NORTHWESTERN
UNIVERSITY

Computer Science Department

Technical Report
Number: NU-CS-2026-34

June, 2026

Building Embodied AI End-to-End: From Methods to Systems

Guo Ye

Abstract

Embodied artificial intelligence aims to build agents that perceive, reason, and act in the physical world. The last decade has produced extraordinary methodological advances, yet a working embodied agent is more than a method: it is a method delivered through the infrastructure that turns a learned policy into reliable behavior on a real robot. This dissertation argues that building embodied AI is fundamentally an end-to-end discipline in which methodological advances and systems engineering must be co-designed, and traces that argument across four sole-first-author works spanning multi-agent belief-embedded communication, visual-prompt quadrupedal locomotion, cloud robot deployment, and a touch-aware vision–language–action model for contact-rich manipulation.

Keywords

Embodied AI · Robot Learning · Cloud Robotics · Sim-to-Real

Acknowledgements

I would like to thank everyone—my advisor, committee members, collaborators, friends, and family—who supported me along this journey. It has been hard, with its share of ups and downs, and without their consistent love and help I could not have finished this dissertation.

First, I am deeply grateful to my advisor, Professor Han Liu, for years of patient guidance, intellectual freedom, and unwavering trust. His breadth of vision and insistence on rigor shaped the way I think about research. He is extraordinarily hard-working and sharp on a remarkable range of topics, and he carries that sharpness with a sense of humor that is at once honest and enlightening. He is the kind of person I aspire to be.

I thank my dissertation committee—Prof. Han Liu (chair), Prof. Qi Zhu, Prof. Zhaoran Wang, and Prof. Manling Li—for their time, their careful reading, and their incisive questions, all of which have made this work substantially stronger. I am also grateful to Professor Matt Elwin, the director of the Master of Science in Robotics (MSR) program. Without his kind help, I would not have had access to so much robotics hardware, nor come to know so many MSR students—intelligent, hard-working, and skilled—who have since become close friends. I genuinely enjoyed my time at the Center for Robotics and Biosystems (CRB).

I am fortunate to have worked alongside an exceptional group of collaborators and co-authors. I began my robotics journey with Qinjie Lin during my master's at Northwestern;

we were both deeply enthusiastic about everything robotic. We shared the vision that robotics would become one of the most important research areas of the coming decade—a prediction that has since been borne out, and that we are now part of. Our first collaboration impressed Professor Liu and led to my Ph.D. offer; over the following years, we worked together extensively and built the robotic system that later became the backbone of MagicsOS. I also thank Dr. Shihan Lu for introducing me to the state of the art in vision–language–action models, with whom I explored a great deal on the tactile side of VLAs. Beyond them, I thank my other labmates and collaborators Haoran Lu, Tzung-Han Juang, Zening Luo, Muye Jia, Biswa Sengupta, Tim Tsz-Kit Lau, Zhihan Zhou, Yukun Ma, Jiayi Wang, Hongyu Chen, Jerry Yao-Chieh Hu, Shang Wu, Weimin Wu, Chenwei Xu, Haozheng Luo, Mattson Thieme, Ammar Gilani, Dennis Wu, Yibo Wen, Jihai Zhao, Xu Zhao, Zexi Zhang, and many others. The ideas shared at the lab and at our weekly group meetings shaped much of what is in this dissertation.

To my close friends—Zihao Wang, Qiankai Cao, Binglu Wang, Shinan Wang, Zhaoying Liu, Moqiu Cheng, Rui Pan, Symone Qin,Yifang Wang, and Duruo Li—and my roommates Junhui Li, Pengjia Song, Zhihan Liu, and Junxiu Tang: without you, life during the Ph.D. would not have been nearly as colorful or meaningful.

Finally, to my mother Tingyan Lu and my father Fazhi Ye, words cannot express how grateful I am. My mother devoted all her time to raising me and caring for me, witnessing every milestone all the way to this doctoral degree. Her devotion is worth more than anything I can put into words. She is also remarkably kind to everyone she meets, a quality I am still learning from her. My father works tirelessly; when I was young he often came home late, and it was his hard work that made it possible for me to study

abroad. His life experience has taught me a great deal. Their unconditional love is the reason I always know there is a place I can return to and rest.

Table of Contents

ABSTRACT 3

Acknowledgements 5

Table of Contents 8

List of Tables 11

List of Figures 13

Chapter 1. Introduction 21

1.1. From Disembodied to Embodied Artificial Intelligence 21

1.2. Methods and Systems 22

1.3. Thesis Statement 23

1.4. Dissertation Organization 23

Chapter 2. Method 25

2.1. Learning to Infer Belief Embedded Communication 25

2.1.1. Introduction 26

2.1.2. Related Work 28

2.1.3. Background 31

2.1.4. Belief Embedded Communication 33

2.1.5. Experiment 37

2.1.6. Conclusion 44

2.2. Visual Prompt via Pre-trained Model for Quadrupedal Locomotion 45

2.2.1. INTRODUCTION 46

2.2.2. Related Work 49

2.2.3. Vision Prompt for Adapting Locomotion 50

2.2.4. Experiments and Results 55

2.2.5. Conclusion 61

Chapter 3. System 63

3.1. DOS: A Deployment Operating System for Robots 63

3.1.1. INTRODUCTION 64

3.1.2. RELATED WORK 67

3.1.3. System Design 68

3.1.4. Evaluation 72

3.1.5. Conclusion 78

Chapter 4. Synthesis 80

4.1. DreamTacVLA: Learning to Feel the Future for Contact-Rich Manipulation 80

4.1.1. Introduction 81

4.1.2. Related Work 85

4.1.3. Methodology 87

4.1.4. Experiments 93

4.1.5. Conclusion 101

Chapter 5. Conclusions and Future Work 102

5.1. Summary 102

5.2. Reflection: Why End-to-End? 103

5.3. Limitations 104

5.4. Future Work 105

5.5. Closing Remarks 106

References 107

Appendix A. Supplementary Materials for Chapter 4 122

List of Tables

2.1 Settings of different versions of the PredatorPrey environment 39

2.2 Settings of different versions TrafficJunction environment 41

2.3 Simulations across three different types of environments: Pyramid Stairs, Sparse Discrete Obstacle, and Dense Discrete Obstacle. We evaluate the performance of our models based on three metrics: Distance Covered, Number of Collisions, and Success Rate (SR). The results shows the performance of a State-Only Model, several Prompt-Tuning (ProT) models, and pre-trained (PreT) models. 56

3.1 The following table presents a comparison of inference time for pure edge, pure-cloud, and DOS® system. Inference time is quantified in milliseconds, and it is determined by executing a pre-trained model on the respective platforms. The pure-edge platform involves running inference on a `Turtelbot 3` equipped with on-board computer Raspberry Pi 3, while pure-cloud involves running inference on an EC2 instance with GPU. DOS® system combines the two by offloading inference tasks to the cloud while sending back results to the edge. The table includes average inference time and standard deviation for each platform. 73

4.1 Task Success Rates (%) in Real-world (100 trials). Results are reported as mean ± standard deviation over 3 runs. 94

5.1 Summary of contributions across the four works. 103

List of Figures

2.1 Motivation: To successfully achieve a cooperation based hunting task, the wolves are required to 1) compose their howl stating their strategies to surround the lamb and 2) for other wolves to correctly understand such a strategy being communicated by members of its pack. 27

2.2 The communication architecture under a multi-agent setting. Suppose we have N agents, each of them at time t receives the observation o_t and the communication message c_t. The message is reconstructed by a VAE component which takes in and sums up all the other agents' hidden state. 29

2.3 The variable flow graph of Message and policy generation module. The strip shaped bars and squares are the vectors flow in and out the modules. The message pass through the IBM (Intention Belief Module) to get the predictions of other agent's intention named x_{t+1}. C module then utilize the predictions and agent's own hidden state to generate next timestep's communication vector c_{t+1}. The policy module add in observation and passing through a RNN and policy network outputing action a_{t+1}. 34

2.4 Environments used in our experiment. Left is PredatorPrey Env. The blue circles with numbers represent the predators and the red one is the prey. The azure blocks around the predator illustrate their (limited) visual inputs. The default grid size is 5×5. In our setting, the grid size, number of agents and vision are changeable according to the different difficulty settings. The center is TrafficJunction Env. Each agent represented by colored circles need to pass the crossroads without collision. Their actions are limited to *gas* and *brake*. The right is Level-based Foraging, the largest difference of this environment and PredatorPrey is that it requires a level verification while collecting the targets. 37

2.5 The learning curves of IEC on different difficulty levels of the PredatorPrey Env. It shows that as environment becomes more complicated, the variance notably increases. 39

2.6 Performance of IEC and other baselines listed in 2.1.5.1 during the training protocol. The y-axis is normalized returns and x-axis is the total time steps the algorithm takes. As shown in three subplots, IEC can learn the task faster and achieve higher or equal returns by the end. 40

2.7 The learning curves of IEC on different difficulty level of TrafficJunction Env. 42

2.8 Learning curves with different bits (128,64,32) respectively in three environments. This figure demonstrates that the bit-length (capacity)

of the communication vector – which represents the amount of information it can carry – has distinct influence on learning speed. The size of this influence varies across different tasks. 42

2.9 The blue bars are all function enabled IEC with communication vector of size 128 bits. The orange bar is the returns the algorithm can achieve after disabling the corresponding feature. 43

2.10 The two large black-framed blocks are pre-trained model which means their parameters will keep frozen during training. The other light-blue blocks represents tunable parameters. The system integrates proprioceptive states with vision inputs through a large pre-trained vision model. The proprioceptive encoder processes state information, while the depth encoder captures depth information and transform it to required tensor size. The RGB image go directly into prompt tuning module, these features are combined using prompt tuning to enhance the large pre-trained model's visual understanding. The vision features is then passed to another prompt tuning module specially designed for Control Encoder, the output informs a reinforcement learning (RL) policy to generate motor torques for controlling the quadrupedal robot. 46

2.11 Simulation on different environments with ascending difficulty level. a) Pyramid stairs terrain with random rough surfaces; b) Discrete and sparse rectangular obstacles; c) More dense gaps and high wall obstacles. 52

2.12 Real-world deployment on `Unitree GO1` with varying difficulties. a): Robot walks on even terrain with little disturbance. b): Robot walks over low obstacles. c): Robot faces much larger obstacles and accomplish collision avoidance. 58

3.1 This figure shows DOS® deploy a collision-free navigation algorithm to a swarm of robots. (Turtlebot Burger). The right-top figure is the Operating page where viewing from the coach robot (Pyrobot). User can manually move the robot by the joystick the page provides. 65

3.2 The overall architecture of the system. In our vision, the robot will act as data collector and command executor. Both model training and inference are accomplished at cloud. To achieve such goal, DOS® system built fully on cloud but has ability to interact with external simulators and real-world. The system contains an orcherstrator managing the whole CI/CD pipeline, a bridging tool named DOS® Connect and a monitoring component we call it Analytic Profiler which examine the performance of the model and system. 65

3.3 This figure shows how the DOS® is built on AWS and main instances containing their corresponding functionalities. Each component in pipeline is initialized as one AWS EC2 instance. The data flows in the same Elastic File System (EFS) on AWS that bound to all of them. With `DOS® Connect` installed, user computer, robots and edge devices

are connected to the cloud through a gateway instance that facilitate the communication between the edge and the cloud. The real-time data transfer is achieved by the customized S3 bucket API. 70

3.4 Analytical profiler first retrieves required messages from log EFS and host utilities provided by DOS® system. Based on user-defined rules, it then picks a candidate model from model registry and use canary release to update new policy to the whole fleet. 72

3.5 After the successful deployment of the models, we test the system's performance based on the total number of "too close" warnings and odometry data gathered by robots. In our navigation algorithm, we count a warning number while the minimum value in one frame of laser scan is less than 0.2m. The odometry data is the distance robot traveled in 5 minutes. Using the analytical profiler in DOS®, the different models can be seamlessly switched by certain rules. (c) indicates the exact moment and the total number of warnings when the training job is trigger by scheduler. The orange star is the training with simulator and the blue circle represents real-world deployment. Map in (d) is reconstructed from the real test environment. The green box represents the turtlebot with the same range of scan. 76

3.6 (a) shows the deployment page. Once users finish the customized modules, use the red button shown at the right-bottom to deploy the policy automatically. (b) and (c) are monitoring pages of a simulated mobile and quadrupedal robot deployment. The monitoring page displays data from

analytical profiler like odometry and joint positions. (d) is the deployment to a swarm of real world turtlebots. (e) is the operating page, the view occupying most region is from the camera from a pyrobot[**80**], there are also depth camera view and control joysticks on the right side. and (f) are the deployment to the `Unitree Go1` quadrupedal robot. 76

4.1 Hybrid tactile dataset and the Tactile-DreamVLA inference mechanism. (Top) We collect a large-scale tactile dataset covering 4 manipulation tasks and 9 objects, totaling 2M tactile frames. (Bottom) Our Think–Dream–Act loop executes each step of the policy in two passes. In the **Think** stage, the policy proposes a draft action using the current state and a null tactile prediction. In the **Dream** stage, a frozen V-JEPA2 world model forecasts the tactile outcome of that draft action. In the **Act** stage, the policy integrates both the real observation and the predicted tactile feedback to refine the action. This enables fine-grained corrections for contact-rich manipulation. 82

4.2 The proposed framework operates in two stages. **Stage 1 (Left):** A multimodal encoder E_ψ processes diverse inputs. This stage employs Hierarchical Spatial Alignment (HSA) to effectively fuse the features from different modalities, guided by the $\mathcal{L}_{HSA}$ and $\mathcal{L}_W$ losses. A policy π_θ is trained to output an initial draft action $a^{(t)}_{\text{draft}}$. **Stage 2 (Right):** A world model W_ϕ is trained to predict future tactile image sequences. The policy "dreams" the future tactile feeling (e.g., $H^{(t+N)}_{\text{dream}}$) that would result from its draft action. This predicted future is fed

into an MLP, allowing the policy to refine its plan and output a more robust final action $a_{\text{final}}^{(t)}$. 87

4.3 The three-scale visual hierarchy of our model. Our framework fuses information from three distinct visual modalities. Our Hierarchical Spatial Alignment (HSA) loss is designed to explicitly ground the micro-vision (what the robot feels) within the local and macro visual contexts (what the robot sees). 89

4.4 Visualization of the world model's predicted future-state embedding H_{dream} across training. Initially, the embedding is noisy and unstructured, indicating weak predictive ability. As training advances, the embedding becomes increasingly concentrated and stable, revealing that the world model is learning a coherent representation of future tactile–visual dynamics. 91

4.5 Task suite used to evaluate DreamTacVLA. From left to right: Peg-in-Hole, USB Insert, Gear Assembly, and Tool Stabilization. Each task demands precise, contact-rich manipulation, including aligning tight tolerances, detecting slip, or maintaining stable tool contact. It provides a comprehensive benchmark for assessing tactile-aware policies. 93

4.6 The dataset consists of 80% simulated demonstrations and 20% real-world demonstrations, each containing four task categories: Peg-in-Hole, USB Insert, Gear Assembly, and Tool Stabilization. Blue

segments represent simulated data, while orange segments denote real-world data. 97

4.7 Qualitative comparison of our model's tactile prediction. For both the Peginhole and Tool Stabilization tasks, we visualize the sequence (left to right) comparing our model's Prediction (bottom row) to the Ground Truth tactile data (fourth row). The corresponding tactile images are provide as well. 98

4.8 Ablation studies on model and data scaling. 100

CHAPTER 1

Introduction

1.1. From Disembodied to Embodied Artificial Intelligence

The last decade of artificial intelligence has been defined by an extraordinary string of successes in the *digital* world. Large language models compose essays, write code, and answer questions across nearly every domain of human knowledge. Vision systems classify images, generate photoreal scenes from text, and segment videos with superhuman precision. Yet for all this progress, an AI that can *open a door, walk across a room, find the right tool, and use it* remains a remarkable rarity.

The discrepancy is not an accident. The digital world is forgiving: inputs and outputs are tokens, mistakes have no physical consequence, and data can be replayed, rewound, and scaled at will. The physical world is not forgiving. An embodied agent must perceive a scene under shifting lighting, reason about objects that occlude one another, choose actions whose consequences cannot be undone, and recover from sensor noise and hardware failure—all while operating under a real-time budget on a real machine. The contrast between digital AI's near-magical fluency and embodied AI's everyday brittleness is the central engineering challenge of our era of machine learning.

This dissertation asks a deceptively simple question: *what does it actually take to build an embodied agent that works—end to end?* The answer that emerges across four works is that no single advance suffices. A better policy that lives only in a notebook is a paper,

not a robot. A clever simulator that does not transfer is an exhibit, not training data. A solid software stack wrapped around a brittle decision rule is engineering, not science. Real progress demands the co-design of the *methods* that decide behavior and the *systems* that turn those decisions into reliable behavior on real hardware.

1.2. Methods and Systems

The central thread of this dissertation is the relationship between methods and systems in embodied AI.

Methods. By *methods* I mean the science of behavior: the learning algorithms, model architectures, training procedures, and inductive biases that determine what an agent does. The methods studied in this thesis span multi-agent reinforcement learning with emergent communication, decision transformers prompted by large pre-trained vision models, and vision–language–action models augmented with tactile world models. Methodological contributions are the kinds of advances that journals and conferences typically reward—a new loss, a new architecture, a new training recipe—but on their own they do not produce a working robot.

Systems. By *systems* I mean the infrastructure that makes a method real: the simulators that generate training experience at scale, the data pipelines that capture and replay it, the cloud and on-board software that deploys policies to robot fleets, and the hardware itself. The systems in this thesis evolve from lightweight multi-agent simulators, through GPU-parallel rigid-body physics deployed on a quadruped, to a cloud-based fleet platform with continuous integration and continuous deployment, to an end-to-end data

engine that closes the loop from a tactile digital twin to a physical manipulator. Systems contributions are the kinds of work that quietly determine which methods are even tractable, yet are often invisible in published papers.

The argument of this thesis. The argument I will make across the four works is that methodological breakthroughs realize their potential only when paired with the systems infrastructure that makes them deployable, scalable, and reliable. Equivalently, the systems we build determine which methodological questions can be asked at all. Building embodied AI is, in this sense, an end-to-end discipline rather than a sequence of handoffs from algorithm researchers to systems engineers to robot operators.

1.3. Thesis Statement

Building embodied agents is fundamentally an end-to-end discipline. Methodological advances realize their potential only when co-designed with the systems infrastructure that makes them deployable, scalable, and reliable on real hardware. The four works in this dissertation are concrete demonstrations of this co-design, moving from multi-agent learning and visual-prompt locomotion, through cloud-scale robot deployment, to a touch-aware vision–language–action model for contact-rich manipulation.

1.4. Dissertation Organization

The body of this dissertation is organized into three technical chapters that move from methodological foundations to a full-stack synthesis. Chapter 2 (*Method*) presents two methodological advances in reinforcement learning for embodied agents: an inferring beliefs framework for multi-agent communication, and a visual-prompt architecture for

quadrupedal locomotion that is trained at scale in GPU-parallel simulation and deployed on a real quadruped. Chapter 3 (*System*) presents DOS®, a deployment operating system that brings CI/CD to data-driven robotics and turns one-off robot deployments into a continuous fleet-scale loop. Chapter 4 (*Synthesis*) presents DreamTacVLA, a touch-aware vision–language–action model that unifies methodological and systems innovation for contact-rich manipulation, supported by a high-fidelity tactile digital twin and an end-to-end data engine. Chapter 5 reflects on the end-to-end argument, discusses limitations, and outlines future work.

A note on prior work. The thread of multi-agent and adversarial reasoning that motivates the first work in Chapter 2 traces back to my earlier ICRA 2020 work on collision-free navigation via Markov games [**129**], in which the environment around a mobile robot was modelled as a multi-agent game and trained against path-following adversaries. That paper is co-first-authored and is therefore not the focus of any chapter here, but it is the conceptual ancestor of the work presented in Section 2.1.

A note on the chapters. Each technical chapter (Chapters 2–4) incorporates content from one or two of my sole-first-author publications. Where appropriate, the chapters preserve the structure, equations, figures, and tables of the original publications; the corresponding bibliographic entries appear in the master bibliography at the end of this dissertation.

CHAPTER 2

Method

This chapter presents two methodological advances in reinforcement learning for embodied agents. The first work (Section 2.1) introduces an Inferring Beliefs Module and a Communication Module that together let multiple agents learn to coordinate by inferring one another's intentions through emergent communication, validated across three multi-agent cooperative benchmarks. The second work (Section 2.2) introduces a visual-prompt architecture that adapts large pre-trained vision transformers to quadrupedal locomotion via parameter-efficient prompt and prefix tuning, trained at scale in GPU-parallel simulation and deployed on a Unitree Go1.

These two works are unified by a single methodological thread: *policy capability is unlocked by the right inductive bias plus a training ground rich enough to exercise it*—an insight that becomes the substrate for Chapters 3 and 4.

2.1. Learning to Infer Belief Embedded Communication

> In multi-agent collaboration problems with communication, an agent's ability to encode their intention and interpret other agents' strategies is critical for planning their future actions. This paper introduces a novel algorithm called Intention Embedded Communication (IEC) to mimic an agent's language learning ability. IEC contains a perception module for decoding other agents' intentions in response to their past actions. It

also includes a language generation module for learning implicit grammar during communication with two or more agents. Such grammar, by construction, should be compact for efficient communication. Both modules undergo conjoint evolution - similar to an infant's babbling that enables it to learn a language of choice by trial and error. We utilised three multi-agent environments, namely predator/prey, traffic junction and level-based foraging and illustrate that such a co-evolution enables us to learn much quicker (50 %) than state-of-the-art algorithms like MADDPG. Ablation studies further show that disabling the inferring belief module, communication module, and the hidden states reduces the model performance by 38%, 60% and 30%, respectively. Hence, we suggest that modelling other agents' behaviour accelerates another agent to learn grammar and develop a language to communicate efficiently. We evaluate our method on a set of cooperative scenarios and show its superior performance to other multi-agent baselines. We also demonstrate that it is essential for agents to reason about others' states and learn this ability by continuous communication.

2.1.1. Introduction

Multi-agent reinforcement learning (MARL) has drawn significant attention due to its wide applications in games [**78, 105**]. But when moving into a multi-agent environment, agents get rewards not only depending on the state and its action but also other agents' actions [**116**]. Agents' changing policies bring out the non-stationarities of the multi-agent

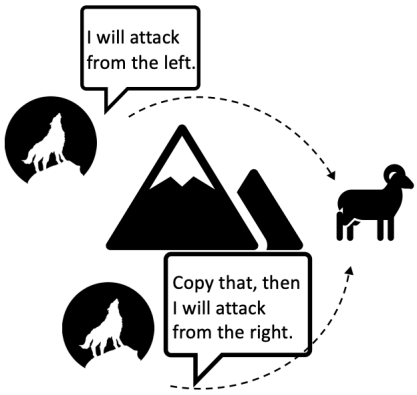


Figure 2.1. Motivation: To successfully achieve a cooperation based hunting task, the wolves are required to 1) compose their howl stating their strategies to surround the lamb and 2) for other wolves to correctly understand such a strategy being communicated by members of its pack.

environment; recent MARL methods mainly utilize opponent modeling [**142**], communication [**109, 106**] and experience sharing [**21**] to address such credit assignment problems [**21, 86**]. The multi-agent environments can be grouped into three groups, cooperative, competitive and hybrid. In this work, we mainly focus on cooperative tasks. Under such a setting, efficient and correct interpretation of others' intentions is crucial in provisioning resources and achieving the ultimate goal. Communicating with other agents is one way of inferring such intentions.

In nature, social animals can quickly establish an adequate understanding of each other's mental state and choose the actions based on that [**117**] in collaborative tasks. That's one of their essential skills to live. Several applications [**57, 79**] introduce the emergence of natural language under such a multi-agent cooperation environment. The ability to infer other agent's implicit motivation and maintain a mental state about their strategies refers to machine theory of mind [**89**]. This work proposes a framework closer to a real situation in which agents learn to reason others' beliefs by continuous exchanges,

recursively (Figure 2.1). Imagine a group of wolves hunting lambs, each of the team member need to understand others' howl and, based on its situation, draw up a response. To mimic this, we utilise variational autoencoders as a comprehension module responsible for translating the received message. The agent will first use it to reason the sender's strategy and characteristics [**116**]; in our work, the agent's following observation and reward is a response to the sender's strategy in the prior state. On one side, this information will be added to its own observation for decision making. On another side, there is a module parametrised by MLP responsible for generating the response on account of its state and inference on others. We found that with the mechanism we defined above, the agents can reach higher performance than other state-of-art methods.

Specifically, our architecture contains three functional modules (see Figure 2.2 and 2.3). First, each agent has an **inferring beliefs module (IBM)**, maintaining its interpretation and modelling others under uncertainty. And a **communication module** (C module) learns how to compress and encode an agent's observation and belief into compact codes. Finally, a **policy module** makes decisions based on the interpretation of the ongoing communication. We put forward three contributions: 1) We propose a generative module to infer other agents' beliefs; 2) We give the agents the ability to learn how to generate effective (concise) messages; 3) We showed that our method could learn to reach higher performance with all these functionalities.

2.1.2. Related Work

2.1.2.1. Modeling Other Agents. Opponent modelling is a popular and promising research direction in multi-agent reinforcement learning (MARL). The unstable nature of

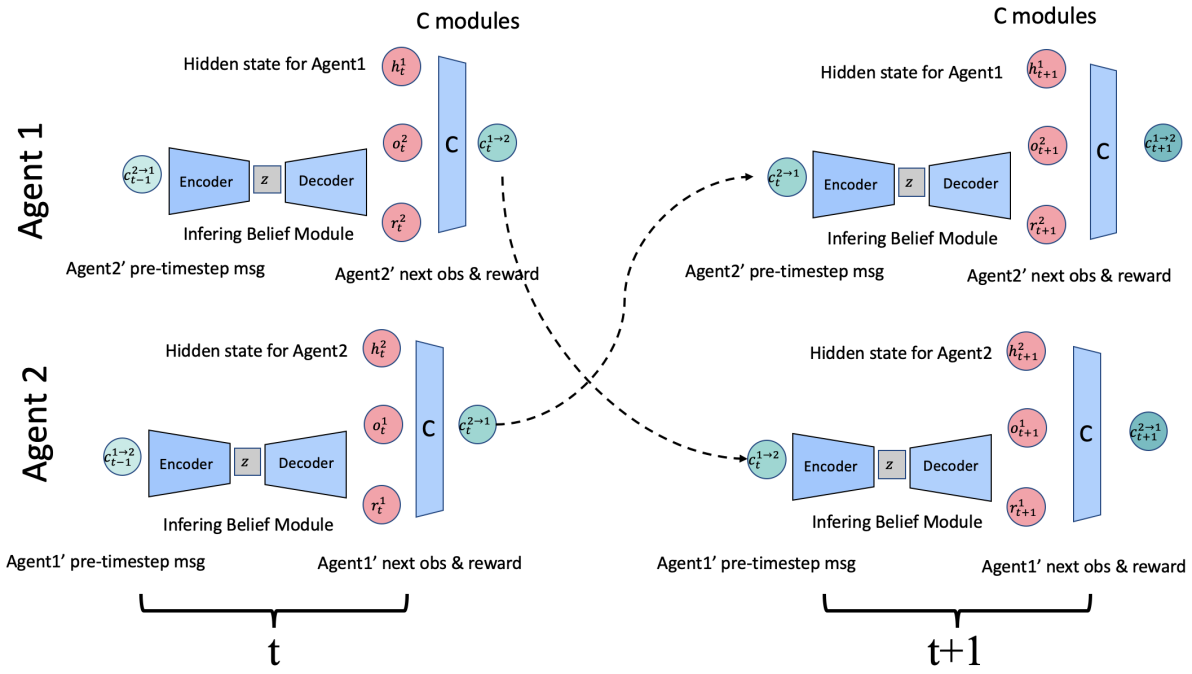


Figure 2.2. The communication architecture under a multi-agent setting. Suppose we have N agents, each of them at time t receives the observation o_t and the communication message c_t. The message is reconstructed by a VAE component which takes in and sums up all the other agents' hidden state.

the shared MARL environment has encouraged works to integrate other agents' models into the controlled agent's policy. By including a belief in the opponent's behaviour, the uncertainty of the environment is partially relieved during training.

He et al. [**32**] propose DRON, building a reinforcement learning framework based on DQN, which jointly learns the policy and the opponent modelling conditioned on observations. Zhang et al. [**133**] takes the model uncertainty into account while trying to make the MARL algorithm more robust. Chrisianos et al. [**21**] propose SePS where they model other agents' resulting observation and reward. They do so by pre-training a clustering mechanism. MoT (Machine theory of Mind) [**89**] creates two latent variables

called agent character m and mental state m_t to build a belief on other agents. They use past trajectory for updating the character and current trajectory for mental state. Zintgraf et al. [**142**] borrow this machine theory of mind idea and pass it to other agents who use Bayesian inference to obtain the predicted next action. Previous methods require access to the opponent's information, Papoudakis et al. [**86**] propose that the agent can reason about others by just accessing its observation.

2.1.2.2. Multi-agent RL based on Communication. For multi-agent reinforcement learning, each agent only has partial observation of the environment, leading to difficulty for learning. It's naturally possible to introduce a communication channel to provide additional information for the agents when they're learning cooperative or competitive tasks. This leads us to two questions. 1) How can we construct the message? and 2) Who to communicate? For the first question, CommNet [**109**] propose a model that can keep a hidden state for each cooperative agent, and the message is the average value of all other agents' hidden state. The fully-connected network brings lots of computational burden, especially as the number of agents increases, the length of the message could increase exponentially. Also, it becomes more difficult for an agent to filter the informative values among accumulated messages. IC3Net [**106**] add a gate mechanism on the top of CommNet, where another binary action is then generated for determining whether the agent would like to send the message. Vertex Attention Interaction Network (VAIN) [**37**] adds an attentional architecture for multi-agent predictive modeling. Instead of averaging the messages, they put weights in front of each vector.

2.1.3. Background

2.1.3.1. Markov Games. We formulate our environment as a Markov game [**102, 68, 130**] with N agents defined by the following tuple:

$$\mathcal{M} = (S, O_1 \dots O_N, A_1 \dots A_N, T, R_1 \dots R_N).$$

Here N is the total number of agents. S denotes the states of all agents. $O_1 \dots O_N, A_1 \dots A_N$ are the sets observations and actions for each agent. A_i is action of Agent i sampled from a stochastic policy π_i and the next state is generated by the state transition function $T : S \times A_1 \times \dots \times A_N \to S$. At each step, every agent gets a reward according to the state and corresponding action $r_i : S \times A_i \times \dots \times A_N \to \mathbb{R}$ along with an observation of the system state $o_i : S \to O$. The policy of agent i is π_i. The objective for the i-th agent is to learn a policy that maximizes the cumulative discounted rewards

$$\mathcal{R}(i) := \mathbb{E}\Big[\sum_{t=0}^{T} \gamma^t r_i^{(t)})\Big], \tag{2.1}$$

where $\gamma \in (0, 1)$ is the discount factor and $r_i^{(t)}$ is the reward received at the t-th step.

2.1.3.2. Variational Autoencoders. Variational auto-encoder (VAE) is a generative model. It can learn a density function $p(z|x)$, where z is a normally distributed latent variable, x is the given input from dataset X. The true posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$ is unknown, so the goal of VAE is to approximate a distribution $q_\phi(\mathbf{z}|\mathbf{x})$ which is parametrized by multilayer perceptrons (MLPs). VAEs also use a MLP to approximate $p_\theta(x|z)$. The

KL-divergence between two is following:

$$D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}\|p_\theta(\mathbf{z}|\mathbf{x}) = logp_\theta(\mathbf{x}) + D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p_\theta(\mathbf{z}))$$

$$-\mathbf{E}_{z\sim q_\phi(\mathbf{z}|\mathbf{x})} logp_\theta(\mathbf{x}|\mathbf{z}),$$

which is equal to

$$logp_\theta(\mathbf{x}) - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}\|p_\theta(\mathbf{z}|\mathbf{x}) =$$

$$\mathbf{E}_{z\sim q_\phi(\mathbf{z}|\mathbf{x})} logp_\theta(\mathbf{x}|\mathbf{z}) - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p_\theta(\mathbf{z})).$$

The right side of the equation is called the evidence lower bound (ELBO). Since the second term of the left side is equal or larger than zero, we can get the following inequation:

$$logp_\theta(\mathbf{x}) \geq \mathbf{E}_{z\sim q_\phi(\mathbf{z}|\mathbf{x})} logp_\theta(\mathbf{x}|\mathbf{z}) - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p_\theta(\mathbf{z})),$$

so minimising the loss function is equivalent to maximize the ELBO. The density forms of $p(\mathbf{z})$ and $q_\phi(\mathbf{z}|\mathbf{x})$ are chosen to be multivariate Gaussian distribution. The empirical objective of the VAE is

$$L_{VAE}(\theta, \phi) = -\mathbf{E}_{z\sim q_\phi(\mathbf{z}|\mathbf{x})} logp_\theta(\mathbf{x}|\mathbf{z}) + D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p_\theta(\mathbf{z})),$$

$$\theta^*, \phi^* = arg\min_{\theta,\phi} L_{VAE}.$$

The recognition and generative models are parametrised using multilayer perceptrons (MLPs).

2.1.4. Belief Embedded Communication

Most opponent modeling methods require direct access of other agents' information [**86**] or history of trajectories [**32, 92, 63**]. To lean close to real-life situations, in our hypothesis, agents can only build their beliefs by conversing with each other. Hence, we want to enable every single agent to comprehend and send the latent messages. Via the information exchange, they also learn to recursively reason, wherein they are making decisions based on the knowledge of what the opponent would do based on how they would behave [**119**].

2.1.4.1. Inferring Belief Module. Just as the human brain has an area for language processing to understand other agent's conversation, we construct a similar functional module in our framework. Essentially, we utilise a VAE (Variational Auto Encoder) to represent the belief module and model the agents' motivation. The message embeds other agents' future states including represented by following observation and rewards. Using this module, an agent can decode intentions from other agents' embedded messages in a decentralised manner instead of direct access to other agents' states.

In addition, every human understands language in the real world using their own implicit model of comprehension. So unlike prior methods, the trained VAE model is not shared by all the agents. We maintain an IBM (Inferring Belief Module) for each agent, i.e., they interpret the received message.

The encoder q and decoder p of VAE are parametrized by ϕ and θ respectively. The input of the encoder are others' embedded message c_t at timestep t which is a $C \times (N-1)$ shape tensor. C is the length of communication vector, N is the total number of agents. We assume each agent can faithfully receive others message without any information loss. The output of the encoder is a m-dimensional Gaussian distribution $N(z; \mu, \sigma^2)$.

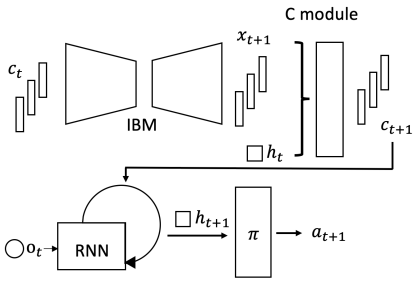


Figure 2.3. The variable flow graph of Message and policy generation module. The strip shaped bars and squares are the vectors flow in and out the modules. The message pass through the IBM (Intention Belief Module) to get the predictions of other agent's intention named x_{t+1}. C module then utilize the predictions and agent's own hidden state to generate next timestep's communication vector c_{t+1}. The policy module add in observation and passing through a RNN and policy network outputing action a_{t+1}.

The decoder predicts the $p(x_{t+1}|z)$, where $x_{t+1} = (o_{t+1}, r_{t+1})$ is the next observation and reward.

The agents' beliefs can be projected to the latent space Z by $q_\phi(z|c_t)$ and $p_\theta(z|x_{t+1})$. Hence the objective is to optimize $D_{KL}(q_\phi(z|c_t) \parallel p_\theta(z|x_{t+1}))$. Since $p(x_{t+1})$ is intractable, it's equal to optimize evidence lower bound (ELBO) which we derive as following:

$$\begin{aligned} logp(x_{t+1}) \geq \mathbb{E}_{q_\theta \sim (z|c_t)}[logp_\theta(x_{t+1}|z) \\ -D_{KL}(q_\theta(z|c) \parallel p(z))] \end{aligned} \tag{2.2}$$

We collect the trajectories and messages for certain period of time. The IBM of each agent is trained independently. While training, the trajectories of all agents are collected and saved to a buffer, for every 50 epochs, we train VAEs 10 times.

2.1.4.2. Policy and Message Generation Module. For agents, the policy module takes in the observation from environments and combines the beliefs of others' intentions

predicted by previous IBM. It outputs the message delivered to others and subsequent actions agents will take.

The new generated communication vector is continuous, it's aggregated by all the IBM predictions that represent the agent's belief and agent's own hidden state (see Figure 2.3). For agent j, c_{t+1}^j is the generated message to others at timestep t shown as following:

$$c_{t+1}^j = \frac{1}{N-1} C(\sum_{j=1}^{N-1} x_{t+1}^j + h_t)$$

$$x_{t+1}^j = IBM(c_t^j)$$

$$h_{t+1} = RNN(o_t + c_{t+1}).$$

The message generation module C is parametrised by a fully connected neural network. Previous IBM(Inferring Belief Module) decodes x_{t+1}^j. There is no importance factor multiplied by each vector. We also prepare an RNN module to keep track of the agent's hidden state like [**106, 109**]. Next timestep's hidden state h_{t+1} is generated by this recurrent module where the input is current step's observation o_t and the new generated communication c_{t+1} The next action is generated using policy network implemented as fully-connected neural network $a_t^j = \pi(h_t)$.

2.1.4.3. Implementation.

Require: Total episode number E, Duration of each episode T, Number of agents N. VAE training interval I.

Initialize policy π_0, VAE models' parameter θ, ϕ, RNN module.

Initialize memory buffer $\mathbf{B}$

Initialize IBM's N replay buffer $\mathbf{D} = \{D^0, \cdots, D^N\}$

3: for $k = 0, \ldots, E-1$ do

Play an episode using current policy π_k.

add trajectory τ_k into memory buffer $\mathbf{B}$

for $j = 0, \ldots, N$ do

add c^{-j}, s^{-j} and r^{-j} into replay buffer D^j

end for

Update policy $\pi_{k+1} = \pi_k + \beta \nabla_\theta \hat{J}_\pi(\pi_k)$ and RNN using the data in memory buffer $\mathbf{B}$.

if k is divisible by I then

for $j = 0, \ldots, N$ do

Update θ, ϕ using samples from D^j

end for

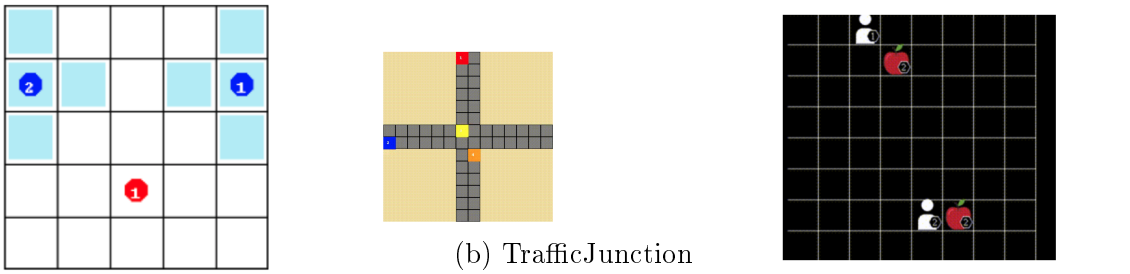
end if

4: end for

5: return optimal π^*, θ^*, ϕ^*

Algorithm. *Intention Embedded Communication*

As Algorithm 2.1.4.3 shows, we first initialise a policy π_0 and N belief modules, each representing specific agent's comprehension of the received message. We train the policy learning module and inferring belief module (IBM), separately. Since IBM is trained along with the policy, as the agents' policy evolves, the comprehension ability updates at set intervals. Specifically, for every I episode, our method collects this period of experience into buffer $\mathbf{D} = \{D^0, \cdots, D^N\}$. For each agent's buffer D^j, it contains others' message c^{-j}, next state s^{-j} and reward r^{-j} for training. In the next update iteration, the buffers are emptied because the context between agents is changed. Memory buffer



(a) PredatorPrey

(b) TrafficJunction

(c) Level-based Foraging

Figure 2.4. Environments used in our experiment. Left is PredatorPrey Env. The blue circles with numbers represent the predators and the red one is the prey. The azure blocks around the predator illustrate their (limited) visual inputs. The default grid size is 5×5. In our setting, the grid size, number of agents and vision are changeable according to the different difficulty settings. The center is TrafficJunction Env. Each agent represented by colored circles need to pass the crossroads without collision. Their actions are limited to *gas* and *brake*. The right is Level-based Foraging, the largest difference of this environment and PredatorPrey is that it requires a level verification while collecting the targets.

B collects all the interactions for each step, including the rewards, then the pairs are used to train the policy π like standard reinforcement learning methods. By the end, we anticipate that all agents can learn to hear (inferring other's messages correctly) and speak (generate informative messages) in an efficient way. We follow the centralized training with decentralized execution (CDTE) paradigm.

2.1.5. Experiment

The following sections present the experiments where we evaluate our approach in multiple multi-agent environments including PredatorPrey [**106**], TrafficJunction [**52**] and Level-base Foraging [**87**] showing in Figure 2.4. These environments require effective coordination between agents. We exhaustively compare our algorithm (IEC) with other

state-of-art multi-agent methods in each environment and probe into the different configurations and parameters that may affect the performance. All results presented are averaged over five independent seeds. Our algorithm's network architecture is kept the same for all runs. The learning rate is 1×10^{-3} and batch size is 500. The buffer size for VAE is 4×10^4; we trained the VAE every 50 episodes.

2.1.5.1. Baselines. We compare our method with following popular multi-agent methods as baselines.

VDN: Value-decomposition Networks [**110**] factorize the joint value function into N Q-functions for N agents. Each agent's value function only relies on its local trajectory.

QMIX: QMIX [**93**] is a value-based off-policy algorithm that extends the VDN with a constraint that enforces the monotonic improvement.

MADDPG: Multi-Agent Deep Deterministic Policy Gradient [**73**] use trajectories of all agents to learn a centralized critic. At execution phase, the actors act in decentralized manner.

2.1.5.2. PredatorPrey. In the PredatorPrey environment, N predators are assigned to capture the prey. The setting is that only after all predators reach the prey's location can the reward be given to them. The observation of a given predator is its concatenation of the nearby grid. The actions are discrete options (`Forward, Backward, Left, Right, Stale`) for all agents. The maximum number of steps for each episode is 20.

The first experiment we conducted is a simple cooperative setting with 5×5 grid as a base map. For every episode, 2 predators and 1 prey are randomly allocated in the grid world. When all the predators reach the prey's location, they are awarded 5 points. We

	Agents Num	Grid Size	Agent's Vision
Easy Version	2	5	0
Medium Version	4	10	1
Hard Version	10	20	1

Table 2.1. Settings of different versions of the PredatorPrey environment

sum up all predators's rewards as an indicator of performance. We run experiments for 5 trials with independent seeds. The learning rate for our algorithm and other baselines is 0.001. As shown in Figure 2.6a, our algorithm converges speedily compared to others. It uses around half of the interactions compared to the second-best MADDPG.

We then tested our algorithm on various environment settings. Specifically, we follow most settings from [**106**] to construct three levels of difficulty with different agents numbers, grid size and agents' vision. The details can be found in Table 2.1. The communication vector length in the three environments is the default 128 lengths. As Figure

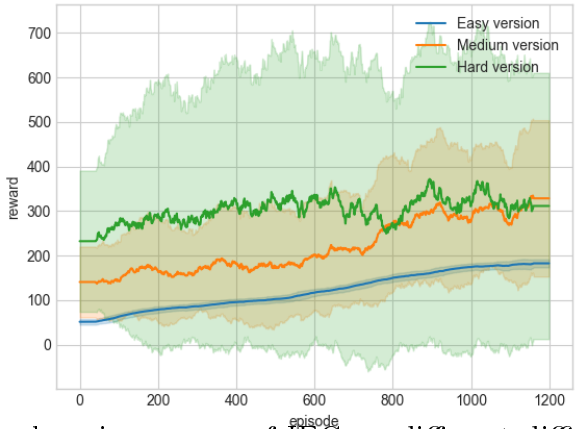
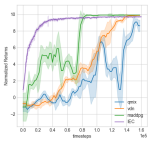
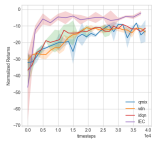


Figure 2.5. The learning curves of IEC on different difficulty levels of the PredatorPrey Env. It shows that as environment becomes more complicated, the variance notably increases.

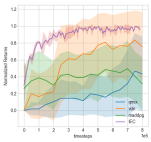
2.6a shows, when the environment becomes more difficult for the agents, the variance



(a) PredatorPrey



(b) TrafficJunction



(c) Level-based Foraging

Figure 2.6. Performance of IEC and other baselines listed in 2.1.5.1 during the training protocol. The y-axis is normalized returns and x-axis is the total time steps the algorithm takes. As shown in three subplots, IEC can learn the task faster and achieve higher or equal returns by the end.

enlarges, since in our setting, we assign the rewards to agents only when all the agents reach the prey's location. As the number of agents increase and the grid size expands, it's challenging for them to finish the task in limited time steps.

We conduct several experiments for investigating the influence of the message size. Our communication consists of the hidden state, and the agent's beliefs of others. The message generation module concatenates these two together and forwards them to a trainable neural network for the final outputs. The length of the message is adjustable, so in the following experiment, we probe into the influence of the length of the different bits. We conduct experiments with 32, 64 and 128 bits of message and show their training process. As observed in the left plot of Figure 2.8, the length of the communication vector has a significant influence on policy optimisation. Longer bits can carry more encoded information and facilitates communication between agents.

2.1.5.3. Traffic Junction. The traffic junction environment we use references to [**106**] and [**52**]. In this environment, cars move across intersections in which they need to avoid collisions with each other. There are adjustable routes users can set. The number of routes and agents define the three levels of difficulty named easy, medium and hard version.

	Agents Num	Grid Size	Max Steps
Easy Version	5	6	20
Medium Version	14	10	40
Hard Version	20	18	80

Table 2.2. Settings of different versions TrafficJunction environment

We first compared our method with other baselines. As Figure 2.6b shows, agents trained by our method can learn to cooperate at a much faster speed and also furnish higher final returns. The other methods are nearly of the same performance. The TrafficJunction environment requires agents to utilise all local observations to get the other agent's position and intention. This setting brings a great advantage for our method because the belief and cooperation information can be encoded into a message. Under our framework, the negotiation can be reached by mutual recursive reasoning to avoid a collision.

The multi-agent traffic junction environment also has three versions with incremental levels. Table 2.2 lists the details. Figure 2.7 shows that the parameters in Table 2.2 pulls down the converged total reward and brings more variance into cooperation among agents. Same as in the Predator-Prey environment, we want to test our algorithm on different bits of message on the Traffic Junction environment. We can tell from the centre plot of Figure 2.8 that, unlike the predator-prey environment, the bit-length of information causes disparity in the final performance; in a traffic junction environment, it primarily affects the learning speed.

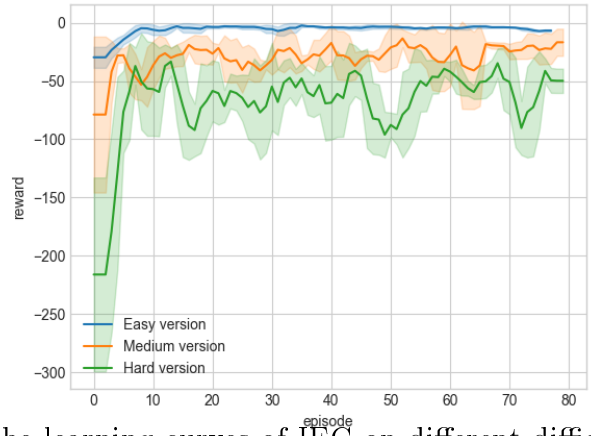


Figure 2.7. The learning curves of IEC on different difficulty level of TrafficJunction Env.

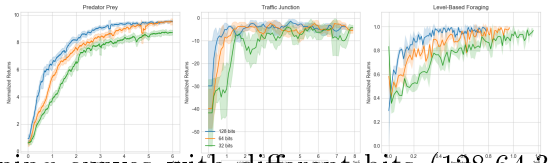


Figure 2.8. Learning curves with different bits (128,64,32) respectively in three environments. This figure demonstrates that the bit-length (capacity) of the communication vector – which represents the amount of information it can carry – has distinct influence on learning speed. The size of this influence varies across different tasks.

2.1.5.4. Level-based Foraging. Level-based Foraging (LBF) is a multi-agent environment focusing on the coordination of the agents involved [**87**]. Compared to the PredatorPrey environment, agents and food in this world are assigned within a level; only when the agent's level is equal to or higher than a certain limit, can it successfully collect the food, as in Figure 2.4. The action space contains four directional moves and a loading action. This environment is more challenging than previous ones due to its additional limitations.

Figure 2.6c shows our algorithm obtains higher learning speed and normalised returns than QMIX and VDN. The constraint that only a certain level can elicit pick up of the

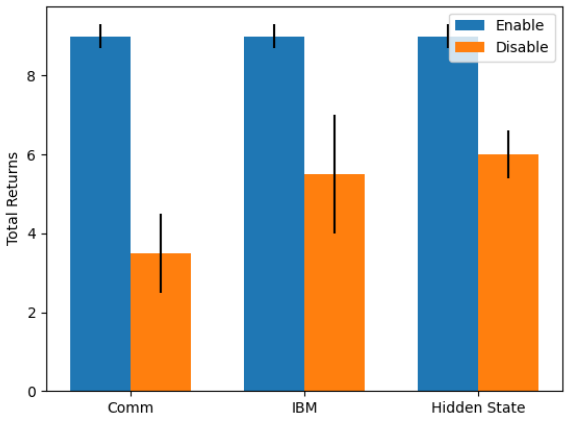


Figure 2.9. The blue bars are all function enabled IEC with communication vector of size 128 bits. The orange bar is the returns the algorithm can achieve after disabling the corresponding feature.

food decreases the probability that the agents randomly hit the target. This also requires richer observation and more efficient cooperation, like not wasting time collecting the same food. In such a scenario, we can see our intention encoded communication plays a vital role in assisting agents in achieving their goals.

2.1.5.5. Ablation Study. This section discusses some key building modules communication, IBM, and hidden state, respectively. They have a significant influence on final performance. We conducted an ablation study by solely disabling these features compared with the fully functional IEC in the PredatorPrey environment. The difficulty level is easy, and the message length stays at 128 bits.

No Communication We analyse the functionality of the communication mechanism. In this setting, agents do not have any exchanges, which means the IBM is also unavailable due to the lack of passing messages; they can only use their observation as input for decision making. Figure 2.9's first pair shows that the predator encounters greater total reward loss than the method with fully functional communication.

No IBM For validation of our Inferring Belief Module (IBM)'s functionality, we run one experiment that all predators only receive others' hidden state as a message, i.e., agents do not exchange each other's embedded intentions during the experiment. Figure 2.9's centre plot shows that without the extra belief inference of other agents' messages cause about 38% performance decrease.

No Hidden State A hidden state is the vector we use to embed the agent's previous state using an RNN. Details can be found in Section 2.1.4.2. As the right plot of Figure 2.9 shows, the algorithm reaches around 2/3 returns compared to IEC if the hidden state is missing in a transmitted message. We can see that the compact information of the agent's state also brings some help to complete the cooperation tasks. But without IBM, it can't reach maximal returns.

2.1.6. Conclusion

This paper proposes IEC framework that can mimic the multiple agent's natural communication in cooperative tasks. We devise two modules: Infer Beliefs Module (IBM) and Message Generation Module. These modules abstractly represent functional areas in the brain for successful comprehension inference and grammar generation. We also adopt the idea that agents learn the communication and underlying grammar by trial and error (co-evolution). We empirically demonstrated that our algorithm could outperform other multi-agent baselines with a significant gain in data efficiency. Such an algorithm stays robust under various environment settings, like different difficulties and message length.

The ablation study also investigates the processes' key factors and reveals their contribution. In future work, we would like to explore more generative structures like GANs for inferring belief modules.

2.2. Visual Prompt via Pre-trained Model for Quadrupedal Locomotion

We introduce an innovative end-to-end quadrupedal locomotion algorithm leveraging pre-trained vision and decision models, thereby pioneering a higher echelon of control and significantly reducing training time. In light of the advancements in large-scale vision and language models, our approach exploits these developments to enhance robotic locomotion strategies. Our algorithm is designed to integrate the outputs from a selection of extensive pre-trained perception models, utilizing them as inputs or "prompts" for a decision-making transformer. This integration facilitates a richer understanding of the environment through advanced RGBD data processing, allowing the robot to have an expanded and more nuanced perception of its surroundings. When implemented in real-world tests on a quadrupedal robot, our hybrid policy demonstrated superiority over other trained methods, particularly excelling in tasks involving collision avoidance. This study indicates that the synergy of pre-trained vision models and decision transformers can indeed pave the way for groundbreaking strides in robotic locomotion, ushering in a new era of efficiency and intelligence in quadrupedal robot navigation.

2.2.1. INTRODUCTION

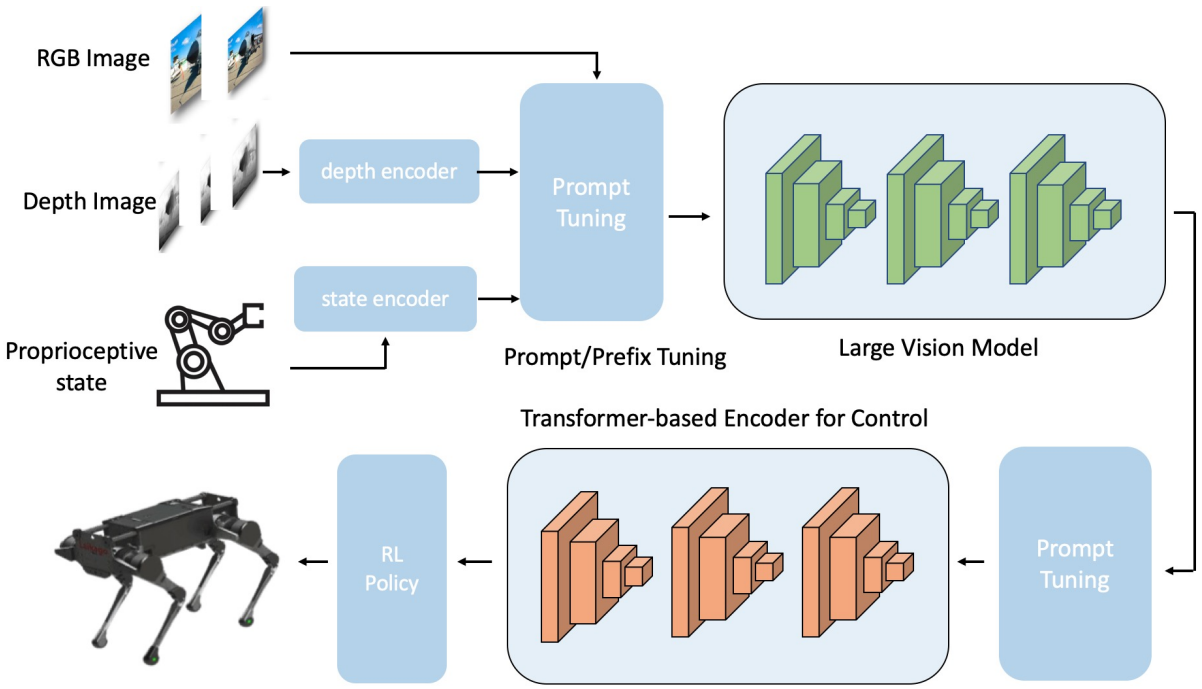


Figure 2.10. The two large black-framed blocks are pre-trained model which means their parameters will keep frozen during training. The other light-blue blocks represents tunable parameters. The system integrates proprioceptive states with vision inputs through a large pre-trained vision model. The proprioceptive encoder processes state information, while the depth encoder captures depth information and transform it to required tensor size. The RGB image go directly into prompt tuning module, these features are combined using prompt tuning to enhance the large pre-trained model's visual understanding. The vision features is then passed to another prompt tuning module specially designed for Control Encoder, the output informs a reinforcement learning (RL) policy to generate motor torques for controlling the quadrupedal robot.

In recent years, the field of robotic locomotion has undergone a transformative shift, driven by rapid advancements in artificial intelligence (AI) and machine learning (ML). Autonomous systems, particularly quadrupedal robots, are increasingly being designed to navigate complex, unstructured environments [**101, 25, 76**], making them ideal for

applications in search and rescue, remote exploration, and industrial automation. However, achieving robust locomotion in these environments remains an enduring challenge, requiring a sophisticated interplay between perception, decision-making, and control mechanisms. Many studies have explored vision-integrated locomotion approaches, but most methods either train vision models from scratch or employ manually engineered perception pipelines, which can be costly in terms of data and training time.

One of the core challenges in robotic locomotion is the ability to navigate dynamic environments with high precision and safety. Traditional approaches to robot navigation often rely on hand-engineered features or conventional control methods, such as model-based planning and rule-based navigation [**77, 31, 49**]. These techniques, though effective in constrained scenarios, tend to struggle in real-world applications where the environment is less predictable. Furthermore, these approaches generally require extensive training and computational resources, which limits their scalability and adaptability.

As AI models become more sophisticated, large-scale vision and language models, such as those used in natural language processing (NLP) and computer vision, have shown great promise in advancing perception and decision-making tasks. Models such as, convolutional neural networks (CNNs) , and transformer-based architectures such as Vision Transformers (ViTs) [**23**], large-vision-language model such as openVLA [**48**] have proven effective in extracting high-dimensional features from complex environments, making them particularly valuable for robotics. However, the integration of these models in robotic systems, especially for tasks such as quadrupedal locomotion, remains an underexplored area. Our approach distinguishes itself by leveraging a pre-trained large vision model to provide visual prompts for locomotion, significantly reducing training overhead while enhancing

perception. Unlike conventional methods that fully fine-tune the perception module, our method employs prompt tuning, where a small number of learnable parameters are added to steer a frozen pre-trained vision model toward locomotion-specific tasks.

This approach is inspired by recent developments in prompt-based learning within the NLP community, where pre-trained models are fine-tuned using task-specific prompts [**75, 128, 65, 141, 74**]. By borrowing this concept, we extend its application to robotic control systems, allowing our robot to effectively interpret and respond to visual cues from its environment. The pre-trained vision models process RGBD data, offering a detailed representation of the surrounding terrain, while the decision transformer utilizes this information to predict optimal actions in real-time.

Our work is also motivated by the need to enhance the robot's ability to avoid collisions, one of the most critical aspects of safe and efficient navigation [**51, 136, 125**]. Collision avoidance requires a fine-grained understanding of the robot's surroundings, which can be significantly improved by leveraging pre-trained models. By integrating visual perception models that have already been trained on large datasets, we eliminate the need for extensive task-specific training, thus accelerating the learning process and reducing computational overhead, and other works have also found visual pre-training crucial in many aspects [**100, 91**].

Contributions of this work: this research makes the following key contributions to the field of robotic locomotion: (i) Novel Integration of Pre-trained Vision Models: We introduce a method that utilizes pre-trained vision models as prompts for a decision-making transformer. This innovative approach allows the robot to leverage rich visual data from its surroundings, resulting in a more nuanced understanding of the environment. (ii)

Improved Locomotion Strategy: By combining visual and proprioceptive inputs, we propose a hybrid locomotion policy that significantly improves the robot's ability to navigate challenging terrains such as stairs, rough surfaces, and narrow pathways. (iii) Reduced Training Time: Our approach minimizes the need for extensive task-specific training by using pre-trained models, thereby reducing the overall training time and computational resources required for the robot to adapt to new environments. (iv) Real-World Validation: We demonstrate the effectiveness of our algorithm through rigorous experiments in both simulated and real-world environments. The results show that our method outperforms traditional approaches, particularly in collision avoidance tasks.

The remainder of the paper is organized as follows: Section II reviews related work in the field of quadrupedal locomotion and pre-trained vision models, Section III describes the design of our vision-prompted locomotion algorithm, and Section IV presents the experiments and results. We conclude with a discussion of future work and potential applications. In particular, we highlight how our approach offers significant improvements in adaptability and efficiency for quadrupedal robots across varying environments.

2.2.2. Related Work

As we forge ahead in the development of our innovative quadrupedal locomotion algorithm, it becomes imperative to acknowledge the groundbreaking works and conceptual frameworks that lay the foundation for our endeavor. Here we delineate two seminal areas of work that significantly influence our research:

Large pre-trained Vision and Language Models. The rapid advancements and unprecedented achievements facilitated by large pre-trained vision and language models

in various domains cannot be overstated [**12, 48, 113, 108**]. These models have redefined the boundaries of what is achievable in tasks involving complex pattern recognition and decision-making processes. Leveraging the insights derived from these developments, our research takes a decisive step towards the exploration of these models' capabilities in enhancing the perception and decision-making faculties of quadrupedal robots, thereby promising a new frontier in robotic navigation and control [**17, 123, 115**].

Quadrupedal Locomotion and Collision Avoidance Quadrupedal locomotion has been a focal point in robotics [**58, 20, 41**], striving to attain a harmonious blend of stability, agility, and adaptability in navigational strategies.

Prior works [**27, 99, 34**] have integrated pre-trained vision models into quadrupedal locomotion pipelines, but these approaches usually decouple perception and control—often training the visual module separately or employing a modular perception-to-control pipeline. Our approach differs by leveraging a frozen pre-trained vision model with prompt tuning, which drastically reduces the number of trainable parameters and removes the need for extensive perception-specific training. This design is end-to-end: only a small set of learnable prompt parameters is jointly trained with the locomotion controller, adapting the fixed pre-trained model to the task without full fine-tuning. As a result, our method achieves improved sample efficiency and faster training convergence compared to approaches that train vision encoders from scratch or fully fine-tune them.

2.2.3. Vision Prompt for Adapting Locomotion

We address the problem of quadrupedal locomotion using visual observations, where continuous RGB-D images serve as environmental prompts to adapt the robot's policy to

heterogeneous environments. Our approach builds on the foundational models presented in [**127, 75**]. The original model contains a transformer-based encoder to process the proprioceptive states then following a standard PPO policy. In our method, a multimodal, pre-trained vision model is employed to effectively perceive the surrounding environment by integrating depth images, RGB images, and proprioceptive states. In addition, we freeze and fine-tuned the pre-trained transformer-based control encoder during training, while keeping the smaller RL component trainable as Figure 2.10 shows. This strategy significantly reduced the overall training cost, making the architecture highly efficient and can be easily extended to various future downstream tasks.

2.2.3.1. Problem Definition. We formulate the visual-locomotion task as a Markov Decision Process (MDP), $(O, S, A, R, P, p_0, \gamma)$, where O is the observation space, S is the state space, A is the action space, $R : S \times A \mapsto \mathbb{R}$ is the reward function, $P : S \times A \mapsto S$ is the dynamics equation, p_0 is the initial state distribution and γ is the discount factor. Our goal is to find a policy $\pi : O \mapsto A$ that maximizes the expected accumulated reward: $J(\pi) = \mathbb{E}_{\tau=(\mathbf{s}_0,\mathbf{a}_0,\ldots,\mathbf{s}_T)} \sum_{t=0}^{T} \gamma^t r\left(\mathbf{s}_t, \mathbf{a}_t\right)$

Algorithm. *Policy Gradient with Visual Prompt*

Initialize RL policy parameters θ

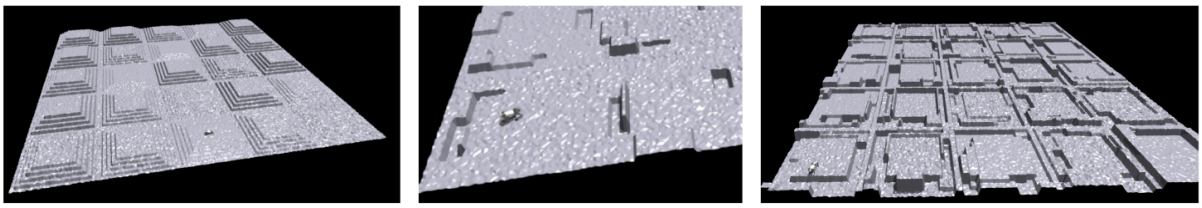
Initialize all encoders and prompt/prefix tunable parameters ρ

for each episode do

    Initialize state s

    Preprocess s through the encoders and prompts/prefix tuning modules to get v

    while not terminal do



a b c

Figure 2.11. Simulation on different environments with ascending difficulty level. a) Pyramid stairs terrain with random rough surfaces; b) Discrete and sparse rectangular obstacles; c) More dense gaps and high wall obstacles.

Sample action a from $\pi_\theta(\cdot|v)$

Execute a, observe next state s' and reward r

Preprocess s' through the encoders and prompts/prefix tuning modules to get v'

Store transition (v, a, r, v') in memory

Update $s \leftarrow s'$

end while

for each step of gradient descent do

Sample mini-batch of transitions (v, a, r, v') from memory

Compute gradient $\nabla_\theta J(\theta)$ using sampled transitions

Update RL policy parameters: $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Update tuning parameters: $\rho \leftarrow \rho + \beta \nabla_\rho J(\rho)$

end for

4: end for

2.2.3.2. Tuning Methods. In our neural network architecture as showed in Figure 2.10, there are two part of large pre-trained models, the first is vision model downloaded from

Huggingface[**70, 14**], the second is pre-trained control policy. Since both of them are transformer based, we deploy the same tuning method on both of them.

Tuning methods designed to fine-tune large pre-trained models (such as transformers) for specific tasks without altering the majority of the pre-trained model's parameters. The main idea behind both techniques is to add small learnable components (either prompts or prefixes) to the input or attention mechanism, allowing the model to adapt to new tasks efficiently. We take two mainstream realizations to insert prompt to transformer model borrowed from the NLP community and explained as follows:

Prompt Tuning (ProT) Assume the transformer model takes as input a sequence of tokens $x = (x_1, x_2, ..., x_n)$, where $x_i \in \mathbb{R}^d$ represents the embedding of the i-th token in the input sequence. The typical input to the transformer is:

$$h_0 = [x_1; x_2; \ldots; x_n]$$

where $h_0 \in \mathbb{R}^{n \times d}$ is the input embedding matrix for the tokens. In Prompt-Tuning, we prepend a learnable prompt matrix $p \in \mathbb{R}^{n \times d}$, where m is the number of prompt tokens (learnable vectors) and dd is the embedding dimension (same as the token embeddings):

$$h'_0 = [p_1; p_2; \ldots; p_m; x_1; x_2; \ldots; x_n]$$

Here, $p_1; p_2; \ldots; p_m$ are the learnable prompt embeddings. The augmented input h'_0 is then fed into the transformer layers. Only the parameters of the prompts pp are fine-tuned during training, while the rest of the model (i.e., the transformer's parameters) remains frozen.

The transformer then processes the sequence as usual:

$$h_{output} = Transformer(h'_0)$$

The key difference is that the prompt tokens pp guide the model's attention layers to adapt to the new task.

Prefix Tuning (PreT) Prefix-Tuning is slightly different from Prompt-Tuning. Instead of adding tokens to the input sequence, Prefix-Tuning appends learnable parameters (called prefixes) to the keys and values of the attention mechanism in each layer of the transformer, influencing how the model attends to different parts of the input. For a transformer, at each layer l, the queries Q^l, keys K^l, and values V^l are calculated as:

$$Q^l = h^{l-1}W_Q^l, K^l = h^{l-1}W_K^l, V^l = h^{l-1}W_V^l$$

Whereh^{l-1} is the hidden state from the previous layer, and W_Q^l, W_K^l, W_V^l are the learned projection matrices for the queries, keys, and values. In Prefix-Tuning, we prepend learnable prefix vectors to the keys and values matrices at each layer. Let the learnable prefix be $p_K^l \in \mathbb{R}^{m \times d_k}$ and $p_V^l \in \mathbb{R}^{m \times d_k}$, where m is the number of prefix tokens, and d_k, d_v are the dimensions of the keys and values, respectively.

The new key and value matrices are:

$$K'^l = [p_K^l; K^l], V'^l = [p_V^l; V^l]$$

This means that the prefixes are appended to the keys and values before calculating the attention score, effectively modifying how the model attends to different parts of the

input:

$$Attention(Q^l, K'^l, V'^l) = softmax(\frac{Q^l(K'^l)^T}{\sqrt{d_k}}V'^l$$

The prefixes p_K^l and p_V^l are the only learnable parameters in Prefix-Tuning, while the rest of the model remains frozen. The prefixes allow the model to adapt its attention mechanism to the new task without modifying the input tokens or the core transformer parameters.

Both techniques offer parameter-efficient ways to fine-tune large pre-trained models, but Prefix-Tuning tends to capture more task-specific information as it directly alters the attention process at each transformer layer, leading to better performance in more complex tasks as Section 2.2.4 shows. Also note that since Prefix-Tuning modifies the attention mechanism by appending learnable parameters to the key and value matrices in each transformer layer. While HuggingFace models like Swin and DINO do not natively support Prefix-Tuning out of the box, we modify their internal attention mechanisms to add prefix tokens.

2.2.4. Experiments and Results

In this section, we empirically demonstrate the effectiveness of utilizing a pre-trained vision model as a prompt for enhancing the performance of a depth-image based locomotion algorithm. We set a goal in 5 meters front of robot and count the collision times in a fixed 1 mininute period. We conduct 100 trials for each method to get the result shows in Table 2.3.

Table 2.3. Simulations across three different types of environments: Pyramid Stairs, Sparse Discrete Obstacle, and Dense Discrete Obstacle. We evaluate the performance of our models based on three metrics: Distance Covered, Number of Collisions, and Success Rate (SR). The results shows the performance of a State-Only Model, several Prompt-Tuning (ProT) models, and pre-trained (PreT) models.

	Pyramid Stairs Terrain			Sparse Discrete Obstacle			Dense Discrete Obstacle		
Model	Distance	Collision #	SR	Distance	Collision #	SR	Distance	Collision #	SR
State-Only Model	4.52	2.14	1.0	3.93	4.31	0.9	2.53	4.5	0.65
ProT-SSLSOD	4.81	1.20	1.0	4.20	2.53	1.0	3.65	3.50	0.85
ProT-Swin	4.95	0.88	1.0	4.64	1.78	1.0	4.12	2.98	0.85
ProT-DINO	4.99	0.96	1.0	4.55	1.97	1.0	4.01	3.22	0.89
PreT-SSLSOD	4.97	0.96	1.0	3.95	1.32	1.0	3.42	2.79	0.93
PreT-Swin	5.0	0.74	1.0	4.02	1.21	1.0	4.22	2.32	0.94
PreT-DINO	4.98	0.88	1.0	3.98	1.45	1.0	4.17	2.43	0.93

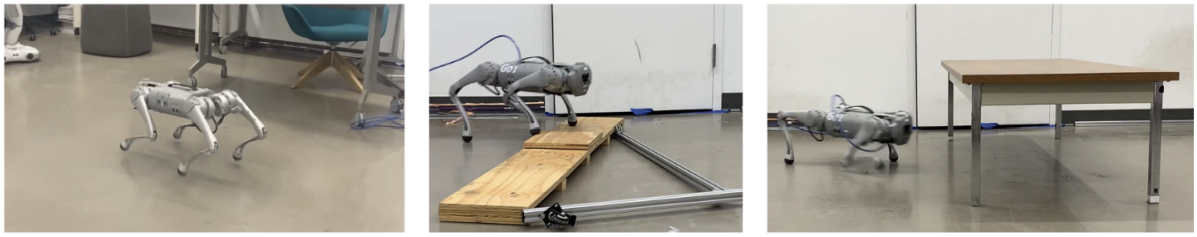
2.2.4.1. Experimental Setup. We conducted experiments in a simulated environment with diverse terrain complexities. The performance of our models was evaluated on three distinct types of environments, each designed to progressively increase in complexity:

Simulation Environments. In our experiments, the quadrupedal robot was deployed in a simulated environment designed to mimic real-world scenarios. These scenarios include a variety of obstacles and challenges, such as: **Pyramid Stairs**: This simulated environment features a series of stepped, pyramid-like structures that the robot must ascend and descend. The regularity of the obstacles makes it a relatively simple environment for navigation, testing the robot's ability to handle controlled elevation changes while maintaining balance and stability. **Sparse Discrete Obstacle**: This environment consists of scattered, discrete obstacles placed sparsely across a flat surface. It represents a moderately challenging terrain, where the robot needs to navigate around obstacles while maintaining a clear path. The gaps between obstacles provide ample space for planning,

but effective obstacle avoidance and pathfinding are still critical for successful navigation. **Dense Discrete Obstacle**: This is the most challenging terrain, densely populated with discrete obstacles. The robot must carefully navigate through tight spaces and avoid collisions. The high obstacle density increases the complexity of the environment, requiring precise control and enhanced perception for successful traversal.

Metrics. To comprehensively evaluate the models, we defined several performance metrics: **Distance**, measures how far the quadrupedal robot is able to travel within a predefined time frame or until the task is completed. To measure the distance covered, the robot is allowed to move for a fixed period(1 minute) or until it reaches the goal. The total distance it travels is recorded and used for comparison across different environments. **Number of Collisions**, is used to evaluate how effectively the robot can avoid obstacles or hazardous terrain features that could destabilize it. Collision avoidance is a fundamental aspect of safe and autonomous robot navigation. In this context, a collision event is recorded whenever any part of the robot's body makes contact with an object in the environment. Over the course of each trial, the total number of collisions is tracked. **Traversal Success Rate** measures the percentage of trials in which the robot successfully completes its navigation task without falling or being incapacitated by excessive collisions. This metric provides insight into the overall robustness of the locomotion system under different conditions. The success rate is determined by calculating the ratio of successful trials, where the robot completes the task without falling or suffering significant collisions to the total number of trials conducted.

Pre-trained Models. Our base model is a trained locomotion policy [**75**]. It only depends on robot state since it's not equipped with any camera or other sensors. After



a b c

Figure 2.12. Real-world deployment on `Unitree GO1` with varying difficulties. a): Robot walks on even terrain with little disturbance. b): Robot walks over low obstacles. c): Robot faces much larger obstacles and accomplish collision avoidance.

several hours training, the policy can seamlessly transplant to a real-world `Unitree GO1` robot. To enhance the perception ability for handling more complex and challenging environment, we add few more pre-trained object detection module into the network as follows: SSLSOD [**139**]: a self-supervised pre-trained RGB-D model that uses cross-modal auto-encoder and depth-contour estimation tasks to capture rich semantic contexts from RGB and depth data, without requiring large-scale labeled datasets. It also incorporates a consistency-difference aggregation (CDA) module to enhance cross-modal feature fusion, providing effective initialization for downstream tasks. Swin Transformer [**70**]: The Swin Transformer is a hierarchical vision transformer model that applies self-attention within shifted windows. It captures both local and global information effectively, making it particularly useful for object detection. DINO [**14**]: DINO is a self-supervised learning approach for ViT models, allowing the model to learn meaningful representations without labeled data. DINO provides strong performance in tasks like object detection and segmentation by leveraging self-supervised pretraining.

2.2.4.2. Simulation Results. The State-Only Model, which does not leverage any visual or pre-trained information, exhibited the lowest performance across all metrics.

As the difficulty of the tasks increases, we observe a corresponding decline in the performance of all models. This is particularly evident when comparing the models' performance across the three environments: Pyramid Stairs Terrain, Sparse Discrete Obstacle, and Dense Discrete Obstacle. For the first two relatively simple environments, namely flat surfaces and rough terrain, almost all models consistently reach the goal with a success rate of 100%. These results suggest that for less complex tasks, both Prompt-Tuning and Prefix-Tuning methods, regardless of the pre-trained model used, are capable of extracting sufficient features from the environment to complete the task effectively.

However, as the complexity of the environment escalates, such as in the Dense Discrete Obstacle scenario, the performance gap between different models becomes more pronounced. It turns out that Prefix-Tuning captures more vision information than Prompt-Tuning when using the same large pre-trained model, such as Swin Transformer or DINO. This result aligns with the notion that Prefix-Tuning allows for a deeper integration of pre-trained features by appending continuous prompts to the input sequence, effectively adapting the transformer's attention mechanisms to the current task without significantly altering the pre-trained model's parameters. This likely leads to more effective feature extraction and better handling of dynamic and unpredictable environments.

Additionally, we observe that the SSLSOD (Self-Supervised Learning Salient Object Detection) model consistently underperforms compared to the Swin and DINO models across all environments. This can be attributed to the significantly smaller parameter

size of the SSLSOD model. While SSLSOD's lighter architecture makes it more computationally efficient, it lacks the capacity to process and fuse visual information as effectively as larger models. As a result, it struggles more with the complex visual cues needed for collision avoidance and stable navigation in challenging terrains. The smaller parameter size limits the model's ability to extract higher-level features that are crucial for successfully navigating dynamic and unpredictable environments, especially in tasks where depth perception and intricate obstacle avoidance are required, such as on slippery surfaces.

In summary, while SSLSOD offers a more lightweight alternative with fewer parameters, its performance trade-off is noticeable in more challenging environments. In contrast, both Prefix-Tuning and Prompt-Tuning with larger pre-trained models like Swin Transformer and DINO perform well in simpler tasks but diverge as the task complexity increases, with Prefix-Tuning consistently capturing more detailed visual information and adapting better to challenging conditions. This indicates that when computational resources are not a limiting factor, larger models with Prefix-Tuning offer a significant advantage in difficult navigation tasks.

2.2.4.3. Real-World Performance. In addition to simulations, we conducted real-world experiments using a `Unitree GO1`. The similar scenarios were replicated in a controlled environment with different number of obstacles. Since the onboard computer can't hold large model, when deploying, all the inference happens on the server then transmit the generated command to robot by a wire. The results in real-world testing showed a similar performance pattern to the simulation, with PreT-Swin achieving the highest success rates. However, the average speed in real-world scenarios was slightly reduced due to unforeseen friction and surface irregularities.

In the real-world experiments, we encountered a few failure cases: Terrain Adaptation Delays: In environment with obstacles, the robot needed a few more cycles to adjust its gait, leading to slower movement compared to its performance on relatively easier environment. A notable success of our model was its ability to perform well in dynamic environments, where human operators placed obstacles in the robot's path mid-task. This adaptability is a direct consequence of using multimodal inputs, allowing the robot to reassess its situation and recalibrate its foot positioning in real time.

2.2.5. Conclusion

In this paper, we presented a novel approach for enhancing quadrupedal locomotion by integrating large pre-trained vision models with reinforcement learning policies through prompt-tuning. Our method leverages the power of pre-trained vision models, providing the robot with a richer understanding of its environment without the need for extensive task-specific training. This leads to a significant reduction in training time while enabling the robot to handle complex terrains with improved perception and decision-making. Through our experiments on various terrains, including pyramid stairs, sparse, and dense obstacle environments, we demonstrated that models utilizing prompt-tuning and pre-trained transformers significantly outperform baseline methods. Particularly, our approach allowed for more effective obstacle avoidance and overall stability in navigation, especially in environments with greater complexity. Additionally, the use of prompt-tuning methods such as ProT and PreT provided parameter-efficient ways to adapt large models, with Prefix-Tuning showing superior performance in handling dynamic and challenging conditions.

While our method achieves promising results, future work will explore the application of other pre-trained models trained on larger, more diverse datasets, and examine how they can further improve quadrupedal locomotion. We also plan to investigate applying our approach to other robotic tasks and reinforcement learning algorithms, potentially expanding the scope of this work to new robotic domains.

In conclusion, the integration of pre-trained vision models with adaptive tuning methods marks a significant advancement in the field of robotic locomotion, offering an efficient, robust, and scalable approach for improving navigation in complex environments.

CHAPTER 3

System

This chapter presents DOS®, a Deployment Operating System for Robots that brings the Continuous Integration / Continuous Deployment (CI/CD) paradigm to data-driven robotics. With the methodological foundations of Chapter 2 in hand—multi-agent belief-embedded communication and visual-prompt locomotion—the natural next question is how such learned policies are reliably deployed, monitored, and continuously improved on real robot fleets. DOS® addresses precisely that question.

3.1. DOS: A Deployment Operating System for Robots

We propose a new system named DOS® (Deployment Operating System for RoboOps) for reliably deploying any data-driven robots in both production and simulation environments. Compared to existing systems, DOS® features a unique CI/CD (continuous integration and continuous deployment) architecture which allows us to seamlessly integrate agile development and reliable operation in a fully automated fashion. With this CI/CD architecture, this paper mainly introduces three essential components that uniquely differentiate DOS® from existing robotic systems: (i) A **cloud orchestrator** that build and schedule pipeline components; (ii) A **bridge tool** named DOS® Connect that make cloud-edge bidirectional communication feasible; (iii) An **analytical profiler** that

collects any set of user-defined performance metrics for system optimization. DOS® significantly increases the reliability and maintainability of the deployed robotic systems. To illustrate this point, we demonstrate performance of DOS® by training and deploying deep reinforcement learning based methods in a challenging real-world environment with a swarm of robots.

3.1.1. INTRODUCTION

In recent years, the robotic landscape has experienced a profound shift, with data-driven robots increasingly permeating various industrial sectors [**3, 9, 43, 59, 22, 38**]. The deployment of these robots in production and simulation environments necessitates systems that are both reliable and flexible to cater to the dynamically changing prerequisites. The integration of agile development practices with robust operational methodologies has hence become a cornerstone in achieving reliable and maintainable robotic systems. To this end, we introduce a novel system Deployment Operating System for RoboOps (DOS) crafted meticulously to address the aforementioned concerns and fill the existing gaps in the deployment landscapes.

DOS stands distinctly in the landscape owing to its unique Continuous Integration and Continuous Deployment (CI/CD) architecture, setting a new paradigm that fosters seamless integration between agile development and reliable operations. This automated architecture not only caters to the immediate needs of the rapidly evolving robotic systems but also presents a visionary approach to handling future challenges with a high degree of competence and precision.

Figure 3.1. This figure shows DOS® deploy a collision-free navigation algorithm to a swarm of robots. (Turtlebot Burger). The right-top figure is the Operating page where viewing from the coach robot (Pyrobot). User can manually move the robot by the joystick the page provides.

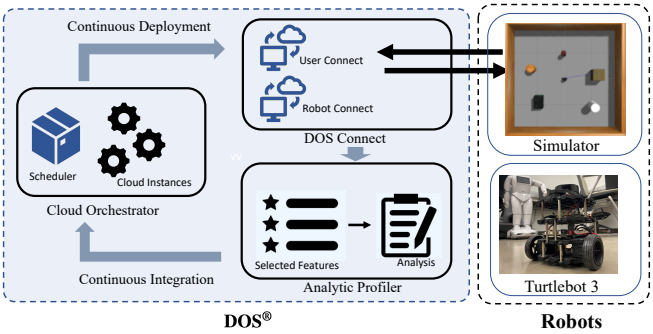


Figure 3.2. The overall architecture of the system. In our vision, the robot will act as data collector and command executor. Both model training and inference are accomplished at cloud. To achieve such goal, DOS® system built fully on cloud but has ability to interact with external simulators and real-world. The system contains an orcherstrator managing the whole CI/CD pipeline, a bridging tool named DOS® Connect and a monitoring component we call it Analytic Profiler which examine the performance of the model and system.

The innovation at the heart of DOS lies in its trifecta of fundamental components, each playing a pivotal role in enhancing the functionality and usability of robotic systems:

(i) Cloud Orchestrator: This segment of DOS is responsible for building and scheduling various pipeline components, creating a system that is both efficient and adaptive to the fluctuating demands of the deployment environments. (ii) DOS Connect: Acting as a bridge tool, DOS Connect facilitates bidirectional communication between cloud and edge components, thereby realizing a comprehensive communication strategy that stands tall

on the pillars of reliability and efficiency. (iii) Analytical Profiler: To ensure an optimized system performance, an analytical profiler is integrated into DOS to gather a user-defined set of performance metrics. This proactive approach to system optimization distinguishes DOS, presenting avenues for unprecedented improvements in system performance based on reliable data analytics. By amalgamating these elements into a singular system, DOS manifests as a powerhouse facilitating increased reliability and maintainability in deployed robotic systems. Moreover, it provides a framework where improvements are not reactive but proactively managed through a continuously monitored pipeline, ensuring optimal performance at all stages of deployment.

To corroborate the effectiveness and efficiency of DOS, we undertake a series of rigorous tests, deploying deep reinforcement learning-based methods in challenging real-world environments utilizing a swarm of robots. These experimental setups have been crafted to elucidate not only the competency of DOS in present scenarios but to also shine a light on its preparedness for future challenges.

In this paper, we delineate the architecture of DOS, delving deep into each component's functionalities and their collaborative potential to revolutionize the deployment of data-driven robots. We further embellish our assertions through practical demonstrations, showcasing the remarkable improvements achieved through DOS deployment in real-world scenarios.

By bringing DOS to the forefront, we aspire to set a new benchmark in the robotic deployment landscape, initiating a wave of reliable, efficient, and automated deployment solutions primed for the next generation of RoboOps.

The rest of the paper is organized as follows. Section 2 introduces recent machine learning deployment system both in academic and industry. Section 3 presents the concept, implementation and components of the DOS® system. Section 4 evaluates DOS® in the context of a participant study. We show the reliability and convenience of our system. Finally, Section 5 provides conclusions and future work.

3.1.2. RELATED WORK

Robotic Deployment Systems The deployment of robotic systems in various environments has been a topic of increasing interest in the recent decade. Previous systems have primarily focused on either production or simulation environments separately, with a limited scope in terms of integration and communication between the two [**96, 45, 2, 107, 81, 95, 135**]. For example, AWS RoboMaker is a development studio integrating Gazebo simulator[**50**]. It contains some common tools and simulators in robotic area, but configuring it can be painful. To deploy the application to hardware, another service called AWS IoT Greengrass 2.0 [**118**] is needed for deploying the robotic applications to a fleet. Teixeira et al. investigate CI/CD techniques in robotic repositories such as ROS packages (Robot Operating System) [**114**]. Canete et al. [**13**] proposed an energy-efficient deployment in edge-based infrastructure. Ahmadighohandizi et al. [**1**] present a framework for IoT applications. They design the workflow under the continuous deployment principle. The advent of DOS aims to amalgamate these environments to provide a seamless transition between the developmental and operational phases, therefore filling a vital gap in the existing literature and practical applications.

Cloud-Edge Communication Prior researches have delved into cloud-edge architectures, highlighting the potential for enhanced communication and data management [**111, 64, 30**]. Zhang et al. [**132**] proposed a fuzzy matching data sharing scheme for secure cloud-edge communication, which allows users to specify their access policies and enables receivers to obtain the data transmitted by the senders if they meet certain criteria. Xu et al. [**122**] designed a collaborative cloud-edge computing framework in distributed neural network to handle the computationally intensive tasks in the IoT environment. Wu et al. [**120**] highlighted the intuitions and key technologies of deep learning-driven wireless communication for edge-cloud computing. DOS takes a step further through the introduction of DOS Connect, a tool designed to facilitate bidirectional communication between cloud and edge components. This advancement stands to revolutionize the existing frameworks, providing a pathway for more nuanced and effective communication strategies in robotic deployments.

3.1.3. System Design

While designing our cloud robotic system, we naturally divided the architecture into two segments: the cloud side and the edge side, as depicted in Figure 3.3. The edge side incorporates users' personal computers, robots, and various sensors, while the cloud aspect shares traits with established machine learning platforms like KubeFlow and MLFlow [**94, 44, 53, 54, 97, 98**]. This segmentation laid the groundwork for our cloud orchestrator to build fundamental infrastructure, catering to routine robotic scenarios such as patrolling mobile robots in warehouses. Given the necessity for frequent model updates based on real-time feedback, we designed a scheduler to appropriately time these updates.

To ensure real-time safety and efficiency, we developed an analytical profiler module to monitor and analyze runtime logs. Despite the challenges in establishing communication between the cloud and robots, particularly due to the complexity of ROS middleware [**16**], we introduced DOS Connect, a tool facilitating a seamless bridge between both realms, simplifying the previously intricate configuration process.

3.1.3.1. Cloud Orchestrator. The backbone of DOS® system is on cloud, here we first choose AWS. But the IaC (Infrastructure as Code) tool we use named terraform can also easily transfer the infrastructure to other cloud platforms like GCP and Azure with little modification. We use this tool to build the DOS® cloud containing following components:

Network To ensure the security of our system, we divide our Virtual Private Cloud (VPC) into public zone and private zone. In public zone, we first construct scheduler EC2 instance that is equipped with a public IP. It acts as gateway connecting user computer and robots, all the data traffic goes through this instance for security reason.

Computing A scheduler is also implemented on scheduler instance. It is basically an Ansible playbook that contains lots of roles, task templates and source code scripts. Given a specific pipeline, the scheduler will launch few workers based on the pre-defined templates that containing the specs of EC2 instance like Amazon AMI, size of RAM and number of GPUs. The scheduler ensures the correctness of dependencies such as Apache Airflow, Apache Mesos, Kubernetes, and Nomad do. However, our tool is much light-wiser than them, we achieve this by automatically checking the completeness of data generated by previous step.

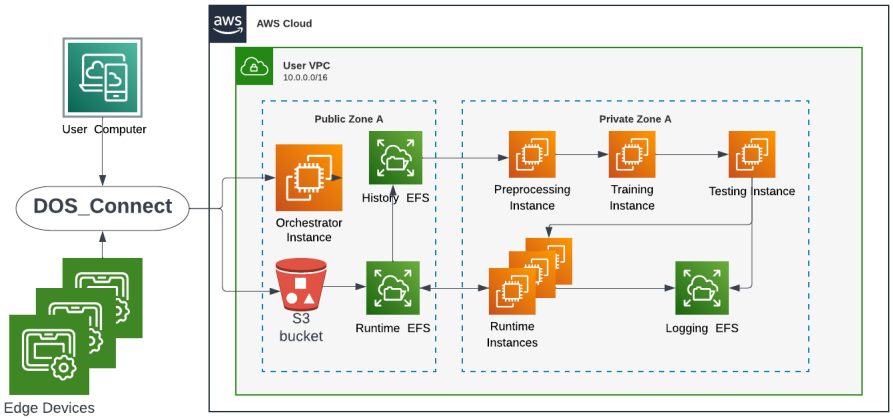


Figure 3.3. This figure shows how the DOS® is built on AWS and main instances containing their corresponding functionalities. Each component in pipeline is initialized as one AWS EC2 instance. The data flows in the same Elastic File System (EFS) on AWS that bound to all of them. With `DOS® Connect` installed, user computer, robots and edge devices are connected to the cloud through a gateway instance that facilitate the communication between the edge and the cloud. The real-time data transfer is achieved by the customized S3 bucket API.

Storage The scheduler will setup an instance to keep reading the new coming data from a S3 bucket which connects to robots and then transfer it to runtime Elastic File System (EFS) on AWS that bound to runtime instance executing real-time inference. The data also saves to history EFS that are bound to all instances within the pipeline. While training, these history data will be used for updating the model to new version.

3.1.3.2. DOS® Connect. Edge clouds are growing in significance due to the escalating requirements for low-latency, real-time data processing. The ideal setup delegates heavy computational tasks to the cloud, leveraging its substantial resources, while the edge end primarily functions as the executor. This strategy is pivotal in modern applications including Internet of Things (IoT), autonomous vehicles, and video streaming, which necessitate parallel and efficient processing of large data batches. The communication

between nowaday's cloud vendor and heterogeneous edge devices is barrier to lots of applications. Cloud vendors provide various tools and services like AWS Transit Gateway, Google Cloud VPN and Azure Application Gateway. All these services are tangled to configure. In our design, we treat all robots simply as a data collector and put all the computing to cloud. We write our own tool named `dos_connect`. It has two versions for different usages. One is for human user connecting to cloud. It can ssh into the terminal in scheduler instance or open VSCode in browser so user can modify and submit the code. Another version is for robots. The tool currently only supports Linux systems, it will install the encrypted login credentials, ROS dependencies and launch scripts refer to specific tasks. A S3 bucket with REST API is used for robots write real-time data into cloud so the data is accessible by the cloud computing resources. The ROS topic messages will be transformed to JSON format and then call the provided API to store it into bucket. We use this as an alternative way of AWS Greengrass 2.0 as it's too complicated for setting up. When all the ROS nodes are running, the underlying program will transform and transfer the ROS messages from user-specified topics to S3 bucket.

3.1.3.3. Analytic Profiler. Analytic Profiler locates at scheduler instance, it serves a web interface so user can check the real-time metrics they're interested in [**66**]. Analytic Profiler collects two kinds of data, user-defined ones like the odometry and total returns. The second is the system-wise metrics including GPU-specific and host-specific[**5**]. We list the metrics in Figure 3.4.

After the deployment, sometimes user want to switch the model based on the performance. We want to make this process automatic and smart based on specific rules. The rules can be defined by users or using tuning methods like bayesian optimization,

Figure 3.4. Analytical profiler first retrieves required messages from log EFS and host utilities provided by DOS® system. Based on user-defined rules, it then picks a candidate model from model registry and use canary release to update new policy to the whole fleet.

reinforcement learning-based optimization and grid search consume and make decisions based on the metrics like loss, accuracy and total rewards. During the transition, we also implemented the concept of a canary release from software deployment. This strategy involves initially updating a small group of robots to test the viability before deploying the update across the entire fleet.

3.1.4. Evaluation

We present a mobile robot example on DOS®. This application covers all components mentioned previously. To show the functionalities of full-fledged DOS® system. We first compare the inference time among pure-edge, pure-cloud and our hybrid system. Then we show the CI/CD ability by running a scheduled end-to-end pipeline which can automatically update the model every 2 hours. For last part, we demonstrate our system on a swarm of mobile robots and the specially designed web interface.

3.1.4.1. Environment Setup. We prepare several mobile robots all with DOS® Connect installed, DOS® hosts at AWS. There are three collision-free navigation policies for testing. The details are listed below.

Real Robot. Specifically, we install DOS® Connect on a mobile robot platform `Turtlebot3 burger`. This robot is equipped with LDS-01, a 2D LIDAR. The on-board

computer of `burger` is `Raspberry Pi 3`. The robot has all the drivers and ROS Kinetic installed and configured, so the user does not need to make any additional changes to it.

Navigation Algorithms. In order to test the generality and robustness of our system, we consider the following three reinforcement learning-based navigation methods: (i) CFNDRL [**130**], a collision-free navigation policy trained by path-following integration of the environment difficulty. (ii) CADRL [**24**], an agent-level decentralized method with each agent accesses precise state information. (iii) DRLMACA [**72**], a fully decentralized method in which the same policy is trained for all agents in several multi-agent environments. All these three algorithms take the laser data and odometry information as input.

Table 3.1. The following table presents a comparison of inference time for pure edge, pure-cloud, and DOS® system. Inference time is quantified in milliseconds, and it is determined by executing a pre-trained model on the respective platforms. The pure-edge platform involves running inference on a `Turtelbot 3` equipped with on-board computer Raspberry Pi 3, while pure-cloud involves running inference on an EC2 instance with GPU. DOS® system combines the two by offloading inference tasks to the cloud while sending back results to the edge. The table includes average inference time and standard deviation for each platform.

Algorithms	DOS®	Pure Edge (CPU only)	Pure Cloud
CFNDRL	121 ± 34	703 ± 502	5.9 ± 2.4
CADRL	104 ± 55	572 ± 138	2.6 ± 1.1
DRLMACA	112 ± 78	560 ± 221	3.6 ± 5.7

3.1.4.2. Inference Time. As the proliferation of IoT devices continues, coupled with a growing demand for real-time data processing, both edge and hybrid cloud computing have emerged as favored alternatives to conventional pure cloud computing solutions. In

this study, we undertake a comparative analysis of the inference time of three distinct collision avoidance algorithms, utilizing DOS®, pure edge, and pure cloud environments.

Given the prevailing trend towards large models, it has become increasingly improbable for users to deploy these models with billions of parameters directly to robots; this remains the case even when certain robots are outfitted with powerful hardware such as the Nvidia Jetson series. Consequently, our system assumes a crucial role in this context, acting as a facilitator to bridge the gap between cloud and edge computing resources. This ensures robots can leverage substantial computational resources, enhancing their functionality without being constrained by their onboard computational limits.

As illustrated in Table 1, when evaluating the inference time across various environments, we observe distinct performances characterized by the computational resources leveraged in each setting.

In a pure cloud setting, we note the most optimal performance, boasting the lowest average inference time of 2.6 ms. This is achieved by harnessing the full potential of extensive cloud computational resources, thereby mitigating any potential latency in algorithm execution. The pure cloud environment exhibits remarkable efficacy, particularly in handling the CADRL algorithm where the inference time is minimized to an average of 2.6 ± 1.1 ms.

Contrastingly, the edge environment, relying solely on on-board CPU-based PyTorch for inference — without the engagement in any pruning or optimization of the original model — understandably incurs the longest inference times, averaging at 572 ms. This is discernibly influenced by the inherent capacity constraints of utilizing a CPU-only setup, which despite its limitations, manifests a noteworthy performance in managing the

CADRL and DRLMACA algorithms with inference times of 572 ± 138 ms and 560 ± 221 ms respectively.

Meanwhile, the DOS® system adeptly navigates a middle ground, striking a harmonious balance between the edge and pure cloud environments. With an average inference time of 104 ms, it showcases a compelling performance, most notably when employing the CADRL algorithm, registering a time of 104 ± 55 ms. However, it is pivotal to note that a substantial portion of the inference time in DOS® is consumed in the transfer processes; approximately 68 ms is accounted for both 'put' and 'retrieve' operations, pointing towards an area with potential for optimization to further enhance the system's efficiency.

Upon scrutinizing the influence of model complexity on the performance across different environments, we identify that CFNDRL emerges as the largest model, succeeded by CADRL and DRLMACA in terms of size. Intriguingly, the variation in model complexities does not markedly impact the total inference time, indicating a level of stability and reliability in performance across different algorithmic complexities.

In conclusion, the analysis delineates a clear landscape where each environment, with its distinctive computational resources and strategies, exhibits unique performances. While the pure cloud environment capitalizes on expansive computational resources to ensure the quickest inference times, the edge environment bears the brunt of capacity limitations. The DOS® system astutely positions itself as a balanced contender, hinting at a promising avenue for fostering efficiency through further optimizations, particularly in data transfer processes.

3.1.4.3. Robotic CI/CD Pipeline. To showcase the functionalities of version control, analytic profiler, as well as the training and deployment processes in our system, we

(a) Total number of Warnings

(b) Odometry

(c) Model Switch Time

(d) Simulation Environment

Figure 3.5. After the successful deployment of the models, we test the system's performance based on the total number of "too close" warnings and odometry data gathered by robots. In our navigation algorithm, we count a warning number while the minimum value in one frame of laser scan is less than 0.2m. The odometry data is the distance robot traveled in 5 minutes. Using the analytical profiler in DOS®, the different models can be seamlessly switched by certain rules. (c) indicates the exact moment and the total number of warnings when the training job is trigger by scheduler. The orange star is the training with simulator and the blue circle represents real-world deployment. Map in (d) is reconstructed from the real test environment. The green box represents the turtlebot with the same range of scan.

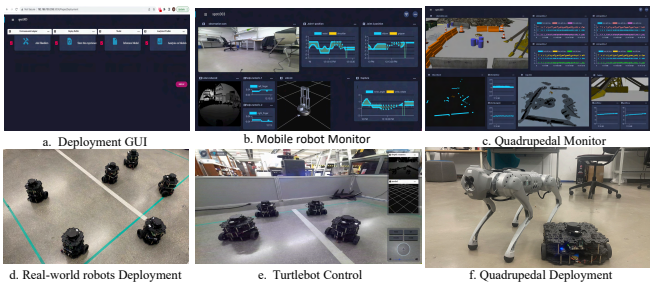


Figure 3.6. (a) shows the deployment page. Once users finish the customized modules, use the red button shown at the right-bottom to deploy the policy automatically. (b) and (c) are monitoring pages of a simulated mobile and quadrupedal robot deployment. The monitoring page displays data from analytical profiler like odometry and joint positions. (d) is the deployment to a swarm of real world turtlebots. (e) is the operating page, the view occupying most region is from the camera from a pyrobot[**80**], there are also depth camera view and control joysticks on the right side. and (f) are the deployment to the `Unitree Go1` quadrupedal robot.

formulated two jobs in accordance with our scheduler format. These are illustrated in Listing **??** and were submitted to the DOS® system for execution.

To elucidate further, we leveraged the inherent cron module, a prevalent time scheduling tool utilized within Ansible playbooks, to organize and schedule the tasks. The first

job, termed "update model," initiate at 9:05, invoking the training code within the simulator at ten-minute intervals thereafter. The second job is scheduled precisely at 9:25 to deploy the most recently trained model directly to the operational real robot. Throughout the operational period of the system, we consistently garnered data at five-minute increments, capturing metrics such as the total count of "too-close warnings" and odometry statistics representing the traveled distance. This data acquisition spanned both simulation and real-world environments, as depicted in Figure 3.5a and 3.5b.

A pertinent observation post deployment is the pronounced reduction in real-world robot odometry. This phenomenon can be attributed to the 'sim-to-real gap. By illustrating this process, we aim to delineate the seamless integration of training and deployment facilitated by the DOS® system, spotlighting its proficiency in harmonizing simulation exercises with real-time operational inputs, thereby championing efficiency and accuracy in robotic functionalities.

Listing 3.1. Example Scheduler Code

```
- name: update model
  cron:
    name: "model pipeline"
    minute: "5/10"
    hour: "9"
    job: python train_cfndrl.py --data_path /mnt/history_efs/observation.txt --
        ↪ log_path /mnt/log_efs/logging_1_30.json
```

```
- name: deploy to simulation and real-time
  cron:
    name: data integrity check
    minute: "25"
    hour: "9"
    job: bash deploy.sh --mode {{ item }} --model_path /mnt/scheduler_efs/
        ↪ model_latest.pth
  loop:
    - realtime
```

3.1.4.4. Multi-agent Deployment and Web Interface. To make the whole system more user-friendly, we develop a web interface for analytical profiler, we also run a quadrupedal example to show its scalability to different robots, see Figure 6f. The monitoring page visualizes users' interested topics and plots them over time. Considering user some time needs to intervene robot's operations, we add an operating page as Figure 7e shows. Users can also use the menu button on the top left to switch among robots. For swarm deployments, we choose the same navigation model CFNDRL and deploy it to 6 `Turtlebot3 Burger` robots, see Figure 6d.

3.1.5. Conclusion

In this study, we successfully introduced DOS, a groundbreaking Deployment Operating System for RoboOps, delineating its unparalleled functionality in orchestrating data-driven robotic deployments in both production and simulation environments. The system stands on three pillars: a sophisticated cloud orchestrator, the innovative DOS Connect

tool enabling bidirectional cloud-edge communication, and an analytical profiler designed for optimizing system performance through user-defined metrics. Our empirical demonstrations validate DOS's efficacy, showcasing a significant enhancement in the reliability and maintainability of deployed robotic systems. DOS holds a promising potential in bridging the existing divide, illustrating a robust pathway to the future of RoboOps deployments. For future work, we plan to deploy multiple tasks on heterogeneous platforms and scale the scope to more robotic applications.

CHAPTER 4

Synthesis

This chapter presents the capstone of the dissertation: DreamTacVLA, a touch-aware vision–language–action model for contact-rich manipulation. With the methodological advances of Chapter 2 and the deployment infrastructure of Chapter 3 in hand, DreamTacVLA brings methodological innovation and systems engineering together in a single end-to-end pipeline. Its hierarchical, touch-aware vision–language–action model is enabled by a high-fidelity tactile digital twin in IsaacSim and delivered through an end-to-end data engine that connects simulation, real hardware, training, and deployment.

4.1. DreamTacVLA: Learning to Feel the Future for Contact-Rich Manipulation

Vision-Language-Action (VLA) models have shown remarkable generalization by mapping web-scale knowledge to robotic control, yet they remain blind to physical contact. Consequently, they struggle with contact-rich manipulation tasks that require reasoning about force, texture, and slip. While some approaches incorporate low-dimensional tactile signals, they fail to capture the high-resolution dynamics essential for such interactions. To address this limitation, we introduce **DreamTacVLA**, a framework that grounds VLA models in contact physics

by learning to feel the future. Our model adopts a hierarchical perception scheme in which high-resolution tactile images serve as micro-vision inputs coupled with wrist-camera local vision and third-person macro vision. To reconcile these multi-scale sensory streams, we first train a unified policy with a Hierarchical Spatial Alignment (HSA) loss that aligns tactile tokens with their spatial counterparts in the wrist and third-person views. To further deepen the model's understanding of fine-grained contact dynamics, we finetune the system with a tactile world model that predicts future tactile signals. To mitigate tactile data scarcity and the wear-prone nature of tactile sensors, we construct a hybrid large-scale dataset sourced from both high-fidelity digital twin and real-world experiments. By anticipating upcoming tactile states, DreamTacVLA acquires a rich model of contact physics and conditions its actions on both real observations and imagined consequences. Across contact-rich manipulation tasks, it outperforms state-of-the-art VLA baselines, achieving up to **95%** success, highlighting the importance of understanding physical contact for robust, touch-aware robotic agents. Code, data, and videos are available at `https://michaelyeah7.github.io/learning-to-feel-the-future/`.

4.1.1. Introduction

Vision-Language-Action (VLA) models enable robots to leverage web-scale knowledge for general-purpose manipulation [**112, 47, 10**], but their success is largely limited to

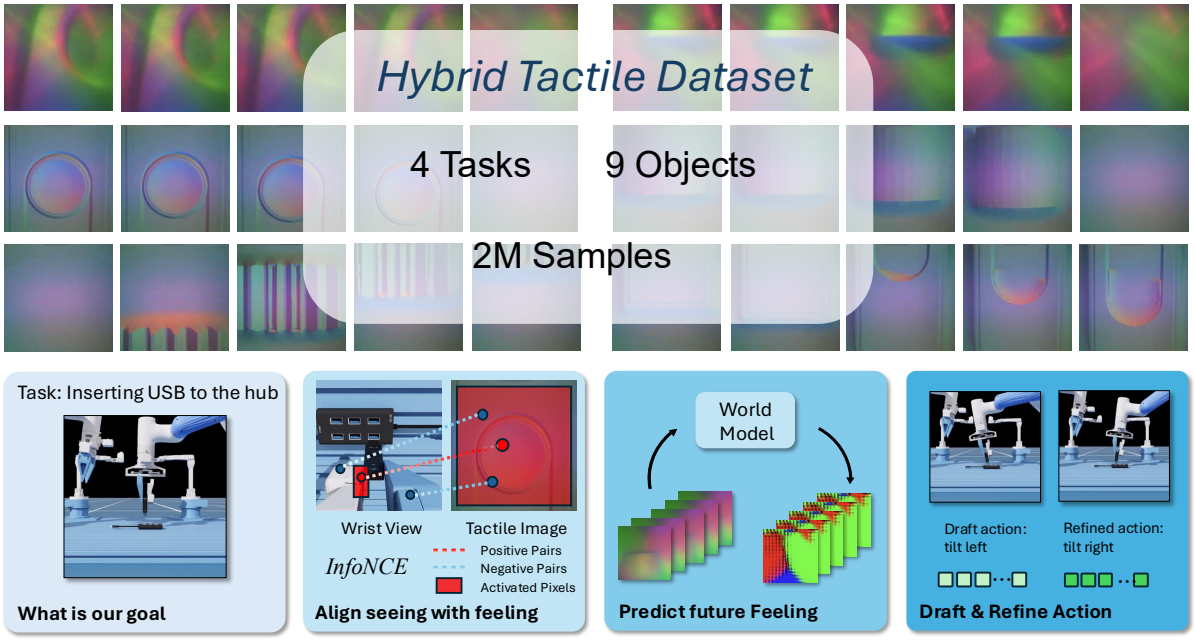


Figure 4.1. Hybrid tactile dataset and the Tactile-DreamVLA inference mechanism. (Top) We collect a large-scale tactile dataset covering 4 manipulation tasks and 9 objects, totaling 2M tactile frames. (Bottom) Our Think–Dream–Act loop executes each step of the policy in two passes. In the **Think** stage, the policy proposes a draft action using the current state and a null tactile prediction. In the **Dream** stage, a frozen V-JEPA2 world model forecasts the tactile outcome of that draft action. In the **Act** stage, the policy integrates both the real observation and the predicted tactile feedback to refine the action. This enables fine-grained corrections for contact-rich manipulation.

visually guided tasks. In contact-rich scenarios such as insertion or deformable object manipulation, VLA agents often fail due to the lack of tactile awareness. Although recent work has introduced tactile inputs into VLA pipelines [**39, 18**], these approaches rely on low-dimensional force or torque signals that are sparse and ambiguous, providing little information about how or where contact occurs.

To create robots capable of human-level dexterity, tactile sensing must be high-resolution and integrated across multiple spatial scales. However, scaling such tactile-aware models

poses a fundamental data challenge: visual tactile sensors are expensive and fragile, making large-scale real-world data collection prohibitively costly. We therefore construct a large-scale tactile data generation pipeline in simulation, complemented by a high-fidelity digital twin of the tactile sensor and manipulation environment to improve sim-to-real transfer. This hybrid strategy enables scalable tactile learning while maintaining physical realism.

However, data alone are not sufficient. Contact-rich manipulation inherently requires reasoning across multiple spatial scales, from global task context to fine-grained contact events. This motivates our hierarchical perception framework, which organizes sensory inputs into three levels: **macro** for arm-level task context, **local** for end-effector visual guidance, and **micro** for fingertip tactile cues such as slip and insertion forces (Figure 4.3).

Integrating information across these scales is non-trivial, as tactile signals differ fundamentally from visual inputs in both form and semantics. To bridge this modality gap, we establish spatial correspondence between vision and touch by mapping tactile activations to their locations in wrist and third-person views using robot kinematics and camera calibration. Based on this alignment, we learn a unified latent representation that enables joint reasoning over what the robot sees and what it feels.

However, alignment alone does not guarantee that VLA models will meaningfully use tactile information. Since vision–language backbones are pretrained without touch, naively appending tactile inputs often leads the model to ignore them. This is problematic because tactile sensing uniquely captures fine-grained contact physics, such as slip and local deformation, that vision cannot provide.

Meanwhile, conventional world models focus on predicting full RGB observations in high-dimensional latent spaces, which is computationally expensive and often unstable. In contrast, vision-based tactile images exhibit simpler structure and more constrained dynamics while remaining highly informative about contact interactions. Motivated by this, we introduce a tactile-centric world model that predicts future tactile signals in latent space (Figure 4.1). This predictive objective compels the model to actively use tactile information and learn the evolution of local contact physics.

Nevertheless, traditional world-model pipelines typically rely on a reward function and an MPC-style planner to derive actions, making them computationally heavy and difficult to deploy in contact-rich manipulation. To avoid this complexity, we adopt a two-stage learning framework that enables tactile-driven policies to emerge directly.

In the first stage, we train the policy using only the unified multimodal perception module, encouraging it to produce a draft action from aligned multi-scale observations. In the second stage, we activate the tactile world model to predict future tactile states, which are fused with the policy to refine actions based on anticipated contact outcomes. This design allows DreamTacVLA to reason over both present observations and imagined tactile futures without explicit planning, remaining lightweight and end-to-end trainable.

In summary, we make the following contributions:

- We introduce a novel contrastive loss for spatial alignment on multi-scale sensor data. This method aligns diverse perceptions into a single unified latent space.
- We introduce a tactile world model trained as a self-supervised objective to “dream" the future. By predicting high-resolution tactile signals, this model learns an implicit understanding of contact physics and material interactions.

- We propose a two-stage “Think-Dream-Act" policy that uses this “dreaming" capability for refinement. The policy first thinks of a draft action, then dreams its tactile consequences using the world model, and finally acts by outputting a refined, more precise command.
- We introduce a large-scale simulated tactile dataset, paired with a high-fidelity digital twin, which enables dense and diverse tactile supervision that would be prohibitively expensive to obtain in the real world.

4.1.2. Related Work

4.1.2.1. Vision-Language-Action (VLA) Models. Vision-Language-Action (VLA) models have become a dominant paradigm for general-purpose robot control, demonstrating strong cross-task and cross-embodiment generalization by scaling data, model capacity, and action representations [**11, 10, 112, 47, 8**]. Recent advances also improve VLA effectiveness via modular architectures and stronger adaptation recipes, boosting real-robot performance and efficiency [**62, 46**]. Despite these advances, most VLAs remain predominantly vision-centric and continue to struggle in contact-rich manipulation where tactile reasoning is critical, motivating efforts to incorporate touch into generalist policies via post-hoc adaptation [**42**]. Detailed discussion and additional references please refer to Appendix A.6.1.

4.1.2.2. Multimodal Grounding for Robotics. Multimodal grounding aligns perception with physical interaction by incorporating spatial and tactile cues beyond RGB. Spatially enhanced VLAs introduce explicit or implicit geometric priors (e.g., egocentric

3D encodings, point clouds, or structured traces) to improve geometric reasoning and manipulation robustness [**88, 60, 67**]. Tactile grounding further augments contact awareness, yet many policies still rely on low-dimensional force or compressed touch signals that miss fine-grained contact geometry; recent work instead fuses high-resolution tactile sensing for multimodal reasoning and reactive correction in contact-rich tasks [**71, 39, 18, 124**]. In parallel, tactile representation learning targets transferable visuotactile embeddings via self-supervision or cross-modal alignment [**126, 26, 36**]. Our work leverages vision-based tactile micro-vision to capture texture, geometry, and shear-induced slip for fine-grained contact modeling. Detailed discussion and additional references please refer to Appendix A.6.2 A.6.3.

4.1.2.3. Predictive World Models in Robotics. Predictive world models learn latent dynamics that capture temporal and physical regularities, enabling agents to anticipate future evolution for decision making [**28, 29**]. Large-scale pretraining further yields structured representations whose latent spaces encode semantics and dynamics beneficial for downstream control [**82, 4**]. Recent VLA architectures integrate such predictive modeling into policy learning by conditioning actions on latent rollouts [**134, 15**]. In the tactile domain, prior work mostly studies representation learning or autoregressive tactile prediction [**35**], whereas our work predicts high-resolution tactile futures and aligns them with visual observations for fine-grained contact reasoning. Detailed discussion and additional references please refer to Appendix A.6.4.

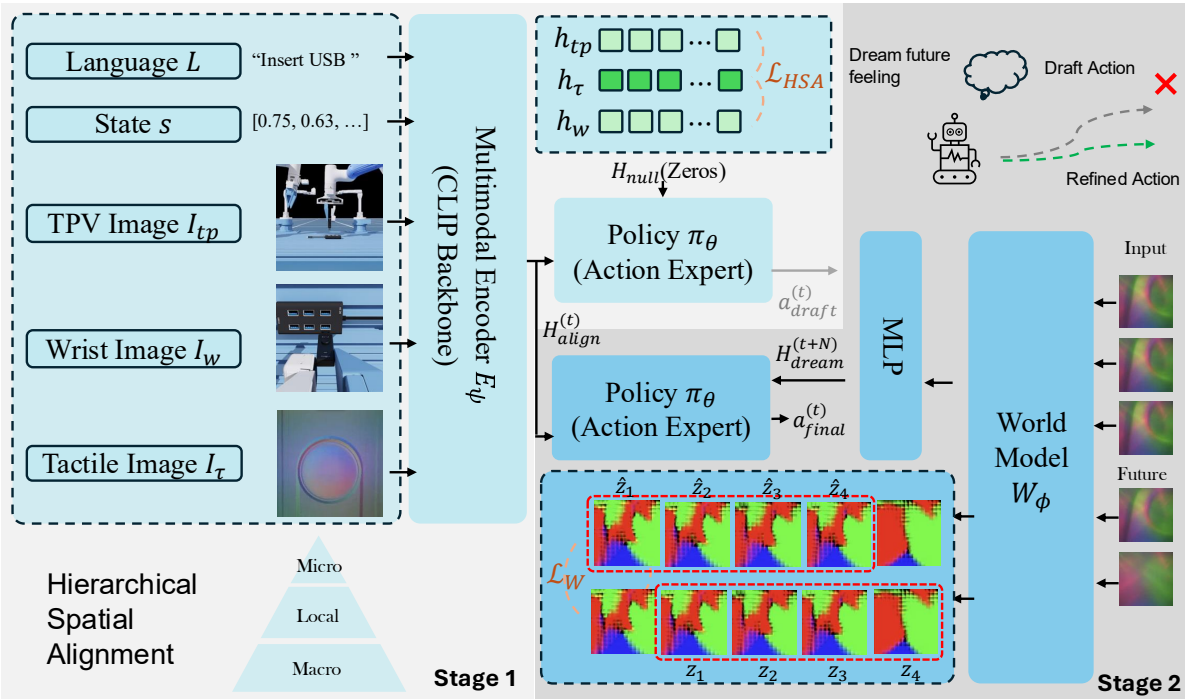


Figure 4.2. The proposed framework operates in two stages. **Stage 1 (Left):** A multimodal encoder E_ψ processes diverse inputs. This stage employs Hierarchical Spatial Alignment (HSA) to effectively fuse the features from different modalities, guided by the $\mathcal{L}_{HSA}$ and $\mathcal{L}_W$ losses. A policy π_θ is trained to output an initial draft action $a^{(t)}_{\text{draft}}$. **Stage 2 (Right):** A world model W_ϕ is trained to predict future tactile image sequences. The policy "dreams" the future tactile feeling (e.g., $H^{(t+N)}_{\text{dream}}$) that would result from its draft action. This predicted future is fed into an MLP, allowing the policy to refine its plan and output a more robust final action $a^{(t)}_{\text{final}}$.

4.1.3. Methodology

Our model, DreamTacVLA, is designed to learn robust, contact-rich manipulation skills by integrating high-resolution vision-based tactile images with standard visual (third-person and wrist camera) and language inputs. Our architecture, shown in Figure 4.2, is a unified, end-to-end framework built on a shared LLM backbone. It consists of three main components:

Multimodal Encoders (E_ψ): We employ modality-specific encoders to process all sensory streams: a CLIP ViT encoder for third-person and wrist images, and also for language prompts, and an MLP for robot state. Each modality produces a set of feature tokens that are concatenated into a unified token sequence. During Stage 1, the vision and tactile encoders are trained with our Hierarchical Spatial Alignment (HSA) loss. This yields a spatially-aligned multimodal representation $H_{align}^{(t)}$.

Tactile World Model (W_ϕ): A world model that acts as an implicit physics engine. It takes the current tactile image and a draft action $a_{draft}^{(t)}$ to predict the future sensory state $H_{dream}^{(t+N)}$, where N is the future horizon predicted by the model.

Unified Policy (π_θ): Our policy consists of a CLIP-based [**90**] multimodal encoder paired with an Action Expert transformer. The Action Expert operates in two passes: first, it drafts an action $a_{draft}^{(t)}$ based only on the current state. Second, it generates a refined, final action $a_{final}^{(t)}$ based on both the current state $H_{align}^{(t)}$ and the dreamed future state $H_{dream}^{(t+N)}$.

This Think–Dream–Act loop, implemented through our two-stage training procedure, enables the policy to internally verify its decisions by forecasting the physical consequences of candidate actions prior to execution.

4.1.3.1. Stage1: Pre-training Spatial Alignment & World Model. The foundational goal of this stage is to train the model's encoders to understand where the tactile sensor is in relation to the visual world, and to learn a baseline action policy. This is achieved by simultaneously optimizing two losses: the action loss ($\mathcal{L}_{action}$) and our novel Hierarchical Spatial Alignment ($\mathcal{L}_{HSA}$) loss.

Hierarchical Spatial Alignment (HSA). To enable the model to fuse information across the three visual scales (TPV, wrist, and tactile), it must understand where the tactile sensor is located within the other camera views. We enforce this understanding through a Hierarchical Spatial Alignment (HSA) loss.

Macro Vision (Third-Person) **Local Vision (Wrist)** **Micro Vision (Tactile)**

Figure 4.3. The three-scale visual hierarchy of our model. Our framework fuses information from three distinct visual modalities. Our Hierarchical Spatial Alignment (HSA) loss is designed to explicitly ground the micro-vision (what the robot feels) within the local and macro visual contexts (what the robot sees).

First, using the robot's forward kinematics and calibrated camera parameters (extrinsics E_{tp}, E_w and intrinsics K_{tp}, K_w), we find the 3D pose of the tactile sensor $P_{sensor}^{(t)} \in SE(3)$. We then project this pose to find its corresponding 2D bounding box in both camera views: $\mathcal{B}_w^{(t)}$ in the wrist view and $\mathcal{B}_{tp}^{(t)}$ in the third-person view.

From an intermediate layer of the LLM, we extract the feature tokens $H_{mid}^{(t)}$. We compute three mean-pooled feature vectors: (1) h_τ: The mean-pooled embedding of all tactile tokens $Z_\tau^{(t)}$. (2) h_w: The mean-pooled embedding of all wrist-view tokens whose spatial positions fall within the projected bounding box $\mathcal{B}_w^{(t)}$. (3) h_{tp}: The mean-pooled embedding of all third-person view tokens within $\mathcal{B}_{tp}^{(t)}$.

We then apply a token-level InfoNCE contrastive loss to pull these corresponding representations together. The loss for aligning the tactile view with the wrist view is:

$$\mathcal{L}_{\text{HSA-W}} = -\log \frac{\exp\left(\frac{h_\tau \cdot h_w}{\kappa}\right)}{\exp\left(\frac{h_\tau \cdot h_w}{\kappa}\right) + \sum_{i=1}^{N_k} \exp\left(\frac{h_\tau \cdot h_{w,i}^{\text{neg}}}{\kappa}\right)}. \tag{4.1}$$

where $h_{w,i}^{\text{neg}}$ are N_k negative samples (e.g., tokens from other regions or other images in the batch) and κ is a temperature parameter. A similar loss, $\mathcal{L}_{\text{HSA-TP}}$, is computed between h_τ and h_{tp}. The total alignment loss is:

$$\mathcal{L}_{\text{HSA}} = \mathcal{L}_{\text{HSA-W}} + \mathcal{L}_{\text{HSA-TP}}. \tag{4.2}$$

This loss explicitly forces the model to learn that the *micro-vision* tactile image corresponds to specific, localized regions in the *macro-vision* camera feeds.

Action Loss. The Action Expert is trained with a behavior cloning objective. Given the aligned multimodal tokens, it predicts an H-step action sequence $\hat{A}^{(t)}$, which we supervise using the expert actions $A^{(t)}$. We apply an ℓ_1 loss over the horizon:

$$\mathcal{L}_{\text{action}} = \frac{1}{H} \sum_{j=0}^{H-1} \left\| \hat{a}_j^{(t)} - a_j^{(t)} \right\|_1. \tag{4.3}$$

This trains the action expert to reproduce expert trajectories from the fused multimodal inputs. The total loss for Stage 1 is a weighted sum of these two objectives:

$$\mathcal{L}_{\text{Stage 1}} = \mathcal{L}_{\text{action}} + \lambda_{\text{HSA}} \mathcal{L}_{\text{HSA}}. \tag{4.4}$$

Upon completion of this stage, we have a competent baseline policy that understands where its tactile sensor is and how to perform basic actions.

In this stage, the goal is to train the HSA encoders and a baseline policy. We don't have a trained world model yet, so we cannot generate a dreamed future. To solve this, we feed the policy a zero-tensor in place of the input $H_{dream}^{(t+N)}$.

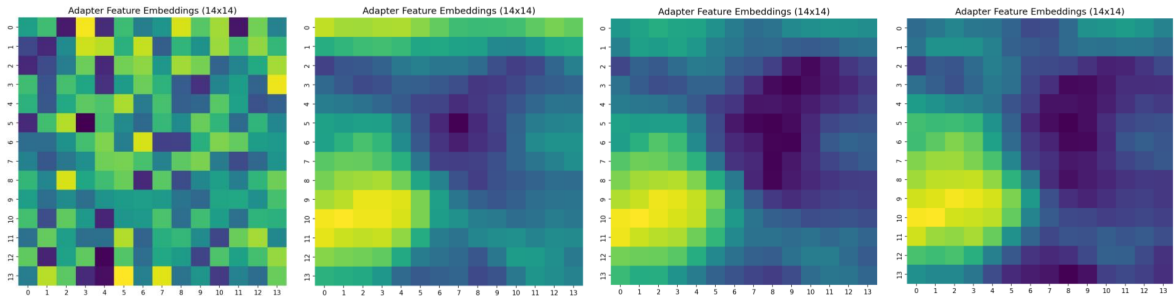


Figure 4.4. Visualization of the world model's predicted future-state embedding H_{dream} across training. Initially, the embedding is noisy and unstructured, indicating weak predictive ability. As training advances, the embedding becomes increasingly concentrated and stable, revealing that the world model is learning a coherent representation of future tactile–visual dynamics.

Pretrained Tactile World Model (W_ϕ). A key component of our architecture is a pretrained, frozen world model, W_ϕ, which functions as a powerful tactile feature extractor. We pre-train this model on a large, unlabeled dataset of tactile image sequences. W_ϕ (V-JEPA2 [**4**]) is trained to be an expert in tactile physics. Its job is to take a tactile image I_τ and encode it in a rich, latent embedding z_τ that captures the underlying physical state, as shown in Figure 4.4.

$$z_\tau = W_\phi(I_\tau)\,. \tag{4.5}$$

Throughout all subsequent training stages, W_ϕ remains frozen, providing a stable and high-quality embedding of tactile information.

4.1.3.2. Stage 2: Finetuning with a Latent Dream. The goal of this stage is to finetune the entire pre-trained system (π_θ and E_ψ) to learn a robust model of physical interaction. We achieve this by introducing a lightweight Forecasting MLP (F_η), which learns to dream of the latent sensory consequences of a draft action. This predicted future tactile embedding is then fed back to the policy, allowing it to make a more informed, physically-grounded final decision. During this stage, the main encoders (E_ψ) and the policy (π_θ) are finetuned, while the new forecasting MLP (F_η) is trained from scratch. The pre-trained tactile world model (W_ϕ) remains frozen to act as a stable feature extractor. We continue to apply the action loss ($\mathcal{L}_{action}$) and the Hierarchical Spatial Alignment loss ($\mathcal{L}_{HSA}$), while adding the new latent forecasting loss, $\mathcal{L}_W$. The Think-Dream-Act pipeline in this stage now functions as follows:

THINK: Policy π_θ generates a draft action $a^{(t)}_{draft}$ based on the current aligned state $H^{(t)}_{align}$ and the null dream H_{null}.

DREAM: MLP Forecasting F_ϕ predicts the future latent tactile state, $H^{(t+N)}_{dream}$. It takes two inputs: the current tactile embedding (from the frozen W_ϕ) and the draft action (from the policy):

$$H^{(t+N)}_{\text{dream}} = F_\eta\left(z^{(t)}_\tau,\, a^{(t)}_{\text{draft}}\right). \tag{4.6}$$

ACT: This predicted future embedding, $H^{(t+N)}_{dream}$, is fed back to the policy π_θ along with the current state $H^{(t)}_{align}$ to produce the refined, final action $a^{(t)}_{final}$.

4.1.4. Experiments

The primary hypothesis of this work is that for a robotic agent to achieve robust, contact-rich manipulation, it must not only react to the physical world but reason about its physical consequences. We posit that this capability is unlocked by combining two key components: (1) a high-resolution, spatially-grounded understanding of the current contact state (enabled by HSA) and (2) a predictive world model that can dream the future tactile images. We design our experiments to rigorously validate this hypothesis by dissecting our model's contributions.

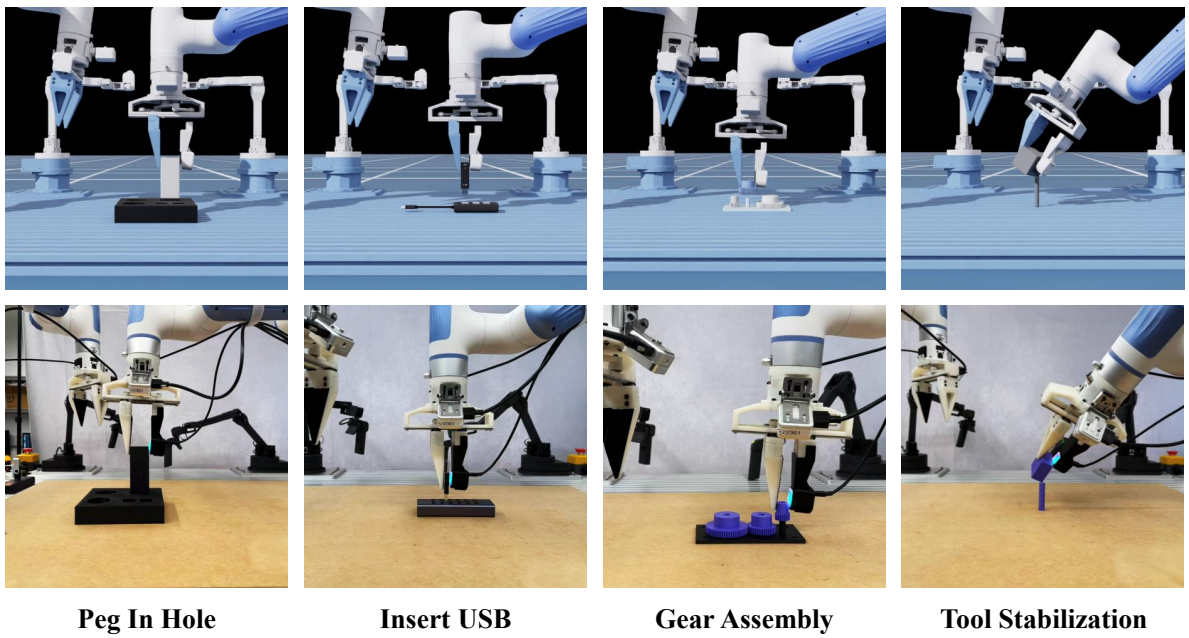


Figure 4.5. Task suite used to evaluate DreamTacVLA. From left to right: Peg-in-Hole, USB Insert, Gear Assembly, and Tool Stabilization. Each task demands precise, contact-rich manipulation, including aligning tight tolerances, detecting slip, or maintaining stable tool contact. It provides a comprehensive benchmark for assessing tactile-aware policies.

4.1.4.1. Experimental Setup.

Table 4.1. Task Success Rates (%) in Real-world (100 trials). Results are reported as mean ± standard deviation over 3 runs.

Model	Peg-in-Hole	USB Insert	Gear Assembly	Pen Stabilize
ACT [**138**]	$35.2 \pm 0.7\%$	$62.6 \pm 0.5\%$	$22.4 \pm 0.8\%$	$19.3 \pm 0.6\%$
Diffusion Policy [**19**]	$35.5 \pm 0.9\%$	$56.3 \pm 0.8\%$	$33.1 \pm 0.7\%$	$30.4 \pm 0.9\%$
$\pi 0$ [**8**]	$48.7 \pm 1.0\%$	$59.4 \pm 0.9\%$	$45.2 \pm 1.1\%$	$41.0 \pm 0.8\%$
ACT (w/ Tactile)	$45.6 \pm 0.3\%$	$63.1 \pm 0.4\%$	$40.2 \pm 0.2\%$	$39.1 \pm 0.7\%$
Ours (HSA-Only, No Dream)	$60.8 \pm 0.9\%$	$63.7 \pm 0.8\%$	$51.5 \pm 1.0\%$	$42.9 \pm 0.7\%$
Ours (No HSA, Dream-Only)	$75.4 \pm 0.8\%$	$75.2 \pm 0.7\%$	$64.9 \pm 0.6\%$	$68.5 \pm 0.9\%$
Ours (HSA & Dream)	$\mathbf{95.0 \pm 0.2}\%$	$\mathbf{85.7 \pm 0.6}\%$	$\mathbf{81.1 \pm 0.4}\%$	$\mathbf{74.6 \pm 0.5}\%$

Implementation Details. The full system consists of a language-conditioned policy, modality-specific encoders, a frozen tactile world model with lightweight adapters, and an action transformer expert. Below we detail each component.

Model Architecture. Policy and Encoders: The policy π_θ (Language Backbone) is initialized from a pretrained CLIP (clip-vit-large-patch14) model [**90**] and finetuned on our dataset. This CLIP model is also responsible for aligning wrist camera and tactile images. The tactile image (I_τ) encoder is a V-JEPA2 model [**4**] (ViTL/ViTG), initialized from its official pre-trained weights.

Action Expert: Our action expert is an action transformer, which is trained to predict a 7-DOF action (6D end-effector pose + 1D gripper state) over a 45-step horizon. The same horizon is used during inference.

World Model and Tactile Adaptation. We employ V-JEPA2 ViT-L/Vit-G as our tactile world model, pretrained on tactile images from our dataset and frozen during policy training. The pretrained encoder (in the case of ViT-L) produces 1024-dimensional patch embeddings. To enable the policy to refine its draft actions using tactile context while still preserving the pretrained representation, we insert a lightweight residual adapter after the

frozen encoder. The adapter processes all patch tokens (not just the CLS token) through a 3-layer bottleneck MLP with GELU activations and dropout (p=0.1). A learnable residual scale, initialized to 0.1, controls the magnitude of adapter features added to the frozen representations. We aggregate the adapted patches using learned attention pooling, where a single learnable query token attends to all 196 adapted patches via 8-head multi-head attention. This architecture adds only 5.5M trainable parameters (1.8% overhead) to the 300M frozen ViT-L, enabling efficient task-specific adaptation while retaining the world model's learned dynamics. The adapter and pooling weights are optimized jointly with the policy using AdamW (lr=$1e-5$, weight decay=$1e-4$).

Simulation & Hardware. We conduct experiments in both simulation and the real world. Our simulation environment is built in IsaacSim [**84**]. To enable realistic, high-fidelity tactile data collection, we integrate a physics-based tactile sensor model based on the work of TacEx [**83**]. This integration follows the Taxim [**104**] style optical and texture-based tactile simulation approach, which synthesizes gel deformation appearance through light-transport modeling and marker-texture warping. This allows us to generate realistic, high-resolution tactile images that closely mimic our real-world sensors, which is critical for large-scale data collection (1000 demonstrations per task) with randomized object poses in parallel environments. Our real-world setup uses a Dobot Xtrainer platform with a parallel gripper, two high-resolution GelSight [**131**] sensors, and two Realsense D405 cameras as wrist and third-person cameras. We collect 100 expert demonstrations for each real-world task. Figure 4.7 provides a qualitative comparison of the data streams from simulation and real-world hardware execution.

Tasks. We evaluate four challenging contact-rich tasks. As shown in Figure 4.5. 1) Peg-in-Hole: A classic robotics task requiring high precision. The port is partially occluded, forcing the policy to rely on tactile feedback for the final alignment. 2) USB Insertion: Inserting a USB-A plug into a port. This task has extremely tight tolerances (sub-millimeters) that are ambiguous from vision alone. 3) Gear Assembly: Sliding a small gear onto a shaft. This requires aligning the gear's hole with the shaft, a task that easily failed due to misalignment. 4) Tool Stabilization: The agent grips a cube and uses one of its vertices to support a thin vertical cylinder on the tabletop, maintaining the cylinder in a stable upright pose under small disturbances. We constructed a hybrid dataset consists of approximately 80% simulated demonstrations and 20% real-world demonstrations across four task categories, as illustrated in Figure 4.6.

Baselines. We compare DreamTacVLA against strong state-of-the-art policies and controlled ablations of our own method. External baselines include ACT [**138**], Diffusion Policy [**19**] and $\pi 0$ [**8**]. We also insclude a standard "ACT + Tactile" baseline to isolate whether the performance gain comes from the complex "Dreaming" architecture or simply from the availability of tactile data. We also evaluate several variants of our model:

Ours (HSA-Only, No Dream): Stage-1 variant that uses HSA-aligned encoders but relies solely on the current state, removing the contribution of the world model.

Ours (No HSA, Dream-Only): Ablation trained without the $\mathcal{L}_{HSA}$ loss, used to test whether spatial alignment can be learned implicitly.

Ours (HSA & Dream): Our full Stage-2 model incorporating both the HSA and the Think–Dream–Act pipeline.

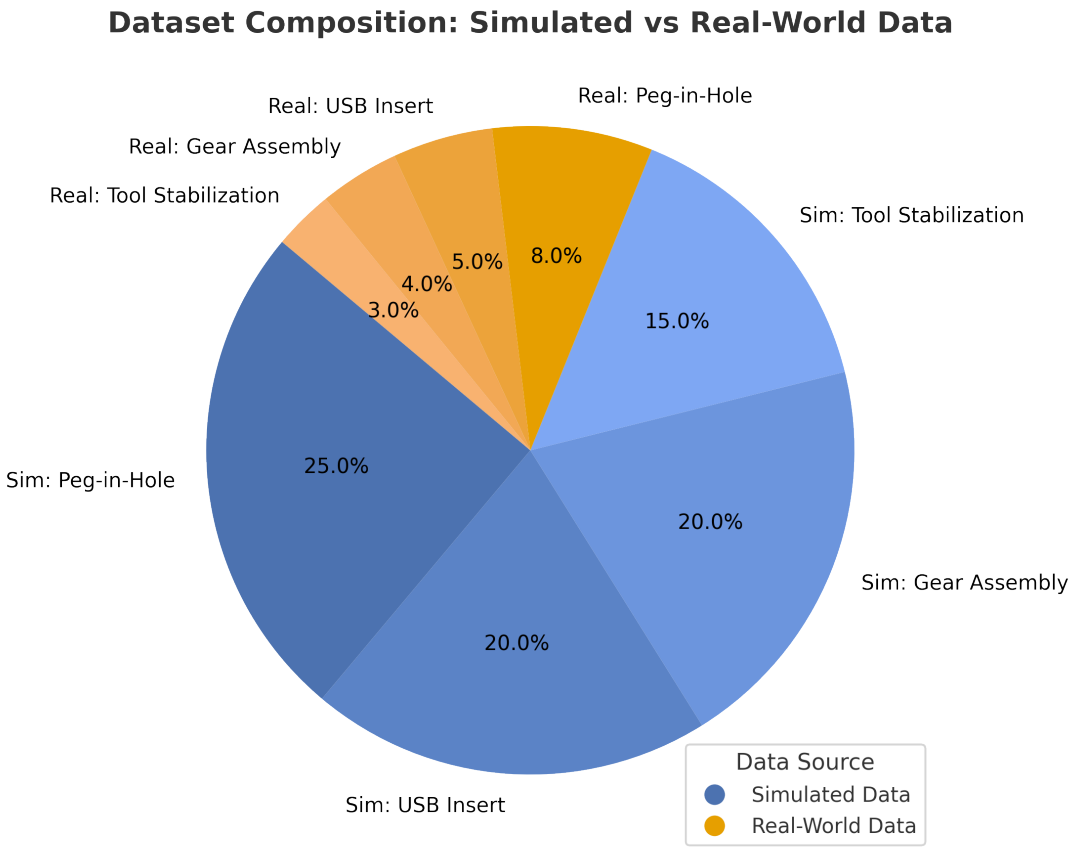


Figure 4.6. The dataset consists of 80% simulated demonstrations and 20% real-world demonstrations, each containing four task categories: Peg-in-Hole, USB Insert, Gear Assembly, and Tool Stabilization. Blue segments represent simulated data, while orange segments denote real-world data.

4.1.4.2. Main Results. We evaluate all models by measuring their task success rate (SR) over 100 trials for each task in four real world tasks. As shown in Table 4.1, our full model achieves the highest performance across all contact-rich manipulation tasks.

Vision-only baselines (ACT [**138**], Diffusion Policy [**19**]) perform poorly, especially on tasks like USB Insertion and Gear Assembly, where visual ambiguity and depth occlusion are significant. The Diffusion Policy baseline demonstrates moderate competence but fails to capture the fine-grained temporal consistency required for stable contact handling,

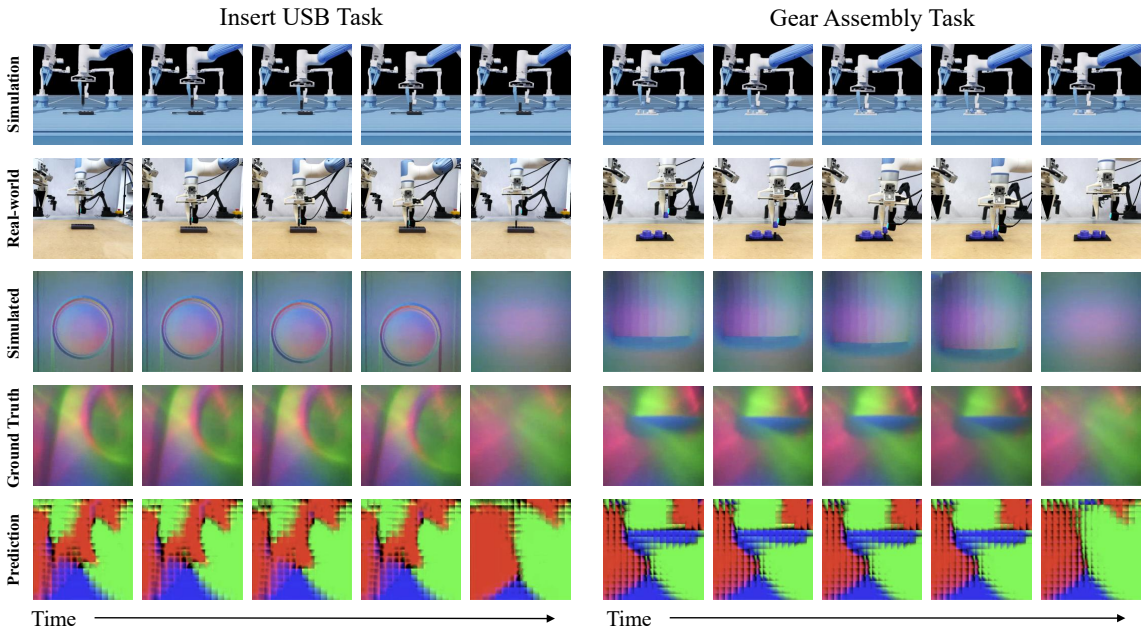


Figure 4.7. Qualitative comparison of our model's tactile prediction. For both the Peginhole and Tool Stabilization tasks, we visualize the sequence (left to right) comparing our model's Prediction (bottom row) to the Ground Truth tactile data (fourth row). The corresponding tactile images are provide as well.

often oscillating or prematurely retracting during insertion. The ACT with tactile baseline also shows that tactile modality can bring some performance gain but to use our HSA and 'Dream' method can better utilize it.

DreamTacVLA consistently outperforms all baseline and ablated models, with the best performance observed in USB Insertion and Peg-in-Hole. Both tasks demand precise micro-slip perception, iterative pose refinement, and stable contact maintenance. These capabilities are difficult to achieve with vision-only or feedforward tactile policies. The Think-Dream-Act mechanism is particularly influential in these settings: as the end-effector approaches the socket or hole, the policy executes controlled residual adjustments rather than committing to a simple open-loop motion. This behavior indicates that the

tactile world model provides high-frequency predictive feedback that guides fine-grained corrections.

These benefits are especially pronounced in Peg-in-Hole, a task where small variations in initial grasp or wrist orientation frequently lead to failure for baselines and ablations. DreamTacVLA handles such variations reliably, even when trained with only 50 demonstrations, suggesting that the combination of high-resolution tactile sensing, Hierarchical Spatial Alignment (HSA) and temporal tactile prediction provides strong physical grounding. The model not only predicts local contact dynamics but also leverages them for robust online refinement, resulting in higher success rates and improved generalization under perturbations.

4.1.4.3. Ablation Studies. We conduct detailed ablations to validate our key design choices.

Effect of HSA and World Model. As shown in Table 4.1, Although the policy still attempts continuous corrections, it frequently misaligns with the socket or hole and fails to recover, revealing that spatial grounding cannot be learned implicitly. Removing the world model ("HSA-Only") preserves coarse alignment but removes temporal foresight; without draft refinement from the dreaming stage, the policy no longer performs the fine residual adjustments needed near the target and behaves inconsistently. The full model (HSA + World Model) achieves an average 22.3% improvement over both ablations, demonstrating that reliable insertion behavior emerges only when spatial grounding and temporal imagination are combined.

World Model Sizes. We ablate the components of our world model ($\mathcal{L}_W$). As shown in Figure 4.8, training a world model to predict *only* future visual images (like DreamVLA

[**134**]) provides a minor boost. The tactile forecast is the most critical component. However, training the model to predict *all* future modalities (Tactile+Vision) yields the best results, as it learns a more consistent cross-modal physics model.

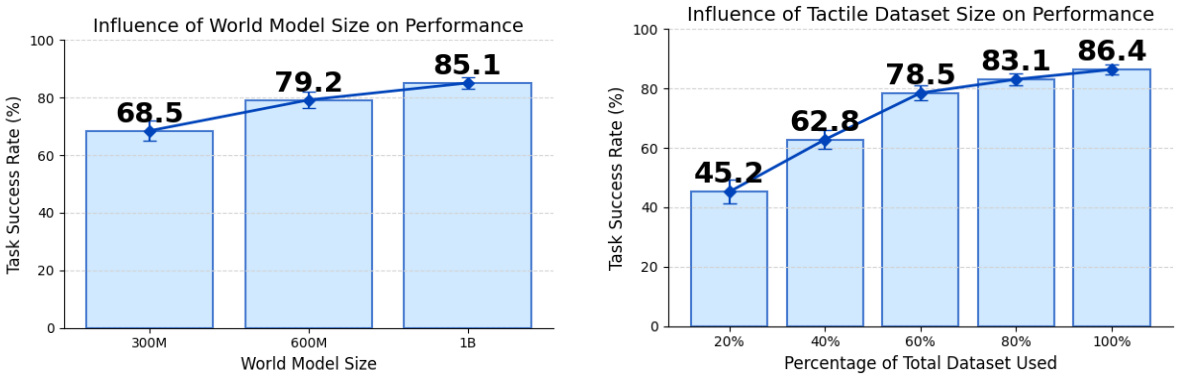


Figure 4.8. Ablation studies on model and data scaling.

Tactile Dataset Size. We further investigated the influence of the tactile dataset size on our model's performance. To do this, we trained separate instances of our model using progressively larger subsets of our collected data, ranging from 20% to 100% of the total available samples. Figure 4.8 illustrates the relationship between dataset size and task success rate. We observed a consistent improvement in performance as the number of training data increased. In particular, the model begins to converge towards stable performance at approximately 60% of the dataset size, suggesting that our current data collection is sufficient for the investigated tasks. However, the continued slight upward trend indicates that further scaling of various tactile data could yield additional, albeit diminishing, marginal gains.

4.1.5. Conclusion

We present DreamTacVLA, a physically grounded Vision-Language-Action framework that addresses the contact-blindness of vision-centric policies. The method combines a Hierarchical Spatial Alignment (HSA) loss that tightly grounds tactile, wrist, and third-person cues, and a Think–Dream–Act strategy that uses a tactile world model to forecast visuotactile outcomes of draft actions, enabling anticipatory contact reasoning.

Across four contact-rich manipulation tasks, DreamTacVLA consistently surpasses vision-only and force-based baselines and achieves near-perfect performance in both simulation and real settings. Ablation studies confirm the complementary roles of tactile grounding and tactile forecasting.

Although Think–Dream–Act adds inference overhead, future work will explore policy distillation and adaptive dreaming for faster single-pass reasoning. Scaling tactile world models with larger multimodal corpora offers a path toward more general agents that can reason about physical interactions with human-like intuition.

CHAPTER 5

Conclusions and Future Work

5.1. Summary

This dissertation has argued that embodied artificial intelligence is fundamentally an end-to-end discipline. Methodological breakthroughs realize their potential only when paired with the systems infrastructure that makes them deployable, scalable, and reliable on real hardware. Four sole-first-author works, organized into three technical chapters, trace this argument from methodological foundations to a full-stack synthesis.

Chapter 2 presented two methodological advances in reinforcement learning for embodied agents. The first introduced an inferring beliefs framework in which multiple agents learn to coordinate by inferring one another's intentions through emergent communication, validated across cooperative multi-agent benchmarks. The second introduced a visual-prompt architecture that adapts large pre-trained vision transformers to quadrupedal locomotion through parameter-efficient prompt and prefix tuning, trained at scale in GPU-parallel simulation and deployed on a Unitree Go1.

Chapter 3 presented DOS®, a cloud-scale deployment operating system that turns one-off robot deployments into a continuous CI/CD loop. Three reusable abstractions—an environment adapter, a data replay reservoir, and an analytical profiler—make fleet-scale operation tractable and turn live operation into continuous improvement of the deployed policies.

Table 5.1. Summary of the four works that make up the dissertation, and the methodological and systems advances each contributes.

Work	**Methodological advance**	**Systems advance**
Belief Embedded Comm. (§2.1)	Multi-agent RL with VAE belief inference	Cooperative multi-agent benchmarks
Visual-Prompt Loco. (§2.2)	Prompt / prefix tuning of pretrained vision	IsaacGym + Unitree Go1 sim-to-real
DOS® (Ch. 3)	CI/CD orchestration for robotics	Cloud robot fleet + reusable abstractions
DreamTacVLA (Ch. 4)	Touch-aware VLA with tactile world model	IsaacSim tactile twin + end-to-end data engine

Chapter 4 presented DreamTacVLA, a touch-aware vision–language–action model that unifies methodological and systems innovation for contact-rich manipulation. A hierarchical spatial alignment loss ties together macro-, local-, and micro-scale visual inputs; a tactile world model lets the policy condition its actions on imagined contact futures; and an end-to-end data engine spans a high-fidelity tactile digital twin and real hardware. DreamTacVLA achieves up to 95% success on real-world contact-rich manipulation.

5.2. Reflection: Why End-to-End?

It is tempting to read the four works as a chronological story: methods improved, infrastructure got better. That reading misses the central insight. The works succeeded because methodological innovation and systems engineering were advanced *together*, and because each constrained and enabled the other.

Methods constrain systems. The kinds of policies a method can learn determine what infrastructure must exist around it. A multi-agent communication policy requires an environment that supports reactive adversaries. A visual-prompt locomotion policy requires

a simulator with the throughput to amortize the data appetite of large pre-trained backbones. A touch-aware vision–language–action model requires a high-fidelity tactile sensor model that does not yet exist in any off-the-shelf simulator. Each method demands a particular infrastructure to be realizable at all.

Systems constrain methods. The infrastructure available to a researcher quietly determines which methods are tractable. GPU-parallel simulation made vision-based locomotion training go from weeks to hours. A fleet-scale deployment platform made it practical to study CI/CD as a research object. An end-to-end data engine made it practical to scale tactile pre-training to a working world model. Without these systems advances, the corresponding methodological work could not have been done.

The two together. The reason for an end-to-end approach is not aesthetic. Each advance in this dissertation is best understood not in isolation but as the result of a method and a system being co-designed: an algorithm whose data appetite the infrastructure can satisfy, paired with infrastructure whose effort the algorithm can repay.

5.3. Limitations

Despite the gains reported across the four works, important limitations remain.

Sim-to-real gap persists. Even high-fidelity twins miss long-tail physics: rare collisions, unusual materials, novel sensor noise. Robust transfer remains the dominant practical obstacle for embodied AI.

Data cost of contact. Tactile and contact-rich data are still expensive despite the hybrid sim/real strategy in Chapter 4. Without further progress in tactile simulation

fidelity or self-supervised tactile learning, the cost of scaling DreamTacVLA-style methods to new embodiments and tasks will remain high.

Generalization. The policies in this thesis remain task- and embodiment-specific. Broad transfer—across robots, tasks, and sensing modalities—is an open problem.

System complexity. End-to-end pipelines are powerful but heavy. The system in Chapter 4 reflects years of engineering investment, and lowering the entry cost for similar pipelines is itself a research direction.

5.4. Future Work

I see four directions in which the end-to-end thesis can be pushed further.

Unified world models across vision, language, and touch. DreamTacVLA shows that predictive tactile reasoning is a powerful substrate for control. The natural extension is a single predictive model spanning vision, language, and touch, so that a robot can imagine the multimodal consequences of any planned action.

Lifelong cloud learning. The DOS® vision of continuous integration extended to lifelong learning: fleets that collect, retrain, and redeploy automatically as the world drifts. The infrastructure exists; the missing pieces are statistical—safe exploration, distributional drift detection, and online evaluation at fleet scale.

Cross-embodiment transfer. Skills that move between quadrupeds, arms, mobile manipulators, and humanoids. Cross-embodiment transfer would amortize the cost of every methodological advance across many hardware platforms, which is the only way the field's data appetite can be sustained.

Foundation tactile models. Pre-trained touch representations as ubiquitous as vision encoders. The lack of a "CLIP for touch" is the largest single blocker for tactile VLAs to proliferate. The hybrid dataset and frozen tactile backbone of Chapter 4 are a step in this direction; the next step is to scale them to a community-shared foundation model.

5.5. Closing Remarks

The thesis I have argued for is, in the end, simple. *Embodied AI is an end-to-end discipline.* The next generation of embodied AI researchers will not be specialists who hand off problems across walls between algorithm and infrastructure. They will be end-to-end roboticists who can co-design the method and the system together, because each only makes sense in the company of the other. The four works in this dissertation are my best attempt to demonstrate that such a discipline is possible. The work that comes after this thesis will, I hope, take that demonstration much further.

References

[1] Farshad Ahmadighohandizi and Kari Systä. Application development and deployment for iot devices. In *European Conference on Service-Oriented and Cloud Computing*, pages 74–85. Springer, 2016.

[2] Raghav Anand, Jeffrey Ichnowski, Chenggang Wu, Joseph M Hellerstein, Joseph E Gonzalez, and Ken Goldberg. Serverless multi-query motion planning for fog robotics. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7457–7463. IEEE, 2021.

[3] Rajesh Arumugam, Vikas Reddy Enti, Liu Bingbing, Wu Xiaojun, Krishnamoorthy Baskaran, Foong Foo Kong, A Senthil Kumar, Kang Dee Meng, and Goh Wai Kit. Davinci: A cloud computing framework for service robots. In *2010 IEEE international conference on robotics and automation*, pages 3084–3089. IEEE, 2010.

[4] Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

[5] Anirban Bhattacharjee, Yogesh Barve, Shweta Khare, Shunxing Bao, Zhuangwei Kang, Aniruddha Gokhale, and Thomas Damiano. Stratum: A bigdata-as-a-service for lifecycle management of iot analytics applications. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 1607–1612. IEEE, 2019.

[6] Raunaq Bhirangi, Venkatesh Pattabiraman, Enes Erciyes, Yifeng Cao, Tess Hellebrekers, and Lerrel Pinto. Anyskin: Plug-and-play skin sensing for robotic touch. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 16563–16570. IEEE, 2025.

[7] Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

[8] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[9] Radhia Bouziane, Labib Sadek Terrissa, Soheyb Ayad, Jean-Franois Brethe, and Okba Kazar. A web services based solution for the nao robot in cloud robotics environment. In *2017 4th International Conference on Control, Decision and Information Technologies (CoDIT)*, pages 0809–0814. IEEE, 2017.

[10] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023. *URL https://arxiv. org/abs/2307.15818*, 2024.

[11] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[12] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-1: Robotics transformer for real-world control at scale, 2023.

[13] Angel Cañete, Mercedes Amor, and Lidia Fuentes. Energy-efficient deployment of iot applications in edge-based infrastructures: A software product line approach. *IEEE Internet of Things Journal*, 8(22):16427–16439, 2020.

[14] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *CoRR*, abs/2104.14294, 2021.

[15] Jun Cen, Chaohui Yu, Hangjie Yuan, Yuming Jiang, Siteng Huang, Jiayan Guo, Xin Li, Yibing Song, Hao Luo, Fan Wang, et al. Worldvla: Towards autoregressive action world model. *arXiv preprint arXiv:2506.21539*, 2025.

[16] Kaiyuan Eric Chen, Yafei Liang, Nikhil Jha, Jeffrey Ichnowski, Michael Danielczuk, Joseph Gonzalez, John Kubiatowicz, and Ken Goldberg. Fogros: An adaptive framework for automating fog robotics deployment. In *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, pages 2035–2042. IEEE, 2021.

[17] Long Chen, Oleg Sinavski, Jan Hünermann, Alice Karnsund, Andrew James Willmott, Danny Birch, Daniel Maund, and Jamie Shotton. Driving with llms: Fusing object-level vector modality for explainable autonomous driving, 2023.

[18] Zhen Cheng, Yuxiang Zhang, Weizhang Zhang, Haolin Li, Kangbo Wang, Li Song, and Hong Zhang. Omnivtla: Vision-tactile-language-action model with semantic-aligned tactile sensing. *arXiv preprint arXiv:2508.08706*, 2025.

[19] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

[20] Suyoung Choi, Gwanghyeon Ji, Jeongsoo Park, Hyeongjun Kim, Juhyeok Mun, Jeong Hyun Lee, and Jemin Hwangbo. Learning quadrupedal locomotion on deformable terrain. *Science Robotics*, 8(74):eade2256, 2023.

[21] Filippos Christianos, Georgios Papoudakis, Arrasy Rahman, and Stefano V Albrecht. Scaling multi-agent reinforcement learning with selective parameter sharing. *arXiv preprint arXiv:2102.07475*, 2021.

[22] Stampfer Dennis, Lotz Alex, Lutz Matthias, and Schlegel Christian. The smartmdsd toolchain: An integrated mdsd workflow and integrated development environment (ide) for robotics software. 2016.

[23] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[24] Michael Everett, Yu Fan Chen, and Jonathan P How. Motion planning among dynamic, decision-making agents with deep reinforcement learning. In *2018 IEEE/RSJ*

International Conference on Intelligent Robots and Systems (IROS), pages 3052–3059. IEEE, 2018.

[25] Gilbert Feng, Hongbo Zhang, Zhongyu Li, Xue Bin Peng, Bhuvan Basireddy, Linzhu Yue, ZHITAO SONG, Lizhi Yang, Yunhui Liu, Koushil Sreenath, and Sergey Levine. Genloco: Generalized locomotion controllers for quadrupedal robots. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 1893–1903. PMLR, 14–18 Dec 2023.

[26] Letian Fu, Gaurav Datta, Huang Huang, William Chung-Ho Panitch, Jaimyn Drake, Joseph Ortiz, Mustafa Mukadam, Mike Lambeta, Roberto Calandra, and Ken Goldberg. A touch, vision, and language dataset for multimodal alignment. *arXiv preprint arXiv:2402.13232*, 2024.

[27] Zipeng Fu, Ashish Kumar, Ananye Agarwal, Haozhi Qi, Jitendra Malik, and Deepak Pathak. Coupling vision and proprioception for navigation of legged robots. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17273–17283, 2022.

[28] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2(3), 2018.

[29] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

[30] Sami Salama Hussen Hajjaj and Khairul Saleh Mohamed Sahari. Establishing remote networks for ros applications via port forwarding: A detailed tutorial. *International Journal of Advanced Robotic Systems*, 14(3):1729881417703355, 2017.

[31] Kaveh Akbari Hamed, Jeeseop Kim, and Abhishek Pandala. Quadrupedal locomotion via event-based predictive control and qp-based virtual constraints. *IEEE Robotics and Automation Letters*, 5(3):4463–4470, 2020.

[32] He He, Jordan Boyd-Graber, Kevin Kwok, and Hal Daumé III. Opponent modeling in deep reinforcement learning. In *International conference on machine learning*, pages 1804–1813. PMLR, 2016.

[33] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[34] Tairan He, Chong Zhang, Wenli Xiao, Guanqi He, Changliu Liu, and Guanya Shi. Agile but safe: Learning collision-free high-speed legged locomotion. *arXiv preprint arXiv:2401.17583*, 2024.

[35] Liang Heng, Haoran Geng, Kaifeng Zhang, Pieter Abbeel, and Jitendra Malik. Vitacformer: Learning cross-modal representation for visuo-tactile dexterous manipulation. *arXiv preprint arXiv:2506.15953*, 2025.

[36] Carolina Higuera, Akash Sharma, Chaithanya Krishna Bodduluri, Taosha Fan, Patrick Lancaster, Mrinal Kalakrishnan, Michael Kaess, Byron Boots, Mike Lambeta, Tingfan Wu, et al. Sparsh: Self-supervised touch representations for vision-based tactile sensing. *arXiv preprint arXiv:2410.24090*, 2024.

[37] Yedid Hoshen. Vain: Attentional multi-agent predictive modeling. *arXiv preprint arXiv:1706.06122*, 2017.

[38] Gaoping Huang, Pawan S Rao, Meng-Han Wu, Xun Qian, Shimon Y Nof, Karthik Ramani, and Alexander J Quinn. Vipo: Spatial-visual programming with functions for robot-iot workflows. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2020.

[39] Jialei Huang, Siyuan Wang, Fulin Lin, Yuxiang Hu, Chen Wen, and Yizhou Gao. Tactile-vla: Unlocking vision-language-action model's physical knowledge for tactile generalization. *arXiv preprint arXiv:2507.09160*, 2025.

[40] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[41] Yongbin Jin, Xianwei Liu, Yecheng Shao, Hongtao Wang, and Wei Yang. High-speed quadrupedal locomotion by imitation-relaxation reinforcement learning. *Nature Machine Intelligence*, 4(12):1198–1208, Dec 2022.

[42] Joshua Jones, Oier Mees, Carmelo Sferrazza, Kyle Stachowicz, Pieter Abbeel, and Sergey Levine. Beyond sight: Finetuning generalist robot policies with heterogeneous sensors via language grounding. *arXiv preprint arXiv:2501.04693*, 2025.

[43] Shinpei Kato, Shota Tokunaga, Yuya Maruyama, Seiya Maeda, Manato Hirabayashi, Yuki Kitsukawa, Abraham Monrroy, Tomohito Ando, Yusuke Fujii, and Takuya Azumi. Autoware on board: Enabling autonomous vehicles with embedded systems. In *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, pages 287–296. IEEE, 2018.

[44] Konstantinos Katsiapis and Kevin Haas. Towards ml engineering with tensorflow extended (tfx). In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3182–3182, 2019.

[45] Ben Kehoe, Sachin Patil, Pieter Abbeel, and Ken Goldberg. A survey of research on cloud robotics and automation. *IEEE Transactions on automation science and engineering*, 12(2):398–409, 2015.

[46] Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

[47] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[48] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model, 2024.

[49] Yunho Kim, Chanyoung Kim, and Jemin Hwangbo. Learning forward dynamics model and informed trajectory sampler for safe quadruped navigation, 2022.

[50] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2149–2154. IEEE, 2004.

[51] Shihan Kong, Jinlin Sun, Aocheng Luo, Wanchao Chi, Chong Zhang, Shenghao Zhang, Yuzhen Liu, Qiuguo Zhu, and Junzhi Yu. A collision-free target tracking controller with uncertain disturbance rejection for quadruped robots. *IEEE Transactions on Intelligent Vehicles*, 9(1):670–680, 2024.

[52] Anurag Koul. ma-gym: Collection of multi-agent environments based on openai gym. `https://github.com/koulanurag/ma-gym`, 2019.

[53] Tim Kraska. Northstar: An interactive data science system. 2021.

[54] Sanjay Krishnan, Jiannan Wang, Eugene Wu, Michael J Franklin, and Ken Goldberg. Activeclean: Interactive data cleaning for statistical modeling. *Proceedings of the VLDB Endowment*, 9(12):948–959, 2016.

[55] Mike Lambeta, Po-Wei Chou, Stephen Tian, Brian Yang, Benjamin Maloon, Victoria Rose Most, Dave Stroud, Raymond Santos, Ahmad Byagowi, Gregg Kammerer, et al. Digit: A novel design for a low-cost compact high-resolution tactile sensor with application to in-hand manipulation. *IEEE Robotics and Automation Letters*, 5(3):3838–3845, 2020.

[56] P. Langley. Crafting papers on machine learning. In Pat Langley, editor, *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pages 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

[57] Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. Multi-agent cooperation and the emergence of (natural) language. *arXiv preprint arXiv:1612.07182*, 2016.

[58] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5(47):eabc5986, 2020.

[59] Yin Lei, Zhou Fengyu, Wang Yugang, Yuan Xianfeng, Zhao Yang, and Chen Zhumin. Design of a cloud robotics visual platform. In *2016 Sixth International Conference on Instrumentation & Measurement, Computer, Communication and Control (IMCCC)*, pages 1039–1043. IEEE, 2016.

[60] Chengmeng Li, Junjie Wen, Yan Peng, Yaxin Peng, Feifei Feng, and Yichen Zhu. Pointvla: Injecting the 3d world into vision-language-action models. *arXiv preprint arXiv:2503.07511*, 2025.

[61] Hao Li, Yizhi Zhang, Junzhe Zhu, Shaoxiong Wang, Michelle A Lee, Huazhe Xu, Edward Adelson, Li Fei-Fei, Ruohan Gao, and Jiajun Wu. See, hear, and feel: Smart sensory fusion for robotic manipulation. *arXiv preprint arXiv:2212.03858*, 2022.

[62] Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, et al. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650*, 2024.

[63] Xun Li and Risto Miikkulainen. Dynamic adaptation and opponent exploitation in computer poker. In *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[64] Jia Zhi Lim and Danny Wee-Kiat Ng. Cloud based implementation of ros through vpn. In *2019 7th International Conference on Smart Computing & Communications (ICSCC)*, pages 1–5. IEEE, 2019.

[65] Jiaying Lin and Rynson W.H. Lau. Self-supervised pre-training for mirror detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12227–12236, October 2023.

[66] Qinjie Lin, Guo Ye, Jiayi Wang, and Han Liu. Roboflow: a data-centric workflow management system for developing ai-enhanced robots. In *Conference on Robot Learning*, pages 1789–1794. PMLR, 2022.

[67] Tao Lin, Gen Li, Yilei Zhong, Yanwen Zou, Yuxin Du, Jiting Liu, Encheng Gu, and Bo Zhao. Evo-0: Vision-language-action model with implicit spatial understanding. *arXiv preprint arXiv:2507.00416*, 2025.

[68] Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163. Elsevier, 1994.

[69] Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. *arXiv preprint arXiv:2410.07864*, 2024.

[70] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al. Swin transformer v2: Scaling up capacity and resolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12009–12019, 2022.

[71] Zhuoyang Liu, Jiaming Liu, Jiadong Xu, Nuowei Han, Chenyang Gu, Hao Chen, Kaichen Zhou, Renrui Zhang, Kai Chin Hsieh, Kun Wu, et al. Mla: A multisensory language-action model for multimodal understanding and forecasting in robotic manipulation. *arXiv preprint arXiv:2509.26642*, 2025.

[72] Pinxin Long, Tingxiang Fanl, Xinyi Liao, Wenxi Liu, Hao Zhang, and Jia Pan. Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6252–6259. IEEE, 2018.

[73] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *arXiv preprint arXiv:1706.02275*, 2017.

[74] Zhao Mandi, Shreeya Jain, and Shuran Song. Roco: Dialectic multi-robot collaboration with large language models, 2023.

[75] Gabriel B Margolis and Pulkit Agrawal. Walk these ways: Tuning robot control for generalization with multiplicity of behavior. *Conference on Robot Learning*, 2022.

[76] Takahiro Miki, Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science Robotics*, 7(62):eabk2822, 2022.

[77] Maria Vittoria Minniti, Ruben Grandia, Farbod Farshidian, and Marco Hutter. Adaptive clf-mpc with application to quadrupedal robots. *IEEE Robotics and Automation Letters*, 7(1):565–572, 2022.

[78] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[79] Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[80] Adithyavairavan Murali, Tao Chen, Kalyan Vasudev Alwala, Dhiraj Gandhi, Lerrel Pinto, Saurabh Gupta, and Abhinav Gupta. Pyrobot: An open-source robotics framework for research and benchmarking. *arXiv preprint arXiv:1906.08236*, 2019.

[81] Sriniketan Mysari and Vaibhav Bejgam. Continuous integration and continuous deployment pipeline automation using jenkins ansible. In *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*, pages 1–4. IEEE, 2020.

[82] Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022.

[83] Duc Huy Nguyen, Tim Schneider, Guillaume Duret, Alap Kshirsagar, Boris Belousov, and Jan Peters. Tacex: Gelsight tactile simulation in isaac sim–combining soft-body and visuotactile simulators. *arXiv preprint arXiv:2411.04776*, 2024.

[84] NVIDIA. Isaac Sim.

[85] Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE, 2024.

[86] Georgios Papoudakis and Stefano V Albrecht. Variational autoencoders for opponent modeling in multi-agent systems. *arXiv preprint arXiv:2001.10829*, 2020.

[87] Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V. Albrecht. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS)*, 2021.

[88] Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, et al. Spatialvla: Exploring spatial representations for visual-language-action model. *arXiv preprint arXiv:2501.15830*, 2025.

[89] Neil Rabinowitz, Frank Perbet, Francis Song, Chiyuan Zhang, SM Ali Eslami, and Matthew Botvinick. Machine theory of mind. In *International conference on machine learning*, pages 4218–4227. PMLR, 2018.

[90] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[91] Ilija Radosavovic, Tete Xiao, Stephen James, Pieter Abbeel, Jitendra Malik, and Trevor Darrell. Real-world robot learning with masked visual pre-training. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 416–426. PMLR, 14–18 Dec 2023.

[92] Roberta Raileanu, Emily Denton, Arthur Szlam, and Rob Fergus. Modeling others using oneself in multi-agent reinforcement learning. In *International conference on machine learning*, pages 4257–4266. PMLR, 2018.

[93] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 4295–4304. PMLR, 2018.

[94] Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, volume 11, page 269. NIH Public Access, 2017.

[95] Pilar Rodríguez, Alireza Haghighatkhah, Lucy Ellen Lwakatare, Susanna Teppola, Tanja Suomalainen, Juho Eskeli, Teemu Karvonen, Pasi Kuvaja, June M Verner, and Markku Oivo. Continuous deployment of software intensive products and services: A systematic mapping study. *Journal of Systems and Software*, 123:263–291, 2017.

[96] Olimpiya Saha and Prithviraj Dasgupta. A comprehensive survey of recent trends in cloud robotics architectures and applications. *Robotics*, 7(3):47, 2018.

[97] Sebastian Schelter, Stefan Grafberger, Philipp Schmidt, Tammo Rukat, Mario Kiessling, Andrey Taptunov, Felix Biessmann, and Dustin Lange. Differential data quality verification on partitioned data. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 1940–1945. IEEE, 2019.

[98] Sebastian Schelter, Dustin Lange, Philipp Schmidt, Meltem Celikel, Felix Biessmann, and Andreas Grafberger. Automating large-scale data quality verification. *Proceedings of the VLDB Endowment*, 11(12):1781–1794, 2018.

[99] Mingyo Seo, Ryan Gupta, Yifeng Zhu, Alexy Skoutnev, Luis Sentis, and Yuke Zhu. Learning to walk by steering: Perceptive quadrupedal locomotion in dynamic environments. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5099–5105. IEEE, 2023.

[100] Dhruv Shah, Błażej Osiński, brian ichter, and Sergey Levine. Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 492–504. PMLR, 14–18 Dec 2023.

[101] Yecheng Shao, Yongbin Jin, Xianwei Liu, Weiyan He, Hongtao Wang, and Wei Yang. Learning free gait transition for quadruped robots via phase-guided controller. *IEEE Robotics and Automation Letters*, 7(2):1230–1237, 2022.

[102] Lloyd S Shapley. Stochastic games. *Proceedings of the national academy of sciences*, 39(10):1095–1100, 1953.

[103] Z. She, S. Wang, G. Lee, and H. Su. Tactile-based policy learning for contact-rich tasks. *Robotics and Autonomous Systems*, 2023.

[104] Zilin Si and Wenzhen Yuan. Taxim: An example-based simulation model for gelsight tactile sensors. *IEEE Robotics and Automation Letters*, 7(2):2361–2368, 2022.

[105] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[106] Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. Learning when to communicate at scale in multiagent cooperative and competitive tasks. *arXiv preprint arXiv:1812.09755*, 2018.

[107] Daniel Sokolowski, Pascal Weisenburger, and Guido Salvaneschi. Automating serverless deployments for devops organizations. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 57–69, 2021.

[108] Austin Stone, Ted Xiao, Yao Lu, Keerthana Gopalakrishnan, Kuang-Huei Lee, Quan Vuong, Paul Wohlhart, Sean Kirmani, Brianna Zitkovich, Fei Xia, Chelsea Finn, and Karol Hausman. Open-world object manipulation using pre-trained vision-language models, 2023.

[109] Sainbayar Sukhbaatar, Rob Fergus, et al. Learning multiagent communication with backpropagation. *Advances in neural information processing systems*, 29:2244–2252, 2016.

[110] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

[111] Ajay Kumar Tanwani, Nitesh Mor, John Kubiatowicz, Joseph E Gonzalez, and Ken Goldberg. A fog robotics approach to deep robot learning: Application to object recognition and grasp planning in surface decluttering. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 4559–4566. IEEE, 2019.

[112] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

[113] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy, 2024.

[114] Sérgio Teixeira, Rafael Arrais, and Germano Veiga. Cloud simulation for continuous integration and deployment in robotics. In *2021 IEEE 19th International Conference on Industrial Informatics (INDIN)*, pages 1–8. IEEE, 2021.

[115] Xiaoyu Tian, Junru Gu, Bailin Li, Yicheng Liu, Yang Wang, Zhiyong Zhao, Kun Zhan, Peng Jia, Xianpeng Lang, and Hang Zhao. Drivevlm: The convergence of autonomous driving and large vision-language models, 2024.

[116] Zheng Tian, Ying Wen, Zhichen Gong, Faiz Punakkath, Shihao Zou, and Jun Wang. A regularized opponent model with maximum entropy objective. *arXiv preprint arXiv:1905.08087*, 2019.

[117] Zheng Tian, Shihao Zou, Ian Davies, Tim Warr, Lisheng Wu, Haitham Bou Ammar, and Jun Wang. Learning to communicate implicitly by actions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7261–7268, 2020.

[118] Jeremy Wallace. Deploy and Manage ROS Robots with AWS IoT Greengrass 2.0 and Docker. `https://aws.amazon.com/blogs/robotics/deploy-and-manage-ros-robots-with-aws-iot-greengrass-2-0-and-docker/`, 2021.

[119] Ying Wen, Yaodong Yang, Rui Luo, Jun Wang, and Wei Pan. Probabilistic recursive reasoning for multi-agent reinforcement learning. *arXiv preprint arXiv:1901.09207*, 2019.

[120] Huaming Wu, Xiangyi Li, and Yingjun Deng. Deep learning-driven wireless communication for edge-cloud computing: opportunities and challenges. *Journal of Cloud Computing*, 9:1–14, 2020.

[121] Zhiyuan Wu, Yijiong Lin, Yongqiang Zhao, Xuyang Zhang, Zhuo Chen, Nathan Lepora, and Shan Luo. Vitacgen: Robotic pushing with vision-to-touch generation. *IEEE Robotics and Automation Letters*, 2025.

[122] Shihao Xu, Zhenjiang Zhang, Michel Kadoch, and Mohamed Cheriet. A collaborative cloud-edge computing framework in distributed neural network. *EURASIP Journal on Wireless Communications and Networking*, 2020(1):1–17, 2020.

[123] Zhenhua Xu, Yujia Zhang, Enze Xie, Zhen Zhao, Yong Guo, Kwan-Yee. K. Wong, Zhenguo Li, and Hengshuang Zhao. Drivegpt4: Interpretable end-to-end autonomous driving via large language model, 2024.

[124] Han Xue, Jieji Ren, Wendi Chen, Gu Zhang, Yuan Fang, Guoying Gu, Huazhe Xu, and Cewu Lu. Reactive diffusion policy: Slow-fast visual-tactile policy learning for contact-rich manipulation. *arXiv preprint arXiv:2503.02881*, 2025.

[125] Chenyu Yang, Guo Ning Sue, Zhongyu Li, Lizhi Yang, Haotian Shen, Yufeng Chi, Akshara Rai, Jun Zeng, and Koushil Sreenath. Collaborative navigation and manipulation of a cable-towed load by multiple quadrupedal robots. *IEEE Robotics and Automation Letters*, 7(4):10041–10048, 2022.

[126] Fengyu Yang, Chao Feng, Ziyang Chen, Hyoungseob Park, Daniel Wang, Yiming Dou, Ziyao Zeng, Xien Chen, Rit Gangopadhyay, Andrew Owens, et al. Binding touch to everything: Learning unified multimodal tactile representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26340–26353, 2024.

[127] Ruihan Yang, Minghao Zhang, Nicklas Hansen, Huazhe Xu, and Xiaolong Wang. Learning vision-guided quadrupedal locomotion end-to-end with cross-modal transformers. *arXiv preprint arXiv:2107.03996*, 2021.

[128] Ruihan Yang, Minghao Zhang, Nicklas Hansen, Huazhe Xu, and Xiaolong Wang. Learning vision-guided quadrupedal locomotion end-to-end with cross-modal transformers. In *International Conference on Learning Representations*, 2022.

[129] Guo Ye, Qinjie Lin, Tzung-Han Juang, and Han Liu. Collision-free navigation of human-centered robots via Markov games. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[130] Guo Ye, Qinjie Lin, Tzung-Han Juang, and Han Liu. Collision-free navigation of human-centered robots via markov games. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11338–11344. IEEE, 2020.

[131] Wenzhen Yuan, Siyuan Dong, and Edward H. Adelson. Gelsight: High-resolution tactile sensing for geom- etry and force. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[132] Chuan Zhang, Mingyang Zhao, Yuhua Xu, Tong Wu, Yanwei Li, Liehuang Zhu, and Haotian Wang. Achieving fuzzy matching data sharing for secure cloud-edge communication. *China Communications*, 19(7):257–276, 2022.

[133] Kaiqing Zhang, Tao Sun, Yunzhe Tao, Sahika Genc, Sunil Mallya, and Tamer Basar. Robust multi-agent reinforcement learning with model uncertainty. In *NeurIPS*, 2020.

[134] Wenyao Zhang, Hongsi Liu, Zekun Qi, Yunnan Wang, Xinqiang Yu, Jiazhao Zhang, Runpei Dong, Jiawei He, Fan Lu, He Wang, et al. Dreamvla: a vision-language-action model dreamed with comprehensive world knowledge. *arXiv preprint arXiv:2507.04447*, 2025.

[135] Yang Zhang, Bogdan Vasilescu, Huaimin Wang, and Vladimir Filkov. One size does not fit all: an empirical study of containerized continuous deployment workflows. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 295–306, 2018.

[136] Zhen Zhang, Jiaqing Yan, Xin Kong, Guangyao Zhai, and Yong Liu. Efficient motion planning based on kinodynamic model for quadruped robots following persons in confined spaces. *IEEE/ASME Transactions on Mechatronics*, 26(4):1997–2006, 2021.

[137] Jialiang Zhao, Yuxiang Ma, Lirui Wang, and Edward H Adelson. Transferable tactile transformers for representation learning across diverse sensors and tasks. *arXiv preprint arXiv:2406.13640*, 2024.

[138] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[139] Xiaoqi Zhao, Youwei Pang, Lihe Zhang, , Huchuan Lu, and Xiang Ruan. Self-supervised pretraining for rgb-d salient object detection. In *AAAI*, 2022.

[140] Ruijie Zheng, Yongyuan Liang, Shuaiyi Huang, Jianfeng Gao, Hal Daumé III, Andrey Kolobov, Furong Huang, and Jianwei Yang. Tracevla: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies. *arXiv preprint arXiv:2412.10345*, 2024.

[141] Wujie Zhou, Fan Sun, Qiuping Jiang, Runmin Cong, and Jenq-Neng Hwang. Wavenet: Wavelet network with knowledge distillation for rgb-t salient object detection. *IEEE Transactions on Image Processing*, 32:3027–3039, 2023.

[142] Luisa Zintgraf, Sam Devlin, Kamil Ciosek, Shimon Whiteson, and Katja Hofmann. Deep interactive bayesian reinforcement learning via meta-learning. *arXiv preprint arXiv:2101.03864*, 2021.

APPENDIX A

Supplementary Materials for Chapter 4

This appendix collects the supplementary materials accompanying Chapter 4. It covers the large-scale simulation pipeline and data collection in IsaacSim, implementation details for the model architecture and training, formal details of Hierarchical Spatial Alignment, the tactile world model pre-training and forecasting loss, evaluation procedures, and an extended related-work discussion.

Appendix Catalogue

This appendix provides additional implementation details, experimental results, and theoretical background for DreamTacVLA. Below is a summary of the contents:

Appendix A: Simulation Pipeline and Large-Scale Data Collection 123
Appendix B: Implementation Details 128
Appendix C: Hierarchical Spatial Alignment (HSA) 132
Appendix D: Tactile World Model Pretraining and Forecasting 134
Appendix E: Evaluation Protocol 136
Appendix F: Extended Related Work 138

A.1. Simulation Pipeline and Large-Scale Data Collection

This section clarifies the role of simulation in DreamTacVLA and explains data quality, expert reliability, and sim-to-real validity. Simulation is used not as a proxy for deployment, but as a scalable and stable substrate for learning spatially grounded tactile representations and tactile future prediction.

A.1.1. IsaacSim Environment

All simulation experiments are conducted in NVIDIA IsaacSim with GPU-accelerated PhysX, using conservative physics settings to ensure stable long-horizon contact under tight tolerances. The simulated robot is a digital twin of the Dobot XTrainer with shared camera extrinsics between simulation and real-world setups. Following the main submission, we employ a high-fidelity visuotactile simulator based on TacEx and Taxim-style

optical and marker-texture warping to synthesize GelSight-like tactile observations, enabling large-scale data collection under randomized object poses in parallel simulation environments.

Table A.1. Simulation configuration (IsaacSim).

Component	**Setting**
Physics timestep	1/60 s
Solver substeps	2
Control frequency	30 Hz
Max episode length	300 steps
Parallel environments	1024 / GPU(24G)
Sensors	TPV RGB, Wrist RGB, Dual Tactile

A.1.2. Automated Expert Demonstrations (cuRobo)

Expert demonstrations are not collected via human teleoperation. Instead, we employ an automated, privileged expert based on cuRobo, which has access to ground-truth object pose, collision geometry, and task-specific termination conditions. The expert operates in task space and generates collision-aware trajectories that satisfy geometric alignment and insertion constraints. These trajectories are time-parameterized (minimum-jerk) and resampled at the control frequency (30 Hz) to produce per-step 6D end-effector Δ-pose + gripper supervision, consistent with the action representation used by the policy, resulting in low-noise and consistent demonstrations for contact-rich manipulation.

This design guarantees demonstration quality: stable contact behavior at sub-millimeter precision would be extremely difficult to achieve via human teleoperation without force or tactile feedback.

Table A.2. cuRobo expert trajectory generation.

Aspect	**Specification**
Planner	cuRobo (batched)
Batch size	256 queries
Planning horizon	32 waypoints (minimum-jerk)
Collision checking	Robot + objects + table
IK tolerance	2 mm position / 2° rotation
Action format	6D EE delta pose + gripper

Table A.3. Task success rate (%) in IsaacSim (100 trials per task).

Model	**Peg-in-Hole**	**USB Insert**	**Gear Assembly**	**Tool Stabilization**
ACT	72.4	78.1	65.3	61.7
Diffusion Policy	75.6	80.2	69.4	66.1
π_0	83.5	85.9	77.2	74.6
Ours (HSA-Only, No Dream)	89.1	90.4	84.6	82.7
Ours (No HSA, Dream-Only)	93.8	94.6	88.9	90.1
Ours (HSA & Dream)	**98.6**	**97.9**	**95.2**	**94.7**

A.1.3. Success-Based Filtering and Synchronization

Only successful rollouts are retained as demonstrations, based on task-specific geometric and temporal criteria (e.g., insertion depth, alignment error, sustained stability). All modalities are logged with strict step-level synchronization; if any modality is missing at a timestep, the entire episode is discarded. This filtering is applied only for constructing expert supervision for behavior cloning; evaluation is performed on fresh randomized initializations.

A.1.4. Performance within IsaacSim

To assess the effectiveness of our method under controlled conditions, we report task success rates measured inside IsaacSim using the same evaluation protocol as in real-world experiments as Table A.3 shows. These simulation-side results provide additional context

that the relative performance trends and ablation ordering observed on real hardware are already consistent in IsaacSim. This consistency suggests that the benefits of HSA and Dreaming are not solely driven by real-world noise or dataset imbalance.

Algorithm. *DreamTacVLA: Two-Stage Training with Think–Dream–Act (Implementation-Matched)*

Require: Dataset $\mathcal{D}_1$ consists of tuples $(L, s^{(t)}, I_w^{(t)}, I_{tp}^{(t)}, I_\tau^{(t)}, a_t, B^{(t)})$

Require: Dataset $\mathcal{D}_2$ consists of tuples $(L, s^{(t)}, I_w^{(t)}, I_{tp}^{(t)}, I_\tau^{(t)}, a_t, I_\tau^{(t+N)}, B^{(t)})$

Require: Modules: multimodal encoders E_ψ, policy π_θ, tactile world encoder W_ϕ (pretrained, frozen), forecasting MLP F_η

Require: Hyperparams: $S_1, S_2, \lambda_{HSA}, \lambda_W, \lambda_{action}$; null-dream token H_{null}

Initialize parameters of E_ψ and π_θ

Set H_{null} (zeros or learned)

Stage 1 (no dreaming): pre-train encoders + policy with Action + HSA

for $s \leftarrow 1$ to S_1 do

  Sample $(L, s^{(t)}, I_w^{(t)}, I_{tp}^{(t)}, I_\tau^{(t)}, a_t, B^{(t)})$ from $\mathcal{D}_1$

  $H_{align}^{(t)} \leftarrow E_\psi(L, I_w^{(t)}, I_{tp}^{(t)}, I_\tau^{(t)}, s^{(t)})$

  $L_{HSA} \leftarrow \textsc{HSAInfoNCE}(H_{\texttt{align}}^{(t)}, B^{(t)})$

  $\hat{a}_t \leftarrow \pi_\theta(H_{\texttt{align}}^{(t)}, H_{\texttt{null}})$

  $L_{action} \leftarrow \textsc{LDiffusion}(\hat{a}_t, a_t)$

  $L \leftarrow L_{action} + \lambda_{HSA} L_{HSA}$

  Backpropagate L and update ψ, θ

end for

Stage 2: finetune with latent dreaming (Think-Dream-Act)

Load pretrained W_ϕ and freeze it (no gradient updates)

Initialize F_η (trainable)

for $s \leftarrow 1$ to S_2 do

Sample $(L, s^{(t)}, I_w^{(t)}, I_{tp}^{(t)}, I_\tau^{(t)}, a_t, I_\tau^{(t+N)}, B^{(t)})$ from $\mathcal{D}_2$

$H_{\texttt{align}}^{(t)} \leftarrow E_\psi(L, I_w^{(t)}, I_{tp}^{(t)}, I_\tau^{(t)}, s^{(t)})$

$L_{HSA} \leftarrow \textsc{HSAInfoNCE}(H_{\texttt{align}}^{(t)}, B^{(t)})$

THINK: compute draft action (stop gradient)

$a_{\texttt{draft}}^{(t)} \leftarrow \textsc{StopGrad}\Big(\pi_\theta(H_{\texttt{align}}^{(t)}, H_{\texttt{null}})\Big)$

DREAM: forecast tactile latent (no grad through W_ϕ)

$z_\tau^{(t)} \leftarrow W_\phi(I_\tau^{(t)})$

$\tilde{z}_\tau^{(t+N)} \leftarrow F_\eta(z_\tau^{(t)}, a_{\texttt{draft}}^{(t)})$

$z_\tau^{(t+N)} \leftarrow \textsc{StopGrad}\Big(W_\phi(I_\tau^{(t+N)})\Big)$

$L_W \leftarrow \left\| \tilde{z}_\tau^{(t+N)} - z_\tau^{(t+N)} \right\|_2^2$

ACT: condition policy on dreamed latent

$a_{\texttt{final}}^{(t)} \leftarrow \pi_\theta(H_{\texttt{align}}^{(t)}, \tilde{z}_\tau^{(t+N)})$

$L_{action} \leftarrow \textsc{LDiffusion}(a_{\texttt{final}}^{(t)}, a_t)$

$L \leftarrow \lambda_{action} L_{action} + \lambda_{HSA} L_{HSA} + \lambda_W L_W$

Backpropagate L and update ψ, θ, η (keep ϕ frozen; no grad through $a_{\texttt{draft}}^{(t)}$)

end for

A.2. Implementation Details

A.2.1. Model Architecture

Overview. Dream-Tac VLA uses a hybrid vision-language-action architecture that combines Action Chunking with Transformers (ACT) [**138**] as the policy backbone with V-JEPA2 [**4**] vision encoders for tactile perception. The architecture processes multi-modal sensory inputs including RGB images from workspace cameras and tactile images from vision-based tactile sensors mounted on the gripper.

Visual Encoders. For RGB camera inputs, we use frozen ResNet-18 [**33**] backbones. Each RGB camera (top view and wrist cameras) has a dedicated ResNet-18 encoder. The backbone outputs are projected to the transformer hidden dimension via a 1×1 convolution layer.

For tactile image inputs, we leverage a frozen V-JEPA2 ViT-Large [**4**] encoder with 1024-dimensional embeddings. The V-JEPA2 encoder is pretrained on large-scale video data and produces rich spatiotemporal representations. To adapt these frozen representations to our downstream manipulation tasks while preserving the pretrained knowledge, we introduce learnable residual adapters on top of the ViT encoder.

Residual Adapter Architecture. The residual adapter operates on patch-level tokens from the frozen ViT encoder. Given patch tokens $\mathbf{z} \in \mathbb{R}^{N \times D}$ where N is the number of patches and $D = 1024$ is the embedding dimension, the adapter applies:

$$\mathbf{z}' = \mathbf{z} + \alpha \cdot \mathrm{MLP}(\mathrm{LayerNorm}(\mathbf{z})) \tag{A.1}$$

where α is a learnable scaling parameter initialized to 0.1 for training stability. The MLP consists of 3 layers with hidden dimension 512, GELU activations, and dropout rate 0.1. The adapted patch tokens are then aggregated using attention pooling with a learnable query token:

$$\mathbf{h}_\tau = \text{AttentionPool}(\mathbf{z}') \in \mathbb{R}^D \tag{A.2}$$

This architecture enables task-specific adaptation of the frozen V-JEPA2 representations with only $\sim$1.5M additional trainable parameters per tactile encoder.

Transformer Architecture. The policy transformer follows the ACT architecture with a CVAE-style encoder-decoder structure:

- **CVAE Encoder**: 4-layer transformer encoder with hidden dimension 512, 8 attention heads, feed-forward dimension 3200, and dropout 0.1. The encoder produces a 32-dimensional latent code during training.
- **Action Decoder**: 7-layer transformer decoder with the same hidden dimensions. The decoder takes as input learnable action queries, proprioceptive state embedding, and the latent code.
- **Action Chunk Size**: 45 timesteps, enabling temporal action prediction for smooth execution.

Fusion Strategy. Visual features from all cameras (RGB and tactile) are concatenated along the sequence dimension before being processed by the transformer decoder. Specifically, ResNet features are flattened to $H \times W$ tokens per RGB camera, and tactile features contribute 1 pooled token per sensor. Sinusoidal positional embeddings are used for RGB features, and learnable position embeddings are used for tactile and proprioceptive tokens.

A.2.2. Training Details

We use AdamW [**33**] optimizer with the following hyperparameters as shown in Table A.4. For RGB images, we apply color jitter augmentation with brightness=0.3, contrast=0.4, saturation=0.5, and hue=0.08 when using diffusion-based policies. Tactile images are only resized to 224×224 without additional augmentation to preserve contact pattern fidelity. Actions and proprioceptive states are normalized using per-dimension mean and standard deviation computed from the training set. Images are normalized using ImageNet statistics (mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]). Training is performed on a single NVIDIA GPU. A typical training run of 20,000 steps takes approximately 8–12 hours depending on GPU type and dataset size. We formalize the full DreamTacVLA training sequence in Algorithm A.1.4.

A.2.3. Inference

During inference, the CVAE encoder is bypassed and the latent code is set to zero (mean of the prior). Actions are predicted in chunks of 45 timesteps. We use temporal aggregation with exponential weighting to smooth action execution:

$$a_t = \sum_{k=0}^{K} w_k \cdot a_t^{(k)} \tag{A.3}$$

where $a_t^{(k)}$ is the action at time t predicted k steps ago, and $w_k \propto \exp(-\lambda k)$ are exponentially decaying weights. The control loop runs at 50 Hz to match the data collection frequency.

Hyperparameter	Value
Transformer	
Hidden dimension	512
Feed-forward dimension	3200
Encoder layers	4
Decoder layers	7
Attention heads	8
Dropout	0.1
Latent dimension (CVAE)	32
Action chunk size	45
Visual Encoders	
RGB backbone	ResNet-18 (frozen)
RGB input resolution	640×480
Tactile encoder	V-JEPA2 ViT-L (frozen)
Tactile embedding dim	1024
Tactile input resolution	224×224
Residual Adapter	
Hidden dimension	512
Depth (layers)	3
Dropout	0.1
Scale init (α)	0.1
Pooling	Attention
HSA Loss	
Temperature (κ)	0.07
Feature extractor	ViT-B/16
HSA weight (β_{HSA})	1.0
Training	
Optimizer	AdamW
Learning rate	1×10^{-5}
Backbone learning rate	1×10^{-5}
Weight decay	1×10^{-4}
Batch size	16
KL weight (β_{KL})	10
Training steps	10,000–20,000

Table A.4. Complete hyperparameter configuration for Dream-Tac VLA.

Joint	θ	d (m)	a (m)	α
1	q_0	0.2234	0	$\pi/2$
2	$q_1 - \pi/2$	0	-0.280	0
3	q_2	0	-0.225	0
4	$q_3 - \pi/2$	0.1175	0	$\pi/2$
5	q_4	0.120	0	$-\pi/2$
6	q_5	0.088	0	0

Table A.5. DH parameters for Dobot Nova 2 robot.

A.3. Hierarchical Spatial Alignment (HSA)

A.3.1. Sensor and Camera Calibration

Extrinsics. We calibrate the rigid transforms from robot base to each camera frame (third-person and wrist) using a standard hand–eye calibration procedure. Let ${T_c}^b$ denote camera extrinsics in the base frame, and T_{ee}^b the end-effector pose from forward kinematics computed using Denavit-Hartenberg (DH) parameters for the Nova 2 robot as shown in Table A.5. The tactile sensor pose $T_{\text{sensor}}^b(t)$ is obtained via a fixed transform T_{sensor}^{ee} mounted on the gripper: $T_{\text{sensor}}^b(t) = T_{ee}^b T_{\text{sensor}}^{ee}$, where

$$T_c^b = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.7 \\ 0.0 & -1.0 & 0.0 & -0.49 \\ 0.0 & 0.0 & 1.0 & 1.14 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

Intrinsics. For each camera, we use intrinsic matrix K and distortion coefficients from calibration. If images are undistorted online, the projection below assumes the rectified

pinhole model. In our case, the value of K is

$$K = \begin{bmatrix} 647.0 & 0.0 & 653.0 \\ 0.0 & 644.0 & 364.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}$$

A.3.2. Projecting Tactile Sensor to Image Bounding Boxes

To localize the tactile sensor in the wrist and third-person images for HSA, we project a set of 3D points that approximate the tactile sensor's visible surface (e.g., 4 corners of a small rectangle / circle sample points) from the sensor frame into each camera: $\mathbf{u} \sim K\ {}^{c}T_{b}\ \mathbf{X}_{b}, \quad \mathbf{X}_{b} = {}^{b}T_{\text{sensor}}(t)\ \mathbf{X}_{\text{sensor}}$. We then take the min/max pixel coordinates over the projected points to form bounding boxes $B_{w}^{(t)}$ and $B_{tp}^{(t)}$. We clip boxes to image bounds and discard frames where (i) the box is fully out of view or (ii) projected depth is negative. We optionally enlarge the boxes by a small margin (e.g., 5–15 pixels) to tolerate calibration noise. If the tactile sensor is occluded in third-person view (common), we keep B_{tp} but down-weight the TPV term in L_{HSA} (see Sec. 9.4) or mask it when visibility checks fail.

A.3.3. Mapping Bounding Boxes to Patch Tokens

For CLIP ViT-L/14 inputs of resolution 224×224 and patch size 16, the patch grid is 16×16. We map bounding box pixels $[x_0, x_1] \times [y_0, y_1]$ to patch indices:

$$i \in \left[\left\lfloor \frac{x_0}{14} \right\rfloor, \left\lfloor \frac{x_1}{14} \right\rfloor\right], \quad j \in \left[\left\lfloor \frac{y_0}{14} \right\rfloor, \left\lfloor \frac{y_1}{14} \right\rfloor\right].$$

Tokens whose patch centers fall inside the box are selected and mean-pooled to form h_w and h_{tp}. The tactile embedding h_τ is mean-pooled from tactile tokens (or from the pooled tactile representation if a pooling head is used).

A.3.4. HSA Loss: Negatives, Temperature, and Weighting

We compute InfoNCE losses for tactile–wrist and tactile–TPV alignment:

$$L_{HSA} = L_{HSA\text{-}W} + L_{HSA\text{-}TP}.$$

Negatives. We use a mix of: in-image negatives: patches outside the bounding box (hard negatives), in-batch negatives: patches from other samples in the batch (diverse negatives).

Temperature. We set κ to a fixed value (0.07) and keep it constant across training.

Visibility-aware weighting. If TPV visibility is unreliable, we apply $L_{HSA} = L_{HSA\text{-}W} + \alpha(t)\, L_{HSA\text{-}TP}$, where $\alpha(t) \in [0, 1]$ is set based on projection validity / confidence.

A.4. Tactile World Model Pretraining and Forecasting Loss

To train our tactile world model, we adopt a self-supervised latent-prediction approach inspired by the **V-JEPA** (Video Joint-Embedding Predictive Architecture) framework. This approach shifts the learning objective from raw sensor reconstruction to the prediction of abstract latent representations.

A.4.1. Latent Masked Forecasting via Teacher Forcing

The pretraining phase utilizes a *teacher-student* architecture to facilitate stable feature extraction from tactile sequences. This process is defined by two primary components:

- **Teacher Encoder:** The teacher receives the complete, unmasked tactile sequence and generates target embeddings. To maintain a stable training target and prevent representation collapse, the teacher's weights are updated using an Exponential Moving Average (EMA) of the student's parameters rather than through direct backpropagation.
- **Student Predictor:** The student is provided with a temporally masked spatial version of the tactile sequence. Its objective is to predict the latent representations of the missing patches, conditioned on the available context and the specific positional encodings of the masked regions.

A.4.2. Forecasting Loss Function

The world model is optimized by minimizing the L_2 distance between the student's predicted embeddings and the teacher's ground-truth embeddings in the latent space. Critically, the loss is calculated only over the masked indices M.

The forecasting loss $\mathcal{L}$ is formulated as:

$$\mathcal{L} = \sum_{t \in M} \| z_{teacher}(t) - \hat{z}_{student}(t) \|_2^2 \tag{A.4}$$

where $z_{teacher}(t)$ represents the latent target produced by the EMA-updated teacher, and $\hat{z}_{student}(t)$ represents the student's prediction for the corresponding masked segment. By training in this latent space, the model learns to capture the underlying physics of tactile interaction—such as shear forces and surface geometry—while remaining robust to the inherent sensor noise found in raw tactile data.

A.4.3. Pretraining Data and Temporal Sampling

We pretrain the tactile world model on unlabeled tactile sequences extracted from both simulation and real demonstrations. We sample short clips of length T with stride s, and randomly choose a prediction horizon $N \in \{1, \ldots, N_{\max}\}$. This yields pairs $(I_\tau^{(t)}, I_\tau^{(t+N)})$ to encourage multi-step predictive structure.

A.5. Evaluation

To evaluate the efficacy of our framework, we conducted extensive experiments on a dual-arm manipulation platform. Our evaluation focuses on the system's ability to perform high-precision tasks where visual and tactile integration is critical.

The hardware setup consists of a Dobot X-Trainer dual-arm system. The sensing suite are two wrist-mounted cameras provide localized views of the grippers, while a single overhead camera captures the global workspace and a GelSight Digit sensor is integrated into the end-effector, providing high-resolution tactile feedback of the contact geometry.

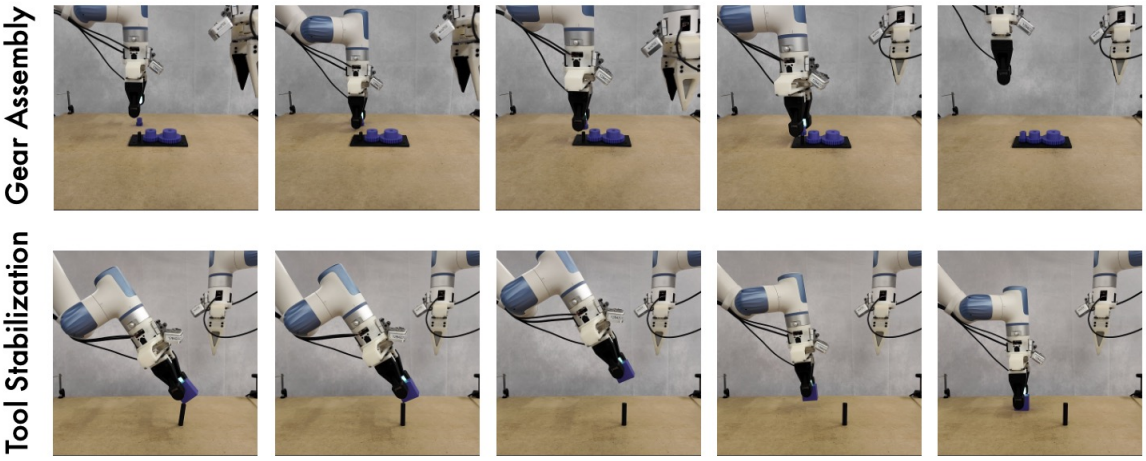


Figure A.1. Keyframes of the gear assembly and tool stabilization task workflow.

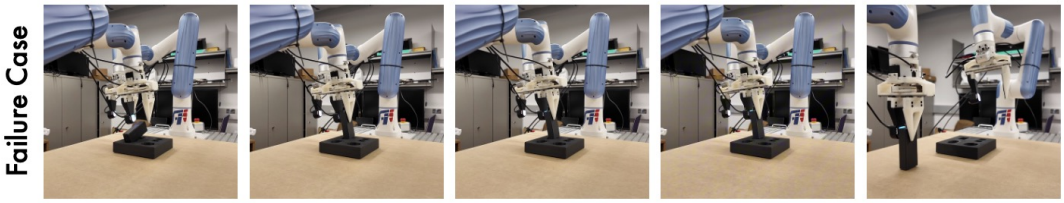


Figure A.2. Keyframes of one example failure case.

During evaluation, objects are placed at fixed initial positions. This protocol allows us to isolate and measure the model's performance on *precision-based* execution and fine-grained adjustments, effectively decoupling the control performance from variables associated with open-world localization. To ensure the statistical significance of our results and assess the reliability of the framework, we conducted over 100 trials for each task. This extensive testing allows us to accurately characterize the success rate and error distributions of the system. We show the rollout sequences of gear assembly and tool stabilization task at FigureA.1. A failure case for Peg In Hole task is also shown at Figure A.2. All the videos and source code can be found at supplementary materials.

A.6. Extended Related Work

A.6.1. Vision-Language-Action (VLA) Models

Vision-Language-Action (VLA) models unify perception, language understanding, and action generation into a single policy for general-purpose robot control. Large-scale systems such as RT-1 [**11**] and RT-2 [**10**] demonstrate that scaling data and model capacity enables strong cross-task and cross-embodiment generalization. Subsequent work extends this paradigm across embodiments, datasets, and action spaces, including Open X-Embodiment [**85**], OpenVLA [**47**], Octo [**112**], RDT-1B [**69**], π_0 [**8**], $\pi_{0.5}$ [**40**], and GR00T N1 [**7**].

Beyond scale, architectural and training refinements further improve VLA effectiveness, including diffusion-based action generation [**19**] and language-conditioned visuomotor learning for compositional generalization [**138**]. Modular designs such as CogACT [**62**] decouple high-level reasoning from low-level control, and optimized adaptation strategies like OpenVLA-OFT [**46**] show that carefully designed fine-tuning can significantly boost real-robot performance and efficiency.

Despite these advances, most VLA models remain vision-centric and struggle in contact-rich manipulation, where RGB observations alone cannot resolve contact state, friction, or incipient slip. Recent work such as FuSe [**42**] explores post-hoc adaptation with tactile and audio signals, but tactile is often treated as auxiliary rather than being integrated into perception, prediction, and action generation.

A.6.2. Spatial Grounding for Robotics

Spatial grounding improves visuomotor reasoning by incorporating geometric structure beyond RGB observations. Recent VLA variants introduce explicit 3D priors, including egocentric position encodings in SpatialVLA [**88**] and point cloud grounding in PointVLA [**60**], which improve generalization in geometry-sensitive manipulation. Complementary approaches model spatial structure implicitly: TraceVLA [**140**] encodes spatiotemporal interaction traces, while Evo-0 [**67**] injects implicit 3D structure into the visual backbone via architectural and training biases. These methods primarily reason at the level of object pose and scene layout, leaving contact-level geometry unmodeled.

A.6.3. Tactile Grounding for Manipulation

Tactile grounding provides contact information unavailable to vision, including contact geometry, friction, and slip, which is critical for contact-rich manipulation. Early approaches rely on low-dimensional force/torque sensing, which supports contact detection and impedance control but discards spatial structure, leading to ambiguous contact states.

Beyond force-based sensing, several works integrate tactile signals into learned policies. See, Hear, and Feel [**61**] studies structured multimodal fusion, while more recent VLA-style approaches incorporate tactile inputs directly. MLA [**71**], Tactile-VLA [**39**], OmniVTLA [**18**], and RDP [**124**] demonstrate improved robustness via vision–tactile fusion and reactive tactile feedback, but typically operate on temporally sparse or spatially compressed tactile representations.

In parallel, tactile representation learning focuses on transferable visuotactile embeddings decoupled from control, including Binding Touch [**126**], TVL [**26**], Sparsh [**36**],

AnySkin [**6**], and T3 [**137**]. To enrich contact signals, ViTacGen [**121**] synthesizes visuotactile images from RGB inputs, but remains constrained by visual observability.

In contrast, vision-based tactile sensors such as GelSight [**131**] and DIGIT [**55**] provide dense micro-vision measurements of surface deformation, encoding texture, geometry, and shear-induced slip, which are essential for modeling fine-grained contact dynamics [**103**]. However, existing approaches largely treat tactile sensing as an auxiliary input or a standalone representation, without tightly integrating tactile prediction into sequential decision making.

A.6.4. Predictive World Models in Robotics

Predictive world models learn latent dynamics that enable anticipation of future observations for planning and control. Early work such as World Models [**28**] and Dreamer [**29**] demonstrate imagination-based control via recurrent latent dynamics. Recent large-scale pretraining methods, including R3M [**82**] and V-JEPA-style approaches [**4**], learn structured predictive representations that transfer effectively to downstream robotic tasks.

Several VLA architectures integrate predictive modeling directly into policy learning by conditioning actions on latent rollouts, including DreamVLA [**134**] and WorldVLA [**15**]. In the tactile domain, ViTacFormer [**35**] explores autoregressive visuotactile prediction, but remains focused on representation learning rather than action-conditioned rollout.

Overall, existing world-model approaches predominantly operate on visual observations or abstract latents, and lack explicit mechanisms for predicting fine-grained contact evolution, motivating extensions toward tactile-aware predictive modeling.