SpatialVAM: Spatial-Aware Multi-View Video Diffusion as a Data-Efficient Robot Policy

Peiyan Li[1,2,*] **Yixiang Chen**[1,2,*] **Yuan Xu**[1,2] **Jiabing Yang**[1,2]
Xiangnan Wu[1,2] **Jun Guo**[4] **Nan Sun**[4] **Long Qian**[5]
Xinghang Li[4] **Xin Xiao**[6] **Minghui Zhang**[1] **Jing Liu**[3]
Nianfeng Liu[3] **Tao Kong**[4,†] **Yan Huang**[1,2,3,†] **Liang Wang**[1,2]

[1]**New Laboratory of Pattern Recognition (NLPR),**
Institute of Automation, Chinese Academy of Sciences
[2]**School of Artificial Intelligence, University of Chinese Academy of Sciences**
[3]**FiveAges** [4]**Tsinghua University** [5]**Xi'an Jiaotong University** [6]**Wuhan University**

https://spatialvam.github.io/

Abstract

Robotic manipulation requires understanding both the 3D spatial structure of the environment and its temporal evolution, yet most existing policies neglect one or both aspects. They often rely on 2D visual observations or backbones pretrained on static image–text pairs, which leads to high data requirements and limited comprehension of environment dynamics. To address this, we introduce SpatialVAM, **the first 3D Video Action Model** that simultaneously predict spatial-aware multi-view heatmap videos and RGB videos. Our key insight is that this design naturally injects 3D information into video foundation models while aligning the representation format between video pretraining and action finetuning. Extensive experiments demonstrate that SpatialVAM enables **data-efficient, robust, generalizable, and interpretable** manipulation. With only ten demonstration trajectories and no additional pretraining, SpatialVAM handles challenging long-horizon and contact-rich tasks, generalizes to out-of-distribution settings, and predicts realistic future videos. Evaluations on Meta-World (22%↑), RoboCasa (15%↑) and real-world robotic platforms (16%↑) show that SpatialVAM consistently outperforms other video action models, vision language action models and 3D-based policies, establishing a new state-of-the-art in data-efficient multi-task manipulation.

1 Introduction

Recent years have witnessed substantial progress in robotics driven by large-scale representation learning, with Vision Language Action (VLA) models and Video Action Models (VAM) emerging as two representative paradigms [1–7]. Despite their impressive performance across diverse tasks and embodiments, existing approaches still fall short of a unified understanding of the environment's *3D spatio-temporal state*: they often lack either temporal modeling of how the world evolves or spatial modeling of the environment's 3D structure.

On the one hand, VLA-based methods commonly adopt backbones pretrained on static image–text pairs [8–11]. However, such pretraining does not capture how scenes evolve over time, limiting a policy's ability to anticipate future states. This contrasts with insights from cognitive science, which suggest that effective action selection relies on anticipating how the environment will change in response to actions, rather than reacting to instantaneous observations [12]. On the other hand, although VAMs inherit the predictive capabilities of video foundation models and naturally model future observations, they predominantly operate in 2D image space and lack explicit modeling of 3D geometric structure. This creates an observation–action mismatch: observations are represented as 2D

*Equal contribution
†Corresponding author

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

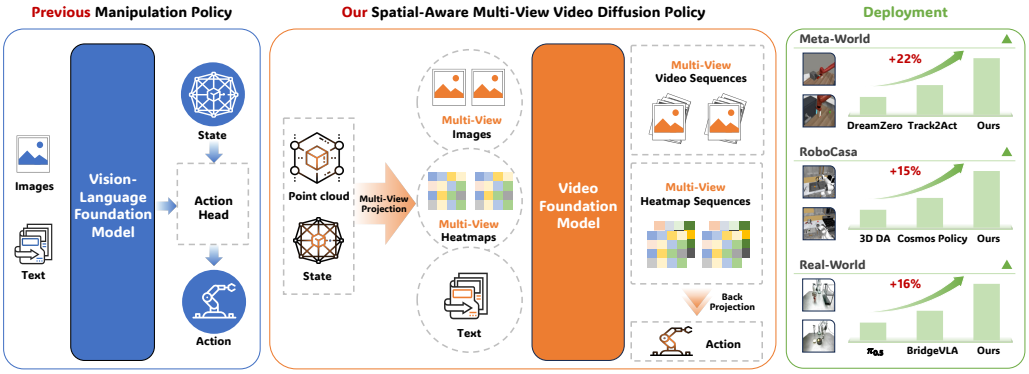

Figure 1: **Overview.** We introduce SpatialVAM, a *spatial-aware multi-view video diffusion policy* that jointly models spatial structure and temporal evolution. Compared to prior manipulation policies, our approach: (1) processes *spatial-aware multi-view images* rather than independent 2D views; (2) represents robot actions as *multi-view heatmaps*, aligning the action space with video pretraining representations; and (3) leverages a *video foundation model*, instead of a vision–language backbone, to jointly model future RGB and heatmap sequences. As a result, SpatialVAM achieves state-of-the-art on Meta-World, RoboCasa and real-world benchmarks, outperforming vision language action models, video action models and 3D-based policies.

videos, while robot actions are executed in 3D physical space. As a result, policies must compensate for this gap with large amounts of training data.

These observations motivate the following question: **Can we design a 3D World-Action Model that incorporates 3D structural priors while explicitly modeling environmental dynamics?** In this paper, we take a step toward this goal. As illustrated in Fig. 1, we propose SpatialVAM, a spatially aware multi-view video diffusion policy that **aligns the representation format of video pretraining with that of action finetuning**. Specifically, SpatialVAM: (1) adopts spatially aware multi-view projections to implicitly encode 3D structure; (2) transforms colored point clouds into multi-view RGB images and represents target positions as multi-view heatmaps using the same projection mechanism; and (3) leverages a unified video foundation model to jointly predict future video frames and heatmap sequences, which are subsequently decoded into continuous action chunks. By representing 3D information within this aligned video-based format, SpatialVAM substantially reduces the gap between perception and control, leading to strong empirical performance.

Extensive experiments on Meta-World [13], RoboCasa [14], and real-world robotic platforms show that SpatialVAM consistently outperforms state-of-the-art VAMs, VLA models, and 3D-based policies. These results highlight key strengths of SpatialVAM: it is **data-efficient**, acquiring challenging skills from as few as 10 demonstrations without pretraining; **robust**, maintaining strong performance even with a single denoising step; **generalizable**, handling unseen backgrounds, object categories, heights, and lighting conditions; and **interpretable**, as predicted videos faithfully reflect executed actions, enabling users to visually assess action safety prior to deployment.

Our contributions are threefold: (i) **Concept and insight:** we analyze key limitations of existing policies and introduce, to our knowledge, the first framework leveraging video foundation models for a 3D World-Action Model; (ii) **Method:** we propose SpatialVAM, a multi-view video diffusion policy that captures spatial structure and temporal dynamics and produces robot actions; and (iii) **Experiments:** we perform extensive simulation and real-world evaluations to demonstrate SpatialVAM's advantages over existing manipulation policies.

2 Related Work

2.1 Vision Language Action Models

Vision Language Action (VLA) models have recently emerged as a dominant paradigm for robot manipulation [1–4, 15, 5, 16–20]. These models typically condition on 2D visual observations and language instructions, leverage large Vision–Language Models (VLMs) for representation learning, and decode continuous actions via flow matching [21], diffusion [22], or fast tokenization [23].

While VLAs demonstrate strong generalization across tasks and embodiments [3, 2, 1], they often require large-scale robot datasets and primarily emphasize action imitation, with limited explicit modeling of environment dynamics. Our approach addresses these limitations by implicitly encoding 3D-related information and integrating environmental dynamics modeling, ultimately achieving significantly higher data efficiency.

2.2 Video Action Models

Video action models have been regarded as a promising way for anticipatory control in robotic manipulation. Existing approaches can be broadly categorized into those that train video prediction and action generation separately, and those that train them jointly.

Separate training. Many methods adopt a two-stage pipeline, first learning to predict future visual observations and then mapping visual representations to actions [24–27]. Variants differ in the intermediate representations used, including RGB videos [24, 25], human demonstration videos [28], optical flow [29], and 2D point trajectories [30].

Joint training. More recent work jointly predicts future videos and actions within a unified framework [31–37, 6, 38–41], enabling tighter perception–action coupling. Our method also follows this paradigm, but differs on implicitly incorporating 3D structural priors and alignment between action finetuning and video pretraining.

2.3 3D Structural Priors for Manipulation

Incorporating 3D structural priors into manipulation policies has been studied through several representations. Point-cloud-based methods directly encode 3D geometry for action prediction [42–45], while other approaches use the knowledge in 3D foundation models as an implicit 3D proxy [46, 47]. More recently, multi-view image representations, combined with known camera transformations, have emerged as another effective form of 3D priors [48, 49].

Our work builds on this multi-view paradigm but differs in two aspects. First, existing approaches rely on vision-language models or task-specific networks, whereas SpatialVAM employs a video foundation model to jointly capture spatial structure and temporal dynamics. Second, prior 3D-aware methods predict only key poses or waypoints, while SpatialVAM directly predicts continuous actions and future observations, enabling more general manipulation skills.

3 Method

As illustrated in Fig. 2 (a), our SpatialVAM consists of three main components, detailed below. In Sec. 3.1, we present the multi-view projection module that transforms colored point clouds into multi-view images and heatmap representations. In Sec. 3.2, we describe the multi-view video diffusion formulation and the corresponding model architecture. In Sec. 3.3, we detail how final actions are decoded from the predicted heatmaps and latent representations. Finally, in Sec. 3.4, we introduce the training and inference procedures of SpatialVAM.

3.1 Multi-View Projection

SpatialVAM takes point clouds and the robot end-effector pose as raw inputs. Since redundant point cloud information can negatively affect subsequent model training [43], we first crop the input point clouds to the workspace boundaries, defined as a cubic volume of $1\,\mathrm{m}^3$.

Instead of processing the cropped point clouds directly, we render them from three fixed virtual viewpoints using orthographic projection [48, 49]. The viewing directions need not be mutually orthogonal and can be flexibly chosen, as long as the target regions are observed as clearly as possible. Although adaptively selecting projection views is a promising direction [50], we leave this exploration to future work. The advantages of virtual-view rendering over directly using the physical-camera images are discussed in Appendix B.1.

For the end-effector pose, rather than encoding it with a separate module, we transform it into multiple heatmaps corresponding to the projection views. After colorization, these heatmaps can be directly

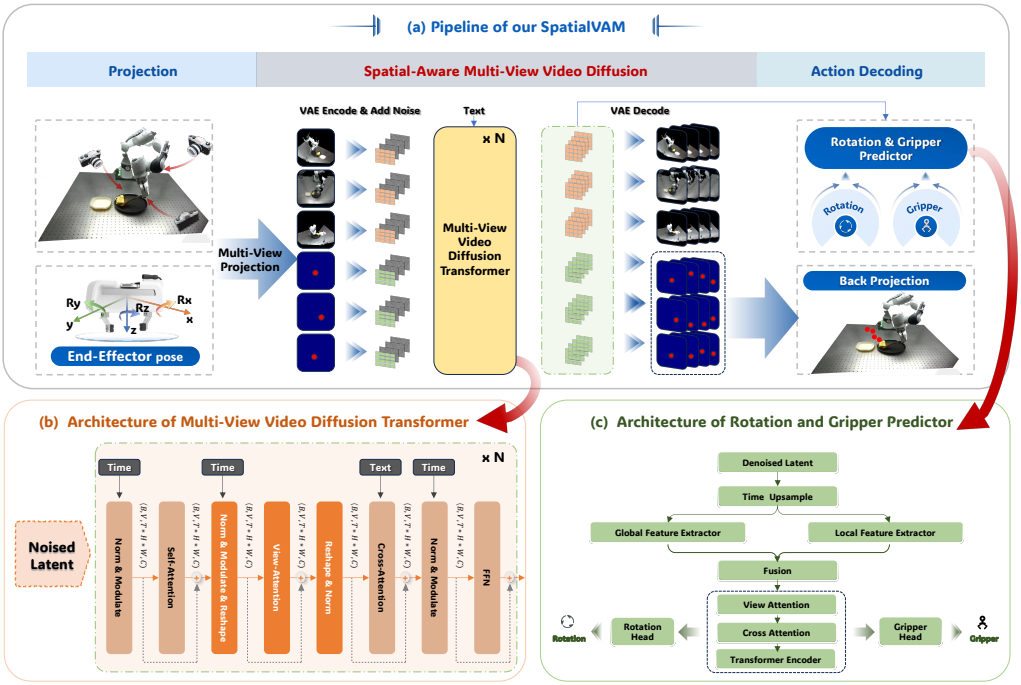

Figure 2: **Overview of SpatialVAM's pipeline**. (a) Point clouds and the end-effector pose are projected into spatial-aware multi-view RGB images and heatmaps, which are encoded to jointly predict future multi-view RGB and heatmap videos via a video diffusion model. Predicted heatmaps are back-projected to recover 3D end-effector positions. (b) The multi-view video diffusion transformer augments a pretrained backbone with view-attention to enable cross-view interaction. (c) A lightweight action decoder predicts end-effector rotation and gripper states from the denoised video latents. Final action chunks combine the predicted positions, rotations, and gripper states.

processed by the VAE encoder of the Video Foundation Model (VFM). Our intuition is that leveraging off-the-shelf pretrained components reduces the representation gap and exploits prior knowledge.

To construct the heatmaps, we first identify the end-effector's corresponding pixel locations on each projection view. Centered at these pixels, we generate Gaussian heatmaps H_i^t with spatial truncation:

$$H_i^t(\mathbf{x}) = \begin{cases} p_i^t(\mathbf{x}), & \text{if } p_i^t(\mathbf{x}) \geq \tau, \\ 0, & \text{otherwise}, \end{cases} \tag{1}$$

where t denotes the timestep, $i \in \{1, 2, ..., n\}$ indexes the view, and $\mathbf{x} = (u, v)$ denotes the pixel location. The untruncated Gaussian distribution is defined as

$$p_i^t(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x} - \widehat{\mathbf{x}}_i^t\|^2}{2\sigma^2}\right), \tag{2}$$

where $\widehat{\mathbf{x}}_i^t$ is the projected pixel location of the end effector at timestep t on view i, σ controls the spatial spread of the heatmap, and τ is a probability threshold.

At this stage, we obtain the current spatial-aware multi-view RGB images and the corresponding heatmap images, which serve as conditioning inputs for the multi-view video diffusion transformer to jointly predict future multi-view RGB videos and heatmap videos.

3.2 Multi-View Video Diffusion

Our multi-view video diffusion transformer is built upon Wan2.2 [51], a 5B-parameter video foundation model pretrained on single-view videos. For the multi-view setting, we augment each Diffusion Transformer (DiT) block [52] with a view-attention module [53], as illustrated in Fig. 2 (b).

During the forward pass, the multi-view RGB and heatmap images are encoded into latents by the pretrained VAE encoder. These latents are then concatenated along the view dimension, followed by

patchifying and flattening across the spatial and temporal dimensions. This yields token sequences of shape $(B, V, T \times H \times W, C)$, where B, V, T, H, W, and C are the batch size, number of views, time length, height, width, and channel count, respectively.

To accommodate the view-attention module, the token sequences are reshaped to $(B, T, V{\times}H{\times}W, C)$, enabling explicit interactions across views at each timestep. After applying view attention, the tokens are reshaped back to their original layout and processed by the remaining transformer layers. All other components follow the original Wan2.2 architecture to preserve the pretrained knowledge.

The transformer is trained to predict the added noise, enabling multi-step denoising to recover clean latent representations. The resulting latents are decoded by the VAE decoder to generate future multi-view RGB and heatmap videos, which are subsequently used for action decoding. We provide the full formulation and additional details of the diffusion process in Appendix A.

3.3 Action Decoding

After obtaining the predicted heatmap sequences, we back-project the peak locations of the three heatmaps at each timestep into a 3D position in the workspace using known camera parameters. Details of the projection and back-projection procedures are provided in Appendix B. By back-projecting heatmaps across all predicted timesteps, we recover a continuous 3D end-effector trajectory.

For rotation and gripper prediction, as illustrated in Fig. 2 (c), we take the denoised latent representations as input. Since the latents are temporally compressed by the VAE, we first upsample them along the temporal dimension. We then employ two convolutional networks to extract global features from the entire latent representation and local features centered around heatmap peak locations, respectively. These features are fused and further aggregated along the view dimension.

The latent corresponding to the first frame serves as the conditioning latent. We apply cross-attention between the predicted latents and the conditioning latent to incorporate conditioning information. The resulting representations are encoded by a lightweight four-layer transformer, followed by two separate MLP heads to predict rotation and gripper actions. Rotation and gripper states are discretized, and we predict their changes at each future timestep relative to the conditioning frame.

3.4 Training & Inference

Our SpatialVAM consists of two trainable modules: (i) a 5B multi-view video diffusion transformer for predicting end-effector positions, and (ii) a lightweight 170M rotation & gripper predictor.

Training. During training, we apply SE(3) augmentations to the input point clouds and end-effector poses before projecting them into multi-view RGB images and heatmaps. These representations are encoded by a shared VAE encoder and concatenated along the view dimension.

To train the multi-view video diffusion transformer, Gaussian noise is added to future-frame latents, with the noise magnitude determined by randomly sampled diffusion timesteps. The model is then trained to predict the noise using an MSE loss. Since we predict both videos and heatmap sequences, the total training loss for the multi-view video diffusion transformer consists of two components:

$$L_{diff} = \lambda L_{vid} + (1 - \lambda) L_{heat}, \tag{3}$$

where L_{vid} is the diffusion loss for the video sequences, L_{heat} is the diffusion loss for the heatmap sequences, and λ is the weight applied to the video diffusion loss (see Sec. 4.4 for an analysis of key parameters' robustness). To reduce computational cost, we adopt LoRA fine-tuning. While we also experimented with full fine-tuning (see Sec. 4.5), we observed no performance gains.

For training the rotation and gripper predictor, we use ground-truth video latents and heatmap latents as inputs. To enhance robustness, we inject a small amount of random noise into the ground-truth latents. The training targets include the end-effector's rotation and gripper state. Rotation is represented using Euler angles, discretized into 72 bins, while the gripper state is modeled as a binary variable. The module is trained using cross-entropy loss, formulated as:

$$L_{pred} = L_{rol} + L_{pit} + L_{yaw} + L_{gri}, \tag{4}$$

where L_{rol}, L_{pit}, L_{yaw} are Euler-angle losses and L_{gri} is the gripper loss.

Inference. During inference, we project the point cloud and end-effector pose at the current timestep into multi-view RGB images and heatmaps, which are encoded to obtain the conditioning latent. After obtaining the predicted clean latents, we decode them through two parallel branches. One branch decodes the latents into multi-view heatmap sequences, which are back-projected to recover 3D position predictions. The other branch feeds the latents into the rotation & gripper predictor to estimate rotation and gripper states. By combining the predicted positions, rotations, and gripper states, we obtain action chunks that are subsequently executed by the robot controller. Additional implementation details and hyperparameters are provided in Appendix C.

4 Experiments

We conduct extensive experiments in both simulation and real-world settings to evaluate the effectiveness of SpatialVAM. Our evaluation is designed to address several key questions: whether SpatialVAM outperforms other Video Action Models (**Q1**), 3D-based policies (**Q2**), and Vision-Language-Action models (**Q3**); whether it generalizes to unseen scenarios (**Q4**); remains robust to changes in deployment conditions and modeling parameters (**Q5**); benefits from each of its architectural components (**Q6**); and produces videos that faithfully reflect executed actions to facilitate safer deployment (**Q7**).

4.1 Meta-World Experiments

Setup and Baselines Meta-World [13] is a standardized tabletop manipulation benchmark with a simulated Sawyer arm. We train SpatialVAM on seven tasks with only five expert demonstrations per task (35 total, a low-data regime), and running 25 random-initialization trials per task with a 600-step success horizon for evaluation. For baselines, we compare our method against standard behavioral-cloning and video-prediction-based baselines. BC-Scratch and BC-R3M [54] are multi-task behavioral-cloning baselines built on a ResNet-18 visual encoder and a CLIP text encoder, where BC-R3M is initialized with R3M-pretrained weights. Diffusion Policy (DP) [22] models action sequences via a diffusion-based formulation. Our video-prediction-based baselines include UniPi [26], AVDC [29], Track2Act [30], and DreamZero [6]. UniPi, AVDC, and Track2Act predict future RGB frames, optical flow, and 2D point trajectories, respectively, and then infer actions from these predicted visual representations. DreamZero is a state-of-the-art video action model that also adopts Wan [51] as its backbone, jointly predicting future RGB videos and the corresponding actions. Additional details on these baselines are provided in Appendix F.

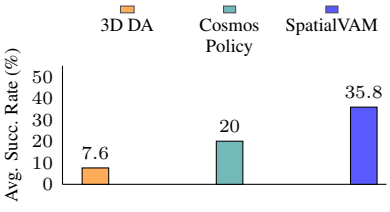

Figure 3: RoboCasa results (10 demos/task, 50 rollouts).

Results. As shown in Tab. 1, standard behavioral cloning methods perform poorly in the low-data regime, achieving only 26.2%, 35.4%, and 37.7% success rates for BC-Scratch, BC-R3M, and DP, respectively. In contrast, video-prediction-based methods yield substantially stronger performance, with AVDC, Track2Act, and DreamZero reaching 58.9%, 67.4%, and 61.1%, respectively. SpatialVAM outperforms all baselines on five out of seven tasks and achieves the best overall average success rate of 89.1% using only 5 demonstrations per task. These results suggest three main conclusions. First, video prediction is highly beneficial for data-efficient manipulation learning. Second, our multi-view video diffusion formulation built on a video foundation model is more effective than prior video-prediction-based approaches. Third, 3D-aware multi-view prediction plays a critical role: although DreamZero uses the same video foundation model as our method, it performs substantially worse. Together, these findings provide a direct answer to **Q1**.

Table 1: **Success rates on seven Meta-World tasks under a low-data regime (5 demonstrations per task).** Each entry reports the number of successful rollouts out of 25 trials. SpatialVAM achieves the highest average success rate, outperforming all baselines.

Method	Meta-World Tasks							Avg. Succ. (%) ↑
	D-Open	D-Close	Btn	Btn-Top	Fct-Cls	Fct-Open	Handle	
UniPi[26]	0/25	9/25	3/25	0/25	1/25	3/25	4/25	11.40
BC-Scratch[54]	6/25	9/25	9/25	3/25	5/25	5/25	9/25	26.20
BC-R3M[54]	1/25	15/25	9/25	1/25	6/25	17/25	13/25	35.40
DP[22]	12/25	12/25	10/25	5/25	6/25	15/25	6/25	37.70
AVDC[29]	18/25	23/25	15/25	6/25	14/25	6/25	21/25	58.90
DreamZero[6]	0/25	11/25	23/25	3/25	**20/25**	**25/25**	**25/25**	61.10
Track2Act[30]	22/25	19/25	14/25	10/25	12/25	22/25	19/25	67.40
SpatialVAM (Ours)	**25/25**	**25/25**	**25/25**	**24/25**	8/25	24/25	**25/25**	**89.10**

4.2 RoboCasa Experiments

To further evaluate the effectiveness of SpatialVAM, we conduct experiments on ten kitchen manipulation tasks from RoboCasa [14], a simulation benchmark built on the MuJoCo stack. RoboCasa allows users to procedurally generate diverse kitchen environments with varying styles, furniture configurations, and object instances. It provides the Franka Emika Panda robot as the default manipulation embodiment, with the end-effector controlled in 6-DoF Cartesian delta space.

We compare SpatialVAM against two baselines on this benchmark. First, Cosmos Policy [7], a representative Video Action Model that formulates both action and value prediction as video diffusion by repeating them as images. While this design helps reduce the gap between video pretraining and action finetuning, it does not incorporate any 3D structural priors. Second, 3D Diffuser Actor [55], which integrates 3D input priors with diffusion-based action modeling. Although it utilizes 3D information, it does not leverage a video foundation model and lacks the ability to predict future environmental evolution.

For each task, we train with only 10 demonstrations and evaluate performance over 50 rollout trials. As shown in Fig. 3, SpatialVAM outperforms both baselines by a substantial margin, further demonstrating its effectiveness and exceptional data efficiency. These results provide additional evidence in support of **Q1** and **Q2**.

4.3 Real-World Experiments

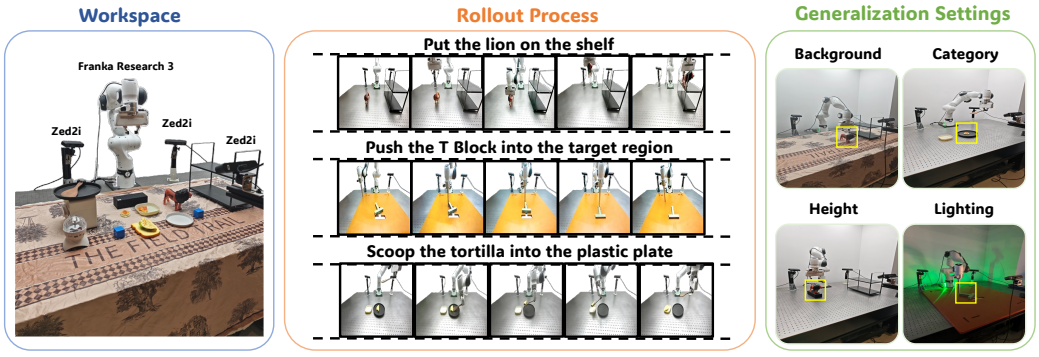

Figure 4: **Real-world experimental setup and tasks.** We evaluate SpatialVAM on three manipulation tasks using a Franka Research 3 robot with three ZED2i cameras. We further assess generalization under variations in background, object height, lighting, and object category.

Setup. As shown in Fig. 4, our real-world platform consists of a Franka Research 3 arm with a standard parallel-jaw gripper, observed by three static ZED 2i depth cameras. We calibrate the cameras with respect to the robot base frame and transform their RGB-D observations into colored point clouds as input to SpatialVAM.

Table 2: **Real-world manipulation results under limited demonstrations.** Success rates over 10 trials per task on three basic and four unseen tasks. All methods use 10 expert trajectories.

Method	Basic Tasks			Unseen Tasks				Avg. Succ. (%)
	Put Lion	Push-T	Scoop Tort.	Put-B	Put-H	Push-L	Scoop-C	
DP3[43]	0/10	0/10	0/10	0/10	0/10	0/10	0/10	0.00
$\pi_{0.5}$[2]	1/10	0/10	0/10	0/10	0/10	0/10	0/10	1.40
UVA[32]	2/10	0/10	0/10	1/10	1/10	0/10	0/10	5.70
BridgeVLA[49]	9/10	0/10	4/10	**8/10**	**7/10**	0/10	1/10	41.42
SpatialVAM (Ours)	**10/10**	**4/10**	**7/10**	5/10	6/10	**3/10**	**5/10**	**57.10**

We evaluate SpatialVAM on three base manipulation tasks: (i) a simple pick-and-place task (*Put Lion*), (ii) a complex pushing T-shaped block task (*Push-T*), and (iii) a contact-rich scooping task (*Scoop Tortilla*). To assess generalization, we introduce variations to the original task settings, resulting in four unseen tasks: (1) **Put-B**: a visually distinct cloth is placed on the table in the *Put Lion* task; (2) **Put-H**: the target lion is placed on a $5.5\,\mathrm{cm}$-high box in the *Put Lion* task; (3) **Push-L**: the ambient lighting is turned off in the *Push-T* task; (4) **Scoop-C**: the tortilla is replaced with plastic noodles, representing an unseen object category. Fig. 4 provides visual illustrations of all tasks.

We collect about ten expert trajectories per task using a SpaceMouse for teleoperation and `frankapy` [56] as the control interface. Each task is evaluated for ten trials. For fair comparison, we photograph each test scene and manually align the environment setup across all evaluated methods, ensuring that every method encounters identical initial scene configurations.

Baselines. We compare our method with representative approaches spanning three dominant paradigms—3D-based policies, video-prediction–based methods, and VLA models. A brief overview is provided here, with a more detailed description in Appendix G.

- **DP3** [43]: a state-of-the-art 3D visuomotor policy that encodes point clouds using an MLP and predicts actions via a diffusion-based policy head.
- $\boldsymbol{\pi_{0.5}}$ [2]: a large-scale vision-language-action model pretrained on diverse data and decoding actions using flow matching.
- **UVA** [32]: a video-prediction–based policy that jointly predicts future videos and actions using unified representations and dual diffusion heads.
- **BridgeVLA** [49]: a 3D vision-language-action model that projects point clouds into 3D-aware multi-view images and predicts heatmaps for action decoding.

Results. The quantitative results are reported in Tab. 2. Under the extremely limited data regime, only BridgeVLA [49] and SpatialVAM achieve reasonable success rates on the basic tasks. DP3 overfits, owing to its simple MLP point-cloud encoder (addressing **Q2**); $\pi_{0.5}$ exhibits limited spatial generalization beyond the training distribution (**Q3**); UVA exhibits qualitatively correct action tendencies but insufficient execution precision (**Q1**); BridgeVLA, although built on the same multi-view heatmap paradigm as SpatialVAM, predicts only single key poses and therefore fails on the continuous-motion *Push T* task and the contact-rich *Scoop Tortilla* task. In contrast, SpatialVAM leverages a video foundation model to predict continuous multi-view heatmap and RGB sequences, enabling it to solve all three basic tasks; detailed failure analyses of SpatialVAM and the baselines are provided in Appendix H. On unseen tasks, SpatialVAM generalizes strongly, achieving the best performance on the *Push-L* and *Scoop-C* settings (**Q4**); it slightly underperforms BridgeVLA on *Put-B* and *Put-H*, a gap we attribute to BridgeVLA's vision–language backbone being pretrained on additional image–text pairs, which better captures such appearance variations. We think scaling up our video pretraining could analogously improve the generalization ability.

4.4 Robustness Analysis

We evaluate the robustness of SpatialVAM from two complementary perspectives: deployment robustness to changes in camera coverage and calibration, and modeling robustness to key hyperparameters and inference settings.

Deployment robustness. Our default setup fuses observations from three physical cameras to obtain a more complete point cloud. To examine whether multiple cameras are essential, we retain only one camera, which substantially reduces the observed point-cloud coverage. Even with this incomplete observation, using the original virtual projection views only moderately reduces the overall success rate from 57.1% to 50.0%. This decline stems largely from a view mismatch: the projection views selected for the three-camera point cloud no longer expose the most informative regions visible to the remaining camera. After adapting the virtual views to better cover the observable workspace, the success rate recovers to 60.0%, comparable to the three-camera result. The projected observations under these three settings are visualized in Fig. 7, highlighting how the adapted views better expose the observable workspace from the single-camera point cloud. These results show that SpatialVAM does not fundamentally rely on multiple physical cameras and remains effective with incomplete point clouds when its virtual projection views are aligned with the observable workspace.

To evaluate robustness to substantial calibration errors, we conduct a deliberately challenging stress test by randomly perturbing the camera extrinsics by $20\,\mathrm{mm}$ in translation and 2° in rotation. When training uses the original calibration but evaluation uses the perturbed calibration, SpatialVAM retains a 51.4% success rate, compared with 57.1% under the original calibration. When training and evaluation share the same perturbation, the success rate reaches 54.3%, approaching the original-calibration result. We attribute this robustness to our projection and heatmap representations: calibration errors become localized spatial displacements after rendering, while the spatially smooth heatmaps reduce sensitivity to exact coordinates and allow consistent calibration biases to be absorbed during training. Thus, highly accurate calibration is not a strict requirement for SpatialVAM. Full protocols and per-task results for both deployment robustness studies are provided in Appendix D.

Modeling robustness. Beyond deployment conditions, we examine whether the strong performance of SpatialVAM depends on careful tuning of its modeling and inference parameters. On Meta-World, we vary three key parameters: the RGB loss weight λ, heatmap standard deviation σ, and inference denoising steps N. The results for λ and σ are summarized in Appendix E. As shown, varying λ by 80% results in only a 3.3% change in the success rate, while varying σ by 133% leads to a mere 2.5% change. These findings show that the performance gains of SpatialVAM do not rely on narrowly tuned hyperparameters.

We next examine the inference denoising steps N, varying it from 1 to 50. The results are presented in Fig. 5 (left). Surprisingly, in contrast to typical video diffusion processes [57, 51], which require around 50 denoising steps, SpatialVAM achieves a comparable success rate with just a single denoising step, substantially reducing inference cost. We believe the main reason for this is that heatmaps have relatively simple distribution modes and lack the high-frequency details present in regular images. Furthermore, our action prediction relies solely on the peak locations of the heatmaps, meaning that the overall quality of the heatmaps is less critical, and thus fewer denoising steps are required (qualitative visualizations under different N are provided in Appendix I). To balance performance with computational efficiency, we recommend setting the denoising step to 5, which allows for 5Hz inference frequency on a single NVIDIA A100 GPU server (see Appendix C).

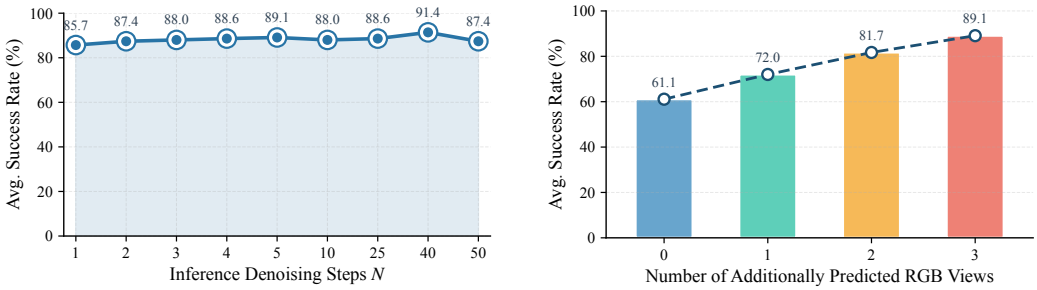

Figure 5: **Robustness and ablation on Meta-World. Left:** Avg. success rate vs. inference denoising steps N; SpatialVAM stays strong even with $N{=}1$. **Right:** Avg. success rate vs. number of additionally predicted RGB views; predicting more views yields monotonic improvements.

Table 3: **Ablation on SpatialVAM design choices.** *View Concat*: concatenate heatmap and RGB latents along the view dimension. *Init W*: use the pretrained video-foundation-model weights. *Trans. Decode*: heatmap-peak back-projection (Heatmap) vs. transformer regression from latents (TF-Regress). *LoRA*: adapt the pretrained transformer via LoRA rather than updating all its weights. Model #1 (bold) is our default; each subsequent row modifies a single design choice.

#	View Concat	Init W	LoRA	Trans. Decode	Avg (%)
1	✓	✓	✓	Heatmap	**89.1**
2	✓	✓	-	Heatmap	87.4
3	-	✓	✓	Heatmap	81.1
4	✓	✓	✓	TF-Regress	76.6
5	✓	-	✓	Heatmap	4.6

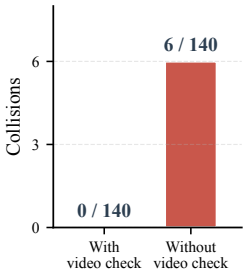

Figure 6: **Collision events with and without video-based action checking** across 140 rollouts from four evaluators; video inspection reduces unsafe behaviors (discussed in Sec. 4.6).

4.5 Ablation Studies

We conduct extensive ablation studies on the Meta-World benchmark to validate the key design choices in SpatialVAM; results are summarized in Tab. 3, ordered by success rate. Model #2 applies full fine-tuning instead of LoRA and reaches 87.4%—comparable to LoRA but with substantially higher computational and memory cost—so we adopt LoRA throughout. Model #3 concatenates heatmaps and RGB frames along the channel dimension (with an additional convolutional adapter to match the DiT's input channels); the resulting information bottleneck drops performance to 81.1%, confirming that view-dimension concatenation better preserves multi-view information. Model #4 bypasses heatmap back-projection and regresses 3D translation directly from the denoised latents with a transformer, and the success rate falls markedly to 76.6%, validating that heatmap-peak back-projection is substantially more effective than latent-space regression for translation decoding. Model #5 removes the pretrained video-foundation-model weights and almost completely fails (4.6%), as the model cannot even fit the training set, underscoring the essential role of large-scale video pretraining for data-efficient manipulation. Finally, Fig. 5 (right) varies the number of additionally predicted RGB views from 0 to 3; success rate grows monotonically from 61.1% to 89.1%, showing that (i) jointly predicting RGB and heatmap videos is markedly better than predicting heatmaps alone, and (ii) predicting more views yields a more complete spatio-temporal understanding and better downstream control. Together these results validate our design choices, answering **Q6**.

4.6 Safer Deployment and Enhanced Explainability

Deploying manipulation policies safely is challenging: the safety of a predicted action sequence is hard to judge from its raw numerical form, and reliable verification often requires costly and potentially unsafe physical execution. In contrast, SpatialVAM produces realistic, temporally consistent multi-view videos and heatmaps that users can visually inspect before execution, providing an interpretable safety check. To quantify this benefit and answer **Q7**, we conducted a user study with four evaluators, each performing 35 rollouts (140 in total), rerunning any rollout that looked unsafe (e.g., potential collisions); Fig. 6 shows that this video-based inspection significantly reduces collisions. Qualitative examples of predicted videos and heatmaps are provided in Appendix I.

5 Conclusion and Future Work

In this paper, we introduced SpatialVAM, a spatial-aware multi-view video diffusion policy that incorporates 3D structural priors into a video foundation model backbone. Our experiments demonstrate that SpatialVAM is data-efficient, robust, generalizable, and interpretable. Nevertheless, our current real-world pipeline assumes at least one fixed RGB-D camera with approximately known calibration, adding deployment complexity. Inference also remains relatively slow: generating a 24-frame action chunk takes approximately 4.6 s on an NVIDIA A100 GPU, making the method unsuitable for certain high-frequency dexterous tasks. In future work, we plan to integrate diffusion acceleration techniques, such as distillation [58], quantization [59], asynchronous execution [60], and asymmetric denoising [61], to enable real-time control in practical robotic applications.

6 Acknowledgments

This work was jointly supported by National Natural Science Foundation of China (62322607, 62236010 and 62276261), Beijing Natural Science Foundation (L252033)

References

[1] Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, et al. π0.6: a vla that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025.

[2] Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. π0.5: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[3] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π0: A vision-language-action flow model for general robot control, 2024. *URL https://arxiv.org/abs/2410.24164*, 2024.

[4] Generalist AI Team. Gen-0: Embodied foundation models that scale with physical interaction. *Generalist AI Blog*, 2025. https://generalistai.com/blog/preview-uqlxvb-bb.html.

[5] Andy Zhai, Brae Liu, Bruno Fang, Chalse Cai, Ellie Ma, Ethan Yin, Hao Wang, Hugo Zhou, James Wang, Lights Shi, et al. Igniting vlms toward the embodied space. *arXiv preprint arXiv:2509.11766*, 2025.

[6] Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi "Jim" Fan, and Joel Jang. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026.

[7] Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos policy: Fine-tuning video models for visuomotor control and planning. *arXiv preprint arXiv:2601.16163*, 2026.

[8] Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, et al. PaliGemma: A versatile 3b vlm for transfer. *arXiv preprint arXiv:2407.07726*, 2024.

[9] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-VL: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.

[10] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025.

[11] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in Neural Information Processing Systems*, 36:34892–34916, 2023.

[12] Andy Clark. Whatever next? predictive brains, situated agents, and the future of cognitive science. *Behavioral and brain sciences*, 36(3):181–204, 2013.

[13] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR, 2020.

[14] Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. In *Robotics: Science and Systems (RSS)*, 2024.

[15] Spirit AI Team. Spirit-v1.5: Clean data is the enemy of great robot foundation models. *Spirit AI Blog*, 2026. https://www.spirit-ai.com/en/blog/spirit-v1-5.

[16] Chilam Cheang, Sijin Chen, Zhongren Cui, Yingdong Hu, Liqun Huang, Tao Kong, Hang Li, Yifeng Li, Yuxiao Liu, Xiao Ma, et al. Gr-3 technical report. *arXiv preprint arXiv:2507.15493*, 2025.

[17] Chi-Lam Cheang, Guangzeng Chen, Ya Jing, Tao Kong, Hang Li, Yifeng Li, Yuxiao Liu, Hongtao Wu, Jiafeng Xu, Yichu Yang, et al. Gr-2: A generative video-language-action model with web-scale knowledge for robot manipulation. *arXiv preprint arXiv:2410.06158*, 2024.

[18] Peiyan Li, Hongtao Wu, Yan Huang, Chilam Cheang, Liang Wang, and Tao Kong. Gr-mg: Leveraging partially-annotated data via multi-modal goal-conditioned policy. *IEEE Robotics and Automation Letters*, 2025.

[19] Delin Qu, Haoming Song, Qizhi Chen, Zhaoqing Chen, Xianqiang Gao, Xinyi Ye, Qi Lv, Modi Shi, Guanghui Ren, Cheng Ruan, et al. Eo-1: Interleaved vision-text-action pretraining for general robot control. *arXiv preprint arXiv:2508.21112*, 2025.

[20] Xinghang Li, Peiyan Li, Minghuan Liu, Dong Wang, Jirong Liu, Bingyi Kang, Xiao Ma, Tao Kong, Hanbo Zhang, and Huaping Liu. Towards generalist robot policies: What matters in building vision-language-action models. *arXiv preprint arXiv:2412.14058*, 2024.

[21] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[22] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

[23] Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.

[24] Yucheng Hu, Yanjiang Guo, Pengchao Wang, Xiaoyu Chen, Yen-Jen Wang, Jianke Zhang, Koushil Sreenath, Chaochao Lu, and Jianyu Chen. Video prediction policy: A generalist robot policy with predictive visual representations. *arXiv preprint arXiv:2412.14803*, 2024.

[25] Jonas Pai, Liam Achenbach, Victoriano Montesinos, Benedek Forrai, Oier Mees, and Elvis Nava. mimic-video: Video-action models for generalizable robot control beyond VLAs. *arXiv preprint arXiv:2512.15692*, 2025.

[26] Yilun Du, Sherry Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Josh Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. *Advances in Neural Information Processing Systems*, 36:9156–9172, 2023.

[27] Liudi Yang, Yang Bai, George Eskandar, Fengyi Shen, Mohammad Altillawi, Dong Chen, Soumajit Majumder, Ziyuan Liu, Gitta Kutyniok, and Abhinav Valada. RoboEnvision: A long-horizon video generation model for multi-task robot manipulation. *arXiv preprint arXiv:2506.22007*, 2025.

[28] Homanga Bharadhwaj, Debidatta Dwibedi, Abhinav Gupta, Shubham Tulsiani, Carl Doersch, Ted Xiao, Dhruv Shah, Fei Xia, Dorsa Sadigh, and Sean Kirmani. Gen2Act: Human video generation in novel scenarios enables generalizable robot manipulation. *arXiv preprint arXiv:2409.16283*, 2024.

[29] Po-Chen Ko, Jiayuan Mao, Yilun Du, Shao-Hua Sun, and Joshua B Tenenbaum. Learning to act from actionless videos through dense correspondences. *arXiv preprint arXiv:2310.08576*, 2023.

[30] 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*, pages 306–324. Springer, 2024.

[31] Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. *arXiv preprint arXiv:2312.13139*, 2023.

[32] Shuang Li, Yihuai Gao, Dorsa Sadigh, and Shuran Song. Unified video action model. *arXiv preprint arXiv:2503.00200*, 2025.

[33] Liudi Yang, Yang Bai, George Eskandar, Fengyi Shen, Mohammad Altillawi, Dong Chen, Ziyuan Liu, and Abhinav Valada. Covar: Co-generation of video and action for robotic manipulation via multi-modal diffusion. *arXiv preprint arXiv:2512.16023*, 2025.

[34] Yixiang Chen, Peiyan Li, Yan Huang, Jiabing Yang, Kehan Chen, and Liang Wang. EC-Flow: Enabling versatile robotic manipulation from action-unlabeled videos via embodiment-centric flow. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11958–11968, October 2025.

[35] Chuning Zhu, Raymond Yu, Siyuan Feng, Benjamin Burchfiel, Paarth Shah, and Abhishek Gupta. Unified world models: Coupling video and action diffusion for pretraining on large robotic datasets. *arXiv preprint arXiv:2504.02792*, 2025.

[36] Yanjiang Guo, Yucheng Hu, Jianke Zhang, Yen-Jen Wang, Xiaoyu Chen, Chaochao Lu, and Jianyu Chen. Prediction with action: Visual policy learning via joint denoising process. *Advances in Neural Information Processing Systems*, 37:112386–112410, 2024.

[37] Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, et al. Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*, 2026.

[38] Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-wam: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026. URL `https://arxiv.org/abs/2603.16666`.

[39] Yixiang Chen, Peiyan Li, Yuan Xu, Qisen Ma, Jiabing Yang, Kai Wang, Jianhua Yang, Dong An, He Guan, Gaoteng Liu, Jianlou Si, Jun Huang, Jing Liu, Nianfeng Liu, Yan Huang, and Liang Wang. Flowwam: Optical flow as a unified action representation for world action models, 2026. URL `https://arxiv.org/abs/2607.13017`.

[40] Teli Ma, Jia Zheng, Zifan Wang, Chunli Jiang, Andy Cui, Junwei Liang, and Shuo Yang. Dit4dit: Jointly modeling video dynamics and actions for generalizable robot control. *arXiv preprint arXiv:2603.10448*, 2026.

[41] Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, et al. Motus: A unified latent action world model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 35101–35113, 2026.

[42] Theophile Gervet, Zhou Xian, Nikolaos Gkanatsios, and Katerina Fragkiadaki. Act3d: 3d feature field transformers for multi-task robotic manipulation. *arXiv preprint arXiv:2306.17817*, 2023.

[43] Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. *arXiv preprint arXiv:2403.03954*, 2024.

[44] Rujia Yang, Geng Chen, Chuan Wen, and Yang Gao. Fp3: A 3d foundation policy for robotic manipulation. *arXiv preprint arXiv:2503.08950*, 2025.

[45] Shizhe Chen, Ricardo Garcia Pinel, Cordelia Schmid, and Ivan Laptev. Polarnet: 3d point clouds for language-guided robotic manipulation. In *Conference on Robot Learning*, pages 1761–1781. PMLR, 2023.

[46] Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, et al. Spatialvla: Exploring spatial representations for visual-language-action model. *arXiv preprint arXiv:2501.15830*, 2025.

[47] Fuhao Li, Wenxuan Song, Han Zhao, Jingbo Wang, Pengxiang Ding, Donglin Wang, Long Zeng, and Haoang Li. Spatial forcing: Implicit spatial representation alignment for vision-language-action model. *arXiv preprint arXiv:2510.12276*, 2025.

[48] Ankit Goyal, Valts Blukis, Jie Xu, Yijie Guo, Yu-Wei Chao, and Dieter Fox. Rvt-2: Learning precise manipulation from few demonstrations. In *RSS 2024 Workshop: Data Generation for Robotics*, 2024.

[49] Peiyan Li, Yixiang Chen, Hongtao Wu, Xiao Ma, Xiangnan Wu, Yan Huang, Liang Wang, Tao Kong, and Tieniu Tan. BridgeVLA: Input-output alignment for efficient 3d manipulation learning with vision-language models. *arXiv preprint arXiv:2506.07961*, 2025.

[50] Yixiang Chen, Yan Huang, Keji He, Peiyan Li, and Liang Wang. VERM: Leveraging foundation models to create a virtual eye for efficient 3d robotic manipulation. *arXiv preprint arXiv:2512.16724*, 2025.

[51] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

[52] William Peebles and Saining Xie. Scalable diffusion models with transformers. *arXiv preprint arXiv:2212.09748*, 2022.

[53] Jianhong Bai, Menghan Xia, Xintao Wang, Ziyang Yuan, Xiao Fu, Zuozhu Liu, Haoji Hu, Pengfei Wan, and Di Zhang. Syncammaster: Synchronizing multi-camera video generation from diverse viewpoints. *arXiv preprint arXiv:2412.07760*, 2024.

[54] Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022.

[55] Tsung-Wei Ke, Nikolaos Gkanatsios, and Katerina Fragkiadaki. 3d diffuser actor: Policy diffusion with 3d scene representations. *Arxiv*, 2024.

[56] Kevin Zhang, Mohit Sharma, Jacky Liang, and Oliver Kroemer. A modular robotic arm control stack for research: Franka-interface and frankapy. *arXiv preprint arXiv:2011.02398*, 2020.

[57] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. CogVideoX: Text-to-video diffusion models with an expert transformer. *arXiv preprint arXiv:2408.06072*, 2024.

[58] Arman Akbari, Ci Zhang, Arash Akbari, Lin Zhao, Yixiao Chen, Weiwei Chen, Xuan Zhang, Geng Yuan, and Yanzhi Wang. Flash-wam: Modality-aware distillation for world action models. *arXiv preprint arXiv:2606.05254*, 2026.

[59] Xingyang Li, Samuel Tesfai, Zhekai Zhang, Haocheng Xi, Shuo Yang, Lvmin Zhang, Yufei Sun, Kelly Peng, Maneesh Agrawala, Ion Stoica, et al. Deltaquant: 4-bit video diffusion models with spatiotemporal delta smoothing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 43578–43588, 2026.

[60] Kevin Black, Manuel Galliker, and Sergey Levine. Real-time execution of action chunking flow policies. *Advances in Neural Information Processing Systems*, 38:33383–33407, 2026.

[61] Jiajun Li, Tiecheng Guo, Yifan Ye, Rongyu Zhang, Xiaowei Chi, Qianpu Sun, Ying Li, Yunfan Lou, Yan Huang, Zhihe Lu, et al. Efficient-wam: A 1b-parameter world-action model with low-cost future imagination. *arXiv preprint arXiv:2606.10040*, 2026.

[62] Markus Schütz, Bernhard Kerbl, and Michael Wimmer. Rendering point clouds with compute shaders and vertex order optimization. In *Computer Graphics Forum*, volume 40, pages 115–126. Wiley Online Library, 2021.

Spatial-Aware Multi-View Video Diffusion

—Appendix—

A Multi-View Video Diffusion

In this section, we provide a detailed description of the multi-view video diffusion process used in SpatialVAM, which is based on the Wan2.2 framework [51]. This process consists of three main components: the 3D VAE encoder, the Diffusion Transformer (DiT) block, and the 3D VAE decoder. We will explain about these components and how they integrate within our pipeline.

A.1 3D VAE Encoder

The first step in the video diffusion process is encoding the input RGB image sequences and heatmap sequences (which should first be colorized to have 3 RGB channels) into a latent space using a 3D Variational Autoencoder (VAE). Both the input RGB sequence and the heatmap sequence are treated as a video sequence $V \in \mathbb{R}^{(1+T)\times H\times W\times 3}$, where $(1+T)$ is the total number of frames, H and W are the spatial dimensions of each frame, and 3 denotes the RGB color channels.

The 3D VAE performs both spatial and temporal compression to reduce the dimensionality of the input video.

Spatial Compression: The spatial compression is achieved by applying a series of 3D convolutions, which reduce the spatial resolution of the input video from $H \times W$ to $H/8 \times W/8$, while expanding the number of channels to $C = 16$. This allows the model to capture important spatial features in a lower-dimensional representation.

Temporal Compression: In addition to spatial compression, the 3D VAE also reduces the temporal resolution by applying 3D convolutions across the time dimension. Specifically, the input video sequence is compressed in the time dimension from $(1+T)$ frames to $[1+T/4]$, where the temporal resolution is reduced by a factor of 4. This temporal compression allows the model to learn key temporal features while reducing the computational cost of processing long video sequences.

The 3D VAE encoding process can be formally expressed as:

$$q(z|V) = \mathcal{N}(z; \mu, \sigma^2) \tag{5}$$

where μ and σ are the mean and variance outputs from the VAE encoder, and z represents the latent variables.

A.2 Diffusion Transformer (DiT)

After obtaining the latent representations from the 3D VAE encoder, these are passed to the Diffusion Transformer (DiT) to model the temporal evolution of the video. The DiT is responsible for predicting future frames of the video through a denoising diffusion process.

The input to the DiT consists of a sequence of latents $\{z_0, z_1, \ldots, z_T\}$, where each z_t represents the concatenated latent representation of the video and heatmap at time step t. During inference, Gaussian noise is added to these latent representations to simulate the diffusion process. The model is trained to denoise the noisy latents at each timestep.

The model predicts the velocity between latent frames using flow matching [21]. This velocity prediction is crucial for generating realistic video frames. The objective is to minimize the difference between the predicted velocity and the ground truth, defined as:

$$v_t = \frac{dz_t}{dt} = z_1 - z_0 \tag{6}$$

During inference, the model predicts the velocity at each timestep to generate temporally consistent video sequences.

A.3 3D VAE Decoder

After the latents are processed by the DiT, they are passed to the 3D VAE decoder to reconstruct the video frames. The decoder takes the output latent representations $\{z_0, z_1, \ldots, z_T\}$ and maps them back to the pixel space, producing the final RGB or heatmap sequences $\hat{V} \in \mathbb{R}^{(1+T)\times H\times W\times 3}$.

The decoding process mirrors the encoding process, with the addition of upsampling operations to reconstruct high-resolution frames from the low-dimensional latent space.

A.4 Efficient Inference with Cache Mechanism

To facilitate efficient inference for long video sequences, Wan incorporates a feature cache mechanism that stores previously computed features for reuse during the inference process. This approach reduces both memory and computational requirements, enabling the model to handle arbitrarily long video sequences.

For further details on the diffusion process, we refer the reader to [51].

B Projection and Back Projection

We adopt the projection and back projection techniques outlined in BridgeVLA [49] and RVT-2 [48], with the following brief overview of the process:

B.1 Why Virtual-View Rendering?

Configurable and task-relevant viewpoints. Virtual-view rendering decouples the model input from the placement of the physical cameras. Raw RGB-D images are restricted to the viewpoints determined by sensor installation, whereas the fused point cloud can be rendered from virtual viewpoints that more clearly expose the workspace and task-relevant interaction regions. The viewing directions, rendering bounds, and spatial resolution can therefore be configured for a given workspace and remain fixed during training and evaluation.

Multi-view representations with fewer physical cameras. Virtual rendering also preserves a multi-view input representation when fewer physical cameras are available. A single RGB-D camera produces a less complete point cloud, and rendering cannot recover surfaces that were never observed. Nevertheless, the observed geometry can still be presented from multiple virtual directions, allowing the model to reason over complementary projections of the available 3D information. As shown in Table 5 and Fig. 7, adapting the virtual views to the workspace observable from one camera yields performance comparable to the original three-camera setting.

Geometrically consistent augmentation. Point-cloud rendering naturally supports consistent 3D data augmentation. During training, we apply the same SE(3) transformation to the point cloud and end-effector trajectory before rendering the transformed observations and heatmaps from the same virtual viewpoints. In contrast, independently applying 2D transformations to raw camera images does not generally preserve the underlying 3D geometry, cross-view consistency, or alignment with the transformed action trajectory.

B.2 Projection Process

The projection procedure involves three primary steps for mapping a point cloud of N points to an RGB and depth image of size (h, w):

Projection To project a 3D point cloud onto a 2D image, we first compute the depth d_n and the pixel coordinates (x_n, y_n) for each point indexed by $n \in \{0, 1, \ldots, N\}$, where f_n is the RGB value. These values are derived using the camera's intrinsic and extrinsic parameters. The 2D pixel index i_n is then calculated as $i_n = x_n \cdot w + y_n$, where w is the image width. This projection process can be efficiently accelerated by leveraging GPU-optimized matrix multiplications.

Table 4: Hyperparameters used for training and inference in SpatialVAM.

Hyperparameter	Multi-View Video Diffusion Transformer	Rotation & Gripper Predictor
Optimizer	AdamW	AdamW
Learning rate	1.0×10^{-4}	1.0×10^{-4}
Epochs	100	100
Batch size	64	64
Weight decay	0	1.0×10^{-5}
Image resolution	256×256	256×256
Heatmap std. (σ)	1.5	–
Predicted video length	24	–
RGB loss weight (λ)	0.5	–
Heatmap loss weight	0.5	–
LoRA modules	`q, k, v, o, ffn.0, ffn.2`	–
LoRA rank	32	–
Training timesteps	1000	–
Inference steps	5	–
Classifier-free guidance	1	–

Z-ordering For each pixel j in the image, we identify the point with the smallest depth d_n among all the points projecting onto that pixel $\{n \mid i_n = j\}$. The RGB value f_n of this point is then assigned to pixel j in the RGB image, and the corresponding depth d_n is stored in the depth image.

To enhance the performance of Z-ordering, we pack the depth and index of each point into a 64-bit integer. The higher 32 bits store the depth, and the lower 32 bits store the point index. This packing allows us to use two CUDA kernels for efficient computation. In the first kernel, we process each point in parallel and attempt to store its packed depth-index value at the corresponding pixel j in a depth-index image using the *atomicMin* operation. This ensures that only the point with the smallest depth at each pixel is retained. In the second kernel, the depth-index is unpacked, and the depth and feature images are reconstructed by referencing the corresponding point features. This technique, initially introduced in [62] for rendering colored point clouds by packing 32-bit color values, is extended here to support images with an arbitrary number of channels.

Screen-Space Splatting While the projection and Z-ordering steps are sufficient for basic image rendering, points are treated as infinitesimal light sources, which can result in visual noise when the resolution of the point cloud in screen space is lower than the image resolution. To address this issue, we apply 3D splatting, where each point is modeled as a disc of radius r facing the camera. This approach is applied in screen space after projection and Z-ordering, reducing the computational burden of these steps. For each pixel j, the algorithm searches within a local neighborhood to find the nearest pixel k with a smaller depth. If pixel k satisfies $d_k < d_j$ and is closer than $r \cdot \text{focal_length}/d_k$, the feature and depth of pixel j are replaced by those of pixel k.

B.3 Back Projection Process

The back-projection procedure aims to identify the most accurate 3D position within the workspace, based on the three heatmaps generated by our virtual cameras. This process discretizes the workspace into a set of 3D points, computing the corresponding locations for each point on the three heatmaps. The probabilities for each point across the three heatmaps are averaged, and the point with the highest average probability is selected as the target 3D point.

For a more detailed explanation of these techniques, we refer the reader to the original works [49, 48].

C Training & Inference Details

In this section, we detail the training and inference procedures for SpatialVAM. The corresponding hyperparameters used during both training and inference are listed in Table 4.

For the Meta-World benchmark, we train SpatialVAM using a total of 32 NVIDIA H20 GPUs for 21.8k steps. The training of the multi-view video diffusion transformer and the Rotation and Gripper

Table 5: **Robustness to physical-camera coverage and virtual-view selection.** Success rates over 10 trials per task on three basic and four unseen tasks.

Setting	Basic Tasks			Unseen Tasks				Avg. Succ. (%)
	Put Lion	Push-T	Scoop Tort.	Put-B	Put-H	Push-L	Scoop-C	
Three Cameras	**10/10**	4/10	7/10	**5/10**	6/10	**3/10**	**5/10**	57.1
One Camera Original Views	8/10	3/10	7/10	**5/10**	5/10	**3/10**	4/10	50.0
One Camera Adapted Views	**10/10**	**5/10**	**8/10**	**5/10**	**7/10**	**3/10**	4/10	**60.0**

Table 6: **Robustness to camera calibration errors.** Success rates over 10 trials per task under the original calibration and two perturbed-calibration settings.

Setting	Basic Tasks			Unseen Tasks				Avg. Succ. (%)
	Put Lion	Push-T	Scoop Tort.	Put-B	Put-H	Push-L	Scoop-C	
Original Calibration	**10/10**	**4/10**	7/10	5/10	**6/10**	3/10	**5/10**	**57.1**
Perturbed Calibration Setting 1	8/10	2/10	6/10	**6/10**	5/10	**4/10**	**5/10**	51.4
Perturbed Calibration Setting 2	**10/10**	**4/10**	**8/10**	4/10	5/10	3/10	4/10	54.3

Predictor takes approximately 42 hours and 36 hours, respectively. For the real-world benchmark, we train the same models using 32 NVIDIA H200 GPUs for 8.6k steps, which takes around 11 hours and 10 hours, respectively.

For real-world deployment, the inference process requires more than 30 GB of GPU memory. Thus, we perform inference on a server with an NVIDIA A100 GPU, utilizing FastAPI for communication between the server and the robot client. To assess inference time, we conduct 30 trials, and the average time taken from point cloud input to the output of an action chunk (of length 24) is 4.6 seconds.

D Deployment Robustness Analysis

We further examine whether the default use of multiple fixed and calibrated physical cameras is essential to SpatialVAM. Following the real-world evaluation protocol in Sec. 4.3, we separately study robustness to reduced camera coverage and calibration errors across the same seven manipulation tasks.

D.1 Robustness to Camera Number

In the default setting, observations from three physical cameras are fused into a colored point cloud. For the single-camera settings, we retain only one of these cameras, substantially reducing point-cloud coverage, and compare the original virtual projection views with adapted views selected to better cover the workspace observable from the remaining camera. All three settings are evaluated on the same seven real-world tasks with 10 trials per task.

D.2 Robustness to Calibration Errors

We randomly perturb the camera extrinsics by $20\,\mathrm{mm}$ in translation and 2° in rotation. Under Setting 1, training uses the original calibration while evaluation uses the perturbed calibration, introducing a train–test mismatch. Under Setting 2, training and evaluation use the same perturbed calibration, representing a consistent calibration bias. We compare both settings with the original calibration on the same seven real-world tasks, using 10 trials per task.

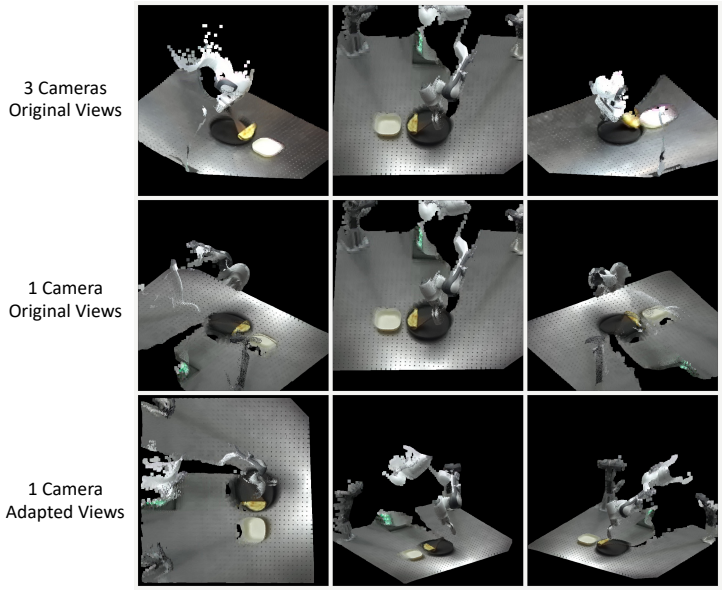

Figure 7: **Projected observations under different physical-camera and virtual-view configurations.** The three rows show, from top to bottom: three physical cameras with the original virtual projection views, one physical camera with the original views, and one physical camera with adapted views. Although using one camera reduces point-cloud coverage, adapting the virtual views better exposes the workspace observable from the remaining camera.

Table 7: Average Success Rate On Meta-World with different RGB Loss Weights and Heatmap Standard Deviations.

Hyperparameter	RGB Loss Weight (λ)			Heatmap Standard Deviation (σ)		
	0.1	0.5	0.9	1.5	2.5	3.5
Avg. Suc. (%)	92	89.1	89.1	89.1	89.7	86.9

E Robustness Analysis for Key Parameters

In this section, we evaluate the robustness of our model with respect to key hyperparameters, specifically the RGB loss weight λ and Heatmap standard deviation σ. We vary these hyperparameters individually, retrain the model, and re-evaluate it on the Meta-World benchmark. To ensure a fair comparison, we keep all other parameters fixed for each experiment. The results are summarized in Table 7. As shown, when the λ parameter is varied by 80%, the success rate changes by only 2.9%; when the σ parameter is varied by 133%, the success rate changes by just 2.2%. These findings highlight that our model is highly robust to variations in these hyperparameters.

F Meta-World Simulation Baselines

We evaluate our method against a diverse set of imitation-learning and video-prediction baselines, commonly benchmarked on the Meta-World dataset. The baseline results presented in our tables follow the numbers from prior works [29, 34]. Below, we summarize the core characteristics and implementations of each baseline.

BC-Scratch and BC-R3M BC-Scratch [54] is a standard multi-task behavioral cloning baseline, trained end-to-end using expert demonstrations. It employs a dual-stream encoder, processing RGB observations with a ResNet-18 and task descriptions with a CLIP text encoder. The encoded features are concatenated and passed through a multi-layer perceptron to directly predict actions. BC-R3M [54] follows the same architecture and training protocol but initializes the visual encoder with R3M-pretrained weights, which improves visual representations and generalization.

Diffusion Policy Diffusion Policy [22] frames action generation as a diffusion process, where continuous action sequences are sampled through iterative denoising, conditioned on a fixed window of past observations. Trained via imitation learning, this approach offers a flexible and expressive model of multimodal action distributions. However, it does not explicitly model future visual observations nor utilize 3D-related information.

UniPi UniPi [26] is a two-stage video-prediction policy learning framework. It first learns a vision-language–conditioned video prediction model to forecast future observations, and then trains an inverse dynamics model using behavioral cloning to map predicted frame transitions to actions. This decomposition allows UniPi to leverage learned visual dynamics, though action generation remains indirect and dependent on the inverse dynamics model's quality.

AVDC AVDC [29] adopts an object-centric, flow-based approach for visuomotor control. It predicts future video frames and extracts object motion trajectories using off-the-shelf point tracking methods. These trajectories are then used to generate robot actions. AVDC leverages a flow-based formulation that decouples perception and control, which helps it achieve robust performance across a range of environments.

Track2Act Track2Act [30] is an object-centric manipulation framework that decomposes action generation into object motion prediction and control. Given visual observations and a goal specification, it first predicts future object trajectories by tracking keypoints in videos. These trajectories then condition a behavior cloning policy to output robot actions. To mitigate execution errors and modeling inaccuracies, Track2Act incorporates a residual action correction module that refines predicted actions during execution.

DreamZero. DreamZero [6] is a 14B video-action model (VAM) built on a pretrained video diffusion backbone. It jointly predicts future video frames and robot actions in an autoregressive manner using flow matching with teacher forcing. Through a series of system- and model-level optimizations, including DreamZero-Flash, which decouples the denoising schedules for video and action prediction, DreamZero achieves a 38$\times$ inference speedup, enabling real-time closed-loop control at 7 Hz. For a fair comparison with our method, we replace its original Wan2.1-I2V-14B backbone with Wan2.2-TI2V-5B to better align the two methods in model capacity. All other settings follow the official recommendations.

G Real-World Baselines

In real-world experiments, we compare SpatialVAM with representative baselines spanning 3D-based visuomotor policies, video-prediction methods, and vision-language-action (VLA) models. Below, we describe the core modeling choices of each baseline and how they are instantiated in our setting.

DP3 DP3 [43] is a 3D-based visuomotor policy that directly maps geometric observations to continuous action sequences. It represents the environment using sparse point clouds reconstructed from depth observations, which are encoded with a lightweight MLP-based point encoder. Actions are generated using a diffusion-based policy head conditioned on the encoded 3D features and robot proprioception.

In contrast to SpatialVAM, DP3 does not leverage the pretrained knowledge of foundation models, nor does it explicitly predict future visual observations. For fairness in our experiments, we use the official DP3 implementation with point clouds constructed from multi-view depth observations.

$\pi_{0.5}$ $\pi_{0.5}$ [2] is a large-scale vision-language-action (VLA) model that unifies perception, language understanding, and action generation within a single autoregressive transformer. It formulates imitation learning as next-token prediction, jointly processing visual observations, language instructions, and robot actions using a pretrained vision-language backbone. One key feature of $\pi_{0.5}$ is its hybrid action representation. During training, actions are discretized into tokens for scalable pretraining, while at inference time, an action expert generates continuous action chunks via flow matching. This design enables efficient non-autoregressive action generation while retaining the language grounding and reasoning capabilities of the underlying model. However, $\pi_{0.5}$ does not explicitly model future

visual observations or incorporate 3D geometric representations. In our experiments, we use the official $\pi_{0.5}$ implementation, inputting 3 RGB observations for fair comparison.

UVA UVA [32] is a video-prediction policy learning framework that jointly models future visual observations and robot actions. It learns a unified latent representation by predicting future video frames and corresponding action chunks over a fixed horizon, encouraging the model to capture scene dynamics. While video and action prediction are trained jointly, policy inference decodes actions directly from the learned latent representation for efficiency. Despite modeling future observations, UVA operates on single-view video inputs and does not leverage multi-view prediction or cross-view geometric consistency. In our experiments, we follow the default settings of UVA, using 4-step history observations and single-view video prediction.

BridgeVLA BridgeVLA [49] is a 3D-aware vision-language-action model that predicts robot actions via heatmap-based keyframe estimation. It represents the scene using point clouds reconstructed from RGB-D observations, which are projected into multiple orthographic views. A vision-language backbone predicts 2D heatmaps for each view, which are back-projected to infer the next end-effector translation. Remaining action components are predicted using an MLP. Action execution is carried out by an external motion planner that moves the robot to the predicted keyframe pose.

BridgeVLA is trained using large-scale 2D heatmap pretraining, followed by 3D action fine-tuning, aligning vision-language pretraining with downstream manipulation tasks. However, it predicts only a single next keyframe at each step and does not explicitly model continuous action execution or future visual evolution. We evaluate BridgeVLA using its official implementation and merged point clouds from multiple depth cameras.

Overall, these baselines cover a broad spectrum of design choices, ranging from direct 3D-to-action mappings, to video prediction, to vision–language–guided key-pose estimation, enabling a comprehensive comparison with our multi-view video diffusion policy.

H Detailed Real-World Analysis

We provide the per-baseline qualitative failure analyses referenced in Sec. 4.3, as well as the primary failure mode of SpatialVAM.

DP3. DP3 severely overfits to the training set. On the *Put Lion* task, despite its training loss decreasing from 10^{-1} to 10^{-8}, the policy fails to approach the target object, instead directly moving toward target regions (the shelf). This failure is attributed to DP3's simple MLP-based point cloud encoder, which struggles with sensor noise and real-world variability.

$\boldsymbol{\pi_{0.5}}$**.** For $\pi_{0.5}$, spatial generalization is limited. On *Put Lion*, it can only approach the target within the training distribution; otherwise, it fails. On *Push T*, it completely fails, showing no success tendency. On *Scoop Tortilla*, it scoops the tortilla reliably but fails to place it on the plate, likely due to plate placement variations in the training data.

UVA. UVA exhibits reasonable action tendencies across all tasks, but its execution accuracy is insufficient. On *Put Lion*, it frequently closes the gripper prematurely. On *Push T*, it tends to over-push the object, entering out-of-distribution states. On *Scoop Tortilla*, the dominant failure mode is inaccurate pouring, where the tortilla often falls outside the plate.

BridgeVLA. BridgeVLA is the strongest competing baseline. Similar to our approach, it projects 3D point clouds into multi-view images and predicts heatmaps before decoding actions. However, it is based on a vision–language model (PaliGemma [8]) pretrained on static image–text pairs, which cannot predict continuous video or heatmap sequences. As a result, it predicts only a single key pose, requiring an external motion planner for actions between poses. This works for *Put Lion*, but fails for *Push T*, where defining key poses is difficult. For contact-rich tasks like *Scoop Tortilla*, intermediate actions are crucial for success, and predicting only a final key pose is insufficient. We observe that BridgeVLA often fails in *Scoop Tortilla*, even with a reasonable key pose.

SpatialVAM failure mode. The primary failure mode of SpatialVAM stems from inaccurate action decoding. For example, on *Scoop Tortilla*, the policy occasionally scrapes across the top surface instead of lifting the tortilla from underneath. We attribute this limitation to the finite resolution of the heatmaps (256×256), where each pixel corresponds to approximately $4\,\mathrm{mm}$. Increasing the heatmap resolution may further improve performance.

I Visualization of Video Prediction

We visualize the video prediction outputs from SpatialVAM for both the Meta-World and real-world (Fig. 8 to Fig. 12) environments. As shown, the generated RGB videos are realistic and align well with the generated heatmaps. This consistency allows the predicted actions to be validated visually, providing an effective tool to avoid unsafe deployments. We further compare the prediction results under different denoising step settings in Fig. 13 and Fig. 14. When the denoising step is set to 1, the visual quality of the generated RGB videos deteriorates noticeably, while the peak locations of the predicted heatmap videos remain stable. Increasing the denoising step to 5 or 50 does not lead to a significant visual difference in the generated outputs.

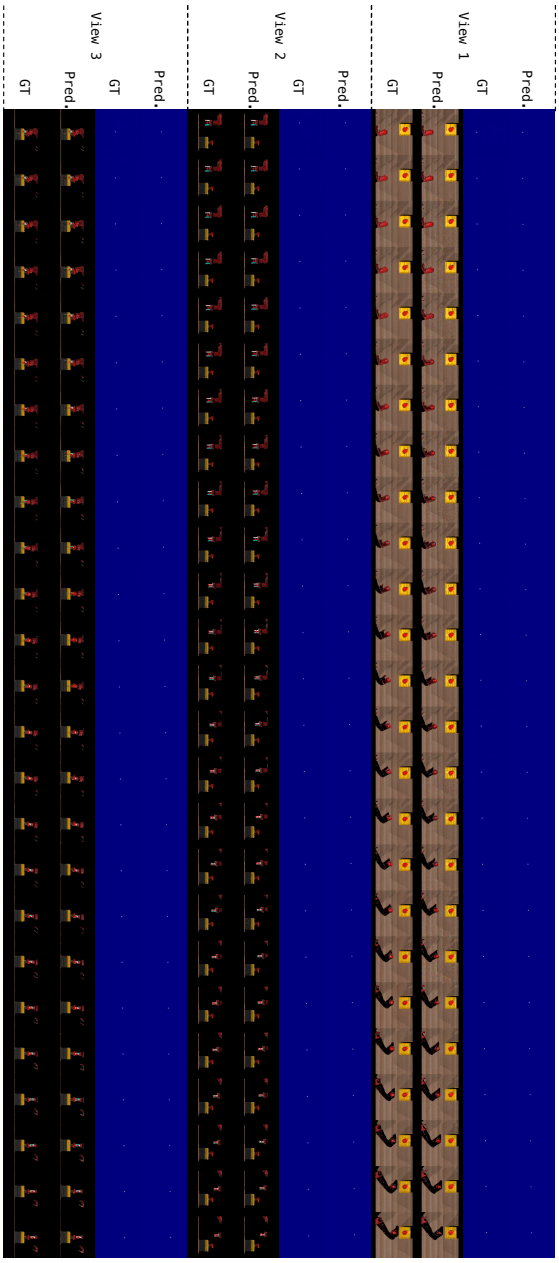

Figure 8: **Visualization of the predicted RGB sequences and heatmap sequences for the *Button-Press-Top* task in Meta-World.** For each view, the first and third rows show predictions from SpatialVAM, while the second and fourth rows show the corresponding ground truth. The peak locations of both predicted and ground-truth heatmaps are overlaid on the predicted and ground-truth RGB images, respectively. The results show that (1) the predicted RGB sequences are visually realistic, and (2) the heatmap peaks closely follow the end effector in the RGB images, indicating strong consistency between the predicted heatmaps and RGB sequences.

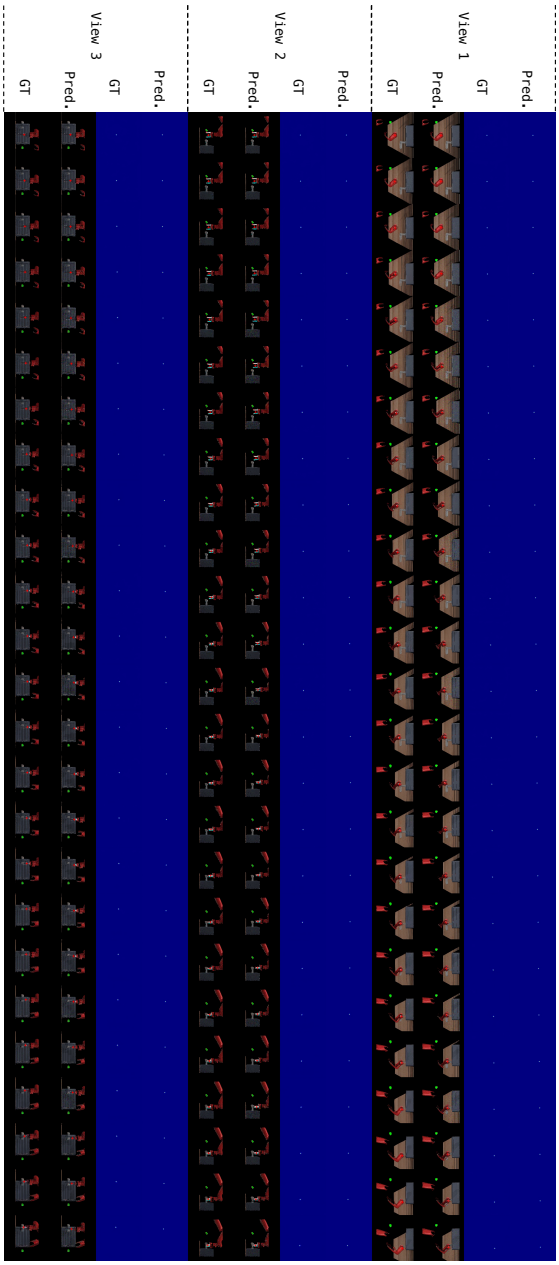

Figure 9: **Visualization of the predicted RGB sequences and heatmap sequences for the *Door-Open* task in Meta-World.** For each view, the first and third rows show predictions from SpatialVAM, while the second and fourth rows show the corresponding ground truth. The peak locations of both predicted and ground-truth heatmaps are overlaid on the predicted and ground-truth RGB images, respectively. The results show that (1) the predicted RGB sequences are visually realistic, and (2) the heatmap peaks closely follow the end effector in the RGB images, indicating strong consistency between the predicted heatmaps and RGB sequences.

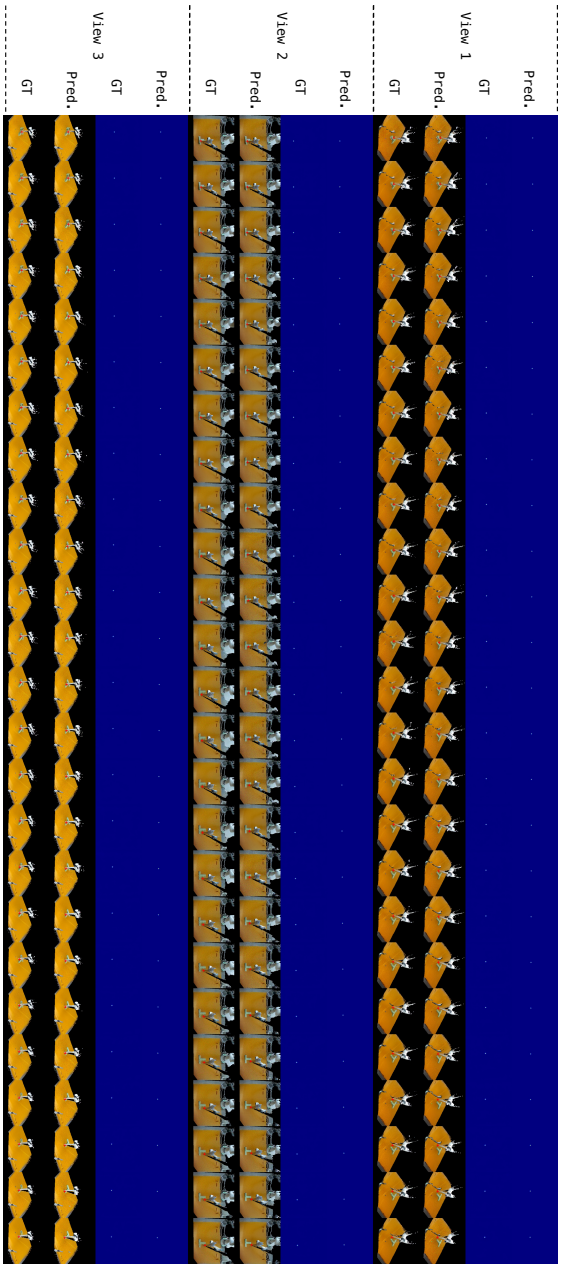

Figure 10: **Visualization of the predicted RGB sequences and heatmap sequences for the *Push-T* task.** For each view, the first and third rows show predictions from SpatialVAM, while the second and fourth rows show the corresponding ground truth. The peak locations of both predicted and ground-truth heatmaps are overlaid on the predicted and ground-truth RGB images, respectively. The results show that (1) the predicted RGB sequences are visually realistic, and (2) the heatmap peaks closely follow the end effector in the RGB images, indicating strong consistency between the predicted heatmaps and RGB sequences.

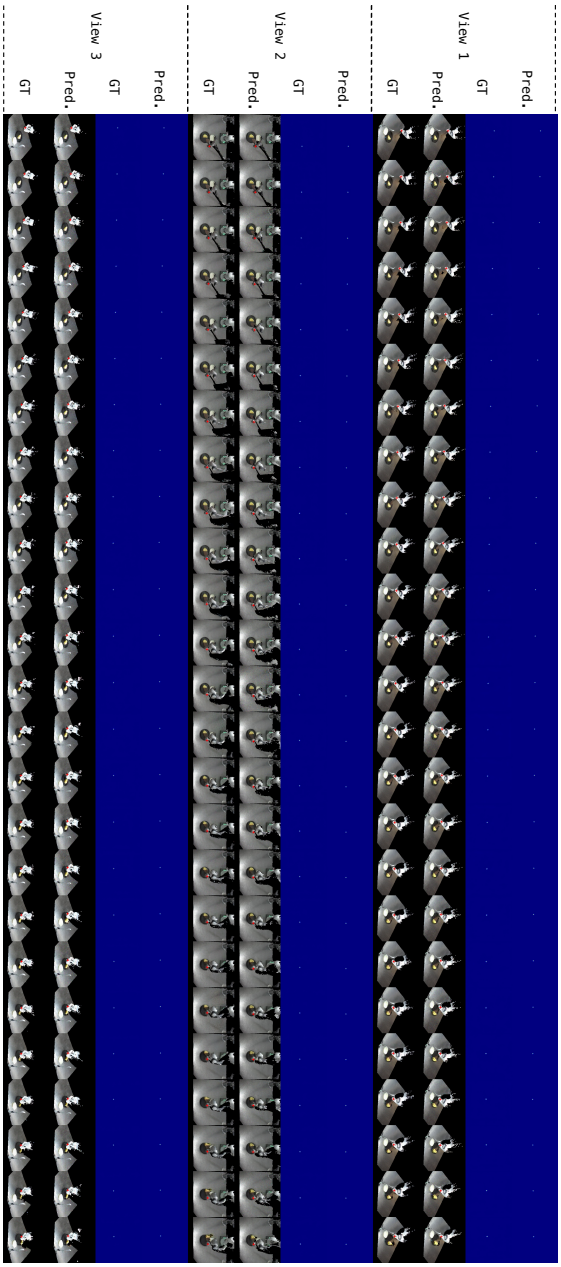

Figure 11: **Visualization of the predicted RGB sequences and heatmap sequences for the *Scoop Tortilla* task.** For each view, the first and third rows show predictions from SpatialVAM, while the second and fourth rows show the corresponding ground truth. The peak locations of both predicted and ground-truth heatmaps are overlaid on the predicted and ground-truth RGB images, respectively. The results show that (1) the predicted RGB sequences are visually realistic, and (2) the heatmap peaks closely follow the end effector in the RGB images, indicating strong consistency between the predicted heatmaps and RGB sequences.

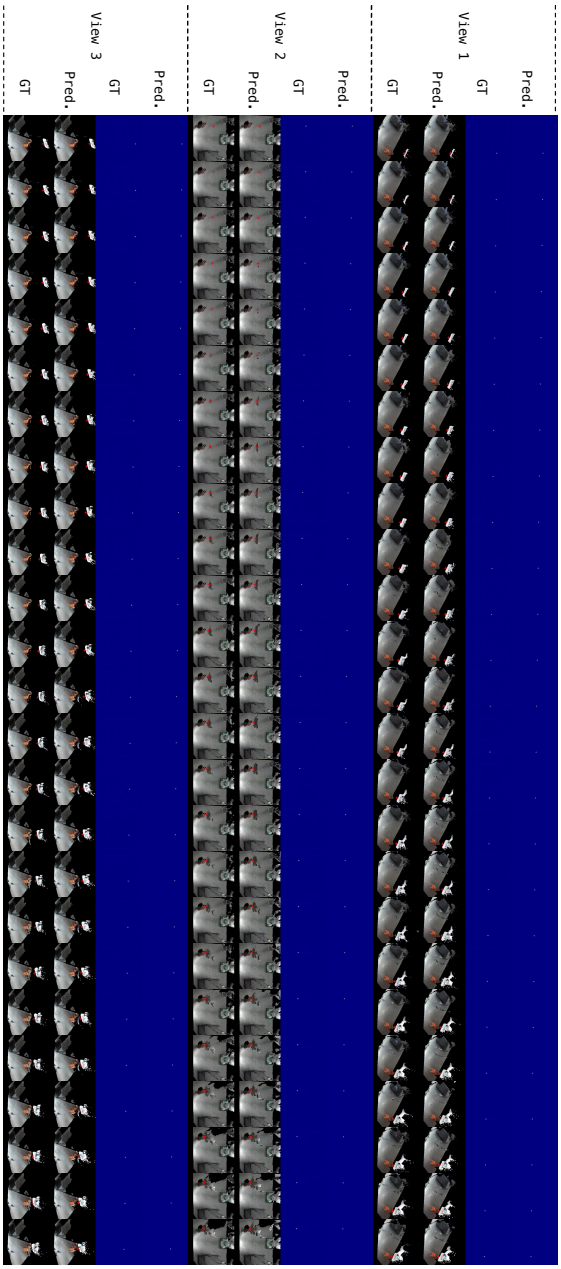

Figure 12: **Visualization of the predicted RGB sequences and heatmap sequences for the *Put Lion* task.** For each view, the first and third rows show predictions from SpatialVAM, while the second and fourth rows show the corresponding ground truth. The peak locations of both predicted and ground-truth heatmaps are overlaid on the predicted and ground-truth RGB images, respectively. The results show that (1) the predicted RGB sequences are visually realistic, and (2) the heatmap peaks closely follow the end effector in the RGB images, indicating strong consistency between the predicted heatmaps and RGB sequences.

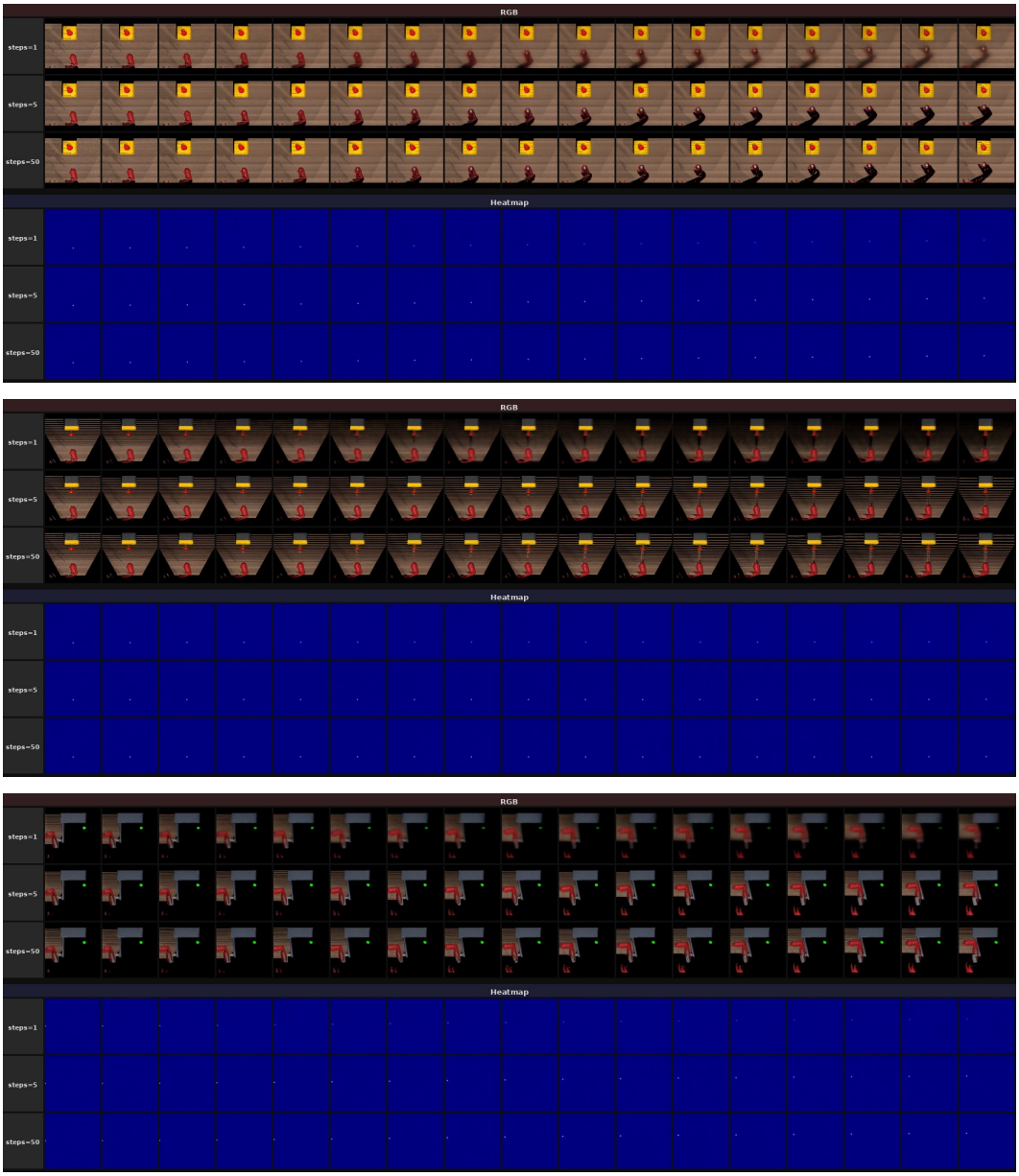

Figure 13: **Visualization of video predictions under different denoising steps (Part I).** Predicted RGB videos and heatmap videos under different denoising step settings. Lower denoising steps lead to visibly lower RGB video quality, while the predicted heatmaps remain relatively stable.

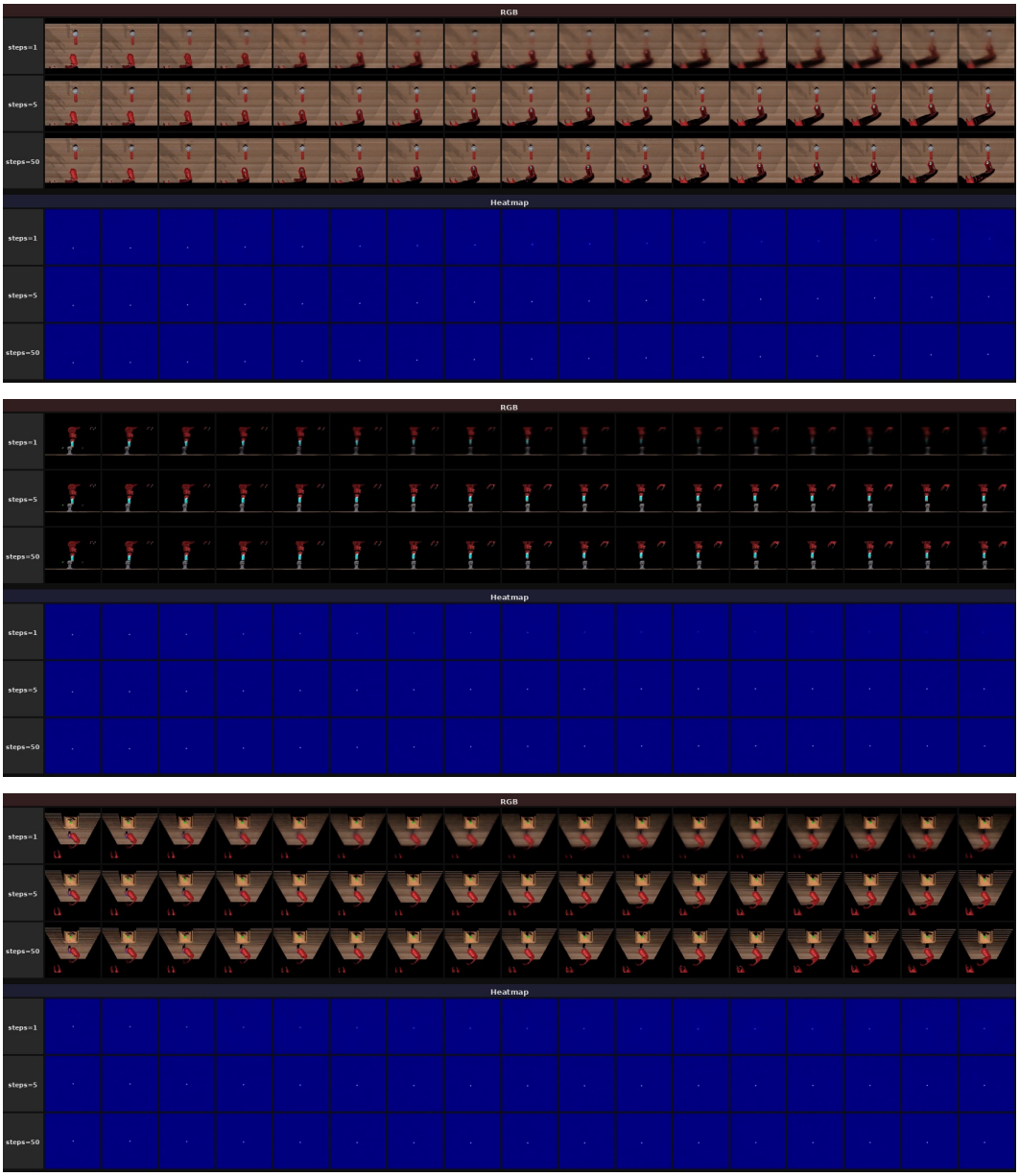

Figure 14: **Visualization of video predictions under different denoising steps (Part II).** Predicted RGB videos and heatmap videos under different denoising step settings. Lower denoising steps lead to visibly lower RGB video quality, while the predicted heatmaps remain relatively stable.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims made in the abstract and introduction accurately reflect the paper's contributions and scope

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have discussed them in Future work.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: The results are measured by experiments, not theoretical analyses.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide detailed descriptions in the method section and appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: The code has not been organized yet.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: All details have been introduced in the main paper and appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: They are introduced in the experiment section.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We have provided them in the experiment section and appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This paper conforms to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This paper is intended solely for academic research, with no expected direct societal impact.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This paper does not have such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have properly cited related papers.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: We do not provide new assets in the paper.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: We do not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: We do not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.