

StereoEngine: Scaling Stereo Bimanual Robot Data without Stereo Cameras

Anonymous Authors



Fig. 1: **StereoEngine generates stereo data for bimanual manipulation.** A mosaic of synthetic stereo pairs produced by our video diffusion framework where each adjacent column pair forms a left–right stereo view. The left half of the gallery shows our first method (StereoEngine-L) which takes a monocular real-world demonstration and synthesizes the corresponding right view to complete the stereo pair. The right half shows our second method (StereoEngine-S), which generates both left and right views jointly from simulated demonstrations, producing large-scale and photorealistic stereo data without any real video as input.

Abstract—Stereo vision is increasingly used for robotic manipulation with a growing body of work incorporating it into Vision-Language-Action (VLA) models. However, the scarcity of large-scale, real-world stereo datasets remains a fundamental bottleneck for scaling such policies. Recent generative monocular-to-stereo methods target settings such as VR content and autonomous driving. No prior work produces realistic synthetic stereo data for robotic manipulation where stereo supervision has come predominantly from simulation. We propose StereoEngine, which introduces two complementary methods to synthesize stereo data for robotic manipulation. In the first, we warp a monocular (left) video into the right view using estimated depth and inpaint the disoccluded regions with a video diffusion model which lifts monocular demonstrations into stereo pairs without training on real stereo. In the second, we render a stereo pair in simulation and jointly generate both photorealistic views which produces consistent left–right data without any real video as input. StereoEngine outperforms existing baselines in generated stereo quality, and using its synthetic data for policy mid-training or task-specific fine-tuning substantially improves downstream bimanual manipulation. Project website: <https://stereoengine.github.io>.

I. INTRODUCTION

Imitation learning from teleoperated demonstrations has enabled increasingly capable bimanual manipulation systems [1]–[3]. Recent stereo-conditioned policies extend this by taking stereo observations as input and exploiting binocular views to improve spatial reasoning, including challenging cases such as transparent objects [4, 5]. However, scaling these stereo-conditioned policies to the point where they can be used

for large-scale general imitation learning remains difficult, as large-scale robot datasets are overwhelmingly monocular and single-arm [6]–[8]. The few datasets captured with stereo cameras, such as DROID [9], are single-arm which means no large-scale bimanual stereo dataset exists. Collecting stereo demonstrations at comparable scale would require expensive stereo rigs and costly re-teleoperation. This motivates scalable methods to synthesize stereo data without additional real-world collection [10]–[13].

While monocular-to-stereo synthesis has emerged as a promising direction [14]–[16], existing works target domains outside of robotic manipulation, such as VR/AR content [17]–[20]. Reconstruction-based novel-view synthesis (e.g., NeRF [21], 3D Gaussian Splatting [22]) requires multi-view datasets and per-scene optimization on static scenes, unlike our dynamic single-view setting. Methods that train on stereo pairs draw on simulation data that suffers from domain gaps or on only a small number of real pairs [23]. To our knowledge, no prior work synthesizes realistic stereo data directly from real robot video data.

To address the data bottleneck, we present StereoEngine, a scalable framework for synthesizing photorealistic stereo data for bimanual manipulation using video diffusion. StereoEngine provides two independent methods. When a real monocular demonstration exists, we convert it into a stereo pair by warping the left view and inpaint the disoccluded regions which are the surfaces that become visible in the right view but were never observed in the left. This preserves the appearance

of the real data, though every generated pair remains tied to an existing demonstration. When no real video exists, we instead generate both views from a simulated trajectory, which allows data generation for entirely new tasks and configurations. Together, the two methods lift the abundant monocular data that already exists and generate new data where none exists, without ever training on real stereo correspondences.

We evaluate StereoEngine across 480 real-world rollouts spanning five bimanual manipulation tasks. See Figure 1 for a visualization of the diverse synthetic stereo pairs produced by StereoEngine. Using ExStereo [24] and StereoVLA [5] as our downstream policies, mid-training [24] them on StereoEngine data—an intermediate stage between a policy’s large-scale pre-training and task-specific fine-tuning—or fine-tuning them on it directly, consistently improves manipulation performance and generalization over baselines.

The contributions of StereoEngine are:

- StereoEngine-L, a monocular-to-stereo pipeline whose inpainter is trained without any real stereo pairs, using warp-derived masks, partial arm masks, and disparity dropout.
- StereoEngine-S, which jointly generates both left and right views from simulation, needing no real video.
- A large-scale bimanual robot stereo dataset released from both methods.
- Real-world experiments showing that StereoEngine data improve stereo-conditioned policies through both large-scale mid-training (StereoEngine-L) and task-specific fine-tuning (StereoEngine-S).

II. RELATED WORK

A. Stereo Vision in Computer Vision

Stereo vision offers an efficient way of recovering 3D structure from image pairs where depth is obtained by estimating the disparity between a rectified left–right pair. This means obtaining 3D information from ordinary RGB cameras without active depth sensors. Recent stereo matching has shifted toward general-purpose foundation models with strong zero-shot generalization, largely by leveraging priors from monocular depth foundation models to generalize across unseen domains and remain reliable on transparent, reflective, and textureless surfaces [25]–[27]. These advances make stereo an accurate and deployable source of 3D geometry well suited to robotics. In this work, we leverage these benefits for robot manipulation by generating large-scale stereo data so that stereo-conditioned policies can scale for training.

B. Monocular-to-Stereo Generation

Monocular-to-stereo generation synthesizes a right-eye view from a monocular left video, originally for immersive 3D content [28]. Most methods require stereo pairs for training, either from large-scale stereo datasets that are expensive to obtain or from synthetic renders with a large domain gap from real video [14, 18]–[20]; training-free [15, 29, 30] and recent self-supervised [16] variants relax this requirement. However, all of these only convert an existing left video and cannot

Method	Domain	Video Gen.	No Stereo Pairs	Variable Baseline	Both Views	Gen. Length
StereoCrafter [14]	VR	✓		✓		25*
SpatialDreamer [17]	VR	✓	✓	✓		8
Mono2Stereo [18]	3D Content					1
GenStereo [19]	XR/Driving			✓		1
StereoWorld [20]	XR	✓				81
SVG [15]	VR	✓	✓	✓		16
Geo. Recip. [16]	VR/AR	✓	✓			— [†]
StereoEngine (Ours)	Robotics	✓	✓	✓	✓	81

TABLE I: Comparison of stereo and view-synthesis methods. A checkmark (✓) indicates the capability is demonstrated or explicitly supported by the method; a blank cell indicates it is not. **Video Gen.**: operates on videos instead of single images. **No Stereo Pairs**: requires no real stereo correspondences for training. **Variable Baseline**: synthesizes stereo videos with a controllable stereo baseline instead of a fixed one. **Both Views**: synthesizes both left and right views. **Gen. Length**: maximum number of frames generated in a single pass. *Extendable to arbitrary length via autoregression. [†]No public code available and generation length not reported in the paper.

produce stereo pairs where none exists. As summarized in Table I, StereoEngine addresses both limitations: it needs no stereo pairs, operates on real robot data, and jointly generates both photorealistic views without any real footage.

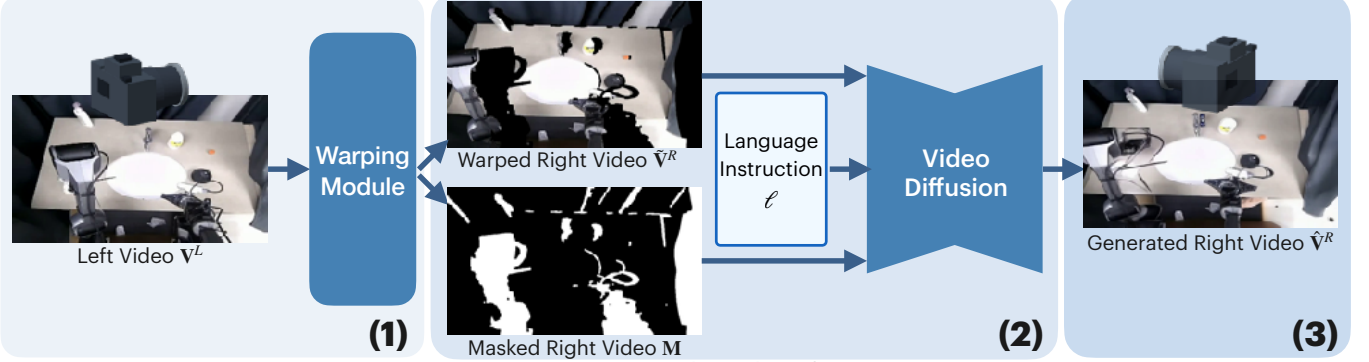
C. Stereo Vision in Robotics

Stereo perception offers a practical route to 3D geometry in manipulation, recovering structure on materials where active depth sensors fail, such as transparent, specular, and textureless surfaces. Stereo can also yield cleaner geometry than active depth sensors, as we show later in Section IV-A1. Prior work adapts stereo depth estimation to these challenging conditions [31, 32] and uses stereo depth for downstream grasping and tracking [33]. More recently, stereo has been integrated into imitation and Vision-Language-Action (VLA) policies. ActiveGlasses [34] reconstructs a point cloud from the stereo pair through a stereo matcher and feeds that to the policy, while StereoPolicy [4] and StereoVLA [5] consume the stereo images directly. All of these methods can benefit from large-scale stereo demonstrations, yet none address how such data can be obtained at scale. Stereo demonstrations can be teleoperated on a stereo rig, but this requires stereo hardware on every platform and teleoperation for every new task. StereoEngine instead generates the stereo data needed to train these policies without additional real-world collection.

D. Video Generation in Robotics

Video generation models have recently been used to scale robot data, either as world models that generate demonstrations for policy learning [35]–[39] or as data engines that augment existing demonstrations while keeping the robot trajectory fixed [40]–[43]. A related group of methods conditions video diffusion on simulation to synthesize new configurations or diverse demonstrations [10, 12]. Closest to our setting is CRAFT [13], which generates a video from Canny edges [44] of a simulated bimanual episode together with a real reference image. However, all of these generate only monocular video, and StereoEngine is the first to extend video generation for robot data to stereo pair synthesis.

StereoEngine-L: Stereo Generation **with** a Ground-Truth Left View



StereoEngine-S: Stereo Generation **without** a Ground-Truth Left View

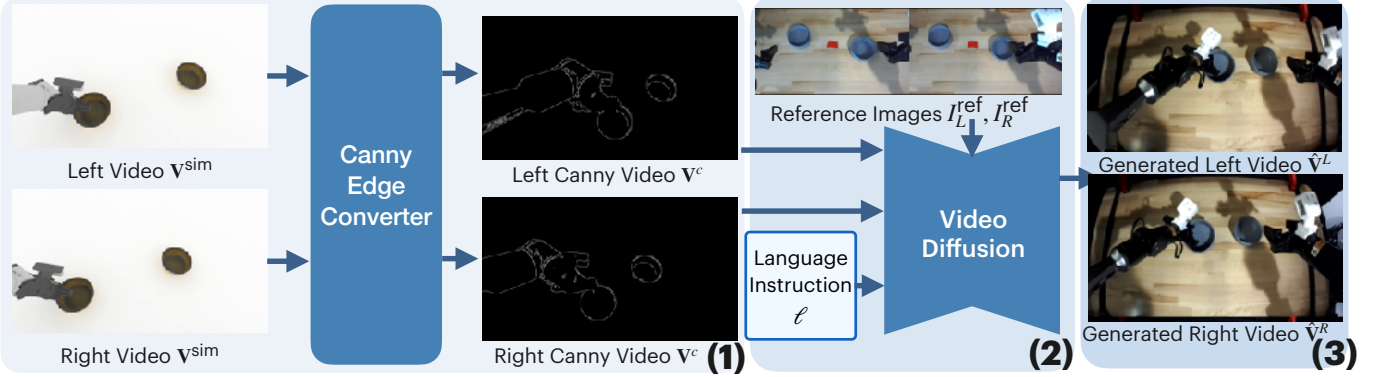


Fig. 2: **Method Overview.** **StereoEngine-L** applies when a ground-truth left video is available. The warping module (1) moves every left pixel to its position in the right view, giving a warped right video and a mask marking the pixels no left pixel could reach. A video diffusion model (2) fills the masked region from these and a language prompt ℓ , giving a right video (3) that pairs with the original left. **StereoEngine-S** applies when no real video exists. We render (1) the episode in simulation from two cameras, convert both renders to Canny edges, and (2) pass the edge videos to the model with two reference images and the instruction ℓ . The edge map defines the scene geometry, while the reference images supply the appearance of the real workspace such as background, lighting, and surface texture. Then (3) a video generation model generates the synthetic video. In both methods, the resulting stereo pairs can be used to train a downstream bimanual stereo-conditioned policy.

III. PROBLEM STATEMENT

We focus on scalable data generation for stereo-based policies in bimanual manipulation, where a policy π_θ with parameters θ is trained from stereo demonstrations. We write a camera’s ground-truth left and right views at time t as I_t^L and I_t^R , and synthesized views with a hat (e.g., $\hat{I}_t^R$). We consider two independent settings for generating synthetic stereo pairs. In the first, a ground-truth left view I_t^L is available and we synthesize the corresponding right view $\hat{I}_t^R$, forming the pair $(I_t^L, \hat{I}_t^R)$. In the second, no ground-truth view is available and we generate both views jointly, $(\hat{I}_t^L, \hat{I}_t^R)$. We present a method for each in Section IV. At deployment, the policy receives a real stereo pair (I_t^L, I_t^R) together with additional context c_t , and produces actions $a_t = \pi_\theta(c_t, I_t^L, I_t^R)$. The context $c_t = \{s_t, \dots, \ell\}$ gathers whatever additional information the policy has access to, such as the robot’s proprioceptive state s_t and a language instruction ℓ . In our experiments, c_t contains only the proprioceptive state s_t .

IV. METHOD: STEREOENGINE

StereoEngine addresses these two settings with two independent methods, each using a video diffusion model to synthesize photorealistic training videos (see Figure 2).

StereoEngine-L targets the first setting. Given a ground-truth left video $V^L = \{I_1^L, \dots, I_T^L\}$ and a language instruction ℓ , it synthesizes the corresponding right video $\hat{V}^R = \{\hat{I}_1^R, \dots, \hat{I}_T^R\}$ through two stages: warping (Section IV-A1) and inpainting (Section IV-A2). We describe how we train the inpainting model in Section IV-A3.

StereoEngine-S targets the second setting. Given a simulation-rendered source video $V^{sim} = \{I_1^{sim}, \dots, I_T^{sim}\}$, reference images I^{ref} , and a language instruction ℓ , it generates both photorealistic views jointly through two stages: simulation trajectory generation (Section IV-B1) and video generation (Section IV-B2). The instruction ℓ is different between the two methods.

A. StereoEngine-L: Stereo Generation from a G.T. Left View

Following prior monocular-to-stereo work [14, 15, 28], we treat the input monocular video as the ground truth (G.T.) left view and synthesize the corresponding right view. This choice of left to right generation instead of right to left is a convention in most stereo generation works and does not affect the method. Because no large-scale bimanual robotic stereo datasets are publicly available, naively training a model to synthesize right views is non-trivial. Our method generates right views without training a model on stereo pairs. Unlike

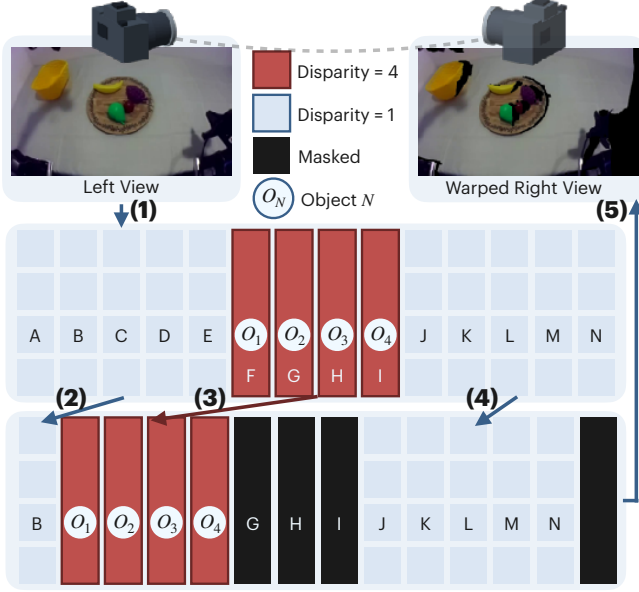


Fig. 3: **Warping Module.** (1) the ground truth left view is abstracted into a grid of pixels, where A to N label the background and O_1 to O_4 label the object, and each pixel is given the disparity it must move by, $d = 4$ for the object and $d = 1$ for the background. (2) every pixel moves left by its own disparity, since the camera translates right. The object moves further and covers C, D, E , and the nearer pixel is kept, so only B survives. (3) the background G, H, I was hidden behind the object and never observed, so nothing can fill the gap it leaves and those columns become the mask. (4) the rest of the background shifts left by one, leaving the right border without any source. (5) the grid is mapped back to pixels to give the warped right view, correct where a pixel arrived and empty where the left camera never saw the surface.

prior monocular-to-stereo methods, StereoEngine-L trains its inpainter only on unpaired monocular robot video, using warp-derived masks, partial arm masks, and disparity dropout to match inference and generalize across stereo baselines.

1) *Warping of the Left Video:* Given the ground-truth left video $\mathbf{V}^L = \{I_t^L\}_{t=1}^T$, we feed $\mathbf{V}^L$ through a monocular video depth model [45], which outputs an inverse depth map $\tilde{d}_t$ for each frame. We place a virtual right camera with the same orientation and intrinsics by performing a horizontal translation of the left camera. Figure 3 illustrates the procedure on a grid of pixels. Each pixel of the left frame corresponds to a 3D point at depth Z and moving the camera sideways by a baseline B shifts that point horizontally by

$$d = \frac{fB}{Z} \text{ pixels}, \quad (1)$$

where f is the focal length in pixels and d is the disparity. Since disparity is inversely proportional to depth, the predicted inverse depth $\tilde{d}$ gives the relative disparity across the image up to an unknown global scale. We fix this scale by setting $d = s\tilde{d}$ with $s = \delta_0 / \text{median}(\tilde{d})$, so that the median disparity equals a chosen δ_0 , which assigns each pixel the disparity it must move by (see Figure 3 stage 1). Any choice of δ_0 produces a geometrically self-consistent pair, and since the downstream point cloud is reconstructed from the stereo pair, the absolute scale is never used. For StereoEngine, we set $\delta_0 = 63$ pixels

($\sim 7.6\%$ of frame width, comparable to prior work [15]) at deployment.

We leverage forward splatting [46] to map each left frame pixel to its target position and splatting it onto the nearest right frame pixel. In the case that several pixels land on the same target, we weight each by its disparity using $w = \sqrt{2^d}$ so the one with the largest disparity dominates and correctly occludes those behind it (see Figure 3 stage 2). However, this mapping does not cover the whole right frame. Surfaces hidden behind an object in the left view become visible once the camera moves right which have no source pixel so we mark every empty target as occluded which gives the disocclusion mask $\mathbf{M}$ (see Figure 3 stage 3). We dilate $\mathbf{M}$ slightly so that it also covers the stretched pixels bordering large depth steps. A mask along the right border is empty as the pixels that would fill it are outside of the left image (see Figure 3 stage 4).

Warping therefore produces a right video $\tilde{\mathbf{V}}^R$ whose filled pixels are ground truth observations from the left view, moved to the positions they occupy under the right camera, together with the mask $\mathbf{M}$ marking every pixel the left camera never observed, which the next stage fills (see Figure 3 stage 5).

2) *Inpainting of the Right Video:* Diffusion-based video generative models learn to approximate a distribution over video sequences through iterative denoising and the model learns the conditional distribution:

$$p_\psi(\hat{\mathbf{V}}^R | \tilde{\mathbf{V}}^R, \mathbf{M}, \ell), \quad (2)$$

where $\tilde{\mathbf{V}}^R$ is the warped right video from Section IV-A1, $\mathbf{M}$ is the disocclusion mask, ℓ is a text prompt, and ψ denotes the model parameters. We use Wan2.2-VACE-Fun [47] as our backbone. Wan2.2-VACE-Fun inpaints poorly zero-shot on this out-of-distribution robot domain, so we fine-tune it for our setting. We pass the warped video and the mask both as the region to be filled and as the control video which tells the model exactly which pixels it may synthesize and which it must preserve. The completed $\hat{\mathbf{V}}^R$ is then paired with $\mathbf{V}^L$ to form the stereo pair.

3) *Training of the Inpainting Model:* Since no large-scale robotic stereo dataset exists, the inpainting model p_ψ from Section IV-A2 cannot be trained on real stereo pairs. It can instead only be trained on ordinary monocular video since the ground truth for a synthetic mask is the clip itself.

We first pull 3,469 episodes of robot data of the same embodiment from public sources such as HuggingFace. We deploy on AgileX PiPER arms in the real world so we collect datasets recorded with PiPER arms. We also include the AgileX Cobot Magic platform which uses the same PiPER arms mounted on a mobile base with a leader and follower system. None of these datasets contain stereo pairs. Our goal is to generate synthetic masks over these ground truth videos.

To generate a mask we apply the same forward-warp operation of Section IV-A1 to each of the clips with the target median disparity δ_0 resampled per clip so that the masks have the same shape and placement distribution as they will have at inference compared to rectangles or free-form strokes used by generic inpainting datasets. We sample δ_0 uniformly between

35 and 95 pixels per clip when building training data to make the model capable of inpainting any baseline.

One failure mode of training on these masks alone is that it learns to inpaint the background and leave the foreground untouched. To teach the model to inpaint the arm, we therefore add partial arm masks that cover a subset of the manipulator. We also apply random disparity dropout to 10% of clips and sample a dropout rate uniformly between 0.05 and 0.25.

B. StereoEngine-S: Stereo Generation w/out a G.T. Left View

StereoEngine-L requires a ground-truth left video to warp. When no such video exists, we instead generate the left and right views in simulation and synthesize both directly, conditioning on a simulated trajectory, a language prompt, and two matching reference images of the real workspace, one for the left view and one for the right, that supply the target appearance. These reference images are easy to obtain as they only require capturing a single stereo photo of the robot scene.

1) *Generating Demonstrations from Simulation:* We leverage RoboTwin [48] to generate the demonstrations. Since the cameras can be placed anywhere in the scene, we position a second virtual camera at a chosen baseline from the first and render a true stereo pair for every frame. The disparity between the two views can be directly retrieved from the simulator’s camera parameters and scene geometry so no depth estimation is needed and no disocclusions have to be filled. However, the simulation cannot provide realistic rendering since its textures, materials, and lighting differ substantially from the real world. We therefore convert each frame into a Canny edge map [44], as also done in CRAFT [13], which preserves the outlines of the arms and objects while removing the simulator’s visual appearance. This leaves the video model free to synthesize a realistic scene while still following the simulated motion. We then tile the left and right frames into a single video frame so that one generation produces both views simultaneously. Because both views are denoised together, they remain consistent with one another.

2) *Generating the Synthetic Left and Right Videos:* Let $\mathbf{V}^c$ denote the tiled Canny control video from Section IV-B1, I^{ref} the reference input, and ℓ the language prompt. Following CRAFT [13], we condition on real reference images of the workspace to guide appearance. Because we generate both views, I^{ref} consists of two matching reference images, I_L^{ref} and I_R^{ref} , captured as a single stereo photo of the workspace and tiled in the same layout as the generated frame. The language prompt ℓ is a short description of the scene and task, for example “two robot arms stacking two bowls on a wooden table.” We use Wan2.2-Fun-A14B-Control [47] as our backbone and generate the tiled stereo video by sampling from

$$p_\phi(\hat{\mathbf{V}} | \mathbf{V}^c, I^{\text{ref}}, \ell), \quad (3)$$

where $\hat{\mathbf{V}}$ is the generated video and ϕ denotes the model parameters. Unlike StereoEngine-L, which fine-tunes a Wan2.2 variant to inpaint the disoccluded regions of a warped view, here the pretrained model performs full video generation zero-shot, synthesizing the entire tiled stereo video from the

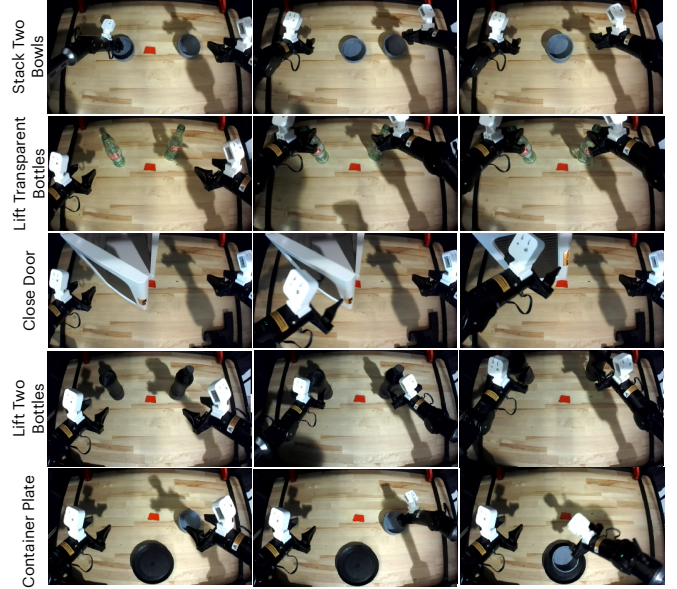


Fig. 4: **Real-world rollouts.** Each row shows a successful policy rollout by ExStereo [24] trained on StereoEngine data, from top to bottom: **Stack Two Bowls**, **Lift Transparent Bottles**, **Close Door**, **Lift Two Bottles**, and **Container Plate**. Both StereoEngine methods test **Stack Two Bowls**, while the other four tasks are tested on one of our methods. Columns show the progression of time (left to right).

conditioning inputs without any fine-tuning. The control video fixes the structure and motion, the reference image provides the real-world appearance, and the language prompt specifies the task. Splitting each generated frame gives the synthetic left and right videos $\hat{\mathbf{V}}^L$ and $\hat{\mathbf{V}}^R$ which form the stereo pair.

V. REAL-WORLD EXPERIMENTS

A. Real-World Experiment Setup

We use two AgileX PiPER robot arms and a ZED Mini 2 camera. An experienced roboticist used a Meta Quest 2 to collect teleoperation data. Because task difficulty varies, we collect or generate a different number of demonstrations for each task. For StereoEngine-L, the mid-training dataset consists of 3469 public monocular episodes lifted into stereo by StereoEngine-L and each baseline mid-trains on the same episodes converted by its own method. We then fine-tune on real teleoperated demonstrations captured with the ZED Mini 2: 200 for **Stack Two Bowls**, 50 for **Lift Transparent Bottles**, and 20 for **Close Door**. For StereoEngine-S, no real demonstrations are used and instead we generate synthetic stereo episodes per task from simulation: 200 for **Stack Two Bowls**, 50 for **Lift Two Bottles**, and 50 for **Container Plate**. We do not report simulation experiments, as our method generates stereo training data for real-world policy learning and is therefore evaluated directly on physical hardware.

We evaluate on three tasks for StereoEngine-L:

- **Stack Two Bowls:** two bowls must be stacked on top of each other in order, testing sequential manipulation.

StereoEngine-L: Left → Right Generation

Base Policy	ExStereo			
	No Mid.	SVG	Stereo Crafter	StereoEngine
Stack Two Bowls	9 / 20	7 / 20	10 / 20	13 / 20
Lift Transparent Bottles	11 / 20	10 / 20	13 / 20	18 / 20
Close Door	16 / 20	14 / 20	17 / 20	19 / 20

StereoEngine-S: Left + Right Generation

Base Policy	ExStereo		StereoVLA	
	Point Clouds	StereoEngine	Point Clouds	StereoEngine
Stack Two Bowls	0 / 20	11 / 20	0 / 20	7 / 20
Lift Two Bottles	0 / 20	13 / 20	0 / 20	9 / 20
Container Plate	0 / 20	15 / 20	0 / 20	11 / 20

TABLE II: **Real-world results.** Success rate over 20 trials (higher is better). The **Base Policy** is the stereo-conditioned policy being trained (ExStereo [24] or StereoVLA [5]), and each column is a different training condition. In **StereoEngine-L**, each column is a different mid-training condition: **No Mid.** trains the policy directly with no stereo mid-training, and the remaining columns mid-train it on stereo data from SVG [15] and StereoCrafter [14] (baselines) or our StereoEngine (highlighted in green); all conditions share the same real fine-tuning data per task. In **StereoEngine-S**, no real demonstrations are used for training, so each column is a different source of task-specific training data: simulated point clouds versus our StereoEngine synthetic stereo. Across both, StereoEngine (**Ours**) gives the best performance.

- **Lift Transparent Bottles:** two transparent glass bottles must be picked up from the table, testing perception of transparent objects.
- **Close Door:** the left arm must close the door of a fridge, testing manipulation of an articulated object.

For StereoEngine-S, we reuse the **Stack Two Bowls** task from StereoEngine-L and evaluate on two extra tasks:

- **Lift Two Bottles:** two non-transparent bottles must be picked up from the table.
- **Container Plate:** one bowl must be stacked on top of a plate.

We do not evaluate the **Lift Transparent Bottles** or **Close Door** tasks under StereoEngine-S. Articulated objects such as a fridge door require substantial per-object tuning to model in RoboTwin, and the difficulty of accurately simulating real-world contact dynamics makes their interactions hard to reproduce faithfully. In addition, current state-of-the-art video diffusion models still struggle to accurately reproduce transparent and highly specular surfaces. Here, we focus on evaluating our method under the current limitations of video generation models. We compare StereoEngine-L against two strong monocular-to-stereo baselines, mid-training ExStereo, a stereo-conditioned policy built on $\pi_{0.5}$ [2], on each one’s stereo data:

- **No Mid.:** Skips mid-training and fine-tunes on the real demonstrations directly.
- **SVG [15]:** a training-free method that uses Zeroscope to synthesize the right view.
- **StereoCrafter [14]:** a training-based method that fine-tunes Stable Video Diffusion on stereo pairs to synthesize the right view.
- **StereoEngine (Ours):** we generate the synthetic stereo dataset with StereoEngine-L.

We do not benchmark against Geometric Reciprocity [16], as it is concurrent work with no publicly available code or weights at the time of publication. We also do not compare against other stereo-conditioned policies since ExStereo already outperforms existing methods at the time of writing. Although ExStereo [24] is not publicly released at the time of publication, we obtained its implementation from its authors, who plan to release their polished version. We will also release our code to support reproducibility.

Method	PSNR↑	SSIM↑	LPIPS↓	CTC↑
StereoCrafter [14]	10.50	0.4574	0.6211	0.9807
SVG [15]	12.83	0.5137	0.5128	0.9752
StereoEngine (Ours)	15.31	0.5304	0.4696	0.9857

TABLE III: **Stereo generation quality for StereoEngine-L**, on 50 held-out real stereo clips. These clips are different from the demonstrations used for training. Each method synthesizes a right view from the same ground-truth left video. CTC measures temporal consistency across generated frames. StereoEngine achieves the best score on every metric. Here, ↑ means higher is better, and ↓ means lower is better.

Since no existing method generates *both* synthetic left and right views, we compare StereoEngine-S against a single baseline. In each case we train both ExStereo [24] and StereoVLA [5]:

- **Point Clouds:** we render ground-truth point clouds directly from the simulator and perturb every point with zero-mean isotropic Gaussian noise ($\sigma=4$ mm per axis) to approximate real-sensor imperfections, then train the policies on this simulated 3D geometry using the same number of episodes as StereoEngine-S.
- **StereoEngine (Ours):** we generate the synthetic stereo dataset with StereoEngine-S, conditioning on the Canny edges and two reference images.

We do not evaluate StereoPolicy [4] as no code or public weights are available at the time of publication.

B. Real-World Results

Real-world results are presented in Table II and qualitative rollouts are shown in Figure 4. StereoEngine demonstrates substantially improved robustness across all tasks compared to baselines. Aggregated over the three tasks, mid-training ExStereo on StereoEngine data succeeds on 50/60 trials (83.3%) versus 36/60 (60.0%) with no mid-training, with StereoCrafter in between at 40/60 (66.7%). SVG performs worst at 31/60 (51.7%), falling below no mid-training on every task. The largest gain is on **Lift Transparent Bottles**, where StereoEngine reaches 18/20 while no mid-training reaches only 11/20 and SVG only 10/20.

1) *StereoEngine-L Real-World Results:* **Stack Two Bowls:** The baselines frequently failed to grasp the bowls and closed the gripper beside or above them. With SVG mid-training the arms sometimes froze or cycled through the

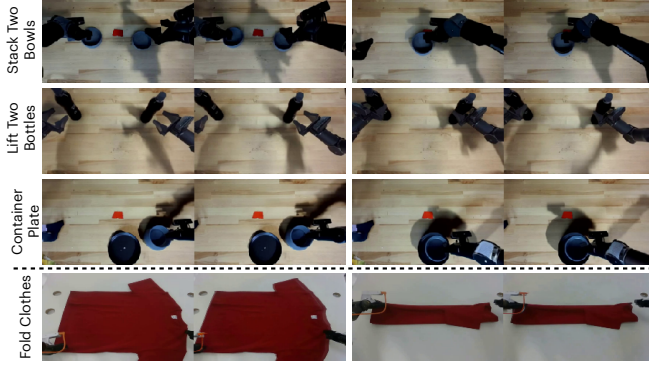


Fig. 5: **Synthetic episode generation.** The top three rows show synthetic episodes generated by StereoEngine-S, one per task, from top to bottom: **Stack Two Bowls**, **Lift Two Bottles**, and **Container Plate**. Below the dashed line, the last row (**Fold Clothes**) shows a large-scale data generation example from StereoEngine-L, where a real monocular demonstration is lifted into stereo. Within each row, adjacent columns form a left-right stereo pair, and successive pairs show the progression of time from left to right.

same motion for certain bowl placements. The baselines also often grasped the left bowl but missed the right one. Even after grasping both, they struggled to place one bowl cleanly on top of the other.

Lift Transparent Bottles: The baselines struggled to grasp the bottles because the most common failure was knocking a bottle over, since the opening to grasp is narrow and small depth errors drive the gripper into it.

Close Door: The baselines often misjudged the distance to the door and missed it, driving the arms into the fridge. The no mid-training baseline in particular reached too far to the right or too high and missed the door entirely. In a few cases the arm pushed with excessive force and tipped the fridge over, which we counted as a failure.

2) *StereoEngine-S Real-World Results:* Training the policy on ground-truth point clouds from the simulator does not transfer to the real world, as every task results in a 0% success rate. We qualitatively observe that there is a substantial sim-to-real gap, because real stereo depth contains substantial structured sensor noise that clean simulated point clouds do not exhibit. Our point cloud baseline already perturbs each point with Gaussian noise ($\sigma=4\text{mm}$) and varying this noise did not yield any meaningful improvement because simple noise does not reproduce the material-dependent dropouts, missing regions, and edge artifacts of real depth sensors. This is consistent with prior work showing that naive noise augmentation is insufficient to close the sim-to-real gap for depth and point cloud policies [49].

Container Plate: Our method occasionally fails to grasp the right bowl, and even when it does, the bowl is sometimes not placed cleanly onto the plate.

Lift Two Bottles: The most common failure was knocking a bottle over, since the opening to grasp is narrow and small depth errors drive the gripper into it.

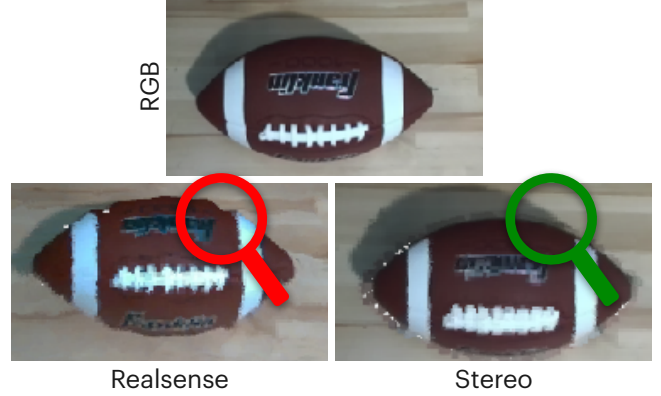


Fig. 6: **Comparison of RealSense and stereo.** **Top:** an American football captured with a standard RGB camera. **Bottom left:** the 3D point cloud from an Intel RealSense D405 depth sensor where the **red** region shows geometric deformation of the ball’s surface. **Bottom right:** the point cloud recovered by FoundationStereo [25] from a stereo pair captured with a ZED Mini 2 where the same **green** region shows minimal deformation of the object.

C. Generation Quality Analysis

Generation Quality. Since no stereo benchmark exists for robotic domains, we build our own evaluation set by collecting play data on our real stereo rig, giving paired real left and right videos to compare against. For StereoEngine-L, we run StereoEngine on the real left video and compare the generated right view to the true captured right view. As Table III shows, StereoEngine beats both StereoCrafter and SVG on every metric—PSNR, SSIM [50], LPIPS [51], and CLIP Temporal Consistency (CTC) [52], which measures frame-to-frame CLIP feature similarity for temporal coherence, including a 2.48 dB PSNR gain over the strongest baseline. Absolute scores are lower than on standard stereo benchmarks because our ground truth is a real second camera with large disparities and imperfect rectification, unlike the perfectly rectified content those baselines are typically evaluated on.

VI. DISCUSSION

As Figure 6 shows, stereo recovers clean geometry on a specular object where an active depth sensor (Intel RealSense D405) deforms the surface, motivating the stereo data StereoEngine generates. StereoEngine provides two independent methods for producing stereo data, based on whether real monocular footage of the task exists. **StereoEngine-L** applies when a monocular dataset already exists, and we highlight two use cases: (1) mid-training a stereo policy on large-scale monocular data lifted into stereo, and (2) adapting an existing monocular dataset for a given task to a stereo policy. **StereoEngine-S** covers the analogous cases when no real monocular video is available: (1) building a large-scale mid-training dataset directly from a simulator, and (2) fine-tuning a stereo policy on a specific task by generating targeted stereo demonstrations.

VII. CONCLUSION

We present StereoEngine, a video diffusion framework for synthesizing stereo data for bimanual manipulation without

training on real stereo correspondences. Given an existing monocular demonstration, we warp the left view and inpaint the disoccluded regions to produce the corresponding right view. When no real video exists, we instead generate both views jointly from a simulated trajectory. Across real-world tasks on AgileX PiPER arms, StereoEngine data improve stereo-conditioned policies through both large-scale mid-training and task-specific fine-tuning, outperforming monocular-to-stereo baselines and simulated point clouds. We hope this enables exciting research on stereo-conditioned foundation policies trained at scale.

REFERENCES

- [1] A. . Team, “Aloha 2: An enhanced low-cost hardware for bimanual teleoperation,” 2024. [Online]. Available: <https://aloha-2.github.io/>
- [2] P. Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi, C. Finn, and et al., “ $\pi_{0.5}$: a Vision-Language-Action Model with Open-World Generalization,” in *Conference on Robot Learning (CoRL)*, 2025.
- [3] P. Intelligence et al., “ $\pi_{0.7}$: a steerable generalist robotic foundation model with emergent capabilities,” 2026. [Online]. Available: <https://arxiv.org/abs/2604.15483>
- [4] E. Han, Y. Jiang, Y. Wang, H. Xiao, H. Huang, J. Xie, J. Wu, L. Fei-Fei, and R. Zhang, “StereoPolicy: Improving robotic manipulation policies via stereo perception,” 2026. [Online]. Available: <https://arxiv.org/abs/2605.09989>
- [5] S. Deng, M. Yan, Y. Zheng, J. Su, W. Zhang, Y. Zeng, X. Zhao, H. Cui, Z. Zhang, and H. Wang, “StereoVLA: Enhancing Vision-Language-Action Models with Stereo Vision,” in *Robotics: Science and Systems (RSS)*, 2026.
- [6] O. X.-E. Collaboration, “Open X-Embodiment: Robotic Learning Datasets and RT-X Models,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.
- [7] H. Fang et al., “Molmoact2: Action reasoning models for real-world deployment,” 2026. [Online]. Available: <https://arxiv.org/abs/2605.02881>
- [8] S. Wu et al., “Robocoin: An open-sourced bimanual robotic data collection for integrated manipulation,” 2026. [Online]. Available: <https://arxiv.org/abs/2511.17441>
- [9] A. Khazatsky et al., “Droid: A large-scale in-the-wild robot manipulation dataset,” in *Robotics: Science and Systems (RSS)*, 2024.
- [10] A. Mandlkar, S. Nasiriany, B. Wen, I. Akinola, Y. Narang, L. Fan, Y. Zhu, and D. Fox, “MimicGen: A Data Generation System for Scalable Robot Learning using Human Demonstrations,” in *Conference on Robot Learning (CoRL)*, 2023.
- [11] Z. Xue, S. Deng, Z. Chen, Y. Wang, Z. Yuan, and H. Xu, “Demogen: Synthetic demonstration generation for data-efficient visuomotor policy learning,” *arXiv preprint arXiv:2502.16932*, 2025.
- [12] J. Yu, L. Fu, H. Huang, K. El-Refai, R. A. Ambrus, R. Cheng, M. Z. Irshad, and K. Goldberg, “Real2render2real: Scaling robot data without dynamics simulation or robot hardware,” 2025. [Online]. Available: <https://arxiv.org/abs/2505.09601>
- [13] J. Chen, I.-C. A. Liu, G. Sukhatme, and D. Seita, “CRAFT: Video Diffusion for Bimanual Robot Data Generation,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2026.
- [14] S. Zhao, W. Hu, X. Cun, Y. Zhang, X. Li, Z. Kong, X. Gao, M. Niu, and Y. Shan, “StereoCrafter: Diffusion-based generation of long and high-fidelity stereoscopic 3d from monocular videos,” *arXiv preprint arXiv:2409.07447*, 2024.
- [15] P. Dai, F. Tan, Q. Xu, D. Futschik, R. Du, S. Fanello, X. Qi, and Y. Zhang, “SVG: 3d stereoscopic video generation via denoising frame matrix,” in *International Conference on Learning Representations (ICLR)*, 2025.
- [16] J. Lu and K. Han, “Geometric Reciprocity: Unlocking Self-Supervision for Stereoscopic Video Generation,” in *International Conference on Machine Learning (ICML)*, 2026.
- [17] Z. Lv, Y. Long, C. Huang, C. Li, C. Lv, H. Ren, and D. Zheng, “SpatialDreamer: Self-supervised Stereo Video Synthesis from Monocular Input,” *arXiv preprint arXiv:2411.11934*, 2025.
- [18] S. Yu, Y. Chen, Z. Qi, Z. Xie, Y. Wang, L. Wang, Y. Shan, and H. Lu, “Mono2stereo: A benchmark and empirical study for stereo conversion,” 2025. [Online]. Available: <https://arxiv.org/abs/2503.22262>
- [19] F. Qiao, Z. Xiong, E. Xing, and N. Jacobs, “Towards open-world generation of stereo images and unsupervised matching,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.
- [20] K. Xing, X. Jin, L. Li, Y. Yin, H. Liang, G. Luo, C. Fang, J. Wang, K. N. Plataniotis, Y. Zhao, and Y. Wei, “StereoWorld: Geometry-aware monocular-to-stereo video generation,” *arXiv preprint arXiv:2512.09363*, 2025.
- [21] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, “NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis,” in *European Conference on Computer Vision (ECCV)*, 2020.
- [22] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, “3d gaussian splatting for real-time radiance field rendering,” in *ACM Transactions on Graphics*, 2023.
- [23] X. Huang, A. K. Singh, F. Dubost, C. N. Vasconcelos, S. Khattar, L. Shi, C. Theobalt, C. Ozireli, and G. Singh, “Restereo: Diffusion stereo video generation and restoration,” 2025. [Online]. Available: <https://arxiv.org/abs/2506.06023>
- [24] I.-C. A. Liu, J. Chen, G. Sukhatme, and D. Seita, “ExStereo: Lifting 2D Vision-Language-Action Models to 3D with Explicit Stereo Representations,” 2026.
- [25] B. Wen, M. Trepte, J. Aribido, J. Kautz, O. Gallo, and S. Birchfield, “FoundationStereo: Zero-shot stereo matching,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.
- [26] L. Bartolomei, F. Tosi, M. Poggi, and S. Mattoccia, “Stereo anywhere: Robust zero-shot deep stereo matching even where either stereo or mono fail,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.
- [27] X. Guo, C. Zhang, Y. Zhang, R. Wang, D. Nie, W. Zheng, M. Poggi, H. Zhao, M. Ye, Q. Zou, and L. Chen, “Stereo anything: Unifying stereo matching with large-scale mixed data,” *arXiv preprint arXiv:2411.14053*, 2024.
- [28] J. Xie, R. Girshick, and A. Farhadi, “Deep3d: Fully automatic 2d-to-3d video conversion with deep convolutional neural networks,” 2016. [Online]. Available: <https://arxiv.org/abs/1604.03650>
- [29] L. Wang, J. R. Frisvad, M. Bo Jensen, and S. A. Bigdeli, “StereoDiffusion: Training-Free Stereo Image Generation Using Latent Diffusion Models,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- [30] J. Shi, Q. Wang, Z. Li, W. Cui, R. Idoughi, and P. Wonka, “DissolveStereo: Coarse depth injection for zero-shot stereo video generation,” *arXiv preprint arXiv:2411.14295*, 2026.
- [31] S. Wei, H. Geng, J. Chen, C. Deng, C. Wenbo, C. Zhao, X. Fang, L. Guibas, and H. Wang, “D3roma: Disparity diffusion-based depth sensing for material-agnostic robotic manipulation,” in *Conference on Robot Learning (CoRL)*, 2024.
- [32] K. Bai, H. Zeng, L. Zhang, Y. Liu, H. Xu, Z. Chen, and J. Zhang, “ClearDepth: Enhanced stereo perception of transparent objects for robotic manipulation,” 2024.
- [33] J. Yu, K. Hari, K. El-Refai, A. Dalil, J. Kerr, C.-M. Kim, R. Cheng, M. Z. Irshad, and K. Goldberg, “Persistent Object Gaussian Splat (POGS) for Tracking Human and Robot Manipulation of Irregularly Shaped Objects,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2025.
- [34] Y. Zou, C. Shi, W. Yu, H. Xue, J. Lv, Y. Pan, C. Wen, and C. Lu, “Activeglasses: Learning manipulation with active vision from ego-centric human demonstration,” 2026. [Online]. Available: <https://arxiv.org/abs/2604.08534>
- [35] J. Jang, S. Ye, Z. Lin, J. Xiang, J. Björck, Y. Fang, F. Hu, S. Huang, K. Kundalia, Y.-C. Lin et al., “Dreamgen: Unlocking generalization in robot learning through video world models,” *arXiv preprint arXiv:2505.12705v2*, 2025.
- [36] S. Gao et al., “Dreamdojo: A generalist robot world model from large-scale human videos,” *arXiv preprint arXiv:2602.06949*, 2026.
- [37] S. Ye et al., “World action models are zero-shot policies,” 2026. [Online]. Available: <https://arxiv.org/abs/2602.15922>
- [38] N. Agarwal et al., “Cosmos world foundation model platform for physical ai,” 2025. [Online]. Available: <https://arxiv.org/abs/2501.03575>
- [39] J. Ye, R. Xue, B. Van Hoorick, R. Li, H. Rajaprakash, P. Tokmakov, M. Z. Irshad, V. Guizilini, and Y. Wang, “RoboDream: Compositional World Models for Scalable Robot Data Synthesis,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2026.
- [40] T. Yu, T. Xiao, A. Stone, J. Tompson, A. Brohan, S. Wang, J. Singh, C. Tan, D. M. J. Peralta, B. Ichter, K. Hausman, and F. Xia, “Scaling Robot Learning with Semantically Imagined Experience,” in *Robotics: Science and Systems (RSS)*, 2023.
- [41] S. Yang, W. Yu, J. Zeng, J. Lv, K. Ren, C. Lu, D. Lin, and J. Pang, “Novel demonstration generation with gaussian splatting enables robust one-shot manipulation,” in *Robotics: Science and Systems (RSS)*, 2025.
- [42] I.-C. A. Liu, J. Chen, G. Sukhatme, and D. Seita, “D-CODA: Diffusion for Coordinated Dual-Arm Data Augmentation,” in *Conference on Robot Learning (CoRL)*, 2025.
- [43] J. Chen, I.-C. A. Liu, G. Sukhatme, and D. Seita, “ROPA: Synthetic Robot Pose Generation for RGB-D Bimanual Data Augmentation,” in *International Conference on Robotics and Automation (ICRA)*, 2026.
- [44] J. Canny, “A Computational Approach to Edge Detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, 1986.
- [45] S. Chen, H. Guo, S. Zhu, F. Zhang, Z. Huang, J. Feng, and B. Kang, “Video depth anything: Consistent depth estimation for super-long videos,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.
- [46] S. Niklaus and F. Liu, “Softmax splatting for video frame interpolation,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [47] T. Wan et al., “Wan: Open and Advanced Large-Scale Video Generative Models,” *arXiv preprint arXiv:2503.20314*, 2025.
- [48] T. Chen et al., “Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation,” in *International Conference on Machine Learning (ICML)*, 2026.
- [49] Y. Zhou, W. Cui, W. Liu, G. Chen, H. Li, and D. Zhao, “Diffudegrasp: Diffusion-based depth noise modeling empowers sim2real robotic grasping,” 2025.
- [50] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.
- [51] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, “The unreasonable effectiveness of deep features as a perceptual metric,” in *CVPR*, 2018.
- [52] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, and et al., “Learning Transferable Visual Models From Natural Language Supervision,” in *International Conference on Machine Learning (ICML)*, 2021.