Title

Integrative Approaches to Behavior Prediction, Generation, and Skill Learning in Autonomous Systems

Author

Sun, Lingfeng

Publication Date

2024

Peer reviewed|Thesis/dissertation

Integrative Approaches to Behavior Prediction, Generation, and Skill Learning in Autonomous Systems

By

Lingfeng Sun

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Mechanical Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Masayoshi Tomizuka, Chair
Professor Roberto Horowitz
Professor Negar Mehr
Professor Jiantao Jiao

Spring 2024

Integrative Approaches to Behavior Prediction, Generation, and Skill Learning in Autonomous Systems

Abstract

Integrative Approaches to Behavior Prediction, Generation, and Skill Learning in Autonomous Systems

by

Lingfeng Sun

Doctor of Philosophy in Engineering - Mechanical Engineering

University of California, Berkeley

Professor Masayoshi Tomizuka, Chair

Analyzing and learning diverse behaviors is pivotal in advancing embodied AI, particularly in the realms of robotics and autonomous driving. This dissertation explores three critical aspects of behavior-related research: prediction, generation, and skill learning.

The research begins by addressing the interactive behavior prediction problem in driving scenarios. It employs probabilistic graphical methods to interpret and model the intention changes of vulnerable road users, providing trajectory predictions in interactive scenarios. It then introduces a learning-based approach that leverages domain-specific knowledge to facilitate joint prediction for vehicle interactions, offering interpretable predictions of multi-modal interactive trajectories.

Subsequently, the focus shifts to modeling and generating interactive behaviors. This includes introducing a generative model for learning conditional trajectory generation in joint interactions from collected datasets, with capabilities for generating critical interactions through controllable parameters in provided road scenarios. Further, the work extends to more generalized and complex indoor scenarios where agents are controlled in distributed settings without communication. Potential games are used to model collaborative behaviors between humans and robots, and online optimizations are used to simulate human-like interactions in challenging scenarios. This framework not only generates diverse interactions but also serves to evaluate navigation algorithms.

The final part of the dissertation explores different methods for learning behavioral skills. This includes a parameter compositional framework that utilizes multi-task reinforcement learning and transfer learning to acquire generalized manipulation skills efficiently. An adaptive energy reward design is then detailed, aiding in natural locomotion behavior learning across various speeds and gaits in quadrupedal robots. Moreover, a generalized framework employing large language models addresses partially observable tasks in robotics, showcasing

the utility of reinforcement and supervised learning across diverse behavioral contexts.

Overall, this dissertation integrates an array of innovative approaches for predicting, generating, and learning behaviors within autonomous systems, advancing the field of embodied AI. These contributions extend the theoretical understanding of complex behavioral dynamics and enhance practical implementations in real-world applications. By introducing robust, scalable, and interpretable models and algorithms, this dissertation aims to increase the adaptability and efficiency of robotic systems across diverse operational environments.

Dedicated to the quiet confidence in one's own journey.

Contents

List of Figures

List of Tables

Acknowledgments

The past six years at Berkeley have been an incredible journey filled with triumphs and trials. I would not have come this far without the tremendous support from many.

I am profoundly grateful to my Ph.D. advisor, Professor Masayoshi Tomizuka. His guidance extends beyond the academic realm, providing support for all different kinds of challenges encountered during my doctoral studies. Professor Tomizuka's humor and optimism have been a great source of comfort in difficult times, and his passion for research and teaching continues to inspire me. These are invaluable lessons I will carry forward in my future endeavors, and I hope I can be a great person and mentor as he is.

I would also like to express my appreciation to Professor Roberto Horowitz, Professor Negar Mehr, and Professor Jiantao Jiao for their invaluable insights as members of my dissertation committee. Special thanks go to Professor Kameshwar Poolla, Professor Somayeh Sojoudi, and Professor Jonathan Shewchuk for their advice during my qualifying exam. I am fortunate to have met such supportive and encouraging faculty at Berkeley.

My internships during my Ph.D. have provided pivotal experiences in my research topics, thanks to my wonderful mentors and collaborators. At Horizon Robotics, working alongside Haichao Zhang and Wei Xu expanded my perspectives on learning and robotics. Collaborations with Zhuo Xu and Wenlong Lu at Everyday Robots (Google X) and Devesh Jha and Diego Romeres at Mitsubishi Electric Research Laboratories (MERL) allowed me to engage with diverse robotic platforms, enriching my research experience.

I am indebted to my collaborators, including Xinghao Zhu, Changhao Wang, Chen Tang, Zhuo Xu, Hengbo Ma, Liting Sun, Xiang Zhang, Pin-Yun Hung, Zhaoheng Yin, Boyuan Liang, Yixiao Wang, Bike Zhang, Zhiqiang Jian, Yaru Niu, and all current and past members of the MSC lab. Your eagerness to engage in discussions has been instrumental to my research. Each project we have embarked on, regardless of its outcome, has been an unforgettable journey.

I must also extend my heartfelt thanks to all my friends in Berkeley and around the world for the countless cherished memories that have sustained my mental health during the tough and challenging times. In particular, Jing Ma, Xia Chen, Tianjun Zhang, Zhizhou Zhang, Yilun Wu, Canran Ji, Shuye Huang, and Ning Ju — your support is my greatest strength.

I am grateful for the PhD journey itself. It has not been a smooth path, but it taught me how to navigate challenges, cope with failures, and discern what must be ignored. As someone who is not naturally inclined to share feelings, I learned to seek encouragement and draw strength from others, reinforcing the importance of self-trust — self-confidence is indeed everything.

Last but not least, my deepest gratitude is reserved for my parents; their unwavering trust and support are beyond measure.

Chapter 1

Introduction

1.1 Understanding Behaviors: From Interpretable Prediction to Interaction Generation

The advent of robotics and autonomous driving has profoundly transformed various sectors, heralding a pivotal shift towards more intelligent and adaptive technologies. These systems' capacity to understand and emulate human behavior, as well as to learn novel behaviors, is fundamental to embodied AI research.

My journey at Berkeley began with a focus on behavior prediction within autonomous driving scenarios, especially interactions among drivers and between drivers and vulnerable road users like pedestrians and cyclists. Predicting these interactive behaviors is crucial for the safe decision-making required in self-driving vehicles. Initially, one of the most significant challenges was data scarcity; interactive data were difficult to collect and label. Although today's availability of large public datasets has improved access to interaction data, the paucity of tail events remains a critical safety-related issue. With limited data coverage, designing methods that make the best use of existing data is essential. The first part of the dissertation addresses this challenge by exploring interpretable methods that incorporate prior knowledge of data generation into prediction processes. Specifically, it discusses two sub-topics in interpretable interactive prediction:*i)* intention prediction in vehicle-pedestrian interaction *ii)* multi-modality in joint prediction. Both areas adhere to the classical prediction formulation of forecasting future behavior based on historical observations, where intention prediction focuses on the causality between high-level decisions and key features, and multi-modality in joint prediction addresses the practical issue of covering as many possible outcomes as feasible to ensure that tail cases are not neglected.

As research progressed, a more fundamental issue emerged: traditional prediction benchmarks often fail to represent complex, infrequent interactions that are crucial for safety and reliability. These metrics might cover most everyday behaviors, yet the rarest events—those that rarely occur but are critical to test—are often overlooked. Thus, the second part of the dissertation shifts focus from prediction to interaction generation. This segment empha-

sizes generating realistic interactions that are underrepresented in existing datasets, which is crucial for both planning and evaluating planners. It explores how a closed-loop behavior generation framework can serve both as a predictive tool and as a reactive evaluation module, testing whether proposed algorithms can appropriately plan actions in interactive scenarios. The discussions extend beyond autonomous driving to include general indoor environments, highlighting the broader applicability of these methods.

1.2 Learning Behaviors: Generalized Skill Learning in Robotics

The exploration of interaction generation naturally leads into the realm of generalized learning, particularly within robotics, where the shift in representations and behaviors poses significant challenges. Unlike prediction tasks, robotic tasks often feature diverse rewards, horizons, and hardware settings, necessitating different input/output representations. The third part of the dissertation introduces various approaches to address these challenges:

1. *Generalized policy:* Development of multi-task reinforcement learning policies aims to find a generalized policy that can efficiently learn tasks with similar hardware and environmental settings but differing rewards.

2. *Generalized reward:* Exploration of generalized energy rewards in locomotion seeks to discover natural locomotion behaviors across multiple speeds.

3. *Generalized planning framework:* Utilization of Large Language Models to develop a planning framework capable of solving open-vocabulary, long-horizon tasks, particularly those that are challenging to describe with dense reward functions.

4. *Generalized representation:* Development of a contrastive learning framework for grasp proposal networks aims to enhance grasp proposal generalizability and quality using limited data.

1.3 Dissertation Outlines and Contributions

This dissertation is the culmination of these efforts, showcasing a suite of innovative methods for predicting, generating, and learning behaviors in autonomous systems. Interpretable predictions of different types of road participants in autonomous driving are introduced in the first part of the dissertation from Chapter 2 to Chapter 3. Going beyond prediction in driving scenarios, critical interaction generation is discussed in Chapter 4, and an extended distributed interaction generation framework for indoor scenarios are introduced in 5. From Chapter 6 to Chapter 9, different approaches to learning behavioral skills in robotics are discussed. Figure 1.1 shows the structure overview of the dissertation. The chapter outline is as follows.

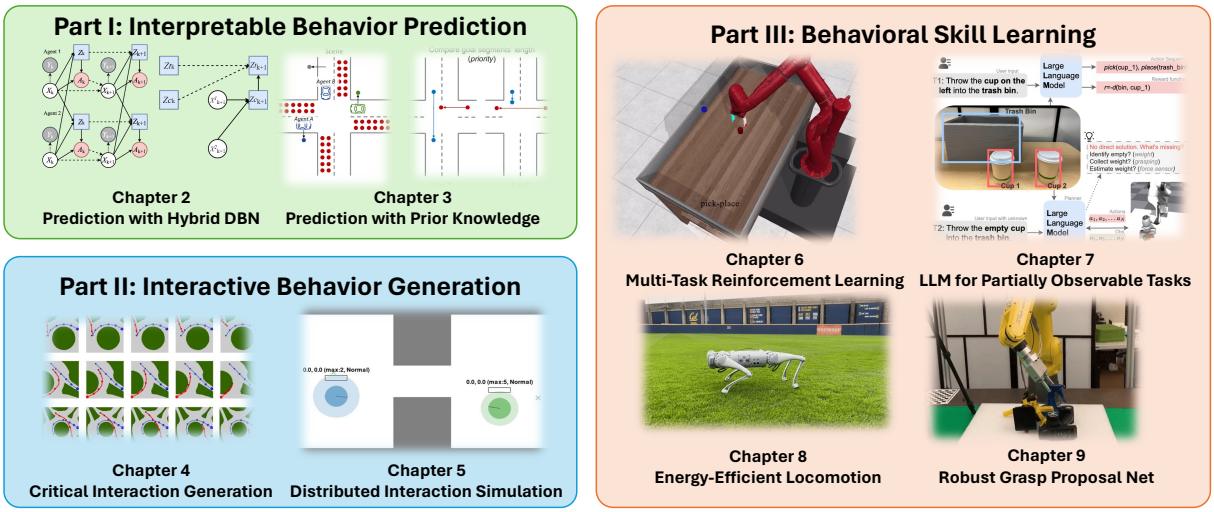


Figure 1.1: Structure of the dissertation

Part I

Prediction for Heterogeneous Agents using Probability Graphical Models

Interactive prediction with multiple traffic participants in highly dynamic scenarios is extremely challenging for autonomous driving, especially when heterogeneous agents such as vehicles and pedestrians are involved. Existing prediction methods encounter problems with interpretability and generalizability to tackle such a complicated task. Chapter 2 constructs an integrated framework to estimate and simultaneously predict the behavior of multiple heterogeneous agents. A Multi-agent Hybrid Dynamic Bayesian Network (MHDBN) method is proposed, which can model the state changes of multiple, heterogeneous agents in various scenarios. We incorporate prior knowledge, such as map information and traffic rules, into the graph structure and use Particle Filter (PF) to track and predict the intentions and trajectories of the agents. Motion data with pedestrian-vehicle interactions from a four-way-stop intersection in the real world is used to design the model and verify the effectiveness of the estimation and interactive prediction capability of the proposed framework. This research was published in [190].

Interpretable Prediction using Domain Knowledge

Motion forecasting in highly interactive scenarios is a challenging problem in autonomous driving. In such scenarios, we need to accurately predict the joint behavior of interacting agents to ensure autonomous vehicles' safe and efficient navigation. Recently, goal-conditioned methods have gained increasing attention due to their performance advantage and ability to capture multimodality in trajectory distribution. Chapter 3 studies the joint trajectory prediction problem with the goal-conditioned framework. In particular, we intro-

duce a conditional-variational-autoencoder-based (CVAE) model to encode different interaction modes into the latent space explicitly. However, we discover that the vanilla model suffers from posterior collapse and cannot induce an informative latent space as desired. We propose a novel approach to address these issues to avoid KL vanishing and induce an interpretable interactive latent space with pseudo labels. The proposed pseudo-labels allow us to incorporate domain knowledge on interaction flexibly. We motivate the proposed method using an illustrative toy example. In addition, we validate our framework on the Waymo Open Motion Dataset with quantitative and qualitative evaluations. This research was published in [186].

Part II

Critical Interaction Generation in Autonomous Driving

Generating diverse and comprehensive interacting agents to evaluate the decision-making modules is essential for the safe and robust planning of autonomous vehicles (AV). Due to efficiency and safety concerns, most researchers choose to train interactive adversary (competitive or weakly competitive) agents in simulators and generate test cases to interact with evaluated AVs. However, most existing methods fail to provide natural and critical interaction behaviors in various traffic scenarios. To tackle this problem, Chapter 4 proposes a styled generative model RouteGAN that generates diverse interactions by controlling the vehicles separately with desired styles. By altering its style coefficients, the model can generate trajectories with different safety levels and serve as an online planner. Experiments show that our model can generate diverse interactions in various scenarios. We evaluate different planners with our model by testing their collision rate in interaction with RouteGAN planners of multiple critical levels. This research was published in [237].

Distributed Human-like Interaction Modeling for Enhanced Human-Robot Collaboration

Interactive behavior modeling of multiple agents is an essential challenge in simulation, especially in scenarios when agents need to avoid collisions and cooperate at the same time. Humans can interact with others without explicit communication and navigate in scenarios when cooperation is required. Chapter 5 aims to model human interactions in this realistic setting, where each agent acts based on its observation and does not communicate with others. We propose a framework based on distributed potential games, where each agent imagines a cooperative game with other agents and solves the game using its estimation of their behavior. We utilize iLQR to solve the games and closed-loop simulate the interactions. We demonstrate the benefits of utilizing distributed imagined games in our framework through various simulation experiments. We show the high success rate, the increased navigation efficiency, and the ability to generate rich and realistic interactions with interpretable parameters. This research was published in [185].

Part III

Multi-task Learning and Transfer Reinforcement Learning

The purpose of multi-task reinforcement learning (MTRL) is to train a single policy that can be applied to a set of different tasks. Sharing parameters allows us to take advantage of the similarities among tasks. However, the gaps between contents and difficulties of different tasks bring us challenges on both which tasks should share the parameters and what parameters should be shared. In this work, we introduce a parameter-compositional approach (PaCo) as an attempt to address these challenges. In Chapter 6, a policy subspace represented by a set of parameters is learned. Policies for all the single tasks lie in this subspace and can be composed by interpolating with the learned set. We demonstrate the state-of-the-art performance on MTRL benchmarks in Meta-World, which contains various manipulation tasks in robotics. In addition, we show our initial attempt at extending PaCo to unseen tasks in a continual setting. The findings of this study were published in [187, 191].

Utilizing LLM for Partial Observable Task Planning

Designing robotic agents to perform open vocabulary tasks has been the long-standing goal in robotics and AI. Large Language Models (LLMs) have recently achieved impressive results in creating robotic agents for performing open vocabulary tasks. However, planning for these tasks in the presence of uncertainties is challenging. It requires chain-of-thought reasoning, aggregating information from the environment, updating state estimates, and generating actions based on the updated state estimates. Chapter 7 presents an interactive planning technique for partially observable tasks using LLMs. In the proposed method, an LLM is used to collect missing information from the environment using a robot and infer the state of the underlying problem from collected observations while guiding the robot to perform the required actions. We also use a fine-tuned Llama 2 model via self-instruct and compare its performance against a pre-trained LLM like GPT-4. Results are demonstrated on several tasks in simulation as well as real-world environments. The findings of this study can be found in [189].

Energy Regularization for Natural Behavior Learning

In reinforcement learning for legged robot locomotion, crafting effective reward strategies is crucial. Pre-defined gait patterns and complex reward systems are widely used to stabilize policy training. Drawing from the natural locomotion behaviors of humans and animals, which adapt their gaits to minimize energy consumption, we propose a simplified, energy-centric reward strategy to foster the development of energy-efficient locomotion across various speeds in quadruped robots. By implementing an adaptive energy reward function and adjusting the weights based on velocity, we demonstrate that our approach enables ANYmal-C and Unitree Go1 robots to autonomously select appropriate gaits—such

as four-beat walking at lower speeds and trotting at higher speeds—resulting in improved energy efficiency and stable velocity tracking compared to previous methods using complex reward designs and prior gait knowledge. The effectiveness of our policy is validated through simulations in the IsaacGym simulation environment and on real robots, demonstrating its potential to facilitate stable and adaptive locomotion. The findings of this study can be found in [113].

Representation Learning for Grasping Skills

While the existing grasp planning algorithm demonstrates proficiency with high-quality observations, its performance is notably compromised under noisy visual observations. Such conditions are frequently encountered due to robot vibration and camera miscalibration. To address this limitation, Chapter 9 introduces a novel contrastive grasp proposal network specifically designed to enhance the robustness of the grasp planner in the face of visual noise. This grasp planer is trained using a synthetic grasp dataset, employing contrastive learning techniques. A key feature of this training process is the deliberate augmentation of each dataset sample to replicate real-world camera noise and extract noise-invariant features. The result is a network capable of reliably identifying 6D collision-free grasps from a single-view depth image. Rigorous experiments with a physical robot validate its effectiveness, underscoring its potential to improve grasp planning reliability in real-world scenarios where visual noise is an unavoidable challenge. This research was published in [254].

Part I

Behavior Prediction

Chapter 2

Prediction for Heterogeneous Agents using Probability Graphical Models

2.1 Introduction

Predicting the future evolution of traffic scenes is essential for autonomous vehicles. It is an essential prerequisite for risk assessment, decision-making, and motion planning modules to enable autonomous driving. Methods employing conventional prediction models can achieve desirable performance in simple scenarios without complex interactions. However, predicting the intentions and future motions of multiple heterogeneous traffic participants, such as vehicles and pedestrians, is still an extremely challenging task.

Fig. 2.1 demonstrates a complicated scenario with a vehicle passing the stop sign and interacting with pedestrians approaching or on the crosswalk. Although vehicles should always yield to pedestrians in such kinds of scenarios according to the right-of-way, impatient human drivers may often accelerate to avoid a long period of waiting. When the attentions of the pedestrians are confirmed, and their distances and velocities are within a safe zone, those drivers may inch forward and finally pass in front of the pedestrians. More pedestrians or even other vehicles or cyclists getting involved in such scenarios makes the interactive prediction problem much more complicated. The changes in the topology and geometry of the scenarios can make the problem even more challenging.

In order to solve such a complicated problem, several challenges have to be overcome when designing a prediction model. First, it should be a highly interpretable model that can easily incorporate prior knowledge such as map information, traffic rules, as well as kinematics and dynamics of different kinds of agents. Also, we need a model which can adjust the number of entities interacting with each other. Moreover, the module designed for one type of agent should be generalizable for either other homogeneous agents or heterogeneous agents. Finally, the model should be transferable to different driving scenarios. Dynamic Bayesian Network (DBN) is a model with interpretability to incorporate prior knowledge with causal relations. It also has great potential to achieve the aforementioned generalizability.

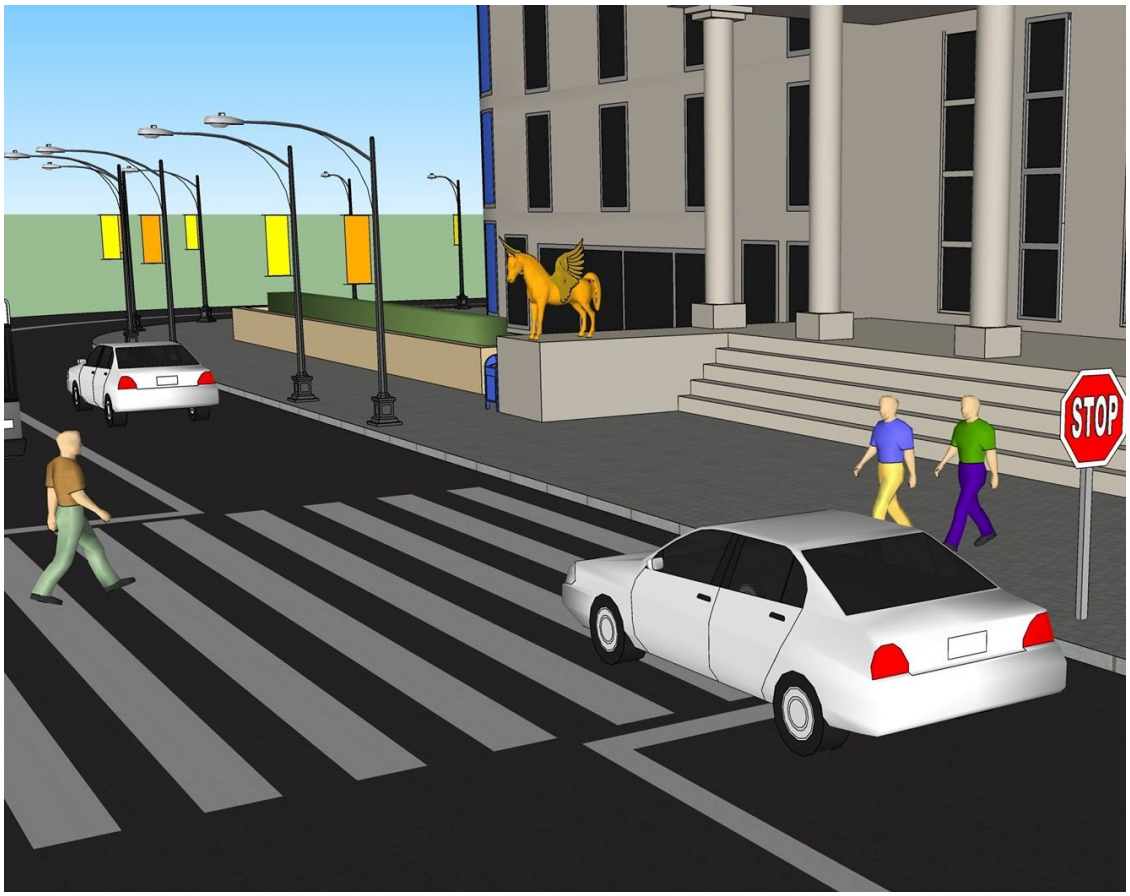


Figure 2.1: A demonstration of interaction between a crossing pedestrian and a vehicle passing the stop sign already.

This chapter proposes an integrated framework for estimation and interactive prediction for multiple heterogeneous traffic participants. We design a Multi-agent Hybrid Dynamic Bayesian Network (MHDBN), which explicitly models the intention and state transition as the traffic scene evolves. The prior knowledge of traffic rules and the interaction between agents are modeled as dependencies and conditional independence in the MHDBN structure. Map information is used to transform the observed positions of agents into a Frenét frame to construct consistent and generalizable features. The Particle filter is used to approximate the complex probabilistic distribution of variables using real-time observations as evidence. The model can estimate the states of multiple, heterogeneous agents at each time step and predict the future intentions and trajectories based on current estimation results.

2.2 Prerequisite

Prediction and Estimation

When predicting the behavior of different agents, both their future motions and intentions are necessary. Considerable amounts of methods have been proposed to tackle interactive prediction problems for pedestrians [159] and vehicles [106][243]. Classification models such as Support Vector Machine (SVM) [215] can recognize discrete intentions but cannot directly predict continuous trajectories. End-to-end deep learning models, such as long short-term memory (LSTM) [11] and convolutional neural network (CNN) [156][49], can provide predictions on both motions and intentions. Although researchers incorporated vehicle dynamics into recurrent models [123] and made the latent space interpretable [76], these black-box models lack interpretability and generalizability in general, and they are typically very data-hungry. Planning-based models such as inverse reinforcement learning [184] is a data-efficient method to predict both intentions and trajectories, and it is highly interpretable to model social interactions [193] and irrational behavior [194]. However, they are sensitive to growing numbers of agents due to the computational complexity of planning algorithms.

Probabilistic Graphical Models (PGM) have the advantage of predicting intentions and trajectories simultaneously for pedestrians [101][70] and vehicles [109] since the intentions can be modeled as states in the structure. They are inherently interpretable due to the causal structure with variables with explicit physical meanings. PGM is generalizable for the prediction of multiple, heterogeneous traffic participants.

The most common interactive prediction problem with heterogeneous agents is pedestrian-vehicle interaction prediction. For pedestrian-only predictions, single-agent PGMs are proposed to track the pedestrian [98], predict discrete intentions [78] and predict intentions and trajectories at the same time [101]. In [101], a context-based Dynamic Bayesian Network (DBN) is used to predict cross intention and path. However, single-agent prediction models can fail when the pedestrians significantly impact the motions of the vehicles. For vehicle-only predictions, multi-agent PGMs are proposed to predict interactive motions [170], but the prediction is still limited to homogeneous agents. Multi-agent model is needed for the interactive prediction of heterogeneous agents.

Datasets

One key challenge in designing interactive models for heterogeneous agents is the data. An overview of publicly available driving datasets is presented in [235]. These datasets may have completely different application focuses. However, datasets focused on vehicles [58] or pedestrians [40][95] usually do not have enough ground truth information for other categories of agents. For datasets focused on the interactions between heterogeneous agents, [169] has limited vehicle reactions, and [153] does not provide information to estimate pedestrian position. We will provide more descriptions of data processing based on existing datasets in Section 2.5.

Hybrid Dynamic Bayesian Networks

Dynamic Bayesian Network (DBN) [138] is a directed graphical model commonly used to model sequential data. The direction in the graph captures the temporal and interpretable causal dependencies between variables.

Exact inference is possible for DBNs where all the hidden variables are discrete. However, in hybrid DBNs where both discrete and continuous hidden variables exist, approximate inference is preferred. For approximate inference in DBNs, deterministic algorithms such as Variational Inference (VI) and stochastic algorithms such as Markov chain Monte Carlo (MCMC) methods [162] are commonly used. MCMC methods can guarantee asymptotic accuracy, but they are computationally expensive. VI methods are faster without an accuracy guarantee. In this chapter, we propose using Sequential Monte Carlo (SMC) methods, also known as Particle filters. Particle filters and their variations, such as Mixture Particle filter [107], are widely used in tracking and prediction problems.

2.3 Methodologies

General Problem Formulation

The traffic scene formulated in this chapter is a generalization of that in [170] by including various kinds of traffic participants with different features. At time step k, each agent $i = 0, 1, 2, ..., N$ under interaction has their continuous state $X_i(k)$, discrete latent state $Z_i(k)$, and action $A_i(k)$. The continuous state describes its behavior features, including kinematic states and gestures. In other words, these are observable features. Variables in the discrete latent state space are defined to facilitate the analysis of its decisions and actions, which are designed for different kinds of agents based on traffic rules. Action describes its movement on the map at a specific time step, which reveals the estimated kinematic transition of each agent under interaction. In addition, an observation $Y_i(k)$ is defined as the noisy measurement of the observable states $X_i(k)$. The trajectories are represented in a Frenét frame [14] with longitudinal and lateral positions with respect to the reference path. More details of the Frenét frame representation will be illustrated in Section 2.5.

There are two main tasks in this problem. The first is to estimate the intention of different agents and justify if the intentions correspond to what people observed in the real world. The second problem is to see if the model can predict the intentions and trajectories well.

Graphical Model

In order to make use of all kinds of features, the interaction described by a multi-agent hybrid Dynamic Bayesian Network (MHDBN) is shown in Figure 2.2. Prior knowledge is included inside the structure in different ways. In detail, map information and traffic rules are used to design latent states of different kinds of agents, while causal dependencies are

used to design conditional dependencies in the graph. Agent dynamic models are used to design the state space of intention and action variables.

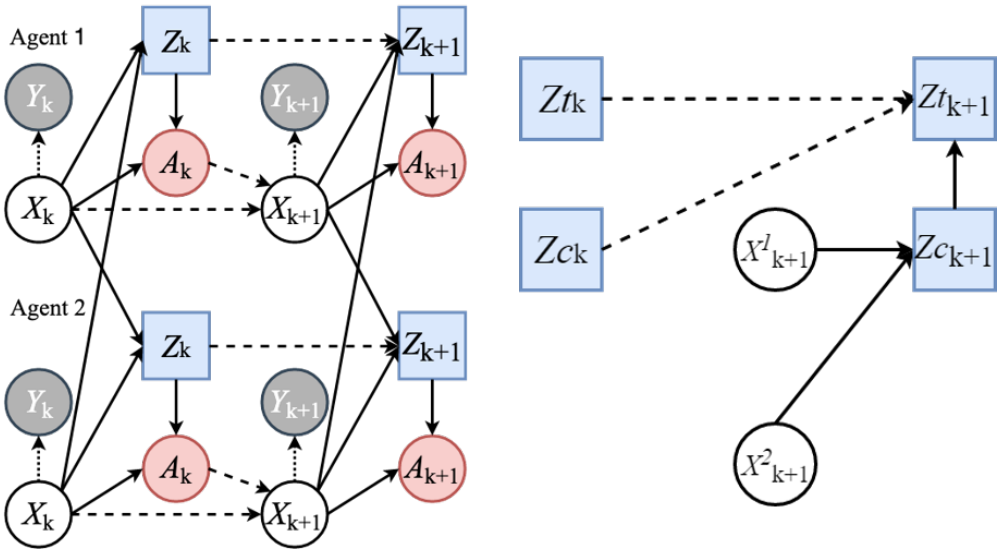


Figure 2.2: The designed multi-agent Hybrid DBN unrolled for two-time slices(left), decomposition of latent state conditional dependency(right). Discrete and continuous nodes are rectangular and circular, respectively. Solid lines, dashed lines, and dotted lines are causal, temporal, and observation dependencies.

Notice the existence of latent discrete variables $Z_i(k)$ designed to include the intentions of each agent. The distribution of $Z_i(k)$ is depended on all the observable states $X_1(k), X_2(k), ..., X_N(k)$ but not on latent states of other agents $Z_j(k)$. This design avoids the acyclic components in the Bayes net.

For each agent i, $A(k)$ is a continuous distribution conditioned on $Z_i(k)$ and $X_i(k)$. Since $Z_i(k)$ is discrete and $X_i(k)$ is continuous, this conditional distribution $P(A(k)|Z(k), X(k))$ can be written as $P_z(A(k)|X(k))$ where z can be any combination of the latent space variables. This works like a switching dynamics model. As a result, $P(X(k+1)|X(k), A(k)) \sim \mathcal{N}(X(k) + A(k), \Sigma_A)$, where Σ_A is the variance matrix of action.

As introduced before, latent discrete variables are defined to help estimate the intentions. One of their use, to serve as indicators for the switching dynamics system, is introduced above. Another important use of these latent variables is incorporating our prior knowledge of traffic interaction inside the Bayes network. The probability of Z_i at time $k+1$, $P(Z_i(k+1)|X_{1:N}(k+1), Z_i(k))$, appears to be a probability of discrete variables conditioned on both continuous and discrete variables. Such a structure is tough to model and infer. Therefore, the assumption that $Z(k+1)$ of each kind of agent can be decomposed as $Z_c(k+1)$ and $Z_t(k+1)$ is made: $Z_c(k+1)$ is the latent variables that have no temporal dependence on $Z(k)$, and $Z_t(k)$ are the latent variables conditioned on $Z(k)$ and $Z_c(k+1)$ as shown in the

right of Figure 2.2. Using conditional independence:

$$P(Z_{c_i,k+1}|X_{1:N,k+1}, Z_{i,k}) = P(Z_{c_i,k+1}|X_{1:N,k+1}) \tag{2.1}$$

This is a conditional probability distribution from continuous variables to discrete ones. This can be modeled as a classifier with probability output.

For time-dependent latent variable at time $k+1$:

$$\begin{aligned} &P(Z_{t_i,k+1}|X_{1:N,k+1}, Z_{i,k}) \\ =&P(Z_{t_i,k+1}|X_{1:N,k+1}, Z_{i,k}, Z_{c_i,k+1}) \cdot P(Z_{c_i,k+1}|X_{1:N,k+1}, Z_{i,k}) \\ =&P(Z_{t_i,k+1}|Z_{i,k}, Z_{c_i,k+1}) \cdot P(Z_{c_i,k+1}|X_{1:N,k+1}) \end{aligned} \tag{2.2}$$

In equation 2.2, the second term is introduced before the first term can be modeled with a conditional probability table. The values in the probability table can either be parameters learned from dataset distribution or values based on prior knowledge.

The observations $Y_i(k)$ provide evidence for the observable states of each agent.

$$P(Y(k)|X(k)) = \Pi_j P(y_j(k)|x_j(k)) \tag{2.3}$$

The Probability of an observation is calculated as the product of all the probability of the observed states, where each state is modeled by a normal distribution $P(y_j(k)|x_j(k)) \sim \mathcal{N}(x_j(k), \sigma^2_{x_j})$. σ_{x_j} is the measurement error coming from the tracking algorithm and measuring devices.

Inference

For a hybrid DBN with mixed continuous and discrete variables, the exact posterior probability for forward filtering of k steps would have to include $|M|^k$ normal distributions for each continuous variable. Therefore, the Particle filter is used to do the approximate inference for estimating Z and X. Given a sample-based representation of N agents with a group of M particles with initial weights of $W(0) = \{w^1, w^2, ...w^M\}$. At time step k, particle j has the form of

$$S_j(k) = [x^j_{1:N}(k), z^j_{c,1:N}(k), z^j_{t,1:N}(k), a^j_{1:N}(k)] \tag{2.4}$$

with weight w^j. The sampling algorithm can be divided into Dynamic update, Causal update, and Observation update. The full process of particle filter is shown in Algorithm 1.

Dynamics update

This update is a one-step prediction of each agent. For each agent i and particle j, $x^j_i(k+1)$ is sampled from $P(X_i(k+1)|X_i(k) = x^j_i(k), A_i(k) = a^j_i(k))$.

Causal update

In this step, causal discrete variable $z^j_{c_i}(k+1)$ is first sampled from $P(Z_{c_i}(k+1)|X_{1:N}(k+1) = x^j_{1:N}(k+1), Z_i(k) = z^j_i(k))$, then $z^j_{ti}(k+1)$ is sampled from the Conditional Probability Table.

Observation update

Observation weight update and re-sampling are the key to the particle filter. For each particle j, their weight is updated by multiplying the probability $P(y_i(k+1)|X_i(k+1) = x^j_i(k+1))$ of each agent. Particles are re-sampled with their current weight distribution when the effective weights are lower than the setting threshold.

Algorithm 1 Particle Filter Update for MHDBN

Input: $S(k), W(k), y_{1:N}(k+1)$
Output: $S(k+1), W(k+1)$
LOOP and Dynamic Update
for $j = 1$ to M **do**
  for $i = 1$ to N **do**
    Sample $x^j_i(k+1)$ from $x^j_i(k), a^j_i(k)$
  end for
end for
LOOP and Causal Update
for $j = 1$ to M **do**
  for $i = 1$ to N **do**
    Sample $z^j_i(k+1)$ from $x^j_{1:N}(k+1), z^j_i(k)$
    Sample $a^j_i(k+1)$ from $x^j_i(k+1), z^j_i(k+1)$
  end for
end for
for $j = 1$ to M **do**
  Update w^j from $y_{1:N}(k+1)$
end for
Normalize weight to get $W(k+1)$
if $M_{effect} < threshold$ **then**
  Re-sample Particles $S(k+1)$ from weight
end if
return $S(k+1), W(k+1)$

With weighted particles as outputs, probabilistic results for both continuous and discrete variables can be calculated.

Prediction

Prediction in hybrid DBN means that the model would develop without observation of all the states. The critical point is to show if the model can capture possible interactions between agents without observing state information. In our particle filter model, since the update of the discrete latent variable states does not require observation, the model should be able to capture the potential conflict between agents.

Each particle has its latent states, and they are sometimes different. Combining the forward simulation of particles in different modes without weight update does not give much information about the correct distribution. Therefore, full confidence is given to the dynamic and causal transition results as observation for next step simulation.

2.4 Pedestrian-Vehicle Interaction Model

As introduced before, the framework proposed can model the interaction behavior of multiple agents with an arbitrary category. Among all traffic participants, pedestrians and vehicles are the most common ones. In this chapter, we focus on the interactions between vehicles and pedestrians in a four-way-stop intersection. The interaction between them usually happens when a pedestrian has the intention to cross the road and the vehicles in other road directions have already passed the stop sign and started accelerating.

State Space Model

For vehicles, the states are defined as $X_v = \{x, y, \theta, v\}$ representing 2D position, heading direction, and velocity, $Z_v = \{Route = [0, 1, 2, ...], Conflict, Avoid\}$ representing the route candidate on map it's following, the judgment on whether a pedestrian is in the conflict area on the road, and the intention of the driver to give way to the pedestrian. Discrete variables are binary if not stated. For pedestrians, the states are defined as $X_p = \{x, y\}$ representing the 2D position, $Z_p = \{Curb, Danger, Cross\}$ representing whether the pedestrian is still on the curb, whether there are vehicle exist in a dangerous area, and whether the pedestrian intends to cross the road. For the separation of latent variables as described in the inference section, it is assumed that $Z_t = \{cross\}$, $Z_c = \{Curb, Danger\}$ for pedestrians, and $Z_t = \{Avoid\}$, $Z_c = \{Conflict, Route\}$ for vehicles.

Action Model

Action models define the dynamic transition over time of different agents. For all agents, $\delta X_t = A(X_t, Z_t)$. For data efficiency and simplicity, basic dynamic models are used to model the switching dynamics since these models are enough to give good filtering and prediction results in the four-way-stop intersection case.

Pedestrians

For pedestrians in the crosswalk case, a linear dynamic model is assumed for pedestrians crossing the road. The average walking speed of pedestrians in this traffic scene is used as a default speed, for pedestrians with a long sequence of motion observed before crossing, average past walking speed is used.

Vehicles

For vehicle dynamics, two simple models are defined for vehicles intended to give way to pedestrians and vehicles passing the intersection. In the first case, the vehicle will decelerate and stop in front of the crosswalk. For the second case, the vehicle will accelerate to the usual driving speed $v_m = 25$km/h on the road after passing the intersection. For both cases, the acceleration is calculated by the constant acceleration assumption.

2.5 Experiments

Dataset and Data Preprocessing

In this chapter, we are using the PedX dataset [95] collected in downtown Ann Arbor, Michigan, USA. The dataset currently consists of data collected from three different four-way-stop intersections. Since the dataset focuses mainly on pedestrians rather than vehicles, we chose one of the three intersections with the heaviest traffic as our target. The PedX dataset collected data with two sets of stereo RGB cameras and four Velodyne HDL-32E LiDAR scanners. both 2D images and 3D point clouds are collected in 6Hz frequency, therefore the time step for analysis is $0.1667s$ for all the following experiments.

Tracking and Smoothing

The PedX dataset doesn't have labels on vehicles. The 2D trajectories of the vehicles are extracted from 3D point clouds using the ground Segmentation methods for vehicles described in [22]. Also, pedestrians and vehicles are frequently occluded by other traffic participants, resulting in missing frames. The optimal filtering method with Kalman Filter and smoothers described in [69] is used to get the trajectory of both pedestrians and vehicles. 2D trajectories are used as ground truth in the experiments, and the observations are assumed to have Gaussian noise. An example frame of data and the tracking result is shown in Figure 2.3.

Frenét Frame for trajectories

Although vehicles and pedestrians in this traffic scene both move in 2D space on the map, they are following limited routes. Unlike pedestrians, who move in a constant direction when crossing, vehicles change their direction when passing the intersection. For vehicles, the two

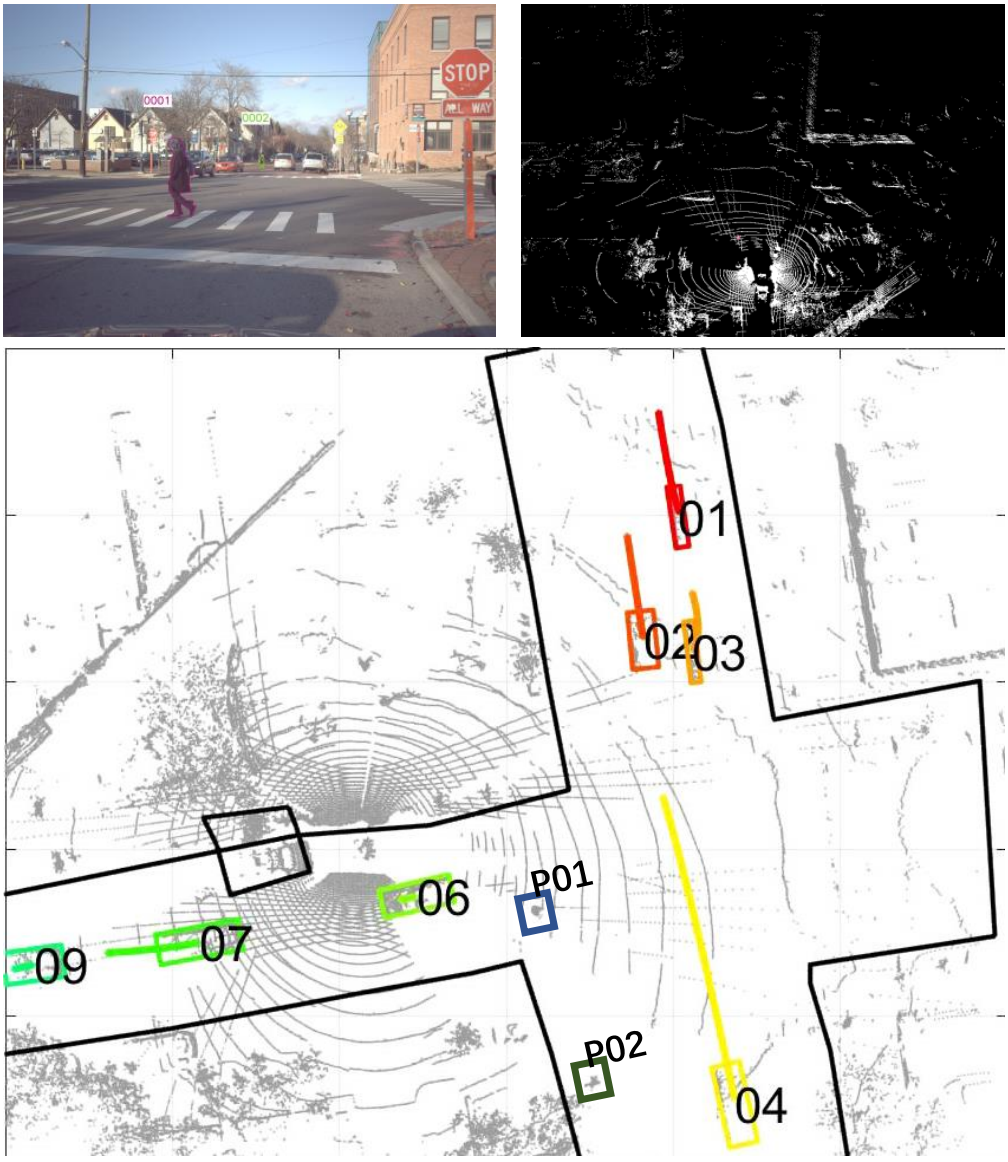


Figure 2.3: Original data format and filtered vehicle and pedestrian trajectories: 2D image from the camera (upper left), 3D point cloud from LiDAR (upper right), tracked and smoothed trajectories of vehicles and pedestrians.

straight lanes they come from and target decide the curve they are following while passing the intersection. With each pair of start point (x_s, y_s) and end point(x_e, y_e), which we extract from past vehicle trajectories, together with the road directions derived from the map, the route can be described with third-order Parametric equations:

$$\begin{cases} x = a_3t^3 + a_2t^2 + a_1t + a_0 \\ y = b_3t^3 + b_2t^2 + b_1t + b_0 \end{cases} \tag{2.5}$$

$t \in [0, 1]$ represents the relative position on the route from start to end. The distance along the curve from starting point s, and the distance to the nearest point on curve d, describes the relative position with respect to the route. Figure 2.4 shows the Frenét frame constructed on this four-way-stop intersection using past trajectories and map information. Curves of the same color in the figure represent routes for vehicles from the same direction based on traffic rules.

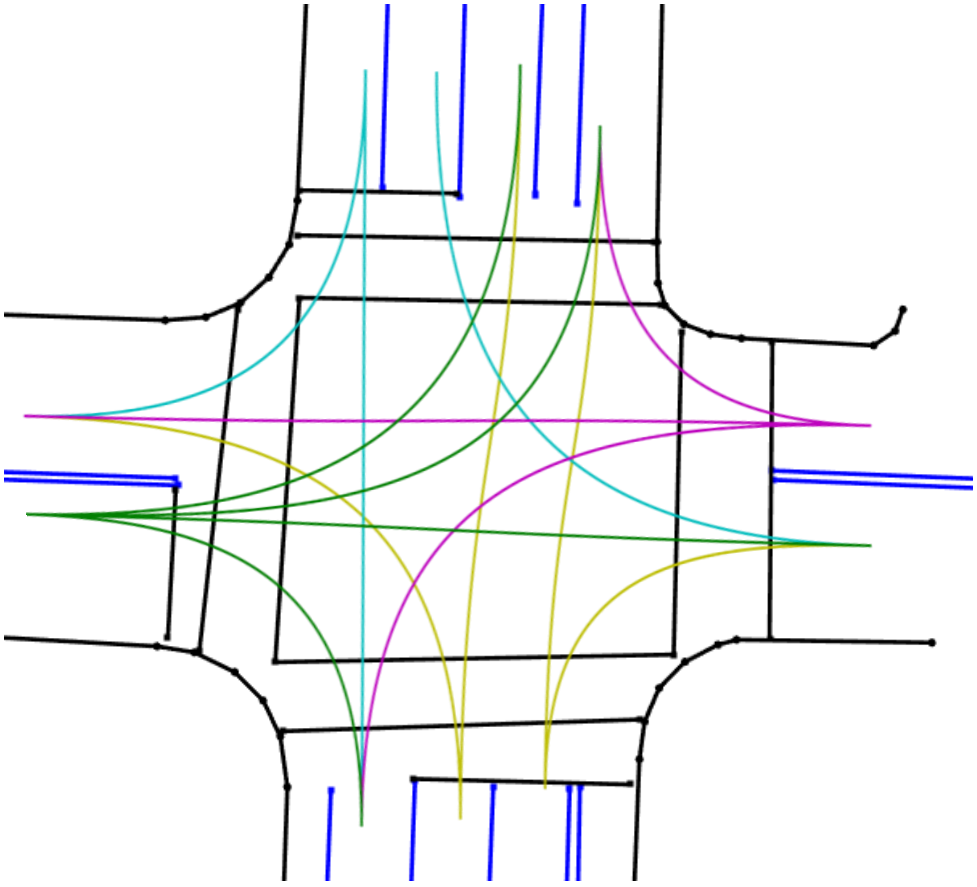


Figure 2.4: The Frenét frame constructed for trajectories in the four-way-stop sign intersection

Intention analysis

The key purpose of intention estimation in this chapter is to give probabilistic results of certain discrete variables in Z that contribute to decisions of road participants. To be specific, most attention is put on *cross* for pedestrian, *route* for vehicles. They are time-depend in most cases and cannot be directly analyzed with single frame information. Therefore, whether the time and trend of these intentions appear to be reasonable will decide if the model captures the intention from the observable states.

In the experiment, the intention trends of a vehicle and pedestrian pair are analyzed when the pedestrian decides to cross the road, and the vehicle in the other road direction has already passed the stop sign. The model is initialized when both participants have entered the zone, and the interaction ends when one leaves.

The result of the estimation in a case where the ground truth is the pedestrian chose to stop and let the passing vehicle go first is shown in Figure 2.5.

To evaluate the position error, the Mean Squared Error (MSE) is used for both tracking and prediction.

$$MSE = \frac{1}{N}\sum_{k=0}^{N}\|X(k) - X_{gt}(k)\|^2 \tag{2.6}$$

In this sequence, tracking errors for pedestrians and vehicles are 0.0625m^2 and 0.4759m^2 for pedestrians and vehicles. Considering the errors from the difference between the Frenét frame trajectories and the real trajectories are systematic errors and cannot be avoided. We can conclude that the assumed switching dynamics model can track the trajectories well.

For this interaction, the human-labeled crossing time step for pedestrian is at $k = 18$. An increase in the probability of crossing appears around 4 steps before the event starts and reaches a high probability to cross at the event time. This means the model captures the interactive reaction of the pedestrian.

Prediction

In prediction, human-defined variables are used to give rational predictions based on the estimation. Short-time trajectory prediction requires the action model to be accurate to describe the motion of the agents, while long-time prediction requires the predicted intention to indicate real behavior. The prediction discussed here is focused more on long-horizon prediction since discrete latent intentions are used to switch the agent dynamic model. For each agent i, set

$$Y_i(k+1) = \mathrm{E}[X_i(k+1)|X_i(k), A_i(k)] \tag{2.7}$$

as observed evidence for step $k+1$. The starting step for prediction also affects the prediction, we use $K = 15$ and $K = 18$ as starting step for prediction in our experiment. The 2D images at the key starting step are shown in Figure 2.6. At $k = 15$, the pedestrian was still waiting and the vehicle was approaching, at $k = 18$ the vehicle just passed the crossing, and the pedestrian started to pass.

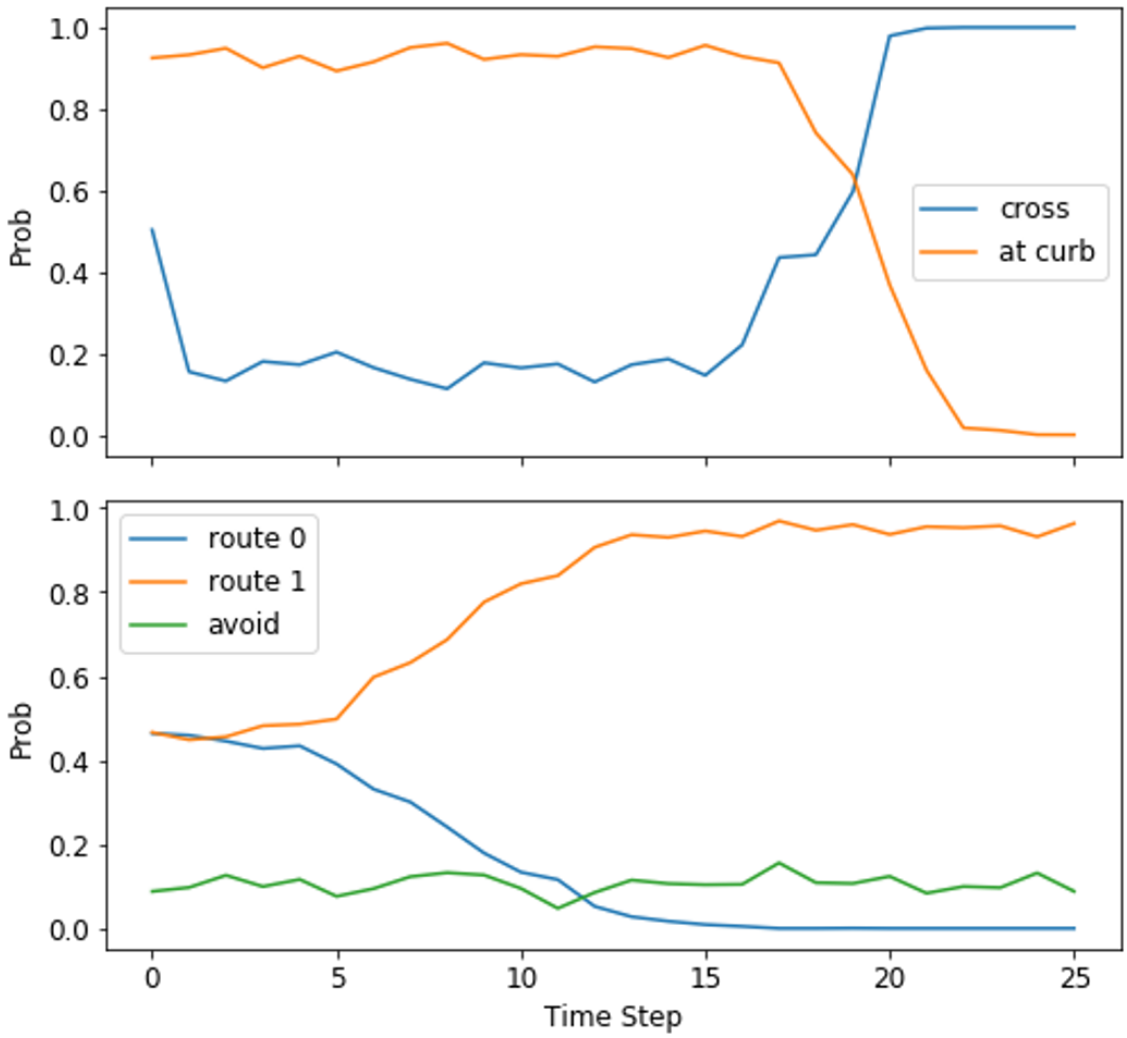


Figure 2.5: The estimation on key intentions of pedestrian(upper) and vehicle(lower) in interaction at the intersection

Figure 2.6: 2D image on starting step of the prediction. The pedestrian and vehicle are labeled with green and purple boxes. The pedestrian was standing on the curb when $k = 15$ (down) and just got started to cross when $k = 18$ (up)

To evaluate the prediction, the intentions predicted are compared to the intention estimated in the previous part. Predicted pedestrian intention change is plotted in Figure 2.7. The predicted behavior starting from $k = 18$ is quite similar to the estimated one, while the predicted behavior starting from $k = 15$ has some delay in the crossing decision. As is shown in Figure 2.6, at $k = 15$ the pedestrian is still standing, while at $k = 18$ it has started to move. The model can predict better after more observation of existing states.

For future trajectory prediction, the predicted trajectory starting from $k = 15$ is shown in Figure 2.8. Delays can be found compared to the ground truth. However, this mostly results from short-term prediction errors of action prediction. The long-term prediction error would significantly improve if better case-defined action models were introduced.

In order to see the effects of taking interaction into account, the prediction error in Table 2.1 compares the prediction error of pedestrian with the baseline Switching Linear Dynamics System (SLDS) introduced in [101] without adding vehicle action into prediction. Notice that The SLDS model cannot predict the behavior if the intention of crossing is very low at predicting start time. Single-agent models cannot give predictions to the vehicle motion.

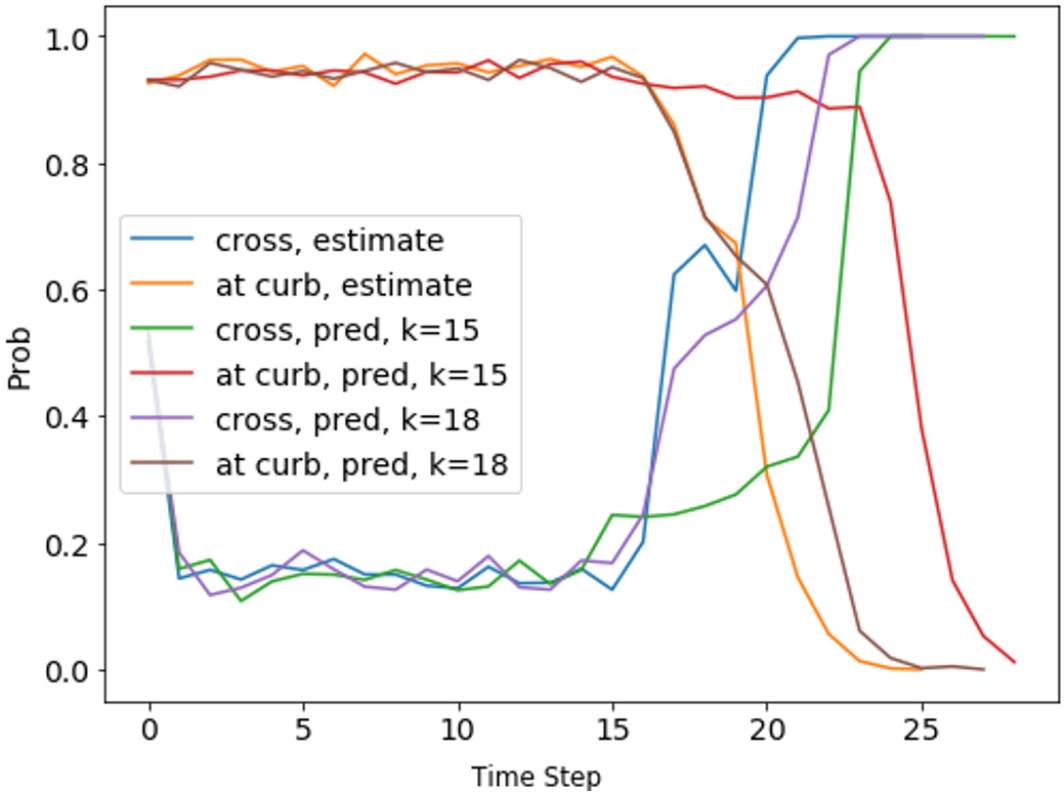


Figure 2.7: Predicted Intention of the pedestrian starting from step $k = 15$ and $k = 18$, compared with estimated intention.

Table 2.1: MSE of prediction

Ours, $k = 18$	Ours, $k = 15$	SLDS, $k = 18$
0.0753	0.3600	0.2203

2.6 Chapter Summary

In this chapter, we proposed an integrated estimation and prediction framework based on a Multi-agent Hybrid Dynamic Bayesian Network (MHDBN) for multiple, heterogeneous agents, where agent interactions, prior knowledge of traffic rules and map information, and switching action models are incorporated into the structure of a hybrid Dynamic Bayesian Network. The proposed framework can estimate and predict the intentions and trajectories of different kinds of agents. The proposed method was designed and tested using a pedestrian-vehicle interaction dataset with real-world motions. With its generalizability, the proposed

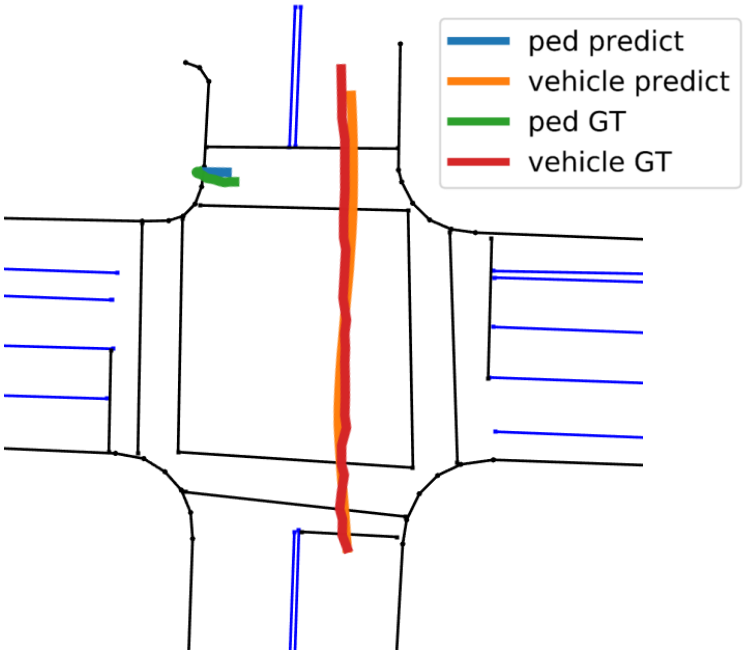


Figure 2.8: Trajectory for pedestrian and vehicle with prediction starting from $k = 15$. The ground truth observed trajectories are also plotted for comparison.

model can be extended to various kinds of agents, such as big buses, cyclists, runners, etc., with corresponding action models and the walking pedestrians and cars in the dataset used. Furthermore, higher-level and context-based features extracted from images such as 2D skeleton poses can facilitate the estimation of gait and intention. Combining the current MHDBN model with more features and advanced sub-modules for intention analysis could help extend the current framework to more complex scenarios.

Chapter 3

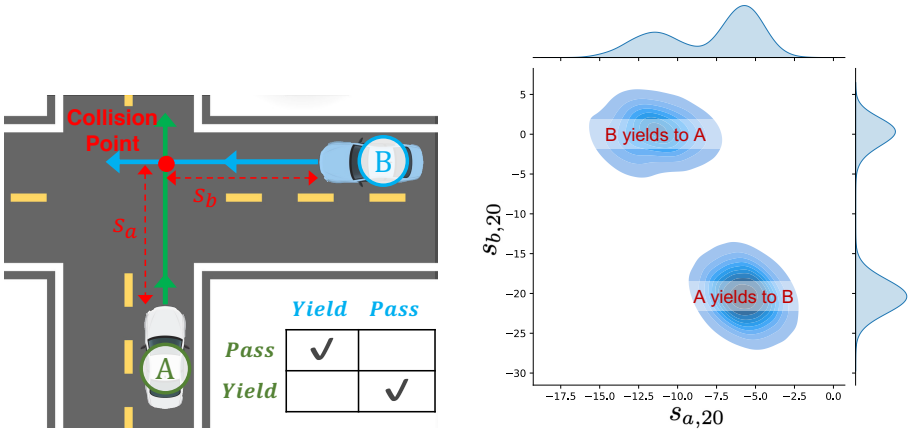
Interpretable Prediction using Domain Knowledge

3.1 Introduction

In the previous chapter, we focused more on vulnerable road users like pedestrians and their intention/trajectory prediction. In this chapter, we focus more on vehicle trajectories, especially in interactive scenarios.

Autonomous vehicles are required to predict other road participants' behaviors to navigate safely and efficiently in complex driving scenarios. Previous prediction benchmarks mainly focus on single-agent settings [244]. When multiple agents exist, each agent's predicted trajectories are evaluated independently. Consequently, predicting the *marginal distribution* of vehicle trajectories suffices for achieving good results on those benchmarks. However, such models may generate unrealistic predictions in highly interactive scenarios. For example, at the intersection illustrated in Fig. 3.1a, each vehicle has two possible motion patterns: entering and yielding before the intersection. A model predicting the marginal distributions might predict infeasible joint behaviors (i.e., both cars follow the same motion pattern). To accurately assess such kinds of interactive behaviors, it is then necessary to predict the *joint distribution* of the interacting agents' future trajectories (Fig. 3.1b). Recently, Waymo provided an interaction prediction benchmark based on the Waymo Open Motion Dataset (WOMD) [44], where the trajectories of two interacting agents are predicted and evaluated jointly. It is an ideal test bed and motivates us to study the interaction prediction problem.

We are particularly interested in the interaction prediction problem under the goal-conditioned framework, as goal-conditioned methods can effectively capture the modalities in trajectory distribution [129, 250, 63]. Under this framework, we first explicitly predict the distribution of an agent's endpoint over a discretized goal set and then complete the trajectories conditioned on the selected goal points. However, previous methods mainly focus on single-agent prediction. For multiple agents, these methods predict the trajectories



(a) (b)

Figure 3.1: A motivating toy example. (a) depicts the scenario of the toy example, where two cars are driving towards a collision point at an intersection. (b) shows the ground-truth joint distribution and marginal distribution of s_a, s_b at the 20^{th} time step when the behavior of the cars is governed by the model described in Sec. 3.3.

independently for each agent. To model the joint distribution of interacting agents' goals, we extend the goal set to a goal-pair set, allowing joint prediction of two agents' endpoints. By choosing a dense set as in [63], this categorical distribution of goal pair can reasonably approximate the joint distribution in any interactive scenarios.

In practice, downstream modules require a small set of representative predictions [33]. The limited onboard computational resource also restricts the number of sampled trajectories. For the downstream module to understand the interactive scenario precisely, ensuring that different interaction modes can be efficiently captured with a limited number of sampled trajectories is critical. To this end, we leverage the Conditional Variational Auto Encoder (CVAE) framework [179, 108, 197] and introduce a discrete latent space to capture the interaction modes explicitly [83, 165]. For instance, in the toy example, we want different latent variables to represent different right-of-ways, corresponding to the two modalities in the joint distribution of goal pairs shown in Fig. 3.1b. However, it is not guaranteed that the model can always learn an informative latent space distinguishing interaction modes useful for downstream modules.

In our goal-conditioned CVAE-based framework, the goal pair follows a categorical distribution. It changes the reconstruction task into a multi-label classification problem. Without knowing the distance between the goal pairs, we find it difficult for the model to distinguish between them. Therefore, it becomes difficult to determine which goal pairs should be encoded into the same latent variable, leading to a posterior collapse in CVAE, resulting in an uninformative latent space.

To tackle this problem, we propose to guide the training with *pseudo labels*[1] designed

[1]For clarification, we refer to any generated labels other than the ground-truth ones as pseudo labels.

based on domain knowledge. For each ground-truth goal pair, we assign positive target values to similar goal pair candidates. The model learns to encode similar goal pairs into the same latent variable by minimizing the distance between the decoded distribution and the pseudo labels. Since the goal pair distribution is defined over a fixed finite set, the pseudo labels can be pre-computed for each goal pair candidate. Therefore, we do not require the computation of pseudo labels to be differentiable. It allows us to incorporate domain knowledge into the pseudo labels flexibly and specify any interaction modes for the latent space to capture.

The main focus of this chapter is: *(1)* To present a goal-conditioned CVAE model for interacting pairs' joint trajectory prediction task. *(2)* To propose a novel and flexible approach to induce an interpretable interactive latent space using pseudo labels. In particular, we introduce three types of pseudo labels corresponding to different domain knowledge on interaction and show that the proposed pseudo labels can effectively enforce an interpretable latent space in an illustrative toy example and on real-world traffic datasets.

3.2 Problem Formulation

Background: Goal-Conditioned Prediction

In general, a trajectory prediction model learns to model the distribution $p(\boldsymbol{y}|\boldsymbol{T})$, where $\boldsymbol{y}$ denotes the future trajectory of the target agent, and $\boldsymbol{T}$ denotes the embedding of the agent's history and context information. In a goal-conditioned trajectory prediction framework, the prediction task consists of two stages: goal prediction and trajectory completion, resulting in the decomposition of $p(\boldsymbol{y}|\boldsymbol{T})$:

$$p(\boldsymbol{y}|\boldsymbol{T}) = \int_{g \in \mathcal{G}} p(\boldsymbol{y}|g, \boldsymbol{T}) \cdot p(g|\boldsymbol{T}) dg,$$

where $\mathcal{G}$ is the goal space. The goal-prediction model $p(\boldsymbol{g}|\boldsymbol{T})$ can capture the multi-modality in driver intention, while the goal-conditioned trajectory completion module models the driving behavior to reach the goals.

The overall framework has three stages. The first stage is *goal distribution prediction*. Depending on the goal space, $p(\boldsymbol{g}|\boldsymbol{T})$ can be modeled as either a continuous or discrete distribution. We are particularly interested in the formulation of [63], in which $\mathcal{G}$ is defined as a dense and discretized goal set covering the drivable area. In such a way, $p(\boldsymbol{g}|\boldsymbol{T})$ directly models the distribution of goal points instead of anchor points as in [250]. The second stage is *goal-conditioned trajectory prediction*, where the conditional distribution of future motions is modeled as a simple unimodal distribution (e.g., Gaussian distribution). The third stage is *sampling and selecting*, where a final small number of predictions are selected to fulfill the requirement of downstream applications. The commonly used techniques are heuristic-based algorithms, such as non-maximum suppression (NMS) [250].

They are not necessarily generated for semi-supervised learning or self-supervised learning.

Goal-conditioned Interactive Prediction

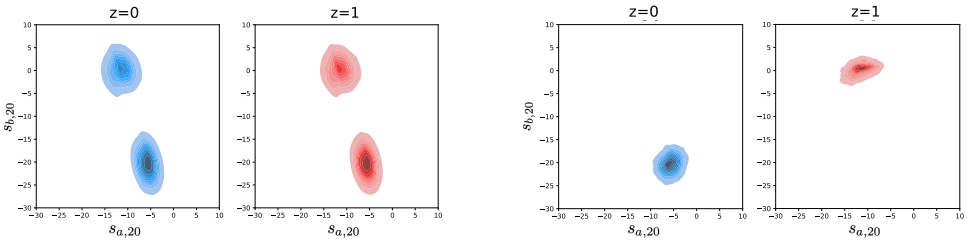
The framework described in Sec. 3.2 is primarily designed for single-agent prediction. The extension of this two-stage prediction scheme to multi-agent settings is not straightforward. In multi-agent trajectory prediction, we need to model the joint distribution of all agents' future trajectories, i.e., $p\left(\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_N | \boldsymbol{T}\right)$. We can decompose the interacting agents at the trajectory completion stage by adopting the assumption that the trajectories are independent after conditioning on the goals. However, we still need to model the joint distribution of their goals, i.e., $p(\boldsymbol{g}_1, \boldsymbol{g}_2, \cdots, \boldsymbol{g}_N | \boldsymbol{T})$. We cannot simply assume the trajectories of interacting agents are independent and decompose the joint distribution into $\prod_{i=1}^{N} p(\boldsymbol{g}_i | \boldsymbol{T})$. The simplified distribution cannot model the interaction between agents, for instance, the fundamental interaction rule —collision avoidance. Meanwhile, if we directly model the joint distribution, we need to select a discrete goal set $\mathcal{G}_i$ for each modeled agent i. The overall dimension of the joint distribution becomes $\prod_{i=1}^{N} |\mathcal{G}_i|$, which grows exponentially with the number of agents.

To mitigate the curse of dimensionality, we first predict the marginal distributions of the goals. Afterward, we use the marginal distributions to prune the goal sets $\{\mathcal{G}_i\}_{i=1}^{N}$. Concretely, we select M goal candidates with the highest marginal probability for each agent. In our experiments, we find that we can reasonably approximate the marginal distribution with $M << |\mathcal{G}_i|$. It is then sufficient to model the distribution of $|M|^N$ goal combinations, which is applicable for the prediction task of interacting pairs.

3.3 Inducing Interpretable Interactive Latent Space with Pseudo Labels

In this section, we take the scenario illustrated in Fig. 3.1a as a running example to introduce the proposed pseudo labels. Specifically, we explain the motivation and demonstrate how the pseudo labels may help induce an interpretable interactive latent space in this toy example.

As shown in Fig. 3.1a, Vehicle A and B are driving towards a collision point. The states of the vehicles are s_a, s_b and v_a, v_b, where $s_{a,b}$ are the displacements of the vehicles A, B relative to the collision point and $v_{a,b}$ are the absolute velocities. Each vehicle is assigned a target position to follow at each step, depending on which vehicle has the "right-of-way". If a vehicle has the right-of-way, we assign a point that is substantially far away along its driving direction as its target point. If the other vehicle has the right-of-way and has not passed the collision point, we assign the collision point as the target point. We assume that the right-of-way is affected by the difference of time headway at the initial time since the car with a shorter headway time to the collision point is more likely to get the right-of-way in interaction. The time headway at timestep t is defined as $T_{\text{head},t} = \max\left(\frac{s_t}{v_t}, 0\right)$. The



(a) Vanilla CVAE Model

(b) CVAE Model with Pseudo Distance Labels

Figure 3.2: Joint goal distributions decoded from different latent variables with different models. With the pseudo distance labels, the model is able to capture the two modes in its latent space. The results are the same when using other pseudo labels.

probability of Vehicle A getting the right-of-way is set as:

$$p_A = 0.5 \left(\tanh \frac{T_{a,\text{head},0} - T_{b,\text{head},0}}{\eta} + 1 \right),$$

where η controls the rate of transition between entering into the intersection and yielding. The dynamics of the vehicles is governed by the intelligent driver model [208]. Each vehicle follows the target position set according to its right-of-way.

The task is to jointly predict the endpoints $\boldsymbol{g} = (\boldsymbol{g}_a, \boldsymbol{g}_b) = (s_{a,20}, s_{b,20})$ of both vehicles after 20 timesteps, given the initial condition $\boldsymbol{T} = (\boldsymbol{T}_a, \boldsymbol{T}_b) = ([s_{a,0}, v_{a,0}], [s_{b,0}, v_{b,0}])$. It is analogous to the goal prediction stage in the goal-conditioned prediction framework. The joint goal distribution is defined over a discrete set $\mathcal{G}_{a,b}$ obtained by discretizing the spaces of $s_{a,20}$ and $s_{b,20}$. Given the same initial conditions, there are two interaction modes, i.e., Vehicle A yields to Vehicle B and vice versa. These two interaction modes result in a multi-modal joint goal distribution as shown in Fig. 3.1b.

To model the joint goal distribution, we leverage the CVAE framework with a discrete latent space. The CVAE model consists of three modules: 1) An encoder $q_\theta(\boldsymbol{z}|\boldsymbol{T}, \boldsymbol{g})$ approximating the posterior distribution of $\boldsymbol{z}$; 2) A conditional prior $p_\phi(\boldsymbol{z}|\boldsymbol{T})$; 3) A decoder $p_\psi(\boldsymbol{g}|\boldsymbol{T}, \boldsymbol{z})$ modeling the conditional joint goal distribution. We use MLPs for all the modules. The model is trained by maximizing the evidence lower bound (ELBO):

$$\begin{aligned} \mathcal{L}(\theta, \phi, \psi) = & -\mathbb{E}_{\boldsymbol{T},\boldsymbol{g},\boldsymbol{y}\sim\mathcal{D}} \Big\{ \mathbb{E}_{\boldsymbol{z}\sim q_\theta(\boldsymbol{z}|\boldsymbol{T},\boldsymbol{g})} \left[f\left(\boldsymbol{y}, p_\psi\left(\cdot|\boldsymbol{T}, \boldsymbol{z}\right)\right)\right] \\ & - \beta D_{KL}\left[q_\theta(\boldsymbol{z}|\boldsymbol{T}, \boldsymbol{g}) \| p_\phi(\boldsymbol{z}|\boldsymbol{T})\right] \Big\}, \end{aligned} \tag{3.1}$$

where $\mathcal{D}$ is the dataset consisting of initial states $\boldsymbol{T}$, goal pairs $\boldsymbol{g}$, and ground-truth labels $\boldsymbol{y}$. The vector $\boldsymbol{y} \in \{0, 1\}^{|\mathcal{G}_{a,b}|}$ collects ground-truth scores of the goal pairs in $\mathcal{G}_{a,b}$. We assign one to the ground-truth goal pair and zero to the others. We choose the Binary Cross-Entropy (BCE) loss as the function f to define the reconstruction loss.

Avoiding KL Vanishing with Pseudo Labels

Our experiments with the CVAE model formulated above show that the KL divergence tends to vanish, and the conditional prior distribution always concentrates on a single value. As shown in Fig. 3.2a, the latent space is completely uninformative. While the decoder can still model the joint distribution, the model does not fulfill our objective to capture interaction modes with the latent space explicitly. This phenomenon is similar to the posterior collapse problem that occurs when an autoregressive decoder is used in sequence modeling [53]. The MLP decoder we use can model the joint distribution without the latent space. With such a powerful decoder, the model is prone to ignoring the latent space to minimize the KL divergence.

We find it difficult for the model to escape from posterior collapse in our case. To gain some insights into the reason behind this, consider a special case where the dataset is collected under the same conditions, and the weight of KL regularization β equals zero. A zero β value occurs during the training procedure when KL annealing [53] is applied to mitigate KL vanishing. We will show that it is still difficult to prevent posterior collapse even if we set $\beta = 0$. In this case, the optimal posterior distribution always assigns all the probability mass to a single latent variable. Consequently, we can consider the VAE as solving a clustering problem. Given a d_z-dimensional discrete latent space, the VAE model essentially clusters $\mathcal{G}_{a,b}$ into d_z subgroups, denoted as $\{S_k\}_{k=1}^{d_z}$, and finds a distribution of goal pairs minimizing the BCE loss for each subgroup. We can easily obtain the minimal value of the BCE loss within a given subgroup analytically. It is then straightforward to see that the optimal clustering scheme essentially minimizes the sum of the objectives over the subgroups, which is the following objective function:

$$\mathcal{L}(\{S_k\}) = \sum_{k=1}^{d_z} \sum_{j \in S_k} (n_j - n_{S_k}) \log(1 - \frac{n_j}{n_{S_k}}) - n_j \log(\frac{n_j}{n_{S_k}}),$$

where we define n_j and n_{S_i} as:

$$n_j = \sum_{i=1}^{|\mathcal{D}|} \mathbf{1}\left(y_j^i = 1\right), \quad n_{S_k} = \sum_{j \in S_k} n_j.$$

In other words, n_j counts how many times the j^{th} goal pair appears in the dataset, and n_{S_k} counts how many times the goal pairs in the subgroup S_k appear in the dataset.

With posterior collapse, the clustering scheme corresponds to having all the elements in a single subgroup while leaving the rest empty. In our toy example, it is easy to check that better solutions do exist, for instance, the one shown in Fig. 3.2. Since the two modes in the joint distribution are separated in the latent space, the goal pairs have a higher likelihood under the decoded distribution conditioned on the latent variable it corresponds to, which leads to a smaller reconstruction error than the trivial solution resulting from posterior collapse. However, it is difficult for the model to escape from the suboptimal solution shown

in Fig. 3.2a. The objective $\mathcal{L}(\{S_k\})$ purely relies on the frequencies of different goal pairs in the dataset. We can interchange goal pairs that appear with similar numbers of times without affecting the objective. It is difficult for the model to learn which two goal pairs should be assigned to the same latent variable to minimize the objective value.

To mitigate this issue, we propose to inform the model of the proximity between goal pairs via pseudo labels. For each goal pair $\boldsymbol{g}_j \in \mathcal{G}_{a,b}$, a pseudo label is a vector defined over $\mathcal{G}_{a,b}$ with values ranging from zero to one, which we denoted as $\hat{\boldsymbol{y}}_j \in [0,1]^{|\mathcal{G}_{a,b}|}$. In $\hat{\boldsymbol{y}}_j$, we assign positive values to the ground-truth goal pair as well as those goal pairs that are "close" to $\boldsymbol{g}_j$ by the distance metric defined by domain knowledge, in contrast to the label from the dataset where the positive value is only assigned to the single ground-truth goal pair. We use the pseudo labels to define the following auxiliary loss function:

$$\alpha \mathbb{E}_{\boldsymbol{T},\boldsymbol{g},\boldsymbol{y}\sim\mathcal{D},\boldsymbol{z}\sim q_\theta(\boldsymbol{z}|\boldsymbol{T},\boldsymbol{g})} \sum_{j=1}^{|\mathcal{G}_{a,b}|} \mathbf{1}(y_j = 1) f\left(\hat{\boldsymbol{y}}_j, p_\phi(\cdot|\boldsymbol{T},\boldsymbol{z})\right),$$

where the function f quantifies the distance between the pseudo labels and the conditional joint goal distribution. By minimizing the auxiliary loss, the model learns to assign high probabilities to both the ground-truth goal pair and those "close" ones specified by the pseudo labels in the distribution conditioned on the same latent variable. Consequently, the model is guided to encode goal pairs that are close to each other into the same latent variable, which prevents the latent space from being totally uninformative. It is worth noting that since the pseudo labels are not required to be generated in a differentiable way, it allows us to flexibly design pseudo labels based on domain knowledge of the proximity between goal pairs. In the next subsection, we will introduce three types of pseudo labels we designed in this chapter.

Pseudo Labels

Pseudo Distance Labels

Since the agents move continuously, their behaviors should be consistent if targeting goal pairs that are close to each other in terms of Euclidean distance. Such goal pairs should then be clustered into the same group. Consequently, we introduce the pseudo distance labels defined as:

$$\hat{\boldsymbol{y}}_{j,i}^{\text{distance}} = \exp\left(-\frac{\|\boldsymbol{g}_j - \boldsymbol{g}_i\|^2}{2\sigma^2}\right), \ i = 1, 2, \cdots, d.$$

It essentially smooths the original singular label with the radial basis (RBF) kernel. We choose f as the BCE loss.

With the auxiliary loss induced by the pseudo distance labels, the CVAE model learns to separate the two interaction modes in the latent space (Fig. 3.2). Also, the prior probabilities of the two latent variables are consistent with the ground-truth probabilities of the corresponding interaction modes in the simulation. The interaction modes can be effectively

separated because the Euclidean distance between goal pairs from different clusters is far away.

Pseudo Marginal Labels

The joint goal distribution is the consequence of the interaction between agents. If Agent A targets the same goal regardless of what goal Agent B follows, we may characterize the interaction by the goal of Agent A. Therefore, we consider goal pairs that share the same goal of one agent closer than those that are totally different. We then define two sets of pseudo-marginal labels:

$$\hat{\boldsymbol{y}}_{j,i}^{\text{marginal},a} = \mathbf{1}\left(\boldsymbol{g}_{j,a} = \boldsymbol{g}_{i,a}\right), \quad \hat{\boldsymbol{y}}_{j,i}^{\text{marginal},b} = \mathbf{1}\left(\boldsymbol{g}_{j,b} = \boldsymbol{g}_{i,b}\right),$$

and the corresponding loss function:

$$\begin{aligned} & f^{\text{marginal}}\left(\hat{\boldsymbol{y}}_j^{\text{marginal},a}, \hat{\boldsymbol{y}}_j^{\text{marginal},b}, p_\phi(\cdot|\boldsymbol{T}, \boldsymbol{z})\right) \\ = & \log\left(\sum_{i=1}^{|\mathcal{G}_{a,b}|} \mathbf{1}\left(\hat{\boldsymbol{y}}_{j,i}^{\text{marginal},a} = 1\right) p_\phi(\boldsymbol{g}_i|\boldsymbol{T}, \boldsymbol{z})\right) \\ + & \log\left(\sum_{i=1}^{|\mathcal{G}_{a,b}|} \mathbf{1}\left(\hat{\boldsymbol{y}}_{j,i}^{\text{marginal},b} = 1\right) p_\phi(\boldsymbol{g}_i|\boldsymbol{T}, \boldsymbol{z})\right). \end{aligned}$$

We essentially maximize the log-likelihood of the ground-truth goal pairs under the marginal goal distributions.

With the pseudo marginal labels, we can guide the CVAE model to perfectly separate the goal pairs into two interaction modes in the toy example. The result is the same as shown in Fig. 3.2. The interaction modes can be perfectly identified because the goal pairs from different clusters happen to have distinct coordinates in both dimensions in our toy example. If only one of the agents changes his behavior in different modes, the pseudo-marginal labels alone will not be helpful.

Pseudo Interaction Labels

The last type of pseudo labels we introduce allows us to incorporate domain knowledge on interaction in a flexible way, which we refer to as pseudo interaction labels. From the perspective of the downstream planner, we may want the latent space to distinguish specific interaction modes for efficient planning and risk evaluation (e.g., collision vs. no collision, yielding vs. passing). If we know that these interaction modes can be identified with certain features, we can design the corresponding pseudo-interaction labels as follows:

$$\hat{\boldsymbol{y}}_{j,i}^{\text{interact}}\left(\boldsymbol{T}\right) = \mathbf{1}\left(h(\boldsymbol{T}, \boldsymbol{g}_i) = h(\boldsymbol{T}, \boldsymbol{g}_j)\right),$$

where the function h maps the goal pair and initial states to a vector of discrete variables characterizing the interaction. We assign positive values to those goal pairs that have the same features as the ground-truth goal pair. It indicates that they are under the same interaction mode as the ground-truth one. Regarding the loss function, maximizing the log-likelihood of positive goal pairs could be misleading. There could be a large ratio of goal pair candidates under the same interaction mode. Inspired by [96], we adopt a loss function to minimize the probabilities of negative labels:

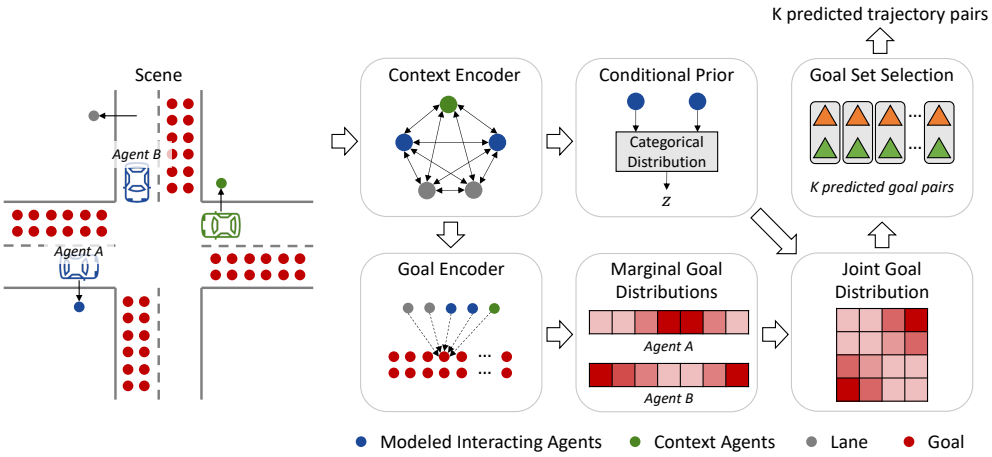
$$\begin{aligned} & f^{\text{interact}}\left(\hat{\boldsymbol{y}}_j^{\text{interact}}, p_\phi(\cdot|\boldsymbol{T}, \boldsymbol{z})\right) \\ = & \sum_{i=1}^{|\mathcal{G}_{a,b}|} \mathbf{1}(\hat{\boldsymbol{y}}_{j,i}^{\text{interact}} = 0) \log\left(1 - p_\phi(\boldsymbol{g}_i|\boldsymbol{T}, \boldsymbol{z})\right). \end{aligned}$$

In the toy example, we adopt an interaction feature indicating which agent has longer displacement in 20 steps, i.e., $\mathbf{1}\left(s_{a,0} - s_{a,20} > s_{b,0} - s_{b,20}\right)$. With this feature, we can identify which agent decides to yield. By incorporating this pseudo interaction label, we are able to separate the interaction modes in the latent space and obtain a model similar to the one shown in Fig. 3.2.

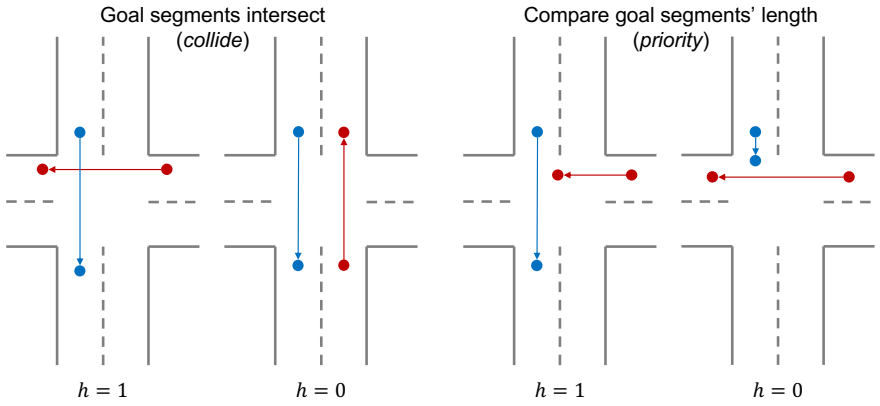
It is worth noting that pseudo-interaction labels are only applied to the distribution decoded from the latent variable to which the ground-truth goal pair belongs. In other words, we only require an interaction mode in the latent space consistent with the ground truth instead of enforcing all the predicted goal pairs to satisfy the constraints. As a result, we can avoid over-regularization and unnecessary bias. Also, we do not require a comprehensive set of pseudo-interaction labels covering all kinds of interactive traffic scenes. We can apply the pseudo labels designed for the specific scenarios of interest without worrying about harming the model performance in the other scenarios. For those scenarios where the designed labels are not applicable, all the goal pair candidates have the same features as the ground-truth one. Therefore, the auxiliary loss is always zero, and the pseudo labels are ignored.

3.4 Framework Architecture

In this section, we introduce the architecture of the model we propose for interactive trajectory prediction. As illustrated in Fig. 3.3a, the model consists of three modules: 1) A marginal goal prediction module which predicts the goal distribution of each interacting agent separately; 2) A joint goal prediction module which explicitly models the joint distribution of goal pairs based on the predicted marginal distributions; 3) A trajectory completion module which predicts the trajectory of each agent conditioned on sampled goal points.



(a) Overall Model Architecture



(b) Pseudo Interaction Labels

Figure 3.3: Overall Model Architecture and Pseudo Interaction Labels.

Modules

Marginal Goal Prediction

We choose DenseTNT [63] as the backbone model when designing the marginal goal prediction module. Specifically, we extract features of maps and agents using the vectorized encoding method proposed in [56]. Afterward, we use the context embeddings to generate goal embeddings for a dense goal set $\mathcal{G}$. The goal set is sampled from the HD maps to cover the drivable area of the modeled agents. We follow DenseTNT to use the attention mechanism in [212] to extract local information between the goals and the scene. We denote the embeddings obtained at this stage for the dense goals and interacting agents as $\boldsymbol{F} \in \mathbb{R}^{|\mathcal{G}| \times d_g}$ and $\boldsymbol{L} \in \mathbb{R}^{2 \times d_v}$ respectively, where d_g and d_v are the dimensions of goal and agent embeddings.

The interaction prediction track of WOMD has a prediction horizon of 8s. It is difficult

to capture the multimodality in long-term trajectory distribution with a single goal point. We follow [64] to model the goal distributions in an autoregressive manner at 3s, 5s, and 8s, respectively. To encourage the usage of interaction information in goal prediction, we add an MLP to update the interacting agents' embeddings at each timestep as follows:

$$\hat{\boldsymbol{L}}_{t,i} = \text{MLP}\left(\boldsymbol{L}_i, \boldsymbol{L}_{-i}, \boldsymbol{F}_{k^i_{1:t-1}}, \boldsymbol{F}_{k^{-i}_{1:t-1}}\right),$$

where $\boldsymbol{F}_{k^i_{1:t-1}}$ collects the embeddings of the i^{th} agent's goals at prior timesteps. The marginal probability of the k^{th} goal for the i^{th} agent at each timestep is then predicted as:

$$\phi^i_{t,k} = \frac{\exp\left(\text{MLP}(\boldsymbol{F}_k, \hat{\boldsymbol{L}}_{t,i})\right)}{\sum_{j=1}^{|\mathcal{G}|}\exp\left(\text{MLP}(\boldsymbol{F}_j, \hat{\boldsymbol{L}}_{t,i})\right)}. \tag{3.2}$$

At the training stage, we follow the well-known practice in autoregressive model training by feeding the ground-truth goals of the previous timesteps.

Joint Goal Prediction

With the marginal goal distributions at timestep t, we first select the top-M goal candidates for each agent based on their marginal probabilities and then models the joint distribution over the M^2 goal pair candidates. As mentioned in Sec. 3.3, we model the joint distribution with a CVAE and utilize the pseudo labels to induce an interpretable interactive latent space. The conditional prior encoder models the distribution of $\boldsymbol{z}$ conditioned on $\boldsymbol{L}$. The posterior encoder further conditions $\boldsymbol{z}$ on $\boldsymbol{F}_{k^1_{1:T}}$ and $\boldsymbol{F}_{k^2_{1:T}}$, i.e., the embeddings of the two agents' ground-truth goals. Both the conditional prior and posterior encoders are modeled with simple MLPs. To decode the joint goal distribution from a sampled $\boldsymbol{z}$, we first obtain a joint agent embedding $\tilde{\boldsymbol{L}}_t \in \mathbb{R}^{1\times d_h}$ as follows:

$$\tilde{\boldsymbol{L}}_t = \text{MLP}\left(\boldsymbol{L}, \boldsymbol{F}_{k^1_{1:t-1}}, \boldsymbol{F}_{k^2_{1:t-1}}, \boldsymbol{z}\right).$$

We obtain the features of goal pairs by concatenating the corresponding goals' embeddings and their marginal probabilities and then encoding them into embeddings of the same dimension as the joint agent embedding with simple MLPs. We denote the resulting goal pair embeddings as $\tilde{\boldsymbol{F}}_t \in \mathbb{R}^{M^2\times d_h}$. We then use the attention mechanism to gather the local information of goal pairs:

$$\begin{aligned}
\boldsymbol{Q}_t &= \tilde{\boldsymbol{F}}_t\boldsymbol{W}^Q, \\
\boldsymbol{K}_t &= \left[\tilde{\boldsymbol{F}}_t\boldsymbol{W}^K_m; \tilde{\boldsymbol{L}}_t\boldsymbol{W}^K_v\right], \\
\boldsymbol{V}_t &= \left[\tilde{\boldsymbol{F}}_t\boldsymbol{W}^V_m; \tilde{\boldsymbol{L}}_t\boldsymbol{W}^V_v\right], \\
\bar{\boldsymbol{F}}_t &= \text{softmax}\left(\frac{\boldsymbol{Q}_t\boldsymbol{K}_t^\intercal}{\sqrt{d_k}}\right)\boldsymbol{V}_t,
\end{aligned}$$

where $\boldsymbol{W}^Q, \boldsymbol{W}_m^K, \boldsymbol{W}_v^K, \boldsymbol{W}_m^V, \boldsymbol{W}_v^V \in \mathbb{R}^{d_h \times d_k}$ are matrices for linear projection, d_k is the dimension of query/key/value vectors. We predict the joint probability of the k^{th} goal pair at the given timestep in a similar way as Eqn. 3.2.

Trajectory Completion

The trajectory completion is similar to the one in [250] and [63]. Given a sequence of goals, we pass their embeddings to a simple MLP to decode the whole trajectory. The trajectories for the two agents are decoded separately. At the training stage, the teacher forcing technique is applied by feeding the ground-truth goal sequences when training the trajectory completion module.

Training Scheme

To train the overall model, we first train the marginal goal prediction module together with the trajectory completion module. The loss function is the same as in [63]. Afterward, we freeze the parameters of these modules and train the joint prediction module. The objective function is essentially ELBO but with the auxiliary losses corresponding to the three types of pseudo labels introduced in Sec. 3.3. In particular, the pseudo interaction labels are defined for each pair of segments connecting goal points at neighboring timesteps (e.g., 0s-3s, 3s-5s, 5s-8s). As illustrated in Fig. 3.3b, for each pair of segments, the pseudo interaction labels are two indicators showing: 1) if the goal segments of the two vehicles intersect; 2) if the goal segment of the first vehicle is longer than the one of the second vehicle. The first feature gives us a hint on whether the two vehicles have a conflict zone along their driving directions. The second feature provides a necessary condition on their right-of-way. If a vehicle has the right-of-way, it should have a larger average speed than the vehicle yielding to it.

Goal Selection

At test time, we need to select a final small number of goal pairs for prediction. The most widely used algorithm is NMS. However, such a heuristic approach is difficult to tune and is not guaranteed to find the optimal solution. To address this issue, an optimization-based approach is proposed in [63] to select a goal set from a predicted distribution. While we may adopt it to select goal pairs at a single timestep, it still remains heuristic when sampling from the latent space as well as the autoregressive model. To ensure a fair comparison among the different model variants studied in Sec. 3.5, inspired by [33], we instead first randomly sample N sequences of goal pairs and then fit them to a Gaussian mixture model (GMM) with K components. We take the mean values of the components as the final K goal pair sequences and set the likelihood of each predicted goal pair sequence as the probability of the corresponding component.

3.5 Experiments

We evaluate the proposed prediction model on WOMD. In particular, we focus on the interaction prediction track, where the future trajectories of an interacting pair for the next 8 seconds are predicted, given the historical observation for the past 1 second. We used the subset of the dataset with labeled interaction pairs of *vehicles* for training and evaluation. With the experiments, we would like to answer:

- Do the pseudo labels induce a meaningful latent space distinguishing different interactive behaviors?
- Does a meaningful latent space improve prediction performance and sampling efficiency?

Model Variants. Our experiment mainly focuses on ablation studies, comparing our model against multiple variants of it. We compare the performance of three models: 1) The *Joint-Vanilla* model, which is our joint prediction model without the pseudo labels; 2) The *Joint-NonInteract* model, which uses pseudo distance and marginal labels in addition to the vanilla version; 3) The *Joint-Full* model, which is the one we propose, i.e., the joint prediction model with the auxiliary losses corresponding to all the proposed pseudo labels (i.e., distance, marginal, interaction). We do not experiment with other methods from the literature since our core contribution lies in utilizing the novel pseudo labels to induce a non-trivial and interpretable latent space. Achieving state-of-the-art performance on the benchmark is not our objective.

Training Settings. To train the overall model, we first train the marginal goal prediction module together with the trajectory completion module following most of the hyperparameters introduced in [63]. Then, we select $M = 65$ goal candidates based on the marginal probability for each agent and train the joint goal prediction module. We add annealing on the KL divergence weight.

Evaluation Metrics. We use these metrics—minADE, minFDE, and mAP—introduced in [44] to evaluate the interactive prediction performance. The metrics for joint prediction involve the predicted trajectories of two interacting vehicles at the same time. The definitions of minADE and minFDE are similar to the single-agent case. However, the displacement errors are computed between the trajectory pairs and their ground-truth labels jointly. The mAP metric is a newly proposed metric for the Waymo Open Challenge. It computes the average precision over eight different ground-truth trajectory primitives defined based on the dataset.

Empirical Prediction Results

In Table 3.1, we compare the prediction performance of the model variants on the validation dataset. We evaluate the prediction over 20000 validation samples in 3s, 5s, and 8s time horizons with the metrics introduced before. The results for the three-time horizons

Table 3.1: Validation Results on All Samples

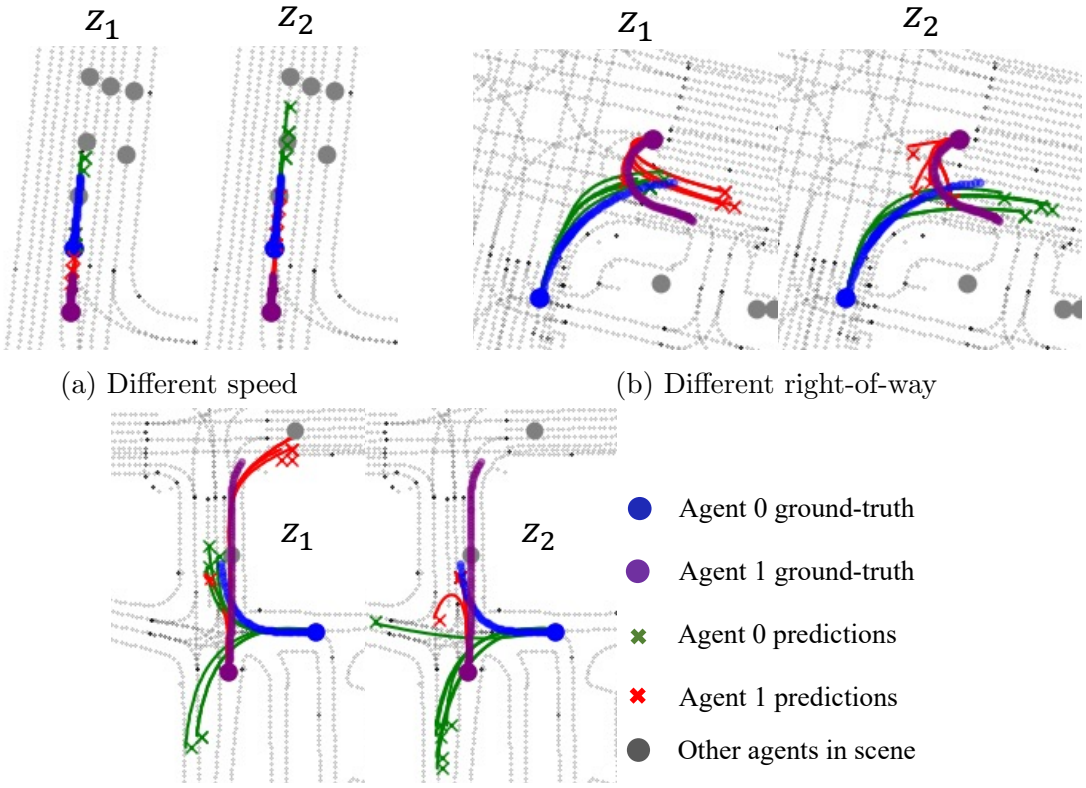
Method	minADE	minFDE	mAP
Joint-Vanilla, N=120	1.58	3.44	0.078
Joint-Full, N=120 (Ours)	1.55	3.33	0.084
Joint-Vanilla N=8	1.98	4.28	0.020
Joint-Full N=8 (Ours)	**1.89**	**4.09**	**0.027**

Table 3.2: Ablation Study on Strong-Interactive Samples

Method	minADE	minFDE
Joint-Vanilla, N=8	1.89 (0.06)	4.11 (0.17)
Joint-NonInteract, N=8	1.88 (0.04)	4.02 (0.07)
Joint-Full, N=8 (Ours)	**1.76** (0.02)	**3.78** (0.04)

are averaged and reported. We show the evaluation results based on different numbers of samples before GMM fitting, with $N = 8$ and $N = 120$. In all the experiments, we set $K = 6$ regardless of the values of N to ensure a fair comparison in prediction errors. From Table 3.1, we can see that the prediction performance is sensitive to the sample number N. With larger N, the sampled trajectories are more likely to cover the multimodality in joint distribution, which leads to more diverse and accurate predictions after GMM fitting. From the table, we can see that the *Joint-Full* model always has better performance under the same sample number N. Note that in online prediction, the maximum allowable N is directly determined by the required computational time. Our purpose is to get accurate and diverse predictions with a small sample number N to enable efficient online inference. We indeed observe a larger improvement with the use of pseudo labels when $N = 8$ compared to $N = 120$.

To evaluate our proposed joint prediction model in highly interactive scenarios, we select a set of strong-interactive cases from the validation dataset. Joint modeling of the behavior of the interacting agents is critical for these highly interactive scenarios, which is the main motivation behind our proposed method. We select the data samples where goal segments of two vehicles intersect by using the pseudo interaction labels introduced in Sec. 3.4. The prediction results of models using different pseudo labels are shown in Table 3.2. Since mAP is extremely sensitive to hyper-parameters when N is small, we do not consider the mAP comparison for quantitative analysis. As the number of selected samples is small compared to the complete validation set (351 of 20000), we evaluate each model three times and report the mean and the standard deviation. We observe a significant improvement in prediction performance and stability by adding interaction pseudo labels (*Joint-Full* model). With a well-trained latent space, we are more likely to cover more interaction patterns even if the number of samples is limited, leading to smaller prediction errors in these strong-interactive



(a) Different speed

(b) Different right-of-way

(c) Different route selection

Figure 3.4: Comparison of 6 sampled first-step goal predictions conditioned on 2 different selected latent z value *Joint-Full*. Different interaction modes can be found in different latent values, meaning we have learned a meaningful latent space.

cases, especially when N is small.

Latent Space learned by CVAE with Pseudo Labels

During training, we indeed observed that pseudo labels, especially marginal pseudo labels, help avoid KL vanishing in most cases. To demonstrate the interactive pattern encoded by latent space, we visualize predicted trajectories for selected interactive scenarios from the dataset, as shown in Fig. 3.4. We use the *Joint-Full* model under different latent variables in the same scenario to make these predictions. Given the historical information, we sample six different goal pairs from the joint goal distribution prediction model conditioning on two different discrete latent variables z with the largest probabilities. In Fig. 3.4, we can clearly see two different interaction modes with different z. The agents either change their speed, route, right-of-way, or combinations of these features when switching the latent variables. Meanwhile, the *Joint-Vanilla* model fails to give a separated latent space (e.g., predictions

sampled from different latent variables are similar) in the same scenarios because of KL vanishing. This shows that our proposed model indeed learns an interpretable latent space capturing the interaction modes inherited from the pseudo labels.

3.6 Chapter Summary

In this chapter, we study the interaction prediction problem under the goal-conditioned framework. To develop an interpretable and sampling-efficient prediction model, we leverage the CVAE framework to capture diverse interaction modes in joint goal distribution explicitly. We find the vanilla model is prone to suffering from posterior collapse, resulting in an uninformative latent space. We explore the underlying reasons in a toy example and propose a general and flexible approach to mitigate this issue with pseudo labels incorporating domain knowledge on interaction. We show that the pseudo labels guide the model to learn an interpretable latent space in our experiments.

Part II

Interactive Behavior Generation

Chapter 4

Critical Interaction Generation in Autonomous Driving

4.1 Introduction

In Chapter 2 and 3, we introduce interpretable prediction on different agents in interactions. In the following two chapters, we focus on how these interactions are generated and whether we can generate realistic tail interactive events.

Self-driving vehicles are expected to make transportation systems much more efficient in the future with smart decision-making systems that can avoid irrational behaviors leading to potential dangers. The safety and robustness evaluation of autonomous vehicle planners during online operation remains an essential but unsolved problem. As in all the other engineering fields, one fundamental philosophy to tackle this problem is through comprehensive testing. In reality, it takes hundreds of miles for autonomous vehicles to encounter various safe-critical cases. Such an evaluation process is time-consuming and risky since we cannot control other traffic participants' behavior on the road. As a result, researchers have proposed multiple evaluation methods in simulator environments, where the key problem degenerates to designing diverse and natural test cases. To take advantage of prior knowledge and collected natural interaction data, researchers propose data-driven learning methods for planner evaluation. In the testing process, *learned* interactive adversary (competitive or weakly competitive) agents are controlled to interact with the tested vehicle, and we use safety metrics like collision rate to evaluate planners' performance. We can figure out possible failure modes by varying the environment and adversary agents in test cases and improving the decision-making algorithms. A common approach in previous test case generation methods is to sample diverse initial states of the adversary agents [39, 38]. However, one drawback of this approach is that they usually assume over-simplified adversary agents with little reaction to the tested vehicle. In reality, the adversary agents usually alter the speed and orientation based on observation of other vehicles during the interaction. Another popular approach is to train adversary agents with Reinforcement Learning (RL) to minimize

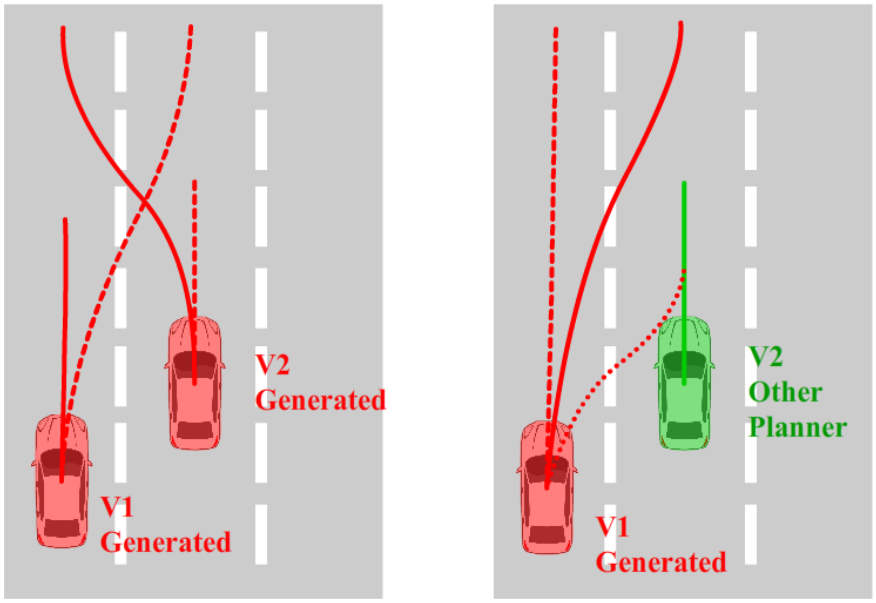


Figure 4.1: **Left:** Previous generative methods produce various interactions by generating the trajectory of all the involved vehicles. **Right:** We plan on a single vehicle to generate diverse interactions and allow the other vehicle to use arbitrary planners (including the proposed RouteGAN).

the driving performance of tested agents [18]. The trained adversary agents have complex driving behavior. Still, their reactions are usually unnatural since they do not learn from human driving data and cannot be generalized to different road structures. To summarize, our main question is:

*Can we design interactive adversary agents for testing, which have **diverse**, **natural** behaviors in **various scenarios**?*

We notice that some previous data-driven trajectory generation methods can produce diverse, near-authentic trajectories [36, 37]. However, these methods aim to generate the whole interaction data. In other words, they generate the trajectory of all the agents jointly rather than controlling a single agent conditioning on its observation of other agents. Therefore, we cannot directly use joint trajectory generation to control adversarial agents for testing case generation. Our proposed method is designed to remove such restrictions and apply data-driven methods to adversary agents training in planner evaluation. In this chapter, we propose RouteGAN, a deep generative model that generates diverse interactive behaviors of a controlled vehicle using observations of the surrounding vehicles. Instead of modeling trajectories jointly as $P(x_1^{0:T}, x_2^{0:T}|h, m)$ where h, m are history and map information respectively, RouteGAN models a single vehicle behavior $P(x_1^{0:T}|x_2^g, h, m, q)$ that reacts to other

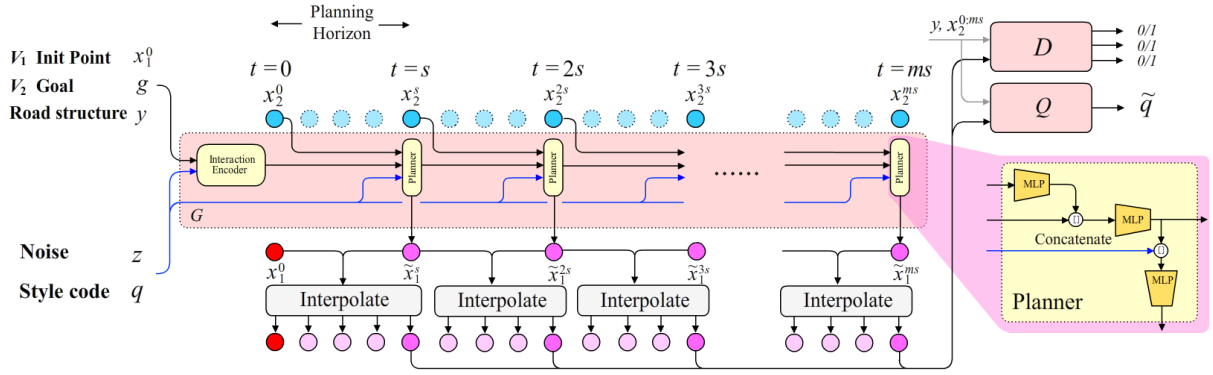


Figure 4.2: This figure shows the overview framework of our proposed algorithm. The generator network G only generates several key waypoints (darker pink circles), and the other intermediate states are from interpolation (light pink circles). Our algorithm can be used for V_1's planning as follows. At each planning step (i.e., $0, s, 2s, ..., ms$), we will use the past and current information of V_2 to generate the next waypoint. Then, we use interpolation to generate all the intermediate points on the trajectory.

tested vehicles as shown in Figure 4.1. To ensure the diversity of generated trajectories, we use a style variable q to control the reacting behavior. In particular, one dimension of the style variable represents how critical the generated trajectory is, which allows us to produce safe, near-critical, and critical interactions using the estimated intentions of surrounding vehicles x_2^g. As a result, RouteGAN can be used as a safe planning module and the adversary agents' planners during planner evaluation.

The goal of this chapter is to.

- Propose RouteGAN, which can produce the styled behavior of a single agent in multi-agent interaction scenarios. The proposed model controls the styles of agents separately and iteratively generates the whole interaction.

- Demonstrate Multi-branch safe/critical discriminators and Auxiliary Distribution Networks that are designed to ensure style control over the generated behavior of the interacting vehicle. Experiments show that the model can generate diverse interactions in various scenarios.

- Use RouteGAN as an online opponent vehicle planner to test the performance of different planners. Results show that varying style input of RouteGAN-controlled opponent vehicles increases the collision rate for rule-based and data-driven planning models.

4.2 Prerequisite

Safe-critical Planner Evaluation

There are various kinds of frameworks designed to test the planner's performance under safe-critical cases. The purpose of these frameworks is to test if autonomous vehicles can make safe decisions while interacting with unknown drivers in different environments. Creating critical test scenarios for testing is one popular direction [38]. [39, 38] use adaptive sampling to generate multi-modal safe-critical initial conditions in cyclists-vehicle interactions. [99] generates critical scenarios with evolutionary algorithms. Another way to test the planners is to change the behaviors of other participating vehicles. Field Operational Tests (FOT)[6] directly use collected data to simulate opponents, [18] uses reinforcement learning (RL) to learn adversarial agents of critical driving styles. An obvious limitation of data-driven methods like FOT is the rareness of natural critical cases. RL-based methods suffer from poor generalization ability under different environments and unnatural generated behavior affected by reward design. Our work follows the second way but aims to extract diverse and controllable behavior from existing driving datasets.

Path Planning and Trajectory Representation

path planning in autonomous driving aims to find a trajectory (curve) that connects a start position and an end position. Such trajectory should avoid collision with obstacles and have some desired properties like smoothness and small curvature. The planned trajectory is usually represented by parametric models. Common parametric models include polynomials [79], splines [180, 100], clothoids [59], and Bezier curves [28, 154]. [155] provides a review of these methods. We use optimization methods to search for the optimal parameters of models that meet our requirements.

Deep Generative Models

In recent years, researchers have proposed various deep learning-based generative models to represent the data distribution. VAE [97] assumes that the latent variable to generate the data follows a Gaussian distribution. VAE is used by recent trajectory methods for latent space interpolation [37]. There are several variants of VAE. One important variant is the CVAE, whose generation process is conditioned on a controllable variable. GAN [62] trains a generator network to produce near-authentic data by an adversary process. InfoGAN [23] proposes to maximize the mutual information between latent variables and the generated data. The mutual information maximization in InfoGAN is implemented by introducing an auxiliary distribution network for variational lower bound maximization. InfoGAN is able to learn disentangled and interpretable latent representation. Such property is used in our work to produce data of various styles.

4.3 Problem Formulation

Trajectory Generation

While trajectory prediction frameworks focus more on predicting future trajectories given historical observations, styled trajectory generation focuses more on generating interactive trajectories based on the initial states and goals of different agents. The reason is that styles can be inferred from historical observations, and they usually remain the same during the whole interaction. We assume that the collected interaction data follows this assumption. As a result, the styled generation problem is formulated as generating trajectories of all k interacting vehicles $\{x_i^{1:T}\}_{i=1}^k$ given their initial states $\{x_i^0\}_{i=1}^k$ and controllable input c corresponding to different interaction factors (styles, goals, etc.) Unlike most previous generation works that jointly produce all vehicles' trajectories, our proposed framework is designed to generate diverse trajectories of one certain vehicle in a finite planning horizon s based on the current state and goals of all the traffic participants $\{x_g\}_{i=1}^k$ to create various kind of reactions. Instead of controlling a single variable c for the whole interaction, we assume different styles for all the vehicles $\{q_i\}_{i=1}^k$. The future trajectory of the ith agent $x_i^{t:t+s}$ is generated using $[\{x_i^t\}_{i=1}^k, \{x_g\}_{i=1}^k, y, q_i]$. The agent plans every s step after observing other vehicles' up-to-date states, and the planning horizon s is a hyperparameter for decision frequency in practice. In this way, we can generate the interacting behaviors of a participating agent with controlled styles. If all vehicles are controlled by their styled generators, we can iteratively generate the joint behavior of multiple vehicles. If we control only the ego vehicle, the generation framework can be directly used as an online planner.

Planning and Planner Evaluation

For simplicity, we consider interactions between two vehicles on the road in the later discussion and denote them as V_1 and V_2, respectively. In the planning setting, as discussed before, the route generator can be used to plan the trajectory of vehicle V_1 given past observations and the estimated goal of an unknown driver V_2. The first dimension of V_1's style variable $q^{(1)} := q_1^{(1)}$ in our generation model can be used to control the conservative degree of the planner.

Using this generator as an opponent agent, we can also perform tests on different planners. Assume V_1 is still controlled by our generator with varying q_1 representing different driving styles, and V_2 is controlled by a rule-based, data-driven, or human-controlled planner, we can perform a safety evaluation on V_2 planner by changing the style of V_1 from conservative to aggressive. For most planners, we can expect safety metrics like collision rate to increase as we increase the critical factor $q^{(1)}$ of our RouteGAN controller.

4.4 Method

Overview

The proposed framework, RouteGAN, is illustrated in Figure 4.2. The generator network G outputs the trajectory as follows. It first encodes V_1's initial positions x_1^0, the final goal g of V_2, the road structure y, the style code q, and the noise z into an initial hidden vector h^{-s}. Here, the road structure y is represented by a bird-eye view image. The positions of V_1 and V_2 are based on such images. We use (-1, -1) and (1, 1) to describe the up left corner and bottom right of y, respectively. The final goal g is the expected position of V_2 in the future and is represented by a coordinate. As is stated above, our planner plans for every s step from time $t = 0$. At time ks, the planner module combines previous hidden vector $h^{(k-1)s}$ with style code q and current observation of V_2 to produce the next hidden vector h^{ks} and next *key waypoint* of V_1, denoted as $\tilde{x}_1^{(k+1)s}$. Then, we use interpolation to generate the trajectory of V_1 between current position x_1^{ks} (i.e., $\tilde{x}_1^{ks}$) and the generated key waypoints $\tilde{x}_1^{(k+1)s}$:

$$x_1^{ks:(k+1)s} = \text{Interpolate}(\tilde{x}_1^{ks}, \tilde{x}_1^{(k+1)s}), k = 0, 1, ..., m-1. \tag{4.1}$$

In order to make the generator produce natural and diverse trajectories, a multi-branch discriminator D (natural) and an auxiliary distribution network Q (diverse) are introduced. Our method assumes access to a safe interaction dataset as well as a critical interaction dataset. The discriminator D should determine whether a trajectory is natural and whether a given interaction is from the safe dataset or the critical dataset. The auxiliary distribution network Q should reconstruct q from the generated interaction to ensure that the generated interaction contains the style information.

In the remaining subsections, we introduce the detailed structure of RouteGAN in 4.4, the loss functions, and the training process of RouteGAN in 4.4. The interpolation method is introduced in 4.4, and finally, in 4.4, we briefly discuss the difference between our methods compared to prior generative methods.

RouteGAN

Generator G

The generator G first packs up the environment information of the interaction.

$$z_{scene} = F_{scene}(y), \tag{4.2}$$

$$z_g = F_g(g), \tag{4.3}$$

$$z_{init} = F_{init}(x_1^0). \tag{4.4}$$

In the above equations, F_{scene} is a Convolutional Neural Network (CNN) [61]. F_g and F_{init} are Multi-Layer Perceptrons (MLP) [61]. Then, we combine these vectors to produce the initial hidden vector.

$$h_{init} = h^{-s} = H_{init}([z_{scene}, z_g, z_{init}, q, z]). \tag{4.5}$$

H_{init} is an MLP. $[\cdot]$ refers to vector concatenation. Then, the generative network iteratively uses the style variable q and the current observation of V_2 to update the hidden vector and predict the future position of V_1. The update of the hidden vector is computed as

$$z_2^t = H_2(x_2^t), t = 0, s, 2s, ..., ms, \tag{4.6}$$

$$h^t = F_{update}([z_2^{t-s}, h^{t-s}]), t = s, 2s, ..., ms. \tag{4.7}$$

H_2 and F_{update} are MLPs. Finally, we use the hidden vector, the style code q, and noise z to predict the next key waypoint.

$$\tilde{x}_1^{t+s} = \tilde{x}_{1,q}^{t+s} = F_{trajectory}([h^t, q, z]), t = 0, s, ..., (m-1)s. \tag{4.8}$$

s $F_{trajectory}$ is an MLP. q takes value from $[-2, 2]^c$, where c is the dimension of q. As we have discussed before, we use the first dimension of q, denoted as $q^{(1)}$, to represent the critical rate of the interaction. z is a random vector and is sampled from a normal distribution.

Discriminator D

The discriminator is trained to distinguish the fake interactions from the real interactions. To model the interaction style, we build two extra branches in the discriminator network for safe and critical interactions. The three branches are denoted as D_{valid}, D_{safe}, and $D_{critical}$ respectively. D_{valid} takes the key point sequence and the scene as input, and it tries to distinguish generated waypoint-scene pair $(\tilde{x}_1^{0,s,...,ms}, y)$ from real waypoint-scene pairs, which are sampled from the dataset. D_{safe} takes the interactions (i.e., the key waypoint sequences of an interactive vehicle pair $(\tilde{x}_1^{0,s,...,ms}, \tilde{x}_2^{0,s,...,ms})$) as input and it tries to distinguish the generated interaction from real, safe interaction pairs drawn from the dataset. Similarly, $D_{critical}$ takes the interactions as input, and it tries to distinguish the generated interaction from real, critical interaction pairs drawn from the dataset.

Auxiliary Distribution Network Q

In order to ensure that the style of the generated key waypoint sequence can be controlled by the style variable q, we use an auxiliary distribution network Q to maximize the mutual information between q and $\tilde{x}_1^{0,s,...,ms}$. This can avoid the case where the model treats style variable q as a dispensable input and only uses past trajectories for future generations. In practice, we train a auxiliary distribution network Q to reconstruct q with $\tilde{x}_1^{0,s,...,ms}, \tilde{x}_2^{0,s,...,ms}$ and y.

Loss functions

We introduce the following loss functions to train our RouteGAN.

Discriminator loss

The loss of D_{valid} is defined as

$$\mathcal{L}^D_{valid} = -\mathbb{E}(\log(D_{valid}(x^{0,s,\ldots,ms}, y)) + \log(1 - D_{valid}(\tilde{x}_1^{0,s,\ldots,ms}, y)).$$

Here, $x^{0,s,\ldots,ms}, y$ is randomly drawn from the dataset. Concretely, we randomly select a trajectory $x^{0:T}$ in the dataset and extract its key waypoints $x^{0,s,\ldots,ms}$. Then, we put $x^{0,s,\ldots,ms}$ and its corresponding scene observation y together to create a real sequence-scene pair. The loss of D_{safe} is defined as

$$\begin{aligned}\mathcal{L}^D_{safe} = -\mathbb{E}(&\log(D_{safe}(\Gamma(x^{0,s,\ldots,ms}_{safe:1}, x^{0,s,\ldots,ms}_{safe:2}))) + \\ &\log(1 - D_{safe}(\Gamma(x_2^{0,s,\ldots,ms}, \tilde{x}_1^{0,s,\ldots,ms}))) + \\ &\log(1 - D_{safe}(\Gamma(x^{0,s,\ldots,ms}_{critical:1}, x^{0,s,\ldots,ms}_{critical:2})))).\end{aligned}$$

In the above equation, $x^{0,s,\ldots,ms}_{safe:1}$ and $x^{0,s,\ldots,ms}_{safe:2}$ are key waypoint sequences extracted from a safe interaction $(x^{0:T}_{safe:1}, x^{0:T}_{safe:2})$ sampled from the dataset. $x^{0,s,\ldots,ms}_{critical:1}$ and $x^{0,s,\ldots,ms}_{critical:2}$ are key waypoint sequences extracted from a critical interaction $(x^{0:T}_{critical:1}, x^{0:T}_{critical:2})$ sampled from the dataset. Γ is a differentiable augmentation operation. Such augmentation is a composition of normalization and random rotation transformation defined as

$$\Gamma(x_1, x_2) = (U(x_1 - \mu(x_1, x_2)), U(x_2 - \mu(x_1, x_2))). \tag{4.9}$$

Here, $\mu(x_1, x_2)$ is the mean position of the two trajectories, U is a random rotation matrix. We find that such an augmentation step can help RouteGAN learn an invariant and robust representation for diverse behavior generation. Minimizing $\mathcal{L}^D_{safe}$ enforces D_{valid} network to discriminate critical interactions and fake interactions from the real, safe interactions. Similarly, the loss function of $D_{critical}$ is defined as

$$\begin{aligned}\mathcal{L}^D_{critical} = -\mathbb{E}(&\log(D_{critical}(\Gamma(x^{0,s,\ldots,ms}_{critical:1}, x^{0,s,\ldots,ms}_{critical:2}))) + \\ &\log(1 - D_{critical}(\Gamma(x_2^{0,s,\ldots,ms}, \tilde{x}_1^{0,s,\ldots,ms}))) + \\ &\log(1 - D_{critical}(\Gamma(x^{0,s,\ldots,ms}_{safe:1}, x^{0,s,\ldots,ms}_{safe:2})))).\end{aligned}$$

The loss function of the discriminator is the combination of the above losses:

$$\mathcal{L}^D = \mathcal{L}^D_{valid} + \mathcal{L}^D_{safe} + \mathcal{L}^D_{critical}. \tag{4.10}$$

Generator loss

The goal of the generator is to generate V_1's future trajectory $\tilde{x}_1$ so as to create authentic interactions which can confuse the discriminator. The loss function of the generator contains

three terms $\mathcal{L}^G_{valid}, \mathcal{L}^G_{safe}$, and $\mathcal{L}^G_{critical}$. They correspond to the three discriminator loss terms above, respectively. They are defined as

$$\mathcal{L}^G_{valid} = \mathbb{E}(\log(1 - D_{valid}(\tilde{x}_1^{0,s,\ldots,ms}, y)), \tag{4.11}$$

$$\mathcal{L}^G_{safe} = \mathbb{E}(\log(1 - D_{safe}(x_2^{0,s,\ldots,ms}, \tilde{x}^{0,s,\ldots,ms}_{1,q^{(1)}<0})), \tag{4.12}$$

$$\mathcal{L}^G_{critical} = \mathbb{E}(\log(1 - D_{critical}(x_2^{0,s,\ldots,ms}, \tilde{x}^{0,s,\ldots,ms}_{1,q^{(1)}>0})). \tag{4.13}$$

Then, the loss of the generator is defined as

$$\mathcal{L}^G = \alpha\mathcal{L}^G_{valid} + \mathcal{L}^G_{safe} + \mathcal{L}^G_{critical}. \tag{4.14}$$

Here α is a hyperparameter balancing the weight of $\mathcal{L}^G_{valid}$. This is designed to encourage diversity since some scenarios in the dataset only provide a single mode, such as following a straight line. An $\alpha < 1$ can introduce more possible modes into these scenarios.

Information loss

The goal of the Q network is to reconstruct the style variable q. Therefore, we minimize the following L_2 loss.

$$\mathcal{L}^Q = \frac{1}{2}\mathbb{E}\|q - Q(\tilde{x}^{0,s,\ldots,ms}_{1,q}, y)\|^2.$$

Road constraint loss

We also find it useful to add in a road constraint loss. This term can ensure that the generated key points lie within the road. It is defined as follows. First, we use a heat map operation $\mathbb{R}^2 \to \mathbb{R}^{w\times h}$ to transform the generated key point onto a 2D map. It is defined as

$$\text{Heatmap}(x)(u, v) = \exp\left(-\frac{(u - x_1)^2 + (v - x_2)^2}{2\sigma^2}\right). \tag{4.15}$$

Here, σ is a hyperparameter. This mapping is differentiable so we can define the road constraint loss as

$$\mathcal{L}_{road} = \text{mean}\left(\frac{1}{m}\sum_{k=1}^{m}(1 - y) \cdot (\text{Heatmap}(\tilde{x}_1^{ks}))\right). \tag{4.16}$$

The overall optimization process proceeds as follows. For each training step, we optimize the D network using the loss function $\mathcal{L}^D$ for four steps with G and Q fixed. Then we fix D network and optimize G and Q using the loss function $\mathcal{L}^{G,Q}$ defined by

$$\mathcal{L}^{G,Q} = L_G + \lambda_1 L_Q + \lambda_2 L_{road}. \tag{4.17}$$

Here, λ_1 and λ_2 are two hyperparameters balancing the weight of each loss term.

Interpolation

To obtain the trajectory of the vehicle between discrete key waypoints, one simple approach is piecewise linear interpolation. However, piecewise linear interpolation will lead to non-smoothness at each key waypoint. Therefore, we only use linear interpolation for the first two key waypoints x_1^0, x_1^s. For the interpolation between the latter key waypoints, we use the Bezier curve, and the process is shown in Figure 4.3. Let P_0, P_1 be two consecutive key waypoints. D_0 is the orientation of the vehicle at P_0. Then, we determine D_1, the orientation of the vehicle at P_1 by the following heuristic. Let α be the angle (with sign) between D_0 and P_0P_1, β be the angle (with sign) between P_0P_1 and D_1. Then, we set $\beta = \alpha k$. k is a scaling parameter and is heuristically defined as 0.25. Finally, the control point C of the Bezier curve is set to be the intersection point of the two lines defined by (P_0, D_0) and (P_1, D_1). It is explicitly given by

$$C = P_0 + \frac{(P_{1x} - P_{0x})D_{1y} - (P_{1y} - P_{0y})D_{1x}}{D_{0x}D_{1y} - D_{0y}D_{1x}} D_0. \tag{4.18}$$

We compute the trajectory γ between P_0 and P_1 using P_0, C and P_1, i.e. $\gamma(t) = P_0(1-t)^2 + 2Ct(1-t) + P_1t^2$.

Discussion

There are two main factors that our model differs from the previously proposed learning based trajectory generation models. First, previous methods usually use the generator network to generate the full trajectory. In contrast, our generator network only generates few key waypoints, and we obtain the *full* trajectory by interpolation. We find this modification crucial in experiments as it makes the generation process focus more on the global shape information rather than local details, which can bring out various styles.

Secondly, we split the discriminator into different branches and introduced an augmentation step. This operation can decouple the interaction from a particular viewpoint and avoid mode collapse.

4.5 Experiments

Settings

We use the Argoverse dataset [16] and the INTERACTION dataset [244] to evaluate RouteGAN. Argoverse is used by the latest interaction generation method CMTS [37]. In our experiments, we adopt the same modified Argoverse trajectory dataset used by CMTS, which contains interaction data in the straight road and intersection scenarios. We also use the interaction data in roundabout scenarios in INTERACTION. Since the number of critical interaction data in these datasets is less than that of safe interactions, we apply some simple

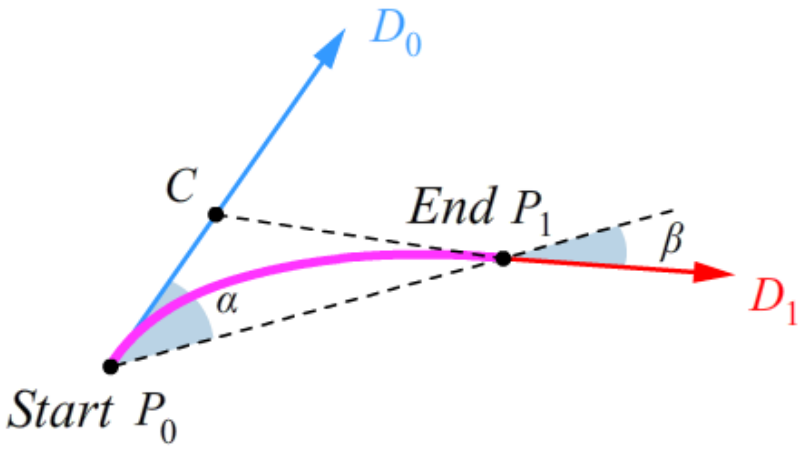


Figure 4.3: Bezier curve interpolation used in the proposed method.

data augmentation tricks to generate some pseudo-critical interactions. The tricks include temporal realignment, which relabels the timestamps of safe interactions to turn them into critical interactions, and local deformation, which slightly alters the trajectory's shape to produce intersections.

Qualitative Case Study

In this part, we visualize some typical generated interactions on Argoverse and INTERACTION. In order to make it easier to understand the effect of $q^{(1)}$, which is the first dimension of the style variable of vehicle V_1, we assume that V_2 simply follows the trajectory in the dataset. We design three common intersection cases:
Case I. V_1 and V_2 go in the same direction. At the beginning, V_1 is behind V_2.
Case II. V_1 and V_2 go in the same direction. At the beginning, V_1 is ahead of V_2.
Case III. V_1 and V_2 go in the opposite direction. Note that in the roundabout scenario, V_1 and V_2 can not go in the opposite direction since it is invalid. Therefore, this case is substituted by intersection.

The generation interaction results of the above cases on two datasets are shown in Figure 4.4. We select two different scenarios for each case. From up to down, we set $q^{(1)}$ to $-2.0, -1.0, 0.0, 1.0, 2.0$ with another dimension of style code q fixed, which shows a transition from safe interaction into critical interaction. Compared with our method, we find that CMTS can not provide promising results on roundabout scenarios though it is verified on Argoverse. Therefore, we do not display its result here, and we refer readers to the paper directly for its result on Argoverse.

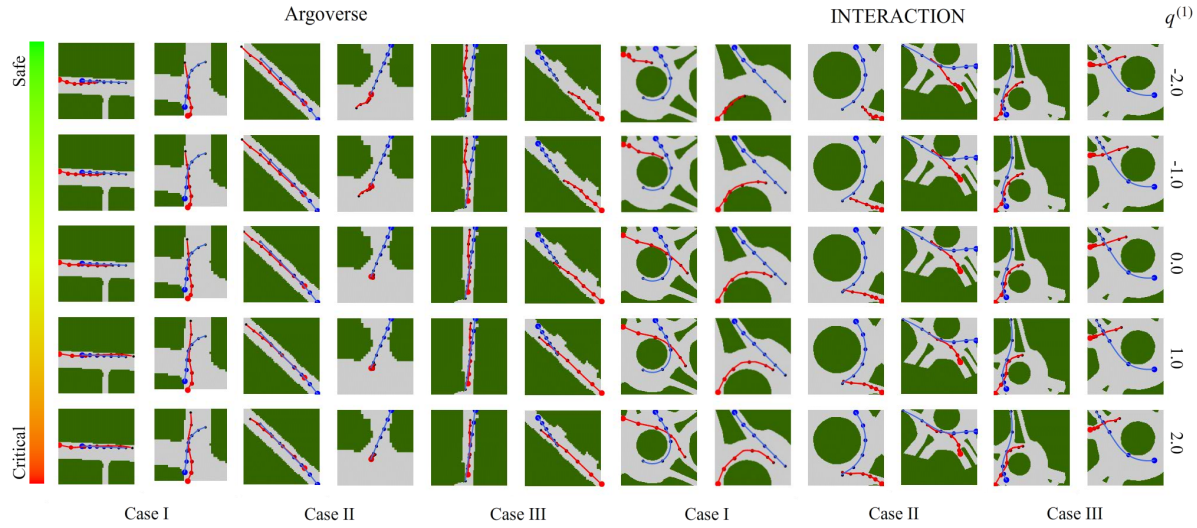


Figure 4.4: Generated interaction examples on Argoverse dataset and INTERACTION dataset. Throughout the later part of this chapter, the red line indicates the trajectory of V_1 (Rou,teGAN) and the blue line indicates the trajectory of V_2. The start point is indicated by the largest circle.

Result on Case I

For the safe interaction, V_1 follows V_2 slowly. As $q^{(1)}$ increases, V_1 gradually speeds up and hits V_2 from behind.

Result on Case II

For the safe interaction, the typical result is that V_1 speeds up and avoids collision with V_2. Another kind of generated interaction is that V_1 turns to another lane and gives its way to V_2. However, as $q^{(1)}$ increases, V_1 stops in the center of the road or even goes backward, which leads to crashes if V_2 does not slow down in time.

Result on Case III, Argoverse

For the safe interaction, V_1 turns to a different lane or slow down. As $q^{(1)}$ increases, V_1 no longer stays away from V_2 and runs into V_2's lane.

Result on Case III, INTERACTION

For the safe interaction, V_1 proceeds based on the priority (i.e., V_2 go first). But as $q^{(1)}$ increases, V_1 no longer waits for V_2 and crashes into it regardless of its lower priority.

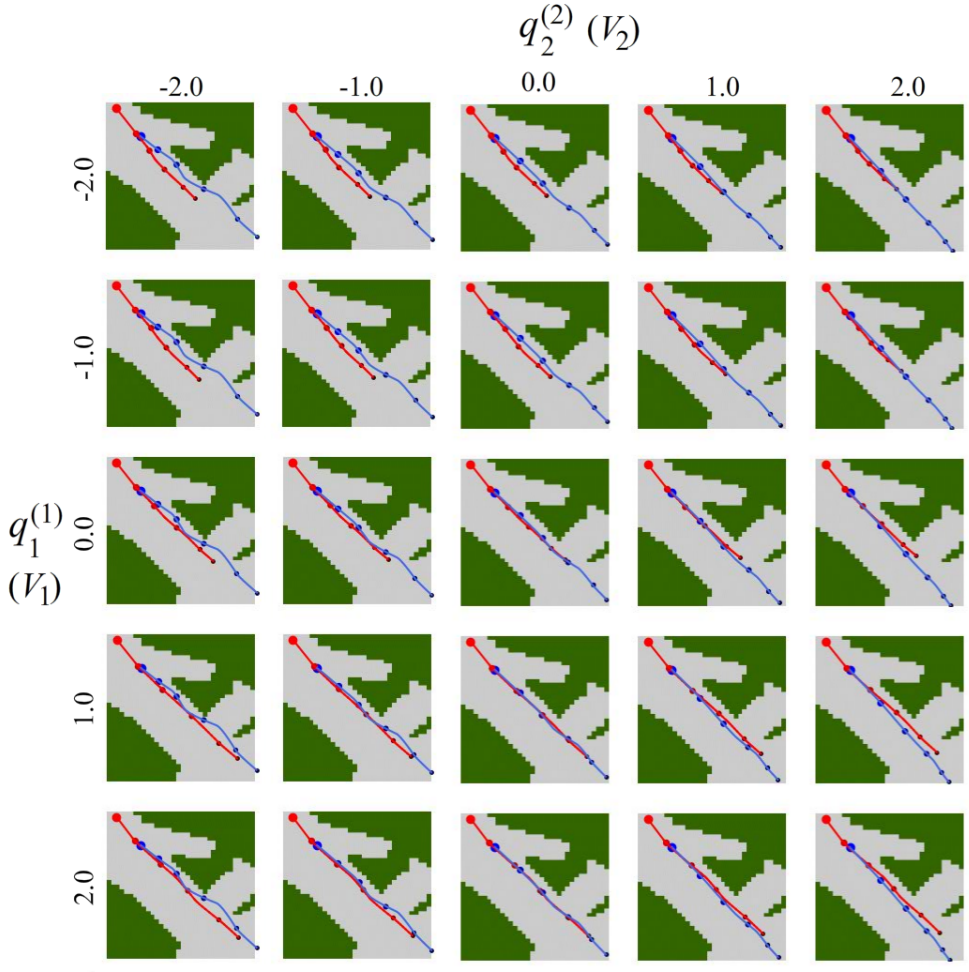


Figure 4.5: The result of generating interactions jointly. Both V_1 and V_2 are controlled by RouteGAN.

Joint Generation of Interactions

We also demonstrate that RouteGAN is able to generate interactions jointly as [36, 37, 65]. In this section, both V1 and V2 are controlled by RouteGAN, and their style codes are denoted as q_1 and q_2, respectively. For a car-following scenario, we sweep $q_1^{(1)}$ and $q_2^{(2)}$ in the latent spaces with step size 1.0 and fix z and all the other dimensions of q_1 and q_2. The reason for this design is that the $q_2^{(2)}$ controls the geometry of the leading car V_2 and $q_1^(1)$ controls the interacting style of V_1 in this car-following case. One example is shown in the Figure 4.5. Similarly, when $q_1^{(1)}$ increases, the generated V_1's trajectories gradually become more critical as expected. $q_2^{(2)}$ controls the 'bending' style of V_2's trajectory.

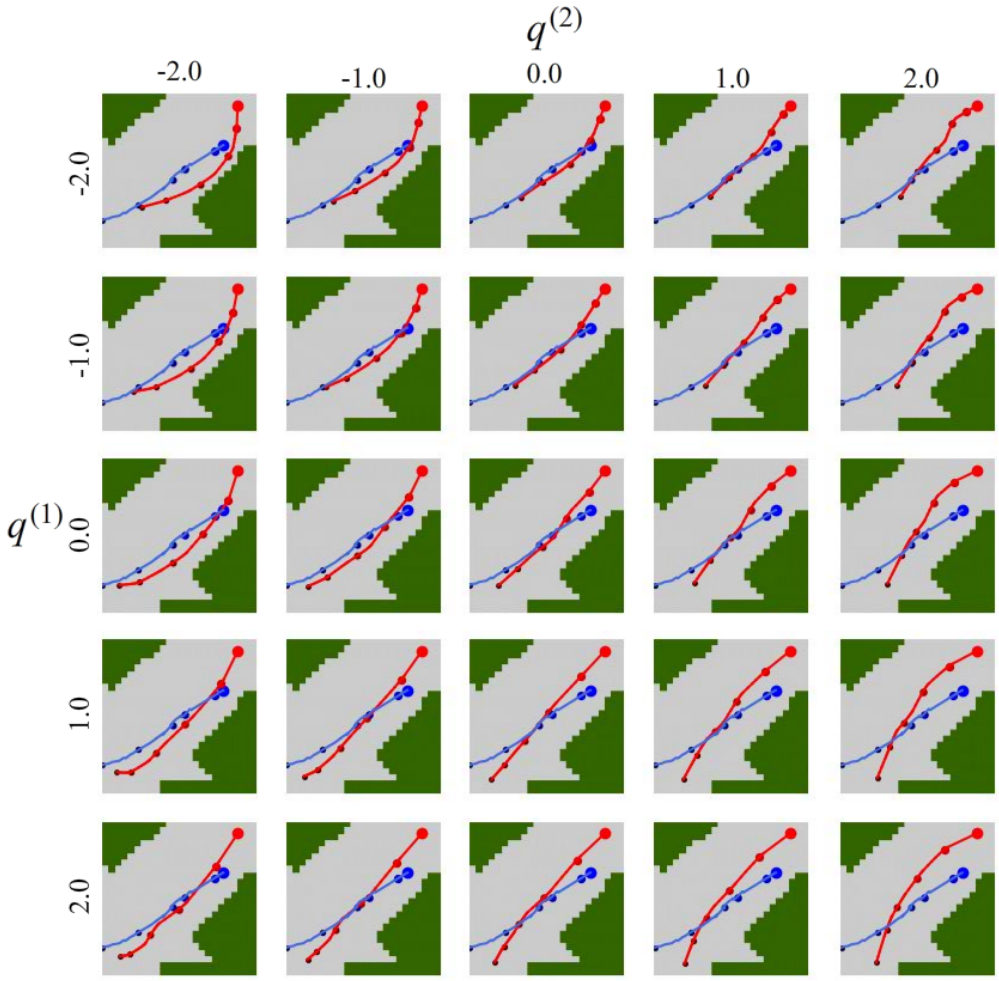


Figure 4.6: The result of latent space sweeping.

Diversity of Generated Interactions

In this subsection, we verify that RouteGAN is able to generate diverse interactions. Controlling both cars with RouteGAN and sweeping the latent space is more complicated since the vehicles are affecting each other, making comparison between interactions hard. Therefore, we control V_1 here to show the diversity of planned trajectories given the same observations. We sweep $q^{(1)}$ and $q^{(2)} := q_1^{(2)}$ in the latent space $[-2.0, 2.0] \times [-2.0, 2.0]$ with step size 1.0 and fix z and other dimensions of q. Then, we use these codes to generate interactions. One example is shown in the Figure 4.6. When $q^{(1)}$ increases, the generated V_1's trajectory gradually intersects with V_2's trajectory as expected. Meanwhile, $q^{(2)}$ controls the 'bending' style of V_1's trajectory. As $q^{(2)}$ varies from -2.0 to 2.0, the trajectory of V_1 will bend upwards and follow V_2 from different directions.

Application: Planner Testing

One direct application of RouteGAN is simulation-based autonomous driving decision system testing. To test a planner's response to various interactions, we can use RouteGAN as its adversary in the interaction. In this experiment, we verify RouteGAN's testing performance. Since $q^{(1)}$ reflects the level of interaction safety, we expect that the collision rate increases as $q^{(1)}$ increases. We assume that the tested system is able to carry out perfect control. In other words, the low-level controller can strictly follow the trajectory calculated by the upper-level planners. We choose the following decision-making systems to test.

Data Planner

A Data planner is a dummy planner that only follows the reference trajectory in the dataset.

IDM Planner

Intelligent driver model (IDM) [117] is a differential dynamic system that models the dynamics of multiple vehicles on an infinitely long line. To apply this model in 2-dimensional planning, we project the coordinate of V_1 onto V_2's reference trajectory and use the Runge-Kutta 4 method [12] to calculate the future position.

Astar Planner

Astar [68] is a basic planning baseline in AI and autonomous driving. It will search for optimal acceleration along the reference trajectory at each time step.

Table 4.1: The collision rates of planners interacting with different styles of RouteGANs

V_1	V_2	Coefficient $q^{(1)}$				
		-2.0	-1.0	0.0	1.0	2.0
RouteGAN	Data	4.2%	9.1%	47.1%	88.6%	95.3%
RouteGAN	IDM	1.4%	3.0%	8.7%	18.1%	24.4%
RouteGAN	Astar	0.0%	0.0%	1.5%	4.8%	6.5%

The testing result is shown in the Table 4.1. We observe that the collision rate increases as $q^{(1)}$ goes up. Since the data planner simply follows the reference regardless of the motion of V_1, the collision rate of it is high when $q^{(1)} > 0$. The collision rate of IDM and Astar planner is in general much lower than that of the data planner. However, we find that IDM planner is still not good enough when it comes to intersection cases since one assumption of IDM planner is that V_1 will move along V_2's reference trajectory. Note that the result for RouteGAN itself is not shown in our planner evaluation. This is because we do not

put collision as a soft or hard penalty in our model to generate critical cases. Therefore, RouteGAN is not the best choice for collision avoidance. From the experiment, we can see that the collision rate in planner evaluation is directly affected by the driving style of adversary agents. Given the same initial conditions, aggressive drivers can better test the robustness of planners in critical cases. Our designed planner evaluation framework can help planners to improve their critical-case performance.

4.6 Chapter Summary

In this chapter, we investigate the problem of generating diverse interactive behavior of controlled vehicles. We propose RouteGAN, a generative model to solve this problem by controlling a style variable to produce safe or critical interaction behaviors with participating vehicle observations. We demonstrate the diversity of generated trajectories in different traffic scenarios, proving its ability to be a safe planner. Finally, we use it as an adversary agent to perform safety evaluations on different planners and validate the collision rate increase by varying the adversary agent's style.

Chapter 5

Distributed Human-like Interaction Modeling for Enhanced Human-Robot Collaboration

5.1 Introduction

In the previous chapter, we discuss interaction generation in autonomous driving scenarios, where roads are designed with structures and interactions happen following traffic rules. We then switch to a more general indoor scenario and discuss the interactions between human humans and human robots.

Modeling the interactive behavior of multiple agents in different scenarios is an essential task in crowd simulation. One of the main challenges is to model the interactive behaviors of multiple agents, especially in narrow scenarios where they have to avoid obstacles simultaneously. The agents' decisions are interdependent, meaning that each agent's decision influences and is influenced by the decisions of the other agents. In real life, humans can cooperate without explicit communication; instead, we make decisions based on observations. In this project, we aim to analyze and model the interactions in a distributed game setting without communication. The problems can be naturally formulated as a multi-agent planning problem with separate goals and a shared environment. Some previous works have used centralized [198, 34] algorithms, which solve trajectories of all agents together to control all robots. However, centralized methods are computationally expensive and require full information about the environment and other agents. In contrast, distributed algorithms separately solve the short-horizon reaction plans [9, 251] or long-horizon trajectory plans [171, 121, 217] for each robot. Distributed methods are more scalable and robust but suffer from deadlocks, especially in human-like interaction cases where no communication is allowed. For instance, as shown in Figure 5.1, in a narrow-way situation, if both agents are in the hallway, they have no collision-free navigation plans to their goals. If agents have no information from the others, they need to estimate other's intentions and figure out how to

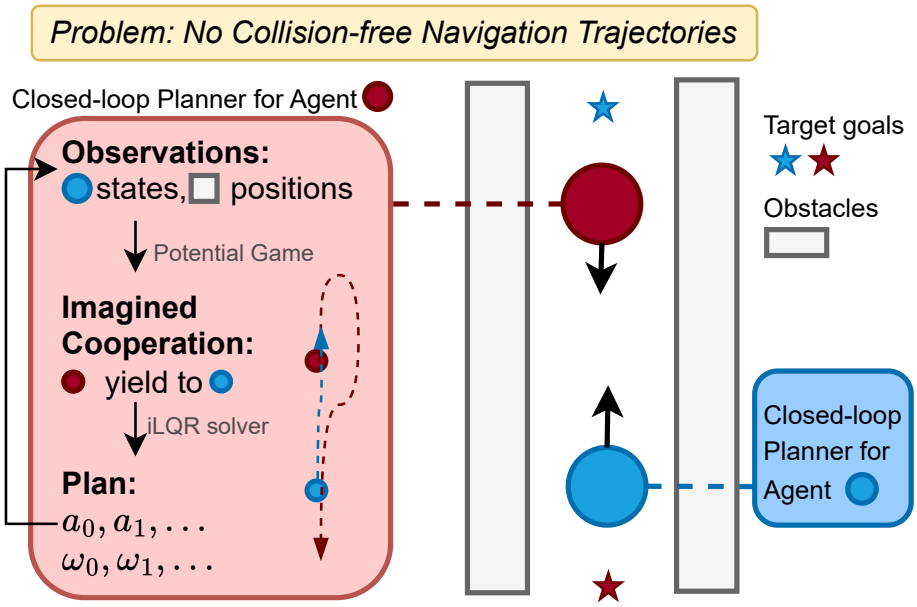


Figure 5.1: A narrow-way problem challenging to solve in the distributed and no-communication setting. There's no collision-free for single-agent navigation, and agents must cooperate (one moving backward to yield the other agent) to solve the problem. We propose adding imagined cooperation in distributed planning to simulate cooperative interactions.

cooperate in the distributed setting.

We propose to model and solve the interaction problem in a distributed manner where each agent imagines a cooperation game with others. There are several works considering game-theoretic frameworks [52, 51, 90, 221] to model cooperative behaviors. We follow the dynamic game formulation introduced by [90], where the multi-agent potential game is formulated into an equivalent single optimal control problem to find cooperative plans for all agents. We assume an "imagined" game exists in all agents' distributed planners to predict others' behavior and use the iterative LQR (iLQR) to solve optimal plans. In addition to the basic formulation in [90], we add collision avoidance to environmental obstacles, observation range, and blind area for agents in the optimization since they are essential causes of human-like interactions.

We demonstrate the effectiveness of introducing the "Imagined Potential Game (IPG)" formulation into the multi-agent planning problem without communication. The open-loop plans are used to simulate the closed-loop behaviors using receding horizon control. Agents may have different open-loop cooperative strategies but can gradually converge to cooperative behaviors in closed-loop simulation. Experiments on the narrow-way scenario with random initialization empirically show the improved success rate and navigation efficiency in simulating cooperative interactions. Furthermore, we show the framework's capability to

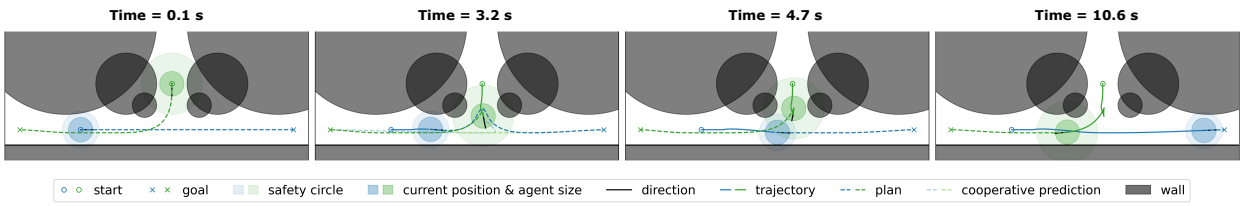


Figure 5.2: An example interaction generated in T-intersection. From left to right is the closed-loop trajectory over time. Solid lines are past trajectories; darker and lighter dotted lines are plans of ego agents and predictions of other agents (e.g., the light blue line is the predicted *green* agent behavior by the *blue* agent). All green lines are from the green agent's planner.

generate diverse and realistic interaction behaviors by varying parameters, such as safety distance or objective weights. Lastly, we demonstrate that the IPG-controlled agents can robustly interact with non-IPG-controller agents. The contribution of this work lies in the following aspects:

1. We propose a multi-agent interaction generation framework using the imagined potential games under a distributed and no-communication setting to simulate interactions in dense obstacle scenarios.

2. We use simulated experiments to empirically demonstrate the improvement in success rate and navigation efficiency for simulating complex interactions.

3. We demonstrate the algorithm can generate diverse and realistic interactions using interpretable parameters and interact with heterogeneous agents.

5.2 Related Works

Multi-robot planning

Multi-robot trajectory planning algorithms can be categorized based on where the computation is done. Two main strategies to solve the problem are *centralized* and *distributed*. Centralized planning algorithms solve the trajectories for all the agents in the same problem utilizing the global information and then send the control commands to the robots. Most previous works use multi-agent path finding (MAPF) solvers with trajectory optimization algorithms[198, 145, 34, 218] to find feasible trajectories for all agents. A series of works [51, 181, 221, 134, 30, 103, 132] models interactions in multi-agent planning via game theoretic frameworks. [90] utilizes potential games [52] for multi-agent trajectory planning with symmetric inter-agent costs. Centralized algorithms can provide theoretical guarantees for planning success and optimality. However, the centralized setting cannot simulate realistic

interactive behaviors in the distributed setting, where single-agent behaviors are affected by other agents' online reactions.

In the distributed setting, each agent runs a separate algorithm to compute its own trajectory. Depending on whether communication exists between the agents. Reactive algorithms[9, 220, 85] like Optimal Reciprocal Collision Avoidance (ORCA) can effectively avoid collisions but fail to avoid deadlocks in environments with dense obstacles[171]. Learning-based reactive strategies[168, 160, 110, 8] are computationally more efficient but suffer from distribution shifts and also experience deadlocks. Another series of works like dMPC[121] and MADER[205] consider longer horizons and generate sequences instead of single actions; however, they require communication of plans for collision avoidance. [225] turns multiple agents into distributed groups and solves the in-group interactions, but the in-group agents are modeled in the centralized setting. RLSS[171] uses a fully distributed setting and requires only the current state sensing of other agents to plan piece-wise Bézier trajectories to improve deadlock performance. However, these methods don't explicitly model the cooperation of agents and, therefore, have trouble simulating interactions when the initial condition is not feasible for collision-free planning.

Interaction Generation

Simulating the behaviors of actors is an important task with a wide range of applications in transportation and robotics research since simulators play essential roles in training and evaluating intelligent agents like indoor robots and autonomous vehicles. Many autonomous-driving research works focus on simulating distributed agents' behaviors in the simulator to generate realistic interactions and reactive agents[233, 196, 237, 17, 247] or analyzing and predicting interactive behaviors of heterogeneous agents[252, 186, 190]. Previous works on crowd simulation[148] focus more on the scalability of simulating agents[203, 204], and grouping in the crowd[167, 125]. In this chapter, we focus on the interactions in scenarios where cooperation is required to simulate multi-agent behaviors.

5.3 Preliminaries

Distributed multi-agent planning

Assume we have N agents in the scenario. For each agent $i, 1 \leq i \leq N$, let the vector $x_i(t) \in \mathbb{R}^{n_i}$ denote the state of agent i, let $u_i(t) \in \mathbb{R}^{m_i}$ denote the control input of agent i at time t. Each agent follows its system dynamics

$$x_i(k+1) = f_i(x_i(k), u_i(k))$$

Unless otherwise specified, throughout this chapter, for variable x, we use a subscript x_i to denote agent i and a superscript x^k or $x(k)$ to indicate the time horizon k. If i, k are

not specified, it means x for all agents or across all time steps. x_{-i} denotes x of all agents excluding agent i.

Obstacles in the scenario are represented by $\{O_j\}_{j=1}^{M}$. Each agent has its initial state x_i^0 and a target goal state in the scenario g_i. All the agents in the scenario navigate to their target goal while avoiding collision with the environment and other agents. Interactions happen when their planned trajectories $\{x_i(0), x_i(1), ..., x_i(T)\}_{i=1}^{N}$ have conflicts and need to interact to reach non-conflict new plans. Control inputs of multiple agents $U = [u_1^{0:T}, u_2^{0:T}, ..., u_N^{0:T}]$ are the plans of all the agents. Under the *distributed setting with no communication*, we assume each agent i is solving an optimal control optimization without knowing others' plans.

$$\begin{aligned} \min_{u_i(0:T), x_i(0:T)} \quad & J_i(x(0), u_i, \tilde{u}_{-i}) \\ \text{s.t.} \quad & x_i(k+1) = f_i(x_i(k), u_i(k)) \\ & h(x_i, \tilde{x}_{-i}, O) \leq 0 \end{aligned} \tag{5.1}$$

$J_i = S_i(x(T), T) + \sum_{k=0}^{T-1} L_i(x(k), \tilde{u}_{-i})$ is the cost function for agent i. The stage cost L_i can include distance, time, and energy costs, and the terminal cost S_i can include goal conditions. The collision-free requirements are described using the constraints $h \leq 0$. The hard constraints in h can be added as weighted cost functions depending on the solver used. $\tilde{u}_{-i}$ and $\tilde{x}_{-i}$ are the estimation of other agents' plans and states to prevent collisions since we assume no communication between agents. To consider other agents during planning, it needs to use predictions. In most cases where environmental constraints are not strict, constant velocity predictions are well enough to provide collision avoidance planning. However, cooperative predictions are required to generate feasible interactions in cases like narrow-way interaction in Figure 5.1. Therefore, we proposed to use an Imagined Potential Game (IPG) framework to predict $\tilde{x}_{-i}$ during the planning of agent i.

Comparison with the *centralized* or *distributed with sharing* setting: In the *centralized* setting, U is solved together in a single large problem given all agents' initial and goal states simultaneously. The weighted (α_i) costs of all agents are optimized in one problem.

$$\begin{aligned} \min_{U} \quad & \sum_{i=1}^{N} \alpha_i J_i(x(0), U) \\ \text{s.t.} \quad & x(k+1) = f(x(k), u(k)), \quad h(x, O) \leq 0 \end{aligned} \tag{5.2}$$

The *distributed setting with sharing* is quite similar to the centralized setting; plans are solved separately but are shared with other agents, enabling accurate predictions in a distributed setting [121] for cooperation. Many game-theoretical interaction models operate in a similar setting. The cost functions of all agents are shared to consider others' behaviors and find equilibrium plans for all agents.

Differentiable Potential Game

We then introduce the definitions of dynamic games (N agents with horizon T) from previous works [90]. We describe a differential game by the compact notation of $\Gamma_{x_0}^{T} = (N, \{U_i\}_{i=1}^{N}, \{J_i\}_{i=1}^{N}, \{f_i\}_{i=1}^{N})$, where x_0 is the initial states of all agents, and each agent seeks to optimize its cost J_i under the dynamic f_i. The cost function $J_i(x(0), U) = S_i(x(T)) + \sum_{k=0}^{T-1} L_i(x(k), U(k))$ consists of running cost L_i and terminal cost S_i. We look for the Nash equilibrium solution of the dynamic game defined by:

Definition 1 *Given a differential game $\Gamma_{x_0}^{T} = (N, \{U_i\}_{i=1}^{N}, \{J_i\}_{i=1}^{N}, \{f_i\}_{i=1}^{N})$, control signal set U is an open-loop Nash equilibrium if, for $i \in [1, ..., N]$:*

$$J_i(x_0, u^*) \leq J_i(x_0, u_i, u^*_{-i}) \tag{5.3}$$

At a Nash equilibrium, no agent has the incentive to change its current control input u_i^* as such a change would not yield any benefits, given that all other agents' controls u^*_{-i} remain fixed. While this equilibrium solution can best represent the cooperative multi-agent behavior in the interaction, finding the Nash equilibrium solution is challenging since there are N coupled optimal control problems to solve simultaneously. Recent progress in solving this problem, especially in robotics applications, find efficient solutions to the problems under certain conditions. As proved in [90], problems in a potential differential game form can be solved by formulating a single centralized optimal control problem. We summarize their main result in the following theorem.

Theorem 1 *For a given differential game $\Gamma_{x_0}^{T} = (N, \{U_i\}_{i=1}^{N}, \{J_i\}_{i=1}^{N}, \{f_i\}_{i=1}^{N})$, if for each agent i, the running cost and terminal cost functions have the structure of*

$$L_i(x(k), u(k)) = p(x(k), u(k)) + c_i(x_{-i}(k), u_{-i}(k)) \tag{5.4}$$

$$S_i\left(x(T)\right) = \bar{s}(x(T)) + s_i(x_{-i}(T)) \tag{5.5}$$

then the open-loop Nash equilibria can be found by solving the following optimal control problem

$$\begin{aligned} \min_{U} \quad & \sum_{k=1}^{T-1} p(x(k), u(k)) + \bar{s}(x(T)) \\ s.t. \quad & x_i(k+1) = f_i(x_i(k), u_i(k)) \end{aligned} \tag{5.6}$$

The key takeaway from this theorem is that one can formulate a differentiable potential game if all the cost function terms can be decomposed into potential functions $(p(\cdot), \bar{s}(\cdot))$ that depend on the full state and control vectors of all the agents, and other cost terms $(c_i(\cdot), s_i(\cdot))$ that have no dependence on the state and control input of agent i. Then the optimal solution for both agents can be solved by the centralized problem Eq.5.6 using only $p, \bar{s}$. The following section will show how this theorem is used for distributed no-communication settings. For each agent in the interaction, it assumes all agents are in the potential game, but it doesn't know the interaction parameters for other agents. Therefore, each agent will solve a separate Imagined Potential Game (IPG) using estimated parameters.

System Notations and Assumptions

To simplify the problem, we make several assumptions on system dynamics. We assume that all agents are modeled using the same unicycle dynamic model. The state vector $x_i = [p_{x,i}, p_{y,i}, \theta_i, v_i]$, the control vector $u_i = [a_i, w_i]$. The discrete-time dynamic equations of the system are:

$$\begin{aligned} p_{x,i}(k+1) &= p_{x,i}(k) + T_s\ v_i(k)\ cos(\theta_i(k)) \\ p_{y,i}(k+1) &= p_{y,i}(k) + T_s\ v_i(k)\ sin(\theta_i(k)) \\ \theta_i(k+1) &= \theta_i(k) + T_s\ w_i(k) \\ v_i(k+1) &= v_i(k) + T_s\ a_i(k) \end{aligned} \tag{5.7}$$

System notations are summarized in Table 5.1.

Table 5.1: Notation describing common variable.

	Definition		Definition (Default value)
p_x, p_y	position in 2D	Q	state weight ([0.01, 0.01, 0, 0])
θ	heading angle	R	input weight ([1, 1])
v	velocity	D	safety weight (40)
a	acceleration	B	back up weight (10)
w	angular velocity	r	safety radius (1.2∼2.0)
O	static obstacles	T_s	sampling time (0.1)

5.4 Distributed multi-agent interaction modeling with imagined potential game

As described in section 5.3, for interactions under the distributed setting with no shared plans, each agent is solving the problem in Eq. 5.1 but needs to estimate the future inputs and states of other agents $(\tilde{x}_{-i}, \tilde{u}_{-i})$ to avoid collision. One common assumption in navigation is the constant velocity prediction, which estimates other agents' states with the current velocity. However, this doesn't work for cooperation-required cases, as shown in Figure 5.1. The potential game formulation in section 5.3 gives optimal actions for cooperative agents but assumes the cost parameters in J_i are pre-known. In this chapter, we use this formulation to make cooperative predictions of other agents with an Imagined Potential Game (IPG) setting but use estimated parameters of other agents to solve the game.

Parameters in agent i's cost function are the goal position g_i and the interaction parameters, including safety radius r_i, and different weights Q_i, R_i, D_i, B_i, each will affect the behavior in interaction. In this project, we assume other agents' long-term goal positions (intentions) are already predicted. Long-term goal prediction and short-horizon interaction

are separate problems, and we focus on the latter. For interaction parameters; agents will assume other agents using the same parameters they have, e.g., $\tilde{r}_j = r_i$ for all $j \neq i$. This assumption is simple, but in practice, we found it strong enough to simulate diverse interactions, especially when all participants are cooperative agents. For cases where inaccurate estimation of others' parameters causes failure, we can design online adaptation rules (be more conservative) in the interaction (see section 5.5).

Solving Potential game with estimated parameters

Based on Theorem 1, Nash equilibrium solutions of all agents can be solved by Eq. 5.6 if the interaction can be formulated into a potential game. Here, we show the running cost function $L_i(x)$ in the potential game, consisting of the stage cost term $C_{tr,i}^{0:T-1}(x_i, u_i)$, the collision avoidance term $C_{a,ij}(x_i, x_j)$ and the reverse avoidance term $C_{b,i}(x_i)$. The stage cost includes the minimum goal distance and inputs penalty terms using the current state and input:

$$C_{tr,i}^{0:T-1}(x_i, u_i) = (x_i - g_i)^\intercal Q_i (x_i - g_i) + u_i^\intercal R_i u_i \tag{5.8}$$

The collision avoidance term is counted when the distance between two agents d_{ij} is smaller than the safety distance:

$$C_{a,ij}(x_i, x_j) = \begin{cases} (d_{ij} - d_{safe})^2 \cdot D_i, & \text{if } d_{ij} < d_{safe} \\ 0, & \text{others} \end{cases} \tag{5.9}$$

and satisfy the symmetric property $C_{a,ij}(x_i, x_j) = C_{a,ji}(x_j, x_i)$ for potential game described in [90].

The reverse avoidance term discourages the agent for moving backward:

$$C_{b,i}(x_i) = \begin{cases} |v_i| \cdot B_i, & \text{if } v_i < 0 \\ 0, & \text{others} \end{cases} \tag{5.10}$$

To prove this running cost function is a potential game, we can represent the $p(x, u)$ and $c_i(x_{-i}, u_{-i})$ in Theorem 1:

$$\begin{aligned} p(x, u) = &\sum_{i=1}^{N} C_{tr,i}^{0:T-1}(x_i, u_i) \\ &+ \sum_{1 \leq i < j} C_{a,ij}(x_i, x_j) + \sum_{i=1}^{N} C_{b,i}(x_i) \end{aligned} \tag{5.11}$$

$$\begin{aligned} c_i(x_{-i}, u_{-i}) = &- \sum_{j \neq i} C_{tr,j}^{0:T-1}(x_j, u_j) \\ &- \sum_{\substack{1 \leq j < k \\ j,k \neq i}} C_{a,jk}(x_j, x_k) - \sum_{j \neq i} C_{b,j}(x_j) \end{aligned} \tag{5.12}$$

With this representation, we show that the running cost $L_i(x_i, u_i) = p(x, u) + c_i(x_{-i}, u_{-i})$ follows the Theorem 1,

Similarly, the terminal cost $S_i(x(T)) = C^T_{tr,i}(x_i, u_i) = \bar{s}(x(T)) + s_i(x_{-i}(T))$, with terminal cost term:

$$C^T_{tr,i}(x_i, u_i) = (x_i(T) - g_i)^\intercal Q_i(x_i(T) - g_i) \tag{5.13}$$

Set $\bar{s}(x(T))$ and $s_i(x_{-i}(T))$ to be :

$$\bar{s}(x(T) = \sum_{i=1}^{N} C^T_{tr,i}(x_i), s_i(x_{-i}(T)) = -\sum_{j\neq i}^{N} C^T_{tr,j}(x_j) \tag{5.14}$$

We have the required terminal cost $S_i(x(T))$ in Theorem 1.

To address the problem in scenarios involving obstacles, we introduce some extra constraints in addition to the existing dynamic constraints, including the state boundary constraint, input constraint, and obstacle avoidance constraint. These constraints can be added as weighted costs into J_i and don't affect the potential game assumptions.

One important difference between centralized planning and the IPG setting is the safety distance d_{safe}. For centralized planning, the maximum safety distance between them is used $d_{safe} = max(r_i, r_j)$, for IPG, each agent assumes others have the same safety distance, $d_{safe,i} = r_i$, unless it changes its estimation.

The full IPG problem for each agent to solve is:

$$\begin{aligned}
\min_{U} \quad & \sum_{k=1}^{T-1} p(x(k), u(k)) + \bar{s}(x(T)) \\
s.t. \quad & x(k+1) = f(x(k), u(k)) \\
& x(0) = x_0 \\
& x_L \leq x(k) \leq x_U \\
& u_L \leq u(k) \leq u_U \\
& r_{obs} - dis(x(k), O_m) \leq 0, m = 1...M \\
& r_i - dis(x_i(k), x_j(k)) \leq 0, i, j = 1...N
\end{aligned} \tag{5.15}$$

where x_U, x_L, u_U, u_L are the state and input boundaries, and r_{obs} is the radius of the circle obstacle.

5.5 Experiments

In this section, we present the simulating results under the distributed multi-agent interaction setting to address the following key questions of interest:

1. Is the proposed framework able to solve the complex (infeasible) cases in navigation under the distributed setting? Does it prevent the failures that result from deadlock and collision?

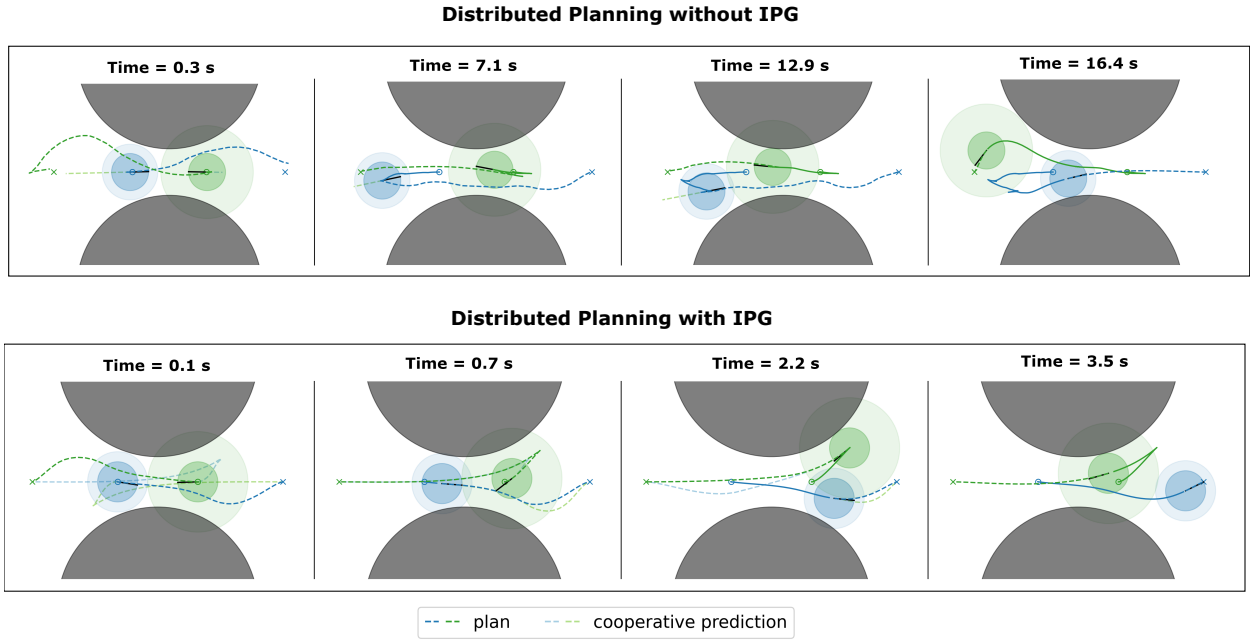


Figure 5.3: In the distributed setting, Vanilla agents without IPG are stuck at a deadlock for a long time and take extremely long time to finish interaction. IPG agents cooperatively interact with each other.

2. Is the proposed framework able to generate rich interactions with various parameters? Is the behavior realistic and diverse for crowd simulation?

Environment settings: We first quantitatively evaluate and demonstrate the interaction generation capability in the classic "narrow way" problem, the most common but challenging indoor navigation scenario where only one agent can pass at one time. Then, we demonstrate and analyze the interactions with more agents or in more complex scenarios.
Comparison baselines: We use two baselines to compare the effect of our proposed framework. 1) *Vanilla:* This is the vanilla version of distributed multi-agent planning. Each agent is running a collision avoidance planning algorithm. The agent will follow the previous plan if the planner cannot generate a feasible collision-free solution. Note that an open-loop collision solution might not cause an actual collision in the closed-loop simulation since distributed agents use wrong predictions during open-loop planning. 2) *Brake:* To avoid collision under infeasible solutions, the agent will brake and stop if the planner solution has collision. This conservative implementation can avoid collisions in simulation but is more likely to cause deadlocks.
Evaluation metrics: We generate 20 test cases with randomized starting and goal points. To make sure interactions happen between agents, we enforce starting and goal points on different sides of the hallway. The safety and navigation parameters are randomly sampled from a pre-defined range for different agents. The metrics for evaluating the interaction

generation are 1) ***Success rate***: We test whether the two agents reach the target goals without collision before the time limit. For failure cases, we separate the failure caused by collision and failure by deadlocks. 2) ***Extra interacting time***: We use the total time for agents to finish interaction under a centralized setting as a baseline to test how much extra time the agents spent in the distributed setting.

Observation range in distributed setting: In the distributed setting, the ego-agent will consider other agents' behaviors unless they are blinded (e.g., there are obstacles on the line connecting the two agents) or other agents are behind the ego-agent and out of a pre-defined sensing range.

Interaction in "narrow way" in distributed setting

Both agents aim to reach the target positions sampled on the opposite side while avoiding collisions with other agents and the walls. The experiment results are shown in Table 5.2.

Table 5.2: Evaluation in Narrow-way Scenario

	Success Rate	Deadlock	Collision	AET(second)
Vanilla	13/20	2/20	5/20	+2.054
Brake	12/20	8/20	0/20	+2.308
IPG(Ours)	20/20	0/20	0/20	+0.395

AET: Average Extra Time compared to Centralized

The proposed method generated interactions successfully under all random settings. The *vanilla* version has a higher success rate than the *the brake* version but caused more collision during closed-loop simulating. For extra time cost for interaction, we use the closed-loop interaction completion time for the same setting under the centralized potential game setting as the baseline and only calculate the average extra time for success cases. All distributed settings experience increased interaction completion time, but our IPG setting is the most efficient in solving interaction.

An illustrative comparison of interaction using different distributed interaction settings is shown in Figure 5.3 using different stages of interactions. The *vanilla* distributed agents stuck at a deadlock for a long time during interaction. We didn't show the *brake* agents since they experienced deadlock and got stuck from the beginning. Using IPG, with an imagined game in mind, agents assume the presence of cooperation and predict others' cooperation using the estimated parameters. This results in one agent yielding to the other or assuming the other's yielding in interaction.

We show how IPG recovers from open-loop deadlocks in Figure 5.4. In the distributed setting, we don't have a guarantee that agents don't get contradicting open-loop plans causing deadlocks. At 3.2s for the *IPG* setting, the blue agent had wrong predictions on the green agent, causing the deadlock. However, as the interaction progressed, we observed the

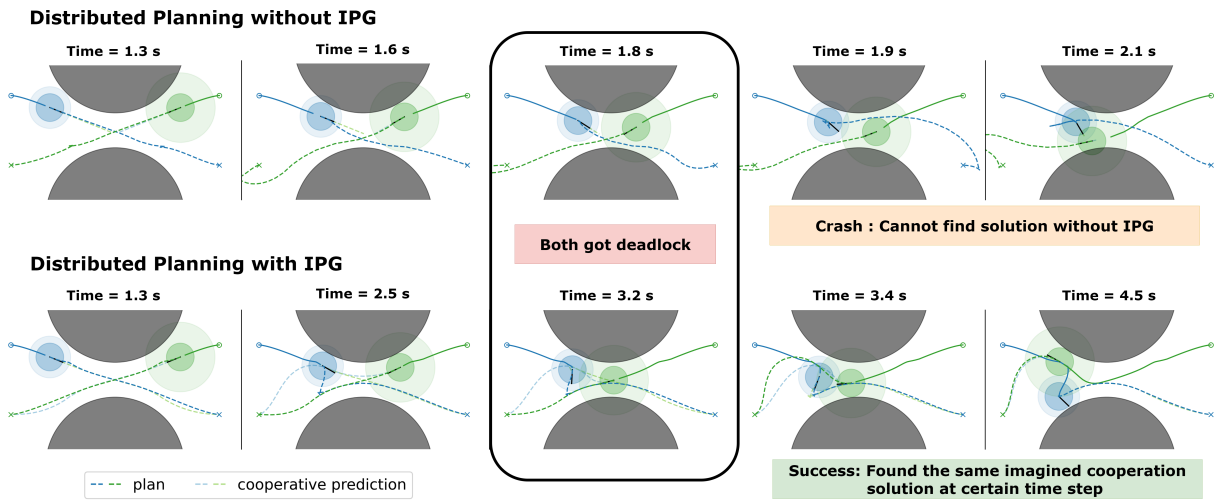


Figure 5.4: Vanilla distributed agents can collide during simulation when facing deadlocks. Agents with IPG also experienced deadlock when the imagined cooperation didn't match. However, IPG agents can converge to success plans in the closed-loop simulation.

predictions converging to correct ones. If agents are assigned non-identical safety parameters, they have a low chance of always getting symmetric and contradicting plans.

Realistic and Diverse Behaviors with IPG

Realistic Behavior in a T-intersection case A typical case and extension for the narrow-way case is the T-intersection case, where the obstacles obscure the agents until one arrives at the intersection. In centralized planning, the optimal cooperative strategy is to wait at the starting point and move until the other agent passes. However, in the distributed setting, the agent cannot observe the other agent until it enters the intersection. Therefore, as shown in Figure 5.2, the agent entered the intersection, realized it had to yield to the other agent, and then retreated to wait. These realistic and human-like reactions happen only in distributed settings. In general, evaluating realistic behavior in rich interactions is hard since these interactions only happen with certain initial conditions, and there's no quantitative metric for realism. We found adding imagined cooperation into planning is an effective way to generate realistic behavior, especially in indoor scenarios, where collision-free trajectories are often not available for some agents.

Interactions of more than two agents Fig 5.5. illustrates the distinct behaviors exhibited by three agents in a centralized and distributed IPG setting. To demonstrate the different behaviors of agents with different safety radii, we chose an open area to interact. In the distributed setting, agents lack the safety distance information of others, and they assume that other agents possess equal safety distance. Therefore, the agent with a larger safety distance (green) will react more conservatively.

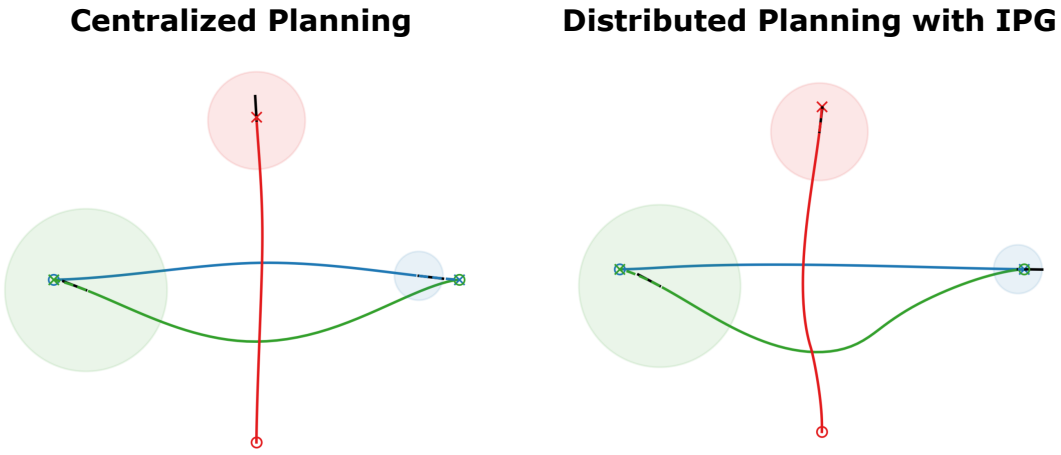


Figure 5.5: Three agents interacting with different safety radii.

Effects of different interaction parameters The effect of safety and cost parameters on agents' behavior is shown in Figure 5.6. The state cost weight Q determines the flexibility in planning, and the safety weight D affects its conservatism in interaction.

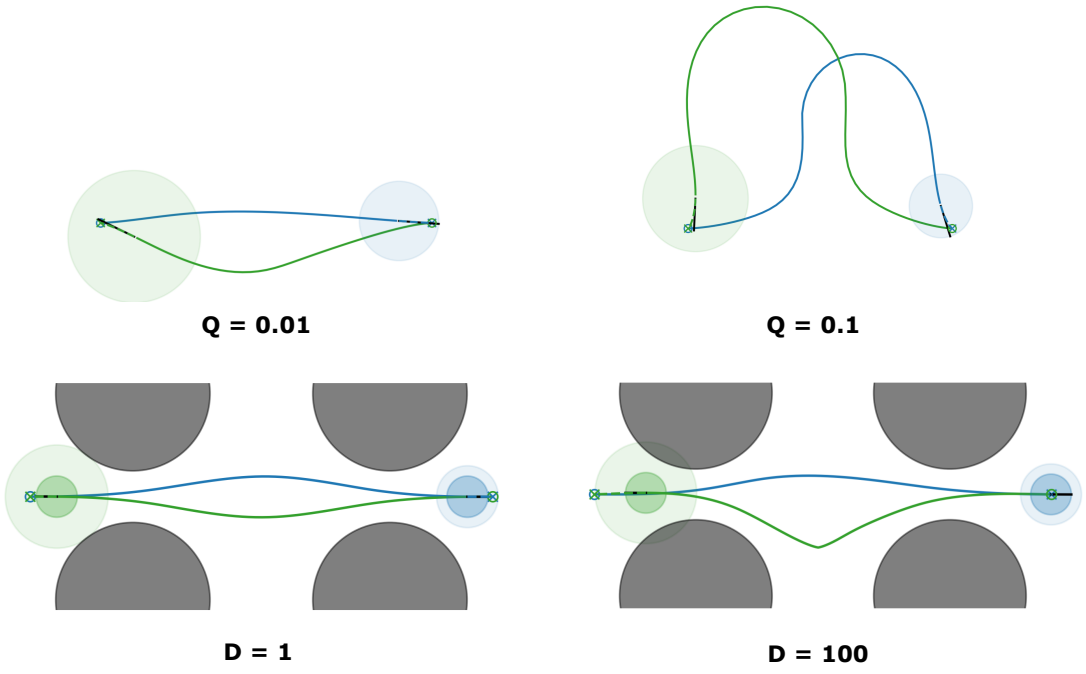


Figure 5.6: Effects of different interaction parameters on behaviors.

Interaction between Heterogeneous agents

In the previous two sections, we've demonstrated the proposed IPG framework for simulating interactions under the distributed setting by preventing collision and deadlocks. An extension of this framework, as mentioned in the introduction, is to use the proposed framework as evaluation agents to interact with tested agents. This requires the IPG agent to interact with different types of agents.

In Figure 5.7, we demonstrate the cases where an IPG agent is asked to interact with a *vanilla* agent using constant velocity prediction and a *ignore* agent ignoring other agents. The IPG agent can react wisely to those non-cooperative agents; however, non-adaptive aggressive parameters might collide or get stuck in interactions; in Figure5.7, the IPG agent increases its safety distance online to resolve the issue. We claim various IPG agents can represent different types of human agents in the simulator to test the robustness and generalizability of social navigation algorithms.

5.6 Discussion

5.7 Chapter Summary

In this chapter, we propose an Imagined Potential Game framework under the distributed without communication setting to model realistic interactions in complex scenarios. We demonstrate the improvement of IPG agents in the distributed setting in success rate and navigation efficiency in simulating interactions and the ability to generate realistic and diverse interactions in different scenarios.

Limitations and Future works The current framework is analyzed in selected scenarios where cooperative interaction is required to solve the problem. In future works, we will adapt the framework for general in-door scenarios where obstacles have random and complex shapes. We also plan to develop a standard test environment, including IPG agents with various interaction parameters to better learn and benchmark social navigation algorithms.

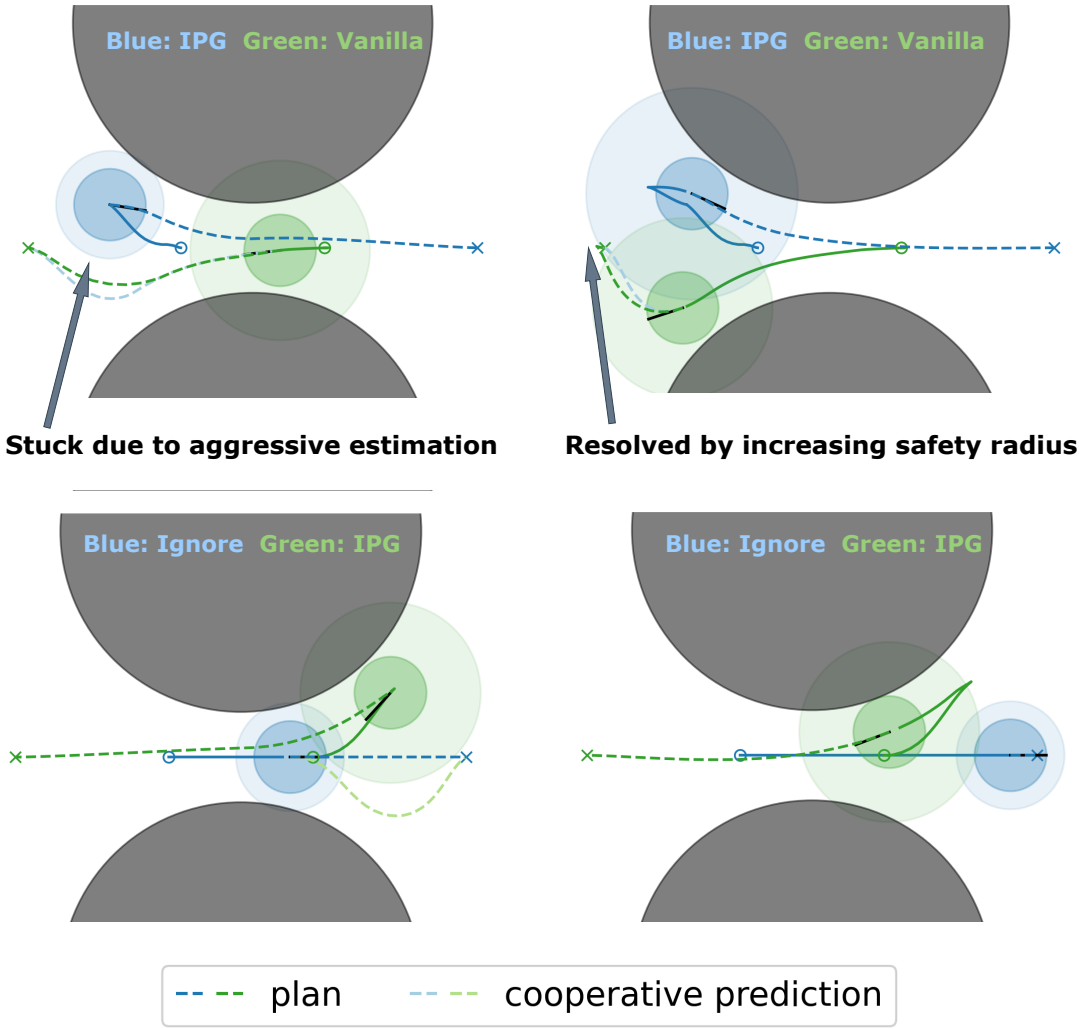


Figure 5.7: Interactions between of heterogeneous agents.

Part III

Generalizable Skill Learning

Chapter 6

Multi-task Learning and Transfer Reinforcement Learning

6.1 Introduction

Deep reinforcement learning (RL) has made massive progress in solving complex tasks in different domains. Despite the success of RL in various robotic tasks, most of the improvements are restricted to single tasks in locomotion or manipulation. Although many similar tasks with different targets and interacting objects are accomplished by the same robot, they are usually defined as individual tasks and solved separately. On the other hand, as intelligent agents, humans usually spend less time learning similar tasks and can acquire new skills using existing ones. This motivates us to think about the advantages of training a set of multiple tasks with certain similarities together efficiently. Multi-task reinforcement learning (MTRL) aims to train an effective policy that people can apply to the same robot to solve different tasks. Compared to training each task separately, a multi-task policy should be efficient in the number of parameters and roll-out environment steps. It should have the potential to generalize to other unseen similar tasks.

The key challenge in multi-task RL methods is determining what should be shared among tasks and how to share. It is reasonable to assume the existence of similarities among all the tasks picked (usually on the same robot) since training completely different tasks together is meaningless. However, the gaps between different tasks can be significant even within the set. For tasks with the same robot but different goals, it's natural to share all the parameters and add the goal into state representation to turn the policy into a goal-conditioned policy. For tasks with different skills, sharing policy parameters can be efficient for close tasks but may bring additional difficulties for uncorrelated skills (e.g., push and peg-insert-side in Meta-World [239]). Also, multiple components in the RL framework can be shared between tasks, for example, the task and input encoder and the policy network. We need to determine how we recognize and solve each task during evaluation.

Recent works on multi-task RL proposed different methods for this problem. We roughly

divide them into three categories. Some researchers focus on modeling share-structures for sub-policies of different tasks [13, 231], while some focus more on algorithms and aim to handle conflicting gradients from different tasks losses during training[238]. In addition, many works attempt to select or learn better representations as better task conditions for the policies. In this paper, we focus on the share-structure design for multiple tasks. We propose a parameter-compositional MTRL method that learns a task-agnostic parameter set forming a subspace in the policy parameter space for all tasks. We infer the task-specific policy in this subspace using a compositional vector for each task. Instead of interpolating different policies' output in the action space, we directly compose the policies in the parameter space. In this way, two different tasks can have identical or independent policies. With different subspace dimensions(i.e., size of parameter set) and additional constraints, this compositional formulation can unify many previous works on sharing structures of MTRL. Moreover, keeping a task-agnostic parameter set brings advantages in extending trained policies to unseen tasks.

The key contributions of this chapter are summarized below. *(i)* We present a general Parameter Compositional (PaCo) MTRL training framework that can learn representative parameter sets used to compose policies for different tasks. *(ii)* We show this formulation can unify many previous MTRL works and introduce stabilization schemes useful for training in our framework. *(iii)* We validate the state-of-the-art performance of PaCo on the Meta-World benchmark without prior task information and verify the learned parameters can be used for unseen tasks in a continual setting.

6.2 Preliminaries

Markov Decision Process(MDP)

A discrete-time Markov decision process is defined by a tuple $(\mathcal{S}, \mathcal{A}, P, r, \eta, \gamma)$, where $\mathcal{S}$ is the state space; $\mathcal{A}$ is the action space; P is the transition process between states; $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function; $\eta \in \mathcal{P}(\mathcal{S})$ is distribution of the initial state, and $\gamma \in [0, 1]$ is the discount factor. At each time step t, the learning agent generates the action with a policy $\pi(a_t|s_t)$ as the decision. The goal is to learn a policy to maximize the accumulated discounted return.

Soft Actor-Critic

In the scope of this work, we will use Soft Actor-Critic (SAC) [66] to train the universal policy for the multi-task RL problem. SAC is an off-policy actor-critic method that uses the maximum entropy framework. The parameters in the SAC framework include the policy network $\pi(a_t|s_t)$ used in the evaluation, and the critic network $Q(s_t, a_t)$ as a soft Q-function. A temperature parameter α is used to maintain the entropy level of policy. In multi-task learning, the one-hot id of task skill and the goal is appended to the state space. Different

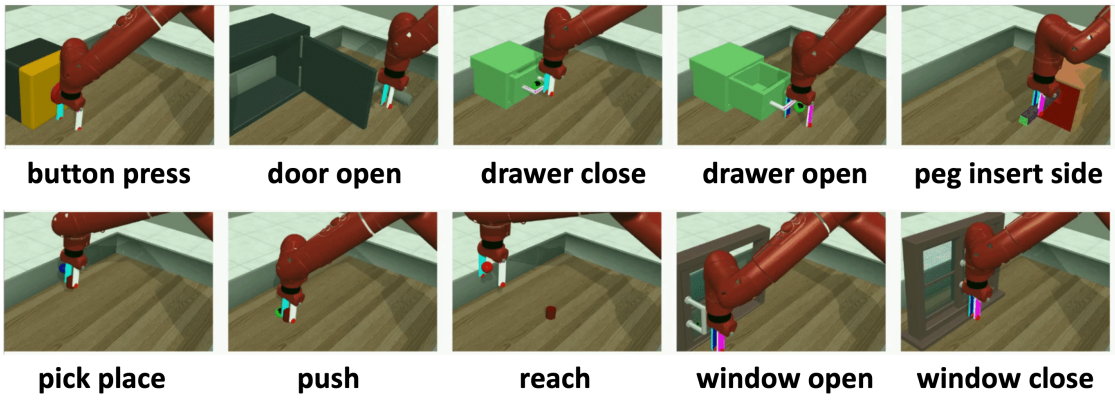


Figure 6.1: **Example tasks** from Meta-World [239]

from single-task SAC, we assign a separate temperature α_τ for each task with different skills. The policy and critic function optimization procedure remains the same as the single-task setting.

6.3 Revisiting and Analyzing Multi-Task Reinforcement Learning and Beyond

In this section, we will revisit the standard MTRL task settings and highlight two inherent challenges to MTRL algorithms on 1) parameter sharing and 2) parameter reusing.

Multi-Task Reinforcement Learning Setting

Each single task can be defined by a unique MDP, and changes in state space, action space, transition, and reward function can result in completely different tasks. In MTRL, instead of solving a single MDP, we solve a bunch of MDPs from a task family using a universal policy $\pi_\theta(a|s)$. The first assumption for MTRL is to have a universal shared state space $\mathcal{S}$, and each task has a disjoint state space $S^\tau \subset S$, where $\tau \in \mathcal{T}$ is any task from the full task distribution. In this way, the policy would be able to recognize which task it is currently solving. Adding the one-hot encoding for task ID is a naive implementation of getting disjoint state space during experiments.

In the general MTRL setting, we don't have strict restrictions on which tasks are involved, but we assume that tasks in the full task distribution share some similarities. In real applications, depending on how a task is defined, we can divide it into **Multi-Goal MTRL** and **Multi-Skill MTRL**. For the previous one, the task set is defined by various "goals" in the same environment. The reward function r^τ is different for each goal, but the state and transition remain the same. Typical examples of this Multi-goal settings are locomotion tasks like *Cheetah-Velocity/Direction* [1] and all kinds of goal-conditioned manipulation

[1] By *Cheetah/Ant-Velocity/Direction*, we refer to the tasks that have the same dynamics as the standard

tasks [149]. For the latter one, besides changes in goals in the same environment, the task set also involves different environments that share similar dynamics (action space). This happens more in manipulation tasks where different environments train different skills of a robot, and one natural example is the Meta-World [239] benchmark, which includes multiple goal-conditioned manipulation tasks using the same robot arm. In this setting, the state space of different tasks changes across different skills since the robot is manipulating different objects (*c.f.* Figure 6.1). In both Multi-goal and Multi-skill settings, we have to form the set of MDPs into a universal Multi-task MDP and find a universal policy that works for all tasks. For multi-goal tasks, we need to append "goal" information into the state; for multi-skill tasks, we need to append "goal" (usually position) as well as "skill" (usually one-hot encoding). After getting state $\mathcal{S}^\tau$, the corresponding transition and reward P^τ, r^τ can be defined accordingly.

Challenges in MTRL and Beyond

Parameter-Sharing. Multi-task learning aims to learn a single model that can be applied to a set of different tasks. Sharing parameters allows us to take advantage of the similarities among tasks. However, the gaps between contents and difficulties of different tasks bring us the challenges on both which tasks should share the parameters and what parameters should be shared. Failure in the design may result in low success rate on certain tasks that could have been solved if trained separately. *This is a challenge in designing an effective algorithm for solving the MTRL task itself.*

Learning Beyond MTRL. Leveraging already acquired knowledge for learning new tasks has the potential to improve training efficiency [93, 92]. In pursuit of this beyond standard MTRL, we need to find what can be reused for the new tasks [176, 73, 7].

For example, in *continual learning*, where different tasks appear sequentially, the choice of sharing scheme is crucial since we need to maintain the high performance of previous tasks while training on the new ones. [93, 92]. *This is a challenge in moving beyond the standard MTRL setting towards a more practical scenario.*

6.4 Parameter-Compositional Multi-Task RL

Motivated by the challenges in training universal policies for multiple tasks discussed in Section 6.3, we will present a Parameter-Compositional approach to MTRL. The proposed approach is conceptually simple yet offers crucial flexibility for extension.

Formulation

This section describes how we formulate the parameter-compositional framework for MTRL.

locomotion tasks but with a goal of running at a specific velocity or in a specific direction.

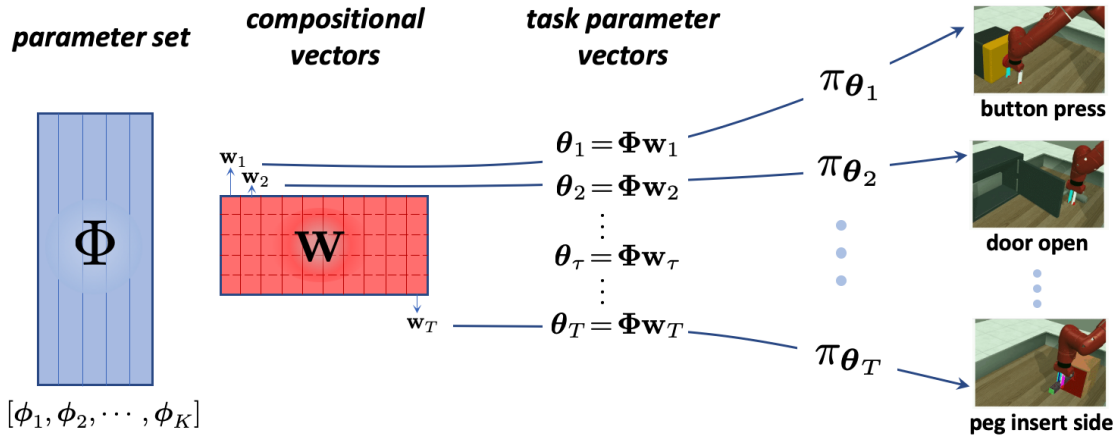


Figure 6.2: **Parameter-Compositional method (PaCo)** for multi-task reinforcement learning. In this framework, the network parameter vector $\boldsymbol{\theta}_\tau$ for a task τ is instantiated in a compositional form based on the *shared* base parameter set $\boldsymbol{\Phi}$ and the *task-specific* compositional vector $\mathbf{w}_\tau$. Then, the networks are used in the standard way for generating actions or computing the loss [66]. During training, $\boldsymbol{\Phi}$ will be impacted by all the task losses, while $\mathbf{w}_\tau$ is impacted by the corresponding task loss only.

Given a task $\tau \sim \mathcal{T}$, where $\mathcal{T}$ denotes the set of tasks with $|\mathcal{T}| = T$, we use $\boldsymbol{\theta}_\tau \in \mathbb{R}^n$ to denote the vector of *all the trainable parameters* of the model (*i.e.*, policy and critic networks) for task τ. We employ the following decomposition for the *task parameter vector* $\boldsymbol{\theta}_\tau$:

$$\boldsymbol{\theta}_\tau = \boldsymbol{\Phi} \mathbf{w}_\tau, \tag{6.1}$$

where $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \cdots, \boldsymbol{\phi}_i, \cdots, \boldsymbol{\phi}_K] \in \mathbb{R}^{n \times K}$ denotes a matrix formed by a set of K parameter vectors $\{\boldsymbol{\phi}_i\}_{i=1}^K$ (referred to as *parameter set*, which is also overloaded for referring to $\boldsymbol{\Phi}$), each of which has the same dimensionality as $\boldsymbol{\theta}_\tau$, *i.e.*, $\boldsymbol{\phi}_i \in \mathbb{R}^n$. $\mathbf{w}_\tau \in \mathbb{R}^K$ is a *compositional vector*, which is implemented as a trainable embedding vector for the task index τ. We refer to a model with parameters in the form of Eqn. (6.1) as a *parameter-compositional* model.

In the presence of a single task, the decomposition in Eqn. (6.1) brings no additional benefits, as it is essentially equivalent to the standard way of parameterizing the model. However, when faced with multiple tasks, as in the MTRL setting considered in this work, the decomposition in Eqn. (6.1) offers opportunities for tackling the challenges posed by the MTRL setting. More concretely, since Eqn.(6.1) decomposes the parameters to two parts: *i)* task-agnostic $\boldsymbol{\Phi}$ and *ii)* task-aware $\mathbf{w}_\tau$, we can share the task-agnostic $\boldsymbol{\Phi}$ across all the tasks while still ensure task awareness via $\mathbf{w}_\tau$, leading to:

$$\begin{aligned} [\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_\tau, \cdots, \boldsymbol{\theta}_T] &= \boldsymbol{\Phi}[\mathbf{w}_1, \cdots, \mathbf{w}_\tau, \cdots \mathbf{w}_T] \\ \boldsymbol{\Theta} &= \boldsymbol{\Phi} \mathbf{W}. \end{aligned} \tag{6.2}$$

For MTRL, let $J_\tau(\boldsymbol{\theta})$ denote the summation of both actor and critic losses implemented in the same way as in SAC [66] for task τ, the multi-task loss is defined as the summation of individual loss J_τ across tasks:

$$J_{\boldsymbol{\Theta}} \triangleq \sum_\tau J_\tau(\boldsymbol{\theta})$$

where $\boldsymbol{\Theta}$ denotes the collection of all the trainable parameters of both actor and critic networks. Together with Eqn.(6.2), it can be observed that the multi-task loss $J_{\boldsymbol{\Theta}}$ contributes to the learning of the model parameters in two ways:

- $\partial J_{\boldsymbol{\Theta}}/\partial \boldsymbol{\Phi} = \sum_\tau \partial J_\tau/\partial \boldsymbol{\Phi}$: all the T tasks will contribute to the learning of the shared parameter set $\boldsymbol{\Phi}$;
- $\partial J_{\boldsymbol{\Theta}}/\partial \mathbf{W} = \sum_\tau \partial J_\tau/\partial \mathbf{w}_\tau$: each task loss J_τ will only impact its own task specific compositional vector $\mathbf{w}_\tau$.

Furthermore, because of the clear separation between task-specific and task-agnostic information, the compositional form of parameters as in Eqn. (6.2) offers opportunities to naturally handle several cases (*e.g.* continual learning, transfer learning, etc.) beyond standard MTRL. For example, in the case of transfer learning, we can transfer the task-agnostic parameter set $\boldsymbol{\Phi}$ to serve as the pre-trained policy basis for further learning. The PaCo framework is illustrated in Figure 6.2.

Stable Multi-Task Reinforcement Learning

One inherent challenge in MTRL is the interference during training among tasks due to parameter sharing. One consequence of this is that the failure of training on one task may adversely impact the training of other tasks [201, 238]. For example, it has been empirically observed that some task losses may explode during training on Meta-World [178], which will contribute a significant portion in updating the shared parameters because of their dominance. As a consequence, this will significantly impact the training of the other tasks through the shared parameters. To mitigate this issue, [178] adopted an empirical trick to stop and discard the whole training once this issue is spotted.[2] Here, we show that in PaCo, there is a natural way to mitigate this issue without resorting to more expensive [238] or ad-hoc schemes [178].

More specifically, once a task loss J_η surpasses some threshold ϵ, because of the clear separation of task-specific parameters $\mathbf{w}_\eta$ from shared parameters $\boldsymbol{\Phi}$, we can straightforwardly mask out J_η from the total loss J to avoid its adverse impacts on others, which will essentially freeze $\mathbf{w}_\eta$ and contribute no gradients to $\boldsymbol{\Phi}$. We refer to this as the *Freeze*.[3] This

[2] https://github.com/facebookresearch/mtrl/blob/eea3c99cc116e0fadc41815d0e7823349fcc0bf4/mtrl/agent/sac.py#L322

[3] Note that in this case, the compositional weight for task τ will not be updated due to the mask-out of the loss. However, there are chances that the update on the parameter groups $\boldsymbol{\Phi}$ from other tasks changes the task parameter $\boldsymbol{\theta}_\tau$ and bring the task loss below the threshold, which could "turn on" the training on task τ again.

Algorithm 2 Parameter-Compositional MTRL (PaCo)

Input: param-set size K, loss threshold ϵ, learning rate λ
while termination condition is not satisfied **do**
  $\boldsymbol{\theta}_\tau = \boldsymbol{\Phi}\mathbf{w}_\tau$ ▷ compose task parameter vector
  $J_\tau \leftarrow J_\tau(\boldsymbol{\theta})$ ▷ loss (actor+critic as in SAC) across tasks
  (*Freeze / Reset Step 1*) $J_\eta \leftarrow 0$ if $J_\eta > \epsilon$
  $J_{\boldsymbol{\Theta}} \leftarrow \sum_\tau J_\tau$ ▷ calculate multi-task loss
  $\boldsymbol{\Phi} \leftarrow \boldsymbol{\Phi} - \lambda \nabla_{\boldsymbol{\Phi}} J_{\boldsymbol{\Theta}}$ ▷ parameter set update
  for each task τ **do**
    $\mathbf{w}_\tau \leftarrow \mathbf{w}_\tau - \lambda \nabla_{\mathbf{w}_\tau} J_\tau(\mathbf{w}_\tau)$ ▷ composition param update
  end for
  (*Reset Step 2*) $\mathbf{w}_\eta \leftarrow$ Eqn.(6.3) if $J_\eta > \epsilon$
end while

is a conservative scheme that largely stabilizes the training but can compromise the overall performance of the final policy (*c.f.* Table 6.2) as it has reduced the opportunity for learning on the masked-out tasks.

Because of the compositional nature of the PaCo model, we can actually do better than pure *Freeze*. This can be achieved by re-initializing $\mathbf{w}_\eta$ without impacting parameters of all others, in addition to loss mask out as in *Reset*, and then keep training as normal. One way is to re-initialize $\mathbf{w}_\eta$ as:

$$\mathbf{w}_\eta = \sum_{j \in \mathcal{V}} \beta_j \mathbf{w}_j, \quad \boldsymbol{\beta} = [\beta_1, \beta_2, \cdots] \sim \Delta^{|\mathcal{V}|-1} \tag{6.3}$$

where $\mathcal{V} \triangleq \{j | J_j \leq \epsilon\}$, and $\boldsymbol{\beta}$ is uniformly sampled from a unit $|\mathcal{V}|-1$-simplex $\Delta^{|\mathcal{V}|-1}$. We refer to this scheme as *Reset*. Compared with *Freeze*, which essentially stops training on the problematic task, *Reset* offers an opportunity for further learning on that task. It is worthwhile to point out that the ability to use *Reset*-like schemes is a unique feature of PaCo due to its clear separation between task-agnostic and task-specific parameters. Previous methods such as Soft Modularization [231] and CARE [178] cannot employ this due to the lack of clear decomposition between the two parts. Empirical results show that this can improve the training and lead to better model performance (*c.f.* Table 6.2). The overall procedure of PaCo is presented in Algorithm 2.

Unified Perspective on Some Existing Methods

Apart from the interesting compositional form and the features of PaCo, it also provides a unified perspective on viewing some existing methods. Using this formulation, we are able to re-derive some existing methods with specific instantiations of $\boldsymbol{\Phi}$ and $\mathbf{w}$.

- **Single-Task Model**: if set $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, \boldsymbol{\phi}_2 \cdots]$ and $\mathbf{w}_\tau$ as a one-hot task-id vector, this esse ially instantiates a single-task model, *i.e.* each task has its dedicated parameters.

- **Multi-Task Model**: if we set $\mathbf{\Phi} = [\boldsymbol{\phi}] \in \mathbb{R}^{n \times 1}$, $\mathbf{w}_1$=$\mathbf{w}_2$=...=1, then all the tasks share the same parameter vector $\theta^\tau = \boldsymbol{\phi}$. By taking the state and the task-id as input, we have the multi-task model.

- **Multi-Head Multi-Task Model**: by setting $\mathbf{\Phi}$ as follows:

$$\mathbf{\Phi} = \begin{bmatrix} \boldsymbol{\phi}' & \boldsymbol{\phi}' & \cdots & \boldsymbol{\phi}' & \cdots & \boldsymbol{\phi}' \\ \boldsymbol{\psi}_1 & \boldsymbol{\psi}_2 & \cdots & \boldsymbol{\psi}_\tau & \cdots & \boldsymbol{\psi}_K \end{bmatrix} \in \mathbb{R}^{n \times K}$$

  where $\boldsymbol{\psi}_\tau$ is the sub-parameter-vector of the output layer for task τ. Setting $\mathbf{w}_\tau$ as a one-hot task-id vector, we recover the multi-head model for MTRL, where all the tasks share the same trunk network parameterized by $\boldsymbol{\phi}'$ with independent head $\boldsymbol{\psi}^\tau$ for each task τ.

- **Soft-Modularization** [231] is the case where $\mathbf{\Phi}$ is in a specially structured form, with the combination done at each level with a "per-level" soft combination vector $\boldsymbol{z}(s, \tau)$ conditioned on current state s and task-id τ

$$\boldsymbol{\theta}_\tau = \begin{bmatrix} [\boldsymbol{\phi}_1^1 & \boldsymbol{\phi}_2^1 & \cdots & \boldsymbol{\phi}_K^1]\boldsymbol{z}^1(s,\tau) \\ & \vdots & & \\ [\boldsymbol{\phi}_1^m & \boldsymbol{\phi}_2^m & \cdots & \boldsymbol{\phi}_K^m]\boldsymbol{z}^m(s,\tau) \end{bmatrix}.$$

A difference is that Soft-Modularization [231] applies the combination on the activation instead of parameters. Nevertheless, the dependency of the combination vector $\boldsymbol{z}(s, \tau)$ on state s makes it diffuse task relevant and task agnostic information together, therefore all the parameters are entangled with state information and is less flexible in some cases, *e.g.*, continuing training on new tasks. Also, an operation like *Reset* is inapplicable to Soft-Modularization [231] because of the mixed role of $\boldsymbol{z}$ on state s and task τ.

6.5 Related Works

Multi-Task Learning. Multi-task learning is one of the classical paradigms for learning in the presence of multiple potentially related tasks [15]. It holds the promise that the joint learning of multiple tasks with a proper way of information sharing can make the learning effective. It has been extensively investigated from different perspectives [89, 102, 172, 177, 249, 200, 2, 118] and been applied in many different fields, including computer vision [245, 91, 120, 183], natural language processing [229, 182] and robotics [88, 4].

Multi-Task Reinforcement Learning. The idea of multi-task learning has also been explored in MTRL, with a similar objective of improving the performance of single-task RL by exploiting the similarities between different tasks. Many different approaches have been proposed in the literature [13, 19, 74, 35, 239, 147, 231, 178, 230, 166]. One of the

[3]To highlight the core algorithm, we have omitted the steps that are identical to standard SAC [66], including environmental unroll, temperature tuning, and target critic update.

most straight-forward approaches to MTRL is to formulate the multi-task model as a task-conditional one [239], as commonly used in goal-conditional RL [149] and visual-language grounding [147]. Although simple and has shown some success in certain cases, one inherent limitation is that it is more vulnerable to negative interferences among tasks because of the complete sharing of network parameters. [13] proposes an approach by assuming the functional approximator for each task is a linear combination of a set of shared feature vectors and then exploits the similarities among different tasks by employing a structured sparse penalty over the combination matrix. [35] utilizes a mix-and-match design of the model to facilitate transferring between tasks and robots. [31] leverages the shared knowledge between multiple tasks by using a shared network followed by multiple task-specific heads. [231] further extends these approaches by softly sharing features (activations) from a base network among tasks by generating the combination weight with an additional modularization network taking both state and task-id as input. Since the base and modularization networks take state and task information as input, there is no clear separation between task-agnostic and task-specific parts. This limits its potential on tasks such as continual learning of a novel task. Differently, PaCo explores a compositional structure in the *parameter space* [150, 228] instead of in feature/activation space [13, 231], and does so in a way such that the task-agnostic and task-specific parts are decomposed. This not only enables natural schemes for stabilizing and improving MTRL training (*c.f.* Sec.6.6), but also facilitates the expansion on the applicability of the method beyond the standard MTRL setting (*c.f.* Sec.6.6).

Conflicting Gradients in Multi-Task Learning. Because of parameter sharing for multiple tasks (thus multiple task losses), the shared parameters are impacted by the gradients from all the task losses. Whenever the gradients are not consistent with each other, there will be conflicts in updating the shared parameter. This issue of conflicting gradients is a general problem that is present in general multi-task learning [238, 223]. [201] bypassed the conflicting gradient issue by discarding parameter sharing but instead distilling each task policy into a centralized policy. [178] alleviates the negative effects of interference by deciding which information should be shared across tasks, using context-based attention over a mixture of state encoders. This demonstrates the benefits of a task-grouping mechanism but requires additional context information. There are also approaches to mitigating the interferences by balancing the multiple tasks from the perspective of loss [74] or gradient [24]. [238] proposes to address the conflicts by gradient projection, which could be less reliable in the case where gradients are noisy, as is the case in RL.

6.6 Experiments

We now empirically test the performance of our Parameter-Compositional Multi-Task RL framework on the Meta-World benchmark [239]. Meta-World benchmark is a robotic environment consisting a number of distinct manipulation tasks (*c.f.* Figure 6.1). Each task itself is a goal-conditioned environment, and the state space of all the tasks has the same dimension. The action space of different tasks is exactly the same, but certain dimensions

in the state space represent different semantic meanings in different tasks (e.g., goal position or object position).

With experiments performed on Meta-World, we would like to answer: ***(i)*** Can PaCo reach state-of-the-art performance in the multi-goal, multi-skill RL setting? ***(ii)*** What are the benefits of using interpolation of parameter sets? What does the policy subspace learn? ***(iii)*** Can the learned policy subspace benefit learning of new tasks beyond the standard MTRL setting?

PaCo Quantitative Results

Benchmarks. In the original Meta-World benchmark [239], each manipulation task is configured with a fixed goal. Therefore, the learned policies are not goal-conditioned ones as it cannot generalize to a task of the same type with different goals. This setting is easier, but more restrictive and less realistic in robotic learning [231]. Following [231], we extend all the tasks in the benchmark to a random goal setting.

Baselines. We compare against *(i)* **Multi-task SAC**: extended SAC [66] for MTRL with one-hot task encoding; *(ii)* **Multi-Head SAC**: SAC with shared a network apart from the output heads, which are independent for each task; *(iii)* **SAC+FiLM**: the task-conditional policy is implemented with the FiLM module [147] on top of SAC; *(iv)* **PCGrad** [238]: a representative method for handling conflicting gradients during multi-task learning via gradient projection during optimization; *(v)* **Soft-Module** [231]: which learns a routing network that guides the soft combination of modules (activations) for each task; *(vi)* **CARE** [178]: a recent method that achieves the state-of-the-art performance on Meta-World benchmark by leveraging additional task-relevant metadata for state representation.[4]

Training Settings. In experiments, the convergence performance is related to the number of parallel environments for training and the number of tasks in the environments. There is a balance between the number of iterations and the number of roll-out samples. Referring to the settings introduced in [178], we use 10 parallel environments for the MT10-rand setting and 20 million environment steps (2 million for each task) during training.

Evaluation Metrics. The evaluation metric for the learned universal policy for all tasks is based on the success rate of the policy for all the tasks. For Meta-World benchmarks, we evaluate each skill with 10 episodes of different sampled goals using the final policy. The success rate is then averaged across all the skills. For each experiment, we train all methods with 3 random seeds. The randomness in the MTRL training is unpredictable; some methods may converge to a higher success rate right after 20M steps, and some may drop if the training continues. Instead of picking the maximum evaluation success rate across training, we use the policy at 20M environment steps for fair evaluation.

PaCo Variants. The most important hyper-parameter we need to determine for the PaCo framework is the number of parameters set m. For a group of similar skills, we may be

[4]Note that the experiments reported in the works mentioned above are implemented and evaluated on Meta-World-V1 and/or with the fixed-goal setting. We adapt these methods and experiment on Meta-World-V2 for experiments.

able to reach a high success rate with one parameter set $K=1$. For skills with high variance, for example, "reach" and "pick-place" (the easiest and hardest skills), sharing parameters is hard. Overall, the increasing in parameter group number K grows with the task number n but essentially at a lower speed. For the benchmark MT10-rand, which contains 10 different manipulation skills, we set $K=3, 5, 8$ as different variations of our method.

Table 6.1: Results on Meta-World [239] MT10 with random goals (MT10-rand).

Methods	Success Rate (%) (mean ± std)
Multi-Task SAC [239]	66.7 ± 8.1
Multi-Head SAC [239]	59.3 ± 9.0
SAC + FiLM [147]	58.4 ± 1.2
PCGrad [238]	63.0 ± 6.0
Soft-Module [231]	64.3 ± 5.7
CARE [178]	79.7 ± 9.3
PaCo (Ours)	**85.0 ± 5.0**

From Table 6.1, we observe an improvement in the average success rate and the stability of training on the MT10-rand benchmark compared to baseline methods. Compared to the previous state-of-the-art method CARE [178], we achieve a better performance without additional task information. For variations in the parameter set, we find $K = 5$ gives the best performance. Results are shown in Table 6.2, more parameter groups don't always help in MTRL training.

Stable MTRL Training

In RL training, failure or gradient explosion may occur due to weight initialization, bad exploration, and many other random factors. This is even worse for the MTRL setting, especially on those parameter-sharing models, since the loss and gradient explosion on certain tasks would influence tasks that share the same policy parameters. In PaCo, we use the *Freeze* and *Reset* schemes introduced in Section 6.4 to avoid the influence of extreme losses of certain tasks. We perform an ablation study on them to analyze their roles in stabilizing the MTRL training. PaCo-*Vanilla* refers to the pure compositional structure setting. PaCo-*Freeze* refers to the setting with the freezing scheme on tasks losses that exceed the loss threshold, PaCo-*Reset* refers to the scheme that, in addition to *Freeze*, also re-initializes the compositional vectors of tasks with extreme losses. In order to show the stability precisely, we don't early-stop the training process even if the loss of some task already explodes.

The final success rates are shown in Table 6.2. It is observed that although PaCo-*Vanilla* has higher final performance, it has a larger variation and is less stable during training, which can lead to exploding loss in some cases. PaCo-*Freeze* variant can prevent the training from collapsing but can potentially compromise the performance because of the reduced

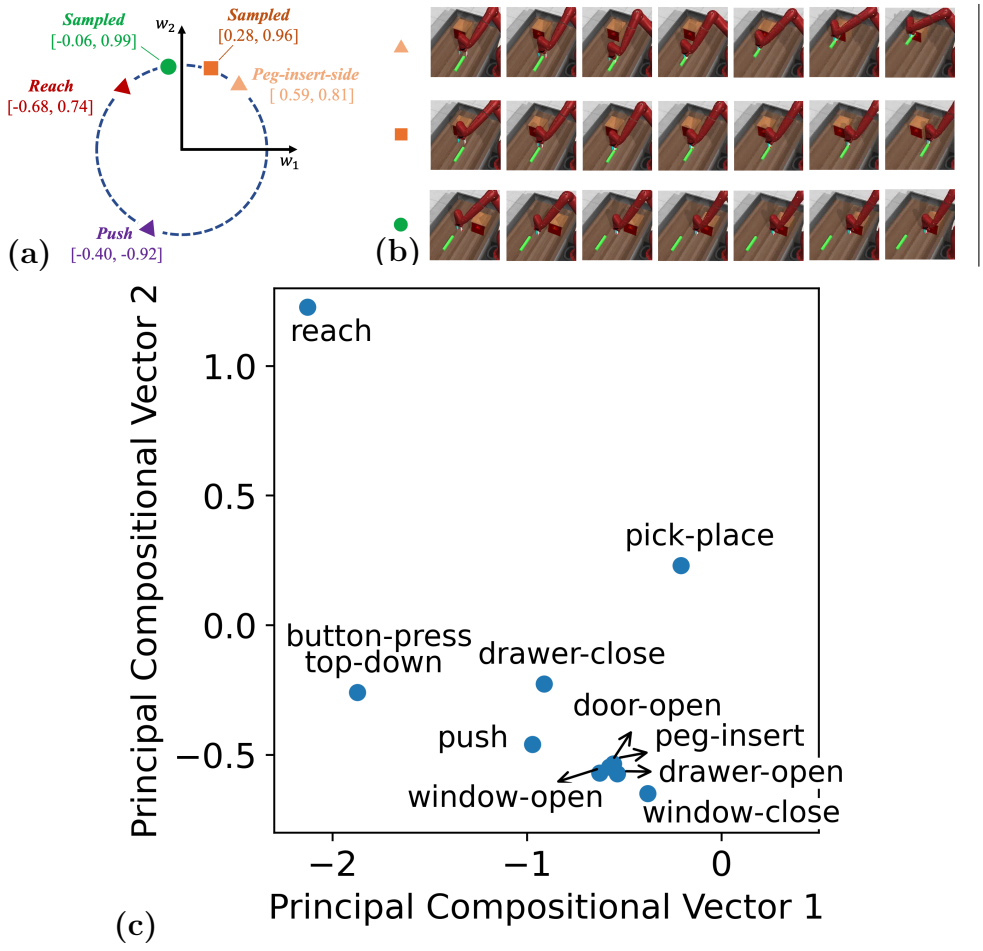


Figure 6.3: **(a)** Compositional vectors of skills *reach, push, peg-inset-side* learned on the restricted unit circle in 2D policy subspace. **(b)** Trained policies (*Reach*, *Push*, *Peg-insert-side*) and interpolated policies between skill *Reach* and *Peg-insert-side*. Visualization is done on the peg-insert-side task. One interpolated policy close to *Peg-insert-side*-skill shows the skill of picking up the peg. Another interpolated policy close to *Reach*-skill shows the insert-like skill without picking up the peg, which also resembles a reaching-skill towards the hole. **(c)** 2D PCA projection of the ten 5D compositional vectors $\{\mathbf{w}_\tau\}$ learned on MT10-rand tasks.

opportunity of learning on the masked-out tasks. Another reason for its lower success rate than PaCo-*Vanilla* is that the model for evaluation is obtained at 20M environment steps, which accidentally coincides with a peak of the fluctuating performance for PaCo-*Freeze*. The PaCo-*Reset* variant improves both the average success rate and the stability of MTRL training.

Table 6.2: Ablations with PaCo Variations on MT10-rand.

Variations	Success Rate (%)
PaCo-*Vanilla*	80.0 ± 8.1
PaCo-*Freeze*	76.6 ± 12.1
PaCo-*Reset* (K=5)	85.0 ± 5.0
PaCo (K=3)	74.4 ± 5.1
PaCo (K=8)	81.0 ± 8.4

Qualitative Understanding of Parameter Groups

Skill Interpolation. To better understand the trained compositional vectors $\{\mathbf{w}_\tau\}$ for each task, we first demonstrate the result of PaCo on a representative 3-task combinations using 2 parameter groups. We include three tasks: *reach*, *push*, and *peg-insert-side* with 50 goals for each in the task set. Using the number of roll-out samples before convergence as a metric for each single task, this combination actually covers simple, medium, and hard skills in the Meta-World. It's hard to learn this task combination using one model like MT-SAC or Multi-Head SAC, usually the success rate for hardest task remains low. After training, PaCo reaches an average success rate of 100% across all skills.

In Figure 6.3a, we show the position of the skill-specific policies in the compositional vector 2D plane. In order to get better visualization on subspace, we added a *normalize* activation after the weights, e.g., all the policies lie on the unit circle in W space. This actually shows: *(i)* We are not learning all the task with one policy, different tasks use different parameters. *(ii)* We are able to find policies for three completely different manipulation skills on this 1D low-dimensional manifold in the policy parameter space. This satisfies the purpose of multi-task learning in saving parameters. The learned two parameter groups are "eigen" policies although they don't describe an independent skill. As a trial, We sampled an interpolation point between the task *reach* and *peg-insert-side* policy. The resulting policy performance is demonstrated in Figure 6.3b. In the peg-insert-side environment, it tried to do the "insert" action to the correct position but is not particularly impacted by the peg; therefore is an "insert" skill that is not particularly customized for the particular *peg-insert-side* task.

Emerged Task Similarities. We visualize the learned compositional vectors in a 2D space via Principal Component Analysis (PCA) in Figure 6.3c, for MT10-rand. The distance in the compositional space actually indicates the similarity between skills. There are several interesting observations:

- Very distinct skills could lie in different part of the skill space, *e.g. reach v.s.* others;
- Skills that are literally related appear to be close, *e.g. window-open v.s door-open*, *window-open v.s. window-close* in Figure 6.3c. This is in fact the core idea explored by CARE [178]: incorporating task relation extracted from sentence-based task descriptions into the model.

Here we observed that meaningful task relations emerged purely from learning without additional task-related meta-data as in CARE [178];

- Another interesting observation is that some skills that are not literally related also appear to be close in the skill space learned by PaCo, *e.g. peg-insert-side v.s. window-open/window-close/door-open/drawer-open*. Although literally distinct, *peg-insert-side* and the other skills mentioned above are related from the perspective of behavior, *i.e.*, first interacting with an object (*e.g.*, *peg/window/door*), and then taking a trajectory of motions to accomplish the subsequent operation, (*e.g. insert/window/door*). Therefore, these literally unrelated skills are inherently semantically related at the skill level. This is something that could be useful but is not able to be leveraged by CARE [178], explaining one of the possible reasons why PaCo is more effective.

Additional Factors in MTRL Training

Ratio of task samples. In the 10 manipulation tasks involved in MT10, there are certain tasks that are harder than others and take more samples to train the optimal policy. Comparing the roll-out samples required to train a successful policy, the *peg-insert-side* and *pick-place* need significantly more samples than others and have worse stability. However, in the MTRL setting, without the information on the difficulty of tasks provided in advance, we can only uniformly roll out samples for each task. In experiments, by manually setting the ratio (larger for hard tasks) of tasks based on difficulties, we can reach the similar performance of PaCo reported in Table 6.1 using only 10-15M samples.

Table 6.3: Compositional structure variation results.

Compositional Variations		Success Rate (%)
Output-Layer	*Actor-only*	60.0 ± 0.0
Full-Net	*Actor-only*	73.3 ± 5.8
	AC-shared (**PaCo**)	85.0 ± 5.0

Compositional structure variations. There are several possible variations in PaCo by employing compositional structure to all (actor and critic) networks (*AC-shared*) or only to the parameters of the actor-network (*Actor-only*). Also, we can choose to apply the structure to all the layers in the network (*Full-Net*) or only to the output layer (*Output-Layer*), which is architecturally similar to Multi-Head SAC. The results in Table 6.3 show that when applied to the output layer only, the performance is comparable to Multi-Head SAC (*c.f.* Table 6.1), although with fewer parameters ($K = 5$) than Multi-Head SAC ($K = 10$). This shows that the compositional structure is also effective in the output layer up to the performance limit imposed by the network architecture. Using the compositional structure for the full networks of both actor and critic can further unleash its potential and give the best performance.

Initial Attempts: PaCo-based Continual Learning

Going beyond MTRL, another question we may ask in the application is how we benefit from the policy trained by PaCo when we meet new tasks. The unique property of a well-separated task-agnostic parameter set and task-specific compositional vector gives us the potential to use PaCo in a more challenging continual setting. The main reason for catastrophic forgetting in continual learning is that the training on new tasks modifies the policies of existing tasks. However, in our PaCo framework, if we can find the policies for new task $\tilde{\tau}$ in the existing policy subspace defined by $\mathbf{\Phi}$ with a new compositional vector $\mathbf{w}_{\tilde{\tau}}$, the forgetting problem can be avoided. With no change on $\mathbf{\Phi}$, we extend the existing parameters to a new task with no additional cost. In reality, there is no guarantee for the existence of such a policy; the relation between skills is quite important. However, in experiments, we do find successful extensions from an existing skill set to a new skill when the skills are similar. For instance, *reach, door-open, drawer-open* **to** *drawer-close*.

In practice, we can design a more general training scheme to learn the policy for a series of tasks. Given a parameter set $\mathbf{\Phi}$ with K parameter groups trained on N tasks, if we find the policy for new tasks in the policy subspace, we save the compositional vector for the new task. If we cannot find the policy in the subspace, we train the new tasks on a new parameter set $\tilde{\mathbf{\Phi}}$ and merge them into a subspace with a higher dimension. Verifying this property on larger skill sets is an interesting future direction and requires more complex experiment designs.

6.7 Chapter Summary

Based on a revisit to MTRL and its challenges both within and beyond its typical settings, we present PaCo, a simple parameter compositional approach, as a way to mitigate some of these challenges. The proposed approach has the benefit of clear separation between task-agnostic and task-specific components, which is not only useful for stabilizing and improving MTRL but also opens the door to transfer and continual learning naturally. Without resorting to more complicated design [231] or additional data [178], PaCo has demonstrated clear improvement over current state-of-the-art methods on standard benchmarks. Furthermore, we have also demonstrated its possibilities in continual learning.

Chapter 7

Utilizing LLM for Partial Observable Task Planning

7.1 Introduction

In the previous chapter, we introduced how we learn generalized skills from the novel compositional learning structure. However, even with learned skills, doing long-horizon tasks is still challenging for robots. In this chapter, we focus on solving the complexity of task planning with the help of Large Language Models.

Designing robots with the physical intelligence to perform open vocabulary tasks requires that robots be able to interpret tasks from an open set of instructions and execute them robustly while performing the required reasoning. One can argue that this could be the most challenging problem facing artificial intelligence (AI). However, designing such agents can truly revolutionize how robots would be integrated into our future society. Recently, large language models (LLMs) [143, 3, 206] are very impressive at solving tasks of different complexities [213, 1, 240, 111, 115]. Large language models can help understand the tasks and decompose them into a sequence of actions, reward functions, or goals for policy, given appropriate prompts and training data. Motivated by these developments, we present a problem of interactive planning in uncertain environments where a robot may not have complete information to perform the task. In these tasks, the robot needs to interact with its environment and collect additional information to complete the task.

Partial observability and uncertainty are the norm, rather than the exception, in the real world. For example, consider task T2 shown in Figure 7.1, where a robot needs to understand how it can gather information to identify the empty cup and then throw it in the bin. Unlike the tasks with complete information, it would be challenging to design a sequence of skills or a suitable reward function that can solve this task. This problem can be formulated as a Partially Observable Markov Decision Process (POMDP)[87]. However, solving POMDPs could be computationally intractable. It requires reasoning in the belief state of the problem and does not scale well with the dimensionality of the problem. Prior

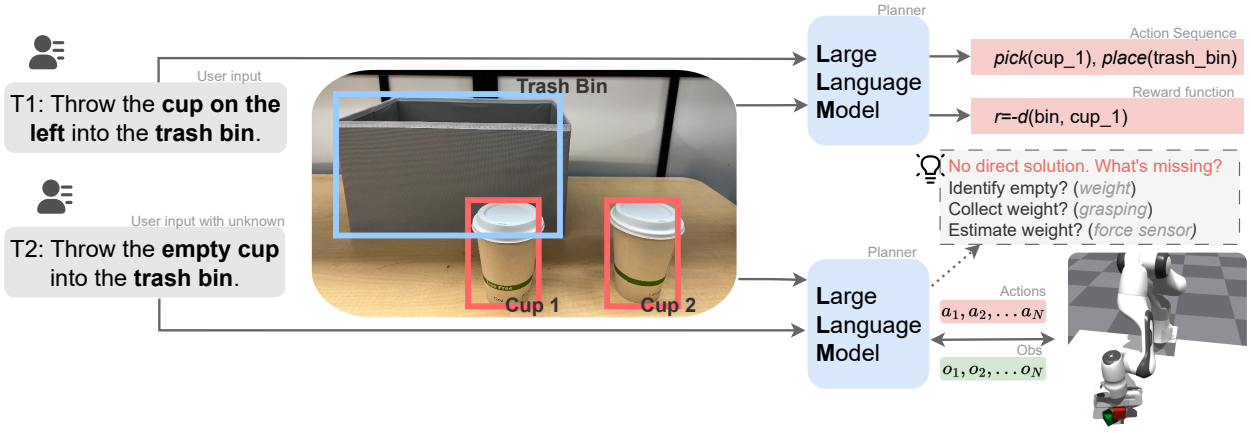


Figure 7.1: An example task where the uncertainty is present in the content of the cups. For task T1, the robot is asked to throw the cup on the left into the bin. An LLM agent can generate feasible action sequences for the robot to perform the task. When asked to throw the empty cup, the agent cannot reason which cup is empty based on current information. It needs to interact with the cups and use feedback from observations (e.g., force sensor reading) to identify the empty cup.

work on using LLMs for robotic tasks has demonstrated good reasoning capability of LLMs as well as mapping of the reasoning to robot actions [1, 77, 42]. Inspired by these advancements, we believe that we can leverage the reasoning and *chain-of-thoughts*(CoT) capability of LLMs to solve partially observable tasks while interacting with the environment. What makes this challenging for current LLMs is the requirement to understand real robot observations from different modalities and use them for task planning.

Most of the prior works using LLMs in robotics focused on step-wise scene and task understanding, making full use of the current available modalities to infer the optimal action and/or reward [1, 77, 240, 42]. In this work, we focus on performing interactive planning under cases of partial observability. This requires planning to aggregate information from the environment, reasoning about the correct state of the system, and updating the state estimates based on the sensor observations collected by the robot. Furthermore, we also try to understand how well a fine-tuned smaller model like Llama2-7B [206] performs in comparison with a pre-trained LLM like GPT-4. The smaller models are generally desirable for practical reasons but it could be challenging to distill the reasoning capability of models like GPT-4 for complex robotic tasks discussed in this paper. To understand this, we propose an instruction data generation pipeline following the self-instruction [222] scheme to understand the limitations of smaller models and potential ways to overcome them.

In summary, this chapter aims to:

- Introduce the Large Language Model for Partially Observable Task Planning(LLM-POP) framework to interactively plan with uncertainties. We demonstrate the frame-

work in simulation and real-world environments.

- Compare the performance of pre-trained LLM and fine-tuned smaller models in the proposed framework for partially observable tasks.

7.2 Interactive LLM Planning with Uncertainties

The objective of the proposed framework is to perform long-horizon robotic tasks in the presence of various kinds of uncertainties using LLMs. These tasks require a closed-loop, interactive planning where the robot should be able to collect useful observations from the environment and then make optimal decisions. An example of such a task is illustrated in Figure 7.1, where the robot's task is to throw the empty cup into the bin. However, there exists uncertainty in the contents of the cups, and therefore, this information needs to be obtained by sensorimotor operations and provided as the feedback to the LLM. For clarity of presentation, this section delves into formulating the underlying problem using the notion of POMDPs. We then elaborate on the pivotal role of LLMs in the interactive planning framework.

Problem Formulation

Partial Observation setting

A POMDP is an extension of a traditional MDP that tackles decision-making scenarios where the agent lacks complete state information. A POMDP is defined by a tuple (S, A, P, R, Ω, O), with Ω as the observation set and O as the observation function. At each time step, the environment is in state $s \in S$. The agent takes action $a \in A$ and causes the environment to transit to s' accordingly to the transition function $P(s'|s, a)$. At the same time step, the agent gets an observation $o \in \Omega$ which depends on the current state of the environment $O(o|s')$. Unlike the policy function in MDP $\pi(a|s)$, which maps the underlying states to the actions, POMDP's policy $\pi(a|b)$ is a mapping from the belief states b to the actions. The belief state b is a probabilistic estimation of the full state s. The updated belief state b' after observing o is described by: $b'(s') = C \cdot O(o|s') \sum_{s \in S} P(s'|s, a)$ where C is a normalizing constant.

We also want the proposed framework to be generalizable to a variety of tasks. For different tasks, τ, the information required to make decisions can differ. This adds additional complexity since now the LLM has to reason about a generalizable state space S. In the open-vocabulary robotics task scenarios, the robot observations are determined by on-board sensors. Not all information about the environment is relevant to the task; some of them can be directly extracted from observations, while some are unknown and require exploration. Thus, we end up getting task-dependent belief state b^τ, and the task-related states s^τ for task τ. Both finding the necessary state abstraction for different tasks and finding the optimal policy π under the task-specific MDP are important in this task-dependent POMDP setting.

Action space of robots

For long-horizon tasks, using a pre-trained set of parameterized skills as action space is a common choice. In this paper, we use a set of parameterized skills like {**pick, place, reach, reset**}. All these skills can be performed using robot observations, and thus, we do not consider partial observability during robot skills execution. It is noted that we do not consider continuous sensory feedback during skill execution– however, that could be incorporated by training skills using RL.

Uncertainties in Tasks

The uncertainty in decision-making in the tasks we test mainly arises from two aspects:
Environmental Uncertainty: These uncertainties arise in the POMDP settings due to the agent's lack of complete environmental knowledge. For example, physical properties of the objects that cannot be directly observed. The uncertainties in the belief b^τ can be reduced with certain observations. This is a major challenge we target to solve in this paper.
Skill Execution Uncertainty: Even with a well-defined plan, the actual execution of actions on robots might not always lead to the expected outcome. This can be mainly attributed to the difference between the transition functions P, P_{real} of the designed and real system as well as unexpected disturbances during execution.

With the challenges explained above, we propose a framework where LLMs are used as policy as well as for state abstraction for the underlying POMDP.

Language-based Planners

Based on the problems described in the previous section, we propose to use an LLM to play a multifaceted role in the interactive planning process:
LLM for State Abstraction: Given the environment description and sensor observations, LLM needs to analyze the available information and abstract sufficient statistics (or the appropriate state) to solve the task. Furthermore, based on the current observations, it needs to reason about what is uncertain. It needs to update its belief based on the observations when prompted with historical information.
LLM as Policy: Given the observation and action space, LLM needs to plan actions that gather environmental information to mitigate the uncertainty and update the agent's belief state. The LLM-based policy is also expected to generate the optimal plan to maximize the reward based on the task description with minimal steps. Also, since we use open-loop parameterized skills for the robot, the LLM is also used to provide feedback to the robot in cases of failure in the execution of these skills. This feedback needs to be provided in a way that is still executable by the robot.

We use LLM to reason about these problems during task execution. It is noted that actions in the POMDP setting is conditioned on new observations and updated beliefs. There are a few additional challenges when using LLM as a closed-loop policy for tasks

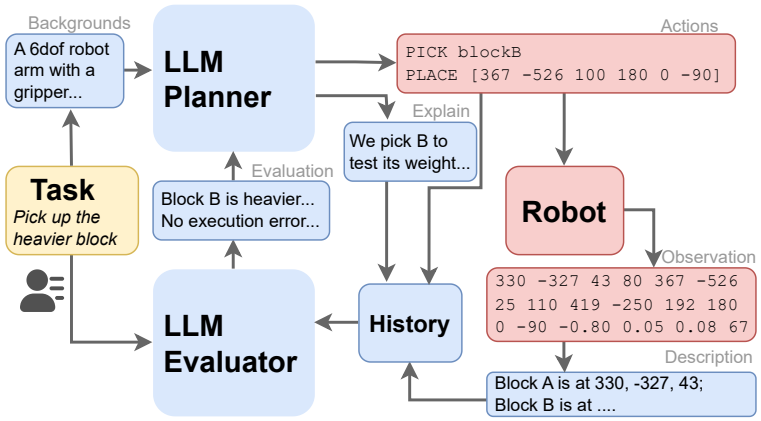


Figure 7.2: An example showing how the framework works during solving the task "Pick up the heavier block". The LLM planner outputs an executable action sequence to the robot. The robot executes the action, and the observation description and action pair are added into the history buffer. The LLM evaluator analyzes the historical information and outputs the updated information to the planner to generate a new plan.

with uncertainties that we consider in the paper. To update the belief state of the task, the LLM must understand the robot observations from different modalities (pose detections, force sensors, etc.). These data formats might be new to the LLM model and, thus, must be properly included in the prompt template for the LLM. Furthermore, the skills available to the robot are parameterized by continuous position and orientation coordinates, which might be challenging to reason about while performing robotic tasks. Similarly, the output of the language model needs to be executable by the robot; the response should be written in a template that the downstream controller can understand. In the next section, we will discuss how we use the LLMs to solve the interactive planning task.

7.3 LLM-POP: Interactive Planning

The proposed framework (LLM-POP) for interactive planning is illustrated in Figure 7.2. As introduced in the problem formulation, the language-based policy in our framework has multiple tasks to do in the planning loop. At each step, the input to the language model contains the **task description** from a user, **the current observation** from the robot, and the **historical action and observation sequence** from previous steps. The model output includes an executable **sequence of actions** and the **corresponding text explanation**. The robot will execute the actions provided by the policy output and return the observations for a next-round query of the LLM. The language model must finish the reasoning task and output the policies in the designed format. The task description is the only user-provided input during the planning process. In the following sections, we show how we use a pre-trained LLM (GPT-4) as well as a fine-tuned smaller model to serve as the planner and evaluator.

Prompt structure for GPT-4

Using powerful LLMs like GPT-4 as interactive planners relies on its strong chain-of-thought reasoning and in-context learning capability. Therefore, the prompt (input of a single round LLM query) to the LLM requires careful design to ensure it can generalize to robotics tasks and avoid hallucination (generating actions in wrong formats or not executable for the robot) in responses.

As shown in Figure 7.3a, the prompt template for the *planner* consists of the following parts:

- *Environment description, action options, output rules*: Background information that help understand the task settings. This information is preset by the user and is constant throughout planning for different tasks.
- *Task description*: texts describing tasks from users. We assume the first two parts should provide enough information for the LLM to understand what's the missing information and what are actions that can collect the information.
- *Example outputs*: in-context examples for planning.
- *Current observation and historical information*: text-format descriptions of current observation and historic information. If the observation is poses and force, use vectors with explanations.

The explanation in output, together with the action sequence, will be included in historical information. This helps the LLM to understand the past actions it has performed and avoid reasoning about it again. Note that the LLM planner needs to specify the parameters in the actions based on its own understanding of the environment, task, and the action space description. For manipulation tasks, this includes location and orientation for the target pose.

As shown in Figure 7.2, along with the LLM *planner*, we also designed an LLM *evaluator* using a similar prompt structure. The evaluator also takes in the background information, task description, and history observations after executing past actions. It evaluates the task execution status and appends it to next-round prompting. As described in Section 7.2, the evaluator here will explicitly ask the LLM to finish the "state abstraction" (analyze what's the missing information), "belief update" in policy (analyze information from historical observations), and "correct execution errors"(identify failures from the history). Although it is possible to put all the requirements into the LLM planner, asking it to do all the analysis and make planning decisions in the response, we find decomposing this into two steps improves the reasoning results.

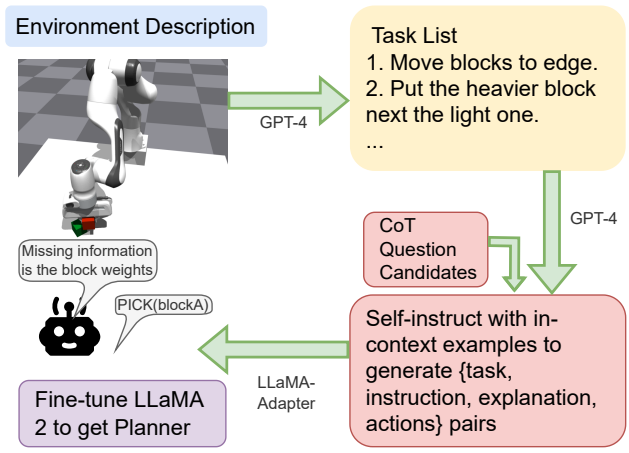
Fine-tuning a smaller model as planner

Fine-tuning a language model, rather than directly querying a GPT-4, not only enables offline deployment but also holds distinct advantages in the context of interactive planning.

[Environment*]
A Franka robot is placed in front of a table with cubes on top...
[Task Description*]
Move the heavier blocks to the corner of the table.
[Action Options]
1) PICK <object>
2) PLACE <location>
3) RESET
[Output Examples**]
EXPLAIN To check the weight of blocks, we need to first...
PICK blockA\n PLACE [0.1 0.2 0.05 0 0 0 1]
[History and Current Observation*]
Round 1: block A at... force reading... after action PICK...
Current: block A at ..., force reading...
[Output Instructions and Rules**]
Think step by step...
Remember to follow the format of output:...

* same for evaluator ** adjust for evaluator

(a) Prompt template for GPT planner and evaluator. The task description is taken from user input; the others are pre-defined according to the environment and robot. Output rules will be adjusted for the evaluator.



(b) The training procedure of the fine-tuned LLMs as an interactive planner as described in Sec 7.3. During inference, questions come from the pre-defined CoT question set; inputs come from robot observation.

Figure 7.3

One prominent reason is the incorporation of multi-modality in the data. Our system doesn't solely rely on text descriptions but also utilizes the robot's observations. While these observations can theoretically be converted into text form, they constitute a novel data type that GPT-4 has not been trained on, thereby resulting in limited zero-shot generalizability. For example, in experiments using GPT-4, if poses in robot observations and action parameters are in different frames of reference, the LLM will have trouble transforming them. A second reason is the requirement of large contexts in the input. A direct query to GPT would necessitate the inclusion of environment settings and generation constraints at each instance, which is inefficient and cost-intensive. The difficulty of fine-tuning a smaller pre-trained LLM model mainly comes from two sides: 1) **Lack of data** for complex tasks. Most robotics data in the wild[216, 10, 48] has no partial observable tasks involved, and force-torque sensor data is usually not included since they are noisy and vary across robots. 2) Smaller models are **worse at reasoning tasks**, CoT is tied with larger models [224].

In order to get the required data to fine-tune a model as a planner in interactive planning under partial observation, we follow the procedure shown in Figure 7.3b, using self-instruct[222] to generate an instruction dataset and fine-tune a LLaMA2-7B[207] model. The full pipeline includes:

Task Generation: The description of the environment, robot, potential uncertainties, action options, and example tasks are provided to GPT-4 to generate a number of tasks that are feasible to solve. We encourage GPT-4 to make the task set diverse in difficulty.

Instruction Generation: The generated tasks are used to generate pairs of instructions and responses, following the self-instruct paradigm. The instruction includes task descriptions and questions, the input encompasses the robot's observations. The output generated by the model includes the same verbal explanations and actions as GPT-4 planners. We add format instructions to guarantee the "response" format.
CoT question designs: Finishing the state abstraction, belief update, and action planning in one query is hard for smaller models. Therefore, we create CoT questions[75] to ask *if missing information exists*, *how to collect information*, and *how to solve the task with fill information*. The planner will choose questions to ask based on binary options in response.
Integrating collected robot observations: For the pre-trained actions, we collect success trajectories of the robot finish the actions and use them as in-context reference examples in the *Instruction Generation* process.
Fine-tuning: For the fine-tuning process, we adopted the LLaMA-adapter [57]. This approach allows us to enhance the model's performance by leveraging a specifically curated dataset and fine-tuning it to our unique task generation and interactive planning scenario.

7.4 Experiments

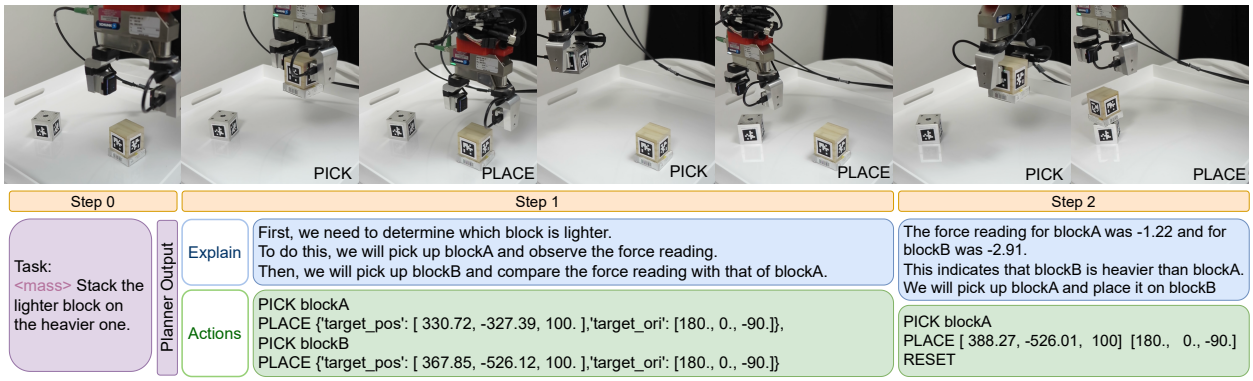


Figure 7.4: An example rollout of LLM-POP solving *T4:Stack the lighter block on the heavier one*. In the first step, the LLM planner figured out the plan to determine which block is lighter by picking and placing up and placing down both blocks. In the second step, the LLM evaluator figured out blockB is heavier and plans to place blockA on blockB. In the next round(now shown in the figure), the evaluator recognized the completion of the task.

The experiments aim to validate the proposed interactive planner and to answer the following three questions:

1. Is the proposed framework able to solve complex tasks with uncertainties?

2. Does the framework apply to sim and real robots with various observation/action spaces?

3. Can the fine-tuned LLM solve partial observable tasks? What are the gaps between GPT-4?

Experimental setup

Environment: LLM-POP is evaluated on a set of manipulation tasks in a tabletop block rearrangement environment. A robot arm with a parallel gripper is equipped with a pre-trained skill set of {Pick, Place, Reach, Reset}. Each scenario is initialized with identical-size blocks with randomized positions and orientations on the table.
Uncertainties: We introduce two uncertainties in this environment 1) *mass*: Density(mass) of the blocks is randomized. 2) *fix*: blocks are randomized to be fixed/movable on the table. We design a task set containing tasks at different difficulty levels (horizon length to solve the task) under uncertainty assumptions to evaluate the planner's performance.
Observations: The robot observations include the pose of the robot end effector, the pose of the blocks on the table, gripper opening positions, and force-torque (F/T) readings.
Pre-trained skills and parameters: *pick(object):* Pick up the specified *object* on the table. *place(pose):* Move the end effector to desired *pose* and open the gripper. *reach(pose):* Move the end effector to desired *pose*. *reset():* Reset the arm and gripper to the initial pose. where *object* is a text name string, *pose* is the position and orientation.
Evaluation metrics: How to evaluate the task success rate is non-trivial since the desired outcome of the tasks in Table 7.1 (e.g., the lighter block is on top of the heavier block) could also be achieved through an incomplete decision-making process (e.g., stack a block on top of the other one that by chance respects the right weight relationship.) For this reason, even the LLM evaluator proposed in Section 7.3 cannot confidently determine the success rate in the tasks, and we eventually relied on a manual check of each experiment.

For each task in the evaluation task set, we evaluate the success rate of finishing the task under ten random initializations on the positions and the uncertainties of blocks (5 for real robot settings since it's harder to randomize the uncertainties). Table 7.1 includes the evaluation tasks we use. Default LLM-POP uses GPT-4 for the planner and evaluator.

Table 7.1: Uncertainty and Task Description Table

Type	Task Descriptions
/	1. Stack one block onto another.
/	2. Move the blocks to the corners of the table.
mass	3. Pick up the heavier block.
mass	4. Stack the lighter block on the heavier.
fix	5. Pick up the movable block and put it at the table corner.
fix	6. Find the movable block and put it on the fixed block.

Simulation: Block manipulation with Pre-trained LLM

We first evaluate our approach for solving the tasks in Table 7.1 in a simulated robotic system in IsaacGym[128]. We use the *FrankaCubeStack* task as a template environment but change the blocks to the same size with random densities for *mass* uncertainty and randomly fix the block on the table for *fix* uncertainty. The action parameters for *place* include 3D target position and quaternion of the end effector. This is different from the block orientation quaternions in the observation, and explicitly including this (compared to using observation-action examples) in the prompt is essential for the LLM planner to get the correct action parameters and understand the observations.

We use two ablations: 1. GPT-3.5 as planner. 2. Remove the evaluator which explicitly asks for state abstraction and belief updates[1]. Evaluation results are shown in Table 7.2.

Table 7.2: Planner Success Rate on Evaluation Task Set

Model	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6
GPT-3.5(w/o E)	5/10	7/10	0/10	0/10	0/10	0/10
GPT-3.5	6/10	6/10	0/10	0/10	0/10	0/10
LLM-POP(w/o E)	10/10	10/10	9/10	4/10	6/10	4/10
LLM-POP	**10/10**	**10/10**	**10/10**	**8/10**	**7/10**	**8/10**
LLM-POP*	5/5	5/5	5/5	4/5	5/5	4/5
FT-Vanilla	4/10	2/10	0/10	0/10	0/10	0/10
FT-CoT	10/10	10/10	4/10	3/10	8/10	6/10

(w/o E): No evaluator. * Real-world experiment. FT: Fine-tuned Llama2

Compared to the GPT-4-based planner, GPT-3.5 is also able to reason correct action sequences for stacking tasks but can not always reason about the geometric (position and orientation) parameters. It can understand what's the missing information for tasks with uncertainty, but fails to generate multi-step plans to collect and update the information.

For the GPT-4 based planner, we observe an improvement in performance, especially on longer-horizon tasks with uncertainty when the evaluator is added to the pipeline. An example is shown in Figure 7.4. The LLM is asked to explicitly explain what is missing and how such information can be analyzed from historical observations. This enforces the LLM to perform "state abstraction" and "belief update" before planning. In experiments, the GPT-4 planner without the evaluator sometimes makes the wrong plan by repeating the same collecting actions even if it already collected sufficient information. As the history of observations grows longer, the chance that LLM makes wrong reasoning also increases. This is partially due to the long-text handling challenge for the current GPT-4 version, and decomposing the reasoning tasks into an evaluator helps improve the stability.

[1]To avoid the uncertainty from GPT versions, we use *gpt-4-0314* for all GPT-4 and *gpt-3.5-turbo-16k-0613* for GPT-3.5 usage.

Hardware: Real robot Block Manipulation with GPT

We implement the same version of our method on a MELFA Assista robot arm with a WSG 32 two-finger gripper. We put AprilTags [141] on the sides of the blocks to get the pose estimate of blocks. We use blocks with different materials and added weight for uncertainty in mass. We use the force-torque (F/T) sensor mounted on the robot's wrist to get force readings. The action parameters for position controls on the real robot are in Euler angles and the angle for the gripper and blocks are in different frames. For safety, we set the task to fail if action parameters get out of safety bounds or a collision happens. Results are in Table 7.2.

The LLM-POP framework (GPT-4) achieves better performance in the real robot compared to the simulation domain. This is mostly because of the very accurate position controller implemented on the real robot leading to fewer execution errors. The stiffness controller and F/T sensor used on the real robot allow us to recover accurate force readings with no noise compared to the "sensor" in the simulator, which is affected by robot movement and gravity. These experiments show that the proposed framework can solve tasks with varying levels of difficulties and uncertainties reliably in simulated as well as real systems.

Simulation: Block manipulation with fine-tuned model

Using the self-instruction method introduced in 7.3, we generate an Alpaca [199]-like dataset and use it to finetune a Llama2-7B [206] model for reasoning. We test it on the same IsaacGym environment. Results are also shown in Table 7.2. *FT-Vanilla* uses directly generated instruction pairs, while *FT-CoT* uses CoT decomposition instruction pairs. The biggest challenge during data generation for both is the correspondence between imagined observation generated by GPT-4 and the ground truth. For example, after picking up the block, the gripper position in the generated data is sometimes not close to the block position (and thus incorrect). This increases the difficulty for fine-tuned models to do correct reasoning based on observations. In the experiments, the fine-tuned model is able to reason about the missing information based on the task description and generate plans to collect the information. The results show that the fine-tuned model benefits from the CoT decomposition of instructions, and failures mostly come from wrong reasoning (wrong "heavy" block based on history). The current gap between the fine-tuned model and GPT-4 lies in the ability to analyze the historical information for updating the information and the ability to avoid and adjust wrong action parameters since it's not included in the current CoT design. A potential improvement is to add an auxiliary task of "observation understanding" in training and use diverse environment settings to improve the reasoning capability. We leave this to our future research.

Common Failures in Sim and Real Experiments

Execution failures

This appears more in the simulation environment when the place action sets a target pose with less tolerance between objects and the robot moves at high speed (The control gains in the simulator are not fine-tuned for various block weights). LLM planners can also generate actions that cause collisions since there's no online collision avoidance in skills.

To explicitly test if the evaluator can help in correcting execution errors, we did an ablation on stacking(*T1*) by adding offsets (1cm) on the grasping position of the block. The initial target placing position will fail because of this offset. With the evaluator included, which asks the LLM to analyze failure action based on history and propose correcting suggestions, the planner outputs a better target position (higher) in the next round. This shows that having an evaluator can actually help to correct execution errors. However, the execution success rate in this test also depends on the lower-level grasping skill; a better grasp of proposal [254] and planning [219] modules can improve the performance. Detailed analysis is deferred to a longer draft of the paper.

Belief update failures

Incorrect plan for collecting information (e.g., trying to reach above the block to measure its weight); wrong analysis results from the observations (e.g., not comparing weight using the force sensor z-axis value but using noise in other axes). In the LLM-POP with GPT-4 case, most failures come from the wrong analysis.

7.5 Chapter Summary

In this chapter, we propose an interactive planning framework LLM-POP that uses LLM to solve tasks under partial observation. The framework is verified in simulated and real robot systems on various partially observable tasks. Task distribution that the current framework can solve strongly depends on the diversity and robustness of the pre-trained skills. Current skills are open-loop actions based on initial observation. If the pre-trained skills are closed-loop policies with collision avoidance and online adjustment, the framework would be able to solve more challenging tasks.

Overall, the gap between fine-tuned model and GPT-4 is clear, especially in reasoning for complex tasks. Our goal is not to replace the GPT-4 but to propose a method for generating self-instruct data for robotic tasks with limited demonstration data. We verify its usage as an interactive planner in manipulation and leave the task of learning more generalizable [191] and transferrable sub-task policies [248, 187], involving more modalities in the observation like image representations in robotics[253, 119, 80] to our future research. Similar settings can generalize to navigation settings and solving complex environments like HomerRobot [234] and social interactive scenarios [185].

Chapter 8

Energy Regularization for Natural Behavior Learning

8.1 Introduction

Humans and animals exhibit various locomotion behaviors at different speeds, optimizing for their energy efficiency. For instance, humans typically walk at low speeds and run at higher speeds, rarely opting for jumping. Prior research demonstrated through optimal control on planar models the correlation between speed and optimal gait choices concerning the cost of transport (CoT). For quadrupeds, the optimal gaits were four-beat walking[1] at low speeds, trotting at intermediate speeds, and trotting/galloping at high speeds [227].

Due to the rich information in the gaits, using a gait as guidance for locomotion policies is popular among lots of reinforcement learning (RL) based methods [175, 130]. However, crafting a versatile and robust locomotion policy that can adapt to and transition between multiple speeds while generalizing across different platforms poses substantial challenges. One of the main challenges here is the reward design. Gait reference can be used as extended state or extra regularization terms in reward functions to provide more supervision. Previous works [130, 163, 50] trained on different quadruped robots within simulation environments like IsaacGym [116, 128] and successfully transferred these policies to physical hardware. However, they often necessitate intricate reward designs and weight tuning. Apart from gait information, reward terms like feet-air time and contact force penalizing [163] were also used to encourage specific behaviors and help stabilize the training. While these additional reward components are aimed at inducing or preventing specific behavioral traits, they inadvertently align with the broader objective of reducing energy costs. This convergence prompts a reconsideration of our reward strategy: could a more straightforward, energy-centric reward term be used to replace the specifically designed terms used in prior policy training? Such a term would encapsulate the core objective of reducing energy consumption and fostering

[1]Four-beat walking, two-beat walking, trotting and galloping are typical gaits for quadruped robots defined in [227] based on feet contact schedule.

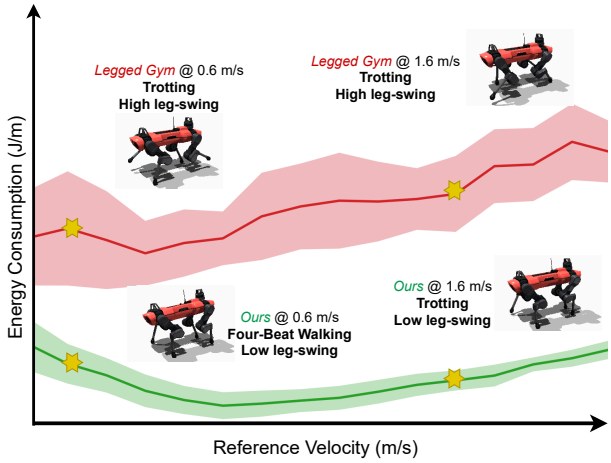


Figure 8.1: Compared to legged-gym baseline [163], our single policy (from one-time RL training) autonomously adopted different energy-efficient gaits (four-beat walking and trotting). It achieved lower energy consumption (low leg-swing) at varying speeds ($0.1m/s$ interval; mean and variance from 50 independent runs).

stable and efficient locomotion.

Building on the concept that energy-efficient gaits correlate with speed [227] and aligning with prior work emphasizing that energy minimization at pre-selected speed results in the emergence of specific gaits [54], this study investigates a more streamlined reward formulation for energy-efficient locomotion. By focusing on energy minimization without intricately designed reward components, we aim to verify if such a simplified approach can yield stable and effective velocity-tracking in quadruped robots across various speeds. Instead of generating multiple velocity-specific energy optimal policies [54], we focus on getting a single energy-optimal policy across all target velocities via RL training.

In our research, we examine the influence of energy regularization weights on policy performance, identifying that both excessively low and high weights can lead to unnatural movements or immobility. Recognizing that energy terms have different scales across velocities and require adaptive velocity-conditioned weights, we first design a non-negative energy reward function and then find an adaptive reward form by interpolating the maximum energy weights at selected speeds to facilitate effective velocity tracking.

Employing this adaptive reward structure within IsaacGym enables the training of robust policies for the ANYmal-C [81] and Unitree Go1 [211] quadruped robots. Our methodology, illustrated in Figure 8.1, identifies appropriate gaits, such as four-beat walking at lower speeds and trotting at higher speeds, without predefined gait knowledge—outperforming baseline approaches [163] in energy efficiency. Our policy also significantly improves velocity tracking and energy consumption performance compared to a policy trained with fixed-weight energy rewards. The trained single policy is deployed on a real Go1 robot to verify

its stable moving and transition locomotion skills in the real world.

The main contribution of this chapter includes:

- Introduction of a streamlined reward formula integrating basic velocity-tracking and adaptive energy minimization to foster stable, velocity-sensitive locomotion policies.
- Demonstration that the derived policies autonomously adopt different energy-efficient gaits at varying speeds without preset gait knowledge.
- Evaluation of the velocity-tracking and energy efficiency across reward structures and weight tunings for ANYmal-C and Unitree Go1, culminating in the real-world application of these policies on a Go1 robot, affirming their efficacy in stable locomotion and gait transition.

8.2 Related Works

Reinforcement Learning for Locomotion Skills

After deep RL demonstrated its capability to fit general policies in an unsupervised manner, researchers actively sought its potential to be deployed on legged locomotion. Hwangbo et al. [82] achieved RL-generated walking policies on a plane ground using a pre-trained actuator net to reduce the sim-to-real gap. Further research also achieved training the policy with adaptive actuator net [84] or motor control parameters [112, 20]. In Hwangbo's follow-up works, locomotion on complicated terrains using a similar approach was also accomplished [133, 105, 27]. With modern GPU-accelerated simulators [116, 128], more time-efficient training frameworks were proposed [163, 131]. Due to the intensive reward engineering in these approaches, the model-free RL tends to converge to a single gait, usually trotting gait, which may not be the most efficient for all terrains and target velocities [151].

Many efforts were made to overcome this limitation by promoting the behavioral diversity of legged robots. Researchers in [32] proposed a hierarchical framework that pre-specifies a set of gait primitives and allows an RL model to choose from them. In [232, 43], gait primitives were parameterized using the contact schedule and RL policy was trained to select these parameters. These works used model predictive control (MPC) as the lower-level controller, which demands preliminary knowledge of locomotion and contact modeling [104]. In [130], behavioral-related arguments such as body height, step frequency, and phase are directly added to the RL model input for end-to-end training. Although various two-beat gaits (where legs touch the ground in pairs) were realized, they cannot generate four-beat gaits (where legs touch the ground in orders) and require manual gait specification in different scenarios. It is more desirable if the quadruped robot can select the most suitable behavior by itself. In [54], a pipeline was proposed to output different gaits under different velocities via minimizing energy consumption. However, this pipeline relies on training and distillation of several velocity-specific RL policies.

Energy Studies on Locomotion

The energetic economy of legged robots has always been an important concern for researchers. The cost of transport, namely the amount of energy used per distance traveled, was introduced in [55, 209], where the minimization of CoT was connected to the choice of speeds [152] and step lengths [210] in human locomotion behaviors. The optimal velocity varies for different gaits, allowing animals to transit between different gaits to move at various speeds.

Conceptual legged models of bipedal [26, 157], and quadruped [94, 137, 227] robots are later designed by researchers to search for energetically optimal motions on various gaits. For quadruped robots, genetic algorithms were used in [94] to study potential gaits at different speeds. Our research is mainly inspired by [227], where optimal control problems are formulated on realistic robot models considering the effects of leg mass, plastic collisions, and damping losses. This chapter uses an unbiased search on energy-efficient locomotion patterns at various velocities and demonstrates the energy vs. velocity curve for different gaits. Inspired by the results, we believe there exists a policy that can transit between gaits at different velocities with an energy-optimization reward. Among previous works, [54] is close to ours in utilizing relations between energy and gaits. It generated multiple policies, with each policy velocity-specific and energy-optimal. In contrast, our work focuses on generating a single energy-optimal policy across various speeds, and it aims to replace the complex-designed reward terms in RL.

8.3 Locomotion Reward Design

A general form of energy regularized locomotion reward takes the following form:

$$R = R_{motion} + R_{energy} + R_{others} \tag{8.1}$$

where R_{motion} encourages accurate velocity tracking, R_{energy} discourages energy consumption and R_{others} includes other necessary rewards to stabilize training. In previous work [54], motion rewards include penalty on linear and angular velocity tracking errors; energy rewards include penalty on motor power with a fixed weight; survival bonus is also added. In experiments, we found the training process unstable potentially due to the negative nature of tracking and energy rewards. Besides, each energy reward weight usually only works within a very narrow range of reference speeds. It is hard to find a single energy reward weight value that works for all reference velocities without knowing more simulation settings and training details. As a result, we proposed the following reward function to promote the automatic generation of energy-efficient behavior of legged robots under various reference velocities.

$$R = \frac{1}{Z(\hat{v}_x)}\Big[R_{lin} + \alpha_{ang} R_{ang} + \alpha(\hat{v}_x) R_{en}\Big] \tag{8.2}$$

where $\alpha_{ang} = 0.5$, $\hat{v}_x$ is the user-specified reference velocity, $\alpha(\hat{v}_x)$ is the adaptive energy reward weight, $Z(\hat{v}_x)$ is the normalizing index for total reward, R_{lin}, R_{ang} are velocity tracking

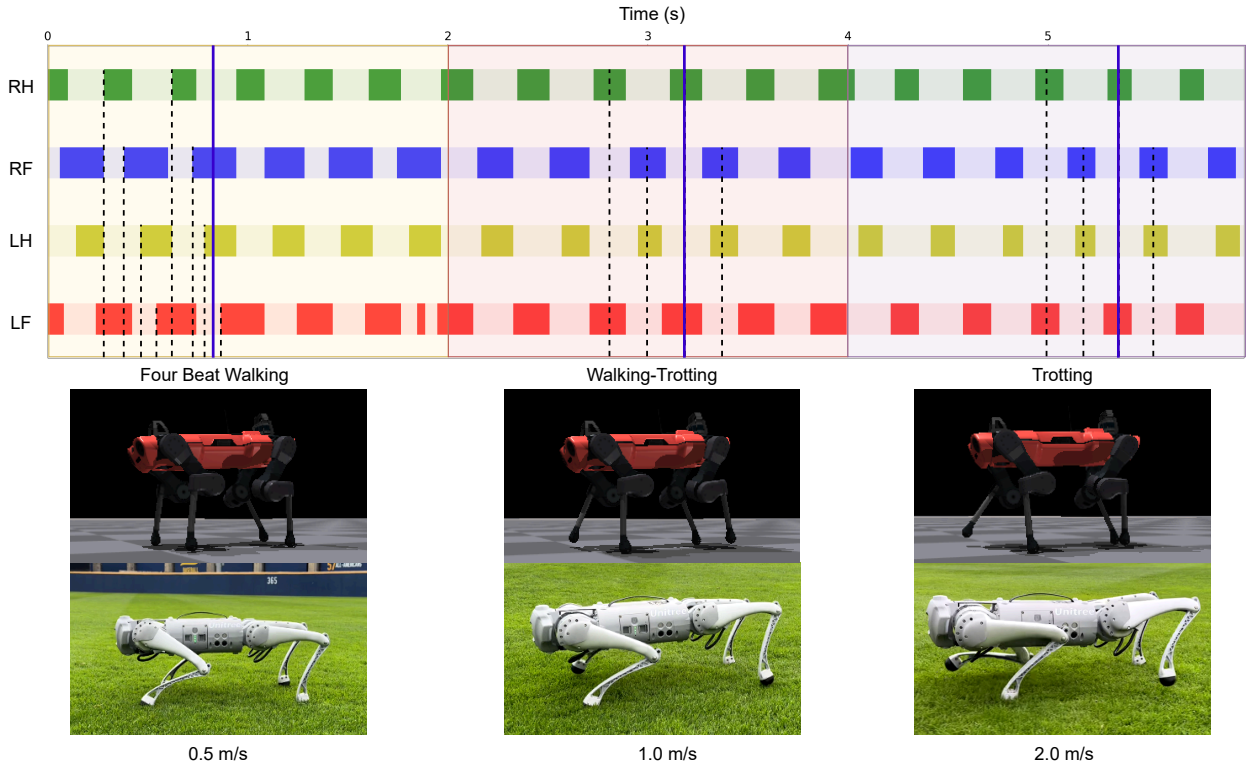


Figure 8.2: **Top row**: Gait plot generated from the ANYmal-C simulation where the legged robot is first commanded to move at 0.5 m/s from 0 to 2 seconds and demonstrated four-beat walking (four legs touch the ground one by one in order, refer to [227] for details), then at 1.0 m/s from 2 to 4 seconds and demonstrated a transition gait between walking and trotting, finally at 2.0 m/s for 4 to 6 seconds and demonstrated trotting (legs touch the ground in pairs). **Middle row**: Snapshots from the ANYmal-C simulation taken approximately at the vertical purple lines in the gait plot. **Bottom row**: Snapshots of Go1 moving on a playground. Its feet contact state approximately corresponds to the vertical purple line in the gait plot.

rewards, and R_{en} is the energy reward. The remaining section elaborates on each component in (8.2).

Motion Rewards

R_{lin} and R_{ang} respectively encourage the legged robot to track the linear reference velocities in two directions $\hat{v}_x, \hat{v}_y$ and angular reference velocities $\hat{\omega}_z$.

$$\begin{aligned} R_{lin} &= \exp\Big(- \frac{|v_x - \hat{v}_x|^2 + |v_y - \hat{v}_y|^2}{\sigma_v} \Big) \\ R_{ang} &= \exp\Big(- \frac{|\omega_z - \hat{\omega}_z|^2}{\sigma_\omega} \Big) \end{aligned} \tag{8.3}$$

$\hat{v}_y$ and $\hat{\omega}_z$ are not user-specified commands, but randomly sampled during training as explained in section 8.4. σ_v and σ_ω are scaling factors depending on the training velocity range. The structure of motion rewards, the coefficient $\alpha_{ang} = 0.5$ for angular velocity tracking, and the scaling coefficients follow the default setting in legged-gym [163].

Energy Rewards

R_{en} rewards the system for consuming less energy while moving.

$$R_{en} = \exp\Big(- \frac{\sum_i |\tau_i||\dot{q}_i|}{\sigma_{en}} \Big) \tag{8.4}$$

The exponential form guarantees a positive reward. τ are each joint's actuated torques, and $\dot{q}$ are joint velocities. We multiply the absolute values of each entry of τ with each entry of $\dot{q}$ and sum them up in (8.4) to follow the fact that a motor does not get charged back even when the applied torque is opposite to the motion [256]. σ_{en} is an energy scaling constant.

Adaptive Energy Weight

$\alpha(\hat{v}_x)$ is the weight of the energy reward terms. Previous works take this value as a constant [54, 256], but we argue that it should be adaptive to the reference velocity $\hat{v}_x$ to achieve suitable behaviors directly from RL. To verify this, we first run RL under fixed sampled $\hat{v}_x$ using pre-selected σ_{en} and try various values of $\alpha(\hat{v}_x)$. When $\alpha(\hat{v}_x)$ is too large, the robot tends to stay unmoved to save energy, neglecting the velocity tracking task. When $\alpha(\hat{v}_x)$ is too small, the robot tends to use a highly inefficient and unnatural way to walk.

To find an adaptive weight, we first collect the largest $\alpha(\hat{v}_x)$ when the velocity tracking error is smaller than a pre-defined small threshold δ for a set of speeds $\hat{v}_x$. After collecting velocity-weight sample pairs $(\hat{v}_{x,j}, \alpha(\hat{v}_{x,j}))_{j=1}^{M}$ for a range of velocities, we see a clear trend (see Figure 8.3): with velocity increasing, the maximum allowed α decreases. This trend corresponds to the fact that the kinetic energy increases quadratically with the velocity. While the motion rewards are expected to converge close to zero, the energy reward has

variant optimal values across velocities. Therefore, we can set a larger weight for lower-speed training but need to decrease it as the velocity increases. When training velocity-conditioned locomotion policies, we linearly interpolate between selected pairs to acquire the velocity-conditioned energy weight $\alpha(\hat{v}_x)$ for the current sampled velocity. We will show in experiments that this is a simple but effective way of training energy-effective policies across different velocities.

Normalization Index

$Z(\hat{v}_x)$ is an adaptive normalization index. Due to the adaptive $\alpha(\hat{v}_x)$, the reward scale under different command velocities varies. This often leads to instability in RL training. As such, for each velocity-weight sample pair $(\hat{v}_{x,j}, \alpha(\hat{v}_{x,j}))$, we also record the final achieved reward $Z(\hat{v}_{x,j})$ and analogously use linear interpolation to get the normalization index curve $Z(\hat{v}_x)$ to stabilize training.

8.4 Locomotion Skill Training Details

Section 8.3 focused on the reward design of the proposed method, which constitutes the most essential part of RL training. However, the energy reward is not a stand-alone one that can be used directly for RL training, and we have to use it together with the basic locomotion rewards. Across quadruped locomotion baselines, the default training settings and locomotion rewards slightly differ. This section explains the basic RL training and simulation settings other than energy regularization. These basic settings can also generate a naive locomotion policy without energy regularization, but these policies usually have low energy efficiency and might have undeployable abnormal behavior. The two baselines we use in this research are *legged-gym* [163] for ANYmal-C and *walk-these-ways* [130] for Unitree Go1.

ANYmal-C Settings

We utilized the robot model and PPO training package in [163]. The system outputs the position command of the 12 joints in the next time step. The system inputs include the linear and angular velocities of the trunk, projected gravity in the robot frame, the commanded x-y velocities, the commanded yaw rate, each joint's position and velocity, as well as the action at previous time step. The commanded y-velocity and yaw rate are fixed at zero here; only the commanded x-velocity will be set. The training episode will reset after 1000 time steps or if any part of the robot except its feet touches the floor.

The policy was trained on a flat ground with the coefficient of friction randomized between $[0.0, 1.5]$. We also disturbed the mass of the robot with a uniform random value in $[-5.0, 5.0]$ kg. A uniformly distributed noise was added to the observation. A random push with x-y velocity uniformly sampled between $[-1, 1]$ m/s lasting 15 seconds was exerted on the robot.

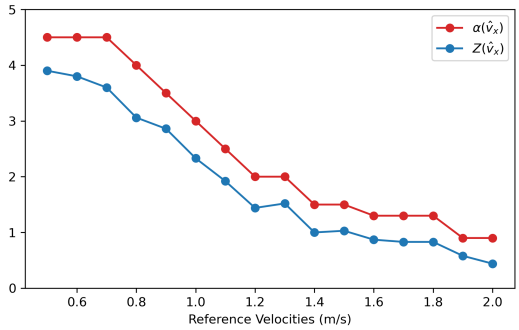


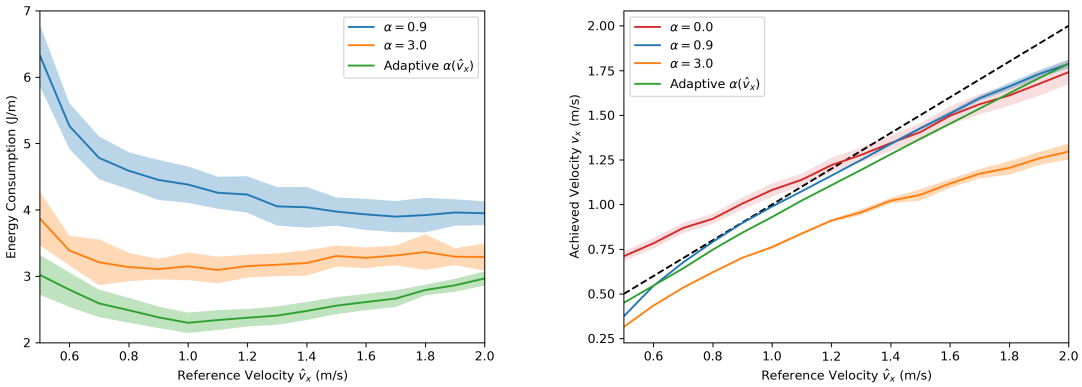
Figure 8.3: $\alpha(\hat{v}_x)$ and $Z(\hat{v}_x)$ corresponding to each reference velocities. These two values are defined in (8.2). For each reference velocity, fixed-velocity trainings were conducted across various $\alpha(\hat{v}_x)$, then we select the largest $\alpha(\hat{v}_x)$ whose velocity tracking error is smaller than a threshold.

All these randomized domain parameters are renewed every time the training episode resets, except observation noise is resampled after every time step.

In motion rewards (8.3), during fixed velocity training (to get results in Figure 8.3) and when the $\hat{v}_x$ is large, the policy may fail to converge to a desirable tracking accuracy. Hence, we use $\sigma_v(\hat{v}_x) = |\hat{v}_x|^2/3$ to overcome this difficulty and σ_ω is fixed at 0.25. In energy rewards (8.4), the energy scaling constant σ_{en} is fixed at 800. To obtain the energy weight curve $\alpha(\hat{v}_x)$ and the normalization index $Z(\hat{v}_x)$, we trained fixed-velocity policies by setting $\hat{v}_x$ at 0.5 to 2.0 m/s, with 0.1 common difference. For each fixed reference velocity, we trained 11 policies using different $\alpha(\hat{v}_x)$ values ranging from 0.5 to 4.5. Figure 8.3 summarizes the corresponding $\alpha(\hat{v}_x)$ and $Z(\hat{v}_x)$ for each reference velocity. These dense parameter pairs are exhibited to visualize the α-$\hat{v}_x$ relation; in real experiments, only a few pairs of parameters at representative speeds are required.

Go1 Settings

We mainly inherited the training methods released by [130]. Similar to [163], the system also outputs the position command of the 12 joints, but its input excludes the linear and angular velocities of the trunk. In addition, the inputs of the previous 30 time steps are also given to the RL system. Compared to ANYmal-C, we found that the energy reward R_{en} alone is insufficient to regularize Go1's behavior, which is likely due to the lighter weight compared to its motor power. Thus, following the settings in [130], we further add a fixed auxiliary reward R_{aux} to (8.2) as an amendment to the normalization index $Z(\hat{v}_x)$. This auxiliary reward is derived mainly from safety concerns, such as penalizing limb-ground collision, out-of-range joint position, and high frequency joint action. The details of R_{aux} can be found on the project website. The training episode will reset after 1000 time steps or if the trunk touches the floor. Similar domain randomization in section 8.4 is applied here.



(a) Comparison of energy consumption

(b) Comparison of velocity tracking error

Figure 8.4: Ablation study in ANYmal-C simulation. Reference velocities are chosen from 0.5 to 2.0 m/s with 0.1 common gap. Results of each velocity are obtained from 50 independent runs. The solid line indicates the average and the shadow indicates the variance. **Left**: Energy consumption per unit moving distance under different reference velocities. We can see that adaptive energy regularization generates the most energy-efficient policy compared to fixed energy regularization. The $\alpha = 0$ policy is neglected because it consumes an uncontrollable amount of energy. **Right**: Velocity tracking error under different reference velocities. Among all fixed energy regularization policies, only $\alpha = 0.9$ has a comparable tracking error. However, the left figure shows that $\alpha = 0.9$ policy is considerably less energy efficient.

Compared to [130], we did not include any gait-related rewards.

We also discovered that curriculum technique is essential even for fixed velocity training. Given a reference velocity $\hat{v}_x$, the sampling range of x-velocity starts with $[-\min\{\hat{v}_x, 1\}, \min\{\hat{v}_x, 1\}]$ m/s. the sampling range increases when the total reward achieves a certain threshold, and the maximal sampling range is set at $[-\hat{v}_x - 0.1, \hat{v}_x + 0.1]$ m/s. The fixed velocity training is considered a success if the x-velocity sampling range expanded to $[-\hat{v}_x - 0.1, \hat{v}_x + 0.1]$ m/s in the end and the tracking error for speed $\hat{v}_x$ is smaller than a threshold δ.

In motion rewards (8.3), both $\sigma_v(\hat{v}_x)$ and σ_ω were fixed at 0.25. In energy rewards (8.4), the energy scaling constant σ_{en} is fixed at 300. We trained fixed-velocity policies by setting $\hat{v}_x$ from 0.5 to 2.5 m/s, with a 0.5 common difference. For each reference velocity, we trained eight policies using different α values ranging from 0.7 to 2.1, then fit the $\alpha(\hat{v}_x)$ curve with the method in section 8.3.

Figure 8.5: Three policies walking at 0.5 m/s. Built-in MPC and *walk-these-ways* (WTW) [130] swing the legs to a redundant height, while our policy only makes necessary lifting of the leg. This demonstrates the energy-efficiency of our policy.

8.5 Experiments

The experiments were designed to show the following after the legged robot was trained using the adaptive energy-regularized reward shown in Equation (8.2).

- With adaptive $\alpha(\hat{v}_x)$ and $Z(\hat{v}_x)$, the legged robot automatically selects suitable behaviors to move with the reference velocity. We also demonstrate that the trained policy is deployable to a real quadruped robot.
- If $\alpha(\hat{v}_x)$ and $Z(\hat{v}_x)$ are constants, the legged robot may not be able to find an energy-efficient walking policy for all target velocities.
- Our method can generate a more energy-efficient walking policy compared to established baselines.

One Policy with Different Gaits at Different Velocities

To demonstrate natural emergence of efficient locomotion gaits, we evaluate the trained walking policy under different reference velocities. Fig. 8.2 demonstrates a trial run where the legged robot was commanded to move at $\hat{v}_x = 0.5$, 1.0 and then 2.0 m/s. Each commanded velocity lasts for two seconds. We plot the gait recorded from the ANYmal-C simulation and show the snapshots for simulated ANYmal-C and real world Go1. We can see that our trained policy exhibits four-beat walking at low speed (0.5 m/s) by moving one leg each time in the order of right hind, right front, left hind and left front. At medium speed (1.0 m/s), the policy exhibits an intermediate gait between walking and trotting. At this gait, the right hind and left front legs move at the same time, while the right front and left hind legs have a displacement in their motions. At high speed (1.0 m/s), the trained policy exhibits a standard trotting gaits, where the right hind and left front legs move together, and the right front and left hind legs move together. This gait transition is endorsed

by previous works [227, 232] that four-beat walking and trotting are respectively the most energy-efficient gaits under low and high speeds.

A closer look at the snapshots in Figure (8.2) uncovers that the swing ratio was also regularized. The legged robot only lifts up its leg to a height necessary to reach the reference velocity $\hat{v}_x$ to avoid wasting energy. At low speed, it only mildly lifts up its feet. As the reference velocity increases, the robot also lifts its feet higher, but only to a necessary height. Without considering energy-efficiency, the trained policy usually tends to lift the feet redundantly high. This will be argued in more detail in section 8.5.

Finally, Figure 8.5 also showcases a successful hardware deployment of our policy. Videos can be found in our project website stated in the abstract.

Ablation Studies with Fixed Energy Rewards

In ANYmal-C simulation, when the reference velocity is fixed at 1.0 and 2.0 m/s, we found that velocity tracking error was reasonably small when α was set as 3.0 and 0.9. Therefore, we run ablation experiments by fixing α at these values and 0.0 to compare with varying $\alpha(\hat{v}_x)$.

Figure 8.4a shows the energy consumption for adaptive α, $\alpha = 0.9$ and $\alpha = 3.0$. The policy with $\alpha = 0.0$ is dropped because it consumes multiple orders more energy. We observe that adaptive α reaches the lowest energy consumption. Figure 8.4b shows the velocity tracking error. Only the $\alpha = 0.9$ policy has comparable tracking accuracy with adaptive α, but Figure 8.4a shows that $\alpha = 0.9$ policy consumes considerable more energy. These observations conclude the non-triviality of finding a constant energy reward scale α and elaborates the usefulness of $\alpha(\hat{v}_x)$ varying with reference velocity.

Comparison to Other Methods on Go1

We also compare our method with built-in MPC and *walk-these-ways* on real world Go1 robot. All policies are commanded to move at 0.5 m/s. Such low velocity requires only mild movement of each leg for Go1. Figure 8.5 shows snapshots of each policy. It can be seen that both built-in MPC and *walk-these-ways* over-swing the legs, which squanders energy. Our policy swings the leg only to a necessary height, which can be visually recognized as the most energy-efficient.

8.6 Discussions

This chapter focuses on developing energy-efficient locomotion strategies for quadruped robots, employing a simple yet effective reinforcement learning approach. However, there are some inherent limitations which provide avenues for future research.

Limitations

The core limitation of the presented approach lies in the requirement for pre-running experiments to determine appropriate energy regularization weights. This necessity stems from our method's reliance on empirical observations to calibrate the weights, which, while effective, does not afford the flexibility in a fully adaptive reinforcement learning system. Although the policy developed is applicable across different speeds post-training, it cannot be obtained with only one training for a new quadruped platform. Moreover, we only tested its generalizability across speeds. We did not verify the adaptive capability of energy rewards in different environments, which would be crucial for deploying these robots in real-world scenarios where they might encounter multiple operational challenges.

Future extensions

Future research could address these limitations to develop methodologies for automatically tuning energy regularization weights within one single reinforcement learning training. This would enable the system to dynamically adjust its strategy in response to multi-task RL [192] or cross-embodiment settings [242, 188, 236].

Moreover, while this study concentrated on locomotion tasks, the underlying principle of leveraging energy efficiency to drive behavior selection holds broader potential. Future work could explore applying this energy-centric approach across different robotic tasks. For instance, manipulation and interaction tasks could also benefit from strategies prioritizing energy efficiency, potentially finding natural, efficient behaviors analogous to those observed in biological systems [25, 214]. Such a framework would align robotic systems more closely with sustainability principles and environmental consciousness.

8.7 Chapter Summary

This chapter presented a novel approach to energy-efficient locomotion in quadruped robots through the implementation of a simplified, energy-centric reward strategy within a reinforcement learning framework. Our method demonstrated that quadruped robots, specifically ANYmal-C and Unitree Go1, could autonomously develop and transition between various gaits across different velocities without relying on predefined gait patterns or intricate reward designs. The adaptive energy reward function, adjusted based on velocity, enabled these robots to select the most energy-efficient locomotion strategies naturally. Our policy showed energy-efficient behaviors and gait transitions in both simulation experiments (ANYmal-C) and hardware experiments (Go1). We also demonstrated the usefulness of adaptive energy regularization via ablation studies.

Chapter 9

Representation Learning for Grasping Skills

9.1 Introduction

Robotic grasping in unstructured environments can benefit applications in manufacturing, retail, service, and warehousing. Grasping unseen objects is, however, highly challenging due to the limitations in perceptions. When objects are cluttered in a bin, exact geometry and position of objects are obscured. Sensing imprecision and deficiency then leads to poor grasp planning execution.

Model-based and learning-based methods could be used to plan grasps across a wide variety of objects. Existing physical grasp analysis techniques, such as grasp quality metrics [161], template matching [173], and wrench space analysis [139], can be used to search for the optimal grasp. These approaches, however, can be less robust in practice due to the perception limitation. Incompletion of the object surface can lead to flawed analysis. An alternative approach is to plan grasps with supervised deep learning. Current methods show that it is preferable to learn grasp quality functions and optimize them at the runtime [127, 126, 135, 136, 146, 45, 47, 46, 114]. Learning intermediary information, such as grasp qualities and success rate, can improve the training efficiency and prediction accuracy. However, the requirement of the sampling or optimization makes the algorithm time-consuming. To tackle this, other methods use end-to-end learning to infer grasp poses from the sensor inputs directly [140, 174]. Nevertheless, these algorithms require larger datasets and elaborate hyper-parameters tunning to reduce the training variance.

Ideally, the grasp planning algorithm generates 6 degrees of freedom (DoF) grasps. Previous works have proposed models that can detect top-down grasps using depth images [127, 86, 144]. However, the top-down nature of such grasps does not allow robots to pick up objects from different orientations, which limits its application in cluttered environments. In this work, we propose to use six variables $(x, y, \theta, \gamma, z, \beta)$ to represent a 6-dimensional grasp in a depth image, as shown in Fig. 9.2. Three planar 3D grasp poses (x, y, θ) in Fig. 9.2(a) are

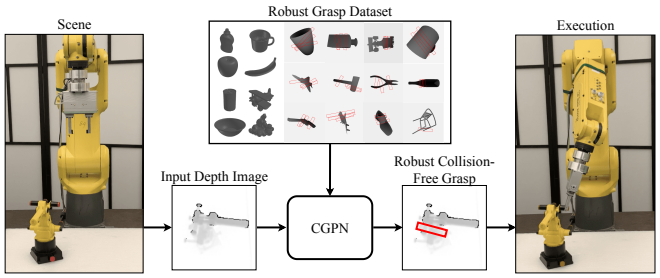


Figure 9.1: CGPN Architecture. The 6-DoF contrastive grasp proposal network (CGPN) is trained offline to infer grasps from depth images using a dataset of synthetic images and grasps. When an object is presented to the robot, a stereo camera captures a depth image; CGPN could rapidly generate 6-DoF robust collision-free grasps, which is executed with the Fanuc robot.

the center position and orientation of the bounding box in the image plane. The bounding box's width w and height h are used in training but not in representing grasps. Three spatial grasp parameters shown in Fig. 9.2(b) are: the tilt angle γ among axis ω, the rotation angle β among the grasp axis φ, and the depth of the grasp z.

In previous works [127, 126, 135, 136], the strategy to train on synthetic datasets and apply to reality has been heavily used. It has been shown that rendered images with numerically computed grasp qualities can ease the data preparation process. The simulation-to-real (sim-to-real) gap, however, is still an open problem for grasp planning. Synthetic data have better resolution and less noise than real images. To tackle this problem, we introduce contrastive learning with sim-to-real depth image processing in this paper. Contrastive learning aims to extract invariant features from augmented images, which improves the overall performance of modeling under vision noise.

In this paper, we propose an end-to-end 6-DoF contrastive grasp proposal network (CGPN). The general framework is shown in Fig 9.1. When an object is presented in the scene, a stereo camera captures a depth image; CGPN rapidly generates 6-DoF robust grasps, which are executed with the Fanuc robot. CGPN is trained with synthetic grasp images with variant augmentation techniques to bridge the sim-to-real gap.

The contributions of this paper are as follows.

1. An end-to-end grasp planning model is proposed to detect grasps efficiently. The model consists of a feature encoder, a rotated region proposal network, a grasp refinement network, and a collision detection module. The model uses single-view depth images as input and infers 6-DoF grasps for parallel-jaw grippers.

2. A contrastive learning module and a depth image processing technique are introduced in the grasp planning to resolve the sim-to-real gap.

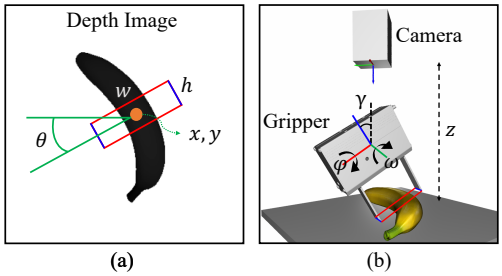


Figure 9.2: Grasp Representation $(x, y, \theta, \gamma, z, \beta)$. The planar 3D grasp pose (x, y, θ) in (a) represents the center position and orientation of the projected bounding box on camera plane. The bounding box's width w and height h are used in training but not in representing grasps. The other 3 grasp parameters are shown in (b): tilt angle γ is the rotation among axis ω, z is the depth of grasp, and gripper angle β is the rotation among the grasp axis φ.

The remainder of this paper is organized as follows. Related works are introduced in Section 9.2. Section 9.3 presents the proposed grasp planning algorithms in detail. Experiments are presented in Section 9.4. Section 9.5 concludes the paper and introduces the future work.

9.2 Related Works

Grasping

Recent learning-based approaches have demonstrated promising efficiency and robustness in the grasp planning task. Some works propose to optimize learned grasp quality metrics at runtime [127, 135, 136, 146, 114, 46]. Sampling grasp candidates among the object's surface, however, can be time-consuming and requires pre-defined heuristics. Other approaches [140, 174, 246, 86, 29, 144, 122] propose to directly infer grasp poses from the raw input with an end-to-end model. These approaches design a deep model to perform grasp pose regression and grasp quality assessment at the same time.

To represent grasps in the input image, rotated rectangle representation, or rotated bounding box, has been utilized by previous work [241]. [241] demonstrates that by using a two-step search strategy, 3-DoF grasps could be located within the image plane. This algorithm, however, uses comprehensive searching in the whole input image, thus is time-consuming.

Region Proposal Network

The success of the region proposal network (RPN) [60, 158] brings focus to the bounding box grasp representation. The objective of RPN is to classify and regress object's bounding boxes in images. Instead of sampling candidates in the original image plane, RPN utilizes

shared feature maps. Classification and regression are performed in the latent feature spaces, which has been shown to have promising accuracy and time complexity. [29, 246, 86, 144, 122] introduce the RPN network to the grasp planning. These approaches, however, focus on 3-DoF top-down grasps and require an additional grasp detector to predict the rotation angle. In this work, we propose to infer rotated grasps directly from the image plane. Besides, we use a downstream grasp refinement network to regress the full 6-DoF grasps.

Contrastive learning

Contrastive learning[67], and many of its recent applications [72, 21] in visual representation learning, can be thought of as training an encoder to extract invariant features for similar images. This encoding mechanism helps maintain the understanding of the same object under correlated views. Contrastive losses measure the similarities of sample pairs in representation space. [41] uses the latent parametric feature to represent instance as a class.[255, 202] introduce memory-bank based methods and momentum to store and update the class representation. For grasping, this helps to reduce the instability of the model resulting from the sim-to-real gap and various input noise in a cluttered environment. In our framework, we generate stable grasp proposals by leveraging recent advances in data augmentation and contrastive loss.

9.3 6-DoF Contrastive Grasp Proposal Network

General Framework

This section introduces the framework of the 6-DoF contrastive grasp proposal network, as shown in Fig. 9.3. The input to the whole pipeline is a single-view depth image of the scene. Segmentation is first applied to the image to separate objects. The CGPN model generates grasps for the object using its segmented depth image.

Training Phase

During training, the CGPN algorithm works as follows. First, two separate data augmentation operators t, t' are sampled from the augmentation family $\mathcal{T}$. Augmentations are applied to the segmented depth image to obtain two correlated views x_q, x_k regarded as a positive pair in the contrastive learning module. Similar to [72], the other inputs in the same batch are viewed as negative samples. Second, the positive and negative samples are fed into the contrastive encoder. We get query q from x_q with query encoder and keys $\{k_i\}_1^N$ from x_k and other negative samples with key encoder. The key encoder is a slowly updated query encoder with no gradient update. Contrastive loss is calculated using these queries and keys, with more information introduced in the loss section. The purpose of this module is to maximize the agreement of encoded feature maps q, k. We assume the encoder can learn invariant representations of the depth images under different augmentation operations,

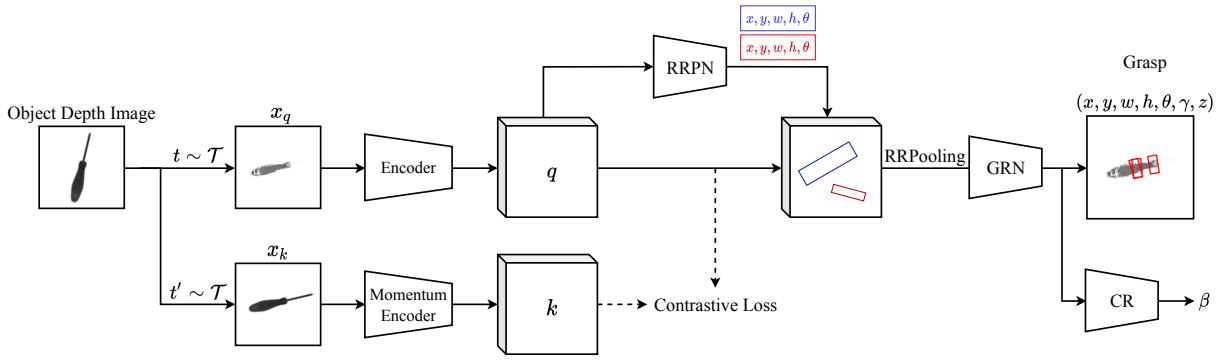


Figure 9.3: CGPN Network Architecture. During the training, a synthetic depth image of the object is feeding into the network. First, two separate data augmentation operators t, t' are applied to the segmented depth image to obtain x_q, x_k, which are then input to the contrastive encoders. Second, extracted feature maps q, k are parsed to the rotated region proposal network (RRPN) to generate grasp regions. Third, a rotated region pooling (RRPooling) module extracts feature vectors from the feature map q using the generated grasp regions. Finally, a grasp refinement network (GRN) infers the tilt angle γ and the depth z using the local feature vectors. A collision refinement is further added to search the rotation angle β.

and the query q is the feature we use for downstream tasks. Third, a rotated region proposal network (RRPN) is introduced to propose 3-DoF grasp regions base on the feature map q. The output of RRPN determines 3-DoF grasp (x, y, θ) and a box shape (w, h). We still need local features to determine other grasp parameters. Fourth, a rotated region pooling (RRPooling) module extracts feature vectors from the feature map q using the predicted rotated bounding box from RRPN. These local feature maps contain depth and other features around the proposed grasp position. Finally, a grasp refinement network (GRN) is designed to infer the tilt angle γ and the depth z using the local feature vectors. This completes a 5-DoF grasp pose together with (x, y, θ). The last DoF β is searched with a collision refinement (CR) module, thus is neglected in the training process. The training loss is the weighted combination of the contrastive loss, region proposal loss, and the grasp refinement loss. After training, we get a query encoder, a RRPN module, and a GRN module used for online testing.

Testing Phase

During the testing phase, we do not need to create positive pairs for the contrastive modules. Instead, we directly put the input depth image into the query encoder. The following process is the same as training. The output of the CGPN model is valid grasp poses for each object. After that, the 5-DoF grasps for each object are projected back to the

original cluttered scene. The last DoF β is determined according to the collision constraints.

Data Augmentation and Contrastive Learning

The instability in real-world grasping is usually brought by occlusion in cluttered scenes, background noise in the environment, and the sim-to-real gap. To overcome this, we design multiple data augmentation operations $\mathcal{T}$ for the input depth image.

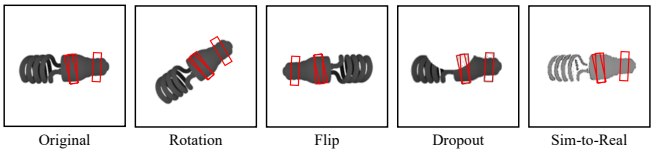


Figure 9.4: Illustrations of the available data augmentation operators in $\mathcal{T}$. Each augmentation can transform the original input with some internal parameters (e.g. rotation degree, flip axis). We use combination of the first four operations and the sim-to-real process as a complete augmentation for a single image.

Spatial/geometric transformation

Spatial operations include rotation, flip, and dropout on the original synthetic depth image. Each augmentation can transform the input image stochastically with some internal parameters. We use composition of these operations to create different observations for the same grasp object. Note that unlike classification labels in vision, the ground-truth high-quality grasp poses would change along with these augmentation operations.

Sim-to-Real Processing

Similar to [195], we design a sim-to-real transfer operation by leveraging several image processing techniques. We randomly paint the pixels black in areas of high Laplacian gradient and edges of the object detected by Canny edge detector[5]. Then we inject realistic noise into the image. The operation is parameterized by a threshold of black paining areas. Compared to spatial transforms, the sim-to-real process does not change the ground-truth pose in the image.

Illustrations of the data augmentation operators are shown in Figure 9.4. Using the introduced operations, we extend the training dataset and create positive pairs from the same input depth image for the contrastive training process. The transformation family $\mathcal{T}$ is created by a random combination of the spatial operations and the sim-to-real transfer as the last operation. All of the operations have an execution probability. In this way, we can train on synthetic data generated in the simulator and use the model to predict valid grasp in real-world scenarios.

Regarding the contrastive learning module in the pipeline, as described in previous sections, we use separate encoders for query input x_q and other positive or negative samples as suggested in [72]. Note that some operations in the data augmentation change the ground-truth in the downstream grasping tasks (e.g. spatial operations), directly maximizing the similarity between q and k^+ might not be suitable for grasping regression. As suggested in [21], we add a multi-layer perceptron (MLP) after the query and key encoders as a projecting head and get $\hat{q}, \hat{k_i}$. This is neglected in the network architecture but implemented in experiments. The query and key vector q, k_i can keep the differences in grasping, but the projecting head would catch the invariant properties among positive pairs. We use q for downstream grasping tasks, but use $\hat{q}$ and $\hat{k_i}$ for calculating contrastive loss.

Rotated Region Proposal

Another core part of our architecture is the rotated region proposal network. Similar to the architecture in [124], which output detected bounding box for scene text, our network receives feature maps from the learned contrastive encoder and output candidate region proposals with class labels and parameters of the positions. The class label determines if the proposal fits any robust grasp, and the parameters (x, y, w, h, θ) gives a rotated bounding box.

Unlike text detection, we do not have a fixed number of non-overlapping labelled bounding boxes for each input image. For each object and its depth image, the high-quality grasp poses may overlap with others. When matching the ground-truth grasps with proposed regions, overlapped grasps may have similar skew intersection over union (IoU) results. Directly choosing the one with the highest IoU may cause all proposals matching the "largest" grasp (i.e., large w, h). To avoid such mode collapse, we introduce to random select among top-k grasps based on IoU during matching.

For parameters of the proposed region, although only (x, y, θ) is used in a grasp pose, the box's width w and height h indicate the range of local features used in the following GRN. The GRN model first aligns and extracts rotated regions of interest by projecting proposals from the RRPN onto the feature map and then use the local features to predict the tilt angle and relative depth of the grasp, which finalizes the 5-DoF grasp $(x, y, \theta, \gamma, z)$.

Collision Refinement

The grasp proposal models mentioned above are used for 5-DoF grasps for a single object. To use the proposed grasps in a cluttered scene, we use collision constraints to infer all 6 DoFs. The proposed 5-DoF grasp $g = (x, y, \theta, \gamma, z) \in \mathcal{G}$ are frozen and a posterior optimization process is used to search for the last rotation angle β:

$$\min_{\beta} \quad \sum_{i}^{N} C(g, \beta, x_i) \tag{9.1}$$

where C is the collision check score for a 6-DoF grasp (g, β) and surrounding objects. $\{x_i\}_{i=1}^{N}$ is the segmented depth image for the N object in the scene.

Sign-distance field is introduced in this paper to model the collision score:

$$C(g, \beta, x_i) = -SD(FK(g, \beta), x_i) \tag{9.2}$$

where $FK(\cdot)$ denotes the forward kinematics function of the robot. $SD(\cdot)$ denotes the signed-distance function of the robot and the object x_i. Given a 5-DoF grasp, there are infinitely many grasp candidates since the rotation among the grasp axis φ is free-floating. By minimizing the negative signed distance between the robot and the object, a unique 6-DoF grasp can be determined.

Loss Design

Region Proposal Loss

Despite the randomness we introduced in positive region matching, most of the model we use in RRPN is similar to [124]. The loss function for the proposal takes the form of:

$$L_p = -\log s_{\text{pos}} + \sum_{v \in \{x,y,\theta,h,w\}} \lambda_i \text{smooth}_{L_1}(v^* - v) \tag{9.3}$$

where s_{pos} is the matching IoU for positive pairs, v^* is the ground-truth value for corresponding variable. The smoothed L_1 loss is:

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases} \tag{9.4}$$

The output (x, y, θ) is grasp parameter and (w, h) are parameters for local feature ranges in original feature map. Therefore we assign more weight on λ_x, λ_y and λ_θ and less on λ_w, λ_h.

Grasp Refinement Loss

To find the best grasp pose, we need not only the rotated planar grasp position proposed by RRPN but also the tilt angle γ and depth z of grasp. While RRPN gives us a rough estimation of grasp positions, GRN uses the proposed region of interests (ROIs) to generate accurate grasp poses with local features. We formulate a weighted refinement loss in Eq. (9.5) to minimize the L1 error of tilt and depth.

$$L_r = \lambda_\gamma \|\hat{\gamma} - \gamma\|_1 + \lambda_z \|\hat{z} - z\|_1 \tag{9.5}$$

Contrastive Loss

Unlike regression loss, the contrastive loss does not have supervised signals. For simplicity, we use q, k_i to represent $\hat{q}, \hat{k_i}$ after the projecting head. Consider a query q encoded from sample x and a dictionary of N keys $\{k_1, k_2, ..., k_N\}$ encoded from different samples. Among all keys, there is one positive key k^+ encoded from x_k similar to x_q and $N-1$ negative keys from other samples in the batch. Using dot product as the similarity metric, the loss function is defined as Eq. (9.6), where τ is a temperature hyper-parameter.

$$L_q = -\log \frac{\exp(q \cdot k^+)}{\sum_{i=1}^{N} \exp(q \cdot k_i)} \tag{9.6}$$

This is the log loss of a N-way softmax-based classifier that tries to classify q as k^+, introduced as InfoNCE in [142].

The overall loss function used to train the contrastive grasp proposal model is:

$$L_{\text{overall}} = \lambda_p L_p + \lambda_r L_r + \lambda_q L_q \tag{9.7}$$

We adjust the relative weight of the three losses at different stages of training. In the beginning, we set the loss of RRPN and GRN low to reduce the contrastive loss. After we get a stable encoder, we increase the weight of RRPN loss but keep the GRN loss weight low to train the 3-DoF grasp position and local feature bounding box. We then gradually increase the weight of GRN to train the overall pipeline for the final 5-DoF grasp. The high weight of GRN loss at an early stage would affect RRPN's training since the models work in series.

9.4 Experiment

Dataset Generation

To generate the grasp sets, we need to sample a large number of grasps for single objects and label the top-ranked robust grasps as ground truths. This is unrealistic for real robots but not hard in simulation environments. We train our CGPN model on the generated single object grasp dataset and use it on real robots. Similar to [127], 1,366 objects are selected from the 3DNet [226] as the object set. 100 antipodal grasps are evenly sampled among the surface for each object. Each grasp is labeled with the robust force closure metric and is represented by its contact points (c_1, c_2) in 3D. Since the CGPN algorithm requires depth images as input, objects and grasps are projected to the image plane. For each selected object, 20 synthetic depths images are rendered from different angles. The object is placed at the center of a regular icosahedron; cameras are placed at each face's center and point to the origin. The distance between the camera and the object is sampled from $\mathcal{U}(\sqrt{3}r_{obj}, 2r_{obj})$, where $\mathcal{U}$ denotes the uniform distribution and r_{obj} is the object bounding ball's radiance. Such selection makes sure that the full object is visible in the camera.

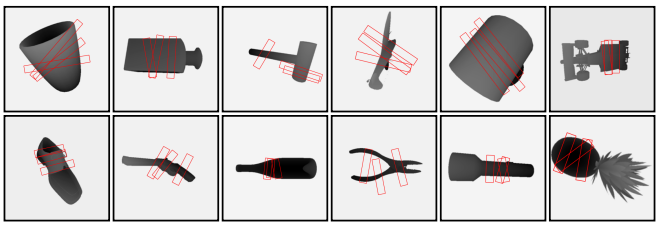


Figure 9.5: Dataset Samples. This figure shows 12 samples from the generated grasp dataset. 3D grasps are projected to the image plane (red rectangles). The tilt angle γ and the distance z are neglected in the plot for simplicity.

Each grasp (c_1, c_2) is then projected to the depth image using projective transformations:

$$\begin{bmatrix} u_i & v_i & f \end{bmatrix}^T = K \cdot \begin{bmatrix} R & t \end{bmatrix} \cdot \begin{bmatrix} X_i & Y_i & Z_i & 1 \end{bmatrix}^T \tag{9.8}$$

where X_i, Y_i, Z_i are positions of the contact point $\{c_i\}_{i=1}^2$ in the camera frame, $K, \begin{bmatrix} R & t \end{bmatrix}$ are the camera's intrinsic and extrinsic matrices, respectively. u_i, v_i are pixel's locations at the image for point c_i. The bounding box's other parameters are then computed. The width w and the height h of the box is set to $\|[u_2 - u_1, v_2 - v_1]\|_2$ and 20 respectively. Grasp depth z, rotation angle θ, and tilt angle γ are computed as $z = \frac{1}{2}(Z_1 + Z_2)$, $\theta = \tan^{-1} \frac{u_2 - u_1}{v_2 - v_1}$ and $\gamma = \tan^{-1} \frac{Z_2 - Z_1}{w}$ respectively.

Ground truth grasps for each image are selected as the top 20% from 100 samples. In this paper, we limit the tilt angle's range to $[-3030$. To give rotation angle θ and tilt angle γ unique definition, we set constrains on the grasp points in the image plane, such that $v_2 > v_1$. In other words, the point $[u_2, v_2]^T$ is always on the right of the point $[u_1, v_1]^T$. Then, we bound θ in the range of $[-9090$, and γ is relabeled with the corresponding sign. The generated training dataset has 24,723 depth images with labeled ground truth grasps. Fig. 9.5 shows some samples from the dataset.

Experiment Results

The proposed CGPN is run on a desktop with GTX2080Ti GPU, 32GB RAM, and 4.0GHz CPU. For the experiment, we use a FANUC LRMate 200iD/7L industrial manipulator with a SMC LEHF20K2-48-R36N3D parallel-jaw gripper for grasping. A Kinect v2 camera is used to capture depth image of the scene. The point cloud library [164] implementation of region growing method is utilized to pre-process and segment the object.

We leverage state-of-the-art model architectures for each submodel. ResNet-50 [71] and rotated region proposal networks [124] are utilized as the encoder and downstream models. The hyper-parameter for RRPN and contrastive learning are mostly the same as introduced in [124, 72] and $\lambda_x = \lambda_y = \lambda_\theta = 5$ and $\lambda_w = \lambda_h = 1$. We design the anchor aspect ratio

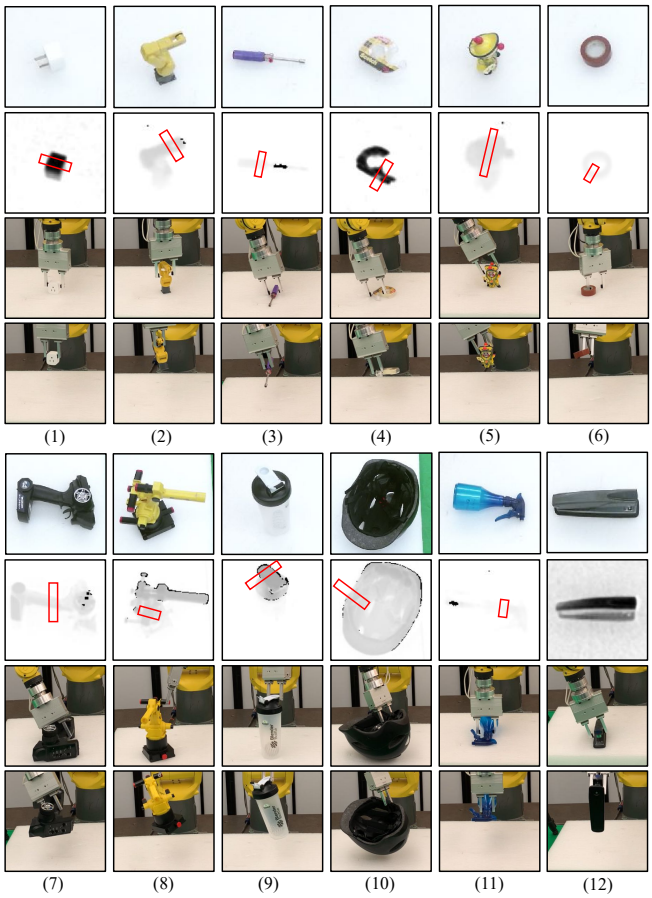


Figure 9.6: (1-12) The grasp planning and execution results on 12 objects with a single depth image. For each column, (Top Two Rows) show perceived RGB and depth images with planned grasps, (Third Row) shows physical grasps reaching the target grasp, and (Bottom Row) shows the execution results. The tilt angle γ, and the distance z are neglected in the plot for simplicity.

as $[0.5, 2]$ to fit to grasp poses better. During the first 20 epochs, $\lambda_p, \lambda_r, \lambda_q$ take values of $(1, 1, 5)$ respectively. After that, they are modified to $(5, 5, 2)$ to stabilize the training.

As introduced in 9.4, the distance between the object and the camera is drawn from $\mathcal{U}(\sqrt{3}r_{obj}, 2r_{obj})$. In reality, such condition is hard to achieve since the camera is usually fixed at a particular point. To tackle this, we propose to re-project the depth image into a virtual camera. Object's point cloud is first generated base on the depth image. r_{obj} is then computed as the radiance of the point cloud's bounding ball. Next, the virtual camera's position is determined by the sampled camera-object distance. Finally, the generated point cloud is projected to the virtual camera to obtain the normalized depth image.

Table 9.1: Performance analysis of the CGPN and baselines on single object grasping tasks.

	Success Rate	Time (sec/grasp)
GPD	72.2%	1.84
CGPN w/o Contrastive	69.4%	0.46
CGPN w/o Data Augmentation	66.7%	0.46
CGPN	**75.0%**	**0.46**

Fig. 9.6 shows the grasp planning and grasp execution results on 12 different objects with a single stereo camera. The top two rows of the figure show the captured RGB-Depth images. Located grasps are labeled in the depth image with red rectangles. The tilt angle γ, and the distance z are neglected in the plot. The physical grasp pose and the execution result of the planned grasp are shown in the bottom two rows. The algorithm is able to find robust grasps for a) small objects close to the ground, b) large objects with graspable regions, and c) objects with complex surfaces.

Table 9.1 compares CGPN with the grasp pose detection (GPD) [146] algorithm, which also focuses on 6-DoF grasp planning with a single camera. To adopt GPD, we train a point-cloud-based grasp evaluation network with the same dataset as CGPN. Each object is grasped three times, results in 36 trials for each algorithm. From experiments, we observe CGPN outperforms GPD in both the grasp success rate and the computation time. One reason behind this might be the robustness of our model under the various camera angles we set during the experiment. Regarding time cost, CGPN generates valid grasps in an end-to-end manner, it does not require the sampling and evaluation procedure and therefore takes less time to plan a grasp.

For ablation study on the data augmentation and contrastive learning module, Table 9.1 also compares the effectiveness of adding augmentation operations on training data and using contrastive loss in the sense of object grasp success rate. As can be seen, the sim-to-real gap significantly affects the performance of the grasp proposal network. Ignoring vision noises and training on the synthetic dataset may yield poor results in practice.

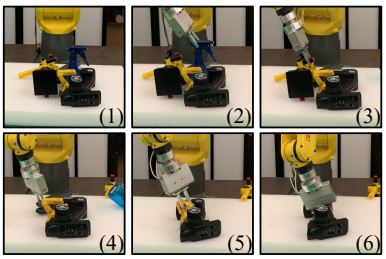


Figure 9.7: (1-6) show a sequence of proposed grasps in a cluttered scene.

Cluttered scene grasp

Fig. 9.7 shows a sequence of proposed grasps in a cluttered environment. The grasp sequence is selected according to the graspability, or whether a collision-free grasp exists for a particular object.

Failure cases

Fig. 9.8 displays two typical failures for CGPN, in which no grasp is proposed. The first failure mode occurs because of the unmodeled sim-to-real gap. The object surface's resolution and distance noise are not appropriately handled. The second type of failure occurs when no robust grasp exists with $\gamma \in [-3030$. Compared to 3D representations, the image includes less geometric information, making the end-to-end model hard to infer 6-DoF grasp. It appears that the performance could be improved with comprehensive data augmentations and other grasp representations.

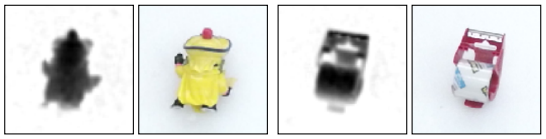


Figure 9.8: Two failure modes of CGPN. (a) shows the unmodeled sim-to-real gap on the object's surface, and (b) shows the limitation of the depth image in representing 6-DoF grasps.

9.5 Chapter Summary

This chapter presents a 6-DoF contrastive grasp proposal network (CGPN) to generate robust grasps on single-view depth images. CGPN is composed of a grasp planning module for 6-DoF grasp detection and a contrastive learning module for sim-to-real gap reduction. The grasp planning module infers 6-DoF grasps based on detected robust grasp regions. An image encoder is used to extract the feature map, followed by a rotated region proposal network to propose planar grasps. Feature vectors are then extracted and refined to 6-DoF grasps. To transfer grasp skills trained in simulation, a contrastive learning module and variant depth image processing techniques are introduced during the training. CGPN can locate 6-DoF collision-free grasps using a single-view depth image within 0.5 seconds. Experiment results show that CGPN outperforms previous grasping algorithms.

Chapter 10

Conclusions and Further Works

This dissertation aims to demonstrate a suite of innovative methods for predicting, generating, and learning behaviors in autonomous systems. Our focus was distributed across three key aspects: Interpretable predictions of different types of road participants in autonomous driving (Chapter 2 and 3), diverse interaction generation (Chapter 4 and 5). and generalized skill learning (Chapter 6, 7, 8, and 9).

In Chapter 2, a Multi-agent Hybrid Dynamic Bayesian Network (MHDBN) method was proposed to model the state changes of multiple, heterogeneous agents in various scenarios. Motion data with pedestrian-vehicle interactions from a four-way-stop intersection in the real world was used to design the model and verify the effectiveness of the proposed framework's estimation and interactive prediction capability.

In Chapter 3, the joint trajectory prediction problem using the goal-conditioned framework was studied. We introduced a conditional-variational-autoencoder-based (CVAE) model to explicitly encode different interaction modes into the latent space. We proposed a novel approach to address the KL vanishing during training and induce an interpretable interactive latent space with pseudo labels. The proposed pseudo-labels allowed us to flexibly incorporate domain knowledge on interaction and provide diverse predictions with finite candidates.

In Chapter 4, a styled generative model RouteGAN was proposed to generate diverse interactions by controlling the vehicles separately with desired styles. By altering its style coefficients, the model can generate trajectories with different safety levels and serve as an online planner. We evaluated different planners with our model by testing their collision rate in interaction with RouteGAN planners of multiple critical levels. It's one of the first few works to utilize controllable trajectory generation for planning and planning evaluation.

In Chapter 5, the interaction generation problem was extended to indoor scenarios. Human-like interactions in this realistic setting were modeled, with each agent acting based on its observation and not communicating with others. We proposed a framework based on distributed potential games, where each agent imagines a cooperative game with other agents and solves the game using its estimation of their behavior. We demonstrated the benefits of utilizing distributed imagined games in our framework through various simulation

experiments. The work is one of the first to discuss interaction generation in extreme cases requiring collaboration.

In Chapter 6, a policy subspace represented by a set of parameters was learned. Policies for all the single tasks lie in this subspace and can be composed by interpolating with the learned set. We demonstrated the state-of-the-art performance on Multi-task reinforcement learning benchmarks in Meta-World, which contains various robotic manipulation tasks. In addition, we showed our initial attempt at extending the Parameter-Compositional structure to unseen tasks in a continual setting.

In Chapter 7, we presented an interactive planning technique for partially observable tasks using LLMs. In the proposed method, an LLM was used to collect missing information from the environment using a robot and infer the state of the underlying problem from collected observations while guiding the robot to perform the required actions. We also used a fine-tuned Llama 2 model via self-instruct and compared its performance against a pre-trained LLM like GPT-4. Results were demonstrated on several tasks in simulation as well as real-world environments.

In Chapter 8, we implemented an adaptive energy reward function. We adjusted the weights based on velocity for locomotion tasks on ANYmal-C and Unitree Go1 robots to autonomously select appropriate gaits with improved energy efficiency and stable velocity tracking compared to previous methods, which used complex reward designs and prior gait knowledge. The effectiveness of our policy was validated through simulations in the Isaac-Gym simulation environment and on real robots, demonstrating its potential to facilitate stable and adaptive locomotion.

In Chapter 9, we introduced the contrastive grasp proposal network (CGPN), a novel approach for generating robust 6-DoF grasps. By leveraging contrastive learning and variant depth image processing, CGPN effectively bridges the sim-to-real gap, achieving superior performance over existing grasping algorithms in real-world scenarios.

There are a lot of directions we plan to extend based on the research presented in this dissertation. A direct extension to the distributed interaction generation framework introduced in 5 is to build a simulation environment where a distributed potential game solver controls each of the agents in the simulation. On the one hand, this environment can simulate diverse interactions in arbitrary new environments by setting different initial conditions and interaction parameters of the agents. On the other hand, such a simulation environment can be used as a testing benchmark to train an interactive policy using reinforcement learning and evaluate whether an existing policy can interact with the reactive agents well. Another extension we are conducting is to extend Chapter 7 to a more practical and general manipulation task setting, where the robot will face failures in deployment due to multiple reasons. Interactive planning using LLMs and online text retrieval can potentially recover the robot from failure using online feedback. With the help of more advanced foundation models, we also hope to solve more generalized and complex manipulation tasks.

Bibliography

[1] Michael Ahn et al. *Do As I Can, Not As I Say: Grounding Language in Robotic Affordances*. 2022. arXiv: 2204.01691 [cs.RO].

[2] Jacob Andreas, Dan Klein, and Sergey Levine. "Modular Multitask Reinforcement Learning with Policy Sketches". In: *Proceedings of the 34th International Conference on Machine Learning*. 2017.

[3] Rohan Anil et al. *PaLM 2 Technical Report*. 2023. arXiv: 2305.10403 [cs.CL].

[4] Himani Arora et al. *Multi-task Learning for Continuous Control*. 2018. arXiv: 1802.01034 [cs.LG].

[5] P. Bao, L. Zhang, and X. Wu. "Canny Edge Detection Enhancement by Scale Multiplication". In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 27.09 (2005), pp. 1485–1490.

[6] Yvonne Barnard et al. "Methodology for Field Operational Tests of Automated Vehicles". In: *Transportation Research Procedia* 14 (Dec. 2016), pp. 2188–2196.

[7] André Barreto et al. "Transfer in Deep Reinforcement Learning Using Successor Features and Generalised Policy Improvement". In: *International Conference on Machine Learning*. 2018.

[8] Sumeet Batra et al. *Decentralized Control of Quadrotor Swarms with End-to-end Deep Reinforcement Learning*. 2021. arXiv: 2109.07735 [cs.RO].

[9] Jur van den Berg et al. "Reciprocal n-Body Collision Avoidance". In: *Robotics Research*. Ed. by Cédric Pradalier, Roland Siegwart, and Gerhard Hirzinger. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 3–19.

[10] Homanga Bharadhwaj et al. "RoboAgent: Towards Sample Efficient Robot Manipulation with Semantic Augmentations and Action Chunking". In: *arxiv* (2023).

[11] Julian Bock et al. "Self-learning Trajectory Prediction with Recurrent Neural Networks at Intelligent Intersections". In: Jan. 2017, pp. 346–351.

[12] Annette Burden, Richard Burden, and J. Faires. *Numerical Analysis, 10th edition*. 2016. ISBN: 1305253663.

[13] Daniele Calandriello, Alessandro Lazaric, and Marcello Restelli. "Sparse Multi-Task Reinforcement Learning". In: *Advances in Neural Information Processing Systems*. 2014.

[14] M.P. do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice-Hall, 1976. ISBN: 9780132125895.

[15] Rich Caruana. "Multitask Learning". In: *Mach. Learn.* 28.1 (1997), pp. 41–75.

[16] Ming-Fang Chang et al. "Argoverse: 3D Tracking and Forecasting With Rich Maps". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019*. Computer Vision Foundation / IEEE, pp. 8748–8757.

[17] Wei-Jer Chang et al. "Editing Driver Character: Socially-Controllable Behavior Generation for Interactive Traffic Simulation". In: *IEEE Robotics and Automation Letters* 8.9 (2023), pp. 5432–5439.

[18] Baiming Chen and Liang Li. "Adversarial Evaluation of Autonomous Vehicles in Lane-Change Scenarios". In: *arXiv preprint, arXiv:2004.06531* (2020).

[19] Jianhui Chen, Jiayu Zhou, and Jieping Ye. "Integrating Low-Rank and Group-Sparse Structures for Robust Multi-Task Learning". In: *International Conference on Knowledge Discovery and Data Mining*. 2011.

[20] Shuxiao Chen et al. "Learning torque control for quadrupedal locomotion". In: *IEEE-RAS International Conference on Humanoid Robots*. 2023.

[21] Ting Chen et al. "A Simple Framework for Contrastive Learning of Visual Representations". In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*. 2020.

[22] Tongtong Chen et al. "Gaussian-Process-Based Real-Time Ground Segmentation for Autonomous Land Vehicles". In: *Journal of Intelligent and Robotic Systems* 76 (2014), pp. 563–582.

[23] Xi Chen et al. "InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems (NeurIPS), 2016*, pp. 2172–2180.

[24] Zhao Chen et al. "GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks". In: *International Conference on Machine Learning*. 2018.

[25] Xuxin Cheng, Ashish Kumar, and Deepak Pathak. "Legs as Manipulator: Pushing Quadrupedal Agility Beyond Locomotion". In: *IEEE International Conference on Robotics and Automation*. 2023.

[26] C Chevallereau and Y Aoustin. "Optimal reference trajectories for walking and running of a biped robot." In: *Robotica* 19.5 (2001), pp. 557–569.

[27] Suyoung Choi et al. "Learning quadrupedal locomotion on deformable terrain". In: *Science Robotics* 8.74 (2023), eade2256.

[28] Ji-wung Choi, Renwick Curry, and Gabriel Elkaim. "Path planning based on bézier curve for autonomous ground vehicles". In: *Advances in Electrical and Electronics Engineering-IAENG Special Edition of the World Congress on Engineering and Computer Science 2008*. IEEE, pp. 158–166.

[29] Fu-Jen Chu, Ruinian Xu, and Patricio A. Vela. "Real-World Multiobject, Multigrasp Detection". In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 3355–3362.

[30] Simon Le Cleac'h, Mac Schwager, and Zachary Manchester. *ALGAMES: A Fast Augmented Lagrangian Solver for Constrained Dynamic Games*. 2021. arXiv: 2104.08452 [cs.RO].

[31] Carlo D'Eramo et al. "Sharing Knowledge in Multi-Task Deep Reinforcement Learning". In: *International Conference on Learning Representations*. 2020.

[32] Xingye Da et al. "Learning a Contact-Adaptive Controller for Robust, Efficient Legged Locomotion". In: *Conference on Robot Learning*. 2020.

[33] Nachiket Deo and Mohan M Trivedi. "Trajectory forecasts in unknown environments conditioned on grid-based plans". In: *arXiv preprint, arXiv:2001.00735* (2020).

[34] Arjav Desai and Nathan Michael. "Online Planning for Quadrotor Teams in 3-D Workspaces via Reachability Analysis On Invariant Geometric Trees". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 8769–8775.

[35] Coline Devin et al. "Learning modular neural network policies for multi-task and multi-robot transfer". In: *International Conference on Robotics and Automation*. 2017.

[36] Wenhao Ding, Wenshuo Wang, and Ding Zhao. "A Multi-Vehicle Trajectories Generator to Simulate Vehicle-to-Vehicle Encountering Scenarios". In: *International Conference on Robotics and Automation (ICRA), 2019*. IEEE, pp. 4255–4261.

[37] Wenhao Ding, Mengdi Xu, and Ding Zhao. "CMTS: A Conditional Multiple Trajectory Synthesizer for Generating Safety-Critical Driving Scenarios". In: *IEEE International Conference on Robotics and Automation (ICRA), 2020*. IEEE, pp. 4314–4321.

[38] Wenhao Ding et al. "Learning to Collide: An Adaptive Safety-Critical Scenarios Generating Method". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020*. IEEE, pp. 2243–2250.

[39] Wenhao Ding et al. "Multimodal Safety-Critical Scenarios Generation for Decision-Making Algorithms Evaluation". In: *arXiv preprint, arXiv:2009.08311* (2020).

[40] Piotr Dollár et al. "Pedestrian Detection: An Evaluation of the State of the Art". In: *PAMI* 34 (2012).

[41] Alexey Dosovitskiy et al. "Discriminative Unsupervised Feature Learning with Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. 2014.

[42] Danny Driess et al. *PaLM-E: An Embodied Multimodal Language Model*. 2023. arXiv: 2303.03378 [cs.LG].

[43] Helei Duan et al. "Sim-to-real learning of footstep-constrained bipedal dynamic walking". In: *IEEE International Conference on Robotics and Automation*. 2022.

[44] Scott Ettinger et al. "Large Scale Interactive Motion Forecasting for Autonomous Driving: The Waymo Open Motion Dataset". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 9710–9719.

[45] Yongxiang Fan, Xinghao Zhu, and Masayoshi Tomizuka. "Optimization Model for Planning Precision Grasps with Multi-Fingered Hands". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 1548–1554.

[46] Yongxiang Fan et al. "Grasp planning for customized grippers by iterative surface fitting". In: *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*. 2018.

[47] Yongxiang Fan et al. "Real-time grasp planning for multi-fingered hands by finger splitting". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 4045–4052.

[48] Hao-Shu Fang et al. *RH20T: A Robotic Dataset for Learning Diverse Skills in One-Shot*. 2023. arXiv: 2307.00595 [cs.RO].

[49] Zhijie Fang and Antonio M. López. "Is the Pedestrian going to Cross? Answering by 2D Pose Estimation". In: *2018 IEEE Intelligent Vehicles Symposium, IV 2018, Changshu, Suzhou, China, June 26-30, 2018*. 2018, pp. 1271–1276.

[50] Gilbert Feng et al. "Genloco: Generalized locomotion controllers for quadrupedal robots". In: *Conference on Robot Learning*. PMLR. 2023.

[51] Jaime F. Fisac et al. "Hierarchical Game-Theoretic Planning for Autonomous Vehicles". In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 9590–9596.

[52] Alejandra Fonseca-Morales and Onésimo Hernández-Lerma. "Potential Differential Games". In: *Dynamic Games and Applications* 8.2 (2018), pp. 254–279.

[53] Hao Fu et al. "Cyclical Annealing Schedule: A Simple Approach to Mitigating KL Vanishing". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2019, pp. 240–250.

[54] Zipeng Fu et al. "Minimizing Energy Consumption Leads to the Emergence of Gaits in Legged Robots". In: *Conference on Robot Learning*. 2021.

[55] G Gabrielli. "What price speed? Specific power required for propulsion of vehicles". In: *Mechanical Engineering-CIME* 133.10 (2011), pp. 4–5.

[56] Jiyang Gao et al. "Vectornet: Encoding hd maps and agent dynamics from vectorized representation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 11525–11533.

[57] Peng Gao et al. *LLaMA-Adapter V2: Parameter-Efficient Visual Instruction Model*. 2023. arXiv: `2304.15010 [cs.CV]`.

[58] Andreas Geiger et al. "Vision meets Robotics: The KITTI Dataset". In: *International Journal of Robotics Research (IJRR)* (2013).

[59] Suhyeon Gim et al. "Clothoids composition method for smooth path generation of car-like vehicle navigation". In: *Journal of Intelligent & Robotic Systems* 88.1 (2017), pp. 129–146.

[60] Ross B. Girshick. "Fast R-CNN". In: *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*. 2015.

[61] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The MIT Press, 2016. ISBN: 0262035618.

[62] Ian J. Goodfellow et al. "Generative adversarial networks". In: *Commun. ACM* 63.11 (2020), pp. 139–144.

[63] Junru Gu, Chen Sun, and Hang Zhao. "Densetnt: End-to-end trajectory prediction from dense goal sets". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 15303–15312.

[64] Junru Gu, Qiao Sun, and Hang Zhao. "DenseTNT: Waymo Open Dataset Motion Prediction Challenge 1st Place Solution". In: *arXiv preprint, arXiv:2106.14160* (2021).

[65] Agrim Gupta et al. "Social GAN: Socially Acceptable Trajectories With Generative Adversarial Networks". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018*, pp. 2255–2264.

[66] Tuomas Haarnoja et al. "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor". In: *International Conference on Machine Learning*. 2018.

[67] R. Hadsell, S. Chopra, and Y. LeCun. "Dimensionality Reduction by Learning an Invariant Mapping". In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. 2006.

[68] P. E. Hart, N. J. Nilsson, and B. Raphael. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". In: *IEEE Transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107.

[69] Jouni Hartikainen, Arno Solin, and Simo Särkkä. *Optimal Filtering with Kalman Filters and Smoothers: a Manual for the Matlab toolbox EKF/UKF Version 1.3*.

[70] Y. Hashimoto et al. "A Probabilistic Model for the Estimation of Pedestrian Crossing Behavior at Signalized Intersections". In: *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. 2015.

[71] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. 2016.

[72] Kaiming He et al. "Momentum Contrast for Unsupervised Visual Representation Learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[73] Nicolas Heess et al. "Learning and Transfer of Modulated Locomotor Controllers". In: *CoRR* abs/1610.05182 (2016).

[74] Matteo Hessel et al. *Multi-task Deep Reinforcement Learning with PopArt*. Tech. rep. DeepMind, 2019.

[75] Namgyu Ho, Laura Schmid, and Se-Young Yun. *Large Language Models Are Reasoning Teachers*. 2023. arXiv: `2212.10071 [cs.CL]`.

[76] Yeping Hu et al. "Multi-modal Probabilistic Prediction of Interactive Behavior via an Interpretable Model". In: *2019 IEEE Intelligent Vehicles Symposium*. 2019.

[77] Wenlong Huang et al. *Inner Monologue: Embodied Reasoning through Planning with Language Models*. 2022. arXiv: `2207.05608 [cs.RO]`.

[78] Yingning Huang et al. "Head pose based intention prediction using Discrete Dynamic Bayesian Network". In: *2013 Seventh International Conference on Distributed Smart Cameras (ICDSC)*. 2013, pp. 1–6.

[79] Uk-Youl Huh and Seong-Ryong Chang. "A G2 Continuous Path-smoothing Algorithm Using Modified Quadratic Polynomial Interpolation". In: *International Journal of Advanced Robotic Systems* 11.2 (2014), p. 25.

[80] Mingxiao Huo et al. *Human-oriented Representation Learning for Robotic Manipulation*. 2023. arXiv: `2310.03023 [cs.RO]`.

[81] Marco Hutter et al. "ANYmal - toward legged robots for harsh environments". In: *Advanced Robotics* 31.17 (2017), pp. 918–931.

[82] Jemin Hwangbo et al. "Learning agile and dynamic motor skills for legged robots". In: *Science Robotics* 4.26 (2019), eaau5872.

[83] Boris Ivanovic and Marco Pavone. "The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs". In: *Proceedings of the International Conference on Computer Vision*. 2019, pp. 2375–2384.

[84] Gwanghyeon Ji et al. "Concurrent training of a control policy and a state estimator for dynamic and robust legged locomotion". In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 4630–4637.

[85] Zhiqiang Jian et al. "Long-Term Dynamic Window Approach for Kinodynamic Local Planning in Static and Crowd Environments". In: *IEEE Robotics and Automation Letters* 8.6 (2023), pp. 3294–3301.

[86] Shan Jiang et al. "Single-Grasp Detection Based on Rotational Region CNN". In: *Advances in Computational Intelligence Systems*. 2020.

[87] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. "Planning and acting in partially observable stochastic domains". In: *Artificial Intelligence* 101.1 (1998), pp. 99–134.

[88] Dmitry Kalashnikov et al. "MT-Opt: Continuous Multi-Task Robotic Reinforcement Learning at Scale". In: *CoRR* abs/2104.08212 (2021).

[89] Zhuoliang Kang, Kristen Grauman, and Fei Sha. "Learning with Whom to Share in Multi-task Feature Learning". In: *International Conference on Machine Learning*. 2011.

[90] Talha Kavuncu, Ayberk Yaraneri, and Negar Mehr. *Potential iLQR: A Potential-Minimizing Controller for Planning Multi-Agent Interactive Trajectories*. 2021. arXiv: 2107.04926 [cs.RO].

[91] Alex Kendall, Yarin Gal, and Roberto Cipolla. "Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2018.

[92] Samuel Kessler et al. "Same State, Different Task: Continual Reinforcement Learning without Interference". In: *CoRR* abs/2106.02940 (2021).

[93] Khimya Khetarpal et al. "Towards Continual Reinforcement Learning: A Review and Perspectives". In: *CoRR* abs/2012.13490 (2020).

[94] K Kiguchi et al. "Energy-optimal gait analysis of quadruped robots." In: *Artificial Life and Robotics*. 6.3 (2002), pp. 120–125.

[95] Wonhui Kim et al. "Pedx: Benchmark dataset for metric 3d pose estimation of pedestrians in complex urban intersections". In: *IEEE Robotics and Automation Letters* (2019).

[96] Youngdong Kim et al. "Nlnl: Negative learning for noisy labels". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 101–110.

[97] Diederik P. Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *International Conference on Learning Representations (ICLR), 2014*.

[98] Tim Klinger, Franz Rottensteiner, and Christian Heipke. "A Dynamic Bayes Network for visual Pedestrian Tracking". In: 2014.

[99] Moritz Klischat and Matthias Althoff. "Generating Critical Test Scenarios for Automated Vehicles with Evolutionary Algorithms". In: *IEEE Intelligent Vehicles Symposium, (IV) 2019*. IEEE, pp. 2352–2358.

[100] Kiyoshi Komoriya and Kazuo Tanie. "Trajectory design and control of a wheel-type mobile robot using B-spline curve". In: *Proceedings. IEEE/RSJ International Workshop on Intelligent Robots and Systems.(IROS 1989) The Autonomous Mobile Robots and Its Applications*. IEEE, pp. 398–405.

[101] Julian F. P. Kooij et al. "Context-Based Pedestrian Path Prediction". In: *ECCV*. 2014.

[102] Abhishek Kumar and Hal Daumé. "Learning Task Grouping and Overlap in Multi-Task Learning". In: *International Conference on International Conference on Machine Learning*. 2012.

[103] Forrest Laine et al. *The Computation of Approximate Generalized Feedback Nash Equilibria*. 2022. arXiv: 2101.02900 [math.OC].

[104] Simon Le Cleac'h et al. "Fast Contact-Implicit Model Predictive Control". In: *IEEE Transactions on Robotics* 40 (2024), pp. 1617–1629.

[105] Joonho Lee et al. "Learning quadrupedal locomotion over challenging terrain". In: *Science Robotics* 5.47 (2020), eabc5986.

[106] Stéphanie Lefèvre, Dizan Vasquez, and Christian Laugier. "A survey on motion prediction and risk assessment for intelligent vehicles". In: *ROBOMECH Journal* 1.1 (2014), p. 1.

[107] J. Li, W. Zhan, and M. Tomizuka. "Generic Vehicle Tracking Framework Capable of Handling Occlusions Based on Modified Mixture Particle Filter". In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. 2018.

[108] Jiachen Li, Hengbo Ma, and Masayoshi Tomizuka. "Conditional generative neural system for probabilistic trajectory prediction". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 6150–6156.

[109] Jiachen Li et al. "Generic Probabilistic Interactive Situation Recognition and Prediction: From Virtual to Real". In: *2018 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE. 2018.

[110] Qingbiao Li et al. "Graph Neural Networks for Decentralized Multi-Robot Path Planning". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 11785–11792.

[111] Shuang Li et al. *Pre-Trained Language Models for Interactive Decision-Making*. 2022. arXiv: 2202.01771 [cs.LG].

[112] Zhongyu Li et al. "Reinforcement learning for robust parameterized locomotion control of bipedal robots". In: *IEEE International Conference on Robotics and Automation*. 2021.

[113] Boyuan Liang et al. *Adaptive Energy Regularization for Autonomous Gait Transition and Energy-Efficient Quadruped Locomotion*. 2024. arXiv: 2403.20001 [cs.RO].

[114] Hongzhuo Liang et al. "PointNetGPD: Detecting Grasp Configurations from Point Sets". In: *2019 International Conference on Robotics and Automation (ICRA)* (2019).

[115] Jacky Liang et al. *Code as Policies: Language Model Programs for Embodied Control*. 2023. arXiv: 2209.07753 [cs.RO].

[116] Jacky Liang et al. "GPU-Accelerated Robotic Simulation for Distributed Reinforcement Learning". In: *Conference on Robot Learning*. 2018.

[117] M. Liebner et al. "Driver intent inference at urban intersections using the intelligent driver model". In: *2012 IEEE Intelligent Vehicles Symposium*. 2012.

[118] Xi Lin et al. "Pareto Multi-Task Learning". In: *Thirty-third Conference on Neural Information Processing Systems (NeurIPS)*. 2019, pp. 12037–12047.

[119] Haotian Liu et al. "Visual Instruction Tuning". In: *ArXiv preprint* abs/2304.08485 (2023).

[120] Shikun Liu, Edward Johns, and Andrew J Davison. "End-to-End Multi-task Learning with Attention". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2019.

[121] Carlos E. Luis, Marijan Vukosavljev, and Angela P. Schoellig. "Online Trajectory Generation With Distributed Model Predictive Control for Multi-Robot Motion Planning". In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 604–611.

[122] Z. Luo et al. "Grasp Detection Based on Faster Region CNN". In: *2020 5th International Conference on Advanced Robotics and Mechatronics (ICARM)*. 2020, pp. 323–328.

[123] Hengbo Ma et al. "Wasserstein Generative Learning with Kinematic Constraints for Probabilistic Interactive Driving Behavior Prediction". In: *2019 Intelligent Vehicles Symposium (IV)*. IEEE. 2019.

[124] Jianqi Ma et al. "Arbitrary-Oriented Scene Text Detection via Rotation Proposals". In: *IEEE Transactions on Multimedia* 20.11 (2018), pp. 3111–3122.

[125] Naveen K. Mahato, Axel Klar, and Sudarshan Tiwari. *Particle methods for multi-group pedestrian flow*. 2017. arXiv: `1607.02326 [physics.soc-ph]`.

[126] J. Mahler et al. "Dex-Net 3.0: Computing Robust Vacuum Suction Grasp Targets in Point Clouds Using a New Analytic Model and Deep Learning". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 5620–5627.

[127] Jeffrey Mahler et al. "Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics". In: *Robotics: Science and Systems (RSS)*. 2017.

[128] Viktor Makoviychuk et al. "Isaac Gym: High Performance GPU Based Physics Simulation For Robot Learning". In: *Conference on Neural Information Processing Systems*. 2021.

[129] Karttikeya Mangalam et al. "It is not the journey but the destination: Endpoint conditioned trajectory prediction". In: *European Conference on Computer Vision*. Springer. 2020, pp. 759–776.

[130] Gabriel B. Margolis and Agrawal Pulkit. "Walk These Ways: Tuning Robot Control for Generalization with Multiplicity of Behavior". In: *Conference on Robot Learning*. 2023.

[131] Gabriel B. Margolis et al. "Rapid locomotion via reinforcement learning". In: *The International Journal of Robotics Research* 43.4 (2024), pp. 572–587.

[132] Negar Mehr et al. "Maximum-Entropy Multi-Agent Dynamic Games: Forward and Inverse Solutions". In: *IEEE Transactions on Robotics* 39.3 (2023), pp. 1801–1815.

[133] Takahiro Miki et al. "Learning robust perceptive locomotion for quadrupedal robots in the wild". In: *Science Robotics* 7.62 (2022), eabk2822.

[134] Kristina Miller and Sayan Mitra. "Multi-agent motion planning using differential games with lexicographic preferences". In: *2022 IEEE 61st Conference on Decision and Control (CDC)*. 2022, pp. 5751–5756.

[135] Arsalan Mousavian, Clemens Eppner, and Dieter Fox. "6-DOF GraspNet: Variational Grasp Generation for Object Manipulation". In: *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. 2019.

[136] A. Murali et al. "6-DOF Grasping for Target-driven Object Manipulation in Clutter". In: *International Conference on Robotics and Automation (ICRA)*. 2020.

[137] A Muraro, C Chevallereau, and Y Aoustin. "Optimal trajectories for a quadruped robot with trot, amble and curvet gaits for two energetic criteria." In: *Multibody System Dynamics* 9.1 (2003), pp. 39–62.

[138] Kevin Patrick Murphy. "Dynamic bayesian networks: representation, inference and learning". In: (2002).

[139] Richard M Murray et al. *A mathematical introduction to robotic manipulation*. CRC press, 1994.

[140] Peiyuan Ni et al. "PointNet++ Grasping: Learning An End-to-end Spatial Grasp Generation Algorithm from Sparse Point Clouds". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 3619–3625.

[141] Edwin Olson. "AprilTag: A robust and flexible visual fiducial system". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2011, pp. 3400–3407.

[142] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. *Representation Learning with Contrastive Predictive Coding*. 2019. arXiv: `1807.03748 [cs.LG]`.

[143] OpenAI. *GPT-4 Technical Report*. 2023. arXiv: `2303.08774 [cs.CL]`.

[144] Dongwon Park and Se Young Chun. "Classification based Grasp Detection using Spatial Transformer Network". In: *CoRR* abs/1803.01356 (2018).

[145] Jungwon Park et al. "Efficient Multi-Agent Trajectory Planning with Feasibility Guarantee using Relative Bernstein Polynomial". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 434–440.

[146] Andreas ten Pas et al. "Grasp Pose Detection in Point Clouds". In: *The International Journal of Robotics Research* 36.13-14 (2017), pp. 1455–1473.

[147] Ethan Perez et al. "FiLM: Visual Reasoning with a General Conditioning Layer". In: *CoRR* abs/1709.07871 (2017).

[148] Benedetto Piccoli and Andrea Tosin. "Pedestrian flows in bounded domains with obstacles". In: *Continuum Mechanics and Thermodynamics* 21.2 (2009), pp. 85–107.

[149] Matthias Plappert et al. "Multi-Goal Reinforcement Learning: Challenging Robotics Environments and Request for Research". In: *CoRR* abs/1802.09464 (2018).

[150] Matthias Plappert et al. *Parameter Space Noise for Exploration*. 2018. arXiv: `1706.01905 [cs.LG]`.

[151] Marc H Raibert. "Trotting, pacing and bounding by a quadruped robot". In: *Journal of Biomechanics* 23 (1990), pp. 79–98.

[152] HJ Ralston. "Energy-speed relation and optimal speed during level walking." In: *Internationale Zeitschrift für angewandte Physiologie einschließlich Arbeitsphysiologie* 17.4 (1958), pp. 277–283.

[153] A. Rasouli, I. Kotseruba, and J. K. Tsotsos. "Are They Going to Cross? A Benchmark Dataset and Baseline for Pedestrian Crosswalk Behavior". In: *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*. 2017.

[154] Joshue Perez Rastelli, Ray Lattarulo, and Fawzi Nashashibi. "Dynamic trajectory generation using continuous-curvature algorithms for door to door assistance vehicles". In: *IEEE Intelligent Vehicles Symposium Proceedings, 2014*. IEEE, pp. 510–515.

[155] Abhijeet Ravankar et al. "Path smoothing techniques in robot navigation: State-of-the-art, current and future challenges". In: *Sensors* 18.9 (2018), p. 3170.

[156] Eike Rehder et al. "Pedestrian Prediction by Planning Using Deep Neural Networks". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)* (2018), pp. 1–5.

[157] C David Remy. "Optimal Exploitation of Natural Dynamics in Legged Locomotion." PhD thesis. Eidgenossische Technische Hochschule., 2011.

[158] Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. 2015.

[159] D. Ridel et al. "A Literature Review on the Prediction of Pedestrian Behavior in Urban Scenarios". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018.

[160] Benjamin Rivière et al. "GLAS: Global-to-Local Safe Autonomy Synthesis for Multi-Robot Motion Planning With End-to-End Learning". In: *IEEE Robotics and Automation Letters* 5.3 (2020), pp. 4249–4256.

[161] Maximo Roa and Raul Suarez. "Grasp Quality Measures: Review and Performance". In: *Autonomous Robots* 38 (2014), pp. 65–88.

[162] A. V. I. Rosti and M. J. F. Gales. "Rao-Blackwellised Gibbs sampling for switching linear dynamical systems". In: *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 1. 2004.

[163] Nikita Rudin et al. "Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning". In: *Conference on Robot Learning*. 2021.

[164] Radu Bogdan Rusu and Steve Cousins. "3D is here: Point Cloud Library (PCL)". In: *IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China, 2011.

[165] Tim Salzmann et al. "Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data". In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16*. Springer. 2020, pp. 683–700.

[166] Elad Sarafian, Shai Keynan, and Sarit Kraus. "Recomposing the Reinforcement Learning Building Blocks with Hypernetworks". In: *Proceedings of the 38th International Conference on Machine Learning*. 2021.

[167] Siamak Sarmady, Fazilah Haron, and Abdullah Zawawi Hj. Talib. "Modeling Groups of Pedestrians in Least Effort Crowd Movements Using Cellular Automata". In: *2009 Third Asia International Conference on Modelling and Simulation*. 2009, pp. 520–525.

[168] Guillaume Sartoretti et al. "PRIMAL: Pathfinding via Reinforcement and Imitation Multi-Agent Learning". In: *IEEE Robotics and Automation Letters* 4.3 (2019), pp. 2378–2385.

[169] Nicolas Schneider and Dariu M. Gavrila. "Pedestrian Path Prediction with Recursive Bayesian Filters: A Comparative Study". In: *Pattern Recognition*. Ed. by Joachim Weickert, Matthias Hein, and Bernt Schiele. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 174–183. ISBN: 978-3-642-40602-7.

[170] Jens Schulz et al. "Interaction-Aware Probabilistic Behavior Prediction in Urban Environments". In: *CoRR* abs/1804.10467 (2018).

[171] Baskın Şenbaşlar, Wolfgang Hönig, and Nora Ayanian. *RLSS: Real-time Multi-Robot Trajectory Replanning using Linear Spatial Separations*. 2022. arXiv: `2103.07588 [cs.RO]`.

[172] Ozan Sener and Vladlen Koltun. "Multi-Task Learning as Multi-Objective Optimization". In: *Advances in Neural Information Processing Systems*. 2018.

[173] N. Shafii, S. H. Kasaei, and L. S. Lopes. "Learning to grasp familiar objects using object view recognition and template matching". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016.

[174] Lin Shao et al. "UniGrasp: Learning a Unified Model to Grasp With Multifingered Robotic Hands". In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 2286–2293.

[175] Jonah Siekmann et al. "Sim-to-Real Learning of All Common Bipedal Gaits via Periodic Reward Composition". In: *IEEE International Conference on Robotics and Automation*. 2021.

[176] Satinder Singh. "Transfer of learning by composing solutions of elemental sequential tasks". In: *Machine Learning* 8 (2004), pp. 323–339.

[177] Virginia Smith et al. "Federated Multi-Task Learning". In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017.

[178] Shagun Sodhani, Amy Zhang, and Joelle Pineau. "Multi-Task Reinforcement Learning with Context-based Representations". In: *International Conference on Machine Learning*. 2021.

[179] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. "Learning structured output representation using deep conditional generative models". In: *Advances in neural information processing systems* 28 (2015), pp. 3483–3491.

[180] Baoye Song, Guohui Tian, and Fengyu Zhou. "A comparison study on path smoothing algorithms for laser robot navigated mobile robot path planning in intelligent space". In: *Journal of Information and Computational Science* 7.1 (2010), pp. 2943–2950.

[181] Riccardo Spica et al. "A Real-Time Game Theoretic Planner for Autonomous Two-Player Drone Racing". In: *IEEE Transactions on Robotics* 36.5 (2020), pp. 1389–1403.

[182] Sandeep Subramanian et al. "Learning General Purpose Distributed Sentence Representations via Large Scale Multi-task Learning". In: *International Conference on Learning Representations*. 2018.

[183] Guolei Sun et al. "Task Switching Network for Multi-Task Learning". In: *IEEE International Conference on Computer Vision*. 2021.

[184] L. Sun, W. Zhan, and M. Tomizuka. "Probabilistic Prediction of Interactive Driving Behavior via Hierarchical Inverse Reinforcement Learning". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018, pp. 2111–2117.

[185] Lingfeng Sun et al. *Distributed Multi-agent Interaction Generation with Imagined Potential Games*. 2023. arXiv: `2310.01614 [cs.RO]`.

[186] Lingfeng Sun et al. "Domain Knowledge Driven Pseudo Labels for Interpretable Goal-Conditioned Interactive Trajectory Prediction". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2022, pp. 13034–13041.

[187] Lingfeng Sun et al. "Efficient Multi-Task and Transfer Reinforcement Learning With Parameter-Compositional Framework". In: *IEEE Robotics and Automation Letters* 8.8 (2023), pp. 4569–4576.

[188] Lingfeng Sun et al. "Efficient multi-task and transfer reinforcement learning with parameter-compositional framework". In: *Robotics and Automation Letters* (2023).

[189] Lingfeng Sun et al. *Interactive Planning Using Large Language Models for Partially Observable Robotics Tasks*. 2023. arXiv: `2312.06876 [cs.RO]`.

[190] Lingfeng Sun et al. "Interactive Prediction for Multiple, Heterogeneous Traffic Participants with Multi-Agent Hybrid Dynamic Bayesian Network". In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. 2019, pp. 1025–1031.

[191] Lingfeng Sun et al. "PaCo: Parameter-Compositional Multi-task Reinforcement Learning". In: *NeurIPS*. 2022.

[192] Lingfeng Sun et al. "Paco: Parameter-compositional multi-task reinforcement learning". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 21495–21507.

[193] Liting Sun et al. "Courteous Autonomous Cars". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 663–670.

[194] Liting Sun et al. "Interpretable Modelling of Driving Behaviors in Interactive Driving Scenarios based on Cumulative Prospect Theory". In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. 2019.

[195] Priya Sundaresan et al. *Learning Rope Manipulation Policies Using Dense Object Descriptors Trained on Synthetic Depth Data*. 2020. arXiv: `2003.01835 [cs.RO]`.

[196] Simon Suo et al. "TrafficSim: Learning To Simulate Realistic Multi-Agent Behaviors". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 10400–10409.

[197] Chen Tang, Wei Zhan, and Masayoshi Tomizuka. "Exploring Social Posterior Collapse in Variational Autoencoder for Interaction Modeling". In: *Advances in Neural Information Processing Systems* 34 (2021).

[198] Sarah Tang and Vijay Kumar. "Safe and complete trajectory generation for robot teams with higher-order dynamics". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 1894–1901.

[199] Rohan Taori et al. *Stanford Alpaca: An Instruction-following LLaMA model*. `https://github.com/tatsu-lab/stanford_alpaca`. 2023.

[200] Yi Tay et al. "HyperGrid Transformers: Towards A Single Model for Multiple Tasks". In: *International Conference on Learning Representations*. 2021.

[201] Yee Teh et al. "Distral: Robust multitask reinforcement learning". In: *Advances in Neural Information Processing Systems*. 2017.

[202] Yonglong Tian, Dilip Krishnan, and Phillip Isola. *Contrastive Multiview Coding*. 2020. arXiv: 1906.05849 [cs.CV].

[203] Wouter van Toll, Norman Jaklin, and Roland Geraerts. "Towards Believable Crowds : A Generic Multi-Level Framework for Agent Navigation". In: 2015.

[204] Wouter G. van Toll, Atlas F. Cook IV, and Roland Geraerts. "Real-time density-based crowd simulation". In: *Computer Animation and Virtual Worlds* 23.1 (2012), pp. 59–69.

[205] Jesus Tordesillas and Jonathan P. How. "MADER: Trajectory Planner in Multiagent and Dynamic Environments". In: *IEEE Transactions on Robotics* 38.1 (2022), pp. 463–476.

[206] Hugo Touvron et al. *Llama 2: Open Foundation and Fine-Tuned Chat Models*. 2023. arXiv: 2307.09288 [cs.CL].

[207] Hugo Touvron et al. *LLaMA: Open and Efficient Foundation Language Models*. 2023. arXiv: 2302.13971 [cs.CL].

[208] Martin Treiber and Arne Kesting. "Traffic flow dynamics". In: *Traffic Flow Dynamics: Data, Models and Simulation, Springer-Verlag Berlin Heidelberg* (2013).

[209] VA Tucker. "The energetic cost of moving about: Walking and running are extremely inefficient forms of locomotion. Much greater efficiency is achieved by birds, fish, and bicyclists." In: *American Scientist* 63.4 (1975), pp. 413–419.

[210] BR Umberger and PE Martin. "Mechanical power and efficiency of level walking with different stride rates." In: *Journal of Experimental Biology* 210.18 (2007), pp. 3255–3265.

[211] *Unitree Robotics, Go1*. https://www.unitree.com/products/go1. Online; accessed Jun. 2022.

[212] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[213] Sai Vemprala et al. *ChatGPT for Robotics: Design Principles and Model Abilities*. Tech. rep. MSR-TR-2023-8. Microsoft, 2023.

[214] Eric Vollenweider et al. "Advanced Skills through Multiple Adversarial Motion Priors in Reinforcement Learning". In: *IEEE International Conference on Robotics and Automation*. 2023.

[215] B. Völz et al. "Feature Relevance Estimation for Learning Pedestrian Behavior at Crosswalks". In: *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. 2015.

[216] Homer Walke et al. *BridgeData V2: A Dataset for Robot Learning at Scale*. 2023. arXiv: 2308.12952 [cs.RO].

[217] Changhao Wang, Jeffrey Bingham, and Masayoshi Tomizuka. "Trajectory splitting: A distributed formulation for collision avoiding trajectory optimization". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, pp. 8113–8120.

[218] Changhao Wang et al. "Bpomp: A bilevel path optimization formulation for motion planning". In: *2022 American Control Conference (ACC)*. IEEE. 2022, pp. 1891–1897.

[219] Jen-Wei Wang et al. *A Simple Approach for General Task-Oriented Picking using Placing constraints*. 2023. arXiv: 2304.01290 [cs.RO].

[220] Li Wang, Aaron D. Ames, and Magnus Egerstedt. "Safety Barrier Certificates for Collisions-Free Multirobot Systems". In: *IEEE Transactions on Robotics* 33.3 (2017), pp. 661–674.

[221] Mingyu Wang et al. "Game-Theoretic Planning for Self-Driving Cars in Multivehicle Competitive Scenarios". In: *IEEE Transactions on Robotics* 37.4 (2021), pp. 1313–1325.

[222] Yizhong Wang et al. *Self-Instruct: Aligning Language Model with Self Generated Instructions*. 2022.

[223] Zirui Wang et al. "Gradient Vaccine: Investigating and Improving Multi-task Optimization in Massively Multilingual Models". In: *International Conference on Learning Representations*. 2021.

[224] Jason Wei et al. *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*. 2023. arXiv: 2201.11903 [cs.CL].

[225] Zach Williams, Jushan Chen, and Negar Mehr. "Distributed Potential iLQR: Scalable Game-Theoretic Trajectory Planning for Multi-Agent Interactions". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 2023, pp. 01–07.

[226] W. Wohlkinger et al. "3DNet: Large-scale object class recognition from CAD models". In: *2012 IEEE International Conference on Robotics and Automation*. 2012, pp. 5384–5391.

[227] Weitao Xi, Yevgeniy Yesilevskiy, and C. David Remy. "Selecting gaits for economical locomotion of legged robots". In: *The International Journal of Robotics Research* 35.9 (2016), pp. 1140–1154.

[228] Hui Xu et al. "Exploring Parameter Space with Structured Noise for Meta Reinforcement Learning". In: *International Joint Conference on Artificial Intelligence*. 2020.

[229] Yichong Xu et al. "Multi-task Learning with Sample Re-weighting for Machine Reading Comprehension". In: *Conference of the North American Chapter of the Association for Computational Linguistics*. 2019.

[230] Zhiyuan Xu et al. "Knowledge Transfer in Multi-Task Deep Reinforcement Learning for Continuous Control". In: *Advances in Neural Information Processing Systems*. 2020.

[231] Ruihan Yang et al. "Multi-Task Reinforcement Learning with Soft Modularization". In: *Advances in Neural Information Processing Systems*. 2020.

[232] Yuxiang Yang et al. "Fast and Efficient Locomotion via Learned Gait Transitions". In: *Conference on Robot Learning*. 2021.

[233] Raymond A. Yeh et al. "Diverse Generation for Multi-Agent Sports Games". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[234] Sriram Yenamandra et al. *HomeRobot: Open Vocab Mobile Manipulation*. 2023.

[235] H. Yin and C. Berger. "When to use what data set for your self-driving car algorithm: An overview of publicly available driving datasets". In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. 2017.

[236] Zhao-Heng Yin et al. "Cross domain robot imitation with invariant representation". In: *IEEE International Conference on Robotics and Automation*. 2022, pp. 455–461.

[237] Zhao-Heng Yin et al. "Diverse Critical Interaction Generation for Planning and Planner Evaluation". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021, pp. 7036–7043.

[238] Tianhe Yu et al. "Gradient Surgery for Multi-Task Learning". In: *Advances in Neural Information Processing Systems*. 2020.

[239] Tianhe Yu et al. "Meta-World: A Benchmark and Evaluation for Multi-Task and Meta Reinforcement Learning". In: *Conference on Robot Learning*. 2019.

[240] Wenhao Yu et al. *Language to Rewards for Robotic Skill Synthesis*. 2023. arXiv: 2306.08647 [cs.RO].

[241] Yun Jiang, S. Moseson, and A. Saxena. "Efficient grasping from RGBD images: Learning using a new rectangle representation". In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 3304–3311.

[242] Kevin Zakka et al. "Xirl: Cross-embodiment inverse reinforcement learning". In: *Conference on Robot Learning*. 2022, pp. 537–546.

[243] W. Zhan et al. "Towards a Fatality-Aware Benchmark of Probabilistic Reaction Prediction in Highly Interactive Driving Scenarios". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018.

[244] Wei Zhan et al. *INTERACTION Dataset: An INTERnational, Adversarial and Cooperative moTION Dataset in Interactive Driving Scenarios with Semantic Maps*. 2019. arXiv: 1910.03088 [cs.RO].

[245] Haichao Zhang et al. "Multi-observation visual recognition via joint dynamic sparse representation". In: *IEEE International Conference on Computer Vision*. 2011.

[246] Hanbo Zhang et al. "ROI-based Robotic Grasp Detection for Object Overlapping Scenes". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 4768–4775.

[247] Qichao Zhang et al. "TrajGen: Generating Realistic and Diverse Trajectories With Reactive and Feasible Agent Behaviors for Autonomous Driving". In: *IEEE Transactions on Intelligent Transportation Systems* 23.12 (2022), pp. 24474–24487.

[248] Xiang Zhang et al. "Efficient Sim-to-real Transfer of Contact-Rich Manipulation Skills with Online Admittance Residual Learning". In: *7th Annual Conference on Robot Learning*. 2023.

[249] Yu Zhang and Qiang Yang. "A Survey on Multi-Task Learning". In: *IEEE Transactions on Knowledge and Data Engineering* 34.12 (2022), pp. 5586–5609.

[250] Hang Zhao et al. "Tnt: Target-driven trajectory prediction". In: *arXiv preprint, arXiv:2008.08294* (2020).

[251] Dingjiang Zhou et al. "Fast, On-line Collision Avoidance for Dynamic Vehicles Using Buffered Voronoi Cells". In: *IEEE Robotics and Automation Letters* 2.2 (2017), pp. 1047–1054.

[252] Zikang Zhou et al. "Query-Centric Trajectory Prediction". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023.

[253] Deyao Zhu et al. "MiniGPT-4: Enhancing Vision-Language Understanding with Advanced Large Language Models". In: *ArXiv preprint* abs/2304.10592 (2023).

[254] Xinghao Zhu et al. "6-DoF Contrastive Grasp Proposal Network". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021.

[255] Chengxu Zhuang, Alex Lin Zhai, and Daniel Yamins. "Local Aggregation for Unsupervised Learning of Visual Embeddings". In: *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019*. 2019.

[256] Ziwen Zhuang et al. "Robot Parkour Learning". In: *Conference on Robot Learning*. 2023.