POINTPILOT: UNIFIED ROBOTIC MANIPULATION VIA OBJECT-CENTRIC 3D POINT FLOW PREDICTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Generalization remains a central challenge in robotics, calling for policies that address domain variations from multiple dimensions such as diverse embodiments, camera poses, and/or scene layouts. In order to approach this problem, we introduce **PointPilot**, a task-focused and 3D aligned manipulation framework built on a 3D space point-flow prediction. Rather than modeling the global scene, PointPilot only represents task-focused objects (robot grippers treated as objects), avoiding the prediction of potentially redundant scene parts unrelated to the manipulation tasks. Moreover, PointPilot aligns 3D spatial representation in one unified coordinate and supports closed-loop and fully automatic action execution across different manipulation modes from object point flow without a separately trained action expert, while existing works typically need an extra action head or are (semi-)automatic and single-arm mode specific. Given sensor observations and language instructions, our model jointly learns subtask planning, task-relevant object grounding, and 3D point flow prediction in an end-to-end way. By processing heterogeneous data in a shared 3D point flow space, PointPilot is able to absorb demonstrations from different embodiments. Extensive simulative and real world experiments validate our method. *Notably*, we demonstrate the zero-shot embodiment generalization in the PointPilot — one trained policy can be deployed onto previously unseen embodiments and successfully complete manipulation tasks, without any additional fine-tuning.

1 INTRODUCTION

A central promise of generalist robot learning is that manipulation policies should improve as they absorb increasingly diverse data. However, realizing this promise requires more than scaling model parameters and demonstration hours. Robot data is heterogeneous not only in visual appearance, task semantics, and environment configuration, but also in embodiments. Native actions from heterogeneous data lack a shared representation form, preventing action prediction from directly serving as a unified supervision space. The choice of prediction space therefore determines whether heterogeneous demonstrations provide compatible supervision or remain isolated by their control interfaces.

Vision–language–action(VLA) models have made substantial progress toward pretraining a single policy across tasks and robot platforms (Open X-Embodiment Collaboration et al., 2023; Ghosh et al., 2024; Black et al., 2025) and finetuning with a specific task. Their apparent unification primarily occurs at the model level, for the reason that learned targets remain robot actions, whose dimensionality, coordinate convention, and control semantics depend on the embodiment and manipulation setup. Existing systems accommodate these differences through data pre-processing, embodiment conditioning, or specialized action heads (Open X-Embodiment Collaboration et al., 2023; Wang et al., 2024; Doshi et al., 2025; Zheng et al., 2025a;b).

Task-relevant object motion offers a natural source of shared supervision because it captures task motion without committing to a robot's joints or visual appearance. This insight has motivated a growing family of representations, including articulation flow, general 3D object flow, object-part scene flow, dense 3D motion fields, language-conditioned point trajectories, and point-cloud world models (Eisner et al., 2022; Yuan et al., 2025; Xu et al., 2025; Tang et al., 2025; Yin et al., 2025; Zhang et al., 2026; Huang et al., 2025b; Lin et al., 2026).

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.

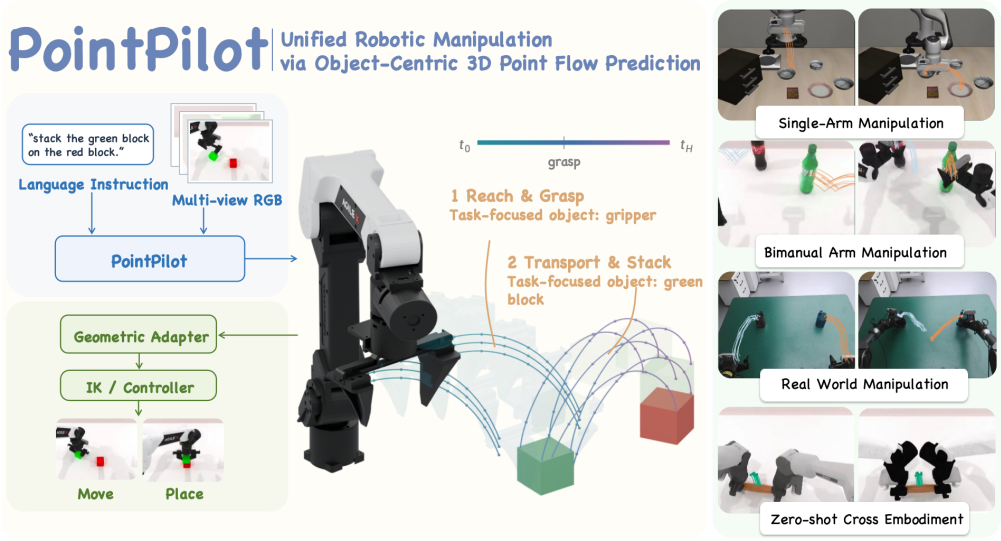

Figure 1: **Overview of PointPilot. Left:** Given a language instruction and multi-view RGB observations, PointPilot predicts task-focused objects' 3D point flow in each subtask. A learning-free adapter converts the predicted flow into robot end-effector pose commands for execution under the coupling condition. **Right:** PointPilot unifies in different manipulation modes and embodiments, thus supporting single-arm and bimanual gripper to execute any manipulation tasks by object-centric prediction. We further demonstrate real-world manipulation and promising zero-shot cross-embodiment skills transfer.

Despite these advances in object-centric prediction for robotic manipulation, a unified and scalable framework remains challenging. To the best of our knowledge, existing approaches generally fall into three patterns: **a)** Without any 3D alignment, they only use object motion as an intermediate latent feature for an additional action decoder, thus hardly processing the cross-embodiment generalization(Xu et al., 2025; Yang et al., 2025; Lin et al., 2026); **b)** depend on manual manipulation or an extra controller to grasp objects at the beginning, causing a modular framework and only support an open-looped or semi-automatic execution (Tang et al., 2025; Yin et al., 2025; Yuan et al., 2025); **c)** or specialize object-centric representations to one particular manipulation mode (Eisner et al., 2022; Lin et al., 2026; Tang et al., 2025; Yang et al., 2025; Yuan et al., 2025). **In particular**, manipulation modes such as single-arm or bimanual arm manipulation may not be a problem in VLA paradigm, but it is a problem in the object-centric prediction. Unlike VLA, different manipulation modes require different numbers of object flows and different object-to-gripper motion assignments. To simplify this, most object-centric methods specify their flow to one object and only execute single-arm tasks.

On the other hand, an ideal unified and scalable object-centric prediction framework should be **manipulation-mode agnostic**, allowing a single model to handle both single-arm and dual-arm manipulation by predicting motion for a variable number of objects; **embodiment-agnostic**, allowing heterogeneous robot demonstrations to be composed despite incompatible native action spaces; and **directly executable**, allowing the object flow prediction to control the robot in a closed loop without an additional embodiment-dependent action decoder. Existing object-centric approaches typically satisfy only a subset of these requirements, leaving object motion either mode-specific, embodiment-dependent, or disconnected from fully automatic execution.

To alleviate this gap, **1)** we introduce a novel manipulation formulation that predicts task-focused future states and robot actions, in a unified 3D point-flow representation of relevant objects. Notable, we treat robot gripper tools as special objects and use the proposed 3D point-flow as an action representation, which accommodates different manipulation modes (e.g., single-arm or bimanual) and robot embodiments (e.g., parallel-jaw or dexterous hand) without platform-specific action experts. Moreover, **2)** we align the 3D coordinates for both multi-view input observations and predicted 3D point flows in a shared reference frame. This alignment establishes cross-view spatial correspondences for visual and geometric fusion while providing a consistent prediction space across robot embodiments. Additionally, **3)** we integrate task understanding, semantic grounding, 3D point flow

prediction and action generation into one model. This enables end-to-end training and fully autonomous closed-loop rollout, compared to various related works (Duan et al., 2024; Huang et al., 2024a;b; Pan et al., 2025; Tang et al., 2025; Yuan et al., 2025) that rely on external modules (e.g., sub-task orchestration or grasp planning) during the real world deployment. With these principles, **4)** we develop a comprehensive data processing pipeline, design the PointPilot model architecture, perform jointly training on heterogeneous robot data sources, and conduct various simulation and real world experiments. PointPilot demonstrates promising generalization and scalability, such as zero-shot cross embodiment transfers.

2 Related Work

2.1 Object-Centric Motion Representations for Manipulation

Prior works have shown that object-centric 3D flow provides a better generalization across unseen objects, scenes, and robot embodiments (Eisner et al., 2022; Tang et al., 2025; Yuan et al., 2025; Yin et al., 2025).

Table 1: **Comparison with prior object-centric methods.** Feat.1 means zero-shot cross-embodiment transfer. Feat.2 means no separate grasping module. Feat.3 means no learned motion-to-action action head. Feat.4 means single-arm manipulation tasks. Feat.5 means bimanual manipulation tasks.

Method	Feat.1	Feat.2	Feat.3	Feat.4	Feat.5
Im2Flow2Act (Xu et al., 2025)	✗	✓	✗	✓	✗
PPI (Yang et al., 2025)	✗	✓	✗	✗	✓
RoboFlow4D (Lin et al., 2026)	✗	✓	✗	✓	✗
General Flow (Yuan et al., 2025)	✗	✗	✓	✓	✗
Object-Part Scene Flow (Tang et al., 2025)	✓	✗	✓	✓	✗
3D Motion Field (Yin et al., 2025)	✗	✗	✓	✓	✗
FlowBot3D (Eisner et al., 2022)	✗	✓	✓	✓	✗
ImaginationPolicy (Lu et al., 2025)	✓	✗	✓	✓	✗
PointPilot (Ours)	✓	✓	✓	✓	✓

However, existing object-centric approaches broadly follow several patterns. One line treats predicted flow or point trajectories as intermediate motion guidance for downstream action expert training (Xu et al., 2025; Yang et al., 2025; Lin et al., 2026; Zhang et al., 2026). One line directly converts predicted objects' flow into robot trajectories, but does not provide a complete autonomous and closed-looped manipulation. They either manually manipulate the gripper or use an extra controller to grasp the object. (Bharadhwaj et al., 2024; Yuan et al., 2025; Zhi et al., 2025). Another line is also often confined to a single manipulation mode, such as the single-arm manipulation (Eisner et al., 2022; Yin et al., 2025; Yuan et al., 2025; Zhi et al., 2025; Lu et al., 2025; Lin et al., 2026). Most works only execute single-arm tasks for the reason that only one object needs to be predicted. The detailed comparison is shown in Table 1. Although RoboFlow4D is not a typical object-centric method, we include it because it represents gripper motion as point flow.

2.2 Cross-Embodiment Policy Transfer

Cross-embodiment policies use embodiment conditioning or shared latent actions to learn from heterogeneous data (Zheng et al., 2025b; Bu et al., 2025). Object-centric 3D flow also supports human-to-robot transfer (Yuan et al., 2025) and cross-platform deployment (Zhi et al., 2025). PointPilot separates task-focused 3D motion prediction from robot-specific execution: a non-learned geometric adapter maps predicted flows to end-effector targets under local rigid coupling. We demonstrate deployment on unseen embodiments without target-robot demonstrations or fine-tuning, using each robot's kinematics and controller.

3 PointPilot: Unified Object-centric 3D Point Flow Prediction

Building on the definition of *task-focused objects*, we introduce *PointPilot*. In this section, we will present the model architecture, covering subtask planning, object grounding, 3D geometric alignment, cross-view fusion , and point-flow predicting, and then describe how the predicted point flow is mapped to executable end-effector trajectories through a learning-free adapter.

3.1 Preliminary: Task-focused objects for 3D Point Flow Learning

We define a *task-focused object* as a physical entity whose motion characterizes the manipulation required at the current stage and can be mapped to the motion of one or both grippers. Specifically, for each gripper, the task-focused object is either the gripper itself or a locally rigid object part currently coupled to it. To support different manipulation modes, each task-focused object is addi-

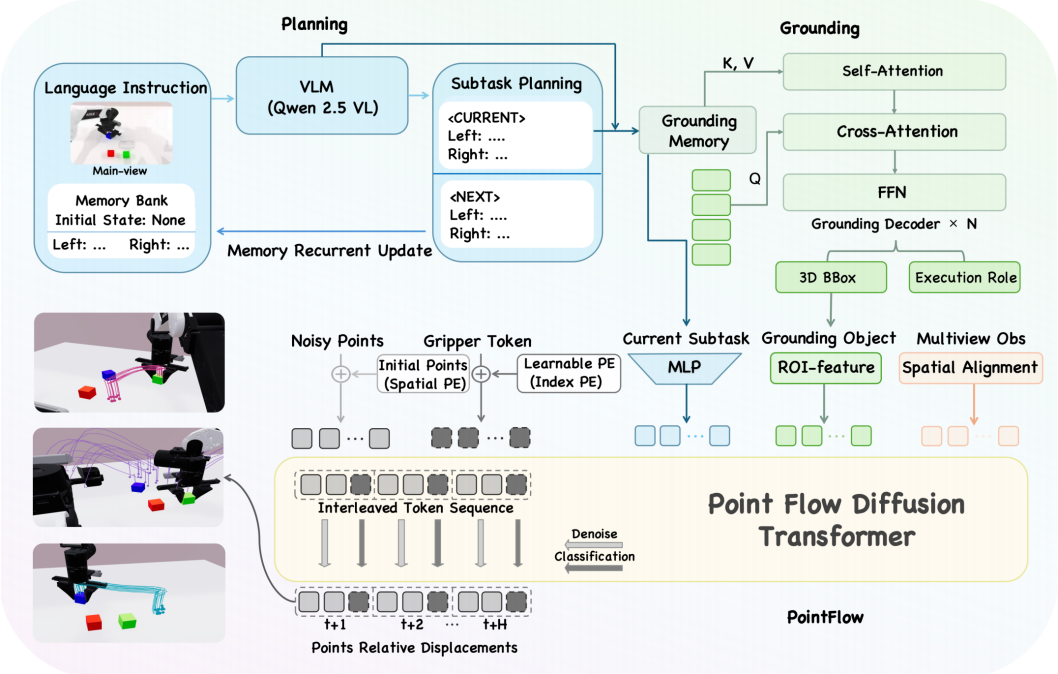

Figure 2: **Method of PointPilot**. **(a) Planning**: A memory-conditioned VLM predicts current and next subtasks. **(b) Grounding**: Task-conditioned 3D grounding localizes task-focused objects and assigns execution roles. **(c) PointFlow**: Multi-view observations are geometrically aligned and used by Point DiT to jointly predict relative 3D point flow and gripper states. Predicted object motion is converted to end-effector trajectories through a learning-free geometry adapter for closed-loop execution.

tionally assigned an *execution role*, indicating the gripper to which its predicted point flow should be mapped.

For example, when the left gripper approaches an object, the left gripper itself serves as the task-focused object with the execution role *Left*. Once the object is grasped and transported, the task-focused object switches from the gripper to the coupled object, whose motion then directly characterizes the desired manipulation.

Under this formulation, PointPilot represents both gripper motion and manipulated-object motion within a shared point-flow space by casting manipulation as the motion of task-focused objects.

3.2 Subtask Planning

We use a memory-conditioned VLM planner to identify the current manipulation stage and anticipate the next subtask. At planning time t, the planner receives the main-view RGB image I_t^{main}, the global language instruction ℓ, and a textual memory bank $\mathcal{M}_t$ that records previously predicted subtasks. At the beginning of each episode, the memory bank is initialized to `NONE`.

Using Qwen2.5-VL as the backbone, the planner generates a structured plan containing `CURRENT` and `NEXT` spans:

$$(\hat{s}_t, \hat{s}_t^+) = VLM_\theta\left(I_t^{\text{main}}, \ell, \mathcal{M}_t\right). \tag{1}$$

Each span specifies a left/right subtask pair, with `NONE` indicating an inactive hand. The `CURRENT` span $\hat{s}_t$ describes the subtasks to execute at the current stage. The `NEXT` span $\hat{s}_t^+$ specifies the immediate next distinct subtask pair expected to begin within the H-step horizon, or `NONE` if no new next subtask is expected. The hidden states associated with `CURRENT` provide task features $\mathbf{Z}_t^{\text{task}}$ for subsequent 3D object grounding and point-flow prediction.

For the next planning call at $t + H$, the memory is updated using only the previously predicted subtask spans:

$$\mathcal{M}_{t+H} = \text{Concat}\left(\mathcal{M}_t, \hat{s}_t, \hat{s}_t^+\right). \tag{2}$$

The detailed design of VLM prompt and memory bank is indicated in the Appendix B.

3.3 LANGUAGE-GUIDED 3D GROUNDING

Geometry Frustum Inspired by PETR (Liu et al., 2022a;b) in autonomous driving and DexBEV (Zhou et al., 2026) in embodied intelligence, we construct the discretized camera frustums for frames. Specifically, we first inject camera-aware 3D positional information into the main-view image tokens. For an image token at $\tilde{\mathbf{u}}_i = [u_i, v_i, 1]^\top$, we sample D depth hypotheses along its corresponding camera ray using linearly increasing discretization (LID). Let N_{img} denote the number of main-view image tokens and D the number of depth hypotheses. For $i \in \{1, \dots, N_{\text{img}}\}$ and $k \in \{0, \dots, D-1\}$, we define

$$z_k = z_{\min} + \frac{z_{\max} - z_{\min}}{D(D+1)} k(k+1) \in [z_{\min}, z_{\max}), \qquad \mathbf{x}^c_{i,k} = z_k \tilde{\mathbf{K}}^{-1} \tilde{\mathbf{u}}_i \in \mathbb{R}^3. \tag{3}$$

where $\tilde{\mathbf{K}} \in \mathbb{R}^{3\times3}$ is the main-camera intrinsic matrix, and c denotes the main-camera coordinate frame. Following these equations, we obtain a ray-aligned 3D coordinate tensor $\mathbf{X}^c_i \in \mathbb{R}^{D\times3}$ for each main-view image token. Collectively, these coordinates form a discretized camera frustum $\mathbf{X}^c \in \mathbb{R}^{N_{\text{img}}\times D\times3}$.

The resulting camera-frame coordinates are then encoded into a 3D positional embedding $\mathbf{e}^{\text{3D}}_i \in \mathbb{R}^C$, which is added to the corresponding final-layer VLM image feature $\mathbf{H}^{\text{img}}_{t,i}$, obtaining d-dimensional geometry-aware image tokens. This construction requires neither observed nor predicted depth.

Grounding & Classification We concatenate these geometry-aware image tokens with the planner's current-subtask token features $\mathbf{Z}^{\text{task}}_t$ to form the grounding memory. A six-layer transformer decoder updates Q learnable queries through self-attention and cross-attention to this memory. Each decoded query predicts a 3D box with center $\mathbf{c} \in \mathbb{R}^3$, dimensions $\boldsymbol{\eta} \in \mathbb{R}^3$, and a 6D rotation representation $\mathbf{r} \in \mathbb{R}^6$, along with an execution role from $\{\text{invalid}, \text{Left}, \text{Right}, \text{Left\&Right}\}$.

The execution role associates each task-focused object with the gripper(s) responsible for its manipulation: Left routes this object's point flow to the left gripper, and the $\text{Left\&Right}$ denotes that both grippers are coupled to the same object and follow its motion.

During training, we use Hungarian matching to establish a one-to-one assignment between predicted queries and ground-truth task-focused objects. Matched queries are supervised with their assigned 3D boxes and execution roles, while unmatched queries are assigned the invalid role and excluded from box regression. The matching cost and loss definitions are detailed in the Appendix B.

3.4 POINT FLOW DIT

Spatial Alignment We use the main-view coordinate as the anchored reference frame for both observations and point flow prediction. Auxiliary-view images are encoded by the shared vision encoder. Following Sec. 3.3, we construct their discretized camera frustums and transform the sampled 3D coordinates into the main-camera frame using calibrated camera extrinsics:

$$\tilde{\mathbf{x}}^c_{i,k} = \mathbf{T}_{c_v \to c} \tilde{\mathbf{x}}^{c_v}_{i,k}, \qquad v \in \mathcal{V}_{\text{aux}} \tag{4}$$

where c_v denotes the coordinate frame of auxiliary views v, and $\mathbf{T}_{c_v\to c}$ is the coordinate transformation from auxiliary views v to the main view. These aligned frustums provide 3D positional embeddings for the auxiliary-view visual features.

Point Flow Prediction For each grounded object, we use Halton sampling within its predicted 3D bounding box to obtain N initial points. PointPilot predicts their future displacements $\Delta\mathbf{p}_h = \mathbf{p}_h - \mathbf{p}_0$ over an H-step horizon.

To augment the condition of the grounded object, each grounded object is represented by an object token $\mathbf{Z}^{\text{obj}}_t$. This token fuses geometry features of its sampled initial points processed by a light PointNet with grounded visual appearance features extracted by Region of Interest Align (ROIAlign) from the projected 3D bounding box region.

For point-flow generation, each future point displacement is represented by a noisy point token. In order to preserve its spatial identity, we augment each token with a spatial position embedding of the corresponding current 3D anchor point in the main-camera coordinate frame. Therefore, the model predicts motion from spatially grounded point tokens rather than from pure noise. Furthermore,

at each future step, noisy point tokens are interleaved with learned gripper tokens, enabling joint prediction of object motion and gripper states. We train the DiT with conditional flow matching to predict the velocity field that transports noisy point tokens to future point displacements. Gripper tokens are not noised and directly predict binary open/close states without the noising-denoising process.

The Point DiT is conditioned on the grounding memory, object tokens, and spatially aligned auxiliary-view visual features.

3.5 From Point Flow to Executable Robot Actions

Before grasping, predicted gripper flow guides execution. After grasping, object flow guides the coupled gripper, assuming a rigid grasp. Our learning-free adapter transforms current and predicted points into the world frame and fits a rigid motion:

$$(\hat{\mathbf{R}}^m_{t,h}, \hat{\mathbf{d}}^m_{t,h}) = \arg\min_{\mathbf{R}\in \mathrm{SO}(3),\, \mathbf{d}\in\mathbb{R}^3} \sum_{n=1}^{N} \left\| \mathbf{R}\mathbf{p}^{m,w}_{t,n} + \mathbf{d} - \hat{\mathbf{p}}^{m,w}_{t+h,n} \right\|_2^2. \tag{5}$$

The fitted relative transform $\Delta\hat{\mathbf{T}}$ updates each assigned end effector's current pose $\mathbf{T}_a$ to its target pose $\hat{\mathbf{T}}_a = \Delta\hat{\mathbf{T}}\mathbf{T}_a$, with all poses expressed in the world frame. For $\mathrm{Left\&Right}$, this transform acts on both grippers' current poses; independent motions use separate object flows. The resulting poses and predicted gripper states are executed through IK and the robot controller. PointPilot observes the updated scene and replans after each action chunk.

3.6 Training Objective

PointPilot is jointly optimized for subtask planning, 3D grounding, point-flow prediction, and gripper control:

$$\mathcal{L} = \lambda_1\mathcal{L}_{\mathrm{plan}} + \lambda_2\mathcal{L}_{\mathrm{ground}} + \lambda_3\mathcal{L}_{\mathrm{pointflow}} + \lambda_4\mathcal{L}_{\mathrm{gripper}}. \tag{6}$$

The VLM is trained with cross entropy in a teacher forcing way. Grounding combines 3D box regression and execution-role classification under Hungarian matching. Conditioned on predicted bounding boxes, point-flow prediction uses conditional flow matching on anchor-relative displacements, and gripper states are supervised with binary cross entropy. Full loss definitions, matching costs, masking rules, and optimization settings are provided in the Appendix B.

4 Data Pipeline for Task-focused Objects Point Flow

We construct PointPilot's stage-level supervision from simulated and real-robot demonstrations through interaction-stage segmentation and 3D trajectory extraction. This aligns task semantics with task-focused object motion without requiring shared low-level action encodings across data sources.

4.1 Stage Segmentation

We primarily segment demonstrations using gripper open/close events. In simulation, gripper states and grasp relationships define interaction boundaries; for real-robot data, the same events are detected from the smoothed gripper-width signal. Contact and articulated-part motion cues are used when gripper events are insufficient. Bimanual interactions are aligned on a shared timeline, and offline stage-level language annotations are matched to the resulting segments.

4.2 Trajectory Extraction

Simulation. We directly retrieve EEF and object pose trajectories from simulator states. For articulated interactions, we track the manipulated moving part rather than the full object assembly. All trajectories are transformed into a consistent reference frame and aligned with the segmented stages.

Real-world. EEF trajectories are obtained directly from recorded robot poses. For grasped-object stages, we recover object geometry once at a visual anchor and propagate it using the recorded EEF motion, avoiding frame-wise tracking. Following the visual-grounding pipeline of MolmoMotion (Zhang et al., 2026), we combine MolmoPoint localization (Clark et al., 2026) with SAM 3 masks (Carion et al., 2026), and lift the selected region to 3D using estimated depth and camera calibration.

Let T_B^A denote the transform from frame B to frame A, and let a be the anchor time. Assuming rigid object–gripper coupling, the object pose during the grasp interval is reconstructed as

$$\widehat{T}_{O_t}^C = T_{E_t}^C \left(T_{E_a}^C\right)^{-1} \widehat{T}_{O_a}^C, \qquad t \in \mathcal{I}_{\text{coupled}}. \tag{7}$$

Here, $\widehat{T}_{O_a}^C$ is the visually estimated anchor pose, while recorded EEF motion provides its temporal propagation. Frames outside the accepted coupling interval are not extrapolated.

Appendix A details validity checks and failure handling for depth calibration, anchor selection, mask filtering, and reconstruction. We align stage intervals, language annotations, and valid trajectories by their original timestamps to construct sparse point-flow supervision.

5 Experiments

We evaluate PointPilot from three perspectives: (1) general manipulation performance on single-arm and bimanual simulation benchmarks and real-world tasks, (2) zero-shot skill transfer across previously unseen robot embodiments, and (3) the controlled modular ablations to indicate the effectiveness of each module especially the necessity of spatial alignment.

5.1 Simulation Result

Training and evaluation setup. PointPilot is jointly trained on demonstrations from LIBERO and RoboTwin, combining different robot embodiments and single-arm and bimanual manipulation within one model. The results on both benchmarks therefore evaluate a shared policy learned from heterogeneous demonstrations. We follow the standard evaluation protocols of LIBERO and RoboTwin 2.0 Clean and report task success rates.

Single-arm manipulation on LIBERO. As shown in Table 2, PointPilot achieves 85.8% average success across the four LIBERO suites, including the best listed results on LIBERO-Spatial (96.0%) and LIBERO-Object (97.6%). It obtains 77.2% on LIBERO-Long. Performance on LIBERO-Goal is mainly limited by three non-coupled, friction-driven tasks; excluding them, success reaches 96.0%. These results demonstrate competitive single-arm manipulation using object-centric point flow without a learned end-effector action decoder.

Table 2: Success rates (%) on LIBERO. All baseline results follow the evaluation reported by RoboFlow4D (Lin et al., 2026).

Method	Spatial	Object	Goal	Long	Average
Octo (Ghosh et al., 2024)	78.9	85.7	84.6	51.1	75.1
CogACT (Li et al., 2024)	87.5	90.2	78.4	53.2	77.3
OpenVLA (Kim et al., 2025)	84.7	88.4	79.2	53.7	76.5
TraceVLA (Zheng et al., 2024)	84.6	85.2	75.1	54.1	74.8
SpatialVLA (Qu et al., 2025)	88.2	89.9	78.6	55.5	78.1
4D-VLA (Zhang et al., 2025)	88.9	95.2	90.9	**79.1**	88.6
DP w/ RoboFlow4D (Lin et al., 2026)	89.8	93.2	85.2	72.0	85.1
DiT w/ RoboFlow4D (Lin et al., 2026)	90.2	97.0	88.4	75.2	87.7
UniAct (ACT head) (Zheng et al., 2025a)	77.0	87.0	77.0	70.0	77.8
SmolVLA (2.25B) (Shukor et al., 2025)	93.0	94.0	**91.0**	77.0	**88.8**
PointPilot (Ours)	**96.0**	**97.6**	72.4	77.2	85.8

Table 3: Success rates (%) on RoboTwin 2.0. Baseline results are reported by RoboTwin 2.0. The GR00T-N1.7 result is from the Easy setting of RoboTwin-Clean2Rand in Qwen-RobotManip (Yuan et al., 2026).

Method	Success rate (%)
DP	28.0
ACT	29.7
RDT-1B	34.5
GR00T-N1.7	43.6
π_0	46.4
DP3	55.2
PointPilot (Ours)	**56.6**

Bimanual manipulation on RoboTwin. The same checkpoint achieves 56.6% success on RoboTwin 2.0 Clean, compared with the reported 55.2% of DP3. Together with the LIBERO results, this demonstrates that a shared object-centric policy can learn from heterogeneous demonstrations and support both single-arm and bimanual manipulation.

5.2 Zero-Shot Cross-Embodiment Transfer

We evaluate zero-shot transfer from the ALOHA embodiment in RoboTwin to ARX and Piper on the tasks in Table 4. The jointly trained policy is deployed without target-embodiment demonstrations or fine-tuning. Predicted object motion is converted into executable commands through the geometric adapter and each target robot's kinematics and controller.

As shown in Table 4, PointPilot presents a much smaller transfer gap, especially in Piper, compared with X-VLA. Piper differs more visibly from ALOHA than ARX does. Thus, X-VLA exhibits a

Table 4: Cross-embodiment success rates (%) on eight tasks (100 trials per task). For each embodiment, the experiment seeds are the same. X-VLA is tested with its official checkpoint not trained on ARX and Piper. In robotwin, not all tasks are valid in all embodiments. "–" indicates this task is incompatible with this embodiment. * denotes the average and Δ are calculated across 7 tasks—— the alarm-clock task is excluded.

Task	ALOHA		ARX		Piper	
	PointPilot	X-VLA	PointPilot	X-VLA	PointPilot	X-VLA
Grab roller	98.0	100.0	85.0	79.0	79.0	21.0
Shake bottle	100.0	100.0	93.0	90.0	75.0	41.0
Shake bottle horizontally	94.0	99.0	90.0	90.0	83.0	41.0
Click alarm clock	75.0	17.0	69.0	62.0	–	–
Click bell	79.0	99.0	75.0	97.0	77.0	1.0
Dump bin into big bin	100.0	81.0	67.0	29.0	69.0	24.0
Stack two bowls	54.0	62.0	41.0	16.0	39.0	0.0
Pick dual bottles	88.0	89.0	85.0	93.0	75.0	49.0
Average	86.0	80.9	75.6	69.5	71.0*	25.3*
Delta compared with ALOHA (pp)	–	–	**-10.4**	-11.4	**-16.6***	-64.7*

substantially larger transfer gap on Piper, whereas PointPilot maintains comparable performance across the two target embodiments, suggesting that its object-centric 3D motion representation reduces sensitivity to robot appearance.

5.3 Real World Results

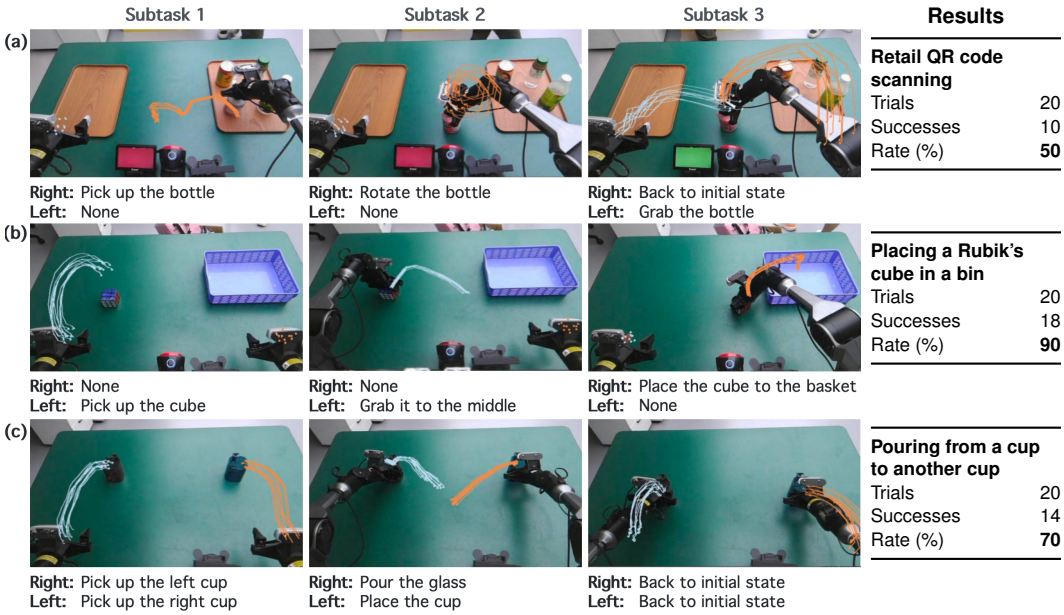

Figure 3: Autonomous closed-loop real-world manipulation with PointPilot. Each row shows successive subtasks and predicted point flows. The rightmost column reports successful trials and success rates over 20 trials per task, with an overall success rate of 70%.

We evaluate PointPilot on three real-world tasks: retail QR code scanning, placing a Rubik's cube into a bin, and pouring from a cup, with 20 trials per task. The policy operates autonomously in a closed loop, replanning from updated observations after each action chunk. Figure 3 reports success rates of 50% (10/20), 90% (18/20) and 70% (14/20) respectively, yielding an overall success rate of 70% (42/60).

The rollouts illustrate distinct demands on task-focused object motion: reorienting a bottle for scanning, sequencing per-arm subtasks for cube placement, and positioning and tilting containers for pouring. These examples connect subtask planning and point-flow prediction to physical execution involving both translation and rotation. Cube placement has the highest observed success rate, while scanning leaves the largest room for improvement. Together, the results demonstrate the

feasibility of the task-focused point-flow interface beyond simulation, while performance remains task-dependent. Rollout videos are provided in the **supplementary material**.

5.4 Ablation Studies

Table 5: Effect of the planning memory bank on LIBERO success rates. Memory consistently improves performance across all four suites, with the largest gain on LIBERO-Long (+22.2 points). Without planning history, the VLM frequently repeats an already executed subtask during consecutive replanning steps, causing the policy to stall.

Suite	w/o Memory	PointPilot	Δ
Spatial	90.4	**96.0**	**+5.6**
Object	89.6	**97.6**	**+8.0**
Goal	65.8	**72.4**	**+6.6**
Long	55.0	**77.2**	**+22.2**
Average	75.2	**85.8**	**+10.6**

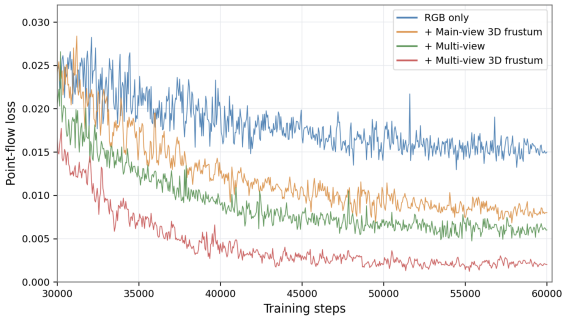

Figure 4: Point-flow loss under different visual-frustum configurations when trained jointly on LIBERO and RoboTwin. Main-view 3D frustum encoding, additional views and only main-view 3D frustum encoding, and multi-view 3D frustum alignment progressively improve convergence.

We ablate the planning memory bank and 3D geometric alignment, measuring their effects on LIBERO task success and point-flow learning, respectively.

Effect of planning memory. As shown in Table 5, removing planning memory lowers average success from 85.8% to 75.2%, with the largest drop on LIBERO-Long (77.2% to 55.0%). The 22.2 percentage-point gap on Long is larger than the 5.6 –8.0 point gaps on the other suites, suggesting that planning history is especially useful for longer subtask sequences. Without history, the VLM can repeat already executed subtasks during replanning and stall execution. Retaining prior plan predictions therefore complements the current visual input with temporal context for subtask selection and stage progression.

Effect of 3D frustum and multi-view alignment. Figure 4 compares visual-geometric configurations under joint LIBERO and RoboTwin training. Adding main-view 3D frustum encoding reduces the final point-flow loss from 0.015 to 0.008, and auxiliary views further reduce it to 0.006. Applying spatially aligned 3D frustum encoding to all views reaches 0.002. In particular, loss decreases further after adding aligned geometric encoding even when multi-view images are already available. This suggests that additional views and geometric alignment make complementary contributions. The results support expressing visual geometry in the same coordinate frame as predicted point motion, rather than relying solely on multi-view appearance.

6 Conclusion and Future Work

PointPilot represents subtask-focused objects with sparse 3D points and execution roles, providing a shared interface across supported manipulation modes and embodiments. Memory-conditioned subtask planning and grounding connect task intent to point-flow prediction, with multi-view geometry and predicted motion aligned in the main-view coordinate. Under stage-local rigid coupling, a non-learned geometric adapter converts predicted flows into end-effector motion without an additional learned continuous-action decoder.

Several limitations suggest directions for further research. The current formulation centers on locally rigid task-focused objects and gripper-based execution. As a scalable object-centric framework, we need to leverage more data (including human hands and dexterous hands) to prove the scalability and show the generalization. Moreover, prediction focuses on the task-focused objects selected for the current stage rather than explicitly forecasting the broader set of task-relevant entities. Jointly modeling manipulated objects and other entities affected by their motion could support richer reasoning about object–object interactions and indirect effects while retaining a sparse, task-focused representation. Finally, reducing the burden of aligning stage semantics, object geometry, execution roles, and motion while preserving semantic–geometric consistency remains a challenge for scaling.

AI USE STATEMENT

We used generative AI tools to assist with initial manuscript editing and literature discovery. AI suggestions informed the development process, while the authors made the final methodological and experimental decisions. We take responsibility for the final content of this work, including all text, claims, code, and artifacts produced with the assistance of generative AI.

ETHICS STATEMENT

PointPilot studies autonomous robotic manipulation, whose deployment can pose physical risks when perception, planning, or execution fails. Our benchmark and real-world evaluations assess performance in controlled settings and do not establish safety for unrestricted deployment, particularly around people or fragile objects. Deployment beyond these settings requires appropriate supervision, collision safeguards, and task-specific risk assessment. Any release and reuse of data or models should respect the licenses and usage conditions of the underlying resources.

REPRODUCIBILITY STATEMENT

The main paper describes the PointPilot pipeline, including subtask planning, language-guided 3D grounding, point-flow prediction, and geometric conversion to robot commands. The appendix provides details of the data pipeline and implementation. The experimental section specifies the evaluation protocols, robot embodiments, and success metrics. Supplementary videos illustrate autonomous closed-loop execution on the real-world tasks. Together, these materials document the model and its evaluation to support reproduction of the reported results.

REFERENCES

Homanga Bharadhwaj, Roozbeh Mottaghi, Abhinav Gupta, and Shubham Tulsiani. Track2Act: Predicting point tracks from internet videos enables generalizable robot manipulation. In *European Conference on Computer Vision (ECCV)*, 2024. URL https://homangab.github.io/track2act/.

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.

Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. Learning to Act Anywhere with Task-centric Latent Actions. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.014. URL https://www.roboticsproceedings.org/rss21/p014.html.

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, Jie Lei, Tengyu Ma, Baishan Guo, Arpit Kalla, Markus Marks, Joseph Greer, Meng Wang, Peize Sun, Roman Rädle, Triantafyllos Afouras, Effrosyni Mavroudi, Katherine Xu, Tsung-Han Wu, Yu Zhou, Liliane Momeni, Rishi Hazra, Shuangrui Ding, Sagar Vaze, Francois Porcher, Feng Li, Siyuan Li, Aishwarya Kamath, Ho Kei Cheng, Piotr Dollár, Nikhila Ravi, Kate Saenko, Pengchuan Zhang, and Christoph Feichtenhofer. SAM 3: Segment anything with concepts. arXiv preprint arXiv:2511.16719, 2026. URL https://arxiv.org/abs/2511.16719.

Christopher Clark, Yue Yang, Jae Sung Park, Zixian Ma, Jieyu Zhang, Rohun Tripathi, Mohammadreza Salehi, Sangho Lee, Taira Anderson, Winson Han, and Ranjay Krishna. MolmoPoint: Better pointing for VLMs with grounding tokens. arXiv preprint arXiv:2603.28069, 2026. URL https://arxiv.org/abs/2603.28069.

Ria Doshi, Homer Rich Walke, Oier Mees, Sudeep Dasari, and Sergey Levine. Scaling cross-embodied learning: One policy for manipulation, navigation, locomotion and aviation. 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. 496–512. PMLR, 06–09 Nov 2025. URL https://proceedings.mlr.press/v270/doshi25a.html.

Jiafei Duan, Wentao Yuan, Wilbert Pumacay, Yi Ru Wang, Kiana Ehsani, Dieter Fox, and Ranjay Krishna. Manipulate-Anything: Automating real-world robots using vision-language models. *arXiv preprint arXiv:2406.18915*, 2024. doi: 10.48550/arXiv.2406.18915. URL https://arxiv.org/abs/2406.18915.

Ben Eisner, Harry Zhang, and David Held. FlowBot3D: Learning 3D Articulation Flow to Manipulate Articulated Objects. In *Proceedings of Robotics: Science and Systems*, New York City, NY, USA, June 2022. doi: 10.15607/RSS.2022.XVIII.018. URL https://www.roboticsproceedings.org/rss18/p018.html.

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.

Haoxu Huang, Fanqi Lin, Yingdong Hu, Shengjie Wang, and Yang Gao. CoPa: General robotic manipulation through spatial constraints of parts with foundation models. *arXiv preprint arXiv:2403.08248*, 2024a. doi: 10.48550/arXiv.2403.08248. URL https://arxiv.org/abs/2403.08248.

Jiahui Huang, Qunjie Zhou, Hesam Rabeti, Aleksandr Korovko, Huan Ling, Xuanchi Ren, Tianchang Shen, Jun Gao, Dmitry Slepichev, Chen-Hsuan Lin, Jiawei Ren, Kevin Xie, Joydeep Biswas, Laura Leal-Taixe, and Sanja Fidler. ViPE: Video pose engine for 3D geometric perception. NVIDIA Research Whitepapers, 2025a. URL https://research.nvidia.com/labs/toronto-ai/vipe/.

Suning Huang, Qianzhong Chen, Xiaohan Zhang, Jiankai Sun, and Mac Schwager. ParticleFormer: A 3D point cloud world model for multi-object, multi-material robotic manipulation. In Joseph Lim, Shuran Song, and Hae-Won Park (eds.), *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 4941–4957. PMLR, 27–30 Sep 2025b. URL https://proceedings.mlr.press/v305/huang25c.html.

Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. ReKep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. *arXiv preprint arXiv:2409.01652*, 2024b. doi: 10.48550/arXiv.2409.01652. URL https://arxiv.org/abs/2409.01652.

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.

Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, Xiaofan Wang, Bei Liu, Jianlong Fu, Jianmin Bao, Dong Chen, Yuanchun Shi, Jiaolong Yang, and Baining Guo. CogACT: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. arXiv preprint arXiv:2411.19650, 2024. URL https://arxiv.org/abs/2411.19650.

Sixu Lin, Junliang Chen, Huaiyuan Xu, Zhuohao Li, Guangming Wang, Yixiong Jing, Sheng Xu, Runyi Zhao, Brian Sheil, Lap-Pui Chau, and Guiliang Liu. RoboFlow4D: A lightweight

flow world model toward real-time flow-guided robotic manipulation. In *International Conference on Machine Learning (ICML)*, 2026. URL https://simonlinsx.github.io/RoboFlow4D_Page/#citation.

Yingfei Liu, Tiancai Wang, Xiangyu Zhang, and Jian Sun. Petr: Position embedding transformation for multi-view 3d object detection, 2022a. URL https://arxiv.org/abs/2203.05625.

Yingfei Liu, Junjie Yan, Fan Jia, Shuailin Li, Aqi Gao, Tiancai Wang, Xiangyu Zhang, and Jian Sun. Petrv2: A unified framework for 3d perception from multi-camera images, 2022b. URL https://arxiv.org/abs/2206.01256.

Dekun Lu, Wei Gao, and Kui Jia. ImaginationPolicy: Towards generalizable, precise and reliable end-to-end policy for robotic manipulation. arXiv preprint arXiv:2509.20841, 2025. URL https://arxiv.org/abs/2509.20841.

Open X-Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhinav Gupta, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta, Andrew Wang, Andrey Kolobov, Anikait Singh, Animesh Garg, Aniruddha Kembhavi, Annie Xie, Anthony Brohan, Antonin Raffin, Archit Sharma, Arefeh Yavary, Arhan Jain, Ashwin Balakrishna, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Blake Wulfe, Brian Ichter, Cewu Lu, Charles Xu, Charlotte Le, Chelsea Finn, Chen Wang, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Christopher Agia, Chuer Pan, Chuyuan Fu, Coline Devin, Danfei Xu, Daniel Morton, Danny Driess, Daphne Chen, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dinesh Jayaraman, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Ethan Foster, Fangchen Liu, Federico Ceola, Fei Xia, Feiyu Zhao, Felipe Vieira Frujeri, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Gilbert Feng, Giulio Schiavi, Glen Berseth, Gregory Kahn, Guangwen Yang, Guanzhi Wang, Hao Su, Hao-Shu Fang, Haochen Shi, Henghui Bao, Heni Ben Amor, Henrik I Christensen, Hiroki Furuta, Homanga Bharadhwaj, Homer Walke, Hongjie Fang, Huy Ha, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jad Abou-Chakra, Jaehyung Kim, Jaimyn Drake, Jan Peters, Jan Schneider, Jasmine Hsu, Jay Vakil, Jeannette Bohg, Jeffrey Bingham, Jeffrey Wu, Jensen Gao, Jiaheng Hu, Jiajun Wu, Jialin Wu, Jiankai Sun, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jimmy Wu, Jingpei Lu, Jingyun Yang, Jitendra Malik, João Silvério, Joey Hejna, Jonathan Booher, Jonathan Tompson, Jonathan Yang, Jordi Salvador, Joseph J. Lim, Junhyek Han, Kaiyuan Wang, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Black, Kevin Lin, Kevin Zhang, Kiana Ehsani, Kiran Lekkala, Kirsty Ellis, Krishan Rana, Krishnan Srinivasan, Kuan Fang, Kunal Pratap Singh, Kuo-Hao Zeng, Kyle Hatch, Kyle Hsu, Laurent Itti, Lawrence Yunliang Chen, Lerrel Pinto, Li Fei-Fei, Liam Tan, Linxi "Jim" Fan, Lionel Ott, Lisa Lee, Luca Weihs, Magnum Chen, Marion Lepert, Marius Memmel, Masayoshi Tomizuka, Masha Itkina, Mateo Guaman Castro, Max Spero, Maximilian Du, Michael Ahn, Michael C. Yip, Mingtong Zhang, Mingyu Ding, Minho Heo, Mohan Kumar Srirama, Mohit Sharma, Moo Jin Kim, Muhammad Zubair Irshad, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J Joshi, Niko Suenderhauf, Ning Liu, Norman Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Osbert Bastani, Pannag R Sanketi, Patrick "Tree" Miller, Patrick Yin, Paul Wohlhart, Peng Xu, Peter David Fagan, Peter Mitrano, Pierre Sermanet, Pieter Abbeel, Priya Sundaresan, Qiuyu Chen, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Mart'in-Mart'in, Rohan Baijal, Rosario Scalise, Rose Hendrix, Roy Lin, Runjia Qian, Ruohan Zhang, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Shan Lin, Sherry Moore, Shikhar Bahl, Shivin Dass, Shubham Sonawani, Shubham Tulsiani, Shuran Song, Sichun Xu, Siddhant Haldar, Siddharth Karamcheti, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Subramanian Ramamoorthy, Sudeep Dasari, Suneel Belkhale, Sungjae Park, Suraj Nair, Suvir Mirchandani, Takayuki Osa, Tanmay Gupta, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Thomas Kollar, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Trinity Chung, Vidhi Jain, Vikash Kumar, Vincent Vanhoucke, Vitor Guizilini, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiangyu Chen, Xiaolong Wang, Xinghao Zhu, Xinyang Geng, Xiyuan Liu, Xu Liangwei, Xuanlin Li, Yansong Pang, Yao Lu, Yecheng Jason Ma, Yejin Kim, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Yilin Wu, Ying Xu, Yixuan Wang, Yonatan

Bisk, Yongqiang Dou, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yue Cao, Yueh-Hua Wu, Yujin Tang, Yuke Zhu, Yunchu Zhang, Yunfan Jiang, Yunshuang Li, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zehan Ma, Zhuo Xu, Zichen Jeff Cui, Zichen Zhang, Zipeng Fu, and Zipeng Lin. Open X-Embodiment: Robotic learning datasets and RT-X models. arXiv preprint arXiv:2310.08864, 2023. URL https://arxiv.org/abs/2310.08864.

Mingjie Pan, Jiyao Zhang, Tianshu Wu, Yinghao Zhao, Wenlong Gao, and Hao Dong. OmniManip: Towards general robotic manipulation via object-centric interaction primitives as spatial constraints. *arXiv preprint arXiv:2501.03841*, 2025. doi: 10.48550/arXiv.2501.03841. URL https://arxiv.org/abs/2501.03841.

Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Jiayuan Gu, Zhigang Wang, Yan Ding, Bin Zhao, Dong Wang, and Xuelong Li. SpatialVLA: Exploring Spatial Representations for Visual-Language-Action Models. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.011. URL https://www.roboticsproceedings.org/rss21/p011.html.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, Simon Alibert, Matthieu Cord, Thomas Wolf, and Remi Cadene. SmolVLA: A vision-language-action model for affordable and efficient robotics. arXiv preprint arXiv:2506.01844, 2025. URL https://arxiv.org/abs/2506.01844.

Weiliang Tang, Jia Hui Pan, Wei Zhan, Jianshu Zhou, Huaxiu Yao, Yun Hui Liu, Masayoshi Tomizuka, Mingyu Ding, and Chi Wing Fu. Embodiment-agnostic action planning via object-part scene flow. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, Proceedings - IEEE International Conference on Robotics and Automation, pp. 2086–2093, United States, 2025. Institute of Electrical and Electronics Engineers Inc. doi: 10.1109/ICRA55743.2025.11127873. URL https://research.cuhk.edu.hk/en/publications/embodiment-agnostic-action-planning-via-object-part-scene-flow/.

Lirui Wang, Xinlei Chen, Jialiang Zhao, and Kaiming He. Scaling proprioceptive-visual learning with heterogeneous pre-trained transformers. In *Advances in Neural Information Processing Systems*, 2024. URL https://liruiw.github.io/hpt/.

Mengda Xu, Zhenjia Xu, Yinghao Xu, Cheng Chi, Gordon Wetzstein, Manuela Veloso, and Shuran Song. Flow as the cross-domain manipulation interface. 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. 2475–2499. PMLR, 06–09 Nov 2025. URL https://proceedings.mlr.press/v270/xu25a.html.

Yuyin Yang, Zetao Cai, Yang Tian, Jia Zeng, and Jiangmiao Pang. Gripper Pose and Object Pointflow as Interfaces for Robotic Bimanual Manipulation. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.160. URL https://www.roboticsproceedings.org/rss21/p160.html.

Zhao-Heng Yin, Sherry Yang, and Pieter Abbeel. Object-centric 3D motion field for robot learning from human videos. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-1872. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/50f91c82158baed92e9b7e5491a6d997-Abstract-Conference.html.

Chengbo Yuan, Chuan Wen, Tong Zhang, and Yang Gao. General flow as foundation affordance for scalable robot learning. 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. 1541–1566. PMLR, 06–09 Nov 2025. URL https://proceedings.mlr.press/v270/yuan25a.html.

Haoqi Yuan, Zhixuan Liang, Anzhe Chen, Ye Wang, Haoyang Li, Pei Lin, Yiyang Huang, Zixing Lei, Tong Zhang, Jiazhao Zhang, Jie Zhang, Jingyang Fan, Gengze Zhou, Qihang Peng, Chenxu Lv, Xiaoyue Chen, An Yang, Fei Huang, Junyang Lin, Dayiheng Liu, Jingren Zhou, Chenfei Wu, and Xiong-Hui Chen. Qwen-RobotManip technical report: Alignment unlocks scale for

robotic manipulation foundation models. arXiv preprint arXiv:2606.17846, 2026. URL https://arxiv.org/abs/2606.17846.

Jiahui Zhang, Yurui Chen, Yueming Xu, Ze Huang, Yanpeng Zhou, Yujie Yuan, Xinyue Cai, Guowei Huang, Xingyue Quan, Hang Xu, and Li Zhang. 4D-VLA: Spatiotemporal vision-language-action pretraining with cross-scene calibration. *arXiv preprint arXiv:2506.22242*, 2025. URL https://arxiv.org/abs/2506.22242.

Jianing Zhang, Chenhao Zheng, Yajun Yang, Max Argus, Rustin Soraki, Winson Han, Taira Anderson, Chun-Liang Li, Shuo Liu, Jiafei Duan, Zhongzheng Ren, Jieyu Zhang, and Ranjay Krishna. MolmoMotion: Forecasting point trajectories in 3D with language instruction. *arXiv preprint arXiv:2606.18558*, 2026. URL https://arxiv.org/abs/2606.18558.

Jinliang Zheng, Jianxiong Li, Dongxiu Liu, Yinan Zheng, Zhihao Wang, Zhonghong Ou, Yu Liu, Jingjing Liu, Ya-Qin Zhang, and Xianyuan Zhan. Universal actions for enhanced embodied foundation models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22508–22519, June 2025a. URL https://openaccess.thecvf.com/content/CVPR2025/html/Zheng_Universal_Actions_for_Enhanced_Embodied_Foundation_Models_CVPR_2025_paper.html.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, Ya-Qin Zhang, Jiangmiao Pang, Jingjing Liu, Tai Wang, and Xianyuan Zhan. X-VLA: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. arXiv preprint arXiv:2510.10274, 2025b. URL https://arxiv.org/abs/2510.10274.

Ruijie Zheng, Yongyuan Liang, Shuaiyi Huang, Jianfeng Gao, Hal Daumé, III, Andrey Kolobov, Furong Huang, and Jianwei Yang. TraceVLA: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies. *arXiv preprint arXiv:2412.10345*, 2024. URL https://arxiv.org/abs/2412.10345.

Hongyan Zhi, Peihao Chen, Siyuan Zhou, Yubo Dong, Quanxi Wu, Lei Han, and Mingkui Tan. 3DFlowAction: Learning cross-embodiment manipulation from 3D flow world model. arXiv preprint arXiv:2506.06199, 2025. URL https://arxiv.org/abs/2506.06199.

Huayi Zhou, Wei Gao, Dekun Lu, Ruiji Liu, Zhanqi Zhang, Ziyang Zhang, Jian Chen, Wenlve Zhou, Sheng Xu, Shumin Li, Kangyi Guo, Shichen Xu, Zixin Huang, Yongyi Su, and Kui Jia. Dexterity-bev: Aligning 3d world and actions for generalizable robot policies learning, 2026. URL https://arxiv.org/abs/2606.02274.

A Data Construction Details

This appendix expands the interaction-stage segmentation and 3D trajectory extraction used to construct task-focused point-flow supervision for PointPilot. We briefly summarize simulation processing and focus on gripper-event detection and geometric reconstruction for our self-collected real-robot recordings. The output consists of stage-aligned entity poses and geometry, together with their temporal correspondences, validity information, and reconstruction status.

A.1 Stage Segmentation

Gripper opening and closing events provide the primary temporal anchors for segmentation. Simulation supplies recorded gripper-state changes; for our real-robot recordings, we recover the corresponding events from the continuous gripper-width signal. We first apply median smoothing and compute the signed numerical gradient with respect to the recorded timestamps. Let s_i denote this gradient at sample i. The automatic event threshold is

$$\tau_s = \max(\tau_{\min}, Q_{0.9}(\{|s_i|\})), \tag{8}$$

where $Q_{0.9}$ denotes the 90th percentile. Contiguous intervals exceeding this magnitude threshold and satisfying a minimum duration in source frames form candidate events. Their direction is determined by the median signed slope: negative for closing and positive for opening.

Nearby events with the same direction are merged. A candidate is accepted only if it agrees with the current open/closed state, reaches the opposite-state plateau, and produces a sufficient change in width. Accepted closing events place a boundary at the end of closure, whereas accepted opening events place one at the onset of opening. The resulting intervals indicate candidate grasp states rather than independently certifying object attachment. Bimanual intervals are synchronized while retaining per-arm information. Simulation additionally uses contact and moving-part cues for interactions not adequately captured by gripper events, such as drawer manipulation. Stage-level language annotations are obtained through a separate offline procedure and associated with the resulting intervals.

A.2 Trajectory Extraction

For simulated demonstrations, we associate recorded object, moving-part, and end-effector (EEF) poses with the corresponding entities and stage intervals, as described in the main paper. Each pose sequence is paired with its entity geometry in a consistent reference frame. We focus below on real-robot reconstruction, where object trajectories must be recovered from visual observations and robot motion rather than directly read from simulator states.

Depth Calibration and Camera Geometry. For our self-collected real-robot recordings, we estimate depth with ViPE (Huang et al., 2025a) on sparse frames associated with grasped-object stages. These depth estimates are used for offline geometric reconstruction, not as inputs to PointPilot at inference. Reconstruction uses the supplied fixed-camera intrinsics and robot-to-camera transforms rather than ViPE's estimated camera parameters. Estimated depth can have substantial scale and offset errors relative to the robot reference. Even with a correct object mask, directly back-projecting this depth can misplace the object relative to the gripper and distort its reconstructed size. These errors affect the object-to-EEF transform used to propagate object motion. We therefore calibrate the estimated depth against recorded EEF positions before reconstructing the visual anchor.

To construct calibration references, we use the supplied camera calibration to express recorded EEF positions in the camera coordinate frame. Their optical-axis coordinates z_i^E provide the robot-derived depth references. Using the camera intrinsics, we project each reference point into the corresponding depth image at (u_i, v_i). We then compute d_i, the median of finite, positive estimated depths in a 15×15 pixel patch centered on this projection. This gives a pair (d_i, z_i^E) linking an image-based depth estimate to the robot reference. Usable pairs from both arms and multiple sampled times are pooled to estimate a shared correction.

We model the systematic discrepancy as $z_i^E \approx \alpha d_i + \beta$, where $\alpha > 0$ adjusts depth scale and β adjusts offset. RANSAC identifies a positive-scale relation supported by multiple observations

and rejects pairs with large residuals. Least-squares fitting on the current inliers then refines the parameters, with the inlier set updated from the residuals. The resulting correction is applied to the corresponding depth maps:

$$D_{\text{cal}}(u, v) = \alpha D_{\text{est}}(u, v) + \beta. \tag{9}$$

The same fitted parameters are applied to all associated frames, rather than fitting each pixel or frame independently. Although the references are sampled near the EEF projections, the correction is applied to the depth maps used for object reconstruction, not only to those local patches.

We accept a fit only when it has sufficient supporting observations, adequate depth variation and correlation, and a scale within the allowed range. If these references are insufficient for a supported fit, the designated failure path retains the supplied camera geometry and uncalibrated depth with an explicit calibration-failure status. The EEF frame origin need not lie on a visible gripper surface, so the sampled patch provides an approximate correspondence rather than an exact depth measurement of that point. The calibration therefore constrains local consistency with the robot reference without establishing scene-wide depth ground truth or an object-trajectory error bound.

Visual Anchor Selection. For a grasped-object stage, we search for a visual anchor within the candidate grasp interval. Adapting the visual-grounding components of MolmoMotion (Zhang et al., 2026), we use MolmoPoint (Clark et al., 2026) to obtain image-plane location cues and SAM 3 (Carion et al., 2026) to generate candidate masks. Localization specifies the manipulated object and the corresponding arm. MolmoPoint processes each candidate image as a short repeated-frame clip; its locations guide selection from text-conditioned SAM 3 masks. When no suitable text-conditioned mask is matched, a small box around a predicted location provides an additional segmentation prompt. These components localize candidate anchors rather than track the object throughout the stage.

We evaluate up to five candidate frames and separate acceptance from ranking. A supported candidate must meet foreground-area and mask-to-EEF proximity thresholds. A stage requires at least $\min(3, N)$ supported candidates, where N is the number examined. Among supported candidates, we select the highest-scoring anchor using

$$Q = 0.25Q_{\text{area}} + 0.40Q_{\text{depth}} + 0.15Q_{\text{boundary}} + 0.20Q_{\text{proximity}}, \tag{10}$$

where the terms quantify mask area, valid-depth coverage, image-boundary margin, and proximity to the projected EEF of the corresponding arm. Support across candidate frames is used as a consistency check; only the selected anchor is used to recover object geometry. We do not fuse multiple anchor poses or interpret the score as a calibrated probability of reconstruction accuracy.

Object Geometry and Motion Reconstruction. Following the reconstruction convention in the main paper, we express reconstructed object geometry in the supplied fixed-camera frame C. At the selected anchor time a, valid masked depth pixels are back-projected using the supplied intrinsics:

$$\mathbf{p}_a^C(u, v) = D_a(u, v) \begin{bmatrix} (u - c_x)/f_x \\ (v - c_y)/f_y \\ 1 \end{bmatrix}, \tag{11}$$

where D_a is the accepted depth map, using the affine correction when available. We cap the point cloud at 30,000 samples, remove the most distant 10% relative to its coordinate-wise median, and compute a right-handed PCA frame. The 2nd and 98th percentiles along each axis determine a robust box center and extent, with a minimum extent for nearly degenerate dimensions. The resulting anchor pose $\widehat{T}_{O_a}^C$ and oriented bounding box (OBB) approximate the visible object region rather than a complete CAD-aligned object model.

We recover temporal object motion through the fixed relative transform described in the main paper:

$$\widehat{T}_O^E = \left(T_{E_a}^C\right)^{-1} \widehat{T}_{O_a}^C, \qquad \widehat{T}_{O_t}^C = T_{E_t}^C \widehat{T}_O^E, \quad t \in \mathcal{I}_{\text{coupled}}. \tag{12}$$

The same transformation propagates the object-local OBB vertices. Thus, visual reconstruction estimates the anchor geometry once, while recorded EEF poses provide the subsequent motion. This construction assumes approximately rigid attachment without appreciable slip within $\mathcal{I}_{\text{coupled}}$; the existence of a stage alone does not establish that assumption. Outside the reconstruction interval, including after release, object poses and vertices are marked unavailable rather than propagated with the former gripper attachment. Errors in anchor geometry, camera alignment, and the attachment assumption can still affect the reconstructed trajectory.

Gripper Geometry and Fallback Handling. We construct a sequence of gripper 3D bounding box from URDF collision geometry, recorded joint states, and EEF poses. The joint states determine the gripper configuration, while the EEF poses place the geometry in the scene. Before combining these sources, we check that the logged EEF poses are consistent with forward-kinematic flange poses under the supplied reference-frame mapping. The resulting bounding boxes describe the gripper's motion and opening without requiring image-based reconstruction.

When object localization or geometric reconstruction fails to produce an accepted object trajectory, we fall back to the corresponding gripper trajectory and OBB derived from recorded EEF poses and robot geometry. Fallback cases are explicitly flagged and treated as gripper geometry rather than successfully reconstructed object motion.

Finally, stage intervals, language annotations, and geometric outputs are associated through source-frame indices and timestamps. Reconstruction validity and fallback provenance are retained separately. The stored object masks correspond to selected anchors rather than dense per-frame segmentation, and the sparse geometry timeline is not assumed to be uniformly sampled. These records provide pose sequences and validity information for constructing PointPilot's point-flow supervision for task-focused objects. The prediction targets are future 3D point displacements relative to their current positions, as defined in the main paper.

B More Details about PointPilot

To those who want to reproduce our work, we detail the computation about PointPilot in this section. At planning anchor t, all boxes and point trajectories use the current main-camera frame c_t. The world-to-camera extrinsic is $\mathbf{E}_t$; bars denote homogeneous coordinates, with $\bar{\mathbf{p}}^{c_t} = \mathbf{E}_t \bar{\mathbf{p}}^{w}$. Both the single-arm and bi-manual configuration use horizon $H = 30$, at most $M = 2$ objects, $N = 30$ points per object, and two grippers. Dimensions and translations are in meters. The future action index $h \in \{1, \ldots, H\}$ is distinct from flow-matching time $\tau \in [0, 1]$. Depth is not needed in PointPilot.

B.1 Subtask Planning

Prompt and task features. The VLM receives the current main-view image and the following prompt; line wrapping is for presentation. No historical image is appended.

```
Instruction: {instruction}
Previous subtasks:
{previous_subtasks}
Predict the current executable left/right subtask
pair and the immediate next distinct pair that
begins within the next {horizon} actions.
Use NONE for an inactive hand. Return exactly:
<CURRENT>
Left: ...
Right: ...
</CURRENT>
<NEXT>
Left: ...
Right: ...
</NEXT>
If no distinct stage begins in the horizon,
return <NEXT> NONE </NEXT>.
```

The four delimiters are atomic special tokens. Only the hidden states inside `CURRENT` are passed downstream. Training uses teacher-forced answers; inference generates an answer and re-encodes it with the same image and prompt to extract these features. `NEXT` contributes language supervision and subsequent memory, but is not a downstream condition.

Labels and memory. Let stage j have interval $[b_j, e_j)$ and canonical left/right label s_j. For effective current stage k, CURRENT is s_k; NEXT is the first later distinct label whose start satisfies $t < b_j \le t + H$, or `NONE`. Instruction paraphrases are sampled from the first at most ten stored candidates; stage labels remain canonical. Training memory is empty for $t < b_0 + H$ and otherwise contains $(s_0, \dots, s_k)$ after adjacent deduplication, including the current stage. During rollout, memory instead starts empty and appends predicted CURRENT followed by NEXT, skipping empty/NONE entries and repetitions of its last item. Training history is constructed from GT annotations, not from VLM rollouts; the two history distributions are not identical.

B.2 3D GROUNDING

Main-view geometry encoding. After Qwen's 2×2 merge of 14×14 ViT patches, each output token corresponds to a 28×28 region of the resized image. For source size (H_0, W_0) and resized size (H', W'), use $\widetilde{\mathbf{K}} = \mathrm{diag}(W'/W_0, H'/H_0, 1)\mathbf{K}$. On the merged grid (h_f, w_f), location (i, j) uses $\tilde{\mathbf{u}}_{ij} = [jW'/w_f, iH'/h_f, 1]^\top$, without a half-cell offset. Its $D = 64$ depth hypotheses are

$$\begin{aligned} z_d &= z_{\min} + \frac{z_{\max} - z_{\min}}{D(D+1)} d(d+1), \\ \mathbf{x}^c_{ij,d} &= z_d \widetilde{\mathbf{K}}^{-1} \tilde{\mathbf{u}}_{ij}, \qquad d = 0, \dots, D-1. \end{aligned} \tag{13}$$

We use $z_{\min} = 0.01$, $z_{\max} = 1.60$, and main-frame workspace bounds $\mathbf{a} = (-1.10, -0.50, 0.00)$, $\mathbf{b} = (2.25, 1.10, 1.60)$. Define $\ell_\epsilon(x) = \log[\max(x, \epsilon)/\max(1 - x, \epsilon)]$ after clipping x to $[0, 1]$, with $\epsilon = 10^{-5}$. The per-token position encoder is

$$\mathbf{e}^{3D}_{ij} = f_{\mathrm{PE}}\left(\mathrm{vec}_{d,xyz}\left[\ell_\epsilon\left(\frac{\mathbf{x}^c_{ij,d} - \mathbf{a}}{\mathbf{b} - \mathbf{a}}\right)\right]\right). \tag{14}$$

Division is elementwise. The shared per-location MLP is implemented as two 1×1 convolutions, $192 \to 8192 \to 2048$, with ReLU between them. These are feature-grid encodings, not measured depth. Out-of-workspace coordinates are clipped, but do not mask valid RGB tokens.

With final-layer VLM image features $\mathbf{H}^{\mathrm{img}}$, CURRENT features $\mathbf{H}^{\mathrm{cur}}$, and $\phi = \mathrm{Linear} \circ \mathrm{LN}$,

$$\mathbf{Z}^{\mathrm{img}} = \phi(\mathbf{H}^{\mathrm{img}} + \mathbf{e}^{3D}), \quad \mathbf{Z}^{\mathrm{task}} = \phi(\mathbf{H}^{\mathrm{cur}}), \quad \mathbf{Z}^{\mathrm{ground}} = [\mathbf{Z}^{\mathrm{img}}; \mathbf{Z}^{\mathrm{task}}]. \tag{15}$$

Here ϕ maps 2048 to 512. Both grounding keys and values use this memory; it contains neither auxiliary-view tokens nor NEXT tokens.

Queries and predictions. $Q = 2$ learned 3D references $\mathbf{r}_q$, initialized uniformly in $[0, 1]^3$, provide query positions through a 3D sine/cosine encoding and an MLP. Query content starts at zero. Six 512-wide decoder layers apply self-attention with position added to query/key, cross-attention to $\mathbf{Z}^{\mathrm{ground}}$, and a feed-forward block, each with a residual connection and post-LayerNorm. Query position is also added to the cross-attention query. Each layer has eight attention heads and FFN width 2048. Only the final normalized queries are supervised.

The box head outputs $(\mathbf{o}_c, \mathbf{o}_s, \mathbf{o}_r)$ of widths $(3, 3, 6)$, decoded as

$$\begin{aligned} \hat{\mathbf{c}}_q &= \mathbf{a} + (\mathbf{b} - \mathbf{a}) \odot \sigma(\mathbf{o}_{c,q} + \ell_\epsilon(\mathbf{r}_q)), \\ \hat{\boldsymbol{\eta}}_q &= \exp(\mathbf{o}_{s,q}), \qquad \hat{\mathbf{R}}_q = \mathrm{GS}_{\mathrm{rows}}(\mathbf{o}_{r,q}). \end{aligned} \tag{16}$$

$\mathrm{GS}_{\mathrm{rows}}$ normalizes the first 3-vector, orthogonalizes and normalizes the second, and uses their cross product as the third row. $\boldsymbol{\eta}$ denotes full side lengths. The box MLP is $512 \to 512 \to 512 \to 12$ with ReLU hidden layers. A separate $512 \to 512 \to 512 \to 4$ head, with LayerNorm/ReLU hidden layers, predicts probabilities $\pi_{q,r}$ over invalid, left, right, and shared roles. Inference maximizes total query log-probability over feasible active-role sets $\varnothing, \{L\}, \{R\}, \{S\}, \{L, R\}$; unused queries are invalid. Slot 0 holds a left/shared object and slot 1 a right-exclusive object.

B.3 Multi-view Geometric Alignment

The main and wrist views share one Qwen visual encoder and merger, but only the main view enters the language transformer. For auxiliary view v, construct rays using its own resized intrinsic matrix as in Eq. equation 13. Given current world-to-camera extrinsics $\mathbf{E}_{v,t}$ and $\mathbf{E}_t$, align their metric coordinates by

$$\bar{\mathbf{x}}^{c_t}_{v,i,d} = \mathbf{E}_t \mathbf{E}^{-1}_{v,t} \bar{\mathbf{x}}^{c_v}_{v,i,d}. \tag{17}$$

The transform precedes workspace normalization and uses the current anchor's wrist calibration, not a fixed initial pose. The transformed coordinates pass through the *same* position encoder in Eq. equation 14, with the same main-frame bounds. For visual-merger output $\mathbf{F}_{v,i} \in \mathbb{R}^{2048}$,

$$\mathbf{Z}^{\text{aux}}_{v,i} = \psi(\mathbf{F}_{v,i} + \mathbf{e}^{3D}_{v,i}), \qquad \psi = \text{Linear}_{2048\to512} \circ \text{LN}. \tag{18}$$

ψ is shared across auxiliary views but separate from ϕ. No view-ID embedding is added. Unlike main-view fusion, auxiliary fusion uses visual-merger features without a language-transformer pass. These tokens are appended only to DiT conditions, not to grounding memory. Patch counts follow each view's actual processor grid; padded tokens and missing views are masked. This geometric branch requires per-view intrinsics and extrinsics, including moving-wrist calibration.

B.4 Point Flow DiT

Anchors and object features. For each selected predicted OBB, deterministic Halton coordinates $\mathbf{u}_n \in [-\frac{1}{2}, \frac{1}{2}]^3$ use bases $(2, 3, 5)$ and indices $n = 1, \ldots, N$. Anchors are

$$\mathbf{p}_{0,m,n} = \hat{\mathbf{R}}_m(\hat{\boldsymbol{\eta}}_m \odot \mathbf{u}_n) + \hat{\mathbf{c}}_m. \tag{19}$$

This is volume sampling, not FPS or depth-surface sampling. Project the eight OBB corners with the main intrinsic, take their pixel extrema, and clip the resulting ROI to the image. All corners must lie in front of the camera; otherwise that downstream object slot is invalid. The object token is

$$\mathbf{Z}^{\text{obj}}_m = f_o\left[f_{\text{PN}}(\mathbf{P}_{0,m});\ f_{\text{box}}(\mathbf{b}^{2D}_m);\ f_{\text{ROI}}(\mathbf{H}^{\text{img}}, \mathbf{b}^{2D}_m)\right]. \tag{20}$$

PointNet max-pools a 1024-wide geometry feature; the image-size-normalized 2D box gives 256 features. ROIAlign extracts a 2×2 crop from raw main-view VLM image features *before* PETR addition, then maps it to 512 features. Their 1792-wide concatenation is projected to 512 dimensions. The grounding-query hidden vector is not added to this token.

Point and gripper tokens. For noisy normalized displacement $\mathbf{x}_{\tau,h,m,n}$, define

$$\begin{aligned} \mathbf{r}^p_{h,m,n} &= W_\Delta \mathbf{x}_{\tau,h,m,n} + b_\Delta + f_g(\gamma(\mathbf{p}_{0,m,n})) + \mathbf{e}^{\text{slot}}_m + \mathbf{e}^{\text{future}}_h, \\ \mathbf{z}^p_{h,m,n} &= f_p\Big([\mathbf{r}^p_{h,m,n}; \mathbf{e}^\tau]\Big), \qquad \mathbf{z}^g_{h,a} = f^{\text{in}}_g\big([\mathbf{e}^{\text{grip}}_a + \mathbf{e}^{\text{future}}_h; \mathbf{e}^\tau]\big). \end{aligned} \tag{21}$$

The 27-wide γ concatenates normalized/clipped XYZ and four bands of sine/cosine features with frequencies $2^j\pi$, $j = 0, \ldots, 3$. XYZ normalization uses center $(0.55, 0.30, 0.72)$ and half extent $(1.75, 0.95, 0.90)$, clipped to $[-1.5, 1.5]$. Both token types share a learned future-step table and a sinusoidal encoding $\mathbf{e}^\tau$ of $\lfloor 1000\tau \rfloor$. Flow time is *concatenated*, not added or supplied through AdaLN. The separate input MLPs use LayerNorm, Linear, SiLU, Linear with widths $1536 \to 768 \to 768$. Gripper tokens contain no current gripper state, GT value, or added noise.

Each future step contains MN point tokens followed by two gripper tokens; 32 learned registers precede the entire sequence. Conditions are

$$\mathbf{C}_t = [\mathbf{Z}^{\text{ground}}; \mathbf{Z}^{\text{obj}}; \{\mathbf{Z}^{\text{aux}}_v\}_v]. \tag{22}$$

The 12-block DiT alternates cross-attention and bidirectional self-attention, with standard pre-layer normalization and a FFN per block. It has 12 heads of width 64, internal width 768, and FFN width 3072. After final LayerNorm and projection to 512, separate LayerNorm–Linear–SiLU–Linear heads output point velocities $\hat{\mathbf{v}} \in \mathbb{R}^3$ and gripper logits $\hat{\ell} \in \mathbb{R}$. Invalid point slots are zeroed and masked as self-attention keys; their outputs are excluded from the loss. Both gripper tokens remain valid. Padding uses one batched forward, not individual sample forwards.

Table 6: Bimanual tensor shapes. N_v is view-dependent (N_0 for the main view) and L_{cur} includes CURRENT-span padding. Sequence length is $32 + 30(2 \cdot 30 + 2) = 1892$. Masks identify valid entries.

Tensor	Shape
Per-view visual/PETR tokens	$[B, N_v, 2048]$
Grounding memory	$[B, N_0 + L_{\text{cur}}, 512]$
Box outputs / role logits	$[B, 2, 12]$ / $[B, 2, 4]$
Anchors / object tokens	$[B, 2, 30, 3]$ / $[B, 2, 512]$
DiT input	$[B, 1892, 768]$
Point velocity	$[B, 30, 2, 30, 3]$
Gripper logits	$[B, 30, 2]$

Inference. Starting with Gaussian point noise, use $K = 16$ forward-Euler evaluations:

$$\mathbf{x}^{(j+1)} = \mathbf{x}^{(j)} + K^{-1}\hat{\mathbf{v}}_\theta(\mathbf{x}^{(j)}, j/K; \mathbf{C}_t). \tag{23}$$

For $j = 0, \ldots, K-1$, time is bucketed as above. Metric displacement is $\Delta\hat{\mathbf{P}} = 0.1\mathbf{x}^{(K)}$. Gripper logits are recomputed, not integrated; the returned open probabilities are $\sigma(\hat{\ell}^{(K-1)})$ from the last evaluation.

B.5 From Point Flow to Executable Robot Actions

Convert the anchor and predicted future points to world coordinates using $\mathbf{E}_t^{-1}$. For corresponding sets $\mathbf{x}_n$ and $\mathbf{y}_n$, let $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ be their centroids. Rigid least-squares fitting has the Kabsch solution

$$\begin{aligned} \sum_n (\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{y}_n - \bar{\mathbf{y}})^\top &= U\Sigma V^\top, \\ D_R &= \operatorname{diag}\big(1, 1, \det(VU^\top)\big), \\ \hat{R} = VD_RU^\top, \qquad \hat{\mathbf{d}} &= \bar{\mathbf{y}} - \hat{R}\bar{\mathbf{x}}. \end{aligned} \tag{24}$$

For the resulting homogeneous transform $\Delta\hat{T}^w_{h,m}$, each role-assigned end effector receives

$$\hat{T}^w_{a,h} = \Delta\hat{T}^w_{h,m} T^w_{a,0}. \tag{25}$$

Each future transform is fitted relative to the same anchor, not accumulated from previous predictions. A shared object applies the same rigid transform to both hands while preserving their different anchor offsets. An unassigned hand holds its anchor pose. This mapping assumes rigid coupling and consumes both translation and rotation. The RoboTwin adapter rejects fits above 5-cm point RMSE and converts predicted open probabilities using hysteresis: above 0.7 opens, below 0.3 closes, otherwise retaining the previous command. Poses and gripper commands are passed to IK/control, followed by a new observation and planning call after the chunk. No learned EEF-pose head is required.

B.6 Training Targets and Objectives

Targets and stage boundaries. Let $T^w_m(t)$ map object-local coordinates to world coordinates. Training uses Hungarian-matched *predicted* boxes to sample anchors, then transports them with GT object motion $\Delta T^w_{h,m} = T^w_m(f_h)[T^w_m(t)]^{-1}$:

$$\bar{\mathbf{p}}^{c_t}_{h,m,n} = \mathbf{E}_t \Delta T^w_{h,m} \mathbf{E}_t^{-1} \bar{\mathbf{p}}^{c_t}_{0,m,n}, \qquad \mathbf{x}_{1,h,m,n} = (\mathbf{p}^{c_t}_{h,m,n} - \mathbf{p}^{c_t}_{0,m,n})/0.1. \tag{26}$$

For ordinary samples in stage k, $f_h = \min(t+h, e_k - 1)$, so point targets hold at the stage endpoint. Gripper labels follow frame $t + h$ inside the stage; beyond it they repeat the next stage's *first* command, or the terminal command if no next stage exists. Both hands use 1 =open and 0 =closed. A special boundary anchor at $b_{k+1} - 1$ is supervised as stage $k+1$, switching its plan, selected objects, and future targets together. Short stages are supported by terminal hold.

Planning and grounding. Planning loss is mean causal cross entropy over assistant-answer tokens, including CURRENT/NEXT delimiters; prompt, image, and padding labels are ignored. For predicted query q and valid GT object j, matching uses

$$C_{qj} = 5\|\hat{\mathbf{c}}_q - \mathbf{c}_j\|_1 + \|\log \hat{\boldsymbol{\eta}}_q - \log \boldsymbol{\eta}_j\|_1 + d_R(\hat{R}_q, R_j) - \log \pi_{q,r_j},$$
$$\sigma^* = \arg\min_{\sigma} \sum_j C_{\sigma(j),j}. \tag{27}$$

Here, σ ranges over injective assignments from ground-truth objects to predicted queries. For $A = \hat{R}R^\top$, the angular error is

$$d_R = \operatorname{atan2}\left(\frac{\|\mathbf{v}(A)\|_2}{2}, \operatorname{clip}\left(\frac{\operatorname{tr} A - 1}{2}, -1, 1\right)\right), \tag{28}$$

where $\mathbf{v}(A) = (A_{32} - A_{23}, A_{13} - A_{31}, A_{21} - A_{12})$. With J matched objects across the batch and unmatched role targets $\bar{r}_{bq} = \text{invalid}$, define

$$L_c = \frac{1}{3J}\sum_j \|\hat{\mathbf{c}}_{\sigma^*(j)} - \mathbf{c}_j\|_1,$$
$$L_s = \frac{1}{3J}\sum_j \|\log \hat{\boldsymbol{\eta}}_{\sigma^*(j)} - \log \boldsymbol{\eta}_j\|_1,$$
$$L_R = \frac{1}{J}\sum_j d_R(\hat{R}_{\sigma^*(j)}, R_j), \tag{29}$$
$$L_{\text{role}} = -\frac{1}{BQ}\sum_{b,q} \log \pi_{bq,\bar{r}_{bq}},$$
$$L_{\text{ground}} = 5L_c + L_s + L_R + L_{\text{role}}.$$

The assignment is per example; the sums above pool its matched pairs. Box losses are zero when $J = 0$. Center error is measured in meters, size error in log space, and role loss is unweighted cross entropy. Unlike the matching cost, center/size losses average over coordinates.

Flow matching and gripper classification. For each example, independently sample noise $\epsilon \sim \mathcal{N}(0, I)$ and $u \sim \text{Beta}(1.5, 1)$, and set $\tau = 0.999(1 - \min(u, 0.999))$. All its tokens share this flow time. The interpolation and target velocity are

$$\mathbf{x}_\tau = (1 - \tau)\epsilon + \tau \mathbf{x}_1, \qquad \mathbf{v}^* = \mathbf{x}_1 - \epsilon. \tag{30}$$

Let w_i mask valid scalar point coordinates and m_j^g mask valid gripper labels. The action objectives are

$$L_{\text{pointflow}} = \frac{\sum_i w_i(\hat{v}_i - v_i^*)^2}{\max(1, \sum_i w_i)}, \qquad L_{\text{gripper}} = \frac{\sum_j m_j^g \,\text{BCEWithLogits}(\hat{\ell}_j, g_j)}{\max(1, \sum_j m_j^g)}. \tag{31}$$

Point validity combines selected-object/projection validity and future-object validity. Terminal-hold targets remain supervised. Both grippers are supervised independently of active object count. Conditions are repeated for two independent noise/time draws per example. The joint loss is

$$L = L_{\text{plan}} + L_{\text{ground}} + L_{\text{pointflow}} + L_{\text{gripper}}. \tag{32}$$

The VLM, including its shared visual encoder, is fully fine-tuned. Predicted box coordinates are detached before anchor sampling and ROI projection; action losses therefore update shared feature paths, including multiview PETR, but not the box head through its geometric outputs. Matching and discrete role selection are not differentiated. No additional EEF-pose, rigidity, or intermediate-decoder-layer loss is used.

C Experimental Settings

Policy interface. PointPilot receives the task instruction, current RGB observations, and generated subtask history, initialized empty at episode start. We use greedy language decoding, a 30-step prediction horizon, and 16 flow-matching integration steps. Predicted point motion is converted to end-effector targets by rigid alignment; gripper commands are also model-predicted. Replanning uses fresh observations after each chunk of 30 targets, unless execution terminates earlier.

Setting	Trials per task	Execution budget
LIBERO Spatial / Object	50	500
LIBERO Goal / Long	50	1000
RoboTwin clean	100	400
Paired cross-embodiment	100 per embodiment	300
Real world	20	600

Table 7: Evaluation budgets. Simulation budgets count environment action calls; physical budgets count predicted waypoint steps, excluding low-level interpolation. A lower task-specific environment limit takes precedence. All settings use horizon and replanning chunk length 30.

C.1 Simulation Experiments

Our model is jointly trained on the Libero and Robotwin 2.0 datasets by 60k steps, using 4*80G A800 GPUs.

LIBERO. We evaluate Spatial, Object, Goal, and LIBERO-10 (Long) using a single main-view RGB observation and a wrist view observation. Fixed episode initializations are shared across policies and identified by task, demonstration, and initial simulator state in the evaluation manifest. Ten zero-action settling steps precede the first policy observation and do not consume the budget in Table 7. Success uses the environment's task predicate. We use MuJoCo 3.3.2 and robosuite 1.4.0 with the corresponding LIBERO assets.

RoboTwin. We evaluate the 50 tasks (100 trials per task) under the clean configuration with the ALOHA-AgileX embodiment and head, left-wrist, and right-wrist RGB views. Candidate seeds are examined in ascending order from 100000; acceptance requires both successful expert planning and satisfaction of the task success predicate. The environment is then reset before evaluating the learned policy. We use SAPIEN 3.0.0b1 and MPLib 0.2.1 with clean task assets; randomized-background results are not mixed into this protocol.

C.2 Real-World Experiments

Platform and tasks. The physical platform is a CobotMagic bimanual system with an external camera and two wrist cameras. The three RGB streams are aligned to a common observation timestamp. We consider QR-code scanning and sorting, cube-to-basket transfer, and pouring glass beads between cups. Task-specific policies use demonstrations with stage annotations, camera-frame object tracks, and binary gripper commands. Each task is trained by 20k steps, using 50 demonstrations.

Execution and assessment. Object arrangements are fixed before policy comparison and independently reset for each trial. A robot-side client converts model predictions using camera-to-base calibration and robot kinematics, with feedback checks before execution. Success requires the complete task: scanning all specified QR codes and placing the scanned objects in the designated tray, placing and releasing the cube in the basket, or transferring the beads into the receiving cup without spilling. Operator corrections are not permitted; safety interventions and policy-induced control failures count as unsuccessful trials. Offline point-flow visualizations are not counted as physical task successes.

C.3 Zero-Shot Cross-Embodiment Transfer

Frozen policy and paired scenes. We transfer an ALOHA-trained policy to ARX-X5 and dual-Piper without target-embodiment demonstrations for task fine-tuning at evaluation time. The same checkpoint, prompt, point-flow sampling, and learned gripper prediction are used for source and target robots. Calibration, robot descriptions, kinematics, and execution interfaces are embodiment-specific. Each source–target comparison uses 100 accepted scenes and equal execution budgets.

Scene pairing preserves object assets and arrangements, table/background configuration, main-camera calibration, and the exact instruction. Equality is checked after natural settling and any

robot initialization adjustment, using tolerances of 2 mm in object position and 2° in orientation. Robot appearance and wrist observations remain embodiment-dependent. For Piper, initial end-effector poses are additionally aligned to the ALOHA reference within 2 mm and 2° per arm. ARX retains its native feasible initialization; identical initial end-effector poses are not assumed for that comparison.

Feasibility and evaluation. Task eligibility requires the target embodiment's task assets and a feasible expert execution. Scenes are accepted only when both source and target experts succeed and the pairing checks pass; Piper also requires successful initial-pose alignment. Experts qualify the initial conditions but do not provide actions or intermediate labels to the policy. The accepted list is frozen before policy testing and shared by the ALOHA reference and corresponding target policies. The metric therefore measures transfer on expert-feasible paired scenes, not arbitrary target initializations.