GEARS: SEEING GEOMETRY, DIFFUSING ACTIONS FOR ZERO-SHOT SIM-TO-REAL DEXTEROUS MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Simulation provides scalable interaction data and privileged geometric labels for dexterous manipulation, yet zero-shot deployment remains limited by coupled visual and physical mismatches. Appearance changes alter the student's visual observations, while hidden simulator states and randomized physical parameters can make multiple expert actions plausible from the same deployable observation. We present Geometric Action Representation for Sim-to-Real (GEARS), a three-stage framework that uses privileged information only during training. GEARS first fine-tunes a visual foundation model with joint inverse-depth and surface-normal prediction and a cross-task consistency objective. This produces a frozen geometric encoder that consumes only RGB at deployment. GEARS then trains a privileged reinforcement learning expert with Virtual Particle Stein Variational Gradient Descent (VP-SVGD), which uses an estimated failure landscape to adapt ten object-pose and physical randomization dimensions. Finally, a diffusion student conditioned on the frozen geometric features and proprioception learns the expert action distribution after hidden simulator information is marginalized. The central design links perception and action modeling: geometric supervision preserves contact-relevant scene structure in the RGB representation, while diffusion represents the action variation induced by unobserved state and physics. Across four chemistry-laboratory tasks on a fixed dual-arm platform, GEARS achieves a 67.5% real-world success rate without real-world policy training, substantially outperforming the strongest controlled baseline at 42.5%, with gains on all four tasks. These results demonstrate the effectiveness of jointly addressing visual domain shift through geometric representation learning and action ambiguity through diffusion-based policy modeling for zero-shot sim-to-real dexterous manipulation.

1 INTRODUCTION

Learning manipulation policies that transfer beyond their training conditions is a central goal of embodied intelligence (Black et al., 2025; Kim et al., 2025; Zitkovich et al., 2023; Ghosh et al., 2024). Simulation supports this goal with scalable interaction data and privileged labels that are expensive to obtain on physical systems (Tobin et al., 2017). However, a policy trained only in simulation must tolerate changes in both visual observations and physical dynamics when it is deployed on real hardware (Tobin et al., 2017; Peng et al., 2018). Robust zero-shot transfer therefore requires both a stable observation interface and a policy that can act under uncertain physical conditions.

The observation gap is especially difficult for transparent and reflective objects. Synthetic RGB images contain renderer-specific textures and lighting, while physical depth sensors can return noisy or missing measurements on these surfaces (James et al., 2019; Sajjan et al., 2020). A manipulation policy does not require photorealistic reconstruction, but it does require object boundaries, shape, and surface orientation for contact. These geometric quantities are less tied to rendering appearance than raw RGB and can be supervised directly in simulation.

The dynamics gap creates a related distillation problem. Automatic domain randomization (ADR) can train a reinforcement learning (RL) expert over varied physical parameters (Akkaya et al., 2019). The expert observes the randomized parameters and full simulator state, but a deployable student observes only images and proprioception. Several privileged states can therefore be compatible with

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

the same student observation yet require different contact strategies; objects with different hidden masses, for example, may require different grasp configurations or motion timing. Mean-squared-error behavior cloning reduces these strategies to a conditional mean, which may lie between feasible actions (Florence et al., 2022). Predicting actions independently at every control step can also produce temporally inconsistent commands during contact-rich execution (Chi et al., 2023; Zhao et al., 2023). Perception and action modeling are thus coupled at the student interface: the visual representation controls which states remain distinguishable, and the policy class controls whether their actions remain distinct.

We propose GEARS (**GE**ometric **A**ction **R**epresentation for **S**im-to-Real) to address this coupled interface. A visual foundation model (Yang et al., 2024; Oquab et al., 2024) is fine-tuned with privileged inverse-depth and surface-normal supervision, including a consistency objective between the two prediction tasks. The prediction heads are omitted from the deployed system, leaving a frozen geometric encoder that consumes RGB at deployment. A privileged expert is trained with Virtual Particle Stein Variational Gradient Descent (VP-SVGD), which uses an estimated failure landscape to adapt the object-pose and physical randomization bounds. A diffusion student (Chi et al., 2023) then approximates the expert distribution that remains after the hidden simulator states and parameters are marginalized. It predicts action chunks and replans after four actions, limiting open-loop execution.

The central contribution of GEARS is a sim-to-real distillation design that connects what the student can perceive with how it represents expert actions. Privileged geometry supervises an RGB encoder that retains contact-relevant shape and surface cues without using depth at deployment. VP-SVGD uses a learned failure landscape to adjust where the privileged expert is trained while preserving uniform sampling within simulator-compatible bounds. Because the student cannot observe the expert's full state or randomized physics, the resulting demonstrations define a conditional distribution rather than a single action target. The diffusion student models this distribution with receding-horizon action chunks. Thus geometry addresses the visual gap, while diffusion addresses the action ambiguity created by privileged expert training. We test each part of this design through controlled comparisons of the visual encoder, action model, and randomization strategy. A 2×2 encoder-policy experiment further measures their interaction. On a fixed dual-arm platform, GEARS succeeds in 54 of 80 real-world trials across four tasks. The strongest controlled baseline succeeds in 34 of 80 trials.

2 Related Work

Sim-to-real transfer for manipulation. Sim-to-real methods address observation mismatch, dynamics mismatch, or both. Domain adaptation aligns visual observations through image translation (James et al., 2019) or neural rendering (Qureshi et al., 2025). System identification calibrates dynamics through Bayesian inference (Ramos et al., 2019) or iterative optimization (Chebotar et al., 2019). Domain randomization instead trains over a distribution of simulator parameters (Tobin et al., 2017) and has enabled dexterous transfer (Akkaya et al., 2019; Handa et al., 2023; Lum et al., 2024b;a). Standard ADR samples uniformly within axis-aligned bounds and expands those bounds according to policy success (Akkaya et al., 2019). VP-SVGD instead estimates how failure varies inside the current range and transports particles toward difficult regions. Particle percentiles update the bounds, but simulator rollouts remain uniformly sampled within resulting axis-aligned box. The key distinction is that adaptation follows a nonuniform failure landscape without replacing the simulator's uniform sampling interface.

Privileged experts are often distilled into deployable students (Chen et al., 2020; Kumar et al., 2021; Cheng et al., 2025; Lin et al., 2025). Our setting adds a distributional ambiguity: simulator states and randomized parameters hidden from the student can map one deployable observation to several expert actions. We use diffusion to model the resulting marginalized action distribution, not to estimate the hidden parameters. Generalist vision-language-action policies learn broad skills from large robot datasets (Black et al., 2025; Kim et al., 2025; Zitkovich et al., 2023; Ghosh et al., 2024). Their training data, embodiments, and action spaces differ from our fixed-platform zero-shot setting. We therefore treat them as complementary approaches rather than controlled baselines.

Visual representations for robotic manipulation. The visual encoder determines which scene properties remain available to a deployed policy across the observation gap (Yardi et al., 2025). Generic robot encoders include R3M (Nair et al., 2023), MVP (Radosavovic et al., 2023), VC-

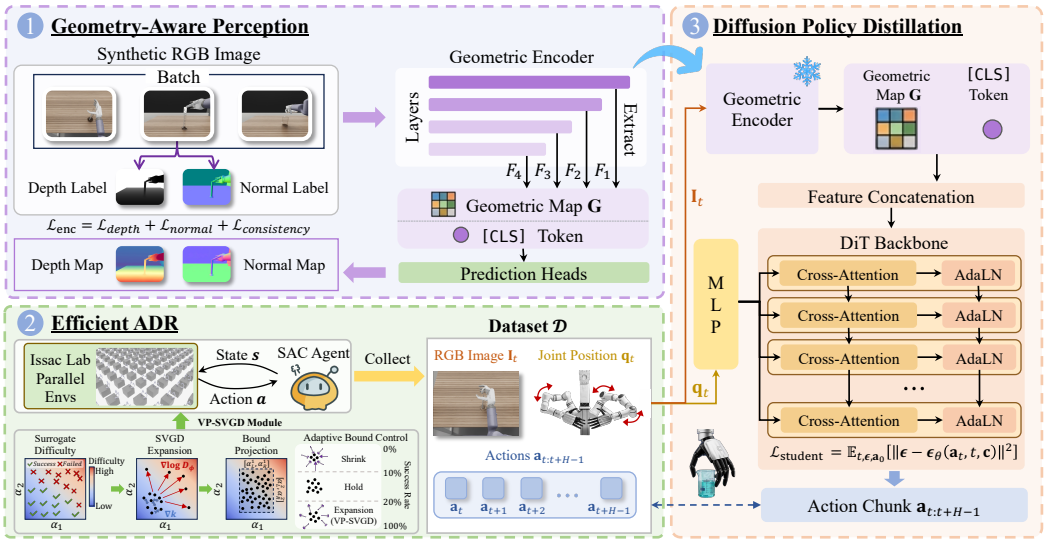


Figure 1: Overview of Geometric Action Representation for Sim-to-Real (GEARS). Stage 1: Synthetic inverse depth and surface normals supervise a geometric encoder. Both prediction heads are omitted from the deployed system. Stage 2: A privileged reinforcement learning expert is trained with Virtual Particle Stein Variational Gradient Descent over ten adaptive object-pose and physical dimensions. Stage 3: A diffusion student uses frozen geometric features and proprioception to generate receding-horizon action chunks. Only the two RGB streams, proprioception, geometric encoder, and student policy are retained for deployment.

1 (Majumdar et al., 2023), SPA (Zhu et al., 2024), and Theia (Shang et al., 2024). A complementary line predicts monocular geometry through MiDaS (Ranftl et al., 2022), Depth Anything (Yang et al., 2024), Metric3D (Yin et al., 2023), or the Dense Prediction Transformer (DPT) (Ranftl et al., 2021). We connect these directions by using simulator geometry to supervise an intermediate policy representation. Depth and normals are training objectives rather than deployed observations. The resulting policy retains dense spatial features while requiring neither real-image supervision nor a physical depth stream.

Generative models for robot control. Generative policies replace point regression with structured action distributions. Implicit Behavioral Cloning (Florence et al., 2022) learns an energy function over actions. Diffusion Policy (Chi et al., 2023) generates action chunks through iterative denoising, while DP3 (Ze et al., 2024) conditions this process on 3D observations. Action Chunking with Transformers (ACT) (Zhao et al., 2023) uses a conditional variational autoencoder, and VQ-BeT (Lee et al., 2024) represents actions with hierarchical vector quantization. Unlike direct policy learning from deployable observations, we use diffusion to distill a privileged, domain-randomized expert after its hidden state and physics have been marginalized.

3 Method

GEARS uses privileged labels, simulator states, and randomized parameters during training. At deployment, it receives only RGB images and proprioception. As shown in Fig. 1, the pipeline has three stages: learning a geometric observation space (§3.1), training a privileged RL expert over adaptive physical variation (§3.2), and distilling the expert into a diffusion student (§3.3).

The three stages form a training-to-deployment pipeline. Geometry supervision uses simulator labels to train an RGB encoder. VP-SVGD selects the physical conditions used to train the privileged expert. The diffusion student learns from these demonstrations using only observations available at deployment. Privileged geometry, simulator state, and randomized parameters therefore guide training but are not required at inference.

3.1 GEOMETRY-AWARE PERCEPTION

Depth and surface orientation are less directly tied to texture and lighting than raw RGB appearance (James et al., 2019; Tobin et al., 2017). We therefore use them as training signals rather than deployed observations. We fine-tune a pretrained visual model to predict both quantities from monocular RGB, supervised by labels rendered in simulation. Joint supervision encourages the shared representation to retain spatial structure that is relevant to contact.

Architecture. The encoder builds on the Small variant of Depth Anything V2 (Yang et al., 2024). This model pairs a DINOv2 Vision Transformer (ViT) backbone with a DPT fusion neck (Ranftl et al., 2021). Given an RGB image $\mathbf{I} \in \mathbb{R}^{3\times H_I \times W_I}$, the ViT backbone extracts features $\{\mathbf{F}_i\}_{i=1}^{4}$ from four intermediate transformer layers together with a global classification `[CLS]` token. The DPT neck fuses these features into a shared geometric map $\mathbf{G} \in \mathbb{R}^{C\times H_G \times W_G}$. We refer to the backbone and fusion neck together as the *geometric encoder*. From $\mathbf{G}$, two lightweight heads predict inverse depth $\hat{\mathbf{u}} \in \mathbb{R}^{1\times H_I \times W_I}$ and unit surface normals $\hat{\mathbf{n}} \in \mathbb{R}^{3\times H_I \times W_I}$. These heads provide complementary geometric supervision and are omitted from the deployed system. The student receives two outputs from the frozen encoder. A learned 1×1 projection of $\mathbf{G}$ preserves dense spatial structure, while the `[CLS]` token provides a global scene summary. The encoder remains fixed throughout student-policy training.

Model Training. The encoder and prediction heads are trained jointly on RGB-depth-normal triplets rendered in Isaac Lab (Mittal et al., 2025). Color jitter varies appearance. Geometric image perturbations are applied consistently to all labels, and normal vectors are rotated with the image. The objective combines inverse-depth, normal, and cross-task consistency terms:

$$\mathcal{L}_{\text{enc}} = \underbrace{\lambda_{\text{ssi}}\mathcal{L}_{\text{ssi}} + \lambda_{\text{gm}}\mathcal{L}_{\text{gm}}}_{\text{depth}} + \underbrace{\lambda_{\text{cos}}\mathcal{L}_{\text{cos}} + \lambda_{\text{geo}}\mathcal{L}_{\text{geo}}}_{\text{normal}} + \underbrace{\lambda_{\text{dn}}\mathcal{L}_{\text{dn}}}_{\text{consistency}} . \tag{1}$$

The scale- and shift-invariant term $\mathcal{L}_{\text{ssi}}$ (Ranftl et al., 2022) supervises relative depth, while $\mathcal{L}_{\text{gm}}$ preserves depth discontinuities. For surface normals, $\mathcal{L}_{\text{cos}}$ measures angular error and $\mathcal{L}_{\text{geo}}$ encourages local coherence away from depth boundaries.

The consistency term $\mathcal{L}_{\text{dn}}$ connects the two prediction tasks. We back-project the aligned depth prediction using camera intrinsics $\mathbf{K}$, derive normals from finite differences, and compare them with the output of the normal head:

$$\mathcal{L}_{\text{dn}} = \frac{1}{|\mathcal{M}|}\sum_{p\in\mathcal{M}}\left(1 - \hat{\mathbf{n}}_p^{\top}\mathbf{n}_p^{(\text{d})}\right) , \tag{2}$$

where $\mathbf{n}_p^{(\text{d})}$ is derived from the aligned inverse-depth prediction and camera intrinsics. The mask $\mathcal{M}$ contains valid pixels and their finite-difference neighbors. Appendix C.1 defines the alignment, mask, and all loss terms.

3.2 ADAPTIVE RL EXPERT TRAINING

The second stage trains a privileged RL expert with ground-truth object state, contact state, and randomized parameters. The expert must cover task-relevant physical variation and provide demonstrations for the vision-based student. Automatic domain randomization varies the environment, while VP-SVGD adapts the bounds of that variation.

Task Formulation. We formulate each manipulation task as a family of Markov decision processes (MDPs) indexed by simulator parameters $\boldsymbol{\xi}$. Each MDP is $(\mathcal{S}, \mathcal{A}, \mathcal{P}_{\boldsymbol{\xi}}, \mathcal{R}, \gamma)$, where $\mathcal{R}$ is the reward function and γ is the discount factor. The privileged expert observes $\boldsymbol{\xi}$ together with joint positions and velocities, object poses and velocities, fingertip velocities, relative fingertip-object positions, contact state, and the previous action. The action space $\mathcal{A}$ consists of arm joint velocity targets and hand joint position targets. The parameter vector $\boldsymbol{\xi}$ contains object-pose, physical, visual, and camera variables. VP-SVGD adapts ten object-pose, mass, and contact dimensions. Object size, lighting, material, and camera parameters are sampled from fixed ranges. Appendix C.2 gives the complete mapping.

Massively Parallel SAC Training. We train the expert with Soft Actor-Critic (SAC) (Haarnoja et al., 2018) in GPU-parallel environments. SAC uses a maximum-entropy objective and off-policy replay. Appendix C.3 states the replay, target-network, and actor-update settings used in our 4,096-environment implementation.

Automatic Domain Randomization via VP-SVGD. Standard ADR samples from an axis-aligned uniform distribution and expands its bounds when the expert reaches a success threshold (Akkaya et al., 2019). This curriculum controls the sampled range but does not model how failure probability varies within it. VP-SVGD retains the uniform sampling interface and uses virtual particles to update the bounds from an estimated failure landscape.

We define a normalized adaptive space $\boldsymbol{\alpha} \in [-1, 1]^{10}$. Its origin is the nominal configuration, and its limits map to the ten adaptive ranges in Appendix C.2. We maintain M particles $\{\mathbf{v}^{(m)}\}_{m=1}^{M}$ in this space and update them in three steps. First, a lightweight difficulty model $D_\psi(\boldsymbol{\alpha})$ estimates failure probability from recent rollouts. It provides a differentiable landscape over the adaptive space. Second, the Stein Variational Gradient Descent (SVGD) update transports particles toward high-failure regions while maintaining coverage through kernel repulsion (Liu & Wang, 2016):

$$\boldsymbol{\phi}^*(\mathbf{v}^{(m)}) = \frac{1}{M}\sum_{j=1}^{M}\Big[\underbrace{k(\mathbf{v}^{(j)}, \mathbf{v}^{(m)})\nabla_{\mathbf{v}^{(j)}} \log D_\psi(\mathbf{v}^{(j)})}_{\text{driving force}} + \underbrace{\nabla_{\mathbf{v}^{(j)}} k(\mathbf{v}^{(j)}, \mathbf{v}^{(m)})}_{\text{repulsive force}}\Big], \tag{3}$$

where $k(\cdot, \cdot)$ is a radial basis function (RBF) kernel with median-heuristic bandwidth. In Eq. (3), the driving force follows the estimated failure landscape, and the repulsive force separates nearby particles. Third, the 5th and 95th particle percentiles define the next axis-aligned bounds. The particles do not enter the simulator directly. The simulator continues to sample uniformly within the projected bounds.

The update is gated by the current success rate. At success rates of at least 20%, the full VP-SVGD cycle executes. Below 5%, the particles contract toward the origin. Otherwise, the adaptive bounds remain fixed. Appendix C.2 gives the update parameters.

3.3 Distillation into a Diffusion Student

Domain randomization broadens expert coverage but turns distillation into a many-to-one problem. The expert observes the simulator state and randomized parameters, whereas the student receives only geometric image features and proprioception. We use a diffusion model as the student to represent the resulting conditional action distribution and generate action chunks.

Let $\boldsymbol{\xi}$ denote the randomized parameters and $\mathbf{s}_t$ the privileged state, including object and contact variables. The student observes $\mathbf{o}_t = (\mathbf{I}_t^{\mathrm{h}}, \mathbf{I}_t^{\mathrm{c}}, \mathbf{x}_t^{\mathrm{prop}})$ but not $(\mathbf{s}_t, \boldsymbol{\xi})$, where $\mathbf{I}_t^{\mathrm{h}}$ and $\mathbf{I}_t^{\mathrm{c}}$ are the head and chest RGB images and $\mathbf{x}_t^{\mathrm{prop}}$ is the proprioceptive vector. Let p_E denote the distribution induced by the successful expert rollouts used for distillation, and let $\mathbf{A}_t = \mathbf{a}_{t:t+H-1}$ denote a length-H action chunk. The student target marginalizes the hidden information:

$$p_E(\mathbf{A}_t \mid \mathbf{o}_t) = \mathbb{E}_{p(\mathbf{s}_t, \boldsymbol{\xi} \mid \mathbf{o}_t)}[p_E(\mathbf{A}_t \mid \mathbf{s}_t, \boldsymbol{\xi})]\ . \tag{4}$$

Equation (4) defines the target of distillation: a mixture over expert chunks associated with privileged conditions that are compatible with $\mathbf{o}_t$. The student does not recover $\boldsymbol{\xi}$ or remove partial observability. At deployment, the student predicts $H = 8$ actions, executes four at 30 Hz, and replans from the next observation. The resulting 7.5 Hz feedback loop limits the duration of open-loop execution.

Data Collection. We roll out the trained expert over the final randomized training distribution and retain successful trajectories to form $\mathcal{D} = \{(\mathbf{I}_t^{\mathrm{h}}, \mathbf{I}_t^{\mathrm{c}}, \mathbf{x}_t^{\mathrm{prop}}, \mathbf{a}_{t:t+H-1})\}$, where $\mathbf{a}_{t:t+H-1}$ contains H consecutive expert actions.

Architecture. The student uses the frozen geometric encoder as its observation backbone and a one-dimensional Diffusion Transformer (DiT) as its denoiser (Peebles & Xie, 2023). It models action chunks $\mathbf{a}_0 \in \mathbb{R}^{H \times d_a}$, where d_a is the per-step action dimension, with conditional denoising diffusion (Chi et al., 2023; Ho et al., 2020). Each camera is encoded independently, and the two feature sets enter the DiT through local and global conditioning pathways.

The dense feature maps are flattened into spatial tokens. An MLP maps proprioception to matching tokens. Together, they provide keys and values for cross-attention at every DiT layer, allowing the denoiser to attend to individual spatial and proprioceptive tokens.

In parallel, the global `[CLS]` tokens from both views are fused with pooled proprioception and the diffusion-time embedding. Adaptive Layer Normalization (AdaLN) uses this vector to modulate

Table 1: Zero-shot sim-to-real success rates on the fixed dual-arm platform. Each method uses the same privileged expert demonstrations and is evaluated over 20 real-world trials per task without real-world policy training. Cells report percentages. "Avg." is the unweighted mean over the four tasks. DP, BC, ACT, and Geo denote Diffusion Policy, behavior cloning, Action Chunking with Transformers, and the geometry-supervised encoder. Bold numbers mark the best value in each column, and underlining marks the second-best value.

Method	Grasping	Pouring	Handover	Lifting	Avg.
Varying visual encoder (DP fixed)					
DP + ResNet	15.0	5.0	0.0	10.0	7.5
DP + R3M	30.0	15.0	5.0	15.0	16.3
DP + DINOv2	45.0	20.0	10.0	20.0	23.8
Varying action model (geometric encoder fixed)					
BC + Geo	50.0	10.0	5.0	25.0	22.5
ACT + Geo	<u>70.0</u>	<u>35.0</u>	<u>25.0</u>	<u>40.0</u>	<u>42.5</u>
GEARS	**90.0**	**65.0**	**55.0**	**60.0**	**67.5**

each DiT layer (Peebles & Xie, 2023). For diffusion time τ, the cosine schedule perturbs the clean chunk $\mathbf{a}_0$ into $\mathbf{a}_\tau$ with Gaussian noise $\boldsymbol{\epsilon}$. Training minimizes

$$\mathcal{L}_{\text{student}} = \mathbb{E}_{\tau,\boldsymbol{\epsilon},\mathbf{a}_0}\left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{a}_\tau, \tau, \mathbf{c})\|^2\right], \tag{5}$$

where $\tau \sim \mathcal{U}[0,1]$, and $\mathbf{c}$ denotes the combined conditioning features. At inference, action chunks are generated with Denoising Diffusion Implicit Models (DDIM) sampling (Song et al., 2021).

4 Experiments

We evaluate the contribution of each GEARS stage to zero-shot transfer, the interaction between perception and action modeling, and the remaining failure modes.

4.1 Experimental Setup

Hardware. As shown in Fig. 2, the platform consists of two seven-degree-of-freedom (DoF) arms (RealMan RM75-6F), each equipped with a six-DoF dexterous hand (RY-H2). An Intel RealSense D435 head camera and D455 chest camera capture 640×480 RGB images. The geometric encoder processes each view independently, and the policy receives features from both views. Only the RGB channels are used; the RealSense depth streams are not provided to the policy.

Simulation. All training is conducted in Isaac Lab (Mittal et al., 2025) with up to 4,096 GPU-accelerated parallel environments. The renderer varies textures, lighting, materials, and camera poses. Simulator depth maps and surface normals provide encoder supervision (§3.1). Appendix C.2 lists the adaptive and fixed-range randomization parameters.

Tasks. We evaluate four chemistry-laboratory tasks (Appendix B.6). *Grasping* requires lifting a transparent beaker from a randomized pose to a target height. *Pouring* requires grasping a beaker and transferring a controlled amount of liquid into a target vessel. In *Bimanual Handover*, the left hand grasps an object and transfers it to the right hand. In *Bimanual Lifting*, both arms grasp a large container and lift it cooperatively.

Baselines. We compare against five controlled baselines that isolate the visual encoder or action model. All methods use the same RL expert demonstrations and are evaluated under the same protocol. DP + ResNet uses Diffusion Policy (DP) (Chi et al., 2023) with a ResNet encoder trained from scratch on RGB images, without geometric supervision. DP + R3M uses a frozen R3M (Nair et al., 2023) encoder pretrained on Ego4D (Grauman et al., 2022) videos. DP + DINOv2 uses a frozen ViT-S DINOv2 (Oquab et al., 2024) encoder without depth or normal fine-tuning. BC + Geo uses deterministic MLP behavior cloning (BC) conditioned on the frozen geometric encoder. ACT + Geo uses ACT (Zhao et al., 2023) with the same encoder. ACT models action chunks with a conditional variational autoencoder, providing a chunking-based alternative to diffusion. The 2×2 analysis in §4.3 evaluates one further combination, BC + DINOv2, under the same protocol.

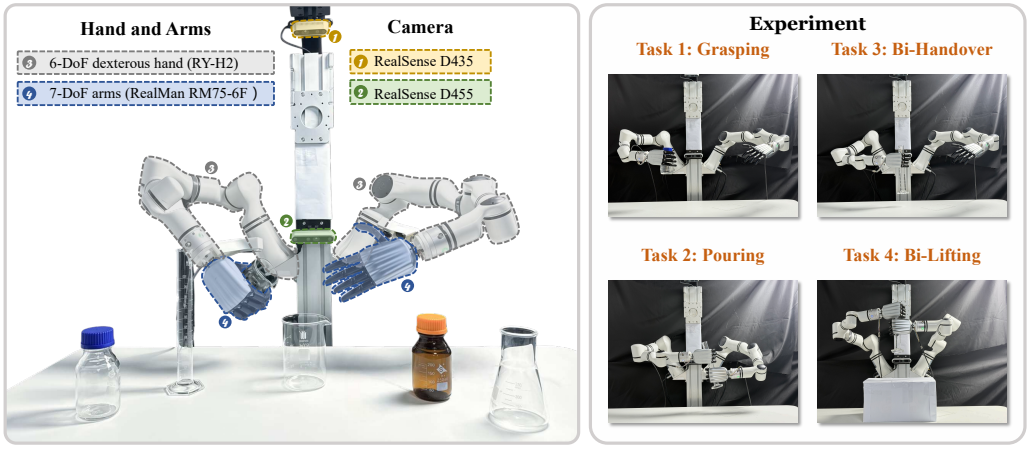


Figure 2: Hardware setup. Two seven-degree-of-freedom arms (RealMan RM75-6F) carry six-degree-of-freedom hands (RY-H2). Head-mounted RealSense D435 and chest-mounted RealSense D455 cameras provide 640×480 RGB views. Their depth streams are not used.

Table 2: Component ablations and encoder-policy interaction. Values are real-world success rates (%) from 20 trials per task. In (a,b), each variant changes one component of GEARS; Avg. is the unweighted task mean. In (c), each encoder-policy combination aggregates 80 trials; the final row ($\Delta_{\text{enc.}}$: Geometry − DINOv2) and column ($\Delta_{\text{pol.}}$: DP − BC) give percentage-point differences, with the bottom-right entry reporting the difference-in-differences. BC and DP denote behavior cloning and diffusion policy. "w/o," Cross-attn, AdaLN, ADR, and DR denote without, cross-attention, adaptive layer normalization, automatic domain randomization, and domain randomization.

(a) Geometric encoder

Variant	Gr.	Po.	Ha.	Li.	Avg.
GEARS (full)	**90.0**	**65.0**	**55.0**	**60.0**	**67.5**
w/o Normal	80.0	50.0	40.0	50.0	55.0
w/o Depth	70.0	40.0	30.0	35.0	43.8
w/o $\mathcal{L}_{\text{dn}}$	85.0	55.0	45.0	50.0	58.8
w/o Fine-tuning	55.0	25.0	15.0	25.0	30.0

(b) Policy & domain randomization

Variant	Gr.	Po.	Ha.	Li.	Avg.
GEARS (full)	**90.0**	**65.0**	**55.0**	**60.0**	**67.5**
w/o Cross-attn	75.0	45.0	35.0	40.0	48.8
w/o AdaLN	80.0	55.0	40.0	50.0	56.3
Uniform ADR	80.0	50.0	40.0	45.0	53.8
No DR	60.0	25.0	15.0	20.0	30.0

(c) Encoder-policy interaction

Encoder	BC	DP	$\Delta_{\text{pol.}}$
DINOv2	11.3	23.8	+12.5
Geometry	22.5	**67.5**	+45.0
$\Delta_{\text{enc.}}$	+11.2	+43.7	**+32.5**

Evaluation. Each method is evaluated in 20 real-world trials per task, with object poses sampled from the corresponding simulation range. Appendix B.4 defines the success criteria and time limits. All diffusion variants use the same DiT and training configuration. The action-model baselines use the same frozen geometric encoder. Here, zero-shot means that no method receives real-world policy training or policy adaptation. All methods share the camera alignment and joint calibration described in Appendix B. This design isolates component choices within a fixed-platform setting. Methods that use additional real-world policy-training data or different embodiments are not directly comparable.

4.2 Main Results

Table 1 reports zero-shot real-world success. The GEARS row contains the highest recorded per-task rates for the full system. GEARS succeeds in 54 of 80 trials, giving 67.5% overall. ACT + Geo is the strongest controlled baseline, with 34 successes in 80 trials and 42.5% overall. The gap is 25.0 percentage points. It appears on every task: 20 points on Grasping and Lifting, and 30 points on Pouring and Handover. The 95% Wilson intervals are 56.6–76.8% for GEARS and 32.3–53.4% for ACT + Geo. A two-sided Fisher exact test on the aggregate counts gives $p = 0.0024$. This test summarizes the recorded counts but does not account for how the highest per-task rates were selected.

The upper block fixes the action model to DP and varies the visual encoder. The ResNet trained from scratch gives 7.5%, while the pretrained R3M and DINOv2 encoders reach 16.3% and 23.8%.

These rows show that visual pretraining helps, but they do not isolate geometry supervision because the encoder architectures differ. The closest control is "w/o Fine-tuning" in Table 2a, which retains the complete geometric-encoder architecture and reaches 30.0%. Appendix C.1 states the remaining ambiguity in the recorded protocol for this control.

The lower block fixes the geometric encoder and changes the action model. Deterministic BC reaches 22.5%, ACT reaches 42.5%, and diffusion reaches 67.5%. The diffusion–ACT gap is 20 points on Grasping and Lifting, but 30 points on Pouring and Handover. These two tasks depend strongly on contact state and release timing, which are hidden from the student. Their larger gaps agree with the marginalized action target in Eq. (4), but the success rates alone do not establish multimodality. Each task rate is based on 20 trials and should be read as descriptive. Appendix A provides additional distributional diagnostics and uncertainty estimates.

4.3 Ablation Studies

Table 2 jointly summarizes the component ablations and encoder-policy interaction. Panels 2a and 2b evaluate the geometric encoder and the policy and domain-randomization components, respectively; panel 2c evaluates their interaction.

Geometric encoder (Table 2a). Removing each geometric objective reduces the observed average success rate. Removing normal supervision lowers success by 12.5 points, while removing depth supervision lowers it by 23.7 points. Depth is the stronger supervised signal in this ablation and supplies spatial-localization cues; normals describe surface orientation at contact. Removing only the depth-normal consistency term lowers success by 8.7 points, which supports coupling the two prediction tasks. The largest change occurs without geometry fine-tuning, where success falls from 67.5% to 30.0%. This controlled variant shows that the pretrained backbone alone does not explain the full-system result.

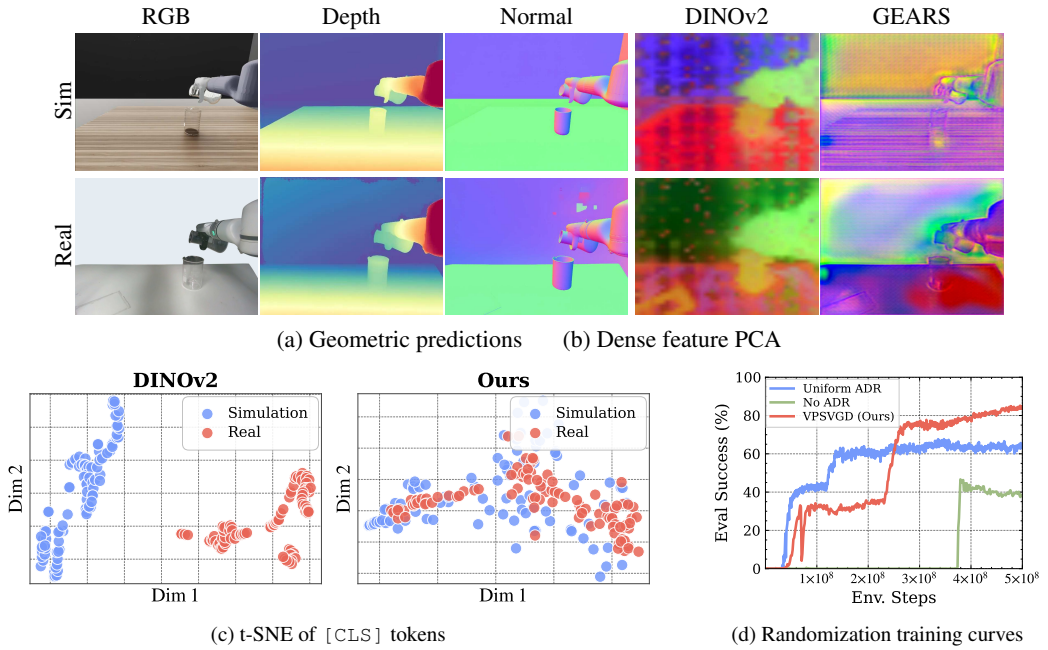
Policy and domain randomization (Table 2b). The two conditioning pathways contribute at different spatial scales. Removing cross-attention, which connects the denoiser to dense visual and proprioceptive tokens, lowers success by 18.7 points. Removing global AdaLN conditioning lowers success by 11.2 points, with the largest task-level change on Handover. These controlled variants support retaining both local and global conditioning. Success is 30.0% without domain randomization, 53.8% with uniform ADR, and 67.5% with VP-SVGD. Uniform ADR and VP-SVGD both sample uniformly inside axis-aligned bounds, but only VP-SVGD updates those bounds from an estimated failure landscape. The observed difference between these two variants is 13.7 points. Because each variant is represented by one trained policy, this comparison does not measure across-seed variability.

Encoder-policy coupling (Table 2c). Table 2c makes the encoder-policy interaction explicit. With DINOv2 features, replacing BC with diffusion adds 12.5 points. With the geometry-supervised encoder, the same replacement adds 45.0 points. Conversely, geometry adds 11.2 points under BC and 43.7 points under diffusion. The resulting difference-in-differences is $+32.5$ points. This is a descriptive interaction contrast from one trained policy per combination, not an estimate of an average interaction across training seeds.

4.4 Analysis

Cross-domain alignment of geometric features. Fig. 3(a,b) compares features for one simulation-real image pair. Predicted depth and normals preserve similar object and support geometry despite large RGB differences. The dense principal component analysis (PCA) colors are also more similar across domains for our features than for DINOv2. Fig. 3(c) projects 100 simulation and 100 real `[CLS]` tokens per encoder with t-distributed stochastic neighbor embedding (t-SNE). The domains appear more interleaved for our encoder than for DINOv2. Panels (b) and (c) are qualitative and do not measure domain distance.

Training progress. In Fig. 3(d), the VP-SVGD run reaches its final success level with fewer environment steps than the uniform-ADR run. This single-run comparison shows the training trajectory used in our experiments. It does not establish average sample efficiency or wall-clock efficiency. Appendix B.3 reports the available cost information.



(a) Geometric predictions (b) Dense feature PCA

(c) t-SNE of [CLS] tokens (d) Randomization training curves

Figure 3: Cross-domain representation analysis. The rows in (a) and (b) show one simulation image and one real image. (a) RGB observations with monocular depth and surface normal predictions from the geometry-supervised encoder. (b) The first three principal components of dense DINOv2 and GEARS features, mapped to RGB color. (c) t-distributed stochastic neighbor embedding projections of [CLS] tokens from 100 simulation and 100 real images per encoder; colors denote domains. Panels (b) and (c) are qualitative and do not measure domain distance. (d) Randomization training curves. Evaluation success is plotted against environment steps for one run of Virtual Particle Stein Variational Gradient Descent (VP-SVGD), uniform automatic domain randomization (ADR), and no domain randomization.

Real-world qualitative results. Appendix A.5 shows successful execution sequences for all four tasks. The frames illustrate transparent-object grasping, controlled pouring, handover, and coordinated lifting. They complement the aggregate success rates but do not isolate individual components.

Failure analysis. The remaining failures point to two concrete bottlenecks. Extreme viewpoints can produce grasp-aperture errors, suggesting insufficient visual coverage near the edge of the randomized viewpoint range. During handover, the left hand can release before the right grasp is secure, indicating that coordination remains implicit in the shared action chunk. Appendix A.4 reports exploratory interventions for both cases; these interventions are separate from the primary results in Table 1.

5 Conclusion

We presented GEARS, a three-stage framework for zero-shot sim-to-real dexterous manipulation. Inverse-depth and surface-normal supervision produces a deployable geometric RGB encoder, VP-SVGD adapts expert-training bounds from estimated failures, and a diffusion student represents action variation after privileged state and randomized physics are marginalized. On a fixed dual-arm platform, GEARS achieves a 67.5% real-world success rate without real-world policy training, substantially outperforming the strongest controlled baseline at 42.5%, with gains on all four tasks. The ablations further support the complementary roles of geometry-supervised representations and diffusion-based action modeling. Deployment requires only RGB observations and proprioception. Evaluation on one dual-arm embodiment and a controlled laboratory setup provides a clear basis for broader studies. Future work will assess additional embodiments and training runs, incorporate contact sensing or latent-state inference to reduce action ambiguity, and extend GEARS to longer-horizon manipulation.

AI USE STATEMENT

Generative AI tools, including large language models, were used to assist with manuscript polishing, code debugging, and data analysis. Writing assistance focused on improving grammar, clarity, and readability. Technical assistance supported the identification and resolution of programming errors and the analysis of experimental results. The authors retain full responsibility for the research and the final manuscript, including the correctness of the code, the validity of the analyses, the accuracy of the reported results, and all text, claims, and artifacts prepared with AI assistance.

REFERENCES

Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving Rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. 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. URL https://www.roboticsproceedings.org/rss21/p010.html.

Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *ICRA*, pp. 8973–8979, 2019.

Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by cheating. In *CoRL*, pp. 66–75, 2020.

Jialong Cheng, Tao Liu, Yuyang Li, Jianlan Yang, Jiangmiao Li, Haichao Zhang, and Jiangping Pang. VIRAL: Visual sim-to-real at scale for humanoid loco-manipulation. *arXiv preprint arXiv:2511.15200*, 2025.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *RSS*, 2023.

David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *NeurIPS*, 2014.

Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *CoRL*, pp. 158–168, 2022.

Dibya Ghosh, Homer Rich Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Quan Vuong, Ted Xiao, Pannag R Sanketi, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. 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. URL https://www.roboticsproceedings.org/rss20/p090.html.

Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, Miguel Martin, Tushar Nagarajan, Ilija Radosavovic, Santhosh Kumar Ramakrishnan, Fiona Ryan, Jayant Sharma, Michael Wray, Mengmeng Xu, Eric Zhongcong Xu, Chen Zhao, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *CVPR*, pp. 18995–19012, 2022.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, pp. 1861–1870, 2018.

Ankur Handa, Arthur Allshire, Viktor Makoviychuk, Alexey Petrenko, Ritvik Singh, Jingzhou Liu, Denys Makoviichuk, Karl Van Wyk, Alexander Zhurkevich, Balakumar Sundaralingam, Yashraj Narang, Jean-Francois Lafleche, Dieter Fox, and Gavriel State. DeXtreme: Transfer of agile in-hand manipulation from simulation to reality. In *ICRA*, pp. 5977–5984, 2023.

Nikolaus Hansen, Sibylle D. Müller, and Petros Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary Computation*, 11(1):1–18, 2003.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In *CVPR*, pp. 12627–12637, 2019.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard (eds.), *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713. PMLR, 06–09 Nov 2025. URL https://proceedings.mlr.press/v270/kim25c.html.

Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. RMA: Rapid motor adaptation for legged robots. In *RSS*, 2021.

Seungjae Lee, Yibin Wang, Haritheja Etukuru, H. Jin Kim, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Behavior generation with latent actions. In *ICML*, 2024.

Toru Lin, Kartik Sachdev, Linxi Fan, Jitendra Malik, and Yuke Zhu. Sim-to-real reinforcement learning for vision-based dexterous manipulation on humanoids. In *CoRL*, 2025.

Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. In *NeurIPS*, 2016.

Tyler Ga Wei Lum, Arthur Allshire, Viktor Makoviychuk, Ankur Handa, and Yashraj Narang. DextrAH-RGB: Visuomotor policies to grasp anything with dexterous hands. *arXiv preprint arXiv:2412.01791*, 2024a.

Tyler Ga Wei Lum, Viktor Makoviychuk, Ankur Handa, Arthur Allshire, and Yashraj Narang. DextrAH-G: Pixels-to-action dexterous arm-hand grasping with geometric fabrics. In *CoRL*, 2024b.

Arjun Majumdar, Karmesh Yadav, Sergio Arnaud, Yecheng Jason Ma, Claire Chen, Sneha Silwal, Arjun Jain, Vincent-Pierre Berber, Alexander Sax, Aravind Rajeswaran, Dhruv Batra, Manolis Savva, Irfan Essa, and Jitendra Malik. Where are we in the search for an artificial visual cortex for embodied intelligence? In *NeurIPS*, 2023.

Mayank Mittal, Pascal Roth, James Tigue, Antoine Richard, Octi Zhang, Peter Du, Antonio Serrano-Muñoz, Xinjie Yao, René Zurbrügg, Nikita Rudin, et al. Isaac lab: A gpu-accelerated simulation framework for multi-modal robot learning. *arXiv preprint arXiv:2511.04831*, 2025.

Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3M: A universal visual representation for robot manipulation. In *CoRL*, pp. 892–909, 2023.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *TMLR*, 2024.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, 2023.

Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *ICRA*, pp. 3803–3810, 2018.

M Nomaan Qureshi, Sparsh Garg, Francisco Yandun, David Held, George Kantor, and Abhisesh Silwal. Splatsim: Zero-shot sim2real transfer of rgb manipulation policies using gaussian splatting. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6502–6509. IEEE, 2025.

Ilija Radosavovic, Tete Xiao, Stephen James, Pieter Abbeel, Jitendra Malik, and Trevor Darrell. Real-world robot learning with masked visual pre-training. In *CoRL*, pp. 416–426, 2023.

Fabio Ramos, Rafael Possas, and Dieter Fox. BayesSim: Adaptive domain randomization via probabilistic inference for robotics simulators. In *RSS*, 2019.

René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *ICCV*, pp. 12179–12188, 2021.

René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(3):1623–1637, 2022. doi: 10.1109/TPAMI.2020.3019967.

Shreeyak Sajjan, Matthew Moore, Mike Pan, Ganesh Nagaraja, Johnny Lee, Andy Zeng, and Shuran Song. Clear grasp: 3D shape estimation of transparent objects for manipulation. In *ICRA*, pp. 3634–3642, 2020.

Jinghuan Shang, Karl Schmeckpeper, Brandon B. May, Maria Vittoria Minniti, Tarik Kelestemur, David Watkins, and Laura Herlant. Theia: Distilling diverse vision foundation models for robot learning. In *CoRL*, 2024.

Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Hervé Jégou, Patrick Labatut, and Piotr Bojanowski. DINOv3, 2025. URL https://arxiv.org/abs/2508.10104.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2021.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IROS*, pp. 23–30, 2017.

Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything V2. In *NeurIPS*, 2024.

Yash Yardi, Samuel Biruduganti, and Lars Ankile. Bridging the sim2real gap: Vision encoder pretraining for visuomotor policy transfer. *arXiv preprint arXiv:2501.16389*, 2025.

Wei Yin, Chi Zhang, Hao Chen, Zhipeng Cai, Gang Yu, Kaixuan Wang, Xiaozhi Chen, and Chunhua Shen. Metric3D: Towards zero-shot metric 3D prediction from a single image. In *ICCV*, pp. 9043–9053, 2023.

Yanjie Ze, Gu Yan, Yuyang Wu, Anirudha Macaluso, Yuying Ge, Jianglong Ye, Nicklas Hansen, Li Erran Li, and Xiaolong Wang. 3D diffusion policy: Generalizable visuomotor policy learning via simple 3D representations. In *RSS*, 2024.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *RSS*, 2023.

Haoyi Zhu, Honghui Peng, Junwei Wu, Siming Xiao, Linggang Yu, Xiaojian Wei, Jie Han, and Lin Ma. SPA: 3D spatial-awareness enables effective embodied representation. *arXiv preprint arXiv:2410.08208*, 2024.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, Michael S. Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J. Joshi, Alex Irpan, Brian Ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. RT-2: Vision-language-action models transfer web knowledge to robotic control. In Jie Tan, Marc Toussaint, and Kourosh Darvish (eds.), *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183. PMLR, 06–09 Nov 2023. URL https://proceedings.mlr.press/v229/zitkovich23a.html.

Supplementary Material

A Additional Experimental Results **15**

A.1 Action Distributions under Hidden Simulator Variables 15

A.2 DINOv3 Visual-Backbone Control 15

A.3 Within-Platform Stress Tests 16

A.4 Failure-Mode Interventions 16

A.5 Real-World Execution Sequences 16

B System Design **16**

B.1 Camera Alignment 16

B.2 Action Space and Control 18

B.3 Deployment and Compute Reporting 18

B.4 Real-World Evaluation Protocol 18

B.5 Robot Joint Calibration 18

B.6 Task Definitions and Reward Functions 19

C Implementation Details **23**

C.1 Geometric Encoder Training 23

C.2 Adaptive Domain Randomization 25

C.3 Massively Parallel SAC Training 27

C.4 Diffusion Student Training 27

A Additional Experimental Results

This section supplements the primary evaluation with an action-distribution visualization, a DINOv3 backbone control, task-level confidence intervals, within-platform stress tests, failure-mode interventions, and successful real-robot execution sequences. The action projection, DINOv3 control, stress tests, and interventions are exploratory and do not change the primary comparison.

A.1 Action Distributions under Hidden Simulator Variables

Equation (4) of the main paper defines the student action distribution after marginalizing unobserved simulator states and randomized parameters. Figure 4 projects logged expert chunks, diffusion samples, and a behavior-cloning (BC) prediction into the same two-dimensional principal-component space. The BC prediction falls between visually separated regions occupied by expert chunks, whereas the diffusion samples overlap several of those regions. This qualitative contrast is consistent with the conditional-mean objective of mean-squared-error BC and the distributional objective of diffusion. The projection does not identify action modes or estimate their probability mass.

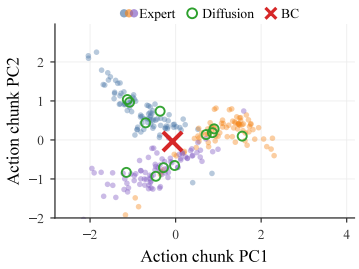


Figure 4: Qualitative projection of action chunks into a shared two-dimensional principal-component space. Filled points are logged expert chunks, green circles are diffusion samples, and the red cross is the behavior-cloning prediction trained with mean squared error. Expert colors are visual aids rather than ground-truth mode labels; the projection is qualitative and does not identify modes or estimate their probability mass.

A.2 DINOv3 Visual-Backbone Control

Table 3 combines the visual-backbone control and task-level confidence intervals. Table 3a reports the DINOv3 (Siméoni et al., 2025) control. Relative to DP + DINOv2, this control replaces only the visual backbone; the expert demonstrations, Diffusion Transformer, optimization, and 80-trial real-world protocol remain fixed. DINOv3 succeeds in 25 of 80 trials (31.3%), compared with 19 of 80 trials (23.8%) for DINOv2 and 54 of 80 trials (67.5%) for GEARS. Because DINOv2 and DINOv3 differ in architecture and pretraining, this comparison does not isolate a single source of the six additional successes. It shows that replacing DINOv2 with DINOv3 alone does not recover the GEARS result. Table 3b reports confidence intervals for the primary task-level results.

Table 3: Additional real-world results and confidence intervals. (a) Controlled visual-backbone comparison over 80 trials. The DINOv2 and DINOv3 controlled systems differ only in the visual backbone; GEARS is the geometry-supervised reference, and DP denotes Diffusion Policy. (b) Task-level results for GEARS with 20 trials per task and 80 trials in total. All intervals are two-sided 95% Wilson confidence intervals (CIs).

(a) Visual-backbone control

Method	**Success**	**Rate (%)**	**95% CI**
DP + DINOv2	19/80	23.8	15.8–34.1
DP + DINOv3	25/80	31.3	22.2–42.1
GEARS	54/80	**67.5**	56.6–76.8

(b) GEARS task results

Task	**Success**	**Rate (%)**	**95% CI**
Grasping	18/20	90.0	69.9–97.2
Pouring	13/20	65.0	43.3–81.9
Bimanual Handover	11/20	55.0	34.2–74.2
Bimanual Lifting	12/20	60.0	38.7–78.1

A.3 Within-Platform Stress Tests

Table 4 combines the within-platform stress tests and targeted interventions. Table 4a reports tests with unseen object shapes, scene clutter, lighting perturbations, and a longer pouring sequence. Every test uses the same GEARS checkpoint, success criteria, and robot calibration as the primary evaluation, without real-world fine-tuning, policy updates, or additional system identification. Success ranges from 50.0% to 60.0% across these shifts, compared with 67.5% in the primary evaluation. These tests measure within-platform robustness and do not evaluate cross-embodiment transfer.

A.4 Failure-Mode Interventions

We evaluate two exploratory interventions motivated by the failure modes reported in the main paper. Adding extreme-viewpoint renders increases Grasping success from 9/20 to 12/20 trials in the extreme-pose setting. For Handover, a proprioceptive release gate delays the left-hand opening command until the right-hand grasp is stable and increases success from 11/20 to 13/20 trials. Neither intervention uses tactile sensing or real-world policy training. Table 4b reports the trial counts and Wilson intervals.

Table 4: Within-platform stress tests and failure-mode interventions. (a) Same-checkpoint tests under changes in object shape, clutter, lighting, and pouring duration. All rows use the primary GEARS checkpoint, success criteria, and robot calibration, without real-world fine-tuning, policy updates, or additional system identification. (b) Additional viewpoint renders for Grasping and a proprioceptive release gate for Handover; neither intervention uses tactile sensing or real-world policy training. Each stress or intervention row contains 20 trials, while the main evaluation aggregates 80 trials. All intervals are two-sided 95% Wilson confidence intervals (CIs).

(a) Stress tests

Setting	**Success**	**Rate (%)**	**95% CI**
Main evaluation	54/80	67.5	56.6–76.8
Unseen shapes	12/20	60.0	38.7–78.1
Clutter	11/20	55.0	34.2–74.2
Lighting	12/20	60.0	38.7–78.1
Longer Pouring sequence	10/20	50.0	29.9–70.1

(b) Targeted interventions

Setting	**Success**	**95% CI**
Extreme-pose Grasping, original	9/20	25.8–65.8
View-augmented Grasping	12/20	38.7–78.1
Handover, original	11/20	34.2–74.2
Handover, release gate	13/20	43.3–81.9

A.5 Real-World Execution Sequences

Figure 5 shows one successful real-robot execution of each evaluated task. In Grasping, the robot approaches, secures, and lifts a transparent beaker. In Pouring, it grasps the source vessel, moves it above the target, pours, and returns it upright. During Handover, the left hand presents the object, the right hand establishes a stable grasp, and the left hand releases. During Bimanual Lifting, both hands grasp and raise the same container while maintaining a coordinated pose. These sequences are qualitative examples; Table 1 of the main paper reports the aggregate results over all 80 trials.

B System Design

Zero-shot deployment requires consistent observation and action interfaces between simulation and the physical robot. This section describes camera alignment, action and control interfaces, deployed components, the evaluation protocol, and joint calibration for the dual-arm platform.

B.1 Camera Alignment

The policy receives 640×480 RGB streams from a head-mounted Intel RealSense D435 and a chest-mounted D455. The cameras use different color sensors and fields of view, so each is cali-

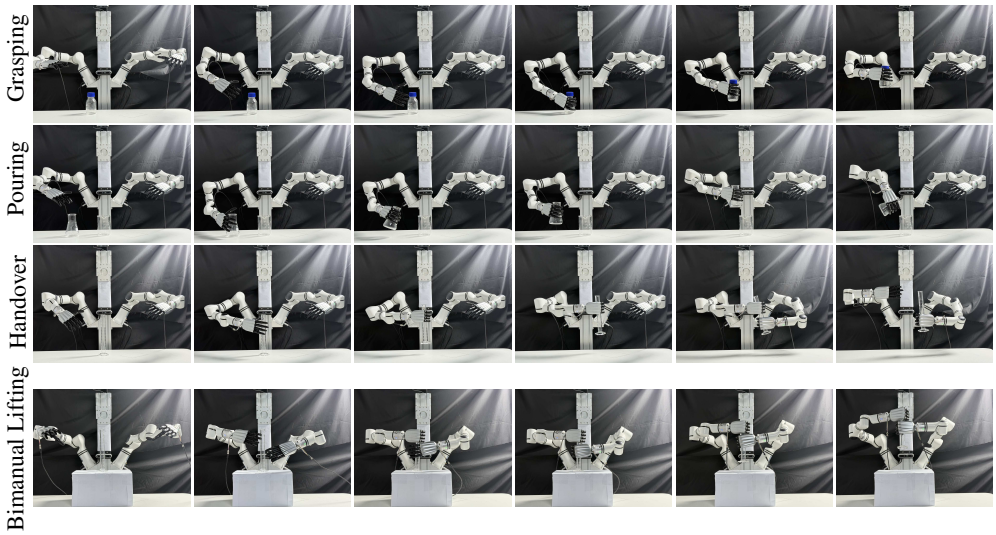


Figure 5: Selected successful real-world execution sequences. Each row shows six time-ordered RGB frames for one task on the fixed dual-arm platform. These sequences are qualitative examples and are not additional evaluation trials. Aggregate results over all 80 trials appear in Table 1 of the main paper.

brated independently. For each stream, we read the factory intrinsics (f_x, f_y, c_x, c_y) at the deployed resolution and configure the corresponding Isaac Lab pinhole projection. Using per-stream intrinsics preserves the different projection geometry of the two cameras.

The deployed streams use a 4:3 aspect ratio, whereas the native D435 RGB frame is 16:9. Figure 6 shows the aspect-ratio mismatch considered during calibration. The schematic represents the deployed D435 frame as a horizontal crop of the native frame; retaining the native horizontal aperture in a 4:3 Isaac Sim render instead expands the vertical extent. This explains why native-resolution projection parameters cannot be transferred directly. Numerical calibration uses the intrinsics reported at 640×480 for each camera.

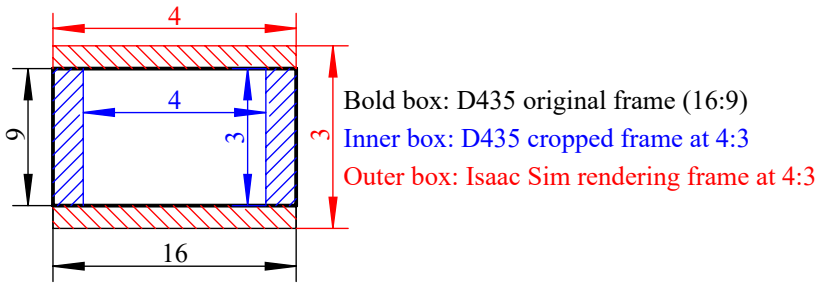


Figure 6: Aspect-ratio handling during camera calibration. The black rectangle denotes the native 16:9 D435 frame, the blue rectangle represents its horizontally cropped 4:3 output, and the red rectangle represents the 4:3 Isaac Sim rendering extent obtained when the native horizontal aperture is retained. The simulated D435 and D455 projections are instead configured independently from the factory intrinsics reported at the deployed 640×480 resolution.

The chest-camera extrinsics are initialized from mechanical measurements. For the adjustable head camera, we place the robot in a known joint configuration and manually adjust the simulated head-camera pitch until the rendered arm projection matches the recorded image. Camera-pose perturbations in Table 9 cover small residual extrinsic errors. No depth stream from either RealSense camera is used by the policy.

B.2 Action Space and Control

Each arm has seven actuated joints controlled by velocity targets, and each RY-H2 hand has six actuated degrees of freedom controlled by position targets. The full action has 26 channels: 14 arm-velocity targets and 12 hand-position targets. The RY-H2 contains additional mechanically coupled passive joints, so the six hand commands refer specifically to actuated degrees of freedom.

The proprioceptive input contains the joint-state signals exposed by the arm and hand controllers in both simulation and reality. Demonstration collection, student training, and deployment use the same channel ordering, which is specified in the accompanying code archive.

The RL expert and diffusion student operate at 30 Hz. The simulator holds each target between decision steps, while the hardware controllers track targets at their native servo rate. The diffusion student predicts eight actions, executes four, and then replans from a new observation, yielding a 7.5 Hz replanning rate. Because the action mixes units, normalization is channel specific. Arm-velocity targets remain velocities and are standardized with their training-set statistics. Hand-position targets are first expressed as offsets from the current hand configuration and then standardized. At deployment, the stored statistics invert these transforms before commands are sent to the two controllers.

B.3 Deployment and Compute Reporting

Deployment retains the two RGB streams, proprioception, the frozen geometric encoder, and the diffusion student. The privileged reinforcement learning (RL) expert, Virtual Particle Stein Variational Gradient Descent (VP-SVGD) wrapper, simulator, and depth and normal prediction heads are used only during training.

Tables 7, 11, and 12 report the architecture and optimization settings for the three training stages. The encoder and reinforcement-learning tables also report the recorded hardware configuration. Wall-clock time, graphics processing unit (GPU) hours, peak memory, and end-to-end inference latency were not logged, so we do not report compute-efficiency comparisons. Figure 3 of the main paper compares environment-step progress for one run per method.

B.4 Real-World Evaluation Protocol

Each method is evaluated for 20 trials per task. Object poses are sampled from the predefined simulation range. Each trial ends at success, the task time limit, or an unrecoverable drop. The task-level rubric below matches the simulation success conditions, and the same rubric is applied to every method.

Grasping succeeds when the object rises by more than 0.15 m and remains held for 0.5 s. Pouring succeeds when 60–90% of the initial liquid reaches the target vessel, the source returns to within 30° of upright, and both conditions hold for 0.5 s. Handover succeeds when the right hand retains the object, the left fingertips no longer contact it, and the object speed remains below 0.05 m/s for 1 s. Bimanual Lifting succeeds when the object rises by more than 0.15 m, both hands retain it, and the left and right fingertip-centroid heights remain within 0.03 m for 1 s. The time limits are 10 s for Grasping and 15 s for the other tasks.

B.5 Robot Joint Calibration

The manufacturer does not expose the low-level proportional-derivative (PD) gains and motor dynamics of the RM75-6F arms and RY-H2 hands. We therefore estimate the simulated actuator parameters from trajectories rather than configuring them from datasheets. Reference trajectories are commanded on the physical robot, the resulting joint positions are recorded at 30 Hz, and the simulator parameters are fitted to reproduce these responses.

We design excitation trajectories that cover the operating range of each joint, including step commands, sinusoidal sweeps at varying frequencies, and segments of the actual manipulation motions. For each trajectory, we log the commanded target $\mathbf{q}_t^{\text{cmd}}$ and the measured joint position $\mathbf{q}_t^{\text{real}}$. In Isaac Lab, the same command sequence is replayed while the joint-level PD stiffness k_p, damping k_d, and friction parameters are treated as free variables. We minimize the tracking error

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_t \left\| \mathbf{q}_t^{\text{sim}}(\boldsymbol{\theta}) - \mathbf{q}_t^{\text{real}} \right\|^2, \tag{6}$$

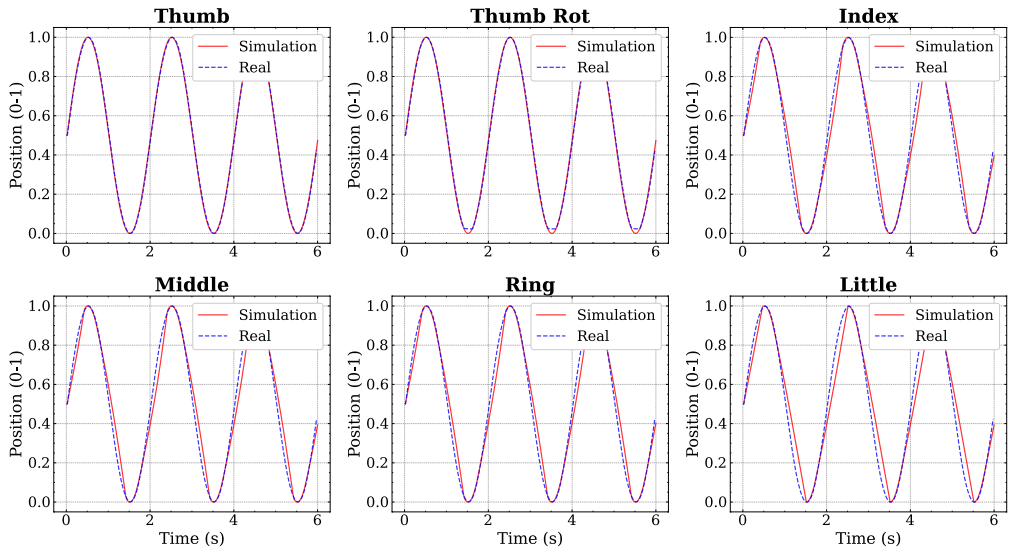


Figure 7: Joint tracking comparison for the real and simulated RY-H2 hand. Each subplot shows one of six actuated degrees of freedom under the same position target. Each real–simulation pair uses one shared normalization to $[0, 1]$ because the joint ranges differ. Overlap measures tracking agreement for these commands and does not evaluate contact dynamics.

where $\boldsymbol{\theta} = \{k_p, k_d, \mu\}$ collects the per-joint stiffness, damping, and friction coefficients, and $\mathbf{q}_t^{\text{sim}}(\boldsymbol{\theta})$ is the simulated joint position under those parameters. We solve Eq. (6) with the derivative-free Covariance Matrix Adaptation Evolution Strategy (CMA-ES) (Hansen et al., 2003).

Figure 7 compares the six actuated degrees of freedom of the RY-H2 hand under identical position targets. For visualization, each real–simulation pair is normalized with one shared per-joint affine transform to $[0, 1]$; the real and simulated curves are not normalized independently. Their close agreement shows that the fitted simulator tracks the tested free-space commands. This comparison does not evaluate contact dynamics. The fitted parameters and calibration script are included in the accompanying code archive.

B.6 Task Definitions and Reward Functions

This subsection defines the four tasks, simulation success criteria, and expert reward functions. Each reward uses phase gates that activate later terms after prerequisite conditions are met. The gates encode the intended task order but do not guarantee a particular learned strategy. Table 5 summarizes the structure, and Table 6 lists the coefficients.

Notation and Reward Structure

We adopt the following notation throughout this subsection. $\mathbf{p}_i^{\text{ft}} \in \mathbb{R}^3$ denotes the position of the i-th fingertip. Indices $i \in \{1, 2, 3\}$ denote the thumb, index, and middle fingers, and $N_f = 3$. The fingertip centroid is $\bar{\mathbf{p}}^{\text{ft}} = \frac{1}{N_f} \sum_i \mathbf{p}_i^{\text{ft}}$. The vector $\mathbf{p}_{\text{obj}}$ denotes the object centroid, and $\mathbf{p}_{\text{gs}}$ denotes the designated grasp site, the geometric center of the grasp region on the object surface. Finally, $\mathbf{q}_t$ is the joint position vector and $\mathbf{a}_t$ is the action at step t. Superscripts L and R distinguish the left and right arms. For bimanual tasks, each arm has its own reward terms. We omit the superscript when the definition is symmetric.

The reward at each step is

$$r_t \;=\; \sum_k w_k \, r_k(\mathbf{s}_t, \mathbf{a}_t) \cdot g_k(\mathbf{s}_t)\,, \tag{7}$$

where $\mathbf{s}_t$ is the simulator task state, r_k is a reward component, w_k is its weight, and $g_k \in \{0, 1\}$ is a phase-dependent binary gate. Progress gates remain active after their prerequisites are met so that later phases preserve earlier task requirements. The only exception is the Handover release transition, which closes the left-grasp gates when the right grasp becomes stable. Regularization

Table 5: Reward structure per task. P1 reaches the object, P2 establishes a grasp, P3 lifts or performs task setup, and P4 completes transfer or pouring. Each cell lists the terms that first activate in that phase; progress terms normally persist across later phases (Eq. (7)). During Handover P4, the left-grasp and hold terms deactivate when the release term activates. Superscripts L and R denote the left and right hands, and LR denotes terms applied to both hands. Subscripts name the rewarded quantity, such as reach, contact, lift, tilt, volume, release, or symmetry. Bold marks terms first introduced by that task. In the success column, Δh is the object height gain, "stable" requires all three task-relevant fingertips to contact the object, "released" requires no such left-fingertip contact, and "sym." requires the fingertip-centroid height difference to remain below 0.03 m. Trailing durations are the required hold times, and velocities are in m/s.

Task	P1	P2	P3	P4	Success condition
Grasping	r_{reach}	r_{finger}, r_{contact}, r_{stable}	$\mathbf{r}_{\text{lift}}$, $\mathbf{r}_{\text{orn}}$	No P4	$\Delta h > h_{\text{target}}$, grasp stable, 0.5 s
Pouring	r_{reach}	r_{finger}, r_{contact}, r_{stable}	r_{lift}, r_{orn}, $\mathbf{r}_{\text{trans}}$	$\mathbf{r}_{\text{tilt}}$, $\mathbf{r}_{\text{vol}}$, $\mathbf{r}_{\text{spill}}$	$\rho \in [\rho_{\min}, \rho_{\max}]$, upright, 0.5 s
Handover	r^L_{reach}, r^L_{finger}, r^L_{contact}, r^L_{stable}	$\mathbf{r}_{\text{present}}$	r^R_{reach}, r^R_{finger}, r^R_{contact}, r^R_{stable}, $\mathbf{r}^L_{\text{hold}}$	$\mathbf{r}^L_{\text{release}}$	R stable, L released, $\|\mathbf{v}_{\text{obj}}\| < 0.05$, 1.0 s
Lifting	r^{LR}_{reach}	r^{LR}_{finger}, r^{LR}_{contact}, r^{LR}_{stable}, $\mathbf{r}_{\text{sync}}$	r_{lift}, r_{orn}, $\mathbf{r}_{\text{sym}}$	No P4	$\Delta h > h_{\text{target}}$, both stable, sym., 1.0 s

terms remain active throughout each episode. All distance-based terms use the ℓ_2 norm unless stated otherwise. Equation (7) is instantiated below with task-specific components and gates.

Table 6: Reported reward weights and numerical constants. The left block lists shared and task-specific reward weights. The right block lists task thresholds and kernel sharpness values. Sharpness values act on distances in meters or angles in radians; λ_h scales a dimensionless height ratio. Task-geometry constants, including the pouring angle and handover pose, are stored in the accompanying configuration files. Shared reward weights were tuned on Grasping and reused across tasks; weights for task-specific reward terms were set separately.

Shared Weights	Value	Task Thresholds	Value
w_{reach}	1.0	h_{target} (Grasping / Bimanual Lifting)	0.15 m
w_{finger}	2.0	d_{prox} (approach $\rightarrow$ grasp)	0.05 m
w_{contact}	1.5	d_{pour} (lift $\rightarrow$ pour)	0.08 m
w_{stable}	3.0	d_{hov} (present $\rightarrow$ R-approach)	0.06 m
w_{lift}	5.0	$\rho_{\min}/\rho_{\max}$ (Pouring)	0.6 / 0.9
w_{orn}	0.5	v_{slip} (grasp stability)	0.1 m/s
w_{act}	0.01	$N_{\min}$ (min contact fingers)	3
w_{jvel}	0.005	δ_g (grasp-confirmation steps)	5
Task-specific Weights		Δt_{hold} (single-arm / bimanual)	0.5 s / 1.0 s
w_{trans} (Pouring)	2.0	$T_{\max}$ (Grasping / others)	300 / 450 steps
w_{tilt} (Pouring)	3.0	**Kernel Sharpness**	
w_{vol} (Pouring)	5.0	λ_r (reaching)	10
w_{spill} (Pouring)	2.0	λ_f (finger)	20
w_{present} (Handover)	2.0	λ_h (lift)	3
w_{hold} (Handover)	3.0	λ_t (transport)	8
w_{release} (Handover)	4.0	λ_θ (tilt)	5
w_{sync} (Bimanual Lifting)	3.0	λ_p (present)	8
w_{sym} (Bimanual Lifting)	2.0	λ_s (symmetry)	15

Shared Reward Components

Four components recur across all tasks, together with two regularizers.

Reaching. An exponential distance kernel guides the fingertips toward the grasp site:

$$r_{\text{reach}} = \exp\big(-\lambda_r \, \|\bar{\mathbf{p}}^{\text{ft}} - \mathbf{p}_{\text{gs}}\|\big) \,, \tag{8}$$

The kernel in Eq. (8) uses $\lambda_r = 10$ and is active from episode start.

Finger closure. Once the hand is within a proximity threshold $d_{\text{prox}} = 0.05\,\text{m}$ of the grasp site, a finger closure term encourages the fingertips to wrap around the object:

$$r_{\text{finger}} = \frac{1}{N_f} \sum_{i=1}^{N_f} \exp\big(-\lambda_f \, \|\mathbf{p}_i^{\text{ft}} - \mathbf{p}_{\text{gs}}\|\big) \,, \tag{9}$$

The closure term in Eq. (9) uses $\lambda_f = 20$. Its per-fingertip formulation encourages all three fingertips to enter the grasp region and yields less reward when only a subset of fingers is close to the grasp site.

Contact consistency. Let $c_i \in \{0, 1\}$ indicate whether the simulator reports contact between the collision geometry of fingertip i and the task object during the current control step. The contact reward encourages multi-finger engagement:

$$r_{\text{contact}} = \frac{1}{N_f} \sum_{i=1}^{N_f} c_i \,. \tag{10}$$

Equation (10) averages the contact indicators and therefore ranges from zero to one.

Grasp stability. We use simultaneous contact by all $N_f = 3$ task-relevant fingertips as an operational stability criterion. This criterion does not establish force closure. The reward also decreases with object motion as a heuristic for grasp stability. In the formula below, $N_{\min} = N_f = 3$:

$$r_{\text{stable}} = \mathbf{1}\Big[\sum_i c_i \geq N_{\min}\Big] \cdot \max\big(1 - \|\mathbf{v}_{\text{obj}}\|/v_{\text{slip}},\ 0\big) \,, \tag{11}$$

In Eq. (11), $v_{\text{slip}} = 0.1\,\text{m/s}$ is the velocity threshold used by the heuristic. Because the term uses absolute object velocity, it does not distinguish commanded object motion from slip relative to the fingertips.

Regularization. Two penalty terms are applied at every step. The action-magnitude term $r_{\text{act}} = -\|\mathbf{a}_t\|^2$ and joint-velocity term $r_{\text{jvel}} = -\|\dot{\mathbf{q}}_t\|^2$ penalize large commands and rapid joint motion, respectively. Their weights, 0.01 and 0.005, are listed in Table 6.

Task-Specific Reward Definitions

We now give each task's phase structure, added terms, and success test.

Grasping. The policy grasps a transparent beaker from a randomized pose and lifts it to a target height $h_{\text{target}} = 0.15\,\text{m}$ above its initial position. We denote the current and initial object heights by h_{obj} and h_{init}. The episode proceeds through three phases.

In Phase 1 (Approach), only r_{reach} is active. The episode advances to Phase 2 once $\|\bar{\mathbf{p}}^{\text{ft}} - \mathbf{p}_{\text{gs}}\| < d_{\text{prox}}$.

In Phase 2 (Grasp), r_{finger}, r_{contact}, and r_{stable} activate to close the fingers around the object. The episode advances to Phase 3 after the grasp remains stable for $\delta_g = 5$ consecutive steps.

In Phase 3 (Lift), a lift reward drives the object upward:

$$r_{\text{lift}} = \tanh\big(\lambda_h \cdot (h_{\text{obj}} - h_{\text{init}})/h_{\text{target}}\big) \,, \tag{12}$$

with $\lambda_h = 3$. The $\tanh$ approaches one near the target height, so additional height yields little additional reward. It is negative for downward displacement ($h_{\text{obj}} < h_{\text{init}}$), which penalizes pushing the object during the grasp-to-lift transition. An orientation penalty keeps the object upright:

$$r_{\text{orn}} = -\big(\theta_{\text{obj}}^{\text{roll}}\big)^2 - \big(\theta_{\text{obj}}^{\text{pitch}}\big)^2 \,, \tag{13}$$

where $\theta^{\text{roll}}_{\text{obj}}$ and $\theta^{\text{pitch}}_{\text{obj}}$ are the object's roll and pitch angles relative to upright.

The episode is successful when $h_{\text{obj}} - h_{\text{init}} > h_{\text{target}}$ and the grasp remains stable for $\Delta t_{\text{hold}} = 0.5\,\text{s}$. It terminates early if the object is dropped ($h_{\text{obj}} < h_{\text{init}} - 0.05\,\text{m}$) or the maximum length $T_{\max} = 300$ steps (10 s at 30 Hz) is reached.

Pouring. This task extends Grasping: after lifting the source beaker, the policy tilts it to pour a controlled amount of liquid into a stationary target vessel. Phases 1–3 reuse the Grasping definitions. Phase 3 adds a *transport reward* that encourages the source beaker to move above the target vessel before tilting:

$$r_{\text{trans}} = \exp\!\left(-\lambda_t \left\|\mathbf{p}^{xy}_{\text{obj}} - \mathbf{p}^{xy}_{\text{tgt}}\right\|\right), \tag{14}$$

In Eq. (14), $\lambda_t = 8$, $\mathbf{p}_{\text{tgt}}$ is the target-vessel centroid, and $\mathbf{p}^{xy}$ denotes horizontal position.

Phase 4 (Pour) activates once the source beaker is within a horizontal radius $d_{\text{pour}} = 0.08\,\text{m}$ of the target vessel. A tilt reward steers the beaker rotation:

$$r_{\text{tilt}} = \exp\!\left(-\lambda_\theta \left|\theta_{\text{src}} - \theta^*\right|\right), \tag{15}$$

where θ_{src} is the roll angle of the source beaker and $\lambda_\theta = 5$. A volume reward measures the fraction $\rho = V_{\text{poured}}/V_{\text{src}}$ of liquid that reaches the target vessel:

$$r_{\text{vol}} = \begin{cases} \rho/\rho_{\min} & \rho < \rho_{\min} \\ 1 & \rho_{\min} \le \rho \le \rho_{\max} \\ \max\!\left(1 - (\rho - \rho_{\max})/(1 - \rho_{\max}),\ 0\right) & \rho > \rho_{\max} \end{cases}, \tag{16}$$

with $\rho_{\min} = 0.6$ and $\rho_{\max} = 0.9$. The volume term in Eq. (16) increases linearly below $\rho_{\min}$, saturates within the target range, and decays above $\rho_{\max}$ to discourage over-pouring. During the tilt phase, θ^* in Eq. (15) is a task-specific fixed pouring angle. Once ρ enters $[\rho_{\min}, \rho_{\max}]$, the controller sets $\theta^* = 0$, so the same term also guides the subsequent uprighting of the beaker. The task configuration distributed with the code archive records the pouring angle. A separate term penalizes liquid that lands outside both vessels:

$$r_{\text{spill}} = -\,N_{\text{floor}} \,/\, N_{\text{total}}\,, \tag{17}$$

In Eq. (17), N_{floor} is the number of particles on the floor and N_{total} is the initial particle count.

The episode is successful when $\rho \in [\rho_{\min},\, \rho_{\max}]$ and the source beaker returns to within 30° of upright. Both conditions must hold for $\Delta t_{\text{hold}} = 0.5\,\text{s}$. The maximum episode length is $T_{\max} = 450$ steps (15 s).

Bimanual Handover. The left hand grasps an object and transfers it to the right hand. A single shared policy controls both arms simultaneously. The episode has four phases.

Phase 1 combines the first two Grasping phases for the left arm. The reward terms r^L_{reach}, r^L_{finger}, r^L_{contact}, and r^L_{stable} use the same gates as in Grasping. The phase ends when the left grasp is stable.

In Phase 2 (Present), a *present reward* guides the object to a predefined handover pose $\mathbf{p}_{\text{hov}}$ at the workspace midline:

$$r_{\text{present}} = \exp\!\left(-\lambda_p \left\|\mathbf{p}_{\text{obj}} - \mathbf{p}_{\text{hov}}\right\|\right), \tag{18}$$

The present reward in Eq. (18) uses $\lambda_p = 8$.

Phase 3 (Right approach and grasp) activates when $\|\mathbf{p}_{\text{obj}} - \mathbf{p}_{\text{hov}}\| < d_{\text{hov}} = 0.06\,\text{m}$. The right-arm reward terms r^R_{reach}, r^R_{finger}, r^R_{contact}, and r^R_{stable} activate. Meanwhile, a *hold reward* $r^L_{\text{hold}} = r^L_{\text{stable}}$ keeps the left hand engaged so that the object does not drop before the right hand secures it.

Phase 4 (Transfer) activates once the right grasp is stable. At this transition, the left-hand finger, contact, stability, and hold gates close. A *release reward* $r^L_{\text{release}} = 1 - r^L_{\text{contact}}$ then encourages the left hand to disengage, while the still-active r^R_{stable} encourages the right hand to retain the object. This phase-specific switch prevents the earlier left-grasp rewards from competing with release.

The episode is successful when the right hand holds the object stably ($\sum_i c^R_i \ge N_{\min}$), none of the three task-relevant left fingertips remains in contact ($\sum_i c^L_i = 0$), and $\|\mathbf{v}_{\text{obj}}\| < 0.05\,\text{m/s}$, all held for $\Delta t_{\text{hold}} = 1.0\,\text{s}$. The maximum episode length is $T_{\max} = 450$ steps.

Bimanual Lifting. Both arms simultaneously grasp a large container and lift it cooperatively. The reward requires both hands to secure the object before the lift phase begins.

In Phase 1 (Dual approach), r^L_{reach} and r^R_{reach} are active simultaneously, each targeting a grasp site on opposite sides of the container. Phase 2 starts once both fingertip centroids are within d_{prox} of their respective grasp sites.

In Phase 2 (Dual grasp), r_{finger}, r_{contact}, and r_{stable} activate independently for each arm. A *synchronization reward* couples the two hands by taking the minimum of their individual stability scores:

$$r_{\text{sync}} = \min\left(r^L_{\text{stable}},\, r^R_{\text{stable}}\right). \tag{19}$$

The minimum in Eq. (19) is high only when both individual stability scores are high. The episode advances to Phase 3 once both grasps remain stable for $\delta_g = 5$ consecutive steps.

In Phase 3 (Cooperative lift), the lift reward (Eq. (12)) and the orientation penalty (Eq. (13)) activate. A *symmetry reward* encourages both hands to remain at similar heights:

$$r_{\text{sym}} = \exp\left(-\lambda_s \left|h^L_{\text{ft}} - h^R_{\text{ft}}\right|\right), \tag{20}$$

In Eq. (20), h^L_{ft} and h^R_{ft} are the vertical positions of the left and right fingertip centroids, and $\lambda_s = 15$. Rewarding height symmetry discourages the container from tilting during the lift.

The episode is successful when $h_{\text{obj}} - h_{\text{init}} > h_{\text{target}}$, both grasps are stable, and $|h^L_{\text{ft}} - h^R_{\text{ft}}| < 0.03\,\text{m}$, all held for $\Delta t_{\text{hold}} = 1.0\,\text{s}$. The maximum episode length is $T_{\max} = 450$ steps.

C Implementation Details

This section specifies geometric encoder training (§C.1), adaptive domain randomization (ADR; §C.2), massively parallel Soft Actor-Critic (SAC) training (§C.3), and diffusion student training (§C.4).

C.1 Geometric Encoder Training

Table 7 summarizes the architecture, data, optimization, and augmentation settings for the geometric encoder.

The encoder builds on the Small variant of Depth Anything V2 (Yang et al., 2024). It pairs a DINOv2 Vision Transformer (ViT)-Small backbone (Oquab et al., 2024) with a Dense Prediction Transformer (DPT) fusion neck (Ranftl et al., 2021). The backbone has patch size 14 and embedding dimension 384. Following the zero-based indexing of the official Small model, layers {2, 5, 8, 11} supply four intermediate token sets. DPT reassembly projects these levels to 48, 96, 192, and 384 channels and fuses them at width 64, matching the official Small configuration. Two 128-channel task heads predict inverse depth and surface normals. A separate learned 1×1 convolution projects the fused map to the 256-channel policy representation exported together with the global classification `[CLS]` token.

Loss Definitions

Let $u^*_p = 1/z^*_p$ be the simulator inverse depth and let $\hat{u}_p$ be the predicted inverse depth. We align the prediction with scale s and shift t:

$$(s^*, t^*) = \arg\min_{s,t} \sum_{p \in \mathcal{M}_0} \left(s\hat{u}_p + t - u^*_p\right)^2. \tag{21}$$

The set $\mathcal{M}_0$ contains finite simulator depths in $[0.2, 1.8]\,\text{m}$. Using the alignment in Eq. (21), we define $\tilde{u}_p = s^*\hat{u}_p + t^*$. The scale- and shift-invariant depth loss and four-scale gradient loss are

$$\mathcal{L}_{\text{ssi}} = \frac{1}{|\mathcal{M}_0|} \sum_{p \in \mathcal{M}_0} \left|\tilde{u}_p - u^*_p\right|, \tag{22}$$

$$\mathcal{L}_{\text{gm}} = \sum_{\ell=0}^{3} \frac{1}{|\mathcal{M}_\ell|} \sum_{p \in \mathcal{M}_\ell} \left\| \nabla\tilde{u}^{(\ell)}_p - \nabla u^{*(\ell)}_p \right\|_1. \tag{23}$$

Table 7: Geometry-supervised encoder configuration. The model uses a DINOv2 small Vision Transformer with patch size 14 and a Dense Prediction Transformer neck. It is trained on 50,000 simulation images, validated on 2,500 images, and optimized for 50 epochs on eight A800 GPUs.

Component	**Value**	**Component**	**Value**
Architecture		*Optimization*	
Backbone	DINOv2 ViT-S (patch 14)	Optimizer	AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.999$)
Feature layers (zero-based)	$\{2, 5, 8, 11\}$	Weight decay	0.01
DPT fusion width	64	Learning rates	5×10^{-6} / 1×10^{-5}
Level output channels	48, 96, 192, 384	Rate schedule	Cosine, min 4×10^{-7}
Head hidden channels	128 / 128	Gradient clip	5.0
Exported channels	256 (1×1 conv.)	Epochs / batch size	50 / 32 × 8 GPUs
Initial checkpoint	Depth Anything V2 Small	Precision	16-bit floating point
Data		*Loss Weights*	
Train / val samples	50,000 / 2,500	λ_{ssi} / λ_{gm} (depth)	1.0 / 1.0
Input resolution	518×518	$\lambda_{\cos}$ / λ_{geo} (normal)	1.0 / 2.0
Scene depth range	$[0.2,\ 1.8]$ m	λ_{dn} (consistency)	0.3
Image normalization	ImageNet μ, σ		
Data Augmentation			
Color jitter ($p = 0.5$)	brightness/contrast/ saturation ± 0.2, hue ± 0.1	Horizontal flip ($p = 0.5$)	Normal x-flip
Geometric perturbation	rotation $\pm 10°$, translation/scale $\pm 5\%$	Gaussian noise ($p = 0.3$)	$\sigma = 0.01$

In Eqs. (22) and (23), superscript ℓ denotes downsampling by 2^ℓ, and $\mathcal{M}_\ell$ is the validity mask at that scale. For unit normals, we use

$$\mathcal{L}_{\cos} = \frac{1}{|\mathcal{M}_0|} \sum_{p \in \mathcal{M}_0} \left(1 - \hat{\mathbf{n}}_p^\top \mathbf{n}_p^*\right) , \tag{24}$$

$$\mathcal{L}_{\text{geo}} = \frac{1}{|\mathcal{M}_0|} \sum_{p \in \mathcal{M}_0} e^{-10\|\nabla z_p^*\|_1} \|\nabla \hat{\mathbf{n}}_p\|_1 . \tag{25}$$

Equations (24) and (25) supervise angular accuracy and normal smoothness away from depth discontinuities, respectively. For $\mathcal{L}_{\text{dn}}$, we form $\tilde{z}_p = 1/\max(\tilde{u}_p, \epsilon)$ with $\epsilon = 10^{-6}$. We back-project $\tilde{z}$ with the rendered camera intrinsics and compute normals from centered finite differences. The mask $\mathcal{M}$ in Eq. (2) of the main paper contains pixels whose center and four finite-difference neighbors belong to $\mathcal{M}_0$ and satisfy $\tilde{u} > \epsilon$.

We render 50,000 RGB-depth-normal triplets in Isaac Lab (Mittal et al., 2025) and reserve 2,500 for validation. Images are rendered at 640×480 and resized to 518×518. The resize rescales the horizontal intrinsic parameters (f_x, c_x) by $518/640$ and the vertical parameters (f_y, c_y) by $518/480$. The consistency loss therefore back-projects depth with intrinsics expressed in the same pixel coordinate system as the network output. The scene depth range is $[0.2, 1.8]$ m, and images use ImageNet normalization.

We train the encoder and decoder jointly for 50 epochs. The effective batch size is 256 on eight NVIDIA A800 80 GB GPUs. Training uses 16-bit floating-point mixed precision. AdamW uses learning rates of 5×10^{-6} for the pretrained backbone and 1×10^{-5} for the decoder and heads. A cosine schedule decays to 4×10^{-7}.

The loss weights in Eq. (1) of the main paper are $\lambda_{\text{ssi}} = 1.0$, $\lambda_{\text{gm}} = 1.0$, $\lambda_{\cos} = 1.0$, $\lambda_{\text{geo}} = 2.0$, and $\lambda_{\text{dn}} = 0.3$.

Data augmentation uses color jitter (brightness, contrast, and saturation ± 0.2; hue ± 0.1; probability 0.5) and random geometric perturbations (rotation $\pm 10°$; translation and scale $\pm 5\%$). The same image-plane warp is applied to RGB, inverse depth, normals, and validity masks. Translation and isotropic scale warp the normal map without changing vector components; in-plane rotation rotates

the normal x–y components by the same angle. Horizontal flips ($p = 0.5$) negate the normal x component. The transformed intrinsics are used whenever the augmented sample contributes to the depth-normal consistency term. We also add Gaussian image noise ($\sigma = 0.01, p = 0.3$).

Prediction Metrics and Results

Table 8 reports depth and surface normal prediction quality on the 2,500-image validation set rendered in Isaac Lab. Depth predictions are aligned to the ground truth via median scaling before evaluation. Normal metrics measure the per-pixel angular error (in degrees) between predicted and ground-truth unit normals.

We monitor the following metrics during training, where $\hat{d}$ denotes the median-scale-aligned prediction and d^* denotes the reference depth. For depth, δ_1 is the fraction of pixels with $\max(\hat{d}/d^*,\ d^*/\hat{d}) < 1.25$; Abs Rel is the mean relative error $|\hat{d} - d^*|/d^*$; RMSE is the root-mean-square error; and SILog is the scale-invariant logarithmic error (Eigen et al., 2014). For normals, we report mean and median angular error and the fraction of pixels below $11.25°$ and $22.5°$.

For "w/o Fine-tuning," the shared pretrained encoder remains at its initialization and does not receive geometry-supervised fine-tuning. The row therefore isolates the effect of adapting the shared representation to the simulation geometry.

Table 8: Geometric prediction quality on 2,500 simulation validation images. Depth is aligned by median scale. δ_1 is the fraction within a factor of 1.25. Abs Rel is mean relative error, root-mean-square error (RMSE) is reported in meters, and SILog is scale-invariant log error. Normal mean and median are in degrees, while the threshold columns report fractions. "Not predicted" denotes an output omitted by construction in a single-task variant. Arrows show the preferred direction, and the best value in each column is bold.

	Depth				Normal			
Variant	δ_1 ↑	Abs Rel ↓	RMSE ↓	SILog ↓	Mean ↓	Med. ↓	11.25° ↑	22.5° ↑
Full model	**0.989**	**0.031**	**0.086**	**0.065**	**2.003**	**0.563**	**0.961**	**0.979**
w/o $\mathcal{L}_{\text{dn}}$	0.984	0.037	0.098	0.078	2.416	0.782	0.948	0.972
w/o Normal	0.986	0.034	0.092	0.071	Not predicted			
w/o Depth	Not predicted				2.587	0.891	0.943	0.968
w/o Fine-tuning	0.941	0.072	0.168	0.134	5.832	2.714	0.876	0.937

The full model obtains $\delta_1 = 0.989$ and a mean normal error of $2.0°$. Removing $\mathcal{L}_{\text{dn}}$ changes Abs Rel from 0.031 to 0.037 and increases mean normal error by $0.4°$; both prediction metrics improve with the consistency term.

The depth-only and normal-only variants have higher errors than the joint model on their retained outputs. Without fine-tuning, Abs Rel more than doubles and mean normal error nearly triples. This setting also has the lowest downstream rate in Table 2a of the main paper. The comparison does not isolate a causal path between prediction error and control success.

C.2 Adaptive Domain Randomization

Table 9 distinguishes parameters with VP-SVGD-controlled bounds from parameters sampled over fixed ranges. Table 10 reports the VP-SVGD hyperparameters.

Randomization Parameters

Table 9 lists all randomized parameters $\boldsymbol{\xi}$. VP-SVGD controls ten scalar dimensions: object x, y, yaw, and mass; three object-contact parameters; and three robot-contact parameters. Size, lighting, appearance, and camera pose are sampled independently from fixed ranges. Object and robot friction are sampled separately; Isaac Lab combines the two surface values according to its contact model (Mittal et al., 2025).

VP-SVGD Bound Updates

Normalized coordinates and physical mapping. VP-SVGD operates in a normalized space $\boldsymbol{\alpha} \in [-1, 1]^{10}$ for the ten "VP" dimensions in Table 9. The origin is nominal, and the two endpoints reach

Table 9: Domain randomization parameters. "VP" marks the ten scalar dimensions whose bounds are controlled by Virtual Particle Stein Variational Gradient Descent (VP-SVGD). "Fixed" marks parameters sampled from the listed range throughout training. Nominal is the default simulator value.

Parameter	**Nominal**	**Range**	**Control**
Global			
Lighting intensity	3000	[1000, 4000]	Fixed
Lighting color (per channel)	0.75	[0.4, 1.0]	Fixed
Object Basic			
Position offset (x, y) (m)	0.0	$[-0.15, 0.15]$	VP (2)
Yaw offset (rad)	0.0	$[-\pi, \pi]$	VP (1)
Mass scale	1.0	[0.5, 2.0]	VP (1)
Size scale	1.0	[0.8, 1.2]	Fixed
Object Physics			
Static friction	0.5	[0.3, 1.2]	VP (1)
Dynamic friction	0.5	[0.2, 1.0]	VP (1)
Restitution	0.0	[0.0, 0.2]	VP (1)
Robot Physics			
Static friction	0.5	[0.3, 1.2]	VP (1)
Dynamic friction	0.5	[0.2, 1.0]	VP (1)
Restitution	0.0	[0.0, 0.2]	VP (1)
Object Visual			
Roughness	0.5	[0.0, 1.0]	Fixed
Metallicity	0.5	[0.0, 1.0]	Fixed
Specular reflectance	0.5	[0.0, 1.0]	Fixed
Color (per channel)	0.0	[0.0, 1.0]	Fixed
Camera			
Position-offset magnitude (m)	0.0	[0.0, 0.01]	Fixed
Rotation-offset magnitude (rad)	0.0	[0.0, 0.05]	Fixed

the listed physical limits. We first define the monotone coordinate map

$$\mathcal{T}_i(a) = \begin{cases} \xi_i^{\text{start}} + a\big(\xi_i^{\text{high}} - \xi_i^{\text{start}}\big), & a \geq 0, \\ \xi_i^{\text{start}} + (-a)\big(\xi_i^{\text{low}} - \xi_i^{\text{start}}\big), & a < 0, \end{cases} , \tag{26}$$

where ξ_i^{start} is the nominal value and $\xi_i^{\text{low}}, \xi_i^{\text{high}}$ are the task-defined limits. Equation (26) sends -1, 0, and 1 to the lower limit, nominal value, and upper limit, respectively.

Let $v_i^{(m)}$ be coordinate i of particle m. The pose coordinates (x, y, yaw) use symmetric magnitudes around the nominal pose. For each such coordinate, m_i is the 95th percentile of $\{|v_i^{(m)}|\}_{m=1}^M$, and the next interval is $[\mathcal{T}_i(-m_i), \mathcal{T}_i(m_i)]$. Mass and the six contact coordinates use separate lower and upper bounds obtained from the 5th and 95th percentiles of $\{v_i^{(m)}\}_{m=1}^M$. Applying $\mathcal{T}_i$ to these percentiles yields the physical bounds. Thus pose ranges expand or contract symmetrically, while asymmetric physical ranges can also shift. Parameters marked "Fixed" are sampled independently from their full ranges and do not enter $\boldsymbol{\alpha}$. The particle update uses Stein Variational Gradient Descent (SVGD) (Liu & Wang, 2016).

ADR update cycle. At each ADR cycle, the simulator samples adaptive values from the current bounds and fixed parameters from Table 9. Binary episode outcomes train a difficulty model $D_\psi(\boldsymbol{\alpha})$ with binary cross-entropy, using target one for failure and zero for success. SVGD transports virtual particles, and kernel repulsion maintains particle diversity. The particle statistics defined above produce the next axis-aligned adaptive bounds.

The particles never enter the simulation. Their coordinate-wise statistics define a uniform box for the next ADR cycle. Particles in different regions can change these percentiles, but simulator sampling remains axis-aligned and uniform rather than following the full particle density.

Table 10: Virtual Particle Stein Variational Gradient Descent hyperparameters. The left block lists particle transport and bound projection settings; symmetric pose ranges use the 95th percentile of particle magnitude. The right block lists the difficulty-model architecture and adaptive curriculum thresholds. The ten dimensions are object x, y, yaw, and mass plus three object-contact and three robot-contact parameters. All values are shared across tasks.

Component	**Value**	**Component**	**Value**
Particle Transport		*Difficulty Model*	
Particles (M)	128	Architecture	MLP $10 \to 64 \to 64 \to 1$
Step size (η)	0.025	Learning rate	3×10^{-3} (AdamW)
Kernel bandwidth	Median heuristic	Updates / cycle (K)	5
Parameter dimensions	10		
Bound Projection		*Adaptive Curriculum*	
Percentiles	5th / 95th	Expansion threshold	Success rate $\geq 20\%$
Minimum normalized width	0.02	Contraction threshold	Success rate $< 5\%$
		Contraction decay (β)	0.01

VP-SVGD Hyperparameters

Table 10 lists the VP-SVGD hyperparameters. The module is implemented as a Gymnasium `Wrapper` that intercepts the `step()` call, requiring no modification to the underlying environment or the RL algorithm.

The difficulty model D_ψ is a three-layer multilayer perceptron (MLP), $10 \to 64 \to 64 \to 1$, with rectified-linear activations and a sigmoid output. AdamW applies five binary-cross-entropy updates per ADR cycle. The 128 particles are initialized at the origin with small Gaussian perturbations. Stein Variational Gradient Descent uses step size 0.025 and a radial basis function (RBF) kernel with median bandwidth. After every transport step, particles are clamped to $[-1, 1]^{10}$. Bound projection uses the statistics above and enforces a minimum normalized width of 0.02 per dimension.

The adaptive curriculum gates the update by the current success rate. When the success rate is at least 20%, the full VP-SVGD cycle executes. When it drops below 5%, the particles contract toward the origin via exponential decay ($\beta = 0.01$). Otherwise, the adaptive bounds remain fixed.

C.3 Massively Parallel SAC Training

Preliminary runs with 4,096 parallel environments were unstable when Soft Actor-Critic (SAC) (Haarnoja et al., 2018) used its initial replay, target-network, and actor-update settings. The reported configuration changes all three, as listed in Table 11.

The critic uses a double-Q architecture, each branch being a three-layer MLP (hidden sizes 1024, 512, 256) with ReLU activations and layer normalization. The actor is a diagonal Gaussian policy parameterized by a three-layer MLP (hidden sizes 512, 256, 128) with log-std bounds $[-5, 5]$ and initial log-std -2.0.

With this architecture fixed, we use three implementation changes for massively parallel training. First, the replay buffer stores 5×10^6 transitions. After each vectorized collection step, we replay the sampled batch of 8,192 transitions for eight gradient epochs before collecting again. The reported 5×10^8 environment steps count aggregate simulator transitions across all parallel environments.

The second adaptation concerns the target network. In preliminary runs, the default slow-tracking target ($\tau = 0.005$) lagged behind the critic as the replay distribution changed. We increase the smoothing coefficient to $\tau = 0.05$ and update the target at every gradient step.

Third, the actor is updated once per four critic updates. This is the setting used for the reported expert. We do not report an isolated ablation of this ratio. The critic and actor learning rates are 5×10^{-4} and 1×10^{-4}, respectively. The target-entropy ratio of 1.0 corresponds to the standard target $-d_a$, where d_a is the action dimension. Evaluation uses fixed observation statistics.

C.4 Diffusion Student Training

Table 12 summarizes the diffusion-student architecture, data, and optimization settings.

Table 11: Soft Actor-Critic hyperparameters for the privileged reinforcement-learning expert. The left block lists environment and replay-buffer settings. The right block lists actor, critic, entropy, and general training settings. Training uses eight A800 80 GB graphics processing units (GPUs).

Component	Value	Component	Value
Environment		*Actor*	
Parallel environments	4,096	Architecture	MLP 512, 256, 128
Random exploration	16 steps per env	Learning rate	1×10^{-4}
Environment steps	5×10^{8}	Weight decay	0.01
		Update frequency	Every 4 critic updates
Replay Buffer		*Critic*	
Capacity	5×10^{6}	Architecture	Double-Q MLP 1024, 512, 256
Batch size	8,192	Learning rate	5×10^{-4}
Replay epochs / collection step	8	Weight decay	0.01
n-step returns	3	Target τ	0.05
		Target update frequency	Every gradient step
Entropy		*General*	
Initial temperature (α)	0.05	Discount (γ)	0.99
Auto-tune	Enabled	Gradient clip	0.5
α learning rate	5×10^{-4}	Observation normalization	Running statistics
Target entropy ratio	1.0	Rate schedule	Linear decay
		Precision	bfloat16 mixed, `torch.compile`

The denoising backbone is a one-dimensional Diffusion Transformer (DiT) with 12 layers, 768-dimensional hidden states, and 12 attention heads. Conditioning enters through local and global pathways. For local conditioning, the geometric encoder exports dense spatial tokens from both views. A multilayer perceptron maps proprioception to 64-dimensional embeddings, which are projected to 768 dimensions and concatenated with the visual tokens along the sequence axis. These tokens provide the cross-attention keys and values at every transformer layer. For global conditioning, we combine the camera `[CLS]` tokens, mean-pooled proprioception, and the Fourier timestep embedding. Adaptive Layer Normalization (AdaLN) uses this vector to modulate each layer. The full spatial token sequence is preserved for cross-attention.

We retain 400 successful expert trajectories per task. Each trajectory records two RGB streams, proprioception, and a 26-dimensional action. The policy observes one timestep, predicts eight actions, executes four, and then replans. During dataset preprocessing, image augmentation is applied before the frozen encoder and the resulting features are cached. No augmentation is applied after caching. For the 224×224 policy input, the ViT uses the backbone's standard bicubic interpolation of positional embeddings from the pretrained grid to the 16×16 patch grid. Action normalization uses the per-channel transformation defined in Appendix B.2.

We use the noise-prediction objective in Eq. (5) of the main paper with a continuous cosine schedule. AdamW optimizes the denoiser at 3×10^{-4}, while the encoder remains frozen. Training uses 50,000 steps, 2,000 warm-up steps, a per-GPU batch size of 32, and bfloat16 mixed precision. An exponential moving average (EMA) with decay 0.999 is used for inference. Denoising Diffusion Implicit Models (DDIM) sampling uses 16 steps. The augmentation configuration includes a $1.1\times$ resize, a random 224×224 crop, and color jitter.

Table 12: Diffusion-student configuration. DiT denotes Diffusion Transformer, DDIM denotes Denoising Diffusion Implicit Models, EMA denotes exponential moving average, and MLP denotes multilayer perceptron. The encoder is frozen; image augmentation precedes encoding, and ViT positional embeddings are interpolated to the policy-input patch grid.

Component	**Value**	**Component**	**Value**
Architecture		*Data*	
DiT layers / dim / heads	12 / 768 / 12	Action dim (d_a) / horizon (H)	26 / 8
Timestep embedding	Fourier	Observation horizon	1 step
Cross-attention	All layers	Image resolution	224×224
Feature aggregation	Full tokens (no pooling)	Action normalization	Velocity z-score; hand-offset z-score
Proprioception MLP width	64	Data frequency	30 Hz
Condition output dim	768		
Diffusion		*Optimization*	
Noise schedule	Cosine, continuous-time	Optimizer	AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.999$)
Prediction target	ϵ	Weight decay	1×10^{-4}
DDIM steps	16	Denoiser / encoder LR	3×10^{-4} / frozen
Sampling temperature	1.0	Rate schedule	Cosine, min 1×10^{-6}
		Warm-up steps	2,000
		Steps / batch	50,000 / 32 per GPU
		EMA decay	0.999
		Gradient clip	1.0
		Precision	bfloat16 mixed, `torch.compile`
Data Augmentation			
Crop	$1.1\times$ resize $\rightarrow 224 \times 224$	Color jitter	Bright./contr./sat. ± 0.1, hue ± 0.02