Appendix

A Related Work

2D and 3D Visual Representations Visual encoders have played a central role in both perception and control, with 2D vision backbones such as DINOv3 [63], CLIP [64], and ViT [15] demonstrating remarkable generalization across diverse visual domains. In contrast, 3D vision encoders explicitly model geometric information using volumetric, point-cloud [65–69], or multi-view representations [70], yet their performance often suffers from the scarcity of large-scale annotated 3D datasets. To mitigate the limited availability of 3D data, some recent approaches aim to "lift" pretrained 2D vision representations into 3D representations (e.g. NeRF [71]) for 3D scene understanding [72–75].

Stereo Vision in Computer Vision Stereo vision provides an efficient mechanism for inferring 3D structure from image pairs. Modern stereo matching methods [33, 76–83] focus on accurate explicit disparity or depth estimation using cost volumes, multi-scale features, and recurrent refinement. Recent work has achieved strong zero-shot transfer under challenging conditions [33, 83], while others introduce scalable architectures for multi-range depth reasoning [76, 78]. Multi-view extensions [84, 85] further push stereo paradigms toward volumetric and 4D understanding. Together, these studies demonstrate stereo vision as a compact pathway to geometric reasoning—lifting 2D image pairs into 3D without volumetric supervision. However, these methods optimize for pixel-level disparity rather than downstream task performance, and typically require additional modules to produce actionable control signals. Our work instead proposes an implicit stereo encoder that directly yields task-aware geometric embeddings for visuomotor learning, bypassing explicit disparity regression.

Stereo Vision for Robotics Stereo vision has been widely adopted in robotics for manipulation [34, 86, 87] and navigation [88, 89]. However, these approaches rely on explicit depth or occupancy estimation and require additional modules to produce actions. Due to the scarcity of large-scale pretrained stereo vision backbones, they also fail to leverage the rich representations of modern pretrained 2D encoders. In contrast, our method directly processes each stereo image with pretrained 2D vision backbones and fuses the resulting features for end-to-end policy learning, enhancing robustness and generalization. While Singh et al. [49] and Lum et al. [90] also use stereo inputs for end-to-end dexterous grasping, they do not study stereo feature fusion or compare against explicit 3D baselines such as point clouds.

B Implementation Details

B.1 StereoPolicy-DP Training

For each stereo image pair, to get effective stereo representations for stereo transformer, we use ResNet18 and FPN to extract the fine-grained visual features. For external camera views, we additionally concatenate frozen DINOv2 features with the task image features. We apply a 4×4 convolution with stride 4 to downscale the feature of DINOv2. The feature is then concatenated with CNN feature to obtain a hybrid feature. The CNN network is thus learned to adapt the ViT features. The concatenated features are then downsampled by an MLP to get projected left/right tokens. For each stereo camera view, the projected left and right tokens are concatenated and passed into a lightweight stereo Transformer with 2 Transformer layers and 8 attention heads. Each layer contains self-attention, cross attention, and an MLP block. We apply 2D RoPE to the query and key projections to preserve spatial information during cross-view correspondence learning. Stereo features from each view are then projected to a 128-dimensional latent space using an MLP to ensure a consistent token dimension for fair comparison. The DINOv2 encoder is frozen throughout training, while rest of vision encoder like CNN, stereo Transformer, and diffusion policy backbone

are trained end-to-end. Stereo fusion is performed independently for each camera view, and the resulting view-level stereo features are concatenated across views together with the low-dimensional proprioceptive state as the observation condition for the diffusion policy.

For RGB and RGBD, we adopt a ResNet18 as the vision encoder. For all diffusion policy, we use a UNet [91] as the policy backbone and replace the BatchNorm layers in the UNet with GroupNorm [92] for stable training [8].

We train diffusion policies from scratch using RGB and stereo. For RGB, we use left cameras as input. All models are trained with a batch size of 64 and a learning rate of 1e-4. The observation horizon is set to 2 timesteps, while the action prediction horizon is set to 16 steps. For Omnigibson tasks, we train for 1,000 epochs, while Robomimic tasks are trained for 500 epochs following common practice in prior work.

B.2 StereoPolicy-VLA Training

For finetuning pre-trained VLA experiments, training is performed on 8 H100 GPUs. We use pre-trained checkpoint GR00T-N1.5 on Robocasa tasks, we train for 60K gradient steps, maintaining the original training protocol as closely as possible to ensure comparability. We set the learning rate to 1e-4. Across all settings, optimization hyperparameters follow the corresponding baseline configurations to isolate the effect of our stereo architecture.

For $\pi 0.5$, we use the pre-trained checkpoints to reproduce fine-tuned performance on RoboCasa-Kitchen. We train it for 80K steps, both with a global batch size of 128. We set the learning rate to 2.5e-5 with cosine decay to 2.5e-6 and 1K warmup steps. At inference, we use an action horizon H = 16 and execute all actions without re-planning

B.3 Baseline Implementations

Monocular RGB Following Robomimic [52] and Diffusion Policy [8], at each timestep t, the sensor inputs include monocular RGB images from each view i, $I_{t,i} \in \mathbb{R}^{H\times W\times 3}$, which are processed by a ResNet18 to obtain vision features.

Multi-view RGB Stereo images from each view are passed to the vision encoders, and the resulting image features are concatenated before the diffusion policy.

RGB-D Each view provides an RGB image and an aligned depth map, enabling explicit spatial structure for downstream processing. Depth maps $\in \mathbb{R}^{H\times W\times 1}$ are repeated to 3 channels and processed by a separate ResNet18 encoder. The resulting RGB and depth features are concatenated.

RGBD-3DDA We follow the same setup as in the original paper [55], use ResNet18 and FPN to extract multi-scale RGB features and reconstruct point-clouds from depth. For fair comparison, Transformer-based fusion in 3DDA is substituted by an MLP compatible with diffusion policies.

Point-clouds with PointNet We reconstruct and fuse multi-view point clouds from RGB-D observations, followed by downsampling them to 4096 points using farthest point sampling (FPS) [56].

PCD-DP3 For point-cloud inputs, we reconstruct and fuse multi-view point clouds from RGB-D observations, downsample them to 4096 points using farthest point sampling (FPS) [56], and encode them using the DP3 encoder [37].

B.4 Evaluation

For real-robot evaluation, we report the average success rate over 20 trials, with randomized initial poses. For simulation tasks, we perform 50 rollouts at every 50 training epochs for Robomimic tasks, and every 250 training epochs for Omnigibson tasks. The highest success rate achieved across training is reported.

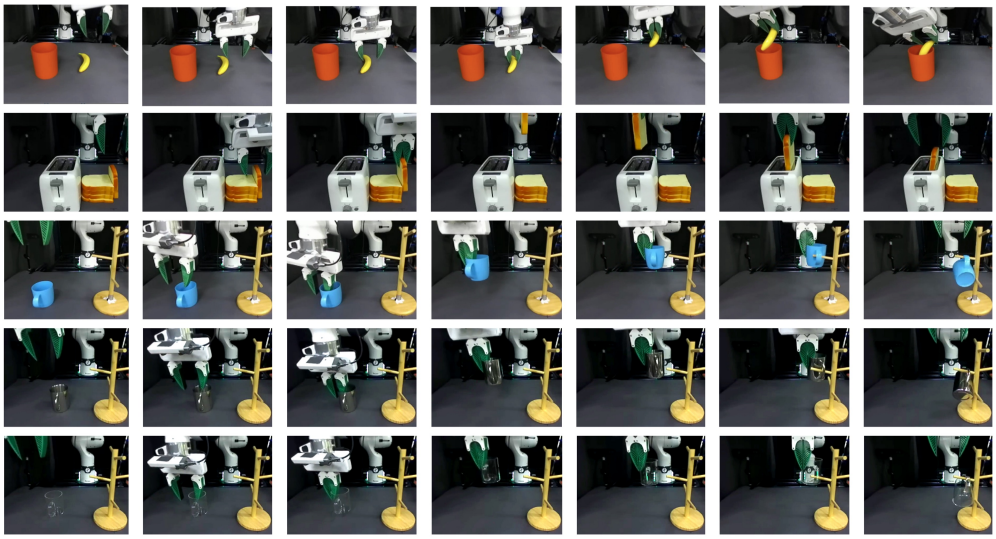

Figure 11: Trajectory of Real-world Tabletop Tasks.

C Task and Data Collection Details

Simulation Tasks

ROBOMIMIC: We exclude *Lift* and *Can* since the success rate has already saturated in the original study paper, and include three more challenging tasks *ToolHang*, *Square*,and *Transport*.

ROBOCASA-KITCHEN: We use 24 kitchen tasks from this benchmark, and the main purpose is to evaluate the pre-trained VLA models (Pi0.5 and GR00T-N1.5).

OMNIGIBSON is a high-fidelity physics simulation environment that models a large variety of object interactions in realistic home-scale scenes, enabling long-horizon mobile manipulation tasks grounded in everyday human activities. We designed four tasks and collect demos with motion planning library cuRobo [93].

Real Robot Tasks. We design five real-world tabletop manipulation tasks, we collected 200 demonstrations for each of the task. The examples of training demo trajectories are shown in Figure 11.

- BANANA PNP: The robot picks up a banana and inserts it into a container.
- TOAST INSERT: The robot picks up a toast and inserts it into the toaster.
- CUP HANG: The robot picks up a mug and hangs it on the holder.
- STEEL CUP HANG: The robot picks up a steel mug and hangs it on the holder.
- GLASS CUP HANG: The robot picks up a glass mug and hangs it on the holder.

For bimanual mobile manipulation, we have following tasks, we collected 75 demonstrations for each of the task.

- TURN ON RADIO: The R1Pro robot approaches the shelf and picks up the radio, turn use left gripper to press the button of radio, and puts the radio back on the shelf.
- TOAST PNP: The R1Pro robot approaches the table, picks up the toast from the plate, hang it over to the left gripper and puts it back to the shelf.

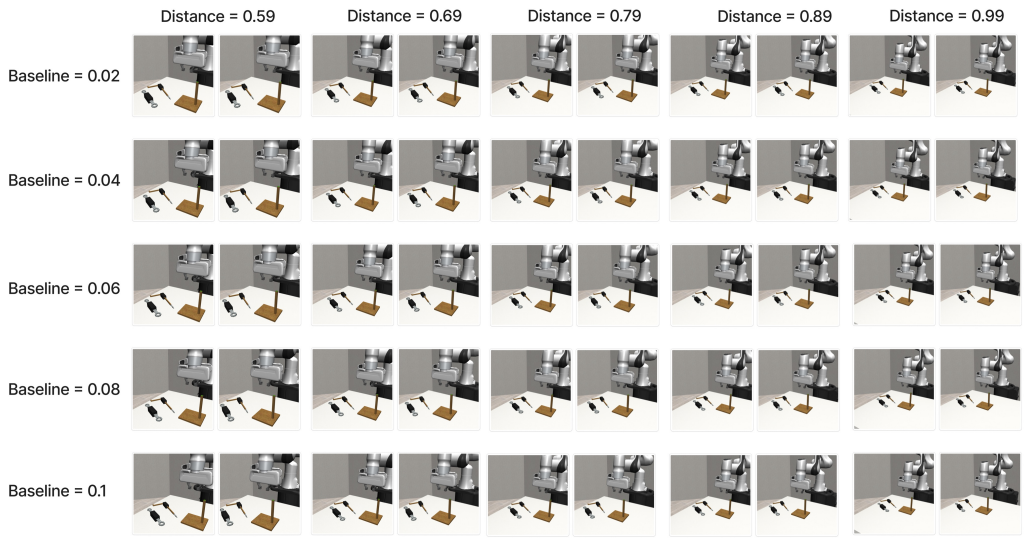

Figure 12: Stereo camera views across different baselines and distances. Baseline indicates the distance between stereo cameras, distance indicates the camera-object distance.

Figure 13: Camera angle view visualization. Experimental results are Figure 8b.

C.1 Real-Robot Hardware Setup

To further validate the real-world applicability, we deploy the stereo policy on a 7-DoF Franka Emika robot arm. For visual perception, we utilize the dual camera setup: a ZED Mini provides an external view, and a wrist-mounted ZED Mini captures a close-range view.

D Hardware and View Configuration Details

D.1 Stereo Baselines

The baseline distance between the two corresponding views is set to 6 cm for external cameras and 2 cm for the wrist camera by default.

D.2 Stereo View Visualization across Different Baseline and Distance

§3.3 ($\mathcal{Q}\mathbf{3}$), we demonstrate that how baseline and target-object distance impacts StereoPolicy. In Figure 12, We visualize all stereo camera views across baselines ranging from 0.02m to 0.10m and object distances ranging from 0.59m to 0.99m that are used in the experiments.

D.3 Camera Angle Mappings

As reported in §3.3 ($\mathcal{Q}\mathbf{2}$), we experiment TOOLHANG task across 10 different camera angle views. Figure 13 shows the 10 views, which are *frontview, frontview-high, agentview-left, agentview-right, agentview-righthigh, agentview-lefthigh, sideview-left, sideview-right, sideview-lefthigh, sideview-righthigh* from left to right.

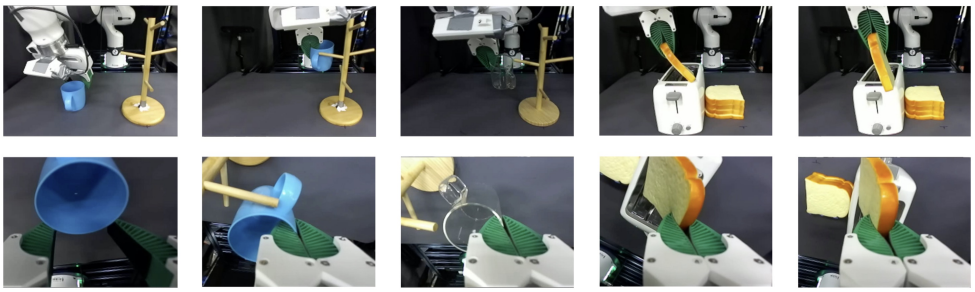

Figure 14: Failure cases of baseline visual modalities on real-world tabletop tasks. RGB, RGB-D, and point-cloud inputs can be inaccurate for fine-grained manipulation: gripper-target alignment can drift, reflective objects can obscure object boundaries, and transparent objects often produce missing or noisy depth and point-cloud geometry.

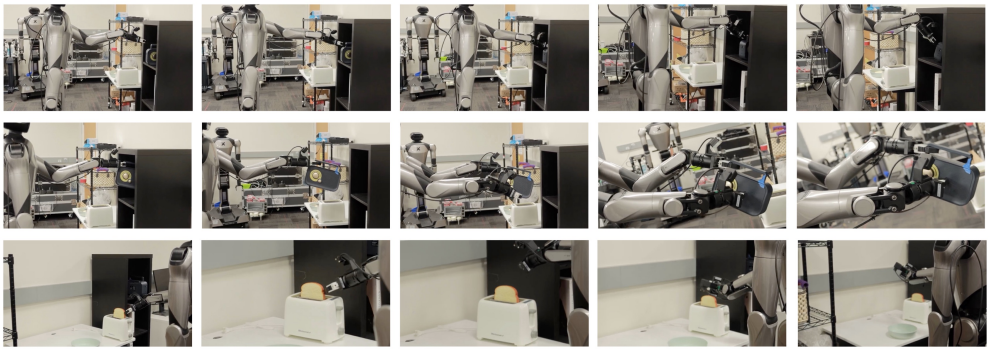

Figure 15: Monocular RGB failures in bimanual mobile manipulation. In the TURN ON RADIO task, RGB policies often fail at precise handle insertion and button pressing due to insufficient depth cues. In the PNP TOAST task, failures commonly occur during grasp alignment and placement due to ambiguous monocular depth cues.

E Additional Experimental Results

E.1 Inference Latency Ablation

We compare the inference latency of RGB-DP, STEREOPOLICY-DP, and 3D Diffuser Actor (3DDA) in Table 3. In Robomimic setup, STEREOPOLICY-DP increases end-to-end inference latency from 621.66 ms to 698 ms, corresponding to a $1.12\times$ overhead over RGB-DP. The additional cost mainly comes from stereo image encoding and stereo feature fusion, while the iterative diffusion-policy backbone remains the dominant component of total inference time. Latency in 3DDA is dominated by reconstructing 3D point-cloud from RGB-D.

Method	Inference latency
RGB-DP	621.66 ms
STEREOPOLICY-DP	698.55 ms
3DDA-DP	705.90 ms

Table 3: Inference latency comparison. RGB-DP and STEREOPOLICY-DP are measured in our setup with batch size 1, observation horizon 2, prediction horizon 16, action dimension 7, and 100 denoising steps. Measured by running on Nvidia H100 GPU.

E.2 StereoPolicy-DP maintains its performance when the wrist camera is removed

We use ToolHang as the diagnostic task to systematically study the design choices. Shown in Table 4, StereoPolicy-DP maintains its performance when the wrist camera is removed, while RGB and RGB-D policies' performance immediately drop. Being able to remove the wrist camera could be helpful in robotic manipulation tasks since the mounted camera could collide with objects.

Setting	RGB	RGB-D	PCD			RGB-D	Multi-View	StereoPolicy-DP
			PointNet	PointNet++	DP3	3DDA		
Default (w/ wristCam)	90.0%	88.0%	32.0%	28.0%	76.0%	92.0%	92.0%	**96.0%**
w/o wristCam	86.0%	84.0%	30.0%	28.0%	62.0%	88.0%	88.0%	**94.0%**

Table 4: Performance on Robomimic ToolHang task.

F Qualitative Failure Cases

Baseline modality failures in tabletop manipulation. Figure 14 shows representative tabletop failures caused by unreliable object geometry and spatial alignment. In Cup Hang tasks, weak object-boundary perception can lead to inaccurate cup localization and failed grasping. Noisy depth observations further cause the cup handle to misalign with the rack peg during hanging. For Glass Cup Hang tasks, specular reflection and transparency can make the cup partially or completely disappear in depth videos and reconstructed point clouds while it is lifted in free space. In Toast Insert tasks, insufficient 3D spatial information or inaccurate depth estimates can cause vertical or lateral misalignment between the toast and the toaster slot.

Monocular RGB failures in mobile manipulation. Figure 15 visualizes representative failure cases of RGB-based VLA policies on mobile manipulation tasks. In the Turn on Radio task, the RGB policy often approaches the radio but fails to insert the gripper into the handle or press the button accurately. In the PnP Toast task, RGB policies can localize the toast but often fail during grasp alignment or final placement, where accurate spatial reasoning is required. In contrast, StereoPolicy benefits from stereo cues and better estimates the relative depth between the gripper and the target affordance.