CONTACT-VLA: ZERO-SHOT PLANNING AND CONTROL FOR CONTACT-RICH MANIPULATION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

By
Berk Çiçek
September 2025

CONTACT-VLA: ZERO-SHOT PLANNING AND CONTROL FOR CONTACT-RICH MANIPULATION

By Berk Çiçek

September 2025

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Salih Özgür Öğüz(Advisor)

Can Alkan

Ramazan Gökberk Cinbiş

Approved for the Graduate School of Engineering and Science:

Orhan Arıkan
Director of the Graduate School

ABSTRACT

CONTACT-VLA: ZERO-SHOT PLANNING AND CONTROL FOR CONTACT-RICH MANIPULATION

Berk Çiçek
M.S. in Computer Engineering
Advisor: Salih Özgür Öğüz
September 2025

Vision-Language-Action (VLA) systems often lack adaptability and explainability due to their blackbox structure and dependency on fixed action sets from extensive tele-operated datasets, limiting their effectiveness in complex, dynamic manipulation scenarios. To address this issue, we propose a novel VLA framework capable of effectively managing complex, dynamic, and contact-rich manipulation tasks. By integrating foundational vision-language models with motion planning and reactive controllers, our system achieves zero-shot planning and adaptive manipulation without relying on extensive tele-operated action datasets. Unlike conventional VLAs, we explicitly separate the roles of Vision-Language Models (VLM) and Large Language Models (LLM): the VLM handles object parameter extraction and environmental modeling, while the LLM generates initial contact strategies and cost estimations. These two components collaboratively contribute to the creation of a simulated environment in which our dynamic planner operates. Additionally, this modular approach significantly enhances both the explainability and performance of the overall framework, as demonstrated through our rigorous ablation studies. Furthermore, we introduce a memory unit to leverage past manipulation experiences, enabling the generalization and efficient reuse of learned contact strategies and parameter adjustments across diverse manipulation scenarios. Experiments conducted on challenging contact-rich tasks validate our framework's robustness and highlight the critical design elements that contribute to its effectiveness.

Keywords: Vision-Language-Action Systems, Foundation Models for Robotics, Contact-Rich Manipulation, Robot Learning.

ÖZET

CONTACT-VLA: TEMAS-YOĞUN MANİPÜLASYON İÇİN SIFIR-ÖRNEK PLANLAMA VE KONTROL

Berk Çiçek
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Danışmanı: Salih Özgür Öğüz
Eylül 2025

Görsel-Dil-Eylem (VLA) sistemleri, kara-kutu yapıları ve geniş ölçekli teleoperasyonla toplanmış verilerden türetilen sabit eylem kümelerine bağımlılıkları nedeniyle uyarlanabilirlik ve açıklanabilirlikten yoksundur; bu da onların karmaşık, dinamik manipülasyon senaryolarındaki yetkinliklerini kısıtlar. Bu sorunu ele almak üzere, karmaşık, dinamik ve temas-yoğun manipülasyon görevlerini etkin biçimde yönetebilen yeni bir VLA modeli öneriyoruz. Bu model sayesinde, temel görsel-dil modellerini hareket planlama ve reaktif denetleyicilerle bütünleştirerek, kapsamlı teleoperasyon eylem veri kümelerine ihtiyaç duymadan sıfır-örnek planlama ve uyarlanır manipülasyon elde ediyoruz. Modelde, geleneksel VLA'lardan farklı olarak, Görsel-Dil Modelleri (VLM) ile Büyük Dil Modellerinin (LLM) rollerini açıkça ayrıştırıyoruz: VLM, nesne parametrelerinin çıkarımı ve ortamın modellenmesinden sorumluyken, LLM başlangıç temas stratejileri ile maliyet kestirimlerini üretir. Bu iki bileşen, dinamik planlayıcımızın çalıştığı benzetim ortamının oluşturulmasına birlikte katkı sağlar. Ek olarak, modüler yaklaşımımızın modelin hem açıklanabilirliğini hem de performansını anlamlı ölçüde artırdığını, titiz yoksayım (ablation) çalışmalarımızla ortaya koyuyoruz. Ayrıca geçmiş manipülasyon deneyimlerinden yararlanmak için bir bellek birimi sunuyor; böylece öğrenilen temas stratejileri ve parametre ayarlamalarının çeşitli manipülasyon senaryolarında genellenmesini ve verimli biçimde yeniden kullanımını mümkün kılıyoruz. Zorlu, temas-yoğun görevler üzerinde yürütülen deneyler, önerilen çerçevenin sağlamlığını doğrulamakta ve etkinliğine katkı sunan kritik tasarım ögelerini vurgulamaktadır.

Anahtar sözcükler: Görsel-Dil-Eylem Sistemleri; Robotik için Temel Modeller; Temas-yoğun Manipülasyon; Robot Öğrenimi.

Acknowledgement

I would like to express my gratitude to The Scientific and Technological Research Council of Turkey (TUBITAK) for their financial support through the 2210-A National Scholarship Programme for MSc Students provided by the Department of Supporting Scientists (BIDEB).

I would like to express my sincere gratitude to my advisor, Assistant Professor Özgür Öğüz, whose guidance and encouragement greatly influenced my journey. His consistent support has been invaluable in helping me realize my potential as an AI researcher.

I extend my appreciation to Assoc. Professor. Can Alkan and Assoc. Professor. Ramazan Gökberk Cinbiş for their willingness to join as members of my thesis jury and valuable feedback.

I am deeply grateful to my family, Handan Çiçek and Murat Çiçek. Their endless support and constant encouragement have been a source of strength throughout my most difficult times. I owe who I am today to them, as their guidance has shaped my journey. No words can truly capture the depth of their sacrifices that made it possible for me to follow my dreams. I will always remain thankful for their unwavering love and support.

I would also like to extend my gratitude to my colleagues at the Learning for Intelligent Robotic Agents Lab, and especially to my close friend Arda Sarp Yenicesu, whose collaboration greatly enriched both my research and overall experience during this journey.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Foundational models have demonstrated significant success in various fields, leading to increased efforts to apply these models within robotics [1, 2]. Particularly, Vision-Language-Action (VLA) systems have garnered considerable attention for their potential in robotic manipulation tasks [3, 4, 5]. However, existing VLA frameworks still struggle to effectively handle contact-rich manipulation tasks, which constitute a substantial portion of daily interactions [6, 7, 8]. These tasks pose significant challenges, as they require not only precise trajectory planning but also sophisticated interaction force management and adaptive control strategies. Achieving success in such complex scenarios typically necessitates extensive training through teleoperation or detailed dynamic modeling, methods that are labor-intensive and reduce generalizability.

Humans, by contrast, rely on initial estimations, subsequently refine their strategies based on sensory feedback, and adjust interactions accordingly [9, 10, 11]. Similar to this cognitive framework, we propose a novel VLA system, **Contact-VLA**, that integrates reasoning, planning, and control modules into a cohesive architecture. Our model begins by extracting the estimations of physical parameters from given textual task descriptions and image of the environment. Subsequently, the planning stage generates initial contact strategies and actions, which are executed in the real environment through reactive control modules.

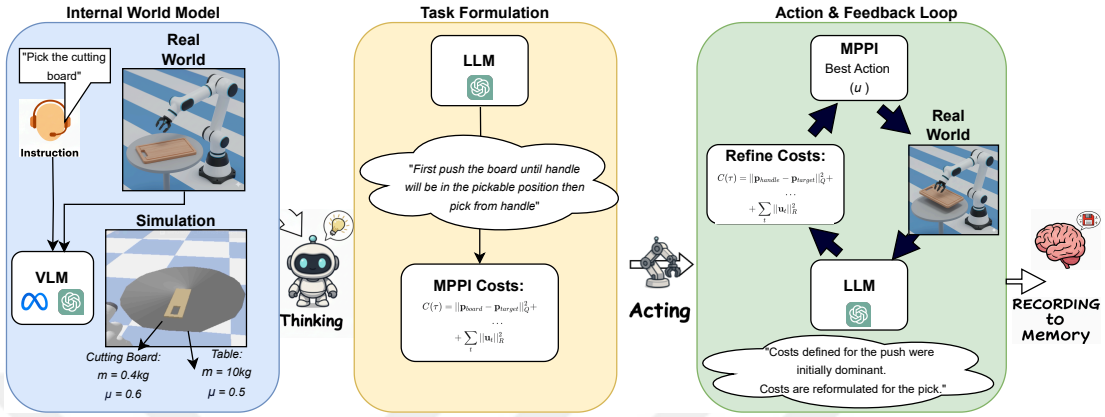

Figure 1.1: The conceptual workflow of Contact-VLA, illustrated with the "pick the cutting board" task. Photorealistic images were synthetically generated.

The outcomes from these actions are continuously monitored and utilized for iterative refinement of plans.

A key innovation of our approach is the strategic integration of Vision-Language Models (VLM) and Large Language Models (LLMs) with motion planners and controllers, substantially enhancing action explainability and enabling more effective reasoning. Our modular structure clearly delineates roles: VLM manages parameter extraction and environment modeling, while LLM provides symbolic reasoning, initial contact strategies, and cost estimations. The reactive controller then applies these symbolic outputs in the physical world, establishing a tight feedback loop between high-level strategy and low-level sensory information. Our main contributions are:

- We propose a novel framework that integrates a Vision-Language-Action (VLA) model with a reactive motion controller, enabling zero-shot planning and robust execution for dynamic, contact-rich manipulation.
- Unlike monolithic VLA architectures, we explicitly separate the roles of the VLM for perception and the LLM for reasoning, a design choice that demonstrably enhances both manipulation performance and system explainability.
- We introduce an LLM-driven, closed-loop feedback mechanism that enables

the system to adapt its plan mid-execution, successfully completing complex, multi-step manipulation sequences.

- Our framework's performance and adaptability are further enhanced by a memory unit that stores and retrieves past experiences to bootstrap effective solutions for novel tasks.

We evaluate Contact-VLA in challenging manipulation tasks such as pushing a box to a wall and flipping it. Our extensive experiments and detailed ablation studies confirm that this modular structure significantly improves system performance and explainability, effectively addressing the limitations of conventional VLA frameworks.

Chapter 2

Related Work

2.1 From End-to-End Policies to Decoupled Reasoning

The advent of foundation models has shifted robotic manipulation towards Vision-Language-Action (VLA) models, which learn general-purpose policies from large-scale datasets [1, 3]. Leading examples like OpenVLA [12], π_0 [13] and RT-X [14] utilize an end-to-end approach, directly mapping multimodal inputs to low-level actions. While powerful, this paradigm's reliance on imitation learning makes it data-dependent and often brittle in novel physical scenarios, particularly those involving complex contact dynamics not well-represented in the training data. To overcome these limitations, an emerging trend decouples high-level reasoning from low-level control. Frameworks like ThinkAct [15], Inner Monologue [16] and those using Embodied Chain-of-Thought (ECoT) [17] leverage LLMs to generate explicit reasoning steps that guide a separate, learned action policy. Similarly, MolmoAct [18] produces mid-level spatial plans as "trajectory traces" before predicting actions, enhancing explainability and steerability. The OneTwoVLA architecture [19] formalizes this by explicitly modeling a System 1 (fast, reactive

acting) and System 2 (slow, deliberate reasoning). Another influential contribution in this direction is ReAct [20] which proposes a prompting framework where large language models generate reasoning traces interleaved with task-specific actions. This synergy between reasoning and acting significantly improves task success and interpretability in interactive environments such as QA, web navigation, and text-based decision-making. While ReAct primarily focuses on knowledge-intensive and textual tasks, it establishes a general principle of coupling reasoning and action within a closed-loop system. Our work, Contact-VLA, extends this philosophy into robotics, grounding the reasoning outputs of an LLM into a motion planning and reactive control framework for contact-rich manipulation. Our work, Contact-VLA, aligns with this decoupling philosophy but takes a distinct neuro-symbolic path by grounding the LLM's reasoning directly in a controller, rather than another learned model.

2.2 Integrating Foundation Models with Motion Planners and Controllers

A promising alternative to end-to-end learning is the integration of foundation models with traditional motion planners, leveraging semantic understanding to guide physically-grounded trajectory optimization. For instance, IMPACT [21] utilizes a VLM to generate a static 3D cost map of the environment, assigning higher costs to fragile objects to enable a planner like RRT* to find paths with "acceptable contact". Similarly, VLMPC [22] embeds a VLM within a Model Predictive Control (MPC) loop, where it provides perceptual guidance by identifying sub-goals and sampling candidate action sequences. Contact-VLA significantly advances this paradigm by elevating the role of the LLM from a perceptual guide to a high-level strategist. Instead of merely identifying objects or goals, our LLM formulates the structure of the Model Predictive Path Integral (MPPI) controller cost function itself and proposes symbolic contact strategies. This approach grounds abstract, commonsense reasoning directly into the mathematical formulation of the optimal control problem, enabling a more profound

and explainable link between high-level intent and low-level dynamic execution.

2.3 Tackling Contact-Rich Manipulation

Contact-rich manipulation remains a grand challenge, as it requires nuanced force control and physical understanding beyond simple trajectory generation. One major line of work addresses this by augmenting perception with physical sensors, such as in ForceVLA [7], TLA [6] and VLA-Touch [23], which explicitly integrate force or tactile data into the policy's input stream. Further research in this direction, such as Reactive Diffusion Policy (RDP) [8] and FACTR [24], proposes specialized architectures and training curricula to more effectively fuse this real-time feedback into a learned policy. While effective, this hardware-centric approach risks creating a new data bottleneck, as it requires large-scale, specialized multimodal demonstration datasets that are difficult to collect [1]. Contact-VLA tackles this problem from a different angle. Although it also leverages real-time force feedback for its reactive controller, it critically eliminates the need for prior demonstration datasets containing such data. Instead, our framework uses the LLM to formulate a high-level strategy and cost function that explicitly reasons about interaction forces, which the MPPI controller then robustly executes by adapting to live sensor data online. This neuro-symbolic approach combines the benefits of physical sensing with the zero-shot reasoning of foundation models, thereby avoiding the imitation learning bottleneck while still achieving precise, force-aware control.

2.4 Qualitative Comparison with State-of-the-Art VLA Manipulation Frameworks

As the comparative analysis in Table 2.1 illustrates, the field of robotic manipulation has historically involved a trade-off. End-to-end VLA models achieve

Table 2.1: Comparison with State-of-the-Art VLA Manipulation Frameworks

Framework	Primary Modality	Planning & Control Strategy	Reasoning Mechanism	Data Requirement
OpenVLA [12]	Vision, Language	End-to-End Learned Policy (Action Token Prediction)	Implicit (in VLM backbone)	Large-scale Imitation Learning Demos
π_0 (pi-zero) [13]	Vision, Language	End-to-End Learned Policy (Flow Matching)	Implicit (in VLM backbone)	Large-scale Imitation Learning Demos
ForceVLA [7], TLA [6]	Vision, Language, **Tactile/Force**	End-to-End Learned Policy	Implicit (in network weights)	Large-scale Tactile/Force Demos
VLA-Touch [23]	Vision, Language, **Tactile**	VLA Policy + Tactile-based Refinement Controller	Explicit VLM Planning + Semantic Tactile Feedback	Leverages pretrained models; no VLA fine-tuning
ThinkAct [15], ECoT [17]	Vision, Language	End-to-End Learned Policy	**Explicit LLM Reasoning** (Chain-of-Thought)	Large-scale Imitation Learning Demos
OneTwoVLA [19]	Vision, Language	Unified Policy (Adaptive Acting & Reasoning)	**Explicit LLM Reasoning** (System 2)	Imitation Demos + Synthetic Reasoning Data
MolmoAct [18]	Vision, Language	Multi-stage Pipeline (Perception -¿ Spatial Plan -¿ Action)	**Explicit Spatial Reasoning** (Trajectory Traces)	Large-scale Imitation Learning Demos
IMPACT [21]	Vision, Language	VLM-based Static Cost Map + RRT*	Implicit (semantic object labeling)	N/A (Planner-based)
VLMPC [22]	Vision, Language	VLM-guided Model Predictive Control (MPC)	**Explicit VLM Reasoning** (for cost & sampling)	N/A (Planner-based)
Contact-VLA (Ours)	Vision, Language, **Tactile/Force**	**LLM-guided MPPI + Reactive Control**	**Explicit LLM Reasoning** (Strategy Formulation)	**Zero-Shot (No Demos)**

impressive behaviors but are fundamentally constrained by large-scale demonstration datasets, while traditional planner-based systems are zero-shot but often lack high-level semantic reasoning. Contact-VLA is designed to synthesize the strengths of these disparate paradigms. To the best of our knowledge, Contact-VLA is the first framework to simultaneously integrate **explicit LLM-driven strategy formulation** with a **dynamic, reactive controller** that leverages real-time **tactile and force feedback**, all while operating in a **zero-shot** manner that completely eliminates the need for prior demonstration data.

Chapter 3

Background

In this section, we introduce the Model Predictive Path Integral (MPPI) controller, which forms the core of our methodology, and provide a formal problem formulation for the contact-rich manipulation tasks we address.

3.1 Model Predictive Path Integral (MPPI)

Model Predictive Path Integral (MPPI) [25] is a sampling-based Model Predictive Control (MPC) algorithm designed to solve stochastic optimal control problems. It is particularly effective for systems with nonlinear and complex dynamics. MPPI operates by simulating thousands of potential control sequences in parallel from the current state to determine the optimal subsequent control input.

Consider a system with discrete-time stochastic dynamics described by $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) + \boldsymbol{\epsilon}_t$, where $\mathbf{x}_t$ is the state of the system, $\mathbf{u}_t$ is the control input, and $\boldsymbol{\epsilon}_t$ represents system noise. The objective of MPPI is to find a control sequence $\mathbf{U} = \{\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{H-1}\}$ that minimizes an expected cost function:

$$J(\mathbf{U}) = \mathbb{E}\left[\phi(\mathbf{x}_H) + \sum_{t=0}^{H-1} q(\mathbf{x}_t, \mathbf{u}_t)\right] \tag{3.1}$$

Here, $\phi(\mathbf{x}_H)$ is the terminal cost, and $q(\mathbf{x}_t, \mathbf{u}_t)$ is the running (or stage) cost, which typically includes terms for tracking error and control effort. H is the planning horizon.

At each time step, MPPI samples K candidate control sequences by adding random noise perturbations to a nominal sequence. These are rolled out in simulation to get trajectories, and the total cost S_k for each is computed. The optimal control is then calculated via an exponential weighting of these costs. This process is repeated at each time step following the receding horizon principle. In our approach, the symbolic reasoning provided by the LLM forms the initial structure for the running cost function $q(\cdot)$.

3.2 Problem Formulation

In this work, we address long-horizon, contact-rich manipulation tasks specified by visual and linguistic commands. Our goal is to develop a robotic system capable of interacting with objects of unknown physical properties (e.g., mass, friction) and generalizing to new scenarios in a zero-shot manner.

We formulate the problem as a partially observable Markov decision process (POMDP). The state of the system, $\mathbf{x}_t \in \mathcal{X}$, includes the robot's state $\mathbf{x}_r(t)$ and the states of environmental objects $\mathbf{x}_o(t)$. The action space, $\mathcal{U}$, consists of continuous control commands applicable to the robot's end-effector.

At the beginning of each task, the system receives an RGB image I and a natural language instruction T. The system's objective is not to learn a fixed policy, but rather to compute, at each step, a control action u_t that leads to a sequence of actions $\mathbf{U} = \{\mathbf{u}_0, \ldots, \mathbf{u}_{H-1}\}$ that successfully completes the task.

The core challenge is to bridge the gap from high-level, multimodal inputs (I, T) to these low-level, continuous control actions. Our approach decomposes this problem into two stages:

1. **Strategy Formulation:** We use the VLM and LLM to translate the multimodal inputs (I, T) into an initial cost function J_0 and contact candidates C_0 for the planner.

2. **Online Planning and Control:** We employ the MPPI planner to compute the optimal action sequence online, guided by the LLM's strategy and adapted using real-time sensory feedback.

This formulation accurately frames our system as an online planner that reasons and computes actions on the fly, rather than an agent executing a prelearned, static policy.

Chapter 4

Methodology

Contact-VLA is a neuro-symbolic framework designed for zero-shot, contact-rich manipulation. It strategically decouples high-level reasoning from low-level control by integrating a VLM, a LLM acting in two distinct roles (Task Formulation and Online Adaptation), a Memory Unit for experience retrieval, and a MPPI for reactive execution. The overall architecture, which features nested feedback loops for rapid and robust adaptation, is illustrated in Figure 4.1. Below, we detail each component of this architecture.

4.1 Environment Perception and World Model Initialization

The first step is to translate raw visual and textual inputs into a structured, physics-aware world model. For this process, we make a key assumption: the system has access to a real-time stream of unlabeled poses and sizes for all interactable objects, for instance, from an external object tracking system. The challenge is that the identity of the object at each pose is unknown. Therefore, unlike traditional VLA approaches that extract object poses and sizes directly from images, our VLM model operates based on the known object poses and sizes

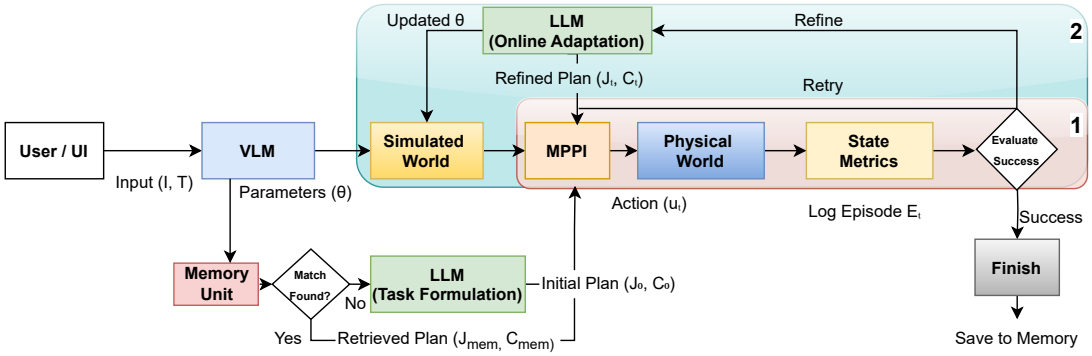

Figure 4.1: The overall architecture of the **Contact-VLA** framework. Given an input image I and task description T, the VLM extracts world parameters θ. If the Memory Unit finds a similar successful experience, its retrieved plan (J_{mem}, C_{mem}) is used to guide the MPPI controller. Otherwise, the LLM (Task Formulation) module generates an initial plan (J_0, C_0). The system then enters the main execution cycle, which is governed by two nested feedback loops labeled (1) and (2). **(1) The Inner Loop** is a high-frequency re-planning cycle. At each step, the MPPI, guided by the current plan, generates an action u_t based on the latest 'State Metrics'. This loop ('Retry') continues until the task succeeds or a refinement is needed. **(2) The Outer Loop** is a low-frequency, high-level adaptation cycle. If the inner loop fails persistently, the 'Refine' path is taken, where the LLM (Online Adaptation) updates both the world model parameters (θ) for the 'Simulated World' and the strategic 'Refined Plan (J_t, C_t)' for the MPPI. Successful episodes are stored back into the Memory Unit.

provided by the external system.

The role of our VLM module is therefore to disambiguate the scene by identifying the objects and grounding them to their corresponding pose data. This process involves:

1. **Object Segmentation:** We first apply the Segment Anything Model (SAM) to the input image I to extract segmentation masks for all objects.

2. **Identification and Parameter Estimation:** These masks, each associated with a known pose, are fed into a multimodal foundation model (GPT-4o) along with the raw image and task description T. The model performs two functions: it identifies the object corresponding to each mask (e.g., "this is the red box") and estimates its unknown physical properties.

The output, $\theta = \text{VLM}(I, T)$, is a structured representation containing each object's semantic label, its now-associated pose and size, and its estimated physical attributes such as mass and friction coefficients. These parameters are crucial as they are used to initialize the internal 'Simulated World' that the MPPI planner will operate on.

4.2 LLM-driven Task Formulation and Memory Retrieval

Once the world is perceived, the system formulates a concrete plan. This is handled by the 'LLM (Task Formulation)' module, which can generate a plan from scratch or leverage past experiences from the 'Memory Unit'.

Memory Retrieval: Before invoking the LLM, the system queries the 'Memory Unit' with the current world parameters θ and task T. Our memory module is based on Retrieval-Augmented Generation (RAG), storing successful "experience episodes" indexed by task definitions and environmental parameters. Instead of relying on predefined similarity metrics, the LLM embeds the current task into a latent semantic space to retrieve the most relevant past experience:

$$(J_{mem}, C_{mem}) = \text{RAG}_{\text{Retrieve}}(T, \theta) \tag{4.1}$$

If a sufficiently similar and successful past experience is found, its stored plan (J_{mem}, C_{mem}) is retrieved, bypassing the computationally expensive LLM call and accelerating performance.

Plan Generation from Scratch: If no suitable memory is found, the 'LLM (Task Formulation)' module is invoked. It acts as a high-level strategist, translating the task T and world parameters θ into a formal optimization problem. Its output is an initial plan tuple (J_0, C_0), where:

- **Initial MPPI Cost Function (J_0):** The LLM generates the mathematical structure and relative weights of a cost function. Specifically, for a given

task, the LLM provides a structured cost functional, for instance:

$$J_0(\mathbf{x}_{0:H}, \mathbf{u}_{0:H-1}) = \sum_{t=0}^{H-1} \Big[w_d \left\| \mathbf{p}_{\text{target}} - \mathbf{p}_{\text{obj}}(t) \right\|^2 + w_c \, \mathbb{I}\{\text{no contact at } t\} + w_u \left\| \mathbf{u}_t \right\|^2 \Big] \tag{4.2}$$

Here, the weights w_d, w_c, w_u are determined by the LLM based on the task description (e.g., for a pushing task, w_c would be high). $\mathbf{p}_{\text{obj}}(t)$ is the object's position at time t, and $\mathbb{I}\{\cdot\}$ is an indicator function penalizing the absence of contact.

- **Initial Contact Strategy (C_0):** The LLM proposes promising surfaces for making contact to guide the planner's exploration. It specifies a set of focused surface regions $\{R_j\}$, from which we generate candidate contact points as:

$$C_0 = \bigcup_{j=1}^{M} \Big\{ c_j + e_j \big(\cos\phi \, \mathbf{t}_{1,j} + \sin\phi \, \mathbf{t}_{2,j} \big) \;\Big|\; \phi = \tfrac{2\pi k}{N_j}, \; k = 0, \ldots, N_j - 1 \Big\} \tag{4.3}$$

where for each region j, c_j is the center, e_j is the radius, and $\{\mathbf{t}_{1,j}, \mathbf{t}_{2,j}\}$ are tangent vectors. This biases sampling towards strategically advantageous regions.

This generated strategy is then used within the MPPI sampling process by biasing the initial control perturbations to explore actions that bring the end-effector closer to the LLM-proposed contact regions, thereby significantly pruning the search space.

4.3 Reactive Planning and Execution (The Inner Loop)

The core of our system is a high-frequency, reactive execution cycle governed by the MPPI controller. This corresponds to the Inner Loop (1) in Figure 4.1.

MPPI Formulation: The MPPI controller solves a stochastic optimal control problem at each timestep. Given a state-transition model $x_{t+1} = f(x_t, u_t) + \epsilon_t$, where x_t is the system state, u_t is the control input, and ϵ_t is system noise, the objective is to find a sequence of control inputs $U = \{u_0, \ldots, u_{H-1}\}$ that minimizes the expected total cost:

$$U^* = \arg\min_U \mathbb{E}\left[\phi(x_H) + \sum_{t=0}^{H-1} q(x_t, u_t)\right] \tag{4.4}$$

where $\phi(x_H)$ is a terminal state cost and $q(x_t, u_t)$ is the running cost at each step. The LLM-generated cost function, J_0 (from Eq. 4.2), directly defines the terms used in this optimization. Specifically, the running cost $q(x_t, u_t)$ is the expression inside the summation of J_0:

$$q(x_t, u_t) = w_d \left\|\mathbf{p}_{\text{target}} - \mathbf{p}_{\text{obj}}(t)\right\|^2 + w_c \, \mathbb{I}\{\text{no contact at } t\} + w_u \left\|\mathbf{u}_t\right\|^2 \tag{4.5}$$

MPPI approximates this optimization by:

1. Sampling K control sequence perturbations $\delta U_k \sim \mathcal{N}(0, \Sigma)$ from a Gaussian distribution.
2. Creating K rollout trajectories by applying the perturbed control sequences $V_k = U_{prev} + \delta U_k$ in the 'Simulated World'.
3. Calculating the total cost $S(V_k)$ for each of the K trajectories.
4. Computing an exponentially weighted average of the perturbations to update the control sequence:

$$U_{new} = U_{prev} + \sum_{k=1}^{K} w_k \delta U_k \quad \text{where} \quad w_k = \frac{\exp\left(-\frac{1}{\lambda} S(V_k)\right)}{\sum_{j=1}^{K} \exp\left(-\frac{1}{\lambda} S(V_j)\right)} \tag{4.6}$$

Following the receding horizon principle, only the first action, u_0, of the newly optimized sequence U_{new} is executed.

Reactive Control Augmentation: To achieve robustness against the inherent sim-to-real gap, we augment the nominal planned action with a real-time feedback term. The final control command ν_t sent to the robot is:

$$\nu_t = u_t + K_f \cdot (x_{\text{des}} - x_{\text{measured}}) \tag{4.7}$$

where u_t is the action computed by MPPI, the error term is calculated from real-time sensors (e.g., force/torque, proprioception), and K_f is a feedback gain matrix. This 'Retry' loop continues at a high frequency, constantly re-planning and correcting based on physical feedback.

4.4 Online Adaptation via LLM-driven Refinement (The Outer Loop)

If the inner loop fails to make progress after a predefined number of attempts, a hyperparameter we denote as N_{retry}, the system triggers the low-frequency Outer Loop (2). This invokes the 'LLM (Online Adaptation)' module, which acts as a diagnostician and re-strategist.

The input to this module is the logged episode data E_t, which contains the history of states, actions, the contact strategies and cost functions that were used, and the estimated physical parameters that led to the failure. By analyzing this rich context, the LLM performs two critical functions:

1. **World Model Correction:** The LLM can refine the initial physical parameter estimates. For example, if the robot pushes an object but the object moves less than predicted, the LLM can infer that its initial estimate of the object's mass was too low and output an 'Updated θ'.

2. **Strategy Refinement:** The LLM can also alter the plan itself. It might change the weights of the cost function (e.g., prioritizing force control over

position accuracy) or propose an entirely new contact strategy. This results in a 'Refined Plan (J_t, C_t)'.

This refined world model and plan are then fed back into the inner loop, allowing the system to learn from its failures and adapt its entire approach within a single task execution.

Chapter 5

Experiments

We conducted a series of experiments in a simulated environment to rigorously evaluate the performance of Contact-VLA. Our evaluation is designed to answer three key research questions: **(RQ1)** How does Contact-VLA perform on complex, contact-rich manipulation tasks in a zero-shot setting compared to state-of-the-art baselines? **(RQ2)** How critical is each core component of our neuro-symbolic architecture—specifically the VLM/LLM role separation, the online refinement loop, and the memory unit—to the overall success? **(RQ3)** Can the system demonstrate robustness and adaptability by reasoning about and recovering from failures?

5.1 Experimental Setup

5.1.1 Simulation Environment:

All experiments were conducted within the PyBullet physics simulator. The robot is a simulated 7-DoF Franka Emika Panda arm with a parallel jaw gripper. Sensory inputs include 224x224 RGB images from a fixed camera, proprioceptive feedback, and force/torque data from the simulated physics engine, which is

essential for our reactive control.

5.1.2 Implementation Details:

The VLM and LLM modules were implemented using the GPT-4o API. The MPPI controller was built on top of PyBullet. To ensure real-time performance, our MPPI implementation is heavily parallelized on a GPU, leveraging principles from [26] to roll out $K = 200$ trajectories over a planning horizon of $H = 50$ steps for each control cycle. The key hyperparameters for the controller are the temperature $\lambda = 0.1$ and the outer-loop trigger threshold $N_{retry} = 15$ persistent failures. The entire framework was run on a desktop computer with an Intel Core i9-13900K CPU, 64GB of RAM, and a single NVIDIA RTX 4060 Ti GPU.

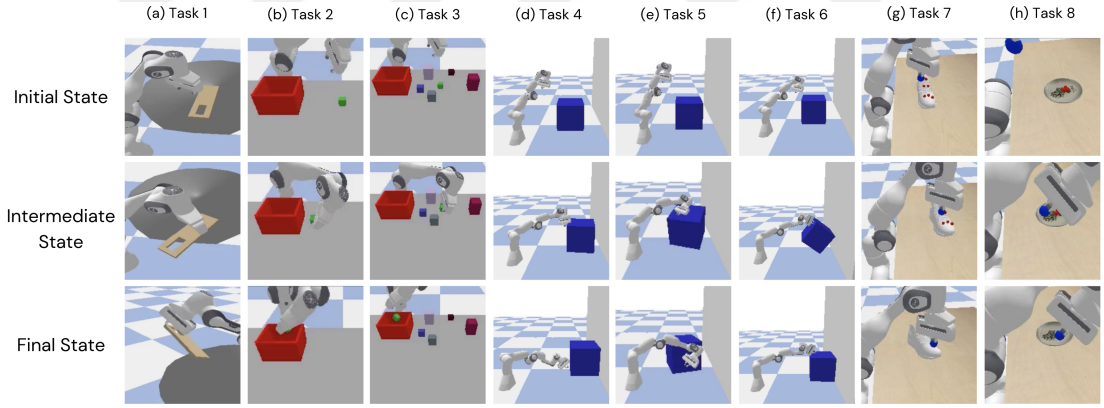

Figure 5.1: Visualization of the eight challenging, contact-rich manipulation tasks used to evaluate Contact-VLA. Each column, from (a) to (h), illustrates a distinct task with snapshots of its initial, intermediate, and final states. The tasks are: (a) Push+Pick Board, (b) Pick Box, (c) Pick+Place Clutter, (d) Push Const. Force, (e) Flip Box, (f) Flip w/ Wall, (g) Wipe Shoe, and (h) Wipe Plate.

5.2 Tasks and Evaluation Metrics:

We evaluated our framework on eight challenging, contact-rich manipulation tasks, shown in Fig. 5.1, designed to be difficult for purely vision-based, collision-avoidant planners. Each task was performed 10 times with randomized initial

object poses. The tasks are as follows: **T1: Push and Pick Cutting Board**, a multi-stage task testing pushing and reasoning about object parts and pose for grasping; **T2: Pick Box & T3: Pick and Place in Clutter**, a standard pick-and-place task to establish a baseline; **T4: Push with Constant Force**, testing the reactive controller's ability to manage force feedback; **T5: Flip Box**, a dynamically complex maneuver; **T6: Flip with Wall**, requiring multi-contact reasoning to use the wall as a fixture; and **T7: Wipe Shoe & T8: Wipe Plate**, which test the ability to maintain contact along curved, non-planar surfaces. **Evaluation Metrics:** We use two primary metrics: **Success Rate** (binary measure across 10 trials) and **Average Completion Time** in seconds for successful trials.

5.3 Comparative Baselines:

We compare Contact-VLA against state-of-the-art methods and three internal ablations. The **State-of-the-Art Baselines** are: **OpenVLA** [12] and π_0 [13], a leading end-to-end VLA models based on imitation learning that directly map multimodal inputs to low-level robot actions. Our **Ablation Baselines** are: **Contact-VLA (w/o Memory)**, which removes the experience retrieval mechanism; **Contact-VLA (w/o Refinement)**, which disables the online adaptation loop; and **Contact-VLA (Unified VLM)**, which uses a single multimodal prompt for both perception and planning to test the importance of separating VLM/LLM roles.

5.4 Results and Analysis

Table 5.1 presents a comprehensive overview of our experimental findings.

Table 5.1: Comprehensive comparison against state-of-the-art baselines and ablation study of our method variants across all tasks. Performance is measured by success rate (x/10 trials) and average completion time in seconds for successful trials.

Method	T1: Push+Pick Board		T2: Pick+Place Box		T3: Pick+Place Clutter		T4: Push Const. Force		T5: Flip Box		T6: Flip w/ Wall		T7: Wipe Shoe		T8: Wipe Plate	
	Success	Time (s)	Success	Time (s)	Success	Time (s)	Success	Time (s)	Success	Time (s)	Success	Time (s)	Success	Time (s)	Success	Time (s)
State-of-the-Art Baselines (PyBullet)																
OpenVLA [12]	0/10	-	0/10	-	0/10	-	0/10	-	0/10	-	0/10	-	0/10	-	0/10	-
π_0 [13]	0/10	-	0/10	-	0/10	-	0/10	-	0/10	-	0/10	-	0/10	-	0/10	-
State-of-the-Art Baselines (LIBERO)																
OpenVLA [12]	0/10	-	10/10	5	10/10	6	0/10	-	0/10	-	0/10	-	0/10	-	0/10	-
π_0 [13]	0/10	-	10/10	5	10/10	6	0/10	-	0/10	-	0/10	-	0/10	-	0/10	-
Our Method (Ablation Study)																
Contact-VLA (Ours, with Memory)	**8/10**	**16**	**10/10**	**9**	**10/10**	**9**	**10/10**	**26**	**10/10**	**8**	**10/10**	**7**	**10/10**	**59**	**10/10**	**10**
Contact-VLA (Full, w/o Memory)	4/10	21	10/10	16	9/10	19	10/10	28	10/10	9	10/10	13	10/10	74	10/10	12
w/o Online Refinement	0/10	–	10/10	8	3/10	9	10/10	21	10/10	8	6/10	6	10/10	55	10/10	9
w/o Planner (w/o Separate Roles)	0/10	–	2/10	13	0/10	–	6/10	24	0/10	–	0/10	–	0/10	–	4/10	10

5.4.1 State-of-the-Art Comparison (RQ1)

Contact-VLA significantly outperforms both baselines, particularly in tasks requiring sophisticated physical reasoning and interaction forces (T1, T4-T8). While OpenVLA and π_0 perform reasonably on simpler pick-and-place tasks (T2, T3), their performance degrades sharply otherwise. Both model's policy, lacking an explicit physical model, fails to generate the non-obvious, multi-contact strategy required for the wall-flip (T6) and they struggle with tasks where interaction force is the critical variable (T4, T7, T8), as their action policy is not trained to optimize for specific force profiles. In contrast, our framework excels by allowing the LLM to directly formulate a cost function that optimizes for these physical interaction dynamics.

A key finding from our results is the two-fold failure of the baseline models, highlighting issues in both generalization and physical reasoning. Firstly, conventional VLA models demonstrate a critical dependency on their training environment. As shown in Table 5.1, both OpenVLA and π_0 fail completely across all tasks in the unseen PyBullet environment, underscoring their inability to generalize. Secondly, even within their native LIBERO environment where they were trained, these baselines only succeed at simple, memorized pick-and-place tasks they have seen before (T2, T3). They consistently fail at all contact-rich tasks (T1, T4-T8) because they lack an explicit physical model to understand and control interaction forces or to generate the complex, multi-contact strategies required for success.

5.4.2 Ablation Study Analysis (RQ2)

Our ablation studies clearly demonstrate the necessity of each component in our architecture.

5.4.2.1 The Synergy of Separated VLM/LLM Roles

The *Contact-VLA (Unified VLM)* variant, which tasked a single VLM with both perception and planning, failed on nearly all complex tasks. This starkly illustrates our core hypothesis: separating the role of a VLM for perception from a dedicated LLM for strategy formulation is crucial for robust performance. The specialized modules provide more reliable and structured outputs for the planner.

5.4.2.2 The Importance of Online Refinement

The *w/o Refinement* variant showed a dramatic performance drop in multi-stage tasks like T1 (Push and Pick Board), with success falling from 4/10 to 0/10. In this task, the initial plan often failed because the VLM's initial friction estimate was slightly off, causing the board to slip during the pick. The full Contact-VLA framework, however, used the outer loop for the LLM to diagnose this from the physical outcome, refine the friction parameter in its world model, and successfully complete the task. This shows the system's ability to learn from failure.

5.4.2.3 The Benefit of Experience Reuse

The full framework *with Memory* consistently achieved the highest success rates and fastest completion times. For instance, in T1 and T3, memory boosted the success rate from 4/10 to 8/10 and 9/10 to 10/10, respectively. By retrieving a successful "push-to-edge" strategy from a past experience, the system provided

the planner with a superior initialization, accelerating convergence and leading to more robust solutions.

5.4.3 Robustness Analysis (RQ3)

5.4.3.1 Analysis of LLM-Guided Contact Strategy

To isolate the contribution of the LLM's initial contact strategy (C_0), we conducted a targeted ablation on the challenging "Flip with Wall" task (T6). We compared the performance of our full framework against a variant where the LLM only provided the cost function (J_0), forcing the MPPI planner to rely on uninformed, random sampling to find useful contact points.

The results, shown in Figure 5.2. The guided trajectory (With Strategy, green) is direct and purposeful, immediately moving the end-effector to the correct corner of the box to initiate the flip. In contrast, the unguided trajectory (Without Strategy, red) is chaotic and inefficient, exploring large, irrelevant portions of the workspace. The planning cost for the unguided agent remains high and erratic, indicating a constant struggle to find a viable plan. This visual difference is confirmed by the quantitative results: the guided approach was **83.9**% **faster** (32 vs. 199 steps) and the end-effector traveled a **63.9**% **shorter path** (1.33 m vs. 3.69 m). This analysis provides clear evidence that the LLM's symbolic contact strategy is critical for transforming a computationally intractable, long-horizon contact problem into a solvable one by intelligently pruning the vast action search space.

5.4.3.2 Deep Dive: Nuanced Force Control in the Wipe Shoe Task

The "Wipe Shoe" task (T7) serves as an excellent case study to highlight the unique advantages of our neuro-symbolic approach. The task presents a significant challenge: the robot must apply sufficient force with a sponge to clean

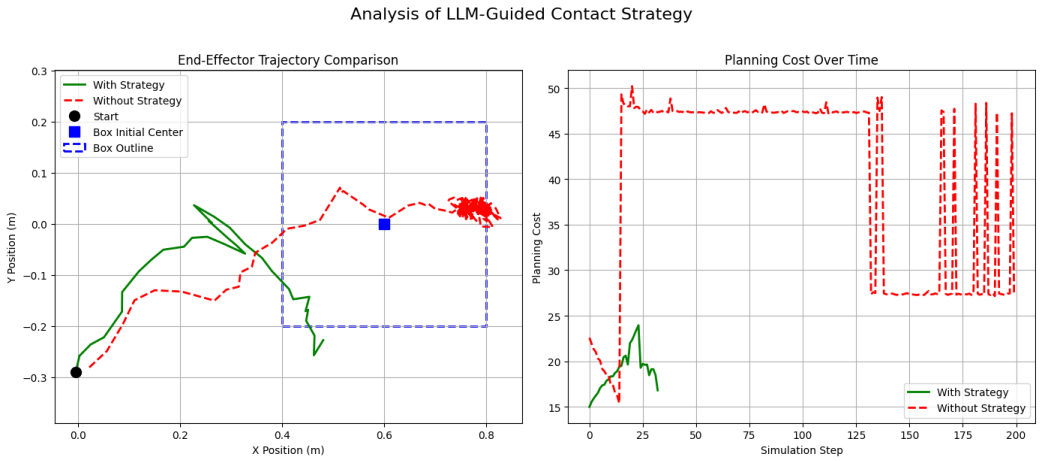

Figure 5.2: Ablation of the LLM-guided contact strategy on the "Flip with Wall" task. **(Left)** The trajectory with the LLM's strategy (green) is direct and efficient, while the unguided trajectory (red) is erratic. **(Right)** The planning cost for the guided agent is significantly lower and more stable, indicating an easier optimization problem.

stains from a curved surface while simultaneously ensuring the un-anchored shoe remains stable and does not get pushed away. This dual-objective problem is difficult for end-to-end policies, which struggle to reason about and explicitly control interaction forces without extensive, task-specific training.

Contact-VLA excels in this scenario by leveraging its hierarchical structure. First, the VLM, drawing on its vast pre-trained knowledge, provides a reasonable initial estimate for the shoe's physical parameters (θ), such as mass and friction, to construct the 'Simulated World'. Subsequently, the 'LLM (Task Formulation)' module translates the implicit instruction "wipe the shoe gently" into a concrete optimization problem for the MPPI controller. The generated cost function, J_0, includes terms that reward reaching the stain locations while heavily penalizing any displacement of the shoe's center of mass.

The MPPI planner then uses its rollouts to find a control policy that optimally balances these objectives. As shown in Figure 5.3, our framework successfully executes this nuanced behavior. The end-effector travels a total path length of 1.33 m, making precise, targeted movements to cover the stains, while the shoe is displaced by only 0.24 m over the entire task. Critically, the reactive controller

maintained an average contact force of 5.48 N on the stains, demonstrating that the system can apply meaningful force while respecting the stability constraints defined by the LLM. This case study illustrates how Contact-VLA translates high-level, abstract goals into precise, force-aware physical interactions, a key capability for contact-rich manipulation.

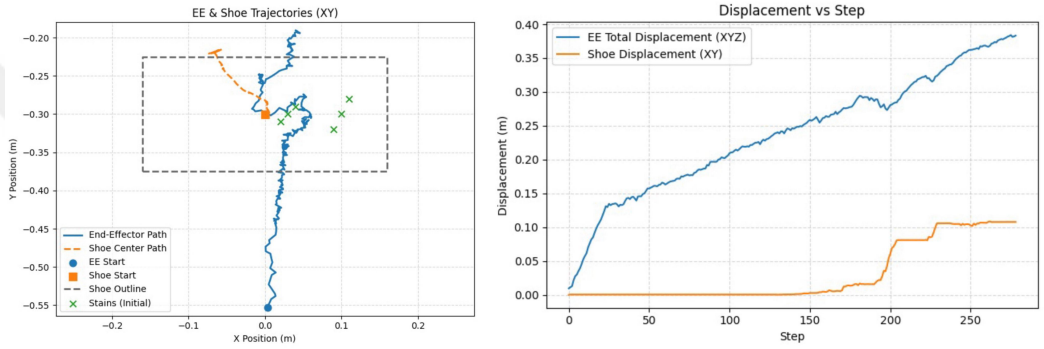

Figure 5.3: Detailed analysis of the "Wipe Shoe" task (T7). **(a)** Top-down view of the end-effector (EE) and shoe center trajectories. The EE path (blue) shows targeted movements towards the stains (green crosses), while the shoe path (orange) shows minimal displacement. **(b)** Cumulative displacement over time. The EE travels significantly further than the shoe, quantitatively demonstrating the controller's ability to stabilize the object while performing the task.

5.4.3.3 Robustness Through Online Parameter Adaptation

Beyond strategy and cost function refinement, Contact-VLA's 'Online Adaptation' module, driven by the LLM, exhibits a crucial ability to correct the agent's internal world model in real-time. To demonstrate this, we intentionally initialized the 'Simulated World' with significantly incorrect physical parameters for the cutting board in a "Push and Pick Cutting Board" task. Specifically, the agent was initially "told" the board's mass was 2.0 kg (ground truth 0.1 kg) and its friction coefficient was 0.9 (ground truth 0.5). These initial biases represent a severe sim-to-real gap or a VLM hallucination.

Figure 5.4 vividly illustrates the adaptation process. The LLM's 'Online Adaptation' module, when triggered by persistent failures in the inner loop (e.g., the

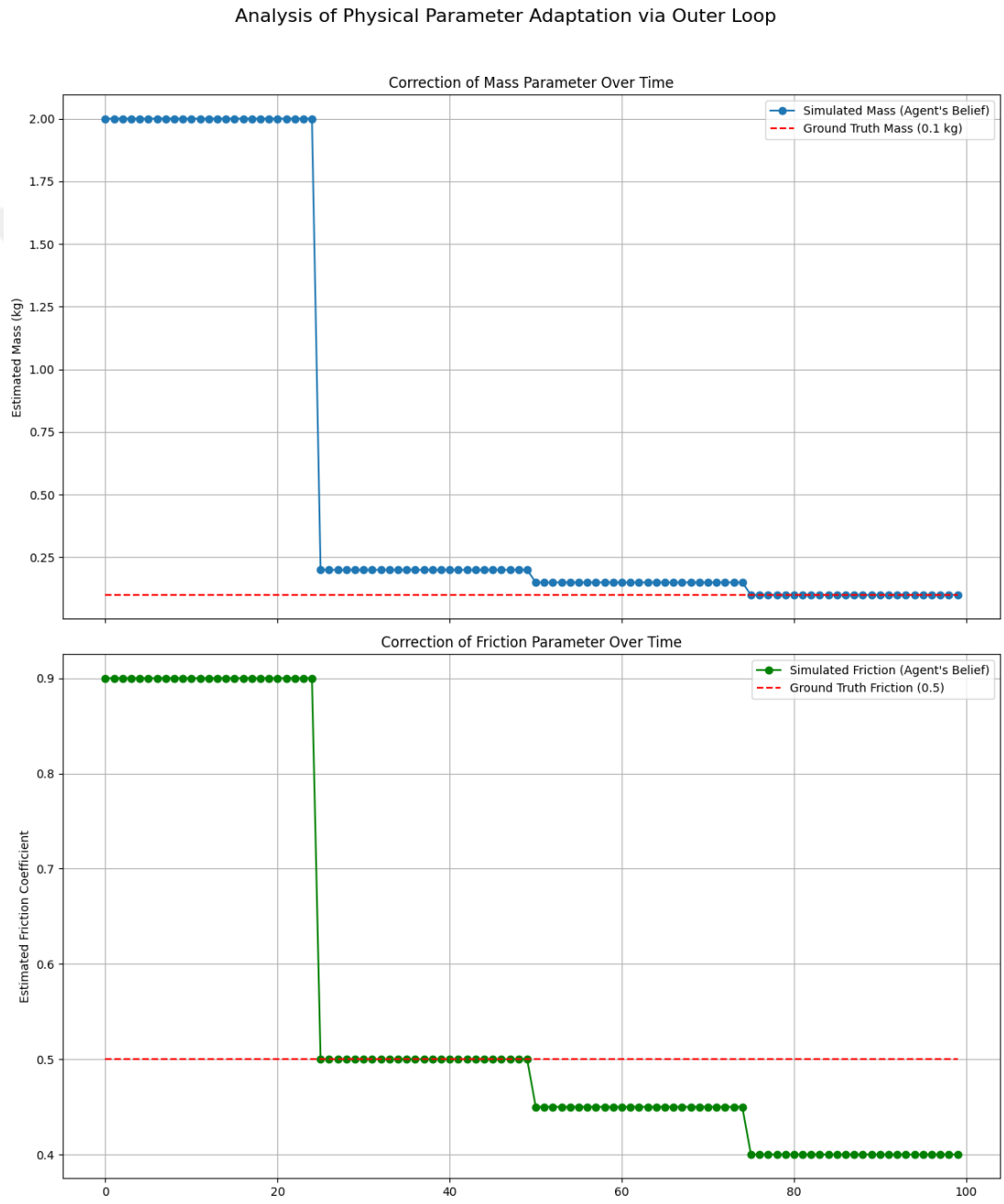

Figure 5.4: Online correction of physical parameter estimates.

board not moving as expected despite high pushing force), diagnosed the discrepancy. Through an iterative refinement process, it progressively adjusted its estimated mass and friction parameters. As shown in the graph, after several adaptation cycles, the agent's belief about both mass and friction converged remarkably close to their true values. This online correction of physical parameters is fundamental to the framework's robustness, allowing it to overcome initial environmental mischaracterizations and successfully execute contact-rich tasks that would otherwise fail due to a misaligned internal world model. This capability is a cornerstone for deploying robots in unknown environments where accurate a priori physical models are often unavailable.

5.4.3.4 Sequential Reasoning and Experience Reuse in the Cutting Board Task

The "Push and Pick Cutting Board" task (T1) is specifically designed to test the framework's capabilities in long-horizon, sequential manipulation. The task's difficulty lies in its two distinct phases: a stable push across a surface followed by a precise grasp of a handle. As evidenced by our results in Table 5.1, this sequential challenge highlights the importance of two core components of our architecture: online adaptation and experience reuse.

First, the long-horizon nature of the task means that small errors in the initial world model can accumulate and lead to failure in the later stages. This is clearly demonstrated by the *w/o Refinement* ablation, which failed entirely on this task (0/10 success rate). While its initial plan was often sufficient for the pushing phase, slight inaccuracies in the estimated friction parameter caused the board to end up in an unexpected final pose, leading to a failed grasp. Our full model, however, leverages the outer loop to learn from the outcome of the push, allowing the 'LLM (Online Adaptation)' to refine its friction estimate and update the plan for the subsequent pick, thereby enabling success.

Second, this task powerfully illustrates the benefit of the 'Memory Unit'. The performance of our full model without memory was respectable (4/10), but the

inclusion of the memory module boosted the success rate significantly to 8/10. This shows that after just a single successful completion, the system can store the entire successful interaction context (the refined parameters, cost function, and contact strategy). When faced with a similar task configuration, it retrieves this proven strategy, providing the MPPI planner with a superior initialization that leads to more robust and efficient execution. This result demonstrates that Contact-VLA can learn from its experiences, and it highlights a clear path towards few-shot performance improvements where the system becomes more adept as it gathers successful episodes.

5.4.3.5 Explainability and Automated Failure Recovery

A key advantage of our neuro-symbolic design is its inherent explainability, particularly during failure recovery. Unlike opaque end-to-end models, Contact-VLA can articulate *why* it failed and *what* it is doing to correct its plan. We demonstrate this with a scenario where the "Flip with Wall" task persistently fails, triggering the Outer Loop.

Instead of just outputting a new set of parameters, the 'LLM (Online Adaptation)' module provides a full natural language diagnosis of the failure and a detailed log of the corrective actions it is taking. As shown by the model's direct output in Appendix A.3, the LLM correctly identified that its initial cost function failed to sufficiently penalize incorrect orientations and did not adequately reward wall contact. It then proceeded to adjust the specific weights of the cost function to remedy this. This capability to "think out loud" is a critical feature for building trust and diagnosing failures in complex robotic systems, providing a level of transparency that is simply not possible with black-box policies.

Chapter 6

Discussions

Our experimental results validate the core hypotheses of Contact-VLA, demonstrating that a modular, neuro-symbolic architecture can overcome the fundamental limitations of end-to-end models in complex, contact-rich manipulation. We discuss the key implications of our findings below.

6.1 The Synergy of Grounded Reasoning and Reactive Control

Our experiments reveal a clear synergy between high-level reasoning and low-level reactive control. The performance gap between Contact-VLA and end-to-end baselines like OpenVLA, especially in tasks requiring non-trivial strategies (e.g., T6: Flip with Wall), highlights the brittleness of purely imitative policies. These models fail because their training data lacks examples of using the environment as a tool. In contrast, Contact-VLA's success stems from its ability to reason: the LLM formulates an explicit optimization problem (J_0) that defines the task's success conditions, while the MPPI controller finds a physically plausible solution. This makes the system's intent transparent and grounds abstract reasoning in a formal control framework, a more robust approach than conditioning a black-box

policy as is done in works like ECoT [17].

6.2 Online Adaptation as an Alternative to Large-Scale Tactile Datasets

A significant implication of our work is a path away from the data-hungry paradigm of modern robotics. State-of-the-art methods like ForceVLA [7] achieve impressive results by incorporating tactile feedback, but this requires creating massive, specialized demonstration datasets. Our results demonstrate a more data-efficient alternative. Contact-VLA also uses real-time force feedback, but its role is redefined: it serves as a signal for *online adaptation*, not offline imitation. The success of our outer feedback loop, particularly in tasks where initial parameter estimates were deliberately inaccurate, proves that the LLM can diagnose physical failures and refine its world model on the fly. This ability to learn from direct interaction significantly lowers the barrier to entry for creating sophisticated, contact-aware robots without relying on pre-collected, large-scale tactile data.

6.3 Zero-Shot Planning and the Path to Lifelong Learning

Perhaps the most significant result is Contact-VLA's ability to perform zero-shot planning for novel tasks. This capability stems from its core design: instead of learning *how to act*, it leverages the pre-existing knowledge of foundation models to reason about *how to plan*. By generating a cost function from a single image and a language command, the system dynamically tackles new objectives. The consistent performance boost provided by the 'Memory Unit' in our ablation studies further points towards a lifelong learning capability. By retrieving and bootstrapping successful strategies, the system becomes more efficient and robust

over time, a contrast to the static nature of policies that require extensive fine-tuning or retraining to adapt [12, 13].

6.4 Internal World Model for Planning and Adaptation

The core of Contact-VLA's reasoning capability lies in its use of an explicit internal world model. This model is a simulated, physics-aware representation of the real world, constructed and parameterized by the VLM's output, θ. This internal model is not merely a passive environment for the planner; it serves two critical functions in our architecture:

1. **A Sandbox for Mental Rehearsal:** The MPPI controller leverages this world model to perform thousands of parallel "mental rehearsals" (rollouts) of potential action sequences. This allows the agent to anticipate the physical consequences of its actions—such as contact forces and object displacement—before executing them in the real world, enabling proactive and intelligent decision-making.

2. **An Adaptable Belief State:** This world model represents the agent's current "belief" about the physical properties of its environment. Crucially, this belief is not static. The outer feedback loop, driven by the 'LLM (Online Adaptation)' module, directly refines this model by updating its parameters (θ) based on discrepancies between predicted and observed outcomes. This turns the world model into a dynamic, adaptable belief state that is continuously improved through physical interaction, connecting our work to principles of online system identification and model-based reinforcement learning.

Chapter 7

Limitations and Future Work

Despite its promising results, Contact-VLA has several limitations that define clear directions for future research.

Fidelity of the Internal World Model: The entire framework is predicated on the quality of the simulated physical world constructed by the VLM. This dependency is a significant limitation; the performance of the MPPI planner is directly correlated with how accurately this internal model reflects real-world physics. We can think of this simulated environment as the robot's "mind," where it mentally rehearses actions before execution. The better this mental model, the more seamlessly the robot can translate its plans into successful real-world actions. Currently, the system is vulnerable to VLM "hallucinations" or inaccuracies—misjudging an object's material could lead to a grossly incorrect estimate for mass or friction. While our feedback loop is designed to correct for such errors, a sufficiently poor initialization could prevent the planner from converging. Future work should focus on creating higher-fidelity simulated worlds, potentially by learning residual dynamics models online to capture unmodeled effects (e.g., non-rigid dynamics, complex friction) and better bridge the sim-to-real gap.

Computational Latency: The sequential nature of our pipeline—VLM perception, LLM reasoning, and MPPI planning—introduces significant computational latency. The inference time of large foundation models, combined with the iterative nature of the MPPI planner, currently limits the framework's applicability to highly dynamic tasks that require millisecond-level reaction times [1, 3]. Future work should explore techniques such as model distillation and quantization for the foundation models. Furthermore, the performance of the MPPI planner could be significantly enhanced by incorporating more advanced sampling strategies, such as those explored in recent literature [26], to increase computational efficiency and improve the robot's real-time adaptation capabilities.

Reliance on Generalist LLMs for Strategy Formulation: A core component of our system is the LLM's ability to generate a viable cost function for the MPPI planner. We currently use a generalist, off-the-shelf LLM (GPT-4o), which performs remarkably well but is not specialized for robotics or physics-based planning. The quality and coherence of the generated cost function are not guaranteed for entirely novel or abstract tasks that fall outside the LLM's vast but general pre-training data. A promising direction for future work is to fine-tune or develop LLMs specifically for the task of generating optimization objectives for robotic control, potentially leading to more robust and efficient strategy formulation.

Dependence on External Object Pose and Size Estimation: A fundamental assumption of our framework is that the system has access to reliable, real-time pose and size information for all interactable objects. This allows our VLM to bypass the challenging step of estimating object states directly from raw images, setting our approach apart from traditional VLA methods. While this design simplifies scene representation and reduces perceptual noise, it introduces a dependency on external tracking systems. In scenarios where accurate object tracking infrastructure is unavailable, noisy, or costly to deploy, the applicability of our method becomes limited. Furthermore, this assumption may hinder the system's scalability to more unstructured environments, such as outdoor or cluttered household settings, where object tracking is unreliable. Future work should aim to relax this dependency, for example by (i) integrating image-based state

estimation as a fallback mechanism, (ii) fusing external tracking with learned perception modules for robustness, or (iii) developing self-supervised approaches that can infer object identities and dynamics from multimodal sensory input. Such extensions would broaden the framework's utility across diverse environments and reduce reliance on strong external assumptions.

Chapter 8

Conclusion

In this paper, we introduced **Contact-VLA**, a novel framework that addresses the challenges of zero-shot, contact-rich manipulation. Our approach departs from conventional end-to-end paradigms by integrating foundation models with a reactive controller. Experiments on challenging tasks demonstrate that this modular, synergistic design enables the system to adapt to unseen scenarios without prior demonstrations, significantly enhancing both performance and explainability over monolithic approaches. We believe this hybrid paradigm—coupling large-scale, pre-trained knowledge with rigorous real-time control—is a promising direction for creating more capable and physically intelligent robotic agents.

Bibliography

[1] R. Firoozi, J. Tucker, S. Tian, A. Majumdar, J. Sun, W. Liu, Y. Zhu, S. Song, A. Kapoor, K. Hausman, *et al.*, "Foundation models in robotics: Applications, challenges, and the future," *The International Journal of Robotics Research*, vol. 44, no. 5, pp. 701–739, 2025.

[2] M. Tayyab Khan and A. Waheed, "Foundation model driven robotics: A comprehensive review," *arXiv e-prints*, pp. arXiv–2507, 2025.

[3] Y. Ma, Z. Song, Y. Zhuang, J. Hao, and I. King, "A survey on vision-language-action models for embodied ai," *arXiv preprint arXiv:2405.14093*, 2024.

[4] Y. Zhong, F. Bai, S. Cai, X. Huang, Z. Chen, X. Zhang, Y. Wang, S. Guo, T. Guan, K. N. Lui, *et al.*, "A survey on vision-language-action models: An action tokenization perspective," *arXiv preprint arXiv:2507.01925*, 2025.

[5] R. Sapkota, Y. Cao, K. I. Roumeliotis, and M. Karkee, "Vision-language-action models: Concepts, progress, applications and challenges," *arXiv preprint arXiv:2505.04769*, 2025.

[6] P. Hao, C. Zhang, D. Li, X. Cao, X. Hao, S. Cui, and S. Wang, "Tla: Tactile-language-action model for contact-rich manipulation," *arXiv preprint arXiv:2503.08548*, 2025.

[7] J. Yu, H. Liu, Q. Yu, J. Ren, C. Hao, H. Ding, G. Huang, G. Huang, Y. Song, P. Cai, *et al.*, "Forcevla: Enhancing vla models with a force-aware moe for contact-rich manipulation," *arXiv preprint arXiv:2505.22159*, 2025.

[8] H. Xue, J. Ren, W. Chen, G. Zhang, Y. Fang, G. Gu, H. Xu, and C. Lu, "Reactive diffusion policy: Slow-fast visual-tactile policy learning for contact-rich manipulation," *arXiv preprint arXiv:2503.02881*, 2025.

[9] J. R. Flanagan, M. C. Bowman, and R. S. Johansson, "Control strategies in object manipulation tasks," *Current opinion in neurobiology*, vol. 16, no. 6, pp. 650–659, 2006.

[10] R. S. Johansson and J. R. Flanagan, "Coding and use of tactile signals from the fingertips in object manipulation tasks," *Nature Reviews Neuroscience*, vol. 10, no. 5, pp. 345–359, 2009.

[11] S. S. Kim, M. Gomez-Ramirez, P. H. Thakur, and S. S. Hsiao, "Multimodal interactions between proprioceptive and cutaneous signals in primary somatosensory cortex," *Neuron*, vol. 86, no. 2, pp. 555–566, 2015.

[12] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. P. Foster, P. R. Sanketi, Q. Vuong, T. Kollar, and C. Finn, "OpenVLA: An open-source vision-language-action model," in *Conference on Robot Learning (CoRL)*, vol. 270, pp. 2098–2120, PMLR, 2025.

[13] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, *et al.*, "π_0: A vision-language-action flow model for general robot control," *arXiv preprint arXiv:2410.24164*, 2024.

[14] A. O'Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. 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)*, pp. 6892–6903, IEEE, 2024.

[15] C.-P. Huang, Y.-H. Wu, M.-H. Chen, Y.-C. F. Wang, and F.-E. Yang, "Thinkact: Vision-language-action reasoning via reinforced visual latent planning," *arXiv preprint arXiv:2507.16815*, 2025.

[16] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, P. Sermanet, N. Brown, T. Jackson,

L. Luu, S. Levine, K. Hausman, and B. Ichter, "Inner monologue: Embodied reasoning through planning with language models," in *Conference on Robot Learning (CoRL)*, 2022.

[17] M. Zawalski, W. Chen, K. Pertsch, O. Mees, C. Finn, and S. Levine, "Robotic control via embodied chain-of-thought reasoning," *arXiv preprint arXiv:2407.08693*, 2024.

[18] J. Lee, J. Duan, H. Fang, Y. Deng, S. Liu, B. Li, B. Fang, J. Zhang, Y. R. Wang, S. Lee, *et al.*, "Molmoact: Action reasoning models that can reason in space," *arXiv preprint arXiv:2508.07917*, 2025.

[19] F. Lin, R. Nai, Y. Hu, J. You, J. Zhao, and Y. Gao, "Onetwovla: A unified vision-language-action model with adaptive reasoning," *arXiv preprint arXiv:2505.11917*, 2025.

[20] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "React: Synergizing reasoning and acting in language models," in *International Conference on Learning Representations (ICLR)*, 2023.

[21] Y. Ling, K. Owalekar, O. Adesanya, E. Bıyık, and D. Seita, "Impact: Intelligent motion planning with acceptable contact trajectories via vision-language models," *arXiv preprint arXiv:2503.10110*, 2025.

[22] W. Zhao, J. Chen, Z. Meng, D. Mao, R. Song, and W. Zhang, "VLMPC: Vision-language model predictive control for robotic manipulation," in *Robotics: Science and Systems (RSS)*, 2024.

[23] J. Bi, K. Y. Ma, C. Hao, M. Z. Shou, and H. Soh, "Vla-touch: Enhancing vision-language-action models with dual-level tactile feedback," *arXiv preprint arXiv:2507.17294*, 2025.

[24] J. J. Liu, Y. Li, K. Shaw, T. Tao, R. Salakhutdinov, and D. Pathak, "Factr: Force-attending curriculum training for contact-rich policy learning," *arXiv preprint arXiv:2502.17432*, 2025.

[25] G. Williams, A. Aldrich, and E. A. Theodorou, "Model predictive path integral control: From theory to parallel computation," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 344–357, 2017.

[26] C. Pezzato, C. Salmi, E. Trevisan, M. Spahn, J. Alonso-Mora, and C. H. Corbato, "Sampling-based model predictive control leveraging parallelizable physics simulations," *IEEE Robotics and Automation Letters*, 2025.

Appendix A

Supplementary Information

A.1 Data Availability

The curated dataset used in this study is open for academic purposes and will be made available upon reasonable request to the author.

A.2 Code Availability

The source code and detailed instructions for deploying the system are available at *https://sites.google.com/view/contactvla*.

A.3 Example of LLM-driven Failure Analysis and Refinement

Below is the direct natural language output from the 'LLM (Online Adaptation)' module after being triggered by persistent failures in the "Flip with Wall" task.

The LLM was provided with the episode history (E_t) and prompted to diagnose the failure and propose a refined cost function.

```
--- Outer Loop Triggered:  Analyzing failure and refining
strategy...  ---
```

LLM Explanation: The previous cost function was failing because it did not sufficiently prioritize the critical aspects of the task, such as ensuring the box is flipped onto its side and maintaining contact with the wall. The rewards and penalties were not balanced effectively to guide the robot towards achieving the task goal. Specifically, the orientation error was not heavily penalized, and the rewards for contact and elevation were not strong enough to drive the desired behavior.

In the new version, the following improvements have been made:

1. **Increased Penalty for Orientation Error**: The weight on the angle error has been increased to emphasize the importance of achieving the correct orientation.
2. **Stronger Rewards for Contact and Elevation**: The rewards for maintaining contact with the wall and elevating the box have been increased, encouraging the robot to push the box against the wall and attempt a flip.
3. **Adjusted Proximity Penalties**: The penalties for the end-effector's distance from the box and the box's distance from the wall have been adjusted to better guide the robot's actions.
4. **Relaxed Orientation Threshold**: The threshold for the orientation reward has been slightly relaxed to allow for some tolerance in achieving the target orientation, which can help in stabilizing the box on its side.

```
These changes aim to create a more balanced and effective cost
function that better aligns with the task objectives.  Simulated
environment's cost function has been updated.
```

Figure A.1: Direct output from the LLM during an online adaptation cycle. The model provides a clear, human-readable diagnosis of the previous plan's failure and details the specific changes made to the cost function for the next attempt.

A.4 Example of an LLM-Generated Cost Function

Figure A.1 shows a direct example of a Python cost function generated by the 'LLM (Task Formulation)' module for the "Wipe Shoe" task (T7). This code serves as the running cost, $q(x_t, u_t)$, within the MPPI objective (Eq. 4.4). The function translates the high-level semantic goal—"wipe the shoe gently without moving it"—into a computable objective that the planner can optimize.

The function skillfully balances multiple, often competing, objectives:

- **Target Reaching:** It encourages the end-effector to move towards a selected contact point on a stain, which is sampled based on the LLM's high-level contact strategy, C_0.
- **Task Progress Reward:** It provides a significant reward (a large negative cost) for each stain that is successfully "removed" (i.e., wiped), directly incentivizing the primary goal.
- **Stability Constraint:** It heavily penalizes any displacement or rotation of the shoe from its initial pose, enforcing the "gently" constraint.
- **Efficiency Penalty:** A minor cost is added for each timestep to encourage the planner to find an efficient solution.

This example demonstrates how Contact-VLA grounds abstract language instructions into a structured, mathematical objective that a model-based planner can directly use to generate physically intelligent behavior.

```
# Implementation of the running cost q(x,u) for the "Wipe Shoe"
    task,
# as generated by the LLM (Task Formulation) module.

def state_cost(self):
    # Get the current 3D position of the end-effector (EE).
    ee_pos = np.array(p.getLinkState(self.panda,
```

```
                                self.grasp_link,
                                computeForwardKinematics=True,
                               physicsClientId=self.cid)[0])

    cost = 0.0

    # 1. Guide the EE towards the target contact point
    # 'best_contact_point' is sampled from the LLM's contact strategy (C_0).
    if hasattr(self, 'best_contact_point') and self.best_contact_point is not None:
        cost += 10.0 * np.linalg.norm(ee_pos - self.best_contact_point)

    # 2. Reward successful wiping of stains
    # A large negative cost (a reward) is given for making progress.
    cost -= 50.0 * len(self._last_removed)

    # 3. Penalize inefficiency
    # A small cost is added at each step to encourage shorter solutions.
    cost += 0.01 * self.current_step

    # 4. Penalize moving the shoe (Stability Constraint)
    pos, ori = p.getBasePositionAndOrientation(self.shoe,
                                               physicsClientId=self.cid)

    # Calculate displacement from the initial position.
    dp = np.linalg.norm(np.array(pos) - np.array(self.shoe_init_pos))

    # Calculate angular rotation from the initial orientation.
    q_diff = p.getDifferenceQuaternion(ori, self.shoe_init_ori)
    w = max(-1.0, min(1.0, q_diff[3]))
    angle = 2.0 * math.acos(w)

    # Add a heavily weighted penalty for any shoe movement.
    weight = 10.0
```

```
    cost += weight * (dp + angle)

    return cost
```

Listing A.1: A Python code snippet for the MPPI running cost function, as generated by our LLM for the "Wipe Shoe" task. The code is syntax-highlighted for clarity to break down how the LLM translates semantic goals into mathematical terms.

A.5 VLM Prompting for Physical Parameter Estimation

This section details the query sent to the VLM (GPT-4o) to perform object identification and physical parameter estimation, which generates the initial world model parameters, θ. The process is designed to translate unstructured visual and language data into a structured, machine-readable format that can initialize our physics simulator.

The prompt, shown in Figure A.2, provides the model with all available context: the full scene image (implicitly via the multimodal model), the natural language task description, and a JSON object containing a list of unidentified objects. Each object in the list is populated with its known 6-DoF pose and size, which we assume are provided by an external tracking system.

The VLM's task is twofold:

1. **Grounding:** It must identify each object and assign a semantic 'label' (e.g., "object_1" is the "cutting_board").
2. **Estimation:** It must use its vast, pre-trained knowledge about the real world to estimate the physical properties of the identified object, such as its 'mass_kg' and 'friction_coeff'.

By constraining the output to be only the completed JSON object, we ensure the response is directly parsable by our system, forming the initial 'Internal World Model'.

```
# Prompt sent to GPT-4o to act as the VLM
You are a robotics expert with a deep understanding of physics.
Your task is to identify objects in a scene and estimate their
    physical properties for a simulation.

Task Description: "Push the cutting board until the handle is off
     the table, then pick it up."

I have segmented the following objects from an image. I know
    their poses and sizes, but not what they are or their physical
     properties. Please identify each object and provide your best
     estimate for its mass (in kg) and friction coefficient.

Objects Data:
{
  "object_1": {
    "pose": [0.5, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0],
    "size": [0.4, 0.25, 0.02],
    "label": "?",
    "mass_kg": "?",
    "friction_coeff": "?"
  },
  "object_2": {
    "pose": [0.75, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0],
    "size": [0.03, 0.1, 0.02],
    "label": "?",
    "mass_kg": "?",
    "friction_coeff": "?"
  }
}

Respond ONLY with the completed JSON object.

# --------------------------------------------------
# Example VLM JSON Response for the task above:
{
```

```
  "object_1": {
    "pose": [0.5, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0],
    "size": [0.4, 0.25, 0.02],
    "label": "cutting_board",
    "mass_kg": 0.4,
    "friction_coeff": 0.4
  },
  "object_2": {
    "pose": [0.75, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0],
    "size": [0.03, 0.1, 0.02],
    "label": "handle",
    "mass_kg": 0.05,
    "friction_coeff": 0.6
  }
}
```

Listing A.2: The structured prompt sent to the multimodal model (GPT-4o) to act as our VLM, along with a representative JSON response for the "Push and Pick Cutting Board" task. The model is tasked with filling in the unknown values ("?").

A.6 LLM Prompting for Online Adaptation

The 'LLM (Online Adaptation)' module is triggered when the system detects persistent failures. Unlike the initial task formulation, the adaptation prompts are designed to be diagnostic, providing the LLM with a history of recent failed interactions to inform its corrections. The module employs two distinct prompting strategies depending on the type of refinement needed: strategy refinement (correcting the plan) and world model correction (correcting physical parameters).

A.6.1 Strategy Refinement Prompt

When the logic of the plan itself is suspected to be flawed, the system asks the LLM to act as a robotics programmer and rewrite the core 'state_cost' function. As shown in Figure A.3, the prompt provides the LLM with the task, the environment's attributes, the recent execution history (e.g., last 5 steps of object positions and resulting costs), and critically, the **current, failing source code** of the cost function. It is then instructed to return a corrected code block and a natural language explanation of its changes. This process is the source of the explainable failure recovery analysis presented in the main text.

```
# Python function that builds the prompt for cost function refinement.
def ask_state_cost_fn(self, task: str, history: list, env: BoxPushEnv):
    # Grab the current, failing source code of the cost function.
    try:
        current_src = inspect.getsource(env.state_cost.__func__)
    except (OSError, IOError):
        current_src = env.state_cost_src

    # --- The prompt sent to the LLM ---
    prompt = (
        f"Task: {task}\n"
        "You have full freedom to compute any cost that helps '{task}'.\n"
        "History of last 5 steps (box_pos, resulting_cost):\n"
        f"<RECENT_EXECUTION_HISTORY>\n\n"
        "Please output TWO things, separated by '---':\n"
        "1) Python code for a method `def state_cost(self): ...` (indented block only)\n"
        "2) A brief explanation why the previous cost was failing and how the new one addresses it.\n\n"
        f"Here is the CURRENT implementation that is failing:\n"
        f"{current_src}"
    )

    # Query the LLM and parse the response (code_str, explanation)
```

```
    resp = openai.chat.completions.create(...)
    content = resp.choices[0].message.content
    code_str, explanation = content.split("---",1)
    return code_str.strip(), explanation.strip()

```

Listing A.3: The Python function and prompt structure used for Strategy Refinement. The LLM is given the failing code and recent history to rewrite the cost function.

A.6.2 World Model Correction Prompt

If the strategy is believed to be correct but the physical outcomes do not match the simulation (e.g., the robot pushes but the object barely moves), the system asks the LLM to act as a physicist and refine the object parameters. The prompt, detailed in Figure A.4, provides the LLM with the agent's current belief about the physical parameters (mass, friction) and the recent execution history. The LLM's task is to analyze the discrepancy between actions and outcomes in the history and propose corrected physical parameters, returning them in a machine-parsable JSON format.

```
# Python function that builds the prompt for physical parameter refinement.
def ask_params_refinement(self, history: list, object_params: list):

    # --- The prompt sent to the LLM ---
    prompt = (
        "We have the following object parameters:\n"
        f"{json.dumps(object_params, indent=2)}\n\n"
        "Recent execution history (last 5 steps):\n"
        f"<RECENT_EXECUTION_HISTORY>\n\n"
        "Based on this, propose refined values for mass and friction_coef."
        "Return ONLY a JSON array of objects with keys "
        "`label`, `mass`, and `friction`.\n"
        "Example output:\n"
```

```
        "[\n"
        "  {\"label\":\"cutting_board\",\"mass\":0.45,\"friction\":0.35}\n"
        "]"
    )

    # Query the LLM and parse the JSON response
    resp = openai.chat.completions.create(...)
    text = resp.choices[0].message.content.strip()
    try:
        return json.loads(text)
    except json.JSONDecodeError:
        # Fallback to extract JSON if LLM adds extra text
        ...

```

Listing A.4: The Python function and prompt structure for World Model Correction. The LLM analyzes the recent history to refine its belief about the object's physical properties.

A.7 LLM-driven Contact Strategy Generation

A key challenge in contact-rich manipulation is determining precisely *where* to make contact with an object. Uniformly sampling an object's entire surface is computationally inefficient and unlikely to yield strategically useful points. To overcome this, Contact-VLA leverages the LLM's commonsense physical reasoning to intelligently narrow the search space. This is achieved through a two-stage process, detailed in Figure A.5, which translates a high-level task into a concrete set of candidate contact points (C_0).

Stage 1: Strategic Region Proposal. First, the 'LLM (Task Formulation)' module queries a foundation model (GPT-4o) with a structured prompt that includes the task description and the VLM-estimated object parameters. The prompt, shown in Figure A.6, explicitly instructs the LLM to act as a robotics

expert and identify 1-3 small, promising surface regions for contact on each relevant object. The LLM is constrained to return this information in a structured JSON format, specifying each region's 3D center, surface normal, radius (extent), and the desired number of samples. For the "Push and Pick Cutting Board" task, the LLM correctly identifies that pushing should occur on the board's main surface, while grasping should target the handle.

Stage 2: Geometric Candidate Sampling. Second, the structured JSON response from the LLM is passed to a geometric sampling function ('sample-points-in-region'). This function translates the LLM's abstract region definitions into a dense set of 3D point coordinates. For each region, it defines a 2D disk in 3D space oriented by the provided center and normal vectors. It then samples the requested number of points within this disk, generating the final set of candidate contact points, C_0, which are then used to bias the MPPI planner's exploration as described in the main text.

```python
# Stage 1: Query the LLM to propose strategic contact regions.
def ask_region_strategy(object_params, task_desc):
    # The prompt is shown in Figure~\ref{fig:contact_prompt}
    prompt = f"..."
    resp = client.chat.completions.create(...)
    return resp.choices[0].message.content

# Stage 2: Sample concrete 3D points from an LLM-defined region.
def sample_points_in_region(region):
    c = np.array(region["center"], dtype=float)
    n = np.array(region["normal"], dtype=float)
    r = float(region["extent"])
    k = int(region["num_samples"])

    # Create two orthonormal tangent vectors to define the plane
   of the disk.
    if abs(n[2]) < 0.9:
        axis = np.array([0.0, 0.0, 1.0])
    else:
        axis = np.array([0.0, 1.0, 0.0])
    t1 = np.cross(n, axis)
    t1 /= np.linalg.norm(t1)
```

```
    t2 = np.cross(n, t1)
    t2 /= np.linalg.norm(t2)

    # Sample k points within the 2D disk defined by the tangents.
    pts = []
    for _ in range(k):
        rho   = np.sqrt(np.random.rand()) * r  # Uniform sampling in a disk
        theta = np.random.rand() * 2 * np.pi
        offset = rho * np.cos(theta) * t1 + rho * np.sin(theta) * t2
        pts.append((c + offset).tolist())
    return pts

```

Listing A.5: The two-stage Python implementation for generating the contact strategy C_0. The 'ask-region-strategy' function queries the LLM for high-level guidance, and 'sample-points-in-region' translates that guidance into concrete 3D coordinates.

```
# The prompt sent to the LLM (Task Formulation) module:
Task: Push the cutting board until the handle is off the table, then pick it up.

You have objects with parameters:
{
  "board": { ... },
  "handle": { ... }
}

Instead of uniform sampling, identify for each object 1-3 small surface regions
where contact is most promising.
For each region, return:
  - center: [x,y,z]
  - normal: unit surface normal [nx,ny,nz]
  - extent: radius in meters around center
  - num_samples: how many points to sample there

Respond ONLY a JSON mapping each label to its "regions" array.
```

```
19
20 # Example LLM JSON Response:
21 {
22   "object": {
23     "regions": [
24       {
25         "center": [0.5, 0.0, 0.01],
26         "normal": [0, 0, 1],
27         "extent": 0.1,
28         "num_samples": 30
29       }
30     ]
31   }
32
```

Listing A.6: The structured prompt and an example JSON response for the "Push and Pick Cutting Board" task. The prompt constrains the LLM to provide a structured, machine-readable output.