

Politecnico di Torino



UNIVERSITY OF CAMBRIDGE

Politecnico di Torino

Double Master's Degree Politecnico di Torino and École Nationale Supérieure d'Informatique et de Mathématiques Appliquées

Data Science and Engineering - Artificial Intelligence

A.Y. 2025/2026

Graduation Session October 2026

PhySVLA: Physics-Informed Safety for Vision-Language-Action based manipulation

This thesis work was carried out during the visiting research period at the University of Cambridge.

Supervisors:
Prof. Giuseppe Averta
Prof. Olaf Wysocki
Prof. Guangming Wang
Prof. Xavier Alameda-Pineda

Candidate:
Claudio Camolese

Abstract

Vision–Language–Action (VLA) models provide general-purpose robotic policies capable of interpreting natural language instructions and executing diverse manipulation tasks. However, task success does not imply safe execution: because collision avoidance is generally absent from their training objective, VLAs can produce semantically correct actions that collide with nearby objects.

Existing methods either inject geometry during training, requiring access to and adaptation of the policy, or modify inference through policy-internal gradients or simplified obstacle models. These requirements limit their modularity and scene generality.

We introduce *PhySVLA*, a task-aware, plug-and-play safety framework that augments frozen VLA policies with potential field based collision avoidance. At its core, PhySVLA learns a differentiable neural potential field over the robot configuration space. The field is trained exclusively from the robot geometry and forward kinematics, without task demonstrations or scene-specific obstacle data, allowing it to generalize zero-shot to previously unseen environments. At inference time, multi-view RGB-D observations are fused into a task-aware point cloud excluding objects and supports requiring intentional contact. Querying the learned field against these obstacle points produces a scene-conditioned safety potential: configurations with low clearance generate repulsive gradients that indicate how the robot should move to increase its clearance from nearby obstacles.

While the VLA provides the task-directed motion, the potential field locally guides the robot through cluttered regions. A Control Barrier Function projection combines these two signals by minimally correcting unsafe commands along the directions induced by the field, while leaving nominal actions unchanged whenever the safety constraints are satisfied. In this way, PhySVLA preserves the semantic and manipulation capabilities of the original policy while enabling local navigation around unseen obstacles without retraining the VLA.

PhySVLA is evaluated with both OpenVLA-OFT and $\pi_{0.5}$ against nine VLA, world-model, and geometry-aware baseline configurations on 1,200 SafeLIBERO rollouts per configuration across the Spatial, Object, and Goal suites. Relative to frozen OpenVLA-OFT, PhySVLA improves Collision Avoidance Rate (CAR) by 316.8% and Safe Success Rate (SSR) by 279.5%. Relative to frozen $\pi_{0.5}$, it improves CAR by 325.2% and SSR by 266.0%.

These results demonstrate that explicit execution-time geometric reasoning can transfer across distinct VLA backbones and substantially reduce collision risk without policy retraining, providing a practical bridge between the semantic capabilities of generalist VLAs and reusable, scene-general safety correction.

Acknowledgements

First and foremost, I would like to express my heartfelt gratitude to my supervisors for guiding and supporting me throughout this journey.
I am sincerely grateful to Prof. Alameda-Pineda for his support and guidance during the development of this thesis.
I would also like to express my deepest gratitude to Prof. Averta, who has been an exceptional mentor throughout this experience. His invaluable advice, constant availability, and willingness to listen have meant a great deal to me. His support has gone far beyond the scope of this thesis, and I am certain that his mentorship will continue to have a profound influence on my academic and professional path in the years to come.
Finally, I would like to extend a very special thank you to Prof. Olaf Wysocki, Prof. Guangming Wang, and the entire Computer Vision for Digital Twins (CV4DT) research group for giving me the opportunity to become part of such a prestigious academic environment as the University of Cambridge.
My gratitude to them goes far beyond the academic experience. From my very first days, they welcomed me with remarkable kindness and made me feel genuinely part of the group. I am deeply grateful not only for everything I learned from them, but above all for the extraordinary people they have been throughout this experience. I truly hope that our paths will cross again and that we will have the opportunity to work together in the future. For everything you have taught me, for the trust you placed in me, and for the memories we shared along the way, I thank you from the bottom of my heart.

Dedico ogni traguardo che ho raggiunto e che raggiungerò nella mia vita alla mia famiglia, senza la quale nulla di tutto questo sarebbe stato possibile.
Ringrazio mia sorella per essermi sempre stata accanto, soprattutto prima di ogni esame, e per essere stata, durante tutto questo percorso, non solo una sorella, ma anche un'amica su cui poter sempre contare.
Ringrazio mia madre per aver trovato sempre il tempo di ascoltarmi, per aver saputo incoraggiarmi nei momenti più difficili e per essere sempre riuscita a farmi sorridere.

Ringrazio mio padre per essere stato un punto di riferimento e un mentore nelle scelte più difficili, e per essersi sempre messo a disposizione ogni volta che ne ho avuto bisogno.

Ringrazio, infine, la mia famiglia per non avermi mai fatto mancare nulla, per avermi sostenuto nei momenti in cui ne avevo più bisogno e, soprattutto, per aver sempre creduto in me.
Spero, con questo traguardo e con quelli che verranno, di riuscire a restituirvi almeno una parte di tutto ciò che voi avete dato a me.

Table of Contents

List of Figures VIII

1 Introduction 1
- 1.1 Problem Definition . . . 1
- 1.2 Motivation . . . 4
- 1.3 Contributions . . . 7

2 Related Work 8
- 2.1 Scope and intervention-point taxonomy . . . 8
- 2.2 VLA backbones and the geometry gap . . . 10
  - 2.2.1 Compared VLA backbones . . . 11
  - 2.2.2 Task success is not a safety specification . . . 12
- 2.3 Post-policy action correction . . . 12
  - 2.3.1 Control barrier functions and minimal intervention . . . 12
  - 2.3.2 VLSA: Vision-Language-Action Models with Plug-and-Play Safety Constraint Layer . . . 13
  - 2.3.3 Superquadric-Based Safety Filtering . . . 15
- 2.4 Guidance inside generative inference . . . 16
- 2.5 Geometry incorporated during training . . . 18
  - 2.5.1 Explicit feasibility supervision . . . 18
  - 2.5.2 Geometric representation alignment: ROCKET . . . 19
  - 2.5.3 World action model: Fast-WAM . . . 20
  - 2.5.4 Geometric backbone and future prediction: GAM . . . 20
  - 2.5.5 Comparison of training-time strategies . . . 22
- 2.6 Differentiable distance fields . . . 22
  - 2.6.1 Scene and workspace distance fields . . . 23
  - 2.6.2 Robot-conditioned and configuration-space fields . . . 24
  - 2.6.3 Implications for learned safety layers . . . 26
- 2.7 Evaluation benchmarks . . . 26

2.7.1 LIBERO . . . 26
2.7.2 SafeLIBERO . . . 26
2.8 Positioning of this thesis . . . 28

3 Theoretical Background 29
3.1 Large Language Models . . . 32
3.2 Vision-Language Models . . . 33
3.3 Vision-Language-Action Models . . . 34
3.3.1 OpenVLA: An Open-Source Vision-Language-Action Model 36
3.3.2 $\pi_{0.5}$: a Vision-Language-Action Model with Open-World Generalization . . . 37
3.3.3 Final considerations . . . 39

4 Methodology 40
4.1 Design objectives and scope of generalization . . . 42
4.2 Making the method scene-free and zero-shot . . . 42
4.2.1 Sampling the configuration space . . . 43
4.2.2 Workspace-point sampling . . . 44
4.2.3 Geometric labels . . . 46
4.3 Training phase . . . 48
4.3.1 PhySVLA: model architecture overview . . . 48
4.3.2 Barrier construction . . . 50
4.3.3 Training objectives . . . 51
4.4 Inference phase . . . 53
4.4.1 Online obstacle point cloud . . . 55
4.4.2 Field query and constraint selection . . . 55
4.4.3 Mapping the gradient to the VLA action space . . . 57
4.4.4 CBF action correction . . . 58
4.4.5 Payload-aware collision avoidance . . . 60

5 Experiments and Results 62
5.1 Experimental protocol . . . 62
5.2 Metrics . . . 63
5.3 Aggregate comparison . . . 64
5.4 Results by SafeLIBERO suite . . . 66
5.4.1 SafeLIBERO Spatial . . . 66
5.4.2 SafeLIBERO Object . . . 66
5.4.3 SafeLIBERO Goal . . . 66
5.5 Results by safety level . . . 67

5.6 Paired effect of action correction 69

6 Discussion 71
6.1 Limitations 71
6.1.1 Out-of-distribution state 72
6.1.2 Hyperparameter selection 73
6.1.3 Grasping phase 74
6.1.4 Access to the robot FK 75
6.2 Ablation Studies 76
6.2.1 VLA with joint-space output 76
6.2.2 VLA with a joint-space controller 78

7 Conclusions and Future Work 80
7.1 Conclusions 80
7.2 Future Work 81
7.2.1 Long-horizon action correction 81
7.2.2 Real-world and embodiment transfer 81
7.3 Final Remarks 82

Bibliography 83

List of Figures

1.1 Examples of scenes involving VLA deployment. Source: Physical Intelligence [5]. 5

2.1 Intervention-point taxonomy for geometry-aware robot policies. The three families differ in policy access, modularity, and the semantics of the safety signal. 9
2.2 Control barrier function method applied to drone navigation. 13
2.3 Workflow of the AEGIS safety layer [6]. 14
2.4 VLSA quadratic programming for obstacle avoidance with the robot end-effector [6]. 14
2.5 Overview of the SQ-CBF [7] framework for geometry-aware safety filtering. 15
2.6 Simulation (left) and real-world experiment (right) on the Unitree Go2 quadruped robot using the Safe-SAGE model [8]. 16
2.7 Pipeline overview of the OmniGuide [9] action generation method. . 16
2.8 VLA training with explicit physical feasibility supervision [10]. . . . 18
2.9 Overview of the ROCKET layer [11]. On the left, a sample from LIBERO to showcase ROCKET's performance. On the right, the outputs from VGGT layers are used to directly predict the depth map, demonstrating that different layers contain rich 3D information. 19
2.10 Three representative WAM paradigms. (A) Joint-modeling WAMs denoise future video and action tokens together. (B) Causal WAMs first generate future observations and then condition action prediction on the generated future representation. (C) Fast-WAM [12] retains video co-training during training but removes explicit future generation at inference time, directly predicting actions from latent world representations in a single forward pass. 21
2.11 Main architecture of the Geometric Action Model (GAM) [13]. . . . 21
2.12 Visualization of a signed distance field around a Panda robot [19]. . 23

2.13 The trajectory (red) of the camera (green) is visualised on top of the zero level set (top row) and SDF slice (middle row) generated by iSDF over time (left to right) [20]. 23
2.14 ReDSDF zero-level reconstruction and planar distance-field slices [21]. 24
2.15 Reconstruction of the distance function used for robot avoidance during human–robot collaboration in ReDSDF [21]. 24
2.16 Illustration of the learned implicit distance isosurfaces $\Gamma(q, y) = 0$ cm (solid) and $\Gamma(q, y) = 10$ cm (transparent) for various configurations of the 7 DoF (Degrees-of-Freedom) Franka Emika Panda robot [22]. 25
2.17 Comparison between CDF and SDF for solving a whole-body inverse-kinematics problem. (a) The initial sampled joint configurations. (b) Gradient projection by CDF. (c) Task space visualization of feasible solutions in (b). (d) Results for distance query-based method with L-BFGS optimizer. We can see that with the baseline SDF approach, the system can get stuck when the gradient of the SDF vanishes or reaches a singularity [23]. 25
2.18 Overview of the LIBERO benchmark suites [24]. 27
2.19 Comparison of the same tasks in the LIBERO dataset [24] and SafeLIBERO [6]. 27

3.1 Schematic representation of a single artificial neuron. 30
3.2 Example of a multilayer perceptron with multiple hidden layers. . . 30
3.3 Attention mechanism in the Transformer architecture [25]. 32
3.4 Vision–language architecture connecting a visual encoder to a language model through a learned projection [26]. 33
3.5 Architecture of the Vision-Language-Action (VLA) paradigm. The model receives a textual prompt and perceptual observations of the scene and produces robot actions. 34
3.6 Architecture of OpenVLA [3]. Visual features extracted by the DINOv2 [28] and SigLIP [29] encoders are projected into the embedding space of a Llama 2 [30] language model, which autoregressively predicts robot action tokens. 36
3.7 Overview of the heterogeneous data sources and policy components used by $\pi_{0.5}$ [16]. 38
3.8 Pre-training and post-training data composition of $\pi_{0.5}$ [16]. 39

4.1 Joint arrangement of the seven-degree-of-freedom Franka Emika Panda. Source: [31]. 41
4.2 Signed distance function (SDF) convention. 41

4.3 Dataset collection procedure used to make the model *scene-free* and *zero-shot*. For each of the M sampled robot configurations, P workspace points are generated. Red points indicate samples in collision with the robot, while non-colliding points are color-coded according to their distance from the robot. The closest sampled point is highlighted in yellow, and the robot keypoints are shown in black. 45
4.4 Representation of the robot geometry and safety margin in three-dimensional space. 47
4.5 Robot geometry and safety margin in a SafeLIBERO task environment [6]. 48
4.6 Architecture of the *PhySVLA* model. 49
4.7 Comparison between the SiLU and ReLU activation functions. . . . 50
4.8 Training plots of $\mathcal{L}_{dist}$, $\mathcal{L}_{sign}$ and $\mathcal{L}_{grad}$ on the training set. 53
4.9 Validation plots of $\mathcal{L}_{dist}$, $\mathcal{L}_{sign}$ and $\mathcal{L}_{grad}$ on the validation set. . . . 53
4.10 Training and validation loss curves, L_{train} and L_{val}. 53
4.11 Inference pipeline of the PhySVLA framework. The model takes as input the task-filtered point cloud of the scene and uses it to construct the corresponding potential field. From this field, it derives a repulsive action, which is then used to correct the nominal action generated by the policy. 54
4.12 RGB-D perception pipeline at time step t during the task *pick up the black bowl between the plate and the ramekin and place it on the plate*, with the back-view camera (left) and wrist camera (right). . . 57

5.1 Aggregate results over the common 1,200-episode pool. The desirable direction is towards the upper left: higher task or safe success with fewer collisions. 65
5.2 Safe success rate versus collision rate for each evaluated SafeLIBERO suite. The ideal direction is towards the upper left. The separation between the two PhySVLA points also exposes the influence of the nominal policy on the final closed-loop outcome. 68
5.3 Safe-success transition matrix over the 1,200 shared episode keys. PhySVLA recovers 612 nominal non-safe-successes and loses 80 nominal safe successes; the diagonal cells contain unchanged outcomes. 70

6.1 Example of VLA *failing the place phase* in the task *pick up the milk and place it in the basket*. 71
6.2 Visualization of *out-of-distribution* data. 72

6.3 Example of VLA *out of distribution (OOD)* in the *pick up the chocolate pudding and place it in the basket* task. 73
6.4 Example of the VLA *failing the grasping phase* in the *pick up the milk and place it in the basket* task. 75
6.5 $\pi_{0.5}$ fine-tuned loss with LoRA. 77

Chapter 1

Introduction

1.1 Problem Definition

This thesis addresses the safe execution of learned robot policies in cluttered manipulation environments.
The objective is to preserve the task-oriented behaviour generated by a policy while preventing actions that may lead to collisions between the robot and its surroundings.

At each time step, denoted by t, a robot policy generates a nominal action according to

$$a_t^{\text{nom}} = \pi_\theta(z_t), \tag{1.1}$$

where π_θ denotes the policy, z_t represents the information available at execution time, and a_t^{nom} is the action proposed by the policy.
Depending on the policy class, z_t may include visual observations, proprioceptive measurements, goal specifications, task embeddings, or natural language instructions. For instance, a Vision-Language-Action model typically combines visual observations with a language command, whereas reinforcement learning and imitation learning policies may operate on state observations, goal vectors, or learned task representations.

The nominal action is assumed to be useful for accomplishing the task, but it is not necessarily safe. A learned policy may correctly identify a target object and generate an appropriate reaching or grasping motion without explicitly accounting for nearby obstacles. Consequently, the robot may move towards the intended target while colliding with the environment during the execution of the motion.
This limitation arises because task-oriented behaviour and geometric safety are generally learned or represented separately. Learned policies acquire manipulation

skills from data, but collision avoidance may be absent from their training objective, encoded only implicitly, or reliable only within the distribution represented by the training data. Therefore, successful task prediction does not imply that the generated motion satisfies explicit geometric safety constraints.
The issue is particularly relevant for articulated manipulators. Avoiding collisions at the end effector alone is insufficient, since a trajectory that is safe for the gripper may still cause the wrist, forearm, elbow, or another intermediate link to collide with the scene. Safety must therefore be evaluated with respect to the full robot geometry and its dependence on the joint configuration.

Let $q_t \in \mathcal{Q}$ denote the robot configuration at time t, where $\mathcal{Q}$ is the robot configuration space. Let $\mathcal{E}_t$ represent the environment information available at execution time. Depending on the perception system, $\mathcal{E}_t$ may consist of RGB-D observations, point clouds, object masks, geometric primitives, occupancy representations, or learned distance fields.
The effect of an action a_t on the robot configuration can be represented through the transition model

$$q_{t+1} = f(q_t, a_t), \tag{1.2}$$

where f may describe the discrete-time robot dynamics, a kinematic update, or a local approximation of the system evolution. The action a_t may correspond to joint velocities, joint displacements, end-effector commands, or another control representation supported by the underlying policy.
To determine whether a robot configuration is safe, a scalar safety function $h(q, \mathcal{E})$ is introduced. This function measures the safety of a configuration q with respect to an environment representation $\mathcal{E}$. The corresponding safe configuration set is defined as

$$\mathcal{Q}_{\text{safe}}(\mathcal{E}) = \{q \in \mathcal{Q} \mid h(q, \mathcal{E}) \geq 0\}. \tag{1.3}$$

Under this convention, $h(q, \mathcal{E}) > 0$ indicates that the robot satisfies the prescribed safety margin, $h(q, \mathcal{E}) = 0$ identifies the boundary of the safe set, and $h(q, \mathcal{E}) < 0$ indicates a collision or a violation of the required clearance.

For collision-avoidance problems, the safety function may be constructed from a signed-distance or clearance measure. To account for the entire manipulator, a full-body safety function can be defined as

$$h(q, \mathcal{E}) = \min_{i \in \mathcal{B}} h_i(q, \mathcal{E}), \tag{1.4}$$

where $\mathcal{B}$ denotes a collection of robot links, collision bodies, or representative points, and $h_i(q, \mathcal{E})$ measures the safety of the i-th robot component. The minimum operator ensures that the global safety value is determined by the part of the robot

closest to violating the safety constraint.

Safety in manipulation is also task dependent. Not every object in the scene should be treated as an obstacle. The target object may need to be approached, contacted, grasped, or moved, whereas surrounding objects and environmental structures should generally be avoided.
Let $\mathcal{Y}_t \subseteq \mathcal{E}_t$ denote the set of objects or scene regions that are considered unsafe for the current task. The safety function can then be expressed as

$$h(q_t, \mathcal{E}_t) \longrightarrow h(q_t, \mathcal{Y}_t). \tag{1.5}$$

This formulation emphasises that safe manipulation requires both geometric reasoning and a task-dependent interpretation of the scene.
Since the nominal policy does not explicitly guarantee that its actions preserve the safe set, an additional mechanism is required. This motivates the introduction of an external safety module that evaluates the action proposed by the policy and modifies it whenever necessary.
The safety module receives the nominal action a_t^{nom}, the current robot configuration q_t, and the perceived environment $\mathcal{E}_t$. It then produces a corrected action according to

$$a_t^{\text{safe}} = \mathcal{F}\left(a_t^{\text{nom}}, q_t, \mathcal{E}_t\right), \tag{1.6}$$

where $\mathcal{F}$ denotes the safety-correction function and a_t^{safe} is the action ultimately executed by the robot.
The desired behaviour of $\mathcal{F}$ is based on the principle of minimal intervention. When the nominal action is safe, the module should leave it unchanged. When the proposed action threatens to violate a safety constraint, the module should modify it by the smallest amount required to recover safety.
This correction can be formulated as the constrained optimisation problem

$$a_t^{\text{safe}} = \arg\min_{a \in \mathcal{A}} \|a - a_t^{\text{nom}}\|^2 \quad \text{s.t.} \quad a \in \mathcal{A}_{\text{safe}}\left(q_t, \mathcal{E}_t\right), \tag{1.7}$$

where $\mathcal{A}$ is the robot action space and $\mathcal{A}_{\text{safe}}(q_t, \mathcal{E}_t)$ is the set of actions that are admissible from the current robot configuration and perceived scene.
Using the transition model introduced in Equation (1.2), the safe action set can be defined as

$$\mathcal{A}_{\text{safe}}(q_t, \mathcal{E}_t) = \left\{a \in \mathcal{A} \mid h\left(f(q_t, a), \mathcal{E}_t\right) \geq 0\right\}. \tag{1.8}$$

This constraint requires the configuration predicted after applying the corrected action to remain within the safe set.

In a differential control setting, safety may instead be imposed through a control barrier function (CBF) condition:

$$\dot{h}(q_t, a) \geq -\alpha \left(h(q_t, \mathcal{E}_t) \right), \tag{1.9}$$

where $\alpha(\cdot)$ is an extended class-$\mathcal{K}$ function that regulates how quickly the system may approach the boundary of the safe set.

The proposed formulation is intentionally policy agnostic. The safety module does not require access to the internal architecture, training objective, or training dataset of the nominal policy. It only assumes access to the action generated by the policy, the current robot configuration, and a representation of the surrounding environment.
As a result, the nominal action may be produced by a reinforcement learning policy, an imitation learning policy, a diffusion-based visuomotor policy, a classical controller, or a Vision-Language-Action model.
In the experimental setting considered in this thesis, nominal actions are generated by a Vision-Language-Action model. Nevertheless, the formulation is not restricted to this policy class and applies more generally to any controller that produces task-directed actions.

The problem considered in this thesis is therefore defined as follows:

Problem Definition

Given a nominal action produced by an arbitrary robot policy, the current configuration of an articulated manipulator, and a task-dependent representation of the surrounding environment, compute a minimally modified action that preserves the task intent of the nominal policy while reducing the risk of collision.
The nominal policy remains responsible for task-level behaviour, whereas an external, policy-agnostic safety module evaluates the geometric feasibility of the proposed motion and intervenes only when the corresponding safety constraints may be violated.

1.2 Motivation

Vision-Language-Action (VLA) models represent one of the most promising directions for generalist robotic manipulation. By combining visual perception, language understanding, and action generation in a single policy, these models can execute manipulation tasks from natural language instructions and visual observations.

Recent systems such as RT-1 [1], Octo [2], OpenVLA [3], and π_0 [4] show that large-scale visuomotor policies can generalize across objects, tasks, and scenes better than traditional task-specific controllers. This ability is central for embodied intelligence: instead of designing or training a different controller for each manipulation task, a single policy can be reused across many instructions and environments.
However, generalization at the semantic and task levels does not automatically imply safe physical execution. A VLA model may correctly interpret the language instruction, identify the target object, and generate an action that is coherent with the task, while still producing a motion that is unsafe from a geometric point of view.

This distinction is particularly important because many VLA models are evaluated in environments that are relatively safe by design, as shown in Figure 1.1. In many benchmark settings, the target object is reachable, the desired motion is geometrically simple, and nearby objects do not strongly interfere with the robot trajectory. Under these conditions, a policy can achieve high task success without demonstrating robust obstacle avoidance.

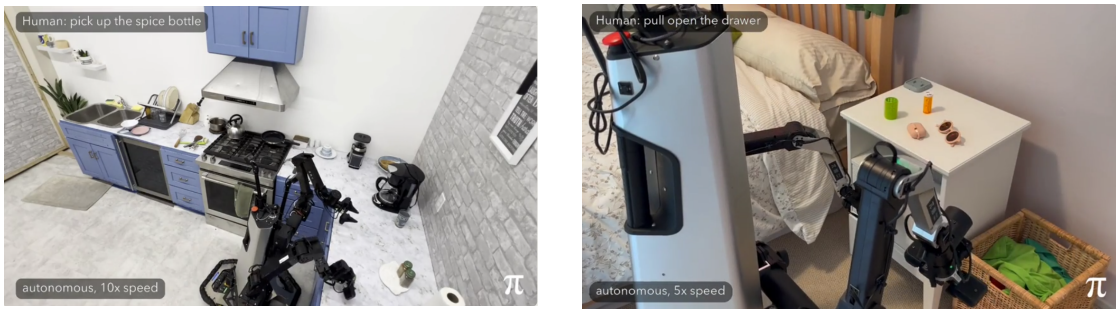


Figure 1.1: Examples of scenes involving VLA deployment. Source: Physical Intelligence [5].

The problem becomes more visible in scenes that are unsafe by design. Obstacles may be close to the target, several objects may jointly constrain the motion, or the robot may need to move through a narrow passage before reaching the object. In such scenarios, the policy must satisfy two requirements at the same time: it must follow the language instruction and avoid unsafe physical interactions. This is a harder problem than standard task completion, because the robot must preserve the task strategy while respecting constraints that depend on the detailed geometry of the scene.
A possible way to incorporate obstacle avoidance is to train a policy with reinforcement learning. Collision penalties, safety costs, or obstacle-avoidance terms can be added to the reward, and the policy can in principle learn to complete the task while avoiding unsafe contacts. However, this approach becomes difficult to scale to generalist manipulation. The reward must encode several objectives at the

same time: reaching the target, avoiding obstacles, maintaining efficient motion, respecting contact constraints, and completing the manipulation task.
This reward-engineering process can require significant computational resources and data collection time. The difficulty is also often task-specific. A reward that works for one task, one object layout, or one obstacle configuration may not work when the shape, position, or number of obstacles changes. As a result, using reinforcement learning to obtain safe behaviour may require repeated training and repeated reward design whenever the task or environment distribution changes.
Imitation learning has a related limitation. A policy can learn obstacle-aware behaviour from expert demonstrations if the demonstrations contain safe trajectories around obstacles. However, collecting demonstrations for many tasks, object arrangements, and obstacle configurations is expensive. Demonstrations also do not provide an explicit safety model; they show what the expert did in the observed cases, but they do not directly define which configurations are safe or unsafe.
Another possible solution is to use a global motion planner. A planner can build a geometric model of the scene and compute a collision-free trajectory to a target pose. This is effective when the goal is known geometrically and the task can be reduced to moving the robot from one configuration to another. However, in language-conditioned manipulation, the nominal policy often contains information that is not captured by a purely geometric planning objective. The policy not only moves the robot toward an object, but it also encodes task-specific manipulation behaviour learned from data. For example, when the instruction involves picking up a cup, the appropriate motion may depend on whether the robot should grasp the handle, approach the cup from a particular side, avoid touching the rim, or prepare for a later placing motion. A generic collision-free trajectory may be geometrically valid, but still poorly aligned with the manipulation strategy implied by the task.

These observations suggest that safety should be added in a way that preserves the role of the learned policy.
For this reason, safe manipulation requires explicit geometric reasoning. A useful safety mechanism should not only detect that an object is present in the scene; it should also estimate how close the robot is to that object and which action direction would increase clearance. Distance-based representations are natural for this purpose. Signed distance fields, configuration-space distance fields, and neural implicit safety functions provide scalar measures of proximity to collision, together with gradients that can be used for reactive correction. Unlike binary collision signals, distance fields provide information before contact occurs, making them suitable for preventive safety filtering.

From a broader perspective, this problem lies at the intersection of Physical AI, robot learning, and physics-informed safety. Physical AI systems must not only

understand visual and linguistic context, but also act reliably in the physical world under geometric constraints.

In summary, the motivation for this thesis comes from a gap between what modern robot policies can achieve and what safe robotic deployment requires. VLA models and other learned policies can generate coherent task-directed actions, but they usually lack explicit geometric safety guarantees. Reinforcement learning and imitation learning can learn obstacle avoidance, but often require task-specific training, reward engineering, or costly demonstrations. Global planners can compute collision-free trajectories, but may overwrite the semantic manipulation behaviour encoded by the learned policy, especially in long-horizon tasks. A plug-and-play safety layer offers a more scalable alternative: preserve the policy as the source of task intent, and add execution-time geometric correction to improve safety in cluttered manipulation scenes.

1.3 Contributions

This thesis makes four main contributions. First, it introduces *PhySVLA*, a task-aware safety layer that improves the execution of frozen VLA policies without policy fine-tuning. Second, it proposes a forward-kinematics-based neural distance field trained on scene-free geometric data and reusable zero-shot across unseen obstacle layouts. Third, it connects payload-aware distance gradients to the VLA action space through a minimally invasive CBF projection.
Finally, it validates the complete system with both OpenVLA-OFT and $\pi_{0.5}$ on SafeLIBERO, showing substantial improvements in collision avoidance and collision-free task success while preserving the task competence of the underlying policies.

Chapter 2

Related Work

This chapter reviews the generalist policies, geometry-aware action methods, and differentiable distance representations most relevant to PhySVLA. Following an intervention-point taxonomy, the action methods are organized according to when geometric information influences the robot command: after policy decoding, during generative inference, or during policy training. This distinction makes their requirements and safety semantics explicit, while the distance-field literature is treated separately as the geometric foundation on which these intervention mechanisms can operate.

2.1 Scope and intervention-point taxonomy

A Vision–Language–Action (VLA) model provides a task-conditioned action prior: it maps observations, language, and proprioception to a nominal action or action chunk. A geometric model estimates whether that motion is physically admissible in the current scene. A safety mechanism determines how geometric information should change the command sent to the robot. Improving one component does not automatically solve the other two.
Let

$$a_t^{\text{nom}} = \pi_\theta(o_{\le t}, q_t, \ell) \tag{2.1}$$

denote the action proposed by a policy with parameters θ, conditioned on the observation history $o_{\le t}$, robot configuration q_t, and language instruction ℓ. The literature considered in this chapter is organized by the point at which geometry can alter this mapping, as shown in Figure 2.1.

1. *post-policy action correction* treats a_t^{nom} as a completed command and projects it onto a locally admissible action set;

2. *in-generation geometric guidance* changes the intermediate samples of a diffusion- or flow-based policy before its final action chunk is decoded;

3. *training-time geometry integration* modifies the policy architecture, or its supervision so that geometric and physical structure is internalized by the action model.

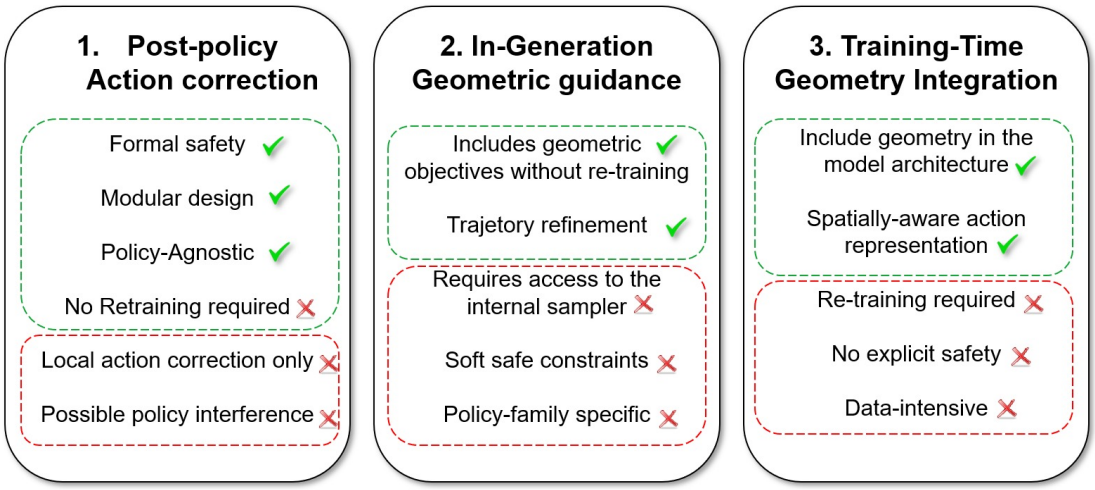


Figure 2.1: Intervention-point taxonomy for geometry-aware robot policies. The three families differ in policy access, modularity, and the semantics of the safety signal.

The three families require different levels of access to the base policy and provide different safety semantics.
Training-time methods can improve spatial representations and the prior probability of feasible motions, but the trained policy does not necessarily verify a constraint in the deployment scene.
In-generation guidance can inject new geometric objectives without retraining, but it requires access to the internal sampler and generally acts through soft energies.
Post-policy filters are compatible with a black-box policy that exposes only its final command, but their guarantees are conditional on the fidelity of the geometric model, controller dynamics, and discretization.
The principal differences between these families, together with the methods reviewed in this chapter, are summarized in Table 2.1.

Neural-JSDF, CDF, iSDF, and ReDSDF are deliberately not presented as a fourth action-intervention family. They provide differentiable geometric quantities, but do not by themselves decide how or when a robot policy action must change.

Table 2.1: Intervention-point taxonomy of the geometry-aware action methods reviewed in this chapter. *Required access* indicates what must be exposed or modified beyond the final robot command.

Method	Intervention	Required access	Principal limitation
		Post-Policy Action Correction	
VLSA/AEGIS [6]	After action decoding	Final command only	Single selected obstacle and end-effector-centric geometry.
SQ-CBF [7]	After action decoding	Final command, fitted primitives, and robot model	Metric barrier quality depends on primitive fitting; task semantics remain external.
Safe-SAGE [8]	Planning and control after policy perception	Semantic scene model, MPC, and CBF controller	Designed for legged navigation rather than articulated manipulation.
		In-Generation Geometric Guidance	
OmniGuide [9]	During diffusion or flow inference	Sampler, gradients, and differentiable kinematics	Soft, policy-family-specific guidance that adds inference cost.
		Training-Time Geometry Integration	
Explicit feasibility [10]	Geometric supervision during fine-tuning	Weights, forward kinematics, and obstacle geometry	Learned feasibility is not re-verified as an online constraint.
ROCKET [11]	Representation alignment during training	Hidden states, weights, and a 3D teacher	Spatial features do not by themselves define clearance or a safe set.
Fast-WAM [12]	Joint video–action training	Video–action architecture and future-video supervision	Video priors are implicit; no distance or collision constraint is evaluated.
GAM [13]	Geometric backbone and policy training	New architecture and large-scale training	Predicted geometry does not define an online admissibility constraint.

2.2 VLA backbones and the geometry gap

Generalist robot policies have progressively moved from task-specific behaviour cloning toward large models trained across tasks, objects, and embodiments. RT-1 represents observations, language, and actions as token sequences and demonstrates large-scale multi-task control on a real robot [1]. Octo provides an open-source transformer policy designed for adaptation across heterogeneous robot datasets [2], while OpenVLA transfers a pretrained vision–language backbone to robot action prediction and makes large-scale VLA fine-tuning accessible [3]. These models are important task-level foundations for the present work: they encode which object should be manipulated and what action is likely to advance the instruction.
Continuous generative policies extend this paradigm by predicting distributions over action chunks rather than a single deterministic command. Diffusion Policy models

the conditional action distribution through iterative denoising [14]; RDT-1B scales a diffusion architecture to bimanual manipulation [15]. The π_0 family instead uses flow matching and a dedicated action expert on top of a pretrained vision–language model [4]. Its successor, $\pi_{0.5}$, adds heterogeneous co-training and high-level semantic prediction to improve open-world generalization [16]. The generative formulation is especially relevant because it creates an internal trajectory-generation process that can be guided at inference time, as discussed in Section 2.4.

2.2.1 Compared VLA backbones

The final evaluation uses two frozen nominal policies, OpenVLA-OFT and $\pi_{0.5}$. Both produce continuous manipulation commands from visual, linguistic, and proprioceptive inputs, but they arise from different action modelling choices. Evaluating the same post-policy safety mechanism with both backbones therefore tests whether its effect is tied to one specific VLA.

OpenVLA-OFT

OpenVLA-OFT is a fine-tuning recipe for OpenVLA that replaces discrete, autoregressive action-token generation with parallel continuous action heads [17]. The method studies both regression and diffusion heads together with action chunking and proprioceptive conditioning. The configuration evaluated in this thesis uses its continuous regression interface, which produces a short action chunk more efficiently than token-by-token decoding. Despite this stronger action interface, obstacle clearance remains implicit in the demonstrations: OpenVLA-OFT does not independently verify that the decoded chunk is collision-free in the current scene.

$\pi_{0.5}$

The $\pi_{0.5}$ policy combines a pretrained vision–language backbone with a flow-matching action expert and a heterogeneous co-training mixture spanning robot experience and vision–language data [16]. At deployment, it maps multiple images, language, and proprioception to continuous operational-space action chunks. Its strong task and semantic prior makes it the most capable nominal backbone in the experiments, but its training objective still does not constitute an explicit online collision constraint.

2.2.2 Task success is not a safety specification

The nominal learning objective of these policies rewards agreement with demonstrated actions. In a simplified behaviour-cloning form,

$$\mathcal{L}_{\text{act}}(\theta) = \mathbb{E}_{(z,a^\star)\sim\mathcal{D}}\left[\ell(\pi_\theta(z), a^\star)\right], \tag{2.2}$$

where z collects the policy inputs and $a^\star$ is an expert action. This objective can transfer rich task semantics, but it does not explicitly measure the clearance of every robot link, the geometry of an unseen obstacle, or the motion of a grasped object. These properties are learned only to the extent that they are identifiable from the demonstrations and represented by the model.
Consequently, the set of actions that appear useful for the task, $\mathcal{A}_{\text{task}}(z_t)$, need not be contained in the set that is safe in the current configuration and scene, $\mathcal{A}_{\text{safe}}(q_t, \mathcal{E}_t)$:

$$\mathcal{A}_{\text{task}}(z_t) \not\subseteq \mathcal{A}_{\text{safe}}(q_t, \mathcal{E}_t). \tag{2.3}$$

A policy may therefore select the correct target and still collide with a distractor through its wrist or elbow.
This gap motivates the three intervention families reviewed next.

2.3 Post-policy action correction

2.3.1 Control barrier functions and minimal intervention

Post-policy methods separate task generation from safety enforcement. The learned policy first returns u_t^{nom}; a controller then computes the closest admissible command. For a controlled system

$$\dot{x} = f(x) + g(x)u, \tag{2.4}$$

and a continuously differentiable barrier $h(x)$, the intended safe set is $\mathcal{C} = \{x : h(x) \geq 0\}$. A standard control barrier function (CBF) condition requires

$$\nabla h(x)^\top (f(x) + g(x)u) \geq -\alpha(h(x)), \tag{2.5}$$

where α is an extended class-$\mathcal{K}$ function. When the regularity, feasibility, and initial-set assumptions hold, enforcing Equation (2.5) renders the safe set forward invariant [18].
The condition is affine in u for control-affine dynamics and can therefore be included in a quadratic program:

$$u_t^{\text{safe}} = \arg\min_u \frac{1}{2}\|u - u_t^{\text{nom}}\|_W^2 \quad \text{s.t.} \quad L_f h(x_t) + L_g h(x_t) u \geq -\alpha(h(x_t)). \tag{2.6}$$

The objective formalizes minimal intervention: a command that already satisfies the constraint is preserved, while an unsafe command is modified only in the directions required by the barrier. This decomposition is attractive for large VLAs because the safety layer need not access the policy weights, training data, or decoding process. An example of this construction for drone navigation is shown in Figure 2.2.

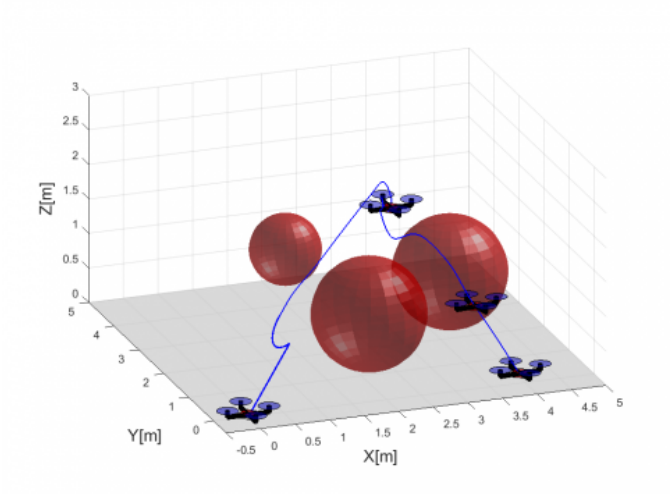


Figure 2.2: Control barrier function method applied to drone navigation.

2.3.2 VLSA: Vision-Language-Action Models with Plug-and-Play Safety Constraint Layer

The VLSA framework [6], instantiated by AEGIS and shown in Figure 2.3, is the closest precedent to the post-policy architecture adopted in this thesis. It treats a pretrained VLA as a nominal controller and places a plug-and-play safety constraint layer after the action decoder. The inference pipeline performs four conceptually distinct operations. First, a vision–language model reasons about the instruction and scene to select the non-target object that poses the most relevant collision risk. Second, open-vocabulary grounding localizes that object. Third, multi-view depth observations are fused and filtered to recover its three-dimensional geometry. Finally, minimum-volume enclosing ellipsoids approximate the obstacle and the end-effector assembly, and a CBF-QP projects the VLA command.

If $h_{\text{ell}}(x; \mathcal{O}_t)$, shown in Figure 2.4, denotes the separation barrier constructed from the selected obstacle $\mathcal{O}_t$, AEGIS follows the structure

$$u_t^{\text{AEGIS}} = \arg\min_u \left\| u - u_t^{\text{VLA}} \right\|_2^2 \quad \text{s.t.} \quad \dot{h}_{\text{ell}}(x_t, u) \geq -\alpha(h_{\text{ell}}(x_t)) \,. \tag{2.7}$$

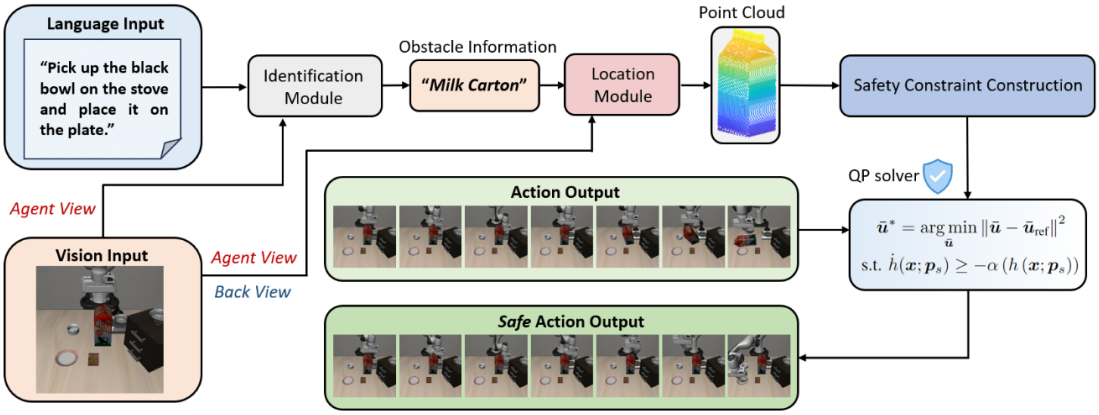


Figure 2.3: Workflow of the AEGIS safety layer [6].

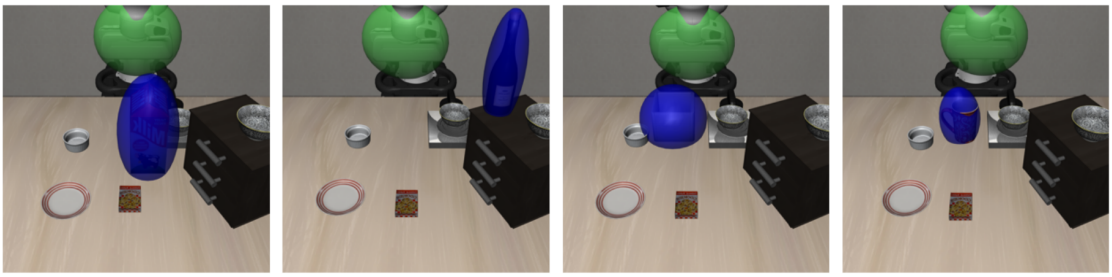


Figure 2.4: VLSA quadratic programming for obstacle avoidance with the robot end-effector [6].

The main advantage is modularity. The same safety layer can be attached to an existing policy without collecting new demonstrations or fine-tuning the VLA, and the optimization has an explicit interpretation as a minimal correction. The semantic front-end also prevents the controller from treating the task target, which often must be contacted, as an ordinary obstacle.
The compact representation that makes AEGIS efficient also limits its geometric coverage. Selecting one critical obstacle does not capture scenes in which several objects jointly restrict the motion. Protecting the end-effector assembly does not prevent an intermediate link from colliding, and an enclosing ellipsoid can be conservative for elongated or non-convex shapes. A grasped payload is not part of the nominal robot envelope. Moreover, the CBF guarantee is conditional on the perception chain: a missed object, incomplete depth, incorrect clustering, or an ellipsoid that fails to enclose the relevant geometry changes the safe set actually enforced by the controller.

2.3.3 Superquadric-Based Safety Filtering

SQ-CBF [7], shown in Figure 2.5, isolates another important limitation of post-policy filtering: an expressive implicit shape function is not necessarily a numerically suitable barrier.

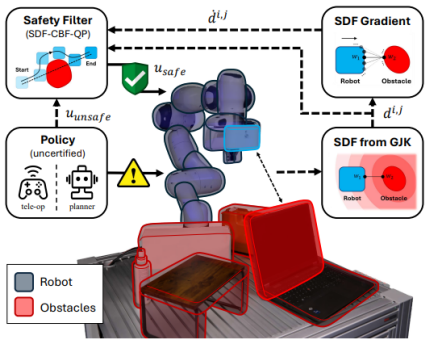


Figure 2.5: Overview of the SQ-CBF [7] framework for geometry-aware safety filtering.

Superquadrics compactly represent a wide range of objects, but the magnitude and conditioning of the raw implicit-function gradient do not correspond to Euclidean clearance. SQ-CBF instead estimates a signed distance using geometric closest-point computations and stabilizes its derivative through randomized smoothing before applying a CBF filter. The method shows why the quality of ∇h, not only the sign of h, matters for corrective control. Its scope remains geometric: primitive fitting and a known robot model are required, while semantic decisions about which contact is allowed remain external.

Safe-SAGE [8], shown in Figure 2.6, provides a complementary example from legged navigation. It combines semantic perception, Poisson safety functions, model-predictive guidance, and a CBF layer so that different object classes can induce different margins and passing behaviours.
Its main conceptual contribution to manipulation is the notion of *semantic blindness*: a purely geometric filter cannot distinguish an object that must be contacted from one that must be avoided.
Nevertheless, Safe-SAGE addresses mobile legged navigation rather than articulated manipulation, and does not solve the mapping from full-arm clearance to a VLA operational-space action.

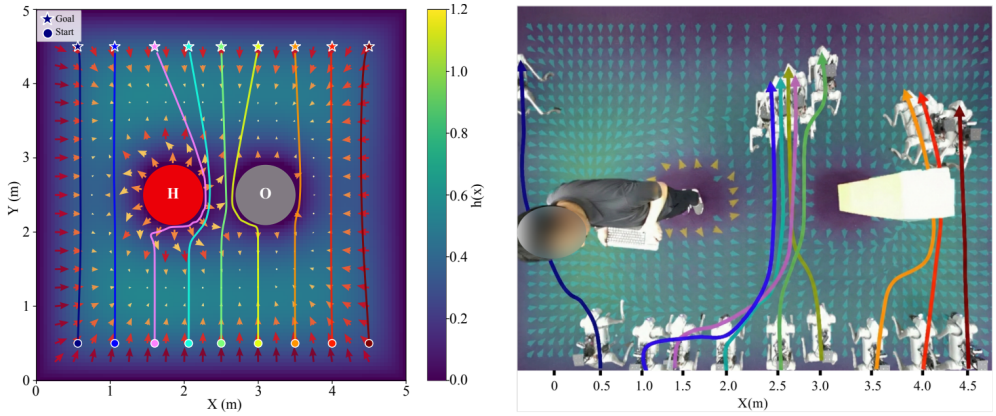


Figure 2.6: Simulation (left) and real-world experiment (right) on the Unitree Go2 quadruped robot using the Safe-SAGE model [8].

2.4 Guidance inside generative inference

Post-policy projection is not the only way to incorporate geometric information without retraining a policy. Generative policies based on diffusion or flow matching construct an action chunk through a sequence of intermediate latent variables. This makes it possible to introduce external objectives directly during generation, modifying the sampling dynamics before the final action trajectory is produced. OmniGuide [9] adopts this principle and introduces a unified inference-time framework in which heterogeneous sources of guidance are represented as differentiable energy fields defined over Cartesian robot trajectories, as illustrated in Figure 2.7.

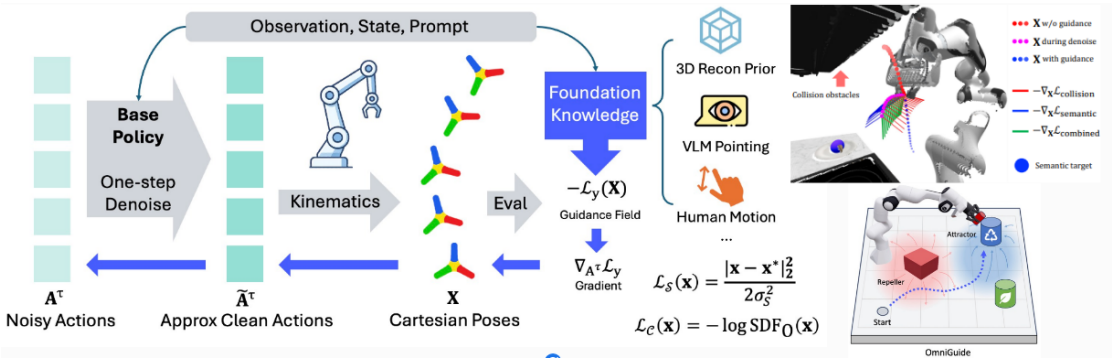


Figure 2.7: Pipeline overview of the OmniGuide [9] action generation method.

Let A^τ denote the noisy action chunk at flow time τ, and let o denote the policy observation. A flow-based VLA defines a velocity field $v_\theta(A^\tau, o)$ that progressively transports the noisy action toward a clean action trajectory. OmniGuide introduces

an additional condition y, representing an external source of guidance, and modifies the velocity field according to [9]

$$v_\theta(A^\tau, o \mid y) = v_\theta(A^\tau, o) + \lambda_\tau \nabla_{A^\tau} \log p(y \mid A^\tau), \tag{2.8}$$

where λ_τ controls the strength of the guidance. At every flow step, the base policy first estimates a clean action chunk from A^τ. After decoding the action representation when necessary, a differentiable kinematic model maps the predicted robot configuration to Cartesian trajectories X. External constraints are then represented by a differentiable energy $\mathcal{L}_y(X)$, so that the guidance term can be written as

$$\nabla_{A^\tau} \log p(y \mid A^\tau) = -\nabla_{A^\tau} \mathcal{L}_y(X). \tag{2.9}$$

The gradient is therefore propagated from Cartesian space through the robot model and the clean-action estimate back to the noisy action variables, steering the generative process itself [9].
The same abstraction is instantiated with several types of spatial guidance. For collision avoidance, OmniGuide constructs a repulsive field from an environment point cloud, using a signed-distance-based representation to penalize robot configurations that approach obstacles. Task-relevant objects and the robot body are removed from the collision geometry using semantic information before the field is evaluated [9]. Semantic guidance instead creates attractive fields around task-relevant locations. Given a target position x^*, the paper models the target likelihood as a Gaussian, yielding the energy

$$\mathcal{L}_S(x) = \frac{|x - x^*|_2^2}{2\sigma_S^2} + \text{Const.} \tag{2.10}$$

This produces a gradient that attracts the generated end-effector trajectory toward the grounded target [9].
This flexibility comes at the cost of tighter coupling with the generative policy. The method requires access to the intermediate flow states, a differentiable estimate of the clean action, and a differentiable mapping from action variables to Cartesian robot geometry. It is therefore fundamentally different from a post-hoc safety layer that operates only on the final policy output. Moreover, the guidance strength introduces a trade-off with the pretrained policy prior: weak guidance may be insufficient to alter an unsafe trajectory, whereas excessively strong guidance can interfere with task-relevant behaviour, a trade-off explicitly observed in the OmniGuide ablations [9].
OmniGuide is consequently best understood as geometry-conditioned generative inference rather than as a hard safety filter such as a CBF projection.

2.5 Geometry incorporated during training

Training-time approaches aim to make the nominal policy itself more spatially or physically informed. They differ in what is supervised: explicit clearance, features distilled from a geometric teacher, future video dynamics, or future geometric representations. Their common limitation is that improving the prior does not by itself enforce an online safe set.

2.5.1 Explicit feasibility supervision

Wei et al. [10] add an explicit obstacle-feasibility objective to the training of a diffusion-based VLA, as shown in Figure 2.8. Predicted action chunks are converted

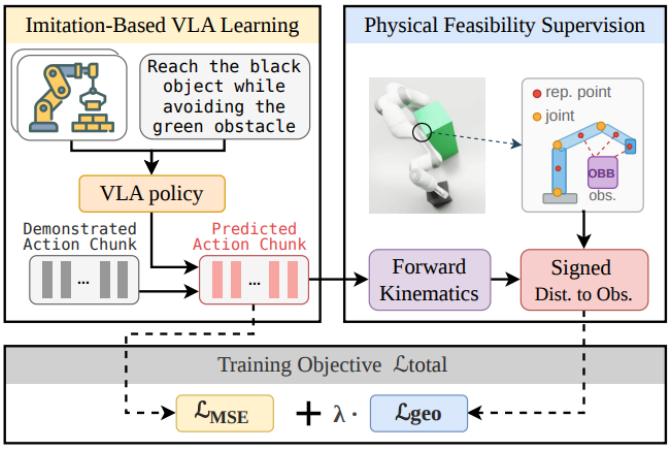


Figure 2.8: VLA training with explicit physical feasibility supervision [10].

into future robot configurations, forward kinematics maps representative link geometry into the workspace, and signed distances to an obstacle bounding box are evaluated.
A margin loss penalizes predicted link states whose clearance falls below δ. In representative form, the objective is

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{act}} + \lambda_{\text{geo}} \frac{1}{|\mathcal{I}|} \sum_{(\tau,j)\in\mathcal{I}} \left[\max(0, \delta - d_{\tau,j})\right]^2, \tag{2.11}$$

where $d_{\tau,j}$ is the clearance of link representation j at predicted horizon step τ, and $\mathcal{I}$ selects states near the margin. The loss provides denser information than a binary collision label and encourages the policy to move in directions that increase clearance.
This approach directly demonstrates the value of physical supervision, but its

assurance is statistical. The forward-kinematics distance term is used to shape the weights during training; the deployed policy does not necessarily re-evaluate those distances in a novel scene. Distribution shift can therefore still produce an unsafe action. The method also requires access to the policy weights and an additional fine-tuning stage, making it unsuitable as a plug-and-play layer for an arbitrary frozen VLA.

2.5.2 Geometric representation alignment: ROCKET

ROCKET [11] targets a different source of error: the weak three-dimensional structure of VLA visual features learned predominantly from two-dimensional data. A frozen 3D vision foundation model acts as a teacher, and hidden states from multiple depths of the VLA are aligned to corresponding teacher representations. Rather than learning an independent projection for each pair of layers, ROCKET shares one projector across the residual stream, as shown in Figure 2.9. This

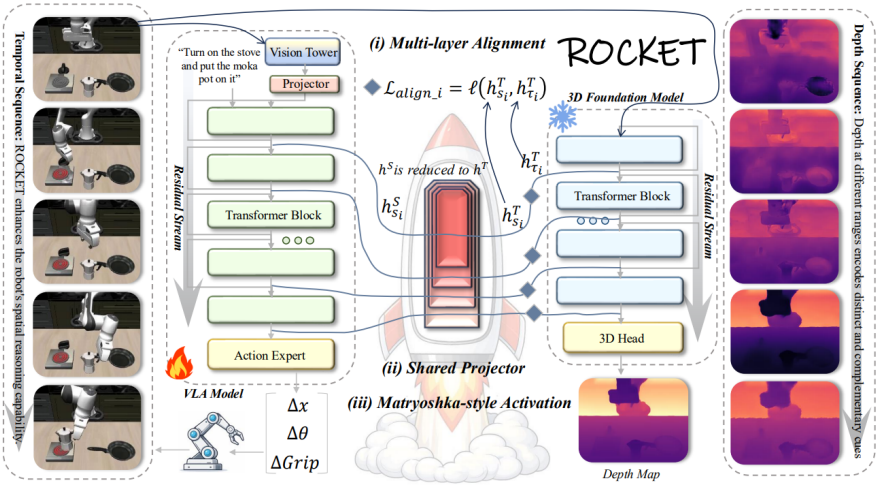


Figure 2.9: Overview of the ROCKET layer [11]. On the left, a sample from LIBERO to showcase ROCKET's performance. On the right, the outputs from VGGT layers are used to directly predict the depth map, demonstrating that different layers contain rich 3D information.

layer-invariant mapping reduces conflicting gradients between alignment objectives. Let $\mathcal{S}$ contain selected student–teacher layer pairs, H_s^{VLA} and H_r^{3D} denote their features, and $P_\phi^{(s)}$ denote the nested view of the shared projector used at depth s. The training objective can be summarized as

$$\mathcal{L}_{\text{ROCKET}} = \mathcal{L}_{\text{act}} + \frac{\lambda_{\text{align}}}{|\mathcal{S}|} \sum_{(s,r)\in\mathcal{S}} \left[1 - \cos\left(P_\phi^{(s)}(H_s^{\text{VLA}}), H_r^{\text{3D}}\right)\right]. \tag{2.12}$$

The method is computationally efficient relative to repeated single-layer searches and improves task success across spatially demanding benchmarks.
ROCKET nevertheless improves representation, not formal safety. It requires the VLA hidden states and weights during fine-tuning, as well as a geometric teacher. At deployment the teacher no longer evaluates a distance or rejects a collision. The aligned latent space contains a stronger spatial prior, but does not define a metric clearance threshold. Reported LIBERO success is therefore evidence of better task execution, not a collision-avoidance guarantee.

2.5.3 World action model: Fast-WAM

World action models (WAMs) learn actions jointly with the visual evolution of the scene. Their usual motivation is that predicting how observations change under interaction forces the model to capture object dynamics and temporal structure. Many WAMs use an imagine-then-execute pipeline in which future frames are iteratively generated before actions are chosen, creating substantial test-time latency.
Fast-WAM [12] separates the effect of future-video supervision during training from the effect of explicit future imagination during deployment. Its architecture, shown in Figure 2.10, groups the current clean frame, noised future video tokens, and noised action tokens in a joint transformer computation. A structured attention mask prevents the action branch from reading ground-truth future information that will not be available at deployment. Video and action flow-matching objectives are optimized jointly:

$$\mathcal{L}_{\text{Fast-WAM}} = \mathcal{L}_{\text{act}} + \lambda_{\text{vid}}\mathcal{L}_{\text{vid}}. \tag{2.13}$$

At inference time, the future-video branch is removed and the action expert predicts from the current visual representation.
Fast-WAM is relevant because it shows how physical and temporal priors can enter the policy through auxiliary world modelling. It is not, however, an explicit geometry or safety method. The video latent does not expose a signed distance, link–obstacle clearance, or constraint that a controller can enforce. The method requires training a large video–action architecture, and unseen obstacle layouts remain governed by the generalization of the learned prior.

2.5.4 Geometric backbone and future prediction: GAM

The Geometric Action Model (GAM) [13], as shown in Figure 2.11, integrates geometry even more deeply by using a pretrained geometric foundation model as the policy backbone.

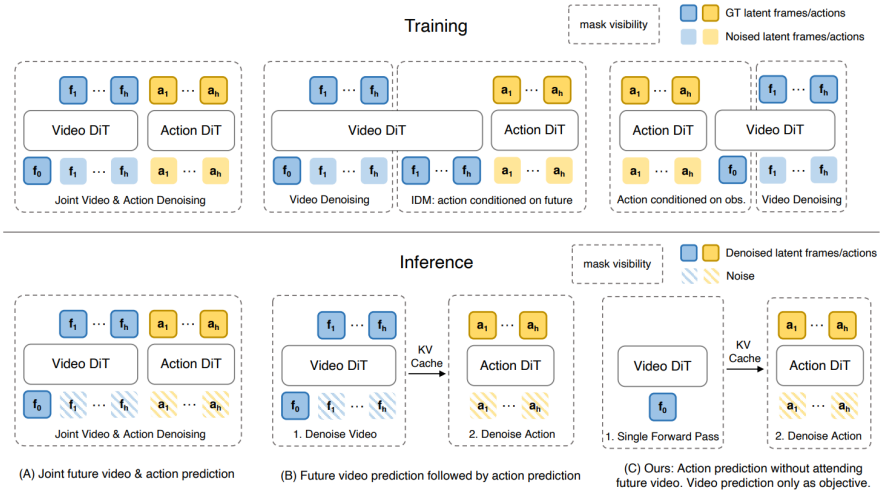


Figure 2.10: Three representative WAM paradigms. (A) Joint-modeling WAMs denoise future video and action tokens together. (B) Causal WAMs first generate future observations and then condition action prediction on the generated future representation. (C) Fast-WAM [12] retains video co-training during training but removes explicit future generation at inference time, directly predicting actions from latent world representations in a single forward pass.

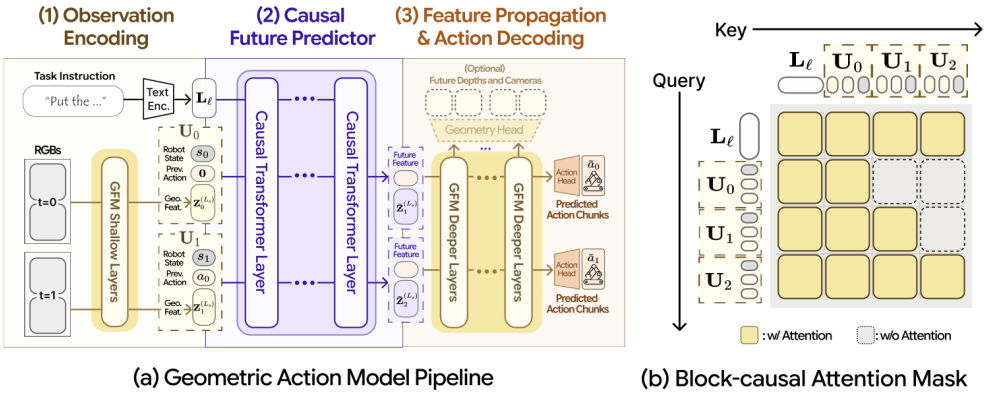


Figure 2.11: Main architecture of the Geometric Action Model (GAM) [13].

Shallow geometric blocks encode multi-view RGB observations into spatial tokens. A causal predictor then fuses these tokens with language, proprioception, and action history to predict future geometric representations. Deeper geometric blocks propagate and decode the predicted structure, while separate heads output the action chunk and future depth.
Geometry is thus not an auxiliary input appended to a conventional VLA; it is

the representational substrate through which observations and future actions are connected.
The training signal combines action imitation, latent future-feature prediction, and depth reconstruction:

$$\mathcal{L}_{\text{GAM}} = \lambda_{\text{act}}\mathcal{L}_{\text{act}} + \lambda_{\text{feat}}\mathcal{L}_{\text{feat}} + \lambda_{\text{depth}}\mathcal{L}_{\text{depth}}. \tag{2.14}$$

Unlike an imagine-then-execute WAM, GAM can produce the geometric prediction and action in one forward pass. Its architecture encourages temporally coherent spatial features and improves robustness to viewpoints and scene layouts.
The integration cost is substantial. GAM requires large-scale pretraining, benchmark-specific fine-tuning, future-feature and depth supervision, and access to the complete policy architecture. Most importantly for this thesis, accurate future depth is not equivalent to an admissible action: equation (2.14) contains no explicit full-body clearance margin or invariant-set constraint. GAM should therefore be viewed as a geometry-native policy, rather than as an execution-time safety filter.

2.5.5 Comparison of training-time strategies

The four strategies encode complementary inductive biases. Explicit feasibility supervision places metric geometry in the loss; ROCKET distils spatial representations from a frozen 3D teacher; Fast-WAM learns implicit physical regularities through video co-training; and GAM makes predicted geometry part of the backbone itself.
These methods can improve the nominal action before a controller intervenes, and may reduce how often a safety filter is required. They also share three limitations for the present problem: they modify or replace the policy, their guarantees depend on training-distribution generalization, and none of them checks a hard multi-link clearance constraint against the current scene at every control step.

2.6 Differentiable distance fields

The intervention mechanisms above need a geometric quantity that is both meaningful and differentiable. Signed distance fields (SDFs), shown in Figure 2.12, provide this bridge.
For a closed body $\mathcal{B} \subset \mathbb{R}^3$, an SDF can be written as

$$d_{\mathcal{B}}(x) = \begin{cases} \inf\limits_{y \in \partial\mathcal{B}} \|x - y\|_2, & x \notin \mathcal{B}, \\ -\inf\limits_{y \in \partial\mathcal{B}} \|x - y\|_2, & x \in \mathcal{B}. \end{cases} \tag{2.15}$$

The value measures penetration or clearance, while $\nabla_x d_{\mathcal{B}}$ points locally in the direction of increasing distance.

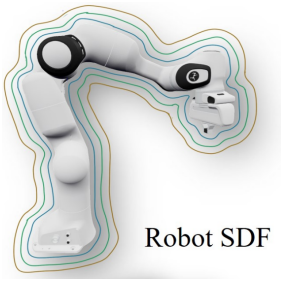


Figure 2.12: Visualization of a signed distance field around a Panda robot [19].

This metric meaning is precisely what a generic occupancy score or unnormalized implicit primitive lacks.

2.6.1 Scene and workspace distance fields

iSDF [20], shown in Figure 2.13, learns a continuous scene SDF online from a stream of posed depth images. The representation supports queries at arbitrary three di-

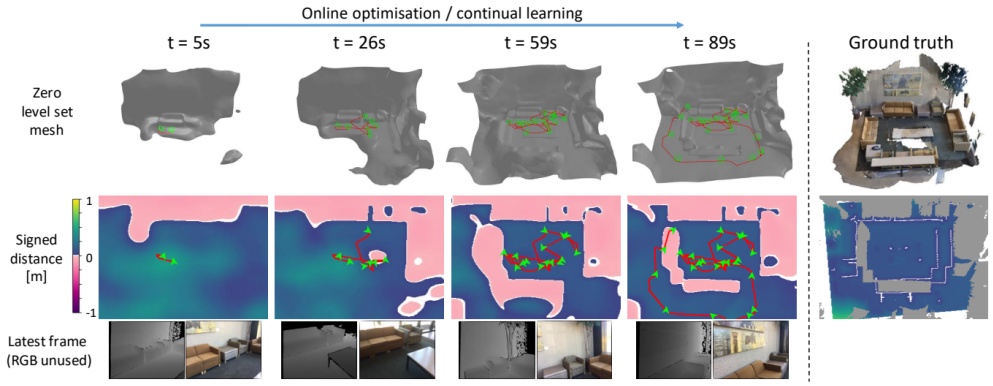


Figure 2.13: The trajectory (red) of the camera (green) is visualised on top of the zero level set (top row) and SDF slice (middle row) generated by iSDF over time (left to right) [20].

mensional points and jointly captures surface geometry and free space. It is valuable for perception and mapping, but it represents the *scene*; robot configuration and articulated full-body distance still have to be supplied by a separate collision model.

In contrast, ReDSDF [21], illustrated in Figure 2.14 and Figure 2.15, focuses on smooth neural distance representations suitable for articulated bodies and reactive motion generation. Its regularization encourages useful gradients across different spatial scales, which is important when a controller must respond both far from

and close to an obstacle.

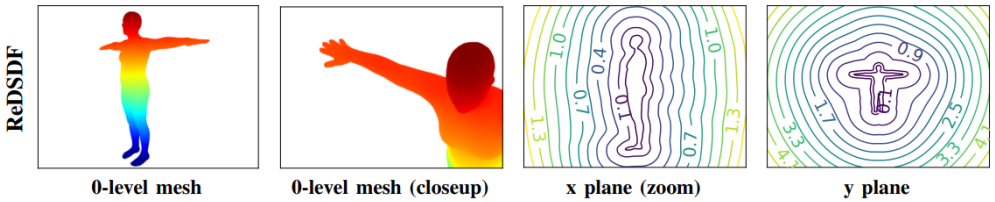


Figure 2.14: ReDSDF zero-level reconstruction and planar distance-field slices [21].

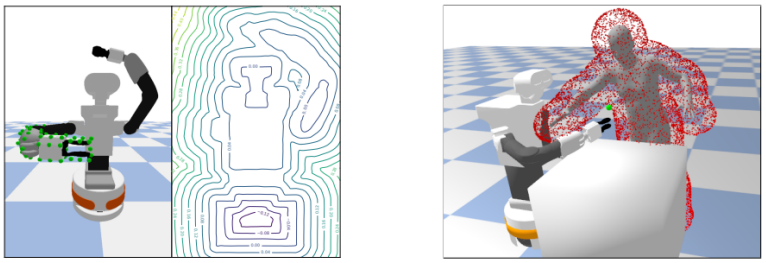


Figure 2.15: Reconstruction of the distance function used for robot avoidance during human–robot collaboration in ReDSDF [21].

2.6.2 Robot-conditioned and configuration-space fields

For an articulated robot, workspace distance alone is insufficient because the occupied body changes with the joint configuration. Neural-JSDF [22] explicitly conditions the representation on both configuration and query point. For $q \in \mathbb{R}^{n_q}$ and $y \in \mathbb{R}^3$, the learned function has the form

$$\Gamma(q, y) : \mathbb{R}^{n_q} \times \mathbb{R}^3 \longrightarrow \mathbb{R}^K, \qquad \Gamma_k(q, y) \approx d_{\mathcal{B}_k(q)}(y), \tag{2.16}$$

where $\mathcal{B}_k(q)$ is robot link k transformed by forward kinematics. The closest-body clearance, as shown in Figure 2.16, is obtained by minimizing over links and obstacle queries. Automatic differentiation provides $\nabla_q \Gamma_k(q, y)$, a joint-space direction that can be inserted directly in QP inverse kinematics or sampling-based model-predictive control.

In contrast to an end-effector-only distance, this representation can react to an obstacle near the forearm even when the gripper trajectory is clear.

Neural-JSDF is an important geometric foundation for this thesis, but it is not

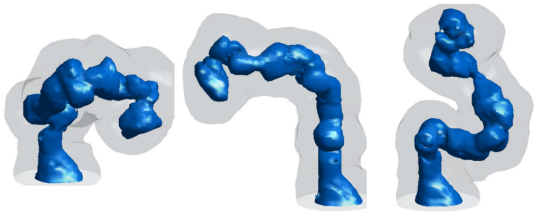


Figure 2.16: Illustration of the learned implicit distance isosurfaces $\Gamma(q, y) = 0$ cm (solid) and $\Gamma(q, y) = 10$ cm (transparent) for various configurations of the 7 DoF (Degrees-of-Freedom) Franka Emika Panda robot [22].

a complete VLA safety system. The field is robot-specific and approximate, and its published control evaluation represents obstacles by collections of spheres; a point-cloud implementation is left as future work.
The QP controller remains local and can stop in a local minimum, while the field has no notion of task semantics, permitted contact, a grasped payload, or the action parameterization used by a VLA. A separate perception and correction pipeline is still required.

Configuration Space Distance Fields (CDFs) [23] define distance to collision directly in robot configuration space, as shown in Figure 2.17, and expose gradients suitable for trajectory optimization. This avoids repeatedly interpreting a workspace

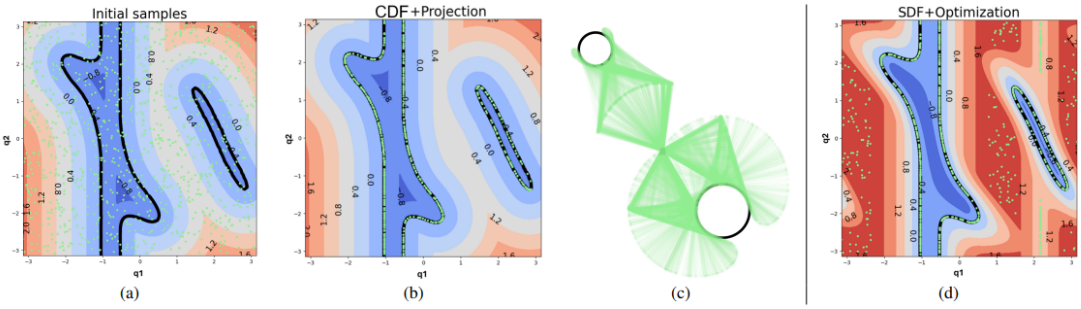


Figure 2.17: Comparison between CDF and SDF for solving a whole-body inverse-kinematics problem. (a) The initial sampled joint configurations. (b) Gradient projection by CDF. (c) Task space visualization of feasible solutions in (b). (d) Results for distance query-based method with L-BFGS optimizer. We can see that with the baseline SDF approach, the system can get stuck when the gradient of the SDF vanishes or reaches a singularity [23].

gradient for each link and makes the topology of the collision region explicit. CDFs can also be approximated and composed for manipulation planning.

Their challenge is the dimensionality and scene dependence of configuration space: changing the obstacle arrangement changes the collision boundary, and a dense field becomes expensive as the number of joints grows. CDF is therefore a powerful planning representation, but it is not itself a task-aware interface to a frozen generalist policy.

2.6.3 Implications for learned safety layers

The distance-field literature establishes three requirements for a useful learned safety signal. First, the value must correspond to metric clearance closely enough that a margin has physical meaning. Second, its gradient must remain well-conditioned near the safety boundary because this derivative determines the correction direction. Third, the representation must separate reusable robot geometry from scene geometry if it is to generalize to unseen layouts without scene-specific retraining. Even when these requirements hold, a field alone does not select task-relevant obstacles or decide how much of the nominal command to retain. Those roles belong to the semantic perception and action-filtering components.

2.7 Evaluation benchmarks

2.7.1 LIBERO

LIBERO [24], illustrated in Figure 2.18, evaluates language-conditioned manipulation and knowledge transfer across four suites: Spatial, Object, Goal, and Long. The suites vary spatial relations, object identity, goal conditions, and long-horizon composition.
Its standard outcome is *task success*: whether the policy reaches the environment-defined goal within the episode horizon. This makes LIBERO well suited to measuring instruction following and policy adaptation, and it is widely used by the VLA and geometry-aware training methods discussed above.
As discussed, task success alone is not a safety metric. Moreover, the original benchmark does not systematically place obstacles so that avoidance is required. High LIBERO success should therefore not be interpreted as evidence of collision-free execution.

2.7.2 SafeLIBERO

SafeLIBERO, introduced with VLSA/AEGIS [6], augments the semantic structure of LIBERO with obstacle interventions designed to expose this gap, as shown in Figure 2.19. The benchmark selects four tasks from each of the four LIBERO

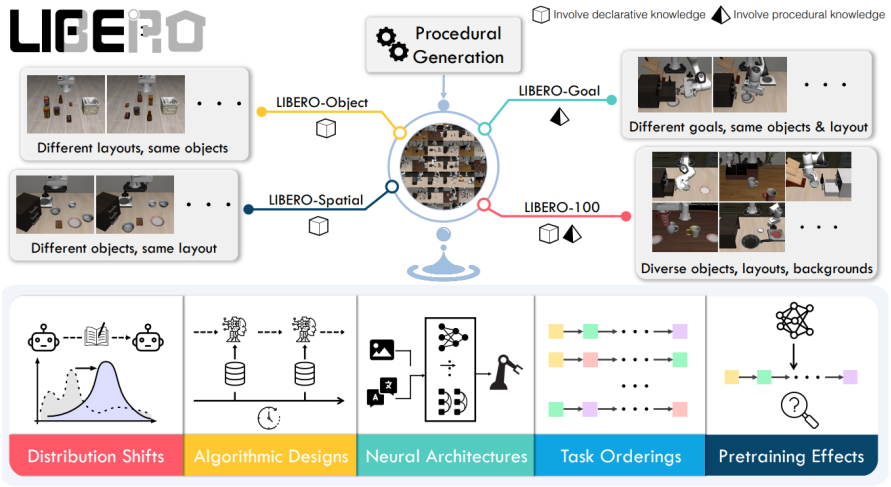


Figure 2.18: Overview of the LIBERO benchmark suites [24].



Figure 2.19: Comparison of the same tasks in the LIBERO dataset [24] and SafeLIBERO [6].

suites and defines two safety levels per task. Level I places an obstacle near the target region, testing whether the robot can preserve the contact needed for the task while avoiding a nearby distractor. Level II places an obstacle farther from the target but in the nominal approach route, testing trajectory-level avoidance.
SafeLIBERO separates task and safety outcomes. *Task success rate* (TSR) measures whether the requested manipulation is completed, whereas *collision avoidance rate* (CAR) measures the fraction of episodes without an unsafe collision. *Execution time steps* (ETS) provides an additional view of efficiency.
Reporting TSR and CAR separately prevents a collision followed by successful task completion from being counted as fully safe behaviour. It also reveals the opposite failure mode: an overly conservative filter may avoid every collision by preventing

useful motion. A credible safety layer must improve CAR while retaining as much TSR as possible.
SafeLIBERO remains a simulation benchmark and cannot capture every source of real-world uncertainty, including depth noise, calibration drift, compliant contact, and unmodelled dynamics. Nevertheless, it is directly relevant to the present problem because it preserves language-conditioned manipulation while making obstacle avoidance necessary and measurable.

2.8 Positioning of this thesis

The reviewed methods expose a clear design space. Training-time geometry can produce a stronger nominal policy, but requires modifying that policy and relies on learned generalization at deployment.
OmniGuide [9] preserves the weights and can introduce new geometric objectives online, but requires a differentiable generative sampler and provides soft guidance. AEGIS [6] offers the modularity and minimal-intervention semantics needed for a frozen black-box VLA, but its compact single-obstacle, end-effector-centric geometry does not cover cluttered full-arm manipulation.
Neural-JSDF [22] and configuration-space distance fields (CDFs) [23] provide the missing geometric derivatives, yet do not by themselves connect perception, task semantics, and the policy action interface. Therefore, the method developed in this thesis belongs to the *post-policy-correction* family. The nominal robotic policy remains frozen, and only its decoded command is modified.
The contribution is a reusable, policy-agnostic, scene-independent physics-informed geometric model supplying joint-space clearance gradients, while a task-aware execution-time layer uses them to minimally correct an otherwise unchanged generalist policy.
Chapter 3 introduces the theoretical background needed for this construction, and Chapter 4 presents the proposed methodology in detail. Finally, Chapter 5 presents the final results and Chapter 6 discusses the method's limitations.

Chapter 3

Theoretical Background

Artificial Intelligence (AI) is a broad field concerned with the design of systems that can perform tasks usually associated with intelligent behaviour, such as perception, reasoning, decision-making, and adaptation.
In robotics, AI methods enable robots to interpret sensory information, choose actions, and adapt their behaviour to different environments. Classical AI methods often relied on hand-designed rules or symbolic reasoning. In contrast, modern AI is largely based on machine learning, where the behaviour of a system is learned from data rather than being explicitly programmed for every possible situation. Machine learning can be understood as the process of fitting a parameterized model to data.
Given an input x, a model f_θ produces an output

$$\hat{y} = f_\theta(x), \tag{3.1}$$

where θ denotes the model parameters. The objective of learning is to choose the parameters θ such that the prediction $\hat{y}$ is close to the desired target y. In robotics, x may represent an image, a point cloud, a robot configuration, or a task instruction, while y may represent a manipulation action, a class label, or a desired locomotion.
Deep learning is a subset of machine learning based on artificial neural networks. A neural network is a composition of many simple computational units, called neurons, organized in layers. Each neuron, as shown in Figure 3.1, receives a vector of inputs, computes a weighted sum, adds a bias term, and applies a nonlinear activation function. For a single neuron, this operation can be written as

$$z = w^\top x + b, \tag{3.2}$$

$$\hat{y} = \sigma(z), \tag{3.3}$$

where x is the input vector, w is the vector of weights, b is the bias, $\sigma(\cdot)$ is a nonlinear activation function, and $\hat{y}$ is the neuron output.

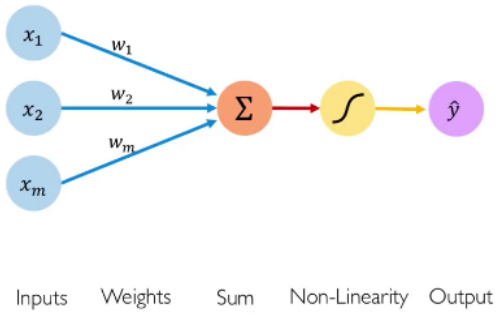


Figure 3.1: Schematic representation of a single artificial neuron.

The activation function is important because it allows the network to represent nonlinear relationships between inputs and outputs. Without nonlinear activations, a network composed of many layers would still behave like a single linear transformation.
A deep neural network, shown in Figure 3.2, is obtained by stacking many such layers.

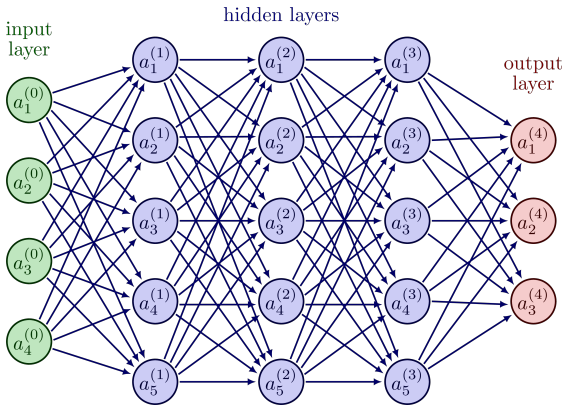


Figure 3.2: Example of a multilayer perceptron with multiple hidden layers.

If $a^{(0)} = x$ denotes the input, the output of layer l can be written as

$$a^{(l)} = \sigma\left(W^{(l)} a^{(l-1)} + b^{(l)}\right), \tag{3.4}$$

where $W^{(l)}$ and $b^{(l)}$ are the weights and biases of layer l.
The final network output is obtained after applying all layers:

$$\hat{y} = f_\theta(x) = a^{(L)}. \tag{3.5}$$

To train a neural network, one must define a loss function. The loss function measures how far the model prediction is from the desired target.
Given a dataset

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}, \tag{3.6}$$

a generic training objective is

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \ell(f_\theta(x_i), y_i), \tag{3.7}$$

where $\ell(\cdot)$ measures the error on a single sample.
The purpose of training is to find parameters that minimize the loss:

$$\theta^\star = \arg\min_\theta \mathcal{L}(\theta). \tag{3.8}$$

This optimization is usually performed with gradient-based methods. Starting from an initial set of parameters, the optimizer repeatedly updates the network in the direction that reduces the loss.
A simple gradient descent update is

$$\theta_{k+1} = \theta_k - \eta \nabla_\theta \mathcal{L}(\theta_k), \tag{3.9}$$

where η is the learning rate and $\nabla_\theta \mathcal{L}$ is the gradient of the loss with respect to the network parameters.
The gradients used in this update are computed through backpropagation. Backpropagation is an efficient application of the chain rule of calculus to neural networks. During the forward pass, the input is propagated through the network to produce a prediction and compute the loss. During the backward pass, the derivative of the loss is propagated backward through the layers, allowing the algorithm to compute how each weight and bias contributed to the final error.
Consider the scalar composition

$$h^{(1)} = f_1(x; \theta_1), \qquad h^{(2)} = f_2(h^{(1)}; \theta_2), \qquad \hat{y} = f_3(h^{(2)}; \theta_3). \tag{3.10}$$

For a parameter θ_1 belonging to the first layer, the chain rule gives

$$\frac{\partial \mathcal{L}}{\partial \theta_1} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial h^{(2)}} \frac{\partial h^{(2)}}{\partial h^{(1)}} \frac{\partial h^{(1)}}{\partial \theta_1}. \tag{3.11}$$

Backpropagation evaluates these products efficiently from the output layer to the input layer.

3.1 Large Language Models

Large Language Models (LLMs) are neural networks trained to process and generate natural language. Their central objective is to model text as a sequence of tokens. Tokens are discrete units obtained by splitting text into words, subwords, characters, or byte-level units. In an autoregressive language model, the model predicts the next token conditioned on the previous tokens:

$$p(x_t \mid x_1, x_2, \ldots, x_{t-1}). \tag{3.12}$$

By repeatedly applying this next-token prediction process, the model can generate complete sentences, answer questions, follow instructions, summarize text, or produce structured outputs such as code and JSON.
Most modern LLMs are based on the Transformer architecture [25], shown in Figure 3.3.

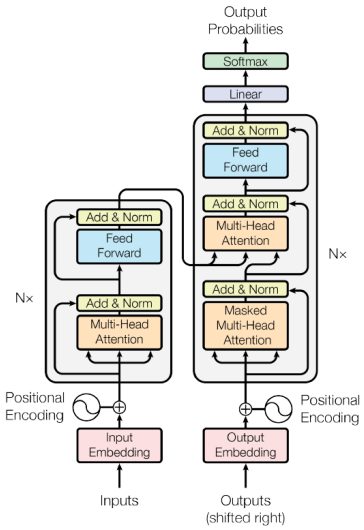


Figure 3.3: Attention mechanism in the Transformer architecture [25].

Given a sequence of token embeddings, the model computes query, key, and value vectors. The attention operation is defined as

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}} + M\right)V, \tag{3.13}$$

where Q, K, and V are the query, key, and value matrices, d_k is the key dimensionality, and M is the attention mask. In an autoregressive language model, $M_{ij} = -\infty$ for $j > i$, preventing a token from attending to future tokens.

An LLM is typically trained in two broad stages. The first stage is pre-training, where the model learns general linguistic patterns, factual associations, and reasoning behaviours from large-scale text corpora. The second stage is post-training, where the model is adapted to follow instructions and produce outputs aligned with user intent. Post-training often includes supervised fine-tuning and reinforcement learning with Verifiable Rewards (*RLVR*). After this process, the model can be prompted to perform tasks that were not explicitly hard-coded, such as extracting information and generating texts.

3.2 Vision-Language Models

Vision-Language Models (VLMs) extend language models by allowing them to process visual inputs together with text. As shown in Figure 3.4, instead of receiving only a sequence of text tokens, a VLM receives both an image and a textual prompt. The model then generates an answer conditioned on both modalities:

$$X_a = f_\theta(X_v, X_q), \tag{3.14}$$

where X_v is the input image, X_q is the textual prompt, and X_a is the generated output.

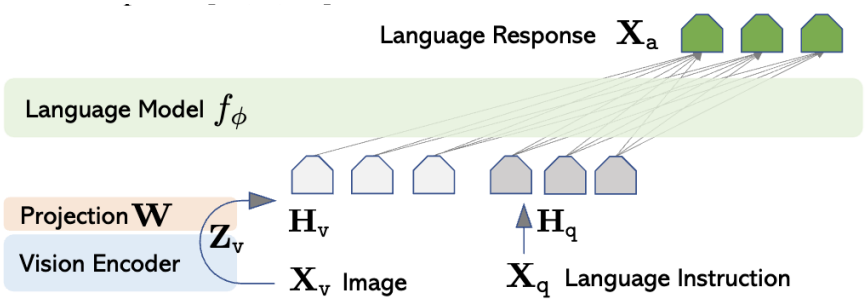


Figure 3.4: Vision–language architecture connecting a visual encoder to a language model through a learned projection [26].

A typical VLM contains three main components. The first is a vision encoder, which converts the image into visual features. In many recent architectures, the vision encoder is based on a Vision Transformer (ViT) [27]. The image is divided into patches, each patch is converted into an embedding, and the sequence of visual tokens is processed by attention layers. The second component is a projection or merging module, which maps the visual features into the same embedding space as the language model. The third component is the language model itself, which processes the combined visual and textual tokens and generates the output.

This architecture allows the model to reason jointly over what is written in the prompt and what is visible in the image. For example, a VLM can answer questions about image content, describe a scene, read text in an image, and identify relationships between objects.
VLMs are especially important in robotics because robot tasks are often specified through both language and perception. The language instruction defines the desired behaviour, while the image defines the current state of the environment. A text-only model can parse the instruction, but cannot know what is actually visible in the scene. A vision-language model can condition its answer on both the task description and the image.
However, VLMs also have limitations. Many VLMs generate textual outputs, and their visual grounding ability may not always be precise enough for geometric control. Even when a VLM can output object locations, these locations are often represented as text, bounding boxes, or points, rather than as dense geometric models.

3.3 Vision-Language-Action Models

Vision-Language-Action (VLA) models, as illustrated in Figure 3.5, are generalist robot policies that connect visual perception, language understanding, and action generation in a single learned system.

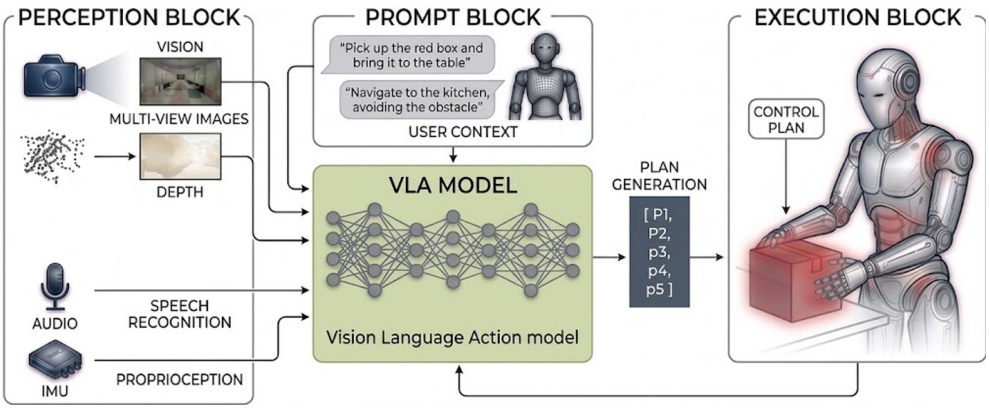


Figure 3.5: Architecture of the Vision-Language-Action (VLA) paradigm. The model receives a textual prompt and perceptual observations of the scene and produces robot actions.

Given visual observations o_t, the current robot configuration q_t, and a language

instruction ℓ, a VLA predicts a task-conditioned action:

$$a_t^{\mathrm{VLA}} = \pi_\theta(o_t, q_t, \ell), \tag{3.15}$$

where a_t^{VLA} denotes the action proposed by the policy.

The observation may contain one or more camera images, proprioceptive information, and other robot state variables, depending on the architecture and robot platform. Images, language tokens, proprioceptive states, and actions are converted into representations that can be processed by a large neural network, often based on a transformer architecture. The model is then trained to predict actions conditioned on the current observation and task instruction. This allows the policy to reuse semantic knowledge from large vision-language models while learning how to ground it in physical robot actions. Most VLA models are trained through imitation learning on robot demonstration datasets. Given a dataset of demonstrations

$$\mathcal{D} = \{(o_t, \ell, a_t)\}_{t=1}^{N}, \tag{3.16}$$

the model is optimized to assign high probability to the demonstrated action under the current observation and instruction:

$$\max_\theta \mathbb{E}_{(o_t,\ell,a_t)\sim\mathcal{D}} \left[\log \pi_\theta(a_t \mid o_t, \ell)\right]. \tag{3.17}$$

In practice, modern policies often predict an action chunk rather than a single action. An action chunk is a short sequence of future actions:

$$a_{t:t+H} = (a_t, a_{t+1}, \ldots, a_{t+H}), \tag{3.18}$$

where H is the prediction horizon. The corresponding objective becomes

$$\max_\theta \mathbb{E}_{(o_t,\ell,a_{t:t+H})\sim\mathcal{D}} \left[\log \pi_\theta(a_{t:t+H} \mid o_t, \ell)\right]. \tag{3.19}$$

Action chunking improves temporal consistency because the policy does not reason only about the next instantaneous command, but about a short future segment of motion. This is particularly useful in manipulation, where smooth reaching, grasping, and placing behaviours often require coordinated sequences of actions. A VLA model therefore combines two complementary capabilities. First, it provides semantic grounding: it can interpret the language instruction, identify task-relevant objects, and infer what behaviour is appropriate in the current scene. Second, it provides a learned action prior: it predicts actions that are likely to be useful based on the demonstrations seen during training. This makes VLA models attractive for generalist manipulation, where the same policy should handle many tasks, objects, and scene layouts.

The experimental evaluation uses two base policies to produce nominal actions: OpenVLA-OFT and $\pi_{0.5}$. The proposed safety module is placed after either policy and therefore does not modify its internal architecture or training procedure.

3.3.1 OpenVLA: An Open-Source Vision-Language-Action Model

OpenVLA is an open-source Vision-Language-Action model with the objective of providing a general-purpose robotic policy that can be openly accessed, fine-tuned, and deployed on different robotic platforms [3].
The main idea behind OpenVLA, as shown in Figure 3.6, is to reuse the visual and semantic representations learned by large pretrained Vision-Language Models and adapt them directly to robot control. Instead of constructing a robotic policy entirely from scratch, OpenVLA starts from a pretrained multimodal model and fine-tunes it to predict robot actions from visual observations and natural-language instructions. In this way, knowledge acquired from large-scale vision and language pretraining can be transferred to robotic manipulation, improving generalization across objects, scenes, and task instructions.

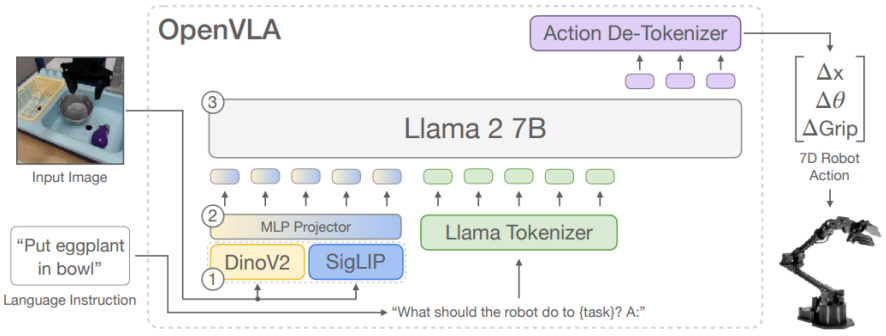


Figure 3.6: Architecture of OpenVLA [3]. Visual features extracted by the DINOv2 [28] and SigLIP [29] encoders are projected into the embedding space of a Llama 2 [30] language model, which autoregressively predicts robot action tokens.

As illustrated in Figure 3.6, the architecture consists of three main components: a visual encoder, a projector, and a large language model backbone. The visual encoder combines features extracted by two pretrained models, SigLIP and DINOv2. SigLIP provides high-level semantic visual information, while DINOv2 contributes more fine-grained spatial features, which are particularly useful for robotic manipulation. The resulting visual representations are concatenated and passed through a two-layer MLP projector that maps them into the embedding space of a 7B-parameter Llama 2 language model.

Given a visual observation o_t and a natural-language instruction ℓ, OpenVLA predicts the robot action at time t,

$$\pi_\theta(a_t \mid o_t, \ell), \tag{3.20}$$

where a_t is a seven-dimensional robot control action. The action controls the motion of the robot end-effector together with the gripper state. Continuous robot actions are converted into discrete tokens that can be generated directly by the Llama 2 backbone. The model generates these tokens autoregressively using the same next-token prediction mechanism employed for natural language generation. During training, a standard cross-entropy objective is applied only to the predicted action tokens. An important design choice is that the visual encoder is not frozen during robot training.

Despite these advantages, the original OpenVLA formulation presents important limitations for real-time robot control. First, the model predicts only a single action at each inference step. Moreover, because the seven action dimensions are represented as tokens, they are generated sequentially through autoregressive decoding. Predicting several future actions would therefore require generating a long sequence of action tokens one after another. This limits inference throughput and makes techniques such as action chunking difficult to implement efficiently.

These limitations motivate the later introduction of *OpenVLA-OFT*, an optimized fine-tuning formulation of OpenVLA proposed by Kim et al. [17]. OpenVLA-OFT does not replace the pretrained OpenVLA foundation model; instead, it changes how OpenVLA is adapted to downstream robot tasks. In particular, the Optimized Fine-Tuning (OFT) recipe replaces autoregressive action generation with parallel decoding, predicts chunks of future actions simultaneously, represents actions directly as continuous values, and trains the action predictions using an ℓ_1 regression objective. These modifications substantially reduce the computational cost of action generation while also enabling temporally consistent action chunks and higher-frequency control.

3.3.2 $\pi_{0.5}$: a Vision-Language-Action Model with Open-World Generalization

The $\pi_{0.5}$ model is a Vision-Language-Action model introduced by Physical Intelligence with the objective of achieving open-world generalization in robotic manipulation [16]. It builds on the earlier π_0 architecture and extends it through the heterogeneous data sources summarized in Figure 3.7. The main idea of the paper is that a generalist robot policy should not learn only from direct robot demonstrations collected on a single platform. Instead, it should transfer knowledge from many sources, including data from different robots, high-level semantic task annotations, verbal instructions, object detection tasks, and multimodal web data.



Figure 3.7: Overview of the heterogeneous data sources and policy components used by $\pi_{0.5}$ [16].

This design is motivated by the difficulty of deploying robots outside controlled laboratory environments. A policy trained only on a narrow set of demonstrations may learn useful low-level skills, but may fail when the same skills must be applied to a new object, in a new room, or as part of a longer task.
Architecturally, $\pi_{0.5}$ follows the structure of modern multimodal transformer-based models. Its inputs are represented as multimodal tokens, including image tokens, text tokens, proprioceptive state tokens, and action-related tokens. The model is initialized from a pretrained vision-language backbone, which provides visual and semantic understanding. On top of this backbone, the model learns to process robot observations and generate both text outputs and action outputs.
The training recipe of $\pi_{0.5}$, as shown in Figure 3.8, is divided into two main phases. In the pre-training phase, the model is trained on a broad mixture of data sources. These include mobile manipulation data, non-mobile robot data collected in diverse environments, cross-embodiment laboratory data, high-level subtask prediction data, and multimodal web data such as captioning, visual question answering, and object localization. The purpose of this stage is to expose the model to a wide range of visual, semantic, and robotic behaviours. In the post-training phase, the model is specialized for mobile manipulation and equipped with a continuous action expert mechanism. This stage keeps the language and semantic capabilities of the pretrained model while improving the quality and efficiency of low-level action

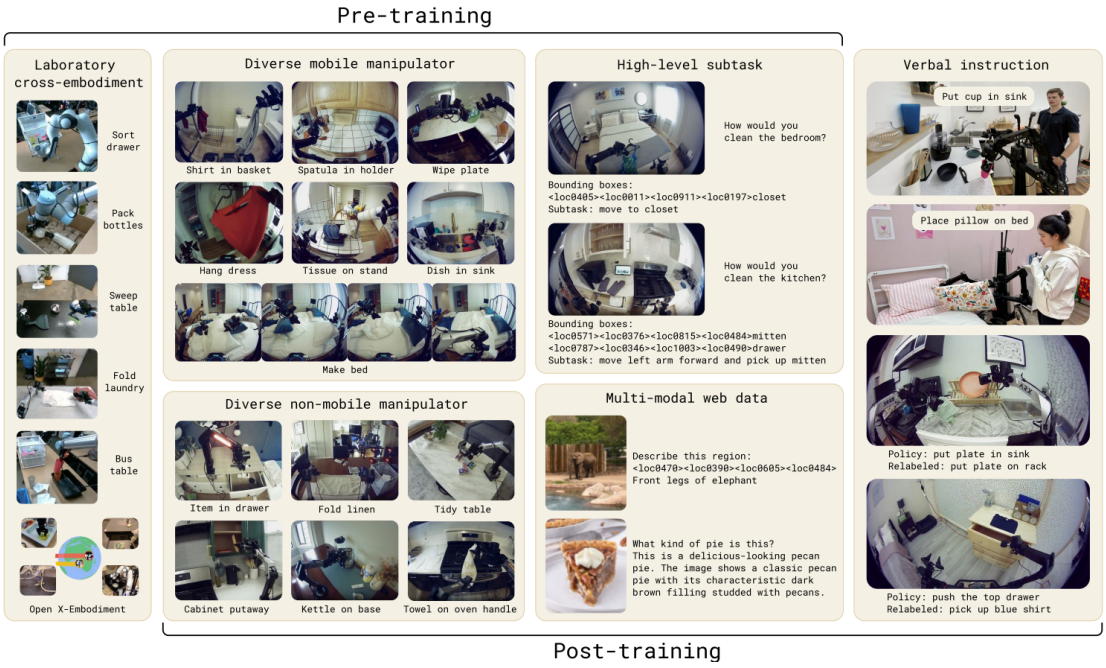


Figure 3.8: Pre-training and post-training data composition of $\pi_{0.5}$ [16].

generation.
The main advantage of $\pi_{0.5}$ is its ability to combine several levels of generalization. At the low level, the model learns manipulation behaviours from robot action data. At the high level, it learns to decompose complex instructions into subtasks. At the semantic level, web and vision-language data improve its ability to recognize objects, interpret scenes, and follow language instructions. This combination allows the model to perform tasks in new environments that were not seen during training. For this thesis, $\pi_{0.5}$ is useful because it represents a strong nominal policy. It can generate meaningful task-directed actions from visual observations and language instructions, and it can perform non-trivial manipulation behaviours without requiring a task-specific controller.

3.3.3 Final considerations

Together, these foundations explain how modern VLA policies transform multimodal observations and language instructions into task-directed robot actions. The following chapter introduces PhySVLA, which preserves this nominal policy behaviour while adding explicit, physics-informed geometric correction at execution time.

Chapter 4

Methodology

This chapter describes the methodology developed to add execution-time collision avoidance to a pretrained Vision-Language-Action policy. The proposed system separates task competence from geometric safety. The VLA policy remains responsible for interpreting the instruction and generating a task-directed command, while a learned distance field and a Control Barrier Function (CBF) filter determine whether that command can be executed safely.

The complete method consists of an offline and an online phase. In the offline phase, a scene-independent dataset is generated by sampling configurations of a seven-degree-of-freedom Franka Emika Panda, shown in Figure 4.1, and query points in its workspace.
Exact distances to a capsule-and-sphere surrogate of the robot are used to train a neural Joint Space Distance Field (JSDF) [22].
In the online phase, depth observations are converted into a point cloud of task-relevant obstacles. The trained JSDF evaluates the clearance between this point cloud and the articulated robot and provides the gradient of the safety function with respect to the joint configuration. This gradient is mapped to the operational-space action used by the VLA, and a CBF filter applies the smallest correction required by the active safety constraints.

Let $q \in \mathbb{R}^7$ denote the arm configuration, $y \in \mathbb{R}^3$ a point in the workspace, and

$$h_\theta(q, y) = \begin{bmatrix} d_{\theta,1}(q, y) & \cdots & d_{\theta,K}(q, y) \end{bmatrix} \tag{4.1}$$

the learned vector of signed distances from y to the K geometric components used to approximate the robot.
In this work $K = 9$: eight capsule channels represent consecutive arm segments, and one spherical channel represents the end-effector.



Figure 4.1: Joint arrangement of the seven-degree-of-freedom Franka Emika Panda. Source: [31].

Given an obstacle point set $\mathcal{Y}_t = \{y_{t,n}\}_{n=1}^{N_t}$, the field is reduced to a safety barrier $h(q_t, \mathcal{Y}_t)$. Its sign follows the convention

$$h > 0 \text{ safe}, \qquad h = 0 \text{ safety boundary}, \qquad h < 0 \text{ violation}. \tag{4.2}$$

This sign convention is visualized in Figure 4.2.

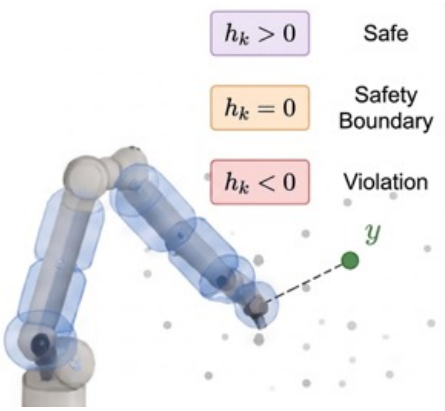


Figure 4.2: Signed distance function (SDF) convention.

The nominal VLA command is modified only if it would reduce this barrier too quickly. Consequently, the learned field is not used as a replacement motion planner. It is a differentiable geometric model that supplies the local information required by the safety filter.

4.1 Design objectives and scope of generalization

The methodology is guided by four design objectives.
First, the safety mechanism must be *policy-preserving*: it should not require access to the parameters, training data, or gradients of the nominal VLA. This makes it possible to keep the semantic and manipulation knowledge already encoded by the pretrained policy.
Second, it must be *scene-free*: a field trained once on robot geometry should accept a different obstacle set at every control step.
Third, it must be *arm-body aware*: the relevant distance is not only the clearance of the end-effector, since an intermediate link can collide while the gripper follows a valid trajectory. In this implementation, "body" denotes the seven-DoF arm plus a terminal gripper proxy; articulated fingers are not modelled.
Fourth, it must satisfy *minimal intervention*: task-directed motion should be preserved whenever it is compatible with the local safety constraints.

These objectives explain the separation between the learned function and the online scene representation:

$$\underbrace{h_\theta(q, y)}_{\text{trained once for the robot}} \quad + \quad \underbrace{\mathcal{Y}_t}_{\text{new scene at time } t} \quad \longrightarrow \quad h_\theta(q_t, \mathcal{Y}_t). \tag{4.3}$$

The obstacle set is an input to the already trained field rather than a property encoded in its parameters. Consequently, a new object shape, obstacle layout, or manipulation task does not by itself require new safety-model training: its geometry is supplied at inference time through new query points. This is the specific sense in which the safety layer is used *zero-shot*.
The scope of this claim is deliberately limited. The model is zero-shot with respect to scenes and tasks for the same Panda embodiment, kinematic convention, and calibrated workspace. It is not zero-shot with respect to a robot with different kinematics or link geometry, for which the FK module and geometric training data would have to be changed. Furthermore, zero-shot geometric reuse does not imply zero-shot semantic perception: the online perception system must still identify which visible regions are obstacles and which objects are intended contact targets. Stating this boundary is important because the generality comes from factorizing robot geometry from scene geometry, not from claiming that every component of the robotic system generalizes without assumptions.

4.2 Making the method scene-free and zero-shot

The safety model is trained on a *scene-free* dataset.
No manipulation rollouts, obstacle layouts, RGB images, or language instructions

are collected. Instead, the dataset describes only the geometry and kinematics of the robot. This choice is central to the proposed methodology. If training samples were collected from a finite set of SafeLIBERO scenes, the learned field could associate safety with particular object arrangements or task trajectories.
By sampling robot configurations and workspace points independently of any scene, the model learns the reusable function $h(q, y)$. At inference time, arbitrary obstacle geometries can then be represented as sets of query points.
This formulation also avoids collecting unsafe demonstrations. A scene-specific dataset would require moving the robot close to many obstacles, including collision states, to supervise the boundary adequately. Such collection is costly on hardware and strongly biased by the exploration policy: configurations that the policy never visits would receive no supervision. In contrast, the scene-free generator obtains exact surrogate-geometry labels without executing a task and can deliberately include penetrations and rare robot configurations. The resulting coverage is therefore determined by a controlled sampling distribution rather than by the behaviour distribution of either evaluation policy.

The dataset is generated in MuJoCo using the same Franka Panda model employed during evaluation, shown in Figure 4.1. For every sampled configuration, MuJoCo provides the reference forward kinematics and positional Jacobians. This avoids a mismatch between the kinematic convention used to label the data and the one used by the simulated controller. It also yields deterministic, noise-free geometric labels, which isolate errors of the learned field from errors of manual annotation.

4.2.1 Sampling the configuration space

The lower and upper joint limits, $q^{\min}$ and $q^{\max}$, are read from the simulator. Their interval is expanded by a fraction $\eta_q = 0.05$, as in the broad configuration-space sampling strategy used for Neural-JSDFs [22]. Defining

$$q^{\text{mid}} = \frac{q^{\min} + q^{\max}}{2}, \qquad q^{\text{half}} = \frac{1 + \eta_q}{2} \left(q^{\max} - q^{\min} \right), \tag{4.4}$$

each configuration is sampled according to

$$q_m \sim \mathcal{U} \left(q^{\text{mid}} - q^{\text{half}}, q^{\text{mid}} + q^{\text{half}} \right). \tag{4.5}$$

Uniform sampling is preferred to sampling configurations from task rollouts because the latter would over-represent a small set of reaching and grasping postures.
The safety gradient is needed precisely when the nominal policy is perturbed away from its usual trajectory; training only on that trajectory would leave the corrective

field least constrained where it is most needed. Uniform sampling instead makes no assumption about the downstream task distribution and supports reuse across SafeLIBERO tasks.
The expansion exposes the model to a small neighbourhood beyond the nominal limits and reduces boundary effects in the learned representation. Without it, configurations close to a joint limit would lie at the edge of the training support, where neural derivatives are generally less reliable. It does not authorize the runtime controller to violate the physical joint limits, which remain independently enforced. The value 5% is therefore a robustness buffer for function approximation, not an enlargement of the admissible control set.

The dataset contains $M = 5000$ configurations and $P = 1000$ query points for each configuration, resulting in five million point–configuration pairs. For every q_m, the simulator computes the positions of nine ordered keypoints,

$$\mathcal{P}(q_m) = \{p_0(q_m), p_1(q_m), \ldots, p_8(q_m)\}\,, \tag{4.6}$$

corresponding to the Panda links from the base to the gripper end-effector. It also records the positional Jacobian

$$J_i(q) = \frac{\partial p_i(q)}{\partial q} \in \mathbb{R}^{3\times 7} \tag{4.7}$$

for each keypoint. These quantities support the distance labels, the on-the-fly local normals used during FK training, and the stored joint-gradient diagnostics.
The two sampling axes serve different purposes. Increasing M improves coverage of robot poses and therefore of kinematic Jacobians, whereas increasing P improves spatial coverage around a fixed pose. The chosen $M = 5000$, $P = 1000$ budget balances these requirements with HDF5 size and second-order automatic-differentiation cost during gradient supervision.

4.2.2 Workspace-point sampling

Uniform sampling alone would allocate most points far from the robot, where distance prediction is comparatively easy and less relevant to collision avoidance. The query distribution is therefore a mixture:

$$y \sim \rho_{\text{near}}\, p_{\text{near}}(y \mid q) + (1 - \rho_{\text{near}})\, p_{\text{uniform}}(y), \qquad \rho_{\text{near}} = 0.5. \tag{4.8}$$

Near-surface points are generated by first selecting an arm segment, sampling a point along its centreline, and applying a random radial displacement. The complete sampling procedure is illustrated in Figure 4.3.

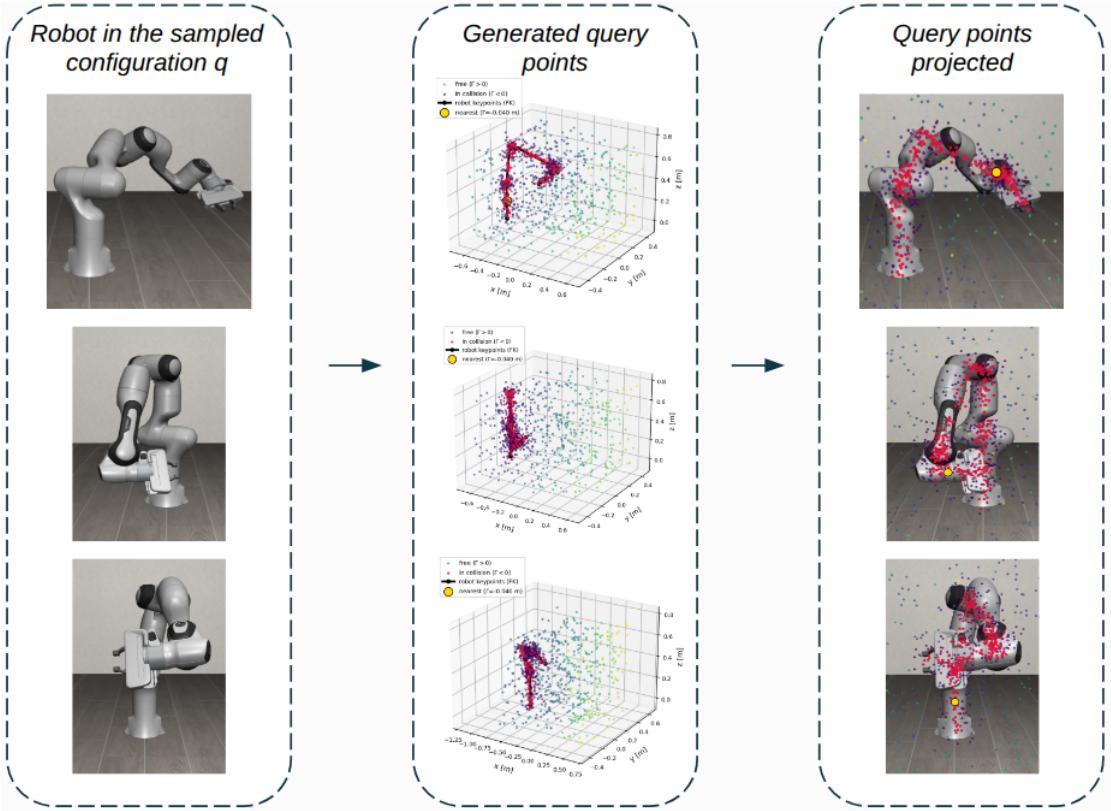


Figure 4.3: Dataset collection procedure used to make the model *scene-free* and *zero-shot*. For each of the M sampled robot configurations, P workspace points are generated. Red points indicate samples in collision with the robot, while non-colliding points are color-coded according to their distance from the robot. The closest sampled point is highlighted in yellow, and the robot keypoints are shown in black.

For the segment from p_i to p_{i+1},

$$c_i(\lambda) = p_i + \lambda(p_{i+1} - p_i), \qquad \lambda \sim \mathcal{U}(0,1), \tag{4.9}$$

and

$$y = c_i(\lambda) + \rho v, \qquad \|v\|_2 = 1, \qquad \rho \sim \mathcal{U}(0, r_i + b). \tag{4.10}$$

Here $r_i = 0.04\,\text{m}$ is the nominal capsule radius and $b = 0.06\,\text{m}$ is the boundary band.

This stratification gives the network dense supervision around contact and around the future safety boundary, while the uniform component preserves coverage of the complete workspace.
The equal mixture is a deliberate compromise. Using only near-surface points

would accurately locate $h = 0$ but leave the field unconstrained when an obstacle first enters the activation region. Using only uniform points would make the comparatively thin collision boundary rare and could produce a low average error together with unsafe sign errors. The mixture couples local precision to global spatial coverage.
The workspace box is shared by data generation and deployment. This prevents the network from being queried far outside its spatial training support. It also bounds the zero-shot claim: novel obstacle shapes and arrangements are supported inside this volume, not at arbitrary uncalibrated world coordinates.

4.2.3 Geometric labels

The articulated robot is approximated by eight capsules and an end-effector sphere. Capsules are used because they offer a differentiable and inexpensive approximation of elongated links, while representing substantially more of the robot body than an end-effector-only model. They also admit an exact point-to-segment distance. Exact collision meshes would model the Panda more faithfully, but point-to-mesh distance and its derivative are more expensive.
Capsules are conservative only if the selected radii actually enclose the relevant collision mesh; this is a calibration property, not an automatic consequence of using capsules. The end-effector receives a separate sphere because it is both geometrically compact and functionally distinct: it is the part of the robot expected to make intentional contact.

For a query point y, let

$$s_i = p_i(q), \qquad e_i = p_{i+1}(q), \qquad \ell_i = e_i - s_i. \tag{4.11}$$

The projection coefficient and closest centreline point are

$$\lambda_i^* = \operatorname{clip}\left(\frac{(y - s_i)^\top \ell_i}{\|\ell_i\|_2^2}, 0{,}1\right), \qquad c_i^* = s_i + \lambda_i^* \ell_i. \tag{4.12}$$

The signed distance to capsule i is then

$$d_i(q, y) = \|y - c_i^*\|_2 - r_i, \qquad i = 0, \ldots, 7, \tag{4.13}$$

while the end-effector channel is labelled as

$$d_{\text{ee}}(q, y) = \|y - p_8(q)\|_2 - r_{\text{ee}}, \qquad r_{\text{ee}} = 0.04\,\text{m}. \tag{4.14}$$

A negative value denotes a point inside the corresponding primitive.
The nominal configuration uses the same 0.04 m radius for the links and end-effector;

a dedicated radius can instead be assigned to the terminal segment because the wrist and gripper are wider than some of the arm links. Keeping this radius distinct prevents the entire arm approximation from becoming unnecessarily conservative. Distances are retained per primitive instead of immediately reducing them to one robot-wide minimum. A scalar minimum would answer whether *some* part of the robot is close to an obstacle, but it would discard which part is responsible.
The nine-channel representation allows the runtime to use different activation policies for the arm and end-effector and to preserve several simultaneous contacts through top-m constraints. This added structure is necessary for manipulation, where contact near the gripper and contact near the elbow have different task meanings.
The collision geometry and the safety margin have separate roles. The network learns distances to primitives of radius r_i; at runtime, a margin $m_s = 0.02\,\text{m}$ moves the zero level set outward:

$$h_i(q, y) = d_i(q, y) - m_s. \tag{4.15}$$

Thus, with the nominal 0.04 m radius, $h_i = 0$ corresponds to a centreline-to-obstacle distance of 0.06 m. The resulting safety envelope is illustrated in Figure 4.4 and instantiated in SafeLIBERO in Figure 4.5.

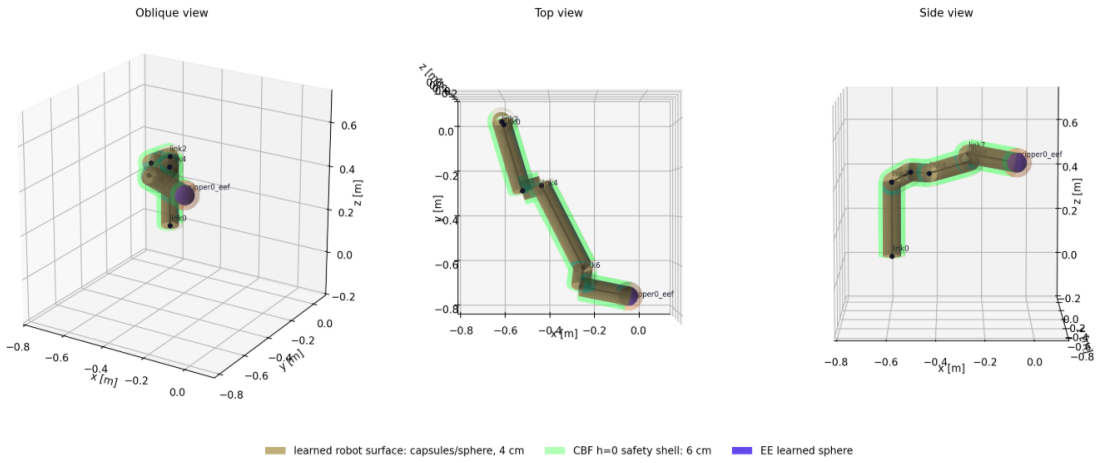


Figure 4.4: Representation of the robot geometry and safety margin in three-dimensional space.

Separating the learned surface from the operational margin makes the clearance adjustable without regenerating the dataset.
The field depends on the seven arm joints but not on the gripper-opening state. Finger bodies are therefore represented only through the fixed terminal capsule-sphere envelope.

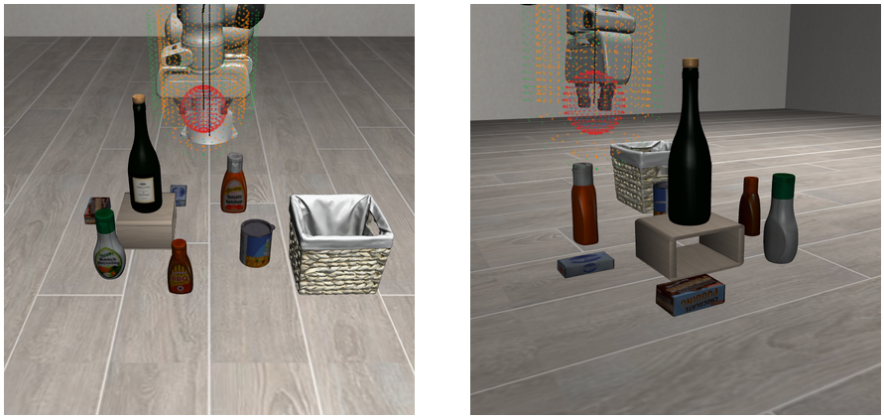


Figure 4.5: Robot geometry and safety margin in a SafeLIBERO task environment [6].

The selected FK training path recomputes local spatial-gradient targets from the stored keypoints. If $p^*(q)$ is the closest point on a robot primitive and

$$n = \frac{p^*(q) - y}{\|p^*(q) - y\|_2} \tag{4.16}$$

is the direction that moves the robot away from the query point, the chain rule gives

$$\frac{\partial d(q, y)}{\partial q} = J_{p^*}(q)^\top n. \tag{4.17}$$

The principal tensors are $q \in \mathbb{R}^{M\times 7}$, query points in $\mathbb{R}^{M\times P\times 3}$, and signed distances in $\mathbb{R}^{M\times P\times 9}$.

4.3 Training phase

The objective of the training phase is not only to reproduce signed-distance values. The runtime controller requires the direction $\nabla_q h$, and an accurate scalar field can still have locally inaccurate derivatives. The design of the network and the loss therefore explicitly preserve the kinematic structure that determines this gradient. The model was trained on an NVIDIA RTX 5090 GPU.

4.3.1 PhySVLA: model architecture overview

The proposed model, denoted *PhySVLA*, is shown in Figure 4.6.
PhySVLA factors the prediction into a differentiable forward-kinematics module

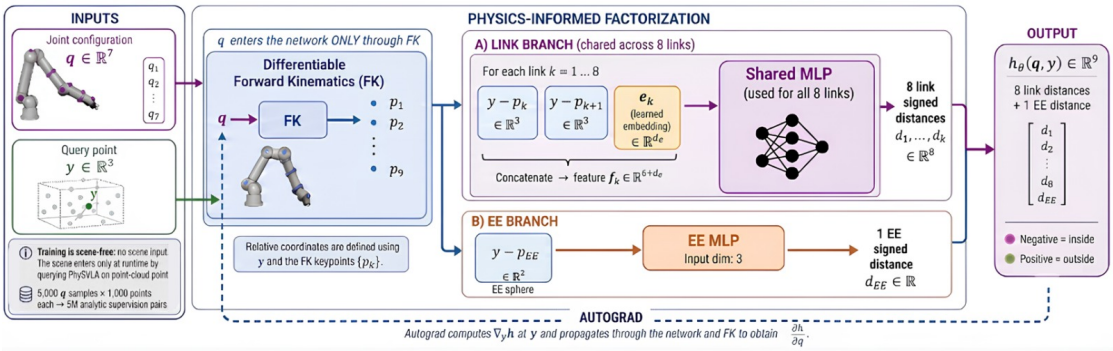


Figure 4.6: Architecture of the *PhySVLA* model.

and local geometry networks. Its essential constraint is that the raw configuration q enters the network *only* through forward kinematics:

$$q \xrightarrow{\text{FK}} \{p_i(q)\}_{i=0}^{8} \xrightarrow{\text{local features}} h_\theta(q, y). \tag{4.18}$$

The differentiable Panda kinematic chain predicts the same nine keypoints used during dataset generation.
For every arm segment, the local feature is

$$\varphi_i(q, y) = [y - p_i(q),\ y - p_{i+1}(q),\ e_i]\,, \tag{4.19}$$

where $e_i \in \mathbb{R}^8$ is a learned link embedding.
A shared multilayer perceptron with four hidden affine layers, SiLU activations, and one scalar output layer maps φ_i to the distance $\hat{d}_i$.
A second MLP processes the end-effector-relative feature

$$\varphi_{\text{ee}}(q, y) = y - p_8(q) \tag{4.20}$$

and predicts $\hat{d}_{\text{ee}}$.
Sharing the arm MLP makes geometrically similar segments contribute to the same local shape estimator, improving parameter efficiency.
The link embedding prevents this sharing from incorrectly forcing every segment to have identical geometry: it supplies a compact identity-dependent correction.
The end-effector uses a distinct branch because its spherical approximation and its intentional-contact role differ from those of an arm capsule.
The local MLPs only need to learn the configuration-independent shape of a link, while the exact dependence on q is carried by the differentiable kinematic chain.
SiLU activations [32] are used because their smooth, continuous derivatives are better suited to fields whose gradients are used by a controller, unlike ReLU derivatives, which are discontinuous at zero, as illustrated in Figure 4.7.

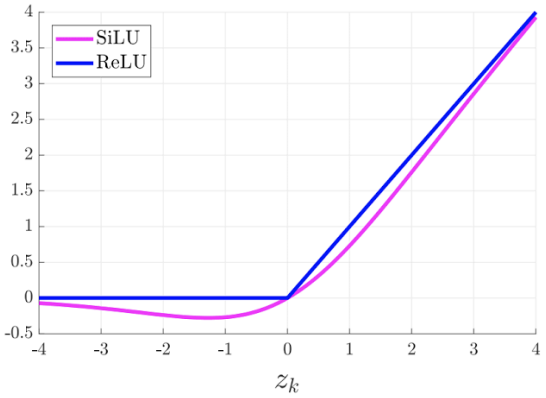


Figure 4.7: Comparison between the SiLU and ReLU activation functions.

Consequently,

$$\frac{\partial \hat{d}_i}{\partial q} = \frac{\partial \hat{d}_i}{\partial \varphi_i} \frac{\partial \varphi_i}{\partial (p_i, p_{i+1})} \frac{\partial (p_i, p_{i+1})}{\partial q}, \tag{4.21}$$

and the last factor is an FK Jacobian rather than an implicit relation that the MLP must rediscover from samples.
This inductive bias improves gradient generalization across the joint space and reduces the amount of geometric structure that must be learned.
It also states explicitly where generalization is expected to originate: new q values are handled by kinematics, while new obstacle shapes are handled by evaluating the same local function at new y values.

4.3.2 Barrier construction

At runtime, a scene contains a variable number of obstacle points. The predicted distances are therefore reduced with a differentiable soft minimum. For values $\{z_n\}_{n=1}^{N}$, the implementation uses the weighted form

$$\mathrm{smin}_\tau(\{z_n\}) = \sum_{n=1}^{N} w_n z_n, \qquad w_n = \frac{\exp(-z_n/\tau)}{\sum_j \exp(-z_j/\tau)}, \tag{4.22}$$

with temperature $\tau = 0.02$.
Compared with a hard minimum, this operation remains differentiable when the identity of the closest point changes.
The temperature controls a bias–smoothness trade-off. As τ decreases, the result approaches the true minimum but its gradient can switch sharply between neighbouring points; a larger τ is smoother but blends non-critical distances and can

overestimate the influence of nearby clutter.
Because Equation (4.22) is a convex weighted average, it is greater than or equal to the hard minimum.
It can therefore overestimate the true closest clearance and is not conservative by construction. The explicit safety margin and the local top-m constraints compensate empirically for this bias, but do not turn the approximation into a mathematical lower bound.
For each channel,

$$\bar{d}_i(q, \mathcal{Y}) = \mathrm{smin}_\tau \left(\{\hat{d}_i(q, y_n)\}_{n=1}^{N} \right). \tag{4.23}$$

The end-effector and arm barriers are

$$h_{\mathrm{ee}} = \bar{d}_{\mathrm{ee}} - m_s, \tag{4.24}$$

$$h_{\mathrm{arm}} = \mathrm{smin}_\tau \left(\{\bar{d}_i\}_{i=0}^{7} \right) - m_s, \tag{4.25}$$

and the global barrier is

$$h_{\mathrm{total}} = \mathrm{smin}_\tau \left(h_{\mathrm{ee}}, h_{\mathrm{arm}} \right). \tag{4.26}$$

The separate values are retained because the end-effector and the rest of the arm play different roles during manipulation.

4.3.3 Training objectives

The total training objective combines distance regression, sign classification and a local gradient alignment:

$$\mathcal{L} = \lambda_d \mathcal{L}_{\mathrm{dist}} + \lambda_s \mathcal{L}_{\mathrm{sign}} + \lambda_g \mathcal{L}_{\mathrm{grad}} \tag{4.27}$$

The distance targets are clipped to $[d_{min}, d_{max}]$ m, with $d_{min} = -0.05$, $d_{max} = 0.30$. Modelling very large positive distances precisely does not improve local collision avoidance, whereas errors near zero directly affect activation of the filter. Clipping negative values also prevents rare deep penetrations from dominating the objective even though the controller only needs a reliable escape direction once the boundary is crossed.
Each target distance d is assigned a weight $w(d)$:

$$w(d) = 1 + a \exp\left(\frac{|d|}{d_{min}} \right) + b\, \mathbb{I}[d < 0], \tag{4.28}$$

which emphasizes the boundary and further penalizes errors inside the robot geometry with $a = 4$, $b = 2$. The regression term is a weighted Smooth-L_1 loss,

$$\mathcal{L}_{\mathrm{dist}} = \frac{1}{|\mathcal{V}|} \sum_{(m,n,i) \in \mathcal{V}} w(d_{mni})\ \mathrm{SmoothL1}\left(\hat{d}_{mni}, \mathrm{clip}(d_{mni}, d_{min}, d_{max}) \right), \tag{4.29}$$

where $\mathcal{V}$ is the set of valid entries.
Smooth-L_1 is less sensitive to occasional large errors than a squared loss while remaining quadratic within its $\beta = 0.01\,\mathrm{m}$ transition and linear for larger residuals. This is useful because the uniform component of the dataset produces many easy, far-field samples whose absolute error should not overwhelm small but safety-critical errors near zero.
Distance regression is complemented by a binary sign loss:

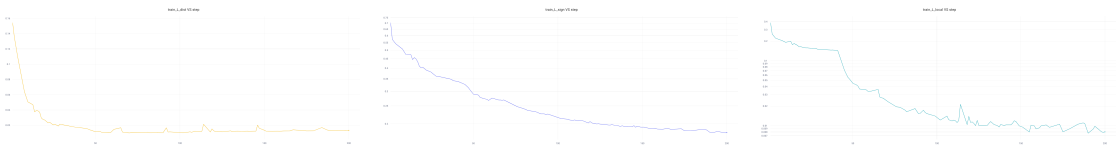
$$\mathcal{L}_{\text{sign}} = \frac{1}{|\mathcal{V}|} \sum_{(m,n,i)\in\mathcal{V}} w(d_{mni})\ \text{BCEWithLogits}\left(\beta_s \hat{d}_{mni}, \mathbb{I}[d_{mni} > 0]\right). \tag{4.30}$$

This term sharpens the learned zero level set and directly discourages *false-safe* predictions, for which the network predicts a positive distance for a colliding sample. The implemented weights are $\lambda_d = 1$ and $\lambda_s = 0.2$.
Regression and sign classification are not redundant: two predictions can have similar metric error while falling on opposite sides of zero, but only one leads to the correct collision decision.
The factor $\beta_s = 30\,\mathrm{m}^{-1}$ converts distances measured in metres into logits with a sufficiently sharp transition around zero.
The third term supervises the local spatial derivative.
For a sampled query point and geometric channel, the exact direction $g_y^* = \partial d/\partial y$ is the unit vector pointing from the closest point on the robot primitive toward y. The model derivative $\hat{g}_y = \partial \hat{d}/\partial y$ is obtained by *automatic differentiation* as shown in Figure 4.6, and their directions are aligned through

$$\begin{aligned}\mathcal{L}_{\text{grad}} &= \frac{1}{|\mathcal{S}|} \sum_{(m,n,i)\in\mathcal{S}} \left(1 - \frac{\left(\dfrac{\partial \hat{d}_{mni}}{\partial y}\right)^{\top} \dfrac{\partial d_{mni}}{\partial y}}{\left\|\dfrac{\partial \hat{d}_{mni}}{\partial y}\right\|_2 \left\|\dfrac{\partial d_{mni}}{\partial y}\right\|_2 + \epsilon}\right) \\ &= \frac{1}{|\mathcal{S}|} \sum_{(m,n,i)\in\mathcal{S}} \left(1 - \frac{\hat{g}_{y,mni}^{\top} g_{y,mni}^*}{\|\hat{g}_{y,mni}\|_2 \|g_{y,mni}^*\|_2 + \epsilon}\right),\end{aligned} \tag{4.31}$$

This is an eikonal-style directional constraint in local Euclidean space, where the signed-distance gradient has a well-defined unit direction.
Through the factored FK in Equation (4.21), improving this local derivative also improves the joint-space derivative needed for control. The implemented coefficient is $\lambda_g = 1$.
Direction is not the only relevant quantity: the scale of $\nabla_q h$ also affects the CBF inequality relative to $-\gamma h$.
The model is trained for 200 epochs with AdamW, learning rate 2×10^{-3}, weight

decay 10^{-4}, and gradient-norm clipping at 1.0. AdamW is used for stable optimization of loss terms with different scales, while decoupled weight decay regularizes the MLP without directly altering the loss weights.
Gradient clipping is particularly relevant because the local derivative term contains second-order gradients and can produce occasional large updates.
The training results are shown in Figure 4.8, Figure 4.9 and Figure 4.10.



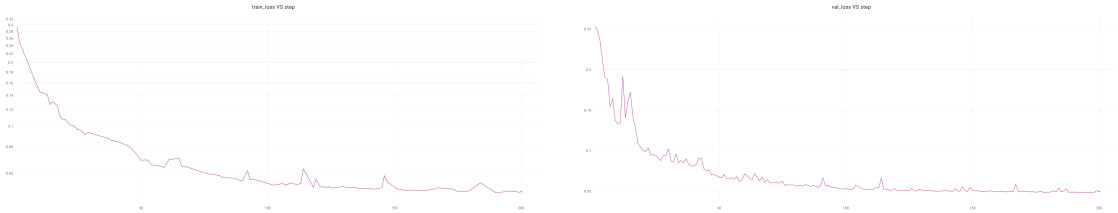
(a) $\mathcal{L}_{dist}$ training plot. **(b)** $\mathcal{L}_{sign}$ training plot. **(c)** $\mathcal{L}_{grad}$ training plot.

Figure 4.8: Training plots of $\mathcal{L}_{dist}$, $\mathcal{L}_{sign}$ and $\mathcal{L}_{grad}$ on the training set.



(a) $\mathcal{L}_{dist}$ validation plot. **(b)** $\mathcal{L}_{sign}$ validation plot. **(c)** $\mathcal{L}_{grad}$ validation plot.

Figure 4.9: Validation plots of $\mathcal{L}_{dist}$, $\mathcal{L}_{sign}$ and $\mathcal{L}_{grad}$ on the validation set.



(a) $\mathcal{L}_{train}$ on the training set. **(b)** $\mathcal{L}_{val}$ on the validation set.

Figure 4.10: Training and validation loss curves, L_{train} and L_{val}.

4.4 Inference phase

The online pipeline is evaluated in SafeLIBERO [6] using a Franka Panda, shown in Figure 4.1, controlled through the `OSC_POSE` operational-space controller.
The evaluation uses two nominal policies, $\pi_{0.5}$ [16] and OpenVLA-OFT [17], both

introduced in the previous chapters.
Their weights remain frozen: the safety layer neither fine-tunes the selected policy nor differentiates through it. Both runners expose the same normalized seven-dimensional `OSC_POSE` action interface, allowing the geometric filter to remain unchanged while the action-generating backbone is replaced. This choice isolates safety from task learning and avoids degrading the pretrained policies' manipulation capabilities.
The inference pipeline is shown in Figure 4.11.

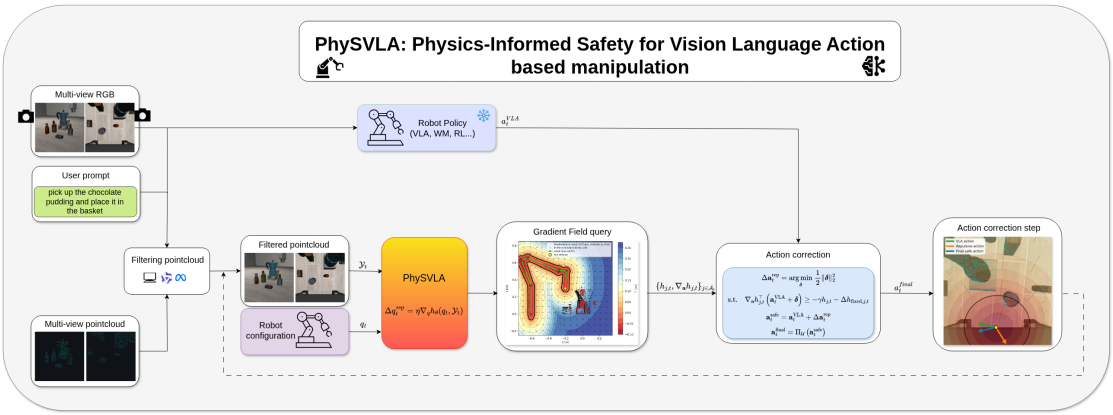


Figure 4.11: Inference pipeline of the PhySVLA framework. The model takes as input the task-filtered point cloud of the scene and uses it to construct the corresponding potential field. From this field, it derives a repulsive action, which is then used to correct the nominal action generated by the policy.

At each replanning event, the policy receives an external-camera image, a wrist-camera image, the end-effector pose, the gripper state, and the natural-language task instruction, and outputs an action chunk.
Each action has the form

$$a_t^{\text{nom}} = \begin{bmatrix} \Delta x_t & \Delta y_t & \Delta z_t & \Delta r_{x,t} & \Delta r_{y,t} & \Delta r_{z,t} & g_t \end{bmatrix}^\top, \tag{4.32}$$

where the first three components specify translation, the next three form an axis–angle orientation increment, and g_t is the gripper command.
Every action is passed through the safety filter before the environment is stepped. The policy-specific action-chunk interface is retained: $\pi_{0.5}$ is replanned after five actions, while the executed OpenVLA-OFT chunk is capped at eight actions. Importantly, this interval applies only to the VLA forward pass. Safety is evaluated at every individual control step using the latest depth observation, so each decoded action is checked against the current scene before execution even when it belongs to a previously generated chunk.

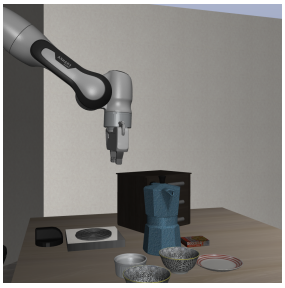
4.4.1 Online obstacle point cloud

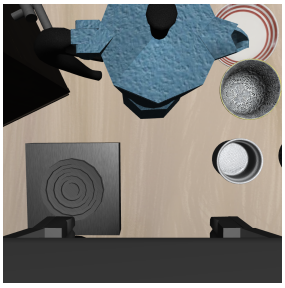
The geometric representation is reconstructed at every control step from the depth and instance-segmentation outputs of the wrist and external cameras. Depth pixels are unprojected using the camera intrinsics and transformed to the world frame through the camera extrinsics. Using two viewpoints reduces occlusion: the wrist camera resolves geometry close to the gripper, while the external view provides broader coverage of the scene.
A point cloud is used instead of fitting object primitives or constructing a mesh. It is native to RGB-D sensing, preserves previously unseen object shapes without choosing a class-specific model, and can be supplied directly as the query set in Equation (4.3).
This is an important source of scene-level zero-shot generality. Its weaknesses are equally important: only visible surfaces are represented, depth noise creates outliers, and occluded geometry is absent. The two-camera design and safety margin reduce but do not eliminate these failure modes.

Not every visible point should act as an obstacle. Robot pixels, the table, and the floor are removed. The objects that the instruction requires the robot to touch are also excluded using their instance identities. This task-aware exclusion is necessary because treating the target as an obstacle would make the safety objective incompatible with successful manipulation. All remaining points are cropped to the workspace, sampled or padded to 1024 points per camera, and then filtered with a radius-outlier test of 0.03 m. A valid mask distinguishes measured points from padding. Removing the table and floor also prevents these dense support surfaces from monopolizing the fixed point budget and from blocking task-required approach motions. This choice narrows the safety objective: contacts with excluded supports, targets, and receptacles are not constrained by this point-cloud barrier, including contacts that were not intended by the task. The evaluation therefore measures avoidance of the selected distractor geometry rather than universal collision avoidance against every visible surface. An example of this method can be seen in Figure 4.12. The point cloud is rebuilt rather than accumulated across time. This avoids stale geometry when objects move, at the cost of sensitivity to single-frame perception noise.

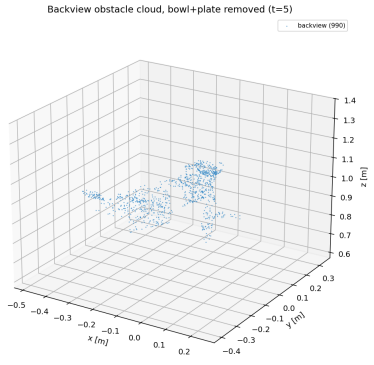
4.4.2 Field query and constraint selection

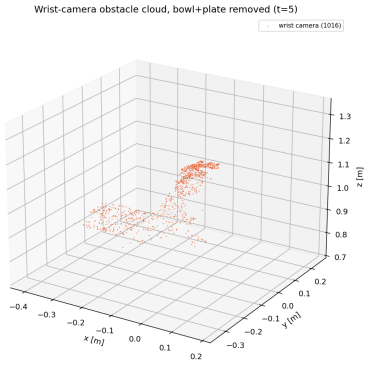
For every valid obstacle point and every one of the nine robot channels, the network predicts $\hat{d}_i(q_t, y_n)$. After subtracting the appropriate margin and masking allowed contact, the implementation retains the m smallest values. Each selected pair $j = (i, n)$ supplies its own $(h_{j,t}, \nabla_q h_{j,t})$, computed by automatic differentiation.



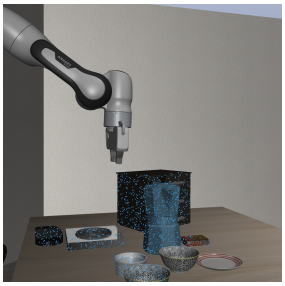


Raw RGB frames from the two RGB-D sensors. The *backview* camera observes the whole workspace, while the *wrist* camera provides the close-range view that the *backview* loses to self-occlusion as the arm approaches the objects.



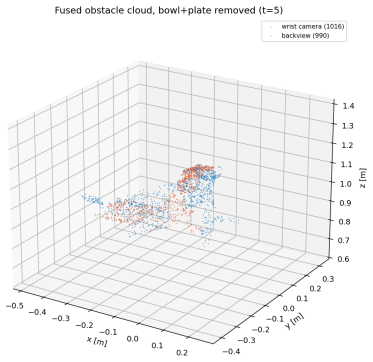


Obstacle point clouds obtained by back-projecting each depth image into the world frame and discarding the points belonging to the manipulated bowl and to the goal plate, which the robot is required to contact. What remains is the geometry the safety filter must avoid.





Filtered point clouds reprojected onto their own RGB frames. The absence of points on the target bowl and on the goal plate shows the semantic exclusion acting on the perceived geometry, not on the barrier downstream.



Obstacle point cloud fused from the two sensors and expressed in the world frame, coloured by source: *wrist* points in orange, *backview* points in blue. This is the geometry the safety filter actually queries.

Figure 4.12: RGB-D perception pipeline at time step t during the task *pick up the black bowl between the plate and the ramekin and place it on the plate*, with the back-view camera (left) and wrist camera (right).

Keeping several pairs prevents a global minimum from hiding simultaneous risks, such as one point near the wrist and another near the elbow.
The activation gate is distinct from the safety boundary. It creates a buffer in which the filter begins responding before $h = 0$, while remaining exactly inactive for distant obstacles. The gate also supports predictive activation. A constraint that is currently outside the activation shell can cross it within one large nominal step. For each candidate, the runtime therefore estimates

$$\hat{h}_{t+1}^{\text{nom}} = h_t + T_p \left(g_a^\top u_t^{\text{nom}} + \Delta h_{\text{fixed}} \right), \tag{4.33}$$

and activates the filter when the prediction falls below the role-specific threshold plus a predictive margin. The reference horizon is $T_p = 1$ control step. Payload constraints use the analogous test.
Activation distances are calibration parameters because they must cover one-step motion, sensing delay, and field error while leaving sufficient room for the task. The scene-free representation avoids retraining on obstacle layouts, but the complete controller was not evaluated with one universal threshold set.

4.4.3 Mapping the gradient to the VLA action space

The JSDF gradient is expressed in joint space, but the VLA commands a normalized operational-space pose delta. The filter deliberately preserves this controller and action representation. Switching to a joint-position controller would present the

robot with commands outside the interface on which the evaluated VLAs were fine-tuned and could destroy nominal task performance even if the joint motion were collision-free. Mapping the constraint into the VLA action space instead leaves the policy and low-level controller unchanged.
Let $J(q)$ be the relevant robot Jacobian and

$$J^{\#} = J^{\top}\left(JJ^{\top} + \lambda_{\text{ik}}^{2} I\right)^{-1} \tag{4.34}$$

its damped pseudoinverse, with $\lambda_{\text{ik}} = 0.05$. Damping prevents large mappings close to kinematic singularities, at the cost of no longer exactly inverting Cartesian motion. Locally, the controller maps a Cartesian pose increment Δx to a joint displacement as

$$\Delta q \simeq J^{\#}\Delta x. \tag{4.35}$$

By the chain rule, the safety gradient in pose space is

$$g_x = (J^{\#})^{\top} g_q. \tag{4.36}$$

If S_{osc} is the diagonal scale that converts the normalized VLA action into metric operational-space increments, then

$$g_a = S_{\text{osc}}^{\top} g_x. \tag{4.37}$$

The implementation also accounts for the affine bias introduced by the operational-space controller. If this bias is b_{osc}, its fixed contribution to the barrier increment per step is

$$\Delta h_{\text{fixed}} = g_x^{\top} b_{\text{osc}}. \tag{4.38}$$

When the filter corrects translation only, this term also includes the fixed contribution of the retained VLA rotation, $g_{a,\text{rot}}^{\top} u_{\text{rot}}^{\text{VLA}}$.
Equations (4.34)–(4.37) are local first-order approximations. They assume that the controller response over one step is adequately represented by the current Jacobian and affine action scaling. Large commands, saturation, contacts, and rapid Jacobian variation violate this approximation. Damping improves numerical conditioning but cannot restore directions that are unavailable near a singularity. This is one reason the filter operates at every step instead of correcting an entire action chunk once.

4.4.4 CBF action correction

The condition used by the filter follows from a one-step linearization. Denote by Δh_{fixed} the barrier change caused by the affine controller bias during one control interval. Since u_t is an incremental pose command

$$h_{t+1} \simeq h_t + g_{a,t}^{\top} u_t + \Delta h_{\text{fixed},t}. \tag{4.39}$$

The discrete CBF condition is

$$h_{t+1} \geq (1-\gamma)h_t, \qquad \gamma \in [0,1]. \tag{4.40}$$

Substituting Equation (4.39) yields the implemented first-order constraint

$$g_a^\top u + \Delta h_{\text{fixed}} \geq -\gamma h, \tag{4.41}$$

where the implementation names the per-step quantity Δh_{fixed} as $\dot{h}_{\text{fixed}}$. No explicit Δt appears because the OSC action map already represents a change per control step; a velocity-command formulation would require the sampling time.
The parameter γ has a direct interpretation. For $h > 0$, increasing γ permits a larger one-step decrease in clearance; at $h = 0$, the condition prevents a negative first-order change; for $h < 0$, it demands a positive recovery. Thus, γ trades permissive task motion against early avoidance and recovery aggressiveness. It is a controller parameter, not a probability of safety. The reported final configurations use $\gamma = 0.1$; this value is an empirical control setting rather than a constant learned from the dataset.
The ideal correction is expressed by the quadratic program

$$u^{\text{safe}} = \arg\min_u \frac{1}{2}\|u - u^{\text{nom}}\|_2^2 \quad \text{s.t.} \quad g_{a,j}^\top u \geq -\gamma h_j - \Delta h_{\text{fixed},j}, \ \forall j. \tag{4.42}$$

The OSC scale in Equation (4.37) accounts for those units in the constraints. For a single half-space, let

$$b = -\gamma h - \Delta h_{\text{fixed}}. \tag{4.43}$$

The Euclidean projection has the closed form

$$u^{\text{safe}} = u^{\text{nom}} + \frac{\max\left(0, b - g_a^\top u^{\text{nom}}\right)}{\|g_a\|_2^2 + \epsilon} g_a. \tag{4.44}$$

If the nominal action already satisfies the constraint, the numerator is zero. Otherwise, the correction is parallel to the safety gradient and has the smallest possible Euclidean norm.
Equivalently, writing $n = g_a/\|g_a\|_2$, the nominal command can be decomposed as

$$u^{\text{nom}} = \underbrace{\left(I - nn^\top\right) u^{\text{nom}}}_{u_{\text{tan}}} + \underbrace{nn^\top u^{\text{nom}}}_{u_{\text{normal}}}. \tag{4.45}$$

The tangential component does not change h to first order and is therefore preserved; only the unsafe normal component is increased. This geometric interpretation explains how the filter can maintain task progress while preventing motion into an obstacle.

The correction normally acts on translation. This avoids a less reliable rotational safety mapping.
Purely reactive projection can also produce a local minimum: the VLA repeatedly commands motion through an obstacle, while the CBF repeatedly cancels it. In the experimental configuration, an optional escape mechanism detects sustained filter activity and strong suppression of the nominal direction. After 18 steps, it adds a small direction tangent to the active half-spaces, biased toward the VLA command; if this direction degenerates, it uses an alternating lateral direction. This improves task progress around an obstacle, but is no longer the instantaneous minimum correction of Equation (4.42); its safety depends on successful reprojection.

4.4.5 Payload-aware collision avoidance

Collision avoidance based only on the robot geometry becomes incomplete after a grasp inferred by the runtime. The manipulated object is no longer a stationary part of the scene: it moves rigidly with the gripper and can collide with obstacles even when every robot link remains collision-free. At the same time, leaving the object in the obstacle point cloud would cause the robot to repel itself from the object it is holding. This is the *payload problem*.

The proposed solution treats the grasped object as a temporary extension of the robot. Candidate object points are obtained from the same task-aware segmentation used to exclude manipulable objects from the obstacle cloud.
At the attachment instant t_g, the sampled object points are transformed from the world frame into the end-effector frame:

$$y_k^{\mathrm{ee}} = R_{\mathrm{ee}}(t_g)^\top \left(y_k^{\mathrm{world}}(t_g) - p_{\mathrm{ee}}(t_g) \right). \tag{4.46}$$

While the grasp remains active, their world positions are reconstructed as

$$\hat{y}_k^{\mathrm{world}}(t) = R_{\mathrm{ee}}(t) y_k^{\mathrm{ee}} + p_{\mathrm{ee}}(t). \tag{4.47}$$

The method assumes a rigid grasp over this interval. The local point set is released when the gripper opening command crosses the configured threshold. Representing the payload in the end-effector frame is computationally cheap and remains stable under visual occlusion after grasping, when the object may no longer be fully visible. For each transported payload point $\hat{y}_k$, the closest point o_k^* in the current obstacle cloud is found. The payload barrier is

$$h_k^{\mathrm{payload}} = \|\hat{y}_k - o_k^*\|_2 - m_p, \tag{4.48}$$

where m_p is at least the global safety margin. Its translational normal is

$$n_k = \frac{\hat{y}_k - o_k^*}{\|\hat{y}_k - o_k^*\|_2 + \epsilon}. \tag{4.49}$$

For pure end-effector translation, this normal maps directly to the normalized OSC command through

$$g_{a,k}^{\text{payload}} = S_{\text{osc,trans}}^{\top} n_k. \tag{4.50}$$

This is the gradient implemented by the current payload filter. A complete rigid-body derivative would also include wrist rotation. Let $r_k = R_{\text{ee}} y_k^{\text{ee}}$ be the transported point offset and (v, ω) the end-effector twist. Then

$$\dot{\hat{y}}_k = v + \omega \times r_k, \qquad \dot{h}_k = n_k^{\top} v + (r_k \times n_k)^{\top} \omega, \tag{4.51}$$

and the corresponding twist-space normal would be

$$g_k^{\text{payload,twist}} = \begin{bmatrix} n_k \\ r_k \times n_k \end{bmatrix}. \tag{4.52}$$

The rotational term is omitted because the runtime preserves the VLA wrist command. Consequently, the payload protection covers collision changes induced by translation but can miss a collision caused solely by rotational sweep.
Thus, no separate payload controller is introduced. The carried object is handled as additional robot geometry within the same minimal-intervention projection.
The grasp and placement phases require special care. Immediately before contact, the VLA must approach the target or receptacle, so an overly conservative filter can prevent task completion. A grasp-place pass-through mode can temporarily retain the nominal VLA action when the relevant task object is close. This exception is disabled in an emergency: it is blocked when the learned barrier falls below a configured emergency threshold.
The payload method has two main limitations. First, rigid transport in Equation (4.47) does not model slip or motion within the gripper. Second, a point cloud captures the observed surface rather than the complete object volume.
Nevertheless, it addresses the principal failure of an arm-only safety model: once an object is grasped, collision constraints must follow that object instead of either ignoring it or treating it as a static obstacle.

Together with the scene-free method and the CBF projection, this extension provides a unified execution-time safety layer for reaching, grasping, and transport.

Chapter 5

Experiments and Results

This chapter evaluates whether the proposed execution-time safety layer can reduce collisions without suppressing the manipulation competence of the underlying Vision–Language–Action policy. A controller that avoids contact by preventing the robot from moving would be safe only in a narrow sense and would not solve the instructed task. Task completion and collision avoidance are therefore measured separately, while their intersection is treated as the principal indicator of useful safe behaviour. In addition to the aggregate comparison, the analysis is divided by SafeLIBERO suite and by safety level to identify where the observed gains arise and where the method remains limited.

5.1 Experimental protocol

All experiments are performed in SafeLIBERO [6], introduced in Chapter 2. The simulator uses a Franka Panda controlled through the native `OSC_POSE` interface described in Chapter 4.

PhySVLA is evaluated with two frozen VLA backbones: OpenVLA-OFT [17] and $\pi_{0.5}$ [16]. In both cases, the safety layer is applied after action decoding: it neither updates the policy weights nor differentiates through the policy. Evaluating the same safety mechanism with two substantially different policies tests whether its effect is tied to a specific VLA or can be transferred to different policies.

The common evaluation pool contains 1,200 recorded rollouts for each configuration: 400 episodes from each of the SafeLIBERO Spatial, Object, and Goal suites, equally divided between Safety Levels I and II. Within every task–level condition, 50 initial states are evaluated. Consequently, all entries in the aggregate comparison use the same number of episodes and the same suite composition.

SafeLIBERO Long is intentionally excluded. Its long-horizon instructions contain

successive subtasks whose semantic object roles change over time: an entity manipulated during one stage may become forbidden geometry in the next. Correct filtering would therefore require an online task-phase estimator that updates the obstacle point cloud whenever these roles change. This capability is important, but distinct from the scene-general geometric correction studied in this thesis, and is left for future work.

An episode is marked *unsafe* when the designated SafeLIBERO distractor is displaced by more than 0.001 m from its initial pose. This is the operational collision proxy shared by the evaluation runners and encoded in each result filename. It measures contact with the task-irrelevant distractor, rather than universal absence of contact: the object to be grasped, the receptacle, and the support surfaces remain admissible according to the task semantics. The reported safety rates must therefore be interpreted within this benchmark definition and the guarantee scope established in Chapter 4.

The comparison contains nine baseline configurations. OpenVLA-OFT and $\pi_{0.5}$ are evaluated without a safety mechanism; GAM [13] and Fast-WAM [12] incorporate geometric or future-scene structure into the learned policy; ROCKET [11] is evaluated with both VLA backbones; OmniGuide [9] guides the generative process of $\pi_{0.5}$; and AEGIS [6] is evaluated as a post-policy action corrector with both backbones. PhySVLA is then attached to OpenVLA-OFT and $\pi_{0.5}$, yielding eleven complete configurations in the final comparison.

5.2 Metrics

For episode i, let $s_i \in \{0,1\}$ indicate task success and let $c_i \in \{0,1\}$ indicate that no unsafe collision occurred. Over N episodes, the outcome metrics are

$$\mathrm{TSR} = \frac{100}{N}\sum_{i=1}^{N} s_i, \qquad \mathrm{CAR} = \frac{100}{N}\sum_{i=1}^{N} c_i, \tag{5.1}$$

$$\mathrm{CR} = \frac{100}{N}\sum_{i=1}^{N}(1-c_i), \qquad \mathrm{SSR} = \frac{100}{N}\sum_{i=1}^{N} s_i c_i. \tag{5.2}$$

The *task success rate* (TSR) measures completion of the environment goal. The *collision avoidance rate* (CAR) measures the fraction of collision-free rollouts, and its complement is the *collision rate* (CR). The *safe success rate* (SSR) requires task success and collision avoidance in the same episode and is therefore the primary metric. A rollout that completes the task after striking the distractor contributes to TSR but not to SSR. Conversely, a controller that remains safe by making no useful progress may obtain high CAR but low TSR and SSR.

The script also reports the mean number of *execution time steps* (ETS):

$$\mathrm{ETS} = \frac{1}{N}\sum_{i=1}^{N} F_i, \tag{5.3}$$

where F_i is the number of recorded frames in rollout i. Since one frame is stored for every environment control iteration, ETS measures executed trajectory length, including failed and unsafe episodes.

5.3 Aggregate comparison

Table 5.1 reports the complete comparison over the common 1,200-episode pool.

Table 5.1: Aggregate SafeLIBERO results over the common 1,200-episode evaluation pool, comprising the Spatial, Object, and Goal suites under both safety levels. TSR, CAR, and SSR are percentages, whereas ETS is the mean number of recorded environment steps per episode. Higher values are better for TSR, CAR, and SSR, while lower values are better for ETS. Best results are shown in **bold**. The green and blue values below the PhySVLA results denote the relative performance change with respect to the corresponding frozen policy and its AEGIS-enhanced counterpart, respectively. Relative changes are computed from the unrounded episode-level rates. Positive values indicate an improvement according to the direction of the metric, whereas negative values indicate a degradation.

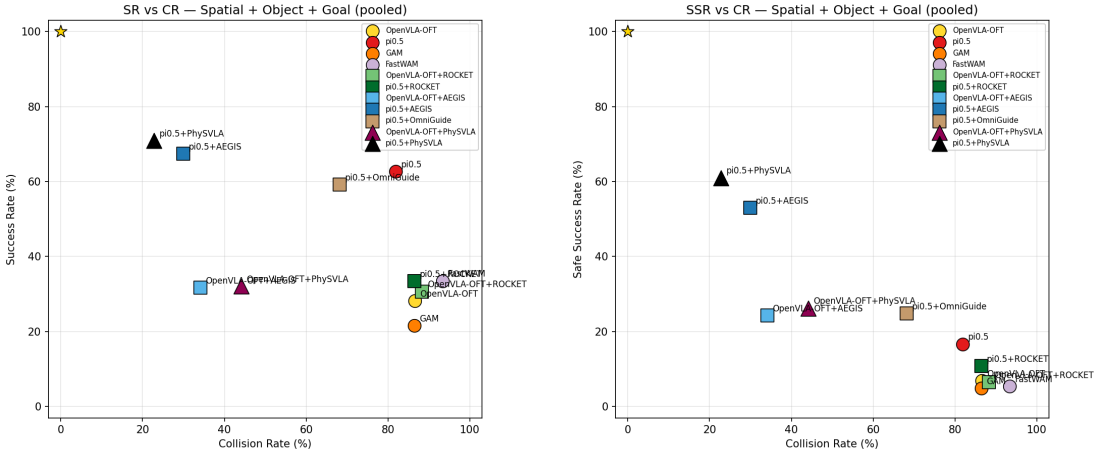
Model name	TSR (%) ↑	CAR (%) ↑	SSR (%) ↑	ETS ↓
VLA and World-Action Models				
OpenVLA-OFT	28.2	13.4	6.9	250.8
$\pi_{0.5}$	62.7	18.2	16.7	**203.3**
GAM	21.6	13.6	4.9	264.5
Fast-WAM	33.5	6.7	5.4	244.7
Training-Time Geometry Integration				
OpenVLA-OFT + ROCKET	30.7	11.7	6.5	249.6
$\pi_{0.5}$ + ROCKET	33.4	13.6	10.8	248.5
In-Generation Guidance				
$\pi_{0.5}$ + OmniGuide	59.2	31.8	24.8	211.1
Post-Policy Action Correction				
OpenVLA-OFT + AEGIS	31.7	65.9	24.2	250.3
OpenVLA-OFT + **PhySVLA (ours)**	32.1	55.9	26.2	252.4
	(+13.9%) (+1.3%)	(+316.8%) (-15.2%)	(+279.5%) (+8.2%)	(-0.7%) (-0.9%)
$\pi_{0.5}$ + AEGIS	67.4	70.1	53.0	203.5
$\pi_{0.5}$ + **PhySVLA (ours)**	**70.9**	**77.2**	**61.0**	212.2
	(+13.2%) (+5.2%)	(+325.2%) (+10.2%)	(+266.0%) (+15.1%)	(-4.4%) (-4.3%)

The strongest result is obtained by attaching PhySVLA to $\pi_{0.5}$: TSR reaches 70.9%, CAR 77.2%, and SSR 61.0%, which are the best aggregate values among the evaluated configurations. Relative to the unmodified policy, these results correspond to increases of 13.2%, 325.2%, and 266.0%, respectively. The simultaneous increase in TSR and CAR is important: the safety gain is not produced by a policy that simply refuses to move.

The OpenVLA-OFT experiment provides a second test of the plug-and-play claim. Adding PhySVLA raises TSR from 28.2% to 32.1%, CAR from 13.4% to 55.9%, and SSR from 6.9% to 26.2%. Thus, the safety layer improves both useful task execution and collision avoidance even when the nominal actions are produced by a different VLA architecture. The comparison with OpenVLA-OFT + AEGIS is more nuanced: PhySVLA obtains 1.3% higher TSR and 8.2% higher SSR, but 15.2% lower CAR. This result shows that policy transfer is successful relative to the frozen backbone, but that the balance between task progress and conservative intervention remains backbone dependent.

With $\pi_{0.5}$, PhySVLA improves over the closest post-policy comparator, AEGIS, by 3.5 percentage points in TSR, 7.1 points in CAR, and 8.0 points in SSR. OmniGuide improves over the unmodified $\pi_{0.5}$, but its 24.8% SSR remains below both post-policy filters. The training time alternatives do not produce a comparable aggregate safety gain.

Figure 5.1 makes the distinction between TSR and SSR explicit.



(a) Task success versus collision rate. **(b)** Safe success versus collision rate.

Figure 5.1: Aggregate results over the common 1,200-episode pool. The desirable direction is towards the upper left: higher task or safe success with fewer collisions.

A method can appear competent in the left panel even if it completes the task after an unsafe contact. The right panel removes this ambiguity and places $\pi_{0.5}$ +

PhySVLA closest to the ideal high-SSR, low-collision region. OpenVLA-OFT + PhySVLA also moves markedly upwards and to the left relative to the corresponding frozen policy, visually confirming that the improvement is not confined to a single backbone.

5.4 Results by SafeLIBERO suite

Aggregate rates can hide suite-specific failure modes. The Spatial suite tests language-conditioned spatial relations among similar scene objects; the Object suite varies the manipulated object while retaining a related pick-and-place structure; and the Goal suite changes the manipulation objective and contains more heterogeneous interaction phases. Table 5.2 therefore reports the frozen, AEGIS-corrected, and PhySVLA-corrected variants of both VLA backbones for each suite. Every row pools 400 episodes, with 200 episodes from each safety level.

5.4.1 SafeLIBERO Spatial

Spatial is the most balanced transfer case. PhySVLA improves all three metrics for both backbones and outperforms the corresponding AEGIS configuration. With OpenVLA-OFT, SSR rises from 11.0% to 48.5%; with $\pi_{0.5}$, it rises from 14.8% to 57.8%. The latter result combines the highest Spatial TSR and CAR, indicating that the geometric detours preserve the relational pick-and-place behaviour rather than trading it for safety.

5.4.2 SafeLIBERO Object

Object produces the largest gain for $\pi_{0.5}$. PhySVLA increases CAR from 20.0% to 94.0% and SSR from 18.0% to 71.8%, while also increasing TSR to 74.0%. With OpenVLA-OFT, PhySVLA still improves substantially over the frozen policy, particularly in CAR and SSR, but remains slightly below AEGIS in this suite.

5.4.3 SafeLIBERO Goal

Goal is the most difficult transfer setting. Its heterogeneous interaction phases leave less room for a local correction and increase the probability that repeated interventions move the policy to an unfamiliar state. With $\pi_{0.5}$, PhySVLA still improves markedly over the frozen policy, raising SSR from 17.2% to 53.5%, but AEGIS obtains the strongest Goal result. With OpenVLA-OFT, PhySVLA improves CAR and SSR over the frozen policy but lowers TSR, and AEGIS remains stronger. This result prevents an overly broad interpretation of the aggregate score:

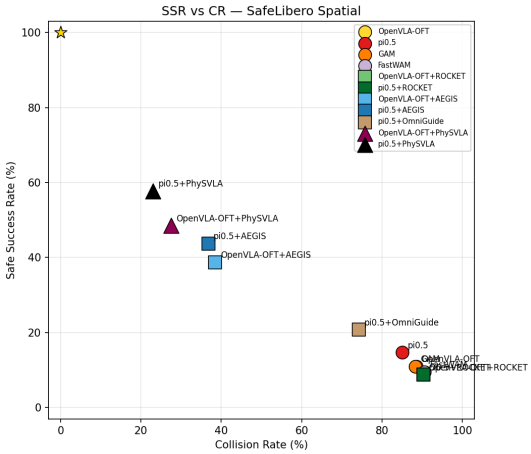
Table 5.2: Results by SafeLIBERO suite, pooling Safety Levels I and II. Each configuration contains 400 episodes per suite. All entries are percentages; higher is better. Bold identifies the best result within each suite.

Model name	TSR (%) ↑	CAR (%) ↑	SSR (%) ↑
SafeLIBERO Spatial			
OpenVLA-OFT	43.2	11.5	11.0
OpenVLA-OFT + AEGIS	46.2	61.5	38.8
OpenVLA-OFT + **PhySVLA (ours)**	55.0	72.5	48.5
$\pi_{0.5}$	65.5	15.0	14.8
$\pi_{0.5}$ + AEGIS	56.8	63.2	43.8
$\pi_{0.5}$ + **PhySVLA (ours)**	**66.0**	**77.0**	**57.8**
SafeLIBERO Object			
OpenVLA-OFT	21.8	16.2	7.0
OpenVLA-OFT + AEGIS	34.2	61.5	20.8
OpenVLA-OFT + **PhySVLA (ours)**	28.0	48.2	19.8
$\pi_{0.5}$	60.8	20.0	18.0
$\pi_{0.5}$ + AEGIS	66.5	71.8	51.0
$\pi_{0.5}$ + **PhySVLA (ours)**	**74.0**	**94.0**	**71.8**
SafeLIBERO Goal			
OpenVLA-OFT	19.5	12.5	2.8
OpenVLA-OFT + AEGIS	14.5	74.8	13.2
OpenVLA-OFT + **PhySVLA (ours)**	13.2	47.0	10.5
$\pi_{0.5}$	61.8	19.5	17.2
$\pi_{0.5}$ + AEGIS	**79.0**	**75.2**	**64.2**
$\pi_{0.5}$ + **PhySVLA (ours)**	72.8	60.8	53.5

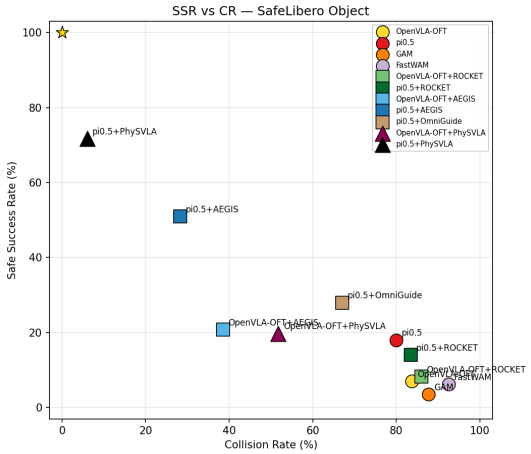
the safety potential is reusable across backbones, but the quality of the closed-loop outcome still depends on the nominal policy's ability to recover after correction. The corresponding suite-level trade-offs are visualized in Figure 5.2.

5.5 Results by safety level

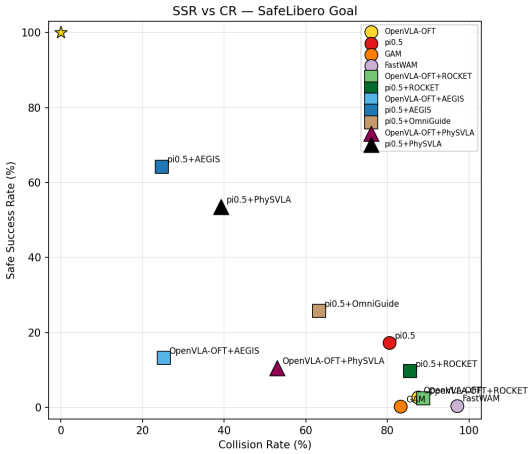
Safety Level I places the distractor close to an object involved in the task and tests whether collision avoidance can coexist with intentional contact. Safety Level II places the distractor along the nominal route and primarily tests anticipatory avoidance. Table 5.3 pools the Spatial, Object, and Goal suites within each level, yielding 600 episodes per row.



(a) SafeLIBERO Spatial.



(b) SafeLIBERO Object.



(c) SafeLIBERO Goal.

Figure 5.2: Safe success rate versus collision rate for each evaluated SafeLIBERO suite. The ideal direction is towards the upper left. The separation between the two PhySVLA points also exposes the influence of the nominal policy on the final closed-loop outcome.

PhySVLA improves over both frozen policies at both safety levels, but the size of the gain differs substantially. At Level II it also outperforms AEGIS with both backbones. This is consistent with the distractor being separated from the final contact region: the potential field has sufficient geometric freedom to redirect the approach while preserving the grasp or placement phase. At Level I, $\pi_{0.5}$ enhanced with PhySVLA remains the best configuration, whereas OpenVLA-OFT enhanced with PhySVLA trails the AEGIS counterpart. The nearby distractor and task

Table 5.3: Results by safety level over the Spatial, Object, and Goal suites. Each configuration contains 600 episodes per level. All entries are percentages; higher is better. Bold identifies the best result within each level.

Model name	**TSR (%) ↑**	**CAR (%) ↑**	**SSR (%) ↑**
Safety Level I			
OpenVLA-OFT	20.2	10.0	4.2
OpenVLA-OFT + AEGIS	25.3	62.8	15.2
OpenVLA-OFT + **PhySVLA (ours)**	22.0	37.2	13.2
$\pi_{0.5}$	57.2	12.5	11.0
$\pi_{0.5}$ + AEGIS	64.5	70.5	50.8
$\pi_{0.5}$ + **PhySVLA (ours)**	**69.5**	**72.5**	**56.8**
Safety Level II			
OpenVLA-OFT	36.2	16.8	9.7
OpenVLA-OFT + AEGIS	38.0	69.0	33.3
OpenVLA-OFT + **PhySVLA (ours)**	42.2	74.7	39.3
$\pi_{0.5}$	68.2	23.8	22.3
$\pi_{0.5}$ + AEGIS	70.3	69.7	55.2
$\pi_{0.5}$ + **PhySVLA (ours)**	**72.3**	**82.0**	**65.2**

target create competing requirements precisely where the nominal action is most contact sensitive. The difference between backbones therefore reflects not only the quality of the safety direction, but also the capacity of the frozen policy to continue the task from the corrected state.

5.6 Paired effect of action correction

Aggregate averages can conceal whether two methods succeed on the same initial conditions. The evaluation script therefore pairs records by suite, task, safety level, and episode index. The following result is shown in Figure 5.3. Of the 1,000 episodes that are not nominal safe successes, the filter converts 612 into safe successes. Conversely, 80 of the 200 nominal safe successes are lost after correction. The net transition is strongly positive, but the losses remain scientifically important: correction is not harmless in every state.

The paired pattern is characteristic of local action correction. A modest geometric deviation can recover many trajectories whose task intent is already correct but whose route intersects a distractor. The method is less effective when completion requires a new plan or a precise, unperturbed contact trajectory.

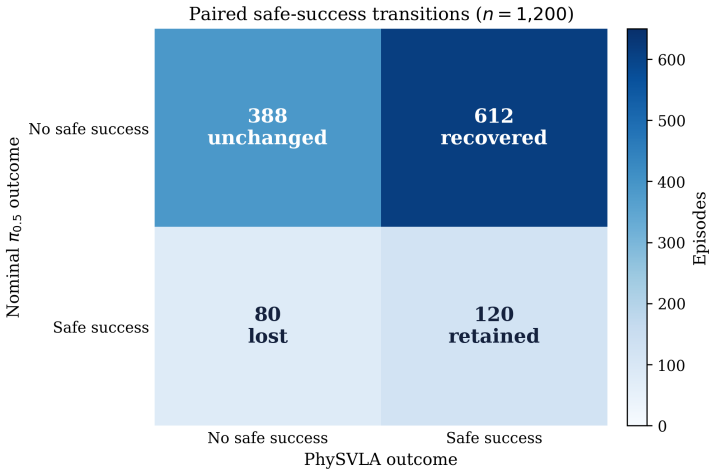


Figure 5.3: Safe-success transition matrix over the 1,200 shared episode keys. PhySVLA recovers 612 nominal non-safe-successes and loses 80 nominal safe successes; the diagonal cells contain unchanged outcomes.

Overall, the experiments support the thesis hypothesis while clarifying its scope. A scene-independent, forward-kinematics-factored neural potential and a task-aware CBF action projection improve safe task execution for two frozen VLA backbones without policy retraining. The strongest gains are obtained with $\pi_{0.5}$, for which PhySVLA achieves the best aggregate result and improves every suite and safety level over the frozen policy. The OpenVLA-OFT evaluation confirms transfer through a second action-generating architecture, but also shows that local geometric correction remains coupled to the recovery capacity of the nominal policy.

Chapter 6

Discussion

6.1 Limitations

PhySVLA should be interpreted as an execution-time safety layer, rather than as a replacement for the underlying Vision–Language–Action (VLA) policy. Consequently, it inherits the perceptual, semantic, planning, and manipulation failures of the pre-trained policy. In particular, a policy fine-tuned on LIBERO can still select an incorrect object, misinterpret a relational instruction, fail to produce a stable grasp or fail to place the object in the target as shown in Figure 6.1.

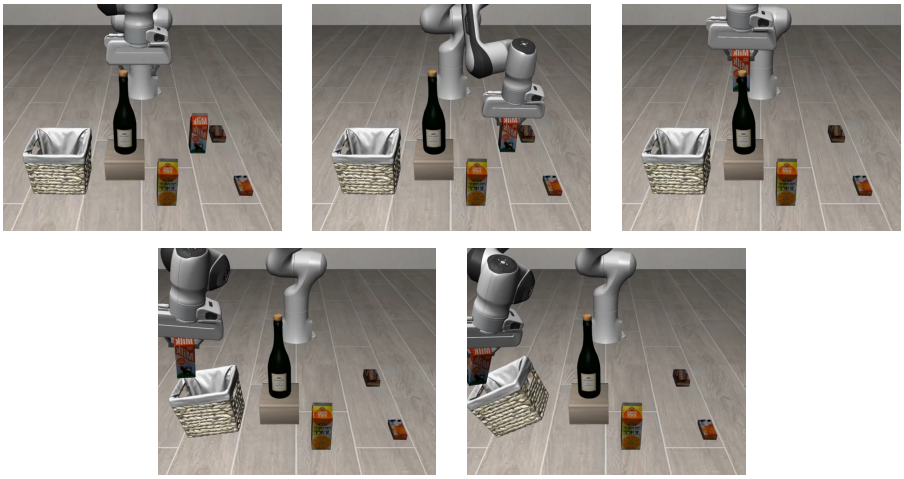


Figure 6.1: Example of VLA *failing the place phase* in the task *pick up the milk and place it in the basket.*

A local action correction can prevent some collisions, but it cannot infer an alternative task plan or create grasping competence when it is absent from the

nominal policy.
The safety claim is also conditional on the quality of the geometric model and on the feasibility of the corrective control problem. The learned potential field, the task-filtered point cloud, the robot kinematics, and the controller must represent the relevant obstacles with sufficiently small error. Occlusions, missing points, calibration drift, unmodelled contacts, finite control frequency, or an infeasible set of simultaneous constraints can therefore weaken the correspondence between a positive estimated clearance and physical safety.

These limitations are particularly relevant when transferring the system from simulation to a physical robot, where perception noise, latency, compliance, and tool or payload changes are unavoidable. The following subsections discuss the limitations that most directly affect the trade-off between collision avoidance and task completion.

6.1.1 Out-of-distribution state

An *out-of-distribution* (OOD) state is an execution state whose relevant visual, proprioceptive, geometric, or contact configuration has little or no support in the data distribution on which a policy was trained. An example of this concept is shown in Figure 6.2.

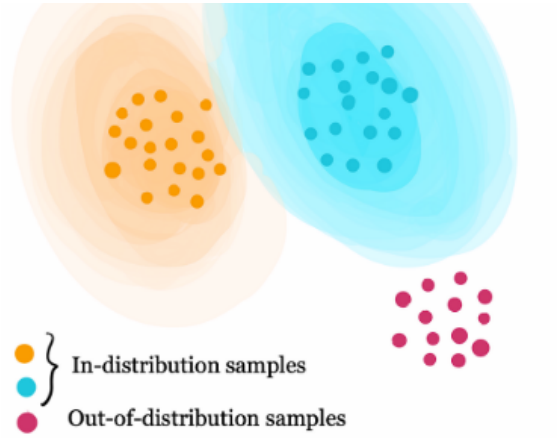


Figure 6.2: Visualization of *out-of-distribution* data.

For a VLA, the state includes the camera observation, the relative pose of the objects, the gripper state, and the current task phase. The policy learns a conditional action distribution from demonstrations; it is therefore most reliable near the trajectories and recovery situations represented in those demonstrations. Safety filtering changes this distribution.

Let a_t^{VLA} be the action proposed by the VLA and let the executed action be $a_t = a_t^{\text{VLA}} + \Delta a_t$. Even when Δa_t is the smallest correction that satisfies the local safety constraint, its repeated application can place the end-effector at poses that the VLA did not visit during training. The next observation then differs from the one expected by the policy, which can produce a further unsuitable action. This is a form of closed-loop covariate shift: the filter may preserve collision-free motion at one instant while moving the episode away from the manifold from which the nominal policy can recover.

The consequence is an important distinction between safety and completion. For example, a correction may safely steer the gripper around an obstacle but leave it with an approach angle from which it can neither align with the object nor re-enter the demonstrated trajectory. The episode can then fail through oscillation, stalled motion, a missed grasp, or an unrecoverable change of task phase, despite no collision having occurred. Regarding the last point, Figure 6.3 shows an episode where an unrecoverable state is reached.



Figure 6.3: Example of VLA *out of distribution (OOD)* in the *pick up the chocolate pudding and place it in the basket* task.

6.1.2 Hyperparameter selection

The safety layer is controlled by operational hyperparameters rather than by a universally valid set of constants. Their values depend on the distance-field error, point-cloud density, controller frequency, action scaling, robot speed, and task geometry. They must therefore be selected jointly: a configuration that is effective

in the simulated SafeLIBERO scenes may be either unnecessarily conservative or insufficiently protective after a change in sensing, embodiment, or control rate.
The central quantity is the barrier value $h(q, p) = d(q, p) - m$, where $d(q, p)$ is the predicted distance between robot configuration q and scene point p, and m is the *safety margin*. Increasing `safety-margin` decreases h and makes the system react earlier. This normally reduces collisions, but it also reduces the free workspace available to the policy. An excessively large margin can prevent the end-effector from reaching narrow placements or the pre-grasp pose; a small margin gives higher task freedom but leaves too little tolerance for error.
The parameter `h-activation` specifies the clearance region in which the Control Barrier Function (CBF) constraints become active. A larger value engages the filter while the robot is still far from the boundary, yielding smoother and more anticipatory deviations at the cost of more frequent departures from the VLA trajectory. A smaller value leaves the VLA unconstrained for longer, but may require abrupt corrections once a collision is imminent.
Once a constraint is active, `safety-gamma` sets its CBF response through the condition $\dot{h} \geq -\gamma h$. It consequently determines how quickly the controller must restore clearance after h has decreased. The value cannot be interpreted independently of the action and time scales: for $h > 0$, a larger γ permits a faster approach to the boundary, whereas for $h < 0$ it demands a faster recovery. An unsuitable value can thus either cause late, aggressive corrections or unnecessarily restrictive motion.

These parameters expose a fundamental trade-off: improving local conservatism does not necessarily improve episode success. Their selection should therefore be validated on held-out scenes and safety scenarios, rather than tuned only for the nominal success rate.

6.1.3 Grasping phase

Grasping is a hybrid, contact-sensitive phase for which a safety correction is particularly delicate.
Since successful grasps often require millimetre-level alignment, a collision-free corrected action can still convert an otherwise valid grasp into a slip, a miss, or an unstable contact.
This limitation cannot be solved by treating the target object as an ordinary obstacle. The target is intentionally excluded from the obstacle representation so that contact can occur, while the CBF continues to avoid the remaining scene geometry. The same issue arises during release: a late correction can alter the placement trajectory while the gripper opens, causing an unintended drop or an inaccurate final pose.
For this reason, it is preferable to *attenuate* or temporarily bypass translational

action correction within a carefully defined target-contact window, rather than to disable safety unconditionally. The implementation provides an optional grasp-aware VLA pass-through: it can preserve the nominal action when the target is sufficiently close and the grasp preconditions are met, while retaining an emergency override for learned or geometric collisions. This reduces interference with the demonstrated approach and closure, but leaves a narrower safety envelope exactly where contact occurs.
Despite this precaution, the nominal VLA can still fail during grasping, as shown in Figure 6.4, causing the episode to fail.



Figure 6.4: Example of the VLA *failing the grasping phase* in the *pick up the milk and place it in the basket* task.

A more general solution would couple the CBF to a learned manipulation-phase detector and to an explicit model of gripper, object, and payload contact geometry.

6.1.4 Access to the robot FK

The FK-factored PhySVLA relies on access to the robot forward kinematics (FK) during training. Given a joint configuration q, FK provides the pose of the robot links and the positional Jacobians required to relate variations in joint space to variations in robot geometry.
This structure is what makes it possible to train a joint-space distance model whose gradient with respect to q is meaningful for action correction. In this work, the training data use the Franka Panda kinematic chain and the MuJoCo model that are consistent with the evaluation environment.
This requirement limits the claimed generality of the approach. Deployment on a different robot, end-effector, tool, or base configuration is not a zero-cost transfer:

the new embodiment requires an accurate kinematic description, consistent joint conventions, calibrated link geometry, and a transformation between the robot, camera, and world frames.
The method can in principle be extended to robots for which an analytic FK model is unavailable by identifying or learning the kinematics, but this moves the limitation into model identification and its uncertainty.

6.2 Ablation Studies

The final PhySVLA design retains the native `OSC_POSE` interface of the pre-trained VLA and maps the joint-space field gradient into that action space, as described in Chapter 4. This choice was not made only for implementation convenience. During development, two alternatives were investigated to apply the safety correction directly in the coordinates in which the distance field is differentiated.
The first changed the policy output by fine-tuning $\pi_{0.5}$ to predict robot joint targets; the second retained the frozen VLA but replaced, and subsequently modified, the low-level controller so that the safety problem could be solved in joint space. This section reports the implementation and the diagnostic evidence that led to rejecting both variants for the final evaluation. These experiments are development ablations and are therefore not part of the 1,200-episode comparison in Chapter 5.

6.2.1 VLA with joint-space output

The most direct way to use a joint-space distance gradient is to make the VLA produce a joint-space command.
To test this hypothesis, the original LIBERO demonstrations from the Spatial, Object, and Goal suites were converted into a single joint-space dataset. It contains 30 tasks, 1,500 demonstrations, and 198,985 transitions at 10 Hz. Each observation contains the external and wrist RGB images, the language instruction, the seven robot joint positions q_t, and one gripper-width value w_t.
The supervised target is

$$a_t^J = \begin{bmatrix} q_{t+1} \\ g_t \end{bmatrix} \in \mathbb{R}^8, \tag{6.1}$$

where q_{t+1} is the next absolute joint configuration recorded in the demonstration and $g_t \in [-1,1]$ is the gripper command that was actually executed. During training, only the arm coordinates are converted to $\Delta q_t = q_{t+1} - q_t$ and normalized; the output transform reconstructs absolute joint targets at inference.

The LIBERO-fine-tuned $\pi_{0.5}$ checkpoint was adapted with LoRA [33] in both the vision–language backbone and the action expert. The policy predicts chunks

of ten targets and was trained for 25,000 optimization steps with a batch size of 16. The final logged flow-matching loss, shown in Figure 6.5, was 0.0086; however, this quantity alone is not a reliable closed-loop criterion because consecutive joint states in the demonstrations are very close. In particular, a nearly stationary prediction can obtain a small one-step joint error while failing to execute the longer manipulation trajectory.

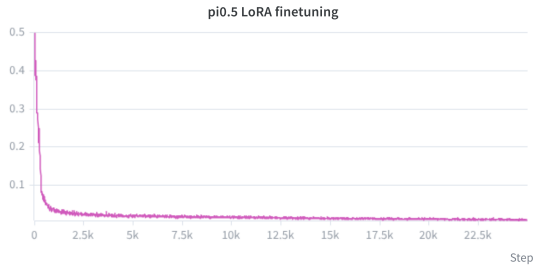


Figure 6.5: $\pi_{0.5}$ fine-tuned loss with LoRA.

At deployment, the environment controller was changed from `OSC_POSE` to `JOINT_POSITION`. If $\hat{q}_t^\star$ denotes the first predicted absolute target, the normalized controller command is

$$u_t^J = \mathrm{clip}\left(\frac{\hat{q}_t^\star - q_t}{s_q}, -\mathbf{1}, \mathbf{1}\right), \qquad s_q = 0.05\,\mathrm{rad}. \tag{6.2}$$

The scale s_q is the output range of the Robosuite joint-position controller. Replanning after every step was used for the suite-level diagnostic rollout, so the policy observed the state resulting from each previous joint command.
The same representation also permits a direct CDF correction [23]. Given the task-filtered scene cloud $\mathcal{Y}_t$, the tested filter forms $h_t = d(q_t, \mathcal{Y}_t) - m$ and projects the predicted increment through

$$\begin{aligned} \Delta q_t^{\mathrm{safe}} &= \arg\min_{\Delta q} \frac{1}{2} \left\| \Delta q - \Delta q_t^{\mathrm{nom}} \right\|_2^2, \\ \text{s.t.} \quad \nabla_q d_t^\top \Delta q &\geq b_t, \qquad b_t = \min(-\alpha h_t, 0). \end{aligned} \tag{6.3}$$

The clipping of b_t to non-positive values was an intentional minimal intervention choice: inside the nominal margin, the filter cancels further approach but does not force the arm to retreat when the VLA is already stationary, tangent to the obstacle, or moving away. The gripper command is left unchanged.

Observed behaviour. The policy learned meaningful joint trajectories on part of the training domain, but its performance was highly task-dependent. Table 6.1

summarizes the closed-loop rollout artifacts that were completed. The Spatial suite was evaluated in full, whereas the Goal run was interrupted after four of ten tasks and is included only as a diagnostic, not as a suite-level benchmark.

Table 6.1: Preliminary closed-loop evaluation of the joint-output $\pi_{0.5}$ policy on standard LIBERO initial states. The Goal result is partial and is reported only to document the ablation.

Suite	Evaluation coverage	Successes / rollouts	TSR (%)
LIBERO Spatial	10 of 10 tasks	254 / 500	50.8
LIBERO Goal	4 of 10 tasks	26 / 188	13.8

Several interacting causes explain this outcome. First, the training trajectories were generated with an operational-space controller and then relabelled with the observed joint states. At deployment, the same targets were executed by a joint-position controller with different stiffness, tracking dynamics, and null-space behaviour. Thus, even a correct one-step label does not reproduce the observation transition on which the demonstration was collected. The resulting controller-induced covariate shift compounds in closed loop.
Second, Cartesian task motion does not define a unique joint target for a seven-degree-of-freedom arm. The dataset records one solution along one demonstrated kinematic branch, while other configurations can realize the same end-effector pose. The experiment does not establish that joint-output VLAs are intrinsically ineffective. A stronger version would require demonstrations collected under the same joint controller used at deployment, more diverse configurations and recovery trajectories, a forward-kinematics consistency loss in task space, and closed-loop rather than one-step model selection. Collecting such data and retuning the policy–controller pair required substantially more time than was available within this thesis.

6.2.2 VLA with a joint-space controller

The second experiment kept the original frozen $\pi_{0.5}$ policy but replaced the low-level `OSC_POSE` controller with `JOINT_POSITION`. Because the VLA still produced a normalized Cartesian pose delta a_t^{OSC}, an adapter first computed a nominal joint increment by damped weighted inverse kinematics:

$$\Delta q_t^{\mathrm{nom}} = \arg\min_{\Delta q} \left\| J(q_t)\Delta q - S_{\mathrm{OSC}} a_t^{\mathrm{OSC}} \right\|_W^2 + \lambda^2 \|\Delta q\|_2^2, \tag{6.4}$$

followed by the joint and controller bounds. A seven-dimensional QP could then minimally project this increment against the active PhySVLA constraints.

This version failed before safety was enabled. In the adapter-only run, two joint coordinates reached the controller limit of ± 0.05 rad in approximately 65% of the steps; the arm oscillated and did not complete the grasp. The failure therefore cannot be attributed to the CBF. A one-step pseudoinverse reproduces only the instantaneous Cartesian increment. It does not reproduce the internal goal integration, impedance, orientation dynamics, torque compensation, or null-space policy of the OSC controller on which $\pi_{0.5}$ was trained. This experiment provided direct evidence that changing the controller underneath a frozen VLA is not behaviourally neutral.

Overall, both ablations were technically informative but insufficiently mature for the final benchmark. The joint-output policy made safety correction simple but sacrificed the frozen, controller-compatible VLA and showed weak, task-dependent competence. Direct controller replacement broke the nominal policy behaviour. These findings motivate the final PhySVLA design: retain the policy and its native controller, preserve nominal actions by construction, and map the physics-informed joint-space potential gradient into the action coordinates at the execution boundary.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

This thesis investigated whether explicit physical knowledge can improve the safety of frozen, generalist Vision–Language–Action (VLA) policies without retraining them or replacing the manipulation behaviour they have already learned. Within the considered experimental setting, the results provide a positive answer. Execution-time geometric reasoning substantially reduces collisions and increases safe task completion, while the original policy remains responsible for understanding the instruction and generating task-directed motion.
To achieve this objective, this work introduced *PhySVLA*, a task-aware, plug-and-play safety layer based on a differentiable, physics-informed potential field. The field is learned from the robot geometry and forward kinematics alone, without task demonstrations, visual observations, or scene-specific obstacle data. At inference time, multi-view RGB-D observations are reconstructed into a task-filtered point cloud. Querying the field against this previously unseen geometry yields joint-space clearance gradients for multiple robot links and, after grasping, for the transported payload. These gradients provide local repulsion from unsafe configurations and are mapped into the native `OSC_POSE` action space. A Control Barrier Function projection then computes the minimum admissible correction to the nominal VLA command. Consequently, the attractive, task-directed behaviour of the VLA and the repulsive geometric guidance of the safety field act through a common control interface, without modifying the policy weights.

The method was evaluated with two frozen VLA backbones, OpenVLA-OFT and $\pi_{0.5}$, against nine VLA and geometry-aware baseline configurations. The

common evaluation comprises 1,200 SafeLIBERO episodes per configuration across the Spatial, Object, and Goal suites and Safety Levels I and II. This evaluation supports two main conclusions. First, PhySVLA transfers across both action policies while using the same geometric safety mechanism. With OpenVLA-OFT, it raises the Task Success Rate (TSR) from 28.2% to 32.1%, the Collision Avoidance Rate (CAR) from 13.4% to 55.9%, and the Safe Success Rate (SSR) from 6.9% to 26.2%. Second, its strongest performance is obtained with $\pi_{0.5}$, for which TSR, CAR, and SSR increase from 62.7%, 18.2%, and 16.7% to 70.9%, 77.2%, and 61.0%, respectively. This is the best aggregate result for all three positive metrics among the evaluated configurations. Relative to $\pi_{0.5}$ + AEGIS, PhySVLA improves TSR by 5.2%, CAR by 10.2%, and SSR by 15.1%.
PhySVLA improves over both frozen policies at both safety levels, demonstrating that its effect is not confined to one backbone or obstacle placement.
The paired analysis further qualifies the aggregate improvement. Among the 1,000 episodes that are not safe successes for frozen $\pi_{0.5}$, PhySVLA recovers 612. Conversely, it loses 80 of the 200 nominal safe successes. The large positive balance confirms that the method changes outcomes rather than merely benefiting from a different episode mixture, but also shows that the intervention is not harmless in every state. PhySVLA should therefore be interpreted as an effective empirical risk-reduction mechanism, not as an unconditional safety certificate.

7.2 Future Work

7.2.1 Long-horizon action correction

In a long-horizon task, an object may serve as an intended contact target during one subtask and become an obstacle in a subsequent one. Handling this setting therefore requires a phase-estimation mechanism capable of tracking task progress and dynamically updating the semantic roles used to filter the point cloud. Such phase estimation is beyond the scope of this thesis and is therefore not considered. Consequently, the SafeLIBERO Long suite is excluded from the evaluation.

7.2.2 Real-world and embodiment transfer

Testing OpenVLA-OFT and $\pi_{0.5}$ provides evidence of architectural modularity, but does not establish universal policy independence. Future experiments should cover further VLA and world-model backbones, alternative action representations, and robots with different kinematic structures. Since the present field explicitly uses forward kinematics, a new embodiment still requires robot-specific geometric training; reducing this cost through compositional link models or cross-embodiment pre-training is an important open direction.

Physical deployment is the next essential validation step. A real system could combine an open-vocabulary vision-language model such as Qwen2.5-VL [34] with SAM 3 [35] to construct the task-aware obstacle cloud from calibrated RGB-D observations. The evaluation must then account for depth noise, occlusion, calibration error, inference latency, controller compliance, and payload slip. Moving obstacles additionally require temporal scene fusion and predictive, velocity-aware constraints rather than a sequence of independent static reconstructions.

7.3 Final Remarks

PhySVLA establishes a practical division of responsibilities for safe generalist robot manipulation: the foundation policy supplies semantic understanding and task intent, while a reusable physics-informed potential field supplies explicit geometric guidance at execution time. The results show that this separation can produce large gains in safe task completion on two substantially different frozen policies, without discarding their learned competence or collecting new task demonstrations. At the same time, the suite- and safety-level analyses identify the remaining challenge: local safety correction must be coordinated with perception, contact intent, and the recovery capability of the nominal policy.
By demonstrating both the effectiveness and the boundaries of this approach, this thesis provides a concrete foundation for future systems that combine general-purpose learned intelligence with explicit physical reasoning.

Bibliography

[1] Anthony Brohan et al. «RT-1: Robotics Transformer for Real-World Control at Scale». In: *Proceedings of Robotics: Science and Systems*. Daegu, Republic of Korea, July 2023. DOI: 10.15607/RSS.2023.XIX.025. arXiv: 2212.06817 [cs.RO]. URL: https://robotics-transformer1.github.io/ (cit. on pp. 5, 10).

[2] Dibya Ghosh et al. «Octo: An Open-Source Generalist Robot Policy». In: *Proceedings of Robotics: Science and Systems*. Delft, Netherlands, July 2024. DOI: 10.15607/RSS.2024.XX.090. arXiv: 2405.12213 [cs.RO]. URL: https://octo-models.github.io/ (cit. on pp. 5, 10).

[3] Moo Jin Kim et al. «OpenVLA: An Open-Source Vision-Language-Action Model». In: *Proceedings of The 8th Conference on Robot Learning*. Ed. by Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard. Vol. 270. Proceedings of Machine Learning Research. PMLR, Nov. 2025, pp. 2679–2713. arXiv: 2406.09246 [cs.RO]. URL: https://proceedings.mlr.press/v270/kim25c.html (cit. on pp. 5, 10, 36).

[4] Kevin Black et al. «π_0: A Vision-Language-Action Flow Model for General Robot Control». In: *Proceedings of Robotics: Science and Systems*. Los Angeles, CA, USA, June 2025. DOI: 10.15607/RSS.2025.XXI.010. arXiv: 2410.24164 [cs.LG]. URL: https://www.roboticsproceedings.org/rss21/p010.html (cit. on pp. 5, 11).

[5] Physical Intelligence. *$\pi_{0.5}$: a VLA with Open-World Generalization*. Apr. 22, 2025. URL: https://www.pi.website/blog/pi05 (cit. on p. 5).

[6] Songqiao Hu, Zeyi Liu, Shuang Liu, Jun Cen, Zihan Meng, Shihefeng Wang, Xiang Li, and Xiao He. *VLSA: Vision-Language-Action Models with Plug-and-Play Safety Constraint Layer*. Accepted to the 2026 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). 2025. arXiv: 2512.11891 [cs.RO]. URL: https://vlsa-aegis.github.io/ (cit. on pp. 10, 13, 14, 26–28, 48, 53, 62, 63).

[7] Haocheng Zhao, Lukas Brunke, Oliver Lagerquist, Siqi Zhou, and Angela P. Schoellig. *SQ-CBF: Signed Distance Functions for Numerically Stable Superquadric-Based Safety Filtering*. 2026. arXiv: 2602.11049 [cs.RO] (cit. on pp. 10, 15).

[8] Lizhi Yang, Ryan M. Bena, Meg Wilkinson, Gilbert Bahati, Andy Navarro Brenes, Ryan K. Cosner, and Aaron D. Ames. *Safe-SAGE: Social-Semantic Adaptive Guidance for Safe Engagement through Laplace-Modulated Poisson Safety Functions*. Accepted to the 2026 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). 2026. arXiv: 2603.05497 [cs.RO] (cit. on pp. 10, 15, 16).

[9] Yunzhou Song et al. *OmniGuide: Universal Guidance Fields for Enhancing Generalist Robot Policies*. 2026. arXiv: 2603.10052 [cs.RO]. URL: https://omniguide.github.io/ (cit. on pp. 10, 16, 17, 28, 63).

[10] Yubai Wei, Chen Wu, and Hashem Haghbayan. *Can Explicit Physical Feasibility Benefit VLA Learning? An Empirical Study*. 2026. arXiv: 2604.17896 [cs.LG] (cit. on pp. 10, 18).

[11] Guoheng Sun, Tingting Du, Kaixi Feng, Chenxiang Luo, Xingguo Ding, Zheyu Shen, Ziyao Wang, Yexiao He, and Ang Li. *ROCKET: Residual-Oriented Multi-Layer Alignment for Spatially-Aware Vision-Language-Action Models*. 2026. arXiv: 2602.17951 [cs.CV]. URL: https://github.com/CASE-Lab-UMD/ROCKET-VLA (cit. on pp. 10, 19, 63).

[12] Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. *Fast-WAM: Do World Action Models Need Test-Time Future Imagination?* 2026. arXiv: 2603.16666 [cs.CV]. URL: https://yuantianyuan01.github.io/FastWAM/ (cit. on pp. 10, 20, 21, 63).

[13] Jisang Han et al. *Geometric Action Model for Robot Policy Learning*. 2026. arXiv: 2606.17046 [cs.RO]. URL: https://cvlab-kaist.github.io/Geometric-Action-Model/ (cit. on pp. 10, 20, 21, 63).

[14] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin C. M. Burchfiel, and Shuran Song. «Diffusion Policy: Visuomotor Policy Learning via Action Diffusion». In: *Proceedings of Robotics: Science and Systems*. Daegu, Republic of Korea, July 2023. DOI: 10.15607/RSS.2023.XIX.026. arXiv: 2303.04137 [cs.RO]. URL: https://diffusion-policy.cs.columbia.edu/ (cit. on p. 11).

[15] 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». In: *International Conference on Learning Representations (ICLR)*. 2025. arXiv: 2410.07864 [cs.RO]. URL: https://rdt-robotics.github.io/rdt-robotics/ (cit. on p. 11).

[16] Kevin Black et al. «$\pi_{0.5}$: a Vision-Language-Action Model with Open-World Generalization». In: *Proceedings of The 9th Conference on Robot Learning*. Ed. by Joseph Lim, Shuran Song, and Hae-Won Park. Vol. 305. Proceedings of Machine Learning Research. PMLR, Sept. 2025, pp. 17–40. URL: https://proceedings.mlr.press/v305/black25a.html (cit. on pp. 11, 37–39, 53, 62).

[17] Moo Jin Kim, Chelsea Finn, and Percy Liang. «Fine-Tuning Vision-Language-Action Models: Optimizing Speed and Success». In: *Proceedings of Robotics: Science and Systems*. Los Angeles, CA, USA, June 2025. DOI: 10.15607/RSS.2025.XXI.017. URL: https://www.roboticsproceedings.org/rss21/p017.html (cit. on pp. 11, 37, 53, 62).

[18] Aaron D. Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. «Control Barrier Functions: Theory and Applications». In: *2019 18th European Control Conference (ECC)*. IEEE, 2019, pp. 3420–3431. DOI: 10.23919/ECC.2019.8796030 (cit. on p. 12).

[19] Yiming Li, Yan Zhang, Amirreza Razmjoo, and Sylvain Calinon. «Representing Robot Geometry as Distance Fields: Applications to Whole-Body Manipulation». In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 15351–15357. DOI: 10.1109/ICRA57147.2024.10611674 (cit. on p. 23).

[20] Joseph Ortiz, Alexander Clegg, Jing Dong, Edgar Sucar, David Novotny, Michael Zollhoefer, and Mustafa Mukadam. «iSDF: Real-Time Neural Signed Distance Fields for Robot Perception». In: *Proceedings of Robotics: Science and Systems*. New York City, NY, USA, June 2022. DOI: 10.15607/RSS.2022.XVIII.012. arXiv: 2204.02296 [cs.RO]. URL: https://joeaortiz.github.io/iSDF/ (cit. on p. 23).

[21] Puze Liu, Kuo Zhang, Davide Tateo, Snehal Jauhri, Jan Peters, and Georgia Chalvatzaki. «Regularized Deep Signed Distance Fields for Reactive Motion Generation». In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 6673–6680. DOI: 10.1109/IROS47612.2022.9981456. arXiv: 2203.04739 [cs.RO] (cit. on pp. 23, 24).

[22] Mikhail Koptev, Nadia Figueroa, and Aude Billard. «Neural Joint Space Implicit Signed Distance Functions for Reactive Robot Manipulator Control». In: *IEEE Robotics and Automation Letters* 8.2 (Feb. 2023), pp. 480–487. DOI: 10.1109/LRA.2022.3227860 (cit. on pp. 24, 25, 28, 40, 43).

[23] Yiming Li, Xuemin Chi, Amirreza Razmjoo, and Sylvain Calinon. «Configuration Space Distance Fields for Manipulation Planning». In: *Proceedings of Robotics: Science and Systems*. Delft, Netherlands, July 2024. DOI: 10.15607/

RSS.2024.XX.131. URL: https://sites.google.com/view/cdfmp/home (cit. on pp. 25, 28, 77).

[24] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. «LIBERO: Benchmarking Knowledge Transfer for Lifelong Robot Learning». In: *Advances in Neural Information Processing Systems*. Vol. 36. 2023, pp. 44776–44791. URL: https://proceedings.neurips.cc/paper_files/paper/2023/hash/8c3c666820ea055a77726d66fc7d447f-Abstract-Datasets_and_Benchmarks.html (cit. on pp. 26, 27).

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. «Attention Is All You Need». In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017, pp. 5998–6008 (cit. on p. 32).

[26] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. «Visual Instruction Tuning». In: *Advances in Neural Information Processing Systems*. Vol. 36. Curran Associates, Inc., 2023, pp. 34892–34916. DOI: 10.52202/075280-1516. URL: https://proceedings.neurips.cc/paper_files/paper/2023/hash/6dcf277ea32ce3288914faf369fe6de0-Abstract-Conference.html (cit. on p. 33).

[27] Alexey Dosovitskiy et al. «An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale». In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=YicbFdNTTy (cit. on p. 33).

[28] Maxime Oquab et al. *DINOv2: Learning Robust Visual Features without Supervision*. 2024. arXiv: 2304.07193 [cs.CV]. URL: https://arxiv.org/abs/2304.07193 (cit. on p. 36).

[29] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. *Sigmoid Loss for Language Image Pre-Training*. 2023. arXiv: 2303.15343 [cs.CV]. URL: https://arxiv.org/abs/2303.15343 (cit. on p. 36).

[30] Hugo Touvron et al. *Llama 2: Open Foundation and Fine-Tuned Chat Models*. 2023. arXiv: 2307.09288 [cs.CL]. URL: https://arxiv.org/abs/2307.09288 (cit. on p. 36).

[31] Amit Rogel, Richard Savery, Ning Yang, and Gil Weinberg. «RoboGroove: Creating Fluid Motion for Dancing Robotic Arms». In: *Proceedings of the 8th International Conference on Movement and Computing*. MOCO '22. New York, NY, USA: Association for Computing Machinery, 2022, 7:1–7:9. DOI: 10.1145/3537972.3537985. URL: https://doi.org/10.1145/3537972.3537985 (cit. on p. 41).

[32] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. «Sigmoid-Weighted Linear Units for Neural Network Function Approximation in Reinforcement Learning». In: *Neural Networks* 107 (2018), pp. 3–11. DOI: 10.1016/j.neunet.2017.12.012 (cit. on p. 49).

[33] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. «LoRA: Low-Rank Adaptation of Large Language Models». In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=nZeVKeeFYf9 (cit. on p. 76).

[34] Shuai Bai et al. «Qwen2.5-VL Technical Report». In: (2025). arXiv: 2502.13923 [cs.CV] (cit. on p. 82).

[35] Nicolas Carion et al. *SAM 3: Segment Anything with Concepts*. 2025. arXiv: 2511.16719 [cs.CV]. URL: https://arxiv.org/abs/2511.16719 (cit. on p. 82).