2.

(a) LanguageRefer successfully selects the referred table at the center with a custom utterance "***table with lots of chairs***." as well as the utterance from the dataset "***this is the large conference table with many chairs***."

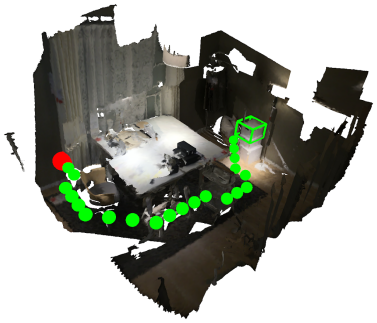
(b) LanguageRefer chooses the correct table on the side with a custom utterance "***a smaller table***." as well as the utterance from the dataset "***the desk directly below the board on the wall***."

(c) LanguageRefer chooses the incorrect table at the center (in red) when the desired target object is the smaller table (in yellow) with a custom utterance "***table without any chairs around***."

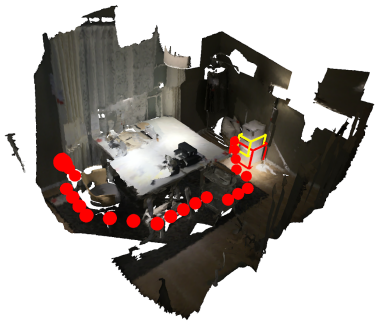
Figure C.5: Qualitative analysis of LanguageRefer result with table as the target class. Figure C.5 (a) and (b) show successful results of LanguageRefer references in an example scene (green) and input utterances. The proposed model predicts correct target objects with custom utterances. Figure C.5 (c) shows an failure case of the proposed method with a custom utterance.



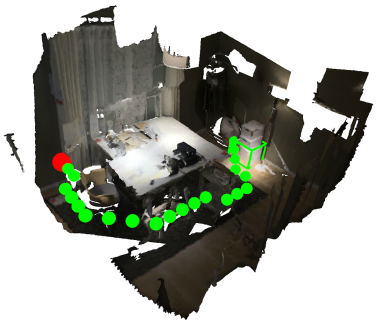
(a) LanguageRefer failed to select the top box from the utterance "***of the three boxes stacked pick the top one***." A yellow bounding-box shows the ground-truth target object and a red bounding-box shows the incorrectly selected target object.



(b) When we provide the ground-truth class labels to the model, it chooses the correct box from the same utterance as in Figure C.6 (a).

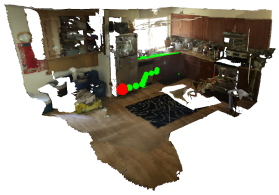


(c) LanguageRefer chooses the incorrect box at the bottom (in red) with the utterance "***middle box in the stack***." indicating the ground-truth box in the middle (in yellow).

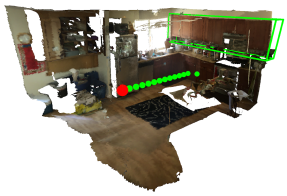


(d) LanguageRefer chooses the correct box at the bottom (in green) with utterances "***the bottom box in the stack of three boxes***." and "***the large white box on the bottom***."

Figure C.6: Qualitative analysis of LanguageRefer result and effect of predicted class labels. Figure C.6 (a) shows a failure case of selecting the top box from the stack of boxes and Figure C.6 (b) shows the corrected reference when the ground-truth class labels are provided. Figure C.6 (c) and (d) shows a failure and a successful case of different boxes, respectively. Ground-truth boxes are shown in yellow, correct and incorrect guesses are shown in green and red, respectively.



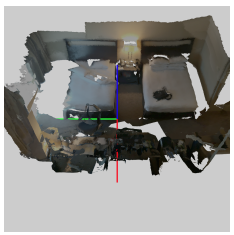
(a) LanguageRefer successfully selects the referred kitchen cabinets with the utterance *"**a double sink is on top of this section of low cabinets**."*



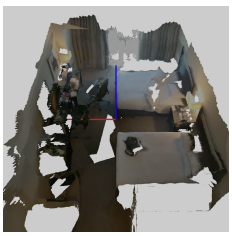
(b) LanguageRefer chooses the correct kitchen cabinets with the utterance *"**higher section of kitchen cabinets**."*

(c) LanguageRefer chooses the incorrect kitchen cabinet on the left side (in red) when the desired target cabinets are in the middle (in yellow) with the utterance *"**lower cabinets next to stove**."*

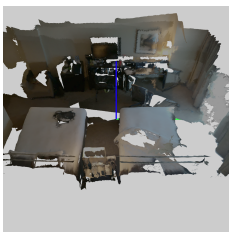
Figure C.7: Qualitative analysis of LanguageRefer result with kitchen cabinets as the target class. Figure C.7 (a) and (b) show successful results of LanguageRefer references in an example scene (green) and input utterances. The proposed model predicts correct target objects with custom utterances. Figure C.7 (c) shows an failure case of the proposed method.



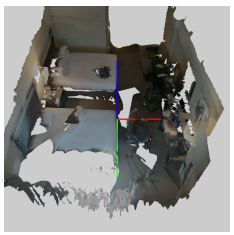
(a) An example of the standard orientation 1.



(b) An example of the standard orientation 2.



(c) An example of the standard orientation 3.



(d) An example of the standard orientation 4.

Figure C.8: Examples of standard orientations for view-point annotation (a-d). We assume that the robot is always inside of the room except for the cases specified by utterances.

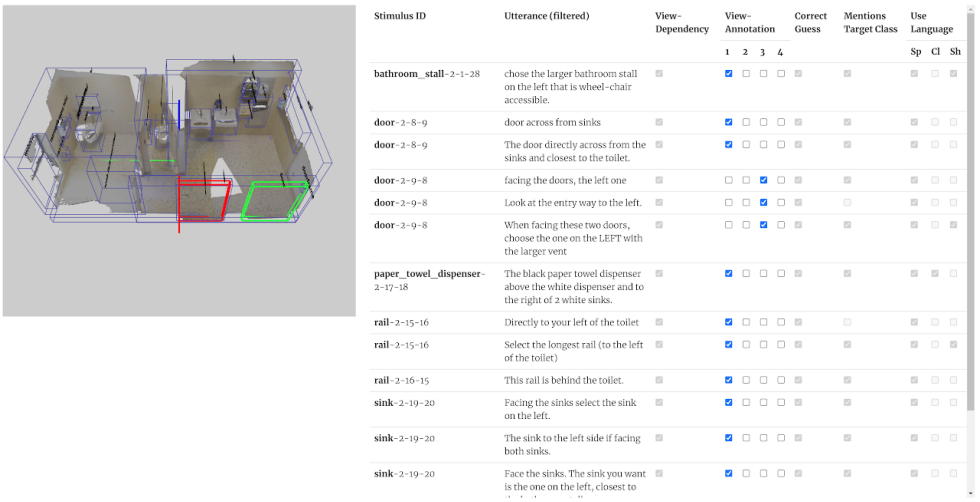


Figure C.9: Interface of the orientation annotation. On the left side, a 3D visualization of the scene with overlaid bounding-boxes and class labels is provided. On the right side, a table of utterance information with orientation annotation checkboxes is shown. By clicking each row in the table, the highlights of the bounding-boxes in 3D visualization is changing with respect to the clicked utterance. A green bounding-box is the true target object while red bounding-box(es) is the distractor object in the same class. Flags 'Correct Guess' and 'Mentions Target Class' indicate whether utterances are considered valid (according to the official ReferIt3D (Achlioptas et al., 2020) evaluation). Flags 'View-Dependency' and 'Use Language' provide information about the utterances. 'Sp', 'Cl', 'Sh' indicate whether the utterance is using spatial relationship, color, and shape, respectively. For instance, the utterance "The black paper towel dispenser above the white dispenser and to the right of 2 white sinks." uses color ("black" and "white") and spatial relationship ("above", "to the right of").