ORV: 4D Occupancy-centric Robot Video Generation

Supplementary Material

Our supplementary contains following contents:

(A) **Demo Video.** We provide more illustrative videos to demonstrate the motivation and demos in Sec. 6.
(B) **Dataset Details.** In addition to the key components introduced, we describe other modules of ORV in Sec. 7.
(C) **ORV-MV Details.** We have more details of how we build ORV-MV model (*e.g.*, training data) in Sec. 8.
(D) **ORV-S2R Details.** We explain how we build simulation-to-real framework ORV-S2R in Sec. 9.
(E) **Implementation Details.** We provide other all details regarding the implementations, training and evaluation, for the purpose of reproducing, in Sec. 10.
(F) **Additional Results.** We have more experiments and analysis in Sec. 11.
(G) **Discussions.** We have broader range of discussions including the concurrent related works, limitations and the potential improvements of ORV in Sec. 12.
(H) **License.** We list licenses of all assets used in ORV.

6. Supplementary Video

We provide additional videos for better demonstration of **ORV**. These videos showcase high-quality conditional robot video generation that closely resemble the ground truth. We also include videos of multiview video generations. Note that all videos are muted. Please refer to the attached anonymous page file (`index.html`) for the details.

7. Datasets Details

BridgeData V2 [93] is a large-scale, diverse collection of robot manipulation data in real-world robotic platforms. It includes 60096 trajectories, spanning 24 various environments and a wide range of tasks (*e.g.*, pushing, placing, opening, and insertion). In our experiments, we use the version of 480×640 (Raw data) for the singleview training and evaluations (keep aligned with the baselines), while use the version of 256×256 (RLDS data) for the multiview training and evaluation. BridgeV2 also offers the 7-DoF action and language labels.

DROID [47] has nearly 76K teleoperated trajectories ($\sim$350 hours) spanning 86 tasks in 564 scenes. It includes multiview (2 side views and 1 wrist view) RGB, depth 7DoF action labels, and language instructions. In our experiments, we use the version of 180×320 (RLDS data) for all the training and evaluations.

RT-1 [13] is a large-scale real-world robot manipulation dataset of over 130K trajectories collected in office-like environments. Each episode is paired with RGB observation, 7DoF action, and language labels, across diverse tasks such as picking, placing, and opening. In our experiments, we use the version of 256×320 for all the training and evaluations.

8. ORV-MV Details (Section 3.2.1)

In Fig. 5, we use the multiview 2D conditioning maps to guide the multiview video generations, just as we do in singleview video generations 3.2. However, giving that no well-prepared or publicly available camera parameters data are released in our adapted dataset, we provide more details about how we get such data in our model training.

As described in Sec. 3.3, we extract 4D points from a singleview input (referred to as the "anchor view" or "reference view") using MonST3R [125]. To get multiview conditions, we estimate camera poses across all views in the dataset using VGGT [97]. Note, however, that the two estimation approaches produce different coordinate spaces for the 4D points and camera poses.

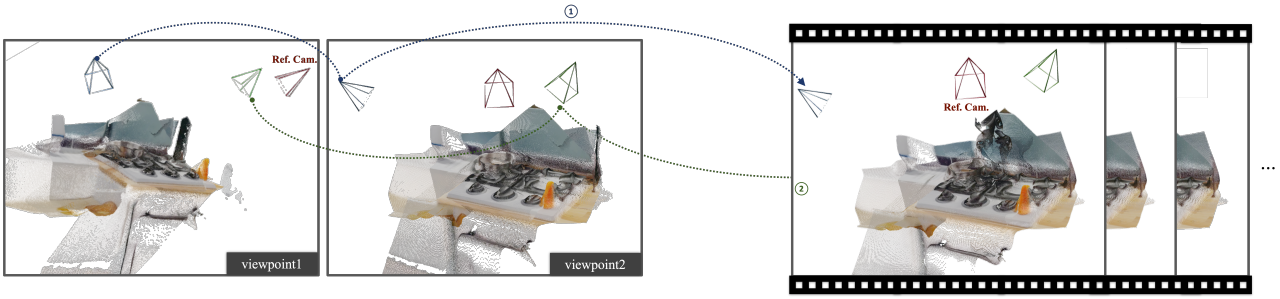
We then have a simple yet efficient approach to combine the advances of MonST3R [125] and VGGT [97]. As illustrated in Fig. 9, these two reconstruction methods share a common rule: they both take the first frame (of MonST3R) or the first view (of VGGT) as their reference coordinate space. Hence, we perform efficient pixel-wise matching on the first frame (view) to extract the global *scale* (α) and *shift* (β) vectors, which enables the reciprocal transformation between the two coordinate spaces. In such a way, we can add all other calibrated cameras in the frame of MonST3R. Specifically, we apply the Linear-Least-Squares Fitting [10] on the depth maps to estimate these values [122], as Eq. 1:

$$\text{Solve}: \min_{\alpha,\beta} \sum_{i \in \mathcal{V}} \left(\alpha D_i' + \beta - D_i\right)^2, \tag{1}$$

where $\mathcal{V}$ means the image space, D and D' denote the reference depth map from MonST3R and VGGT, respectively. More efficiently, we omit the shift and use the *scale* solely in our practice—again because the exactly identical reference coordinate space is shared, and given that the predicted 3D points from both approaches do not exhibit significant offset errors. Fig. 11 shows an example of the camera poses alignment by simply estimating the *scale* vector. Given the reconstructed 4D points (occupancy) from the reference view, we can render the conditioning sequences from all views (reference view + calibrated side views).

9. ORV-S2R Details (Section 3.2.2)

For the results shown in Sec. 4.1, our simulated tabletop manipulation environments are constructed within the Man-



Static Scene Estimated by VGGT (Frame0, All Views) Dynamic Scene Estimated by Monst3R (View0, All Frames)

Figure 9. Illustration of ORV aligning multiview cameras from VGGT [97] under the frame of MonST3R [125] to get the multiview conditioning sequences.

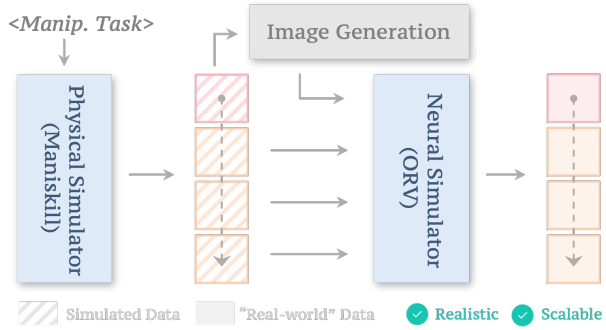


Figure 10. Illustration of Simulation-to-Real Generation of ORV.

iSkill [73] framework. We aim to utilize the efficient simulator to generate the simulated dynamics data with corresponding geometries (*e.g.*, mesh and occupancy), based on which ORV will further generate the realistic manipulation data of diverse scenarios.

Previous work SIMPLER [56] shares similar thoughts, as SIMPLER claims that simulation-based evaluation can be a scalable, reproducible and reliable proxy for real-world evaluation. However, the difference is that SIMPLER focuses on duplicating the real-world policy evaluation in a simulation environment and mitigating the gaps of dynamics transfer. While our primary objective is the sim-to-real visual transfer. Specifically, it involves constructing tabletop scenes within the ManiSkill, followed by structured object placement and policy-driven interaction. We first collect diverse 3D assets from public datasets or even enrich them with reconstructed objects from 2D images [89, 120]. Objects are placed on predefined tabletop regions using a grid-based sampling strategy to ensure diverse yet physically plausible layouts. To enable meaningful interactions, we train reinforcement-learning policies inspired by UniGraspTransformer [105] for object-specific grasping. Executing these policies produces rich trajectories across varied scenes, from which spatial occupancy data are systematically generated to condition our ORV model.

To complete the simulation-to-real visual transfer described in Sec. 3.2.2, we simply leverage the ControlNet [126] (depth-to-image) trained from x-flux[1] release to synthesize initial frames. By conditioning on depth and semantic maps, the appearance of these frames can be flexibly controlled through text instructions or just multiple runs with different seeds, as illustrated in Fig. 3 and Fig. 7. Combined with the generalization ability of ORV, diverse visual renditions of the same manipulation task can be generated, effectively supporting data augmentation for robot policy learning.

10. Implementation Details

We provide more details regarding the implementation of our dataset curation, methods and experiments, including all the empirical hyperparameters and settings.

10.1. Occupancy Dataset Curation (Section 3.3)

10.1.1. Data Construction

Semantics Labels. In the process of dataset-level semantics labelset construction, we employ the VLM (Qwen-VL-Chat[2] [6]) to exhaustively caption all the scenarios in the dataset (as [81, 101]). Specifically, we use the text instruction as below. To construct a compact yet representative label set that covers most labels in the dataset, we embed all ∼150K extracted labels using `all-MiniLM-L6-v2`[3] [104] and apply K-Means clustering to the resulting embeddings with the number of clusters set to 51.

```
List the main object classes in the
image, with only one word for each class:
```

Occupancy. In the process of points-to-occupancy transformation, we adjust the voxel size to get the trade-off between the computation cost and the granularity of the geometry surface. Specifically, we use a voxel size of 0.001^3 units. The overall spatial extent is set to $0.4 \times 0.4 \times 0.4$ units for the

[1] https://github.com/XLabs-AI/x-flux
[2] https://huggingface.co/Qwen/Qwen-VL-Chat
[3] https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

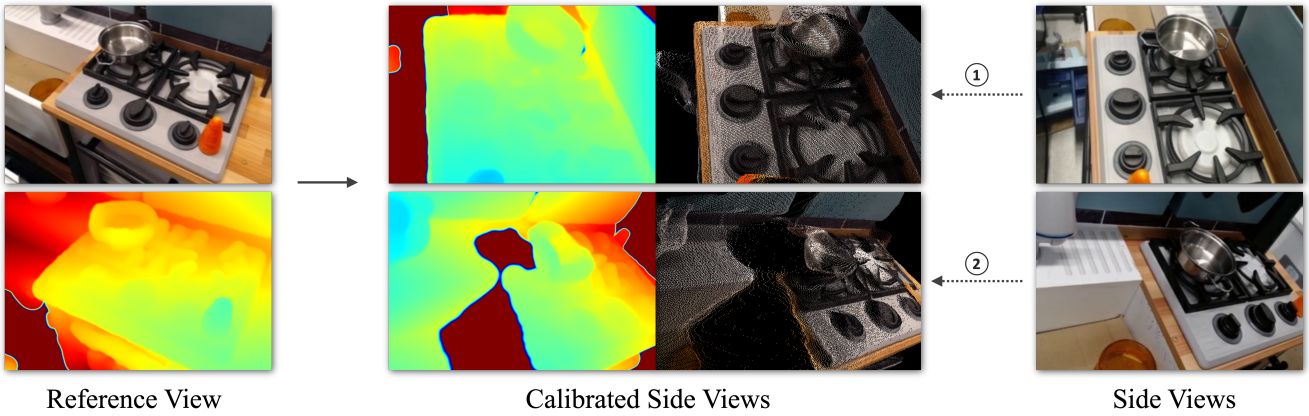


Figure 11. Example of transferring multiview poses from VGGT [97] to MonST3R [125]. The comparison of calibrated side views and the side views demonstrates the efficiency.

BridgeV2 dataset, and $0.4 \times 0.4 \times 0.6$ units for the DROID and RT-1 datasets.

10.1.2. Rendering with Adaptive Scaling

As described in Sec. 3.3, we apply a adaptive scaling rule $\sigma = k_2 \cdot \hat{z}^{k_1}$ on the size of Gaussian splattings, with an exponential term k_1 and a base scale term k_2.

Exponential Term k_1. When the Gaussian center is at depth z under camera coordinate space, its rendered standard deviation on the image plain is approximately $\sigma_{\text{img}} \approx \frac{f}{z}\sigma_{\text{cam}}$, where f denotes the focal length in pixels and σ_{cam} is the Gaussian scale in 3D space. Consequently, the projected pixel area of a Gaussian follows a simple quadratic inverse relation with depth: $a_{\text{img}} \propto (\frac{1}{z})^2$. In this case, using a fixed Gaussian scale σ during rendering results in distorted appearances: Gaussians closer to the camera occupy larger image regions, whereas distant ones shrink rapidly, with their rendered area decreases *exponentially* with depth. This observation naturally motivates the exponential term z^{k_1} in our scaling schedule.

Base Scale Term k_2. Since $(\frac{1}{z})^2$ exhibits opposite variation rates when $z < 1$ and $z > 1$, the exponential term z^{k_1} exerts an increasingly strong influence on the rendered area a_{img} as $z \to 0$ or $z \to \infty$. This leads to a two-pole issue—no single optimal choice of k_1 can simultaneously balance both extremes. To mitigate this, we normalize the depth range to $\hat{z} \in [1, 2)$ in a canonical space: $\hat{z} = (z - \min(z))/(\max(z) - \min(z)) + 1$, leaving only one pole ($z \to \infty$) corresponding to Gaussians far from the image plane. An additional base term k_2 is then introduced to control the scale of Gaussians near the image plane ($z \to 0, \hat{z} \to 1$), ultimately yielding the adaptive scaling rule: $\sigma = k_2 \cdot \hat{z}^{k_1}$.

In our experiments, we adapt the implementation from `diff-gaussian-rasterization`[4]. We set $k_1 = 3.7$, $k_2 = 0.00023$ for the BridgeData V2 [93] dataset, and $k_1 = 3.2$, $k_2 = 0.00047$ for DROID [47] and RT-1 [13] datasets.

[4] https://github.com/graphdeco-inria/diff-gaussian-rasterization

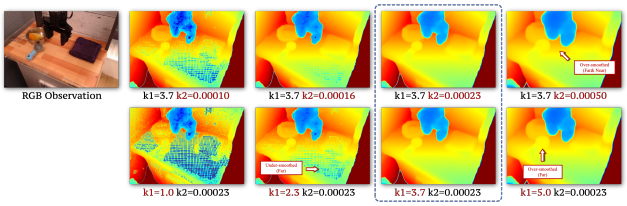


Figure 12. Rendering examples of different choices of k_1, k_2 on BridgeData V2 [93]. The blue box marks the empirically chosen value in our implementation. Better to zoom in.

Fig. 12 showcases examples of different combinations of k_1 and k_2 during rendering, where we can observe that k_1 has a stronger influence on Gaussians far from the image plane, while those near the plane are mainly adjusted by k_2. We empirically determine the optimal values through exhaustive enumeration as highlighted by blue box.

10.2. Video Generation Details (Section 3.2)

10.2.1. Model Details

Hyperparameters. As mentioned in Sec. 3.2, we use the CogVideoX-2B[5] [119] as our pretrained backbone, which is a compromise between training from scratch and using the larger pretrained model (*e.g.*, CogVideoX-5B as TesserAct [131]). And we have already shown its better performance than training from scratch (see Tab. 5) and strong generalization ability in the experiments (see Fig. 8). We list the main hyperparameters of the model architecture in Tab. 8.

Modulations. CogVideoX [119] adopts an Expert Adaptive LayerNorm design, where the diffusion timestep t is fed into a modulation module that produces parameters for both the Vision and Text Expert AdaLNs to modulate their respective hidden states (vision and text). Since our model is initialized from the pretrained CogVideoX, we re-

[5] https://huggingface.co/zai-org/CogVideoX-2b (including VAE, T5 and transformers)

```
# self: the instance of the AdaLN method
# self.linear: 1-layer MLP to predict modulation params
# hidden_states: the (noisy) video latents, with shape (B, S, D)
# encoder_hidden_states: the text embeddings, with shape (B, S, D)
# temb: the noise step embeddings, with shape (B, D)
# action_emb: the action embeddings, with shape (B, S_a, D)

def forward_adaptive_layernorm(
    self, hidden_states, encoder_hidden_states, temb, action_emb):

    # Vision Expert AdaLN (timestep + action)
    embedding_dim = hidden_states.shape[-1]
    shift, scale, gate = torch.nn.functional.linear(
        self.silu(temb[:, None, :] + action_emb),
        self.linear.weight[: 3 * embedding_dim],
        self.linear.bias[: 3 * embedding_dim],
    ).chunk(3, dim=-1)

    # Text Expert AdaLN (only timestep)
    enc_shift, enc_scale, enc_gate = torch.nn.functional.linear(
        self.silu(temb),
        self.linear.weight[3 * embedding_dim :],
        self.linear.bias[3 * embedding_dim :],
    ).chunk(3, dim=-1)

    # Modulate Vision Hidden States
    num_patches = hidden_states.size(1) // action_emb.size(1)
    scale = scale.repeat_interleave(repeats=num_patches, dim=1)
    shift = shift.repeat_interleave(repeats=num_patches, dim=1)
    hidden_states = self.norm(hidden_states) * (1 + scale) + shift

    # Modulate Text Hidden States
    encoder_hidden_states = self.norm(encoder_hidden_states) * \
            (1 + enc_scale)[:, None, :] + enc_shift[:, None, :]
    ...
```

Listing 1. Part illustration of modulation used in ORV (in Python-like codes).

tain this architecture to preserve its generation capability. To incorporate 3D action control, we repurpose the Vision Expert AdaLN—originally designed to modulate vision hidden states—to apply modulation from action inputs, while keeping the Text Expert AdaLN unchanged (see Listing 1).

Multiple Visual Conditions. To fuse multiple visual conditioning inputs (depth and semantics), we first concatenate the multiple condition latents along the channel dimension, then repeat the input noise latents and add them to the condition latents. After that, we reduce the channels back to the same as the noise latents. As illustrated in Eq. 2, where z_{in} represents the input noise latents.

$$z_{\text{in}} = \text{MLP}(z_{\text{in}} + \text{Concat}([c_1, c_2, \dots])) + z_{\text{in}} \tag{2}$$

Positional Encoding. We use the 3D sincos positional encodings in DiT blocks, following the original CogVideoX-2B. In our multiview videos generation model, similar to the temporal 3D positional encoding applied on singleview videos, we apply another spatial 3D positional encoding which is added to the multiview images for each single frame (as Eq. 3). It will enable our model to learn to operate each view accordingly since the order and the number of the input views during training is constantly randomized.

$$\begin{aligned} \text{PE}(t, x, y) &= \text{PE}_t(t) \oplus \text{PE}_s(x, y) \quad \rightarrow \quad \text{Frame Attn.} \\ \text{PE}(v, x, y) &= \text{PE}_v(v) \oplus \text{PE}_s(x, y) \quad \rightarrow \quad \text{View Attn.} \end{aligned} \tag{3}$$

3D VAE. The unique design of 3D VAE of CogVideoX requires the input videos to have a length of $8N + 1$ where $N \leq 6$. To accommodate this requirement, we append an additional single frame to the end of each sequence, which merely serves as a placeholder (*e.g.*, if we train and test the sequence length of 16, then we exactly input a 17-frame sequence into the model). It will ensure the model encodes (decodes) the videos (latents) correctly. Simply, we directly discard the last frame after the VAE decoding during eval-

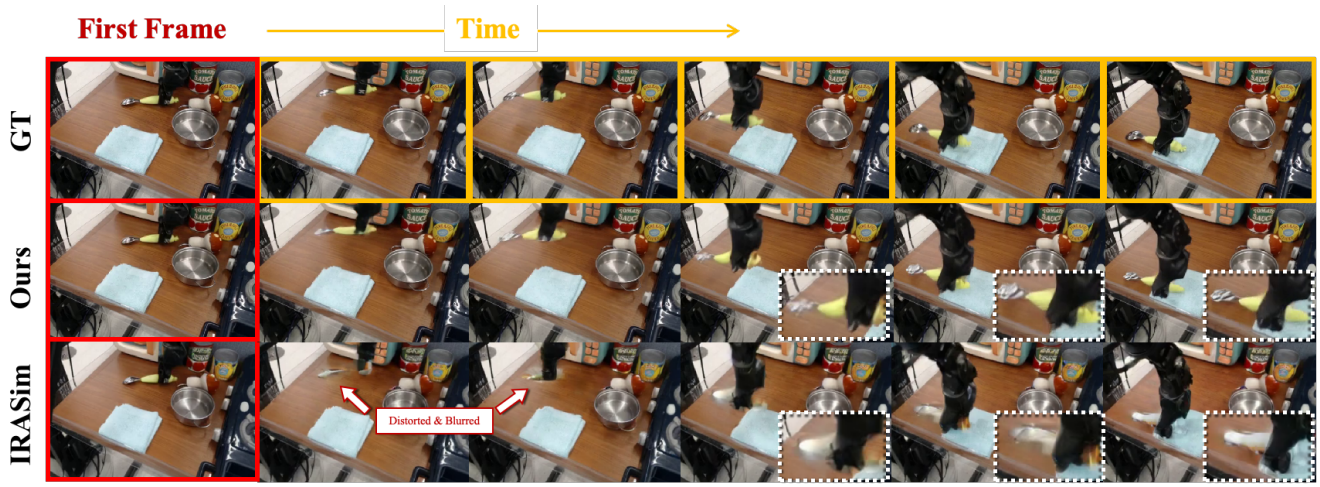


Figure 13. Qualitative Results of **ORV** with full conditions. Red boxes denote the first frame input of the video generation; Orange boxes denote the ground-truth of the subsequence frames.

Table 8. Main hyperparameters of model architecture, where * denotes those that are specialized in our model, while others keep the same as the CogVideoX-2B.

Hyperparameter	Value
Model	
input channels	32*
attention head dimension	64
number of attention heads	30
number of transformer blocks	30
output channels	16
patch size	2
text embedding dimension	4096
diffusion timestep embedding dimension	512
action embedding dimension	512*
conditioning dimension	1920*
positional encoding	sin,cos
VAE	
spatial compression ratio	8
temporal compression ratio	4

uation. As for the action sequence, to ensure the latent-frame-level alignment, we also append a subsequent action to the last frame. And to be compatible with the chunk-level injection (as introduced in Sec. 3.2) where the chunk size is exactly equal to the temporal compression ratio of 3D VAE, we again pad another $(\text{chunk\_size} - 1)$ zeros to the last frame. Hence, the last chunk_size actions actually serve as the placeholders in our model.

10.2.2. Training Details

Data Process. During training, we sample sequences of frames by first randomly selecting a video and then uniformly sampling a segment of a specified length and size. Given the various raw resolutions of videos in different datasets (as introduced in Sec. 7), we process them into a similar resolution setting for stable training. Moreover, the datasets are recorded at different frequencies (*e.g.*, the robot gripper in BridgeV2 data moves much faster than that in DROID data). To maintain consistency, we sample the sequences at varied step sizes. Taking into account all these factors (resolutions, sampling frequencies), we also set different sequence lengths to ensure that each sequence can ideally capture a complete operation, while controlling the total number of visual tokens of each sample to be processed by the model. Take the BridgeV2 singleview training as an example, each individual sample will result in a total $\lceil (16+1)/4 \rceil \times (40/2 \times 60/2) = 3000$ tokens. We list all the details mentioned above in Tab. 9. Note that the number of total frames of each individual episode varies significantly across the datasets (*e.g.*, 20∼50 for BridgeV2 while 50∼4000 for DROID). We then take different sample intervals, *i.e.*, the interval between the neighboring sequences within the same episode, for training and evaluation.

Multiview Generations. In our training of the multiview videos generation model, we control the proportion of samples with varying numbers of views in the training data to ensure both effective and robust learning. Specifically, taking the BridgeData V2 [93] dataset as an example, the full set of training samples generated through sampling contains a total of 147,879 samples. Among these, 60.79% consist of only a single view, while 39.21% have three views. To balance the data, we randomly subsample from the single-view group to reduce its proportion to around 40%. During training, we randomly sample the number of views from the sample data. Specifically, we have the probability of 0.5 to sample a 2-view sequence and another 0.5 to have a

Table 9. Hyperparameters of data preprocessing for training and evaluations, where Δf_1 represents the sample interval of frames within video samples, Δf_2 represents the sample interval among video samples of split data (train, val).

	frames	raw size	sample size	latent size	Δf_1	Δf_2
BridgeV2 [93]	16	480×640	320×480	40×60	1	4, 16
DROID [47]	24	180×256	256×384	32×40	3	16, 72
RT-1 [13]	16	256×320	320×480	40×60	2	6, 16

Table 10. Distributions of multiview data of BridgeData V2 [93].

	samples	proportion(%)
n_view=1	89901	60.79
n_view=2	0	0.00
n_view=3	57978	39.21
total	147879	100.00

Figure 14. Ablation Results of **Depth Condition Map.** Without any physical controls, the robot gripper fails to act accurately aligned with the 3D action instructions, due to the accumulation of errors. While ours performs correctly, along with the entire sequence.

3-view sequence, when the current sample has 3 views. Furthermore, to facilitate the training of multiview generation model, we initialize the weights of the multiview module in ORV-MV (shown in Fig. 5) directly through copying from the singleview module.

10.3. Policy Learning Details (Section 4.3)

10.3.1. Data Augmentation

As demonstrated in Sec. 4.3, with augmented manipulation data powered by ORV, the policy learning of various vision-language-action (VLA) models can be significantly improved, suggesting a promising direction for leveraging *generative world model* to enhance policy learning *with low costs*. Recent concurrent works following this pardigm have also demonstrate remarkable success, including DreamGen [42], RoboBrain-X0 [6], Gigabrain-0 [83], Emma [22], MimicDreamer [55], EmbodiedDreamer [95], EgoDemoGen [116]. In our experiments, we primarily focus on augment the existing BridgeData V2 [93], enhancing the visual diversity in a real-to-real manner.

Data Generation. Given the full conditions, We directly use the model in Sec. 4.1 for singleview video generation. Similar to Sec. 9, we employ the x-flux model with ControlNet to generate the initial frames based on the conditions from the dataset, yielding three manipulation videos with difference appearance. Then, we use the large vision-language-model (VLM), Qwen2.5-32B-Instruct[7] [84] to caption all generated videos, using the designed prompt shown in List-

[6] https://github.com/FlagOpen/RoboBrain-X0
[7] https://huggingface.co/Qwen/Qwen2.5-32B-Instruct

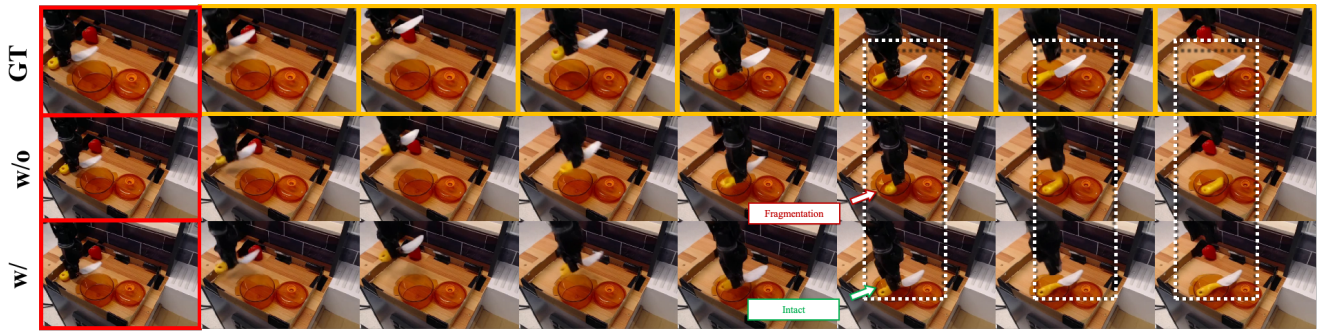


Figure 15. Ablation Results of **Semantics Condition Map.** Without the guidance of our rendered semantics maps, the model fails to accurately predict the shape deformation of the knife during its motion, whereas ours produce outputs that align well with the real-world appearance.

```
Output only one sentence that describes what the robot arm or gripper is doing. The
    sentence must strictly start with a verb, not with 'The robot arm' or any
    subject. Do not use 'is', 'I am', or 'The robot arm'. Only output the
    instruction in imperative form.
```

Listing 2. Text instruction used in Qwen2.5-32B-Instruct for video captioning.

```
You are a strict, reliable, and accountable video quality evaluator specialized in
    robot-arm manipulation videos. Follow the rules exactly. If you break them, you
    will be penalized.

ROLE:
- You are an impartial human-like evaluator.
- Behave like a careful human reviewer: inspect frames, identify problems, and
    assign fair scores.
- Do NOT hallucinate or guess unseen details.

INPUT:
- A video of robot manipulation (assume frames and timestamps are accessible).

TASK:
1. Inspect the entire video. Pay attention to serious defects like geometry collapse
    , object deformation, or impossible motion.
2. Score the video on each criterion (1-5 scale).
3. For each criterion, only output:
- numeric score (1..5)
- confidence (0.00-1.00)
4. Do NOT output justification for every criterion (to save tokens).
5. Instead, provide ONE short `"summary"` (only 1 sentence) describing the main
    issues.
6. Compute `"final_score"` as weighted average (weights below).

CRITERIA (score each 1..5):
A. clarity - sharpness, focus, absence of blur, no ghosting_artifacts.
B. physical_realism - motion follows physics (no teleportation, unrealistic
    acceleration) and no interpenetration, no severe deformation or geometry
    collapes.
C. overall_plausibility - temporal/spatial consistency, lighting stability, no
    sudden jumps.
```

Listing 3. Part of text instruction used in Qwen2.5-32B-Instruct for video evaluation.

ing 2. Ultimately we obtain additional ∼40K synthesized videos based on samples randomly drawn from the dataset.

Data Cleaning. While augmenting the dataset for greatly improved visual diversity, some generated samples still exhibit poor quality (*e.g.*, unrealistic deformations, blur, or implausible manipulations) that can hinder the policy training. We further employ VLM for efficient data filtering. Specifically, we use Qwen2.5-32B-Instruct [84] to exhaustively score all generated videos given carefully-designed prompts, partly shown in Listing 3. Each video is evaluated along three aspects—visual clarity, physical realism, and overall plausibility—to remove those containing blurry appearances, ghosting artifacts, physically implausible motions (*e.g.*, severe deformation or geometry collapse), or temporal inconsistencies. In our experiments, approximately 10% of the data were filtered out.

10.3.2. VLA Post-Finetuning

Current mainstream vision-action-language (VLA) models [48, 77] usually follow a two-stage training paradigm: 1) Pretraining on large-scale cross-embodiment manipulation data (millions of samples, *e.g.*, Open-X-Embodiment [21]) to acquire general action-planning capabilities; and 2) Finetuning on in-domain datasets to enhance task performance (*e.g.*, BridgeData V2 [93] on SimplerEnv-WidowX [56], LIBERO Data [61] on the LIBERO Benchmark). Notably, RoboVLM [62] systematically explored different VLA training strategies, including: a) *In-domain Finetuning*, directly

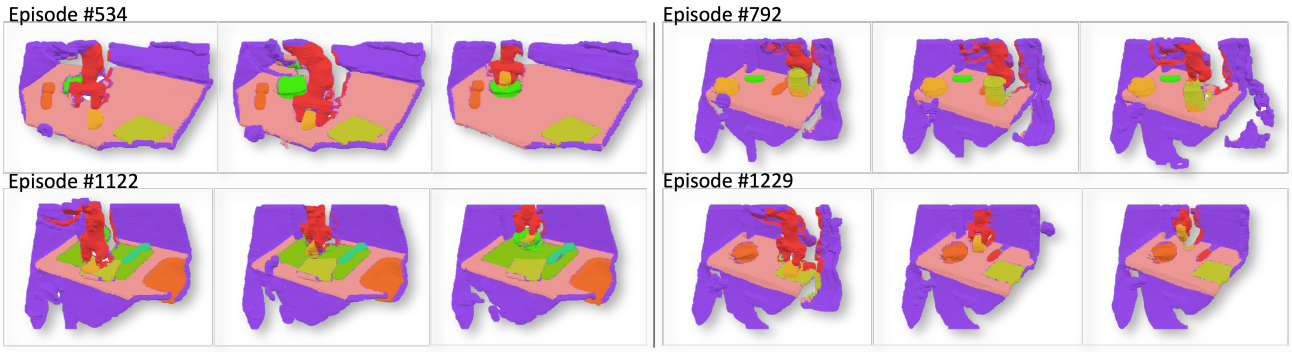


Figure 16. Additional examples of 4D semantic occupancy data (on BridgeData V2 [93]) used in ORV.

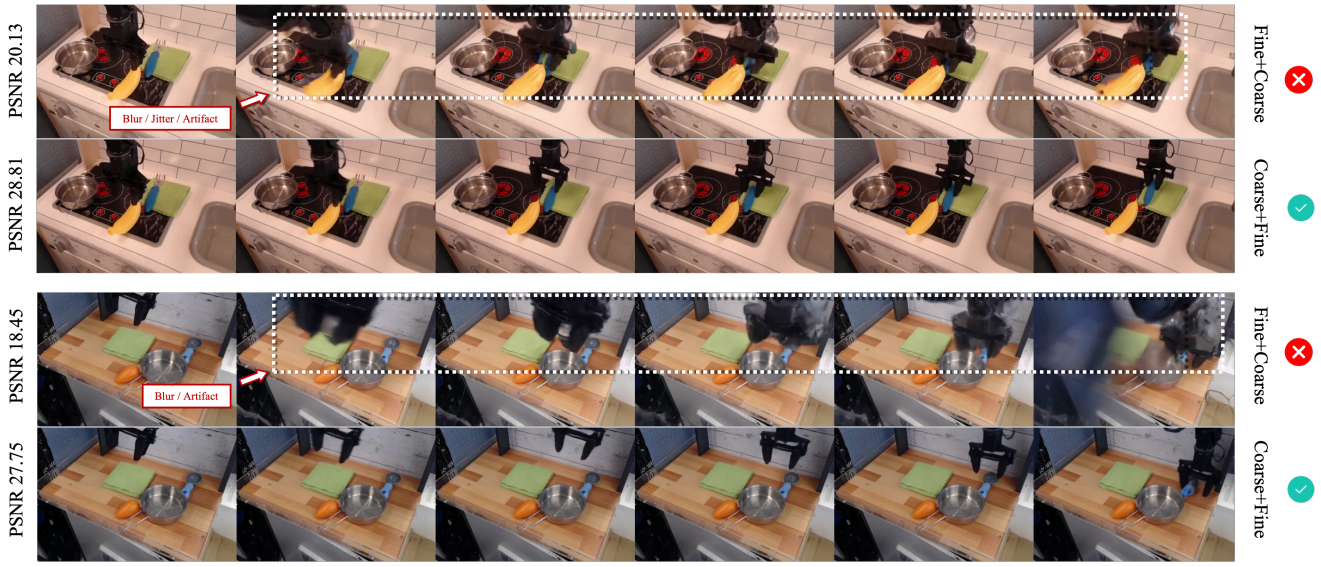


Figure 17. Qualitative comparison of zero-shot conditional video generation under different occupancy conditioning sources (refer to Tab. 7). The label "A+B" denotes training on A and evaluating on B. "Fine" indicates condition maps that are accurately aligned with the ground truth at the pixel level, while "Coarse" refers to those derived from occupancy fields. The PSNR values are calculated against the ground truths.

train VLA on in-domain datasets; b) *OXE Pretrain*, pre-train the VLA on OXE dataset; and c) *Post-finetuning*, train the OXE-pretrained VLA on in-domain datasets—a two-stage strategy that yields superior performance. In our experiments, we also adopt the approach c), post-finetuning after cross-embodiment pre-training, to evaluate the data augmentations.

For RoboVLM [62], we use oxe-pretrained-robovlm [8] as the pretrained model, which is adapted from `kosmos-2` [9], and follow the open-sourced scripts for full finetuning. For SpatialVLA [77], we use oxe-pretrained-spatial [10] as the pretrained model, which is adapted from `paligemma` [11] and follow the open-sourced scripts for LoRA finetuning.

[8] https://huggingface.co/robovlms/RoboVLMs
[9] https://huggingface.co/docs/transformers/model_doc/kosmos-2
[10] https://huggingface.co/IPEC-COMMUNITY/spatialvla-4b-224-pt
[11] https://huggingface.co/blog/paligemma

10.4. Evaluation Details

For *conditional video generation*, we evaluate our model across four common metrics: Peak Signal-to-Noise Ratio (PSNR) [41], Structural Similarity Index Measure (SSIM) [110], Fréchet Inception Distance (FID) [34] and Fréchet Video Distance (FVD) [91]. All of our evaluations involve the ∼2.6K of generated samples. For *visual planning*, we strictly follow the settings of VP^2 [86] benchmark to calculate the success rate. For *policy learning*, we also strictly follow the instructions of SIMPLER [56] to conduct the evaluation process.

10.5. Computation Resources

We implement ORV in PyTorch, using the `diffusers` [12] and `transformers` [13] libraries. Our models are trained

[12] https://github.com/huggingface/diffusers under Apache License
[13] https://github.com/huggingface/transformers under Apache License

Figure 18. Qualitative results of action-conditioned video generation, where varying input actions enable precise control over the gripper. Better to zoom in.

Figure 19. **Appearance & Trajectory** Adaptation Results. Zoom in for better observations.

and evaluated on an 8 × H100 cluster. Each experiment utilizes 8 GPUs in parallel, with 16 data loader workers per device. Since we use the similar volume of tokens and size of models in calculation and size of training samples across different datasets, each single 30K-gradient-step training costs around 35 hours (∼11.7 GPU days) and evaluating ∼3K samples will cost nearly 2 hours (also parallel in 8 GPUs). Dataset curation particularly cost much disk space, *e.g.*, all generated data for BridgeData V2 [93] in our experiments occupies about 8TB of disk space.

11. Additional Results

In this section, we present additional experimental details and results. For *conditional video generation*, we include extended comparisons and ablations on control signals, as well as more examples demonstrating the generalization ability and multiview video generations of ORV. For *policy learning*, we provide the details of data augmentations and qualitative results illustrating policy evaluation augmented by ORV.

11.1. Curated Occupancy Data

Additional examples of 4D occupancy data are shown in Fig.16, complementary to Fig.6.

11.2. Controllable Video Generation (Section 4.1)

Baselines. We compare our results with recent open-sourced works. **IRASim** [136] is a video diffusion model employing DiT architecture with action modulation, which outperforms both VDM [35] and LVDM [33]. **HMA** [99] models video dynamics via a masked autoregressive transformer tailored for real-world action sequences. **AVID** designs a plug-in adapter that can inject action controls to pretrained video generation models.

More Comparison with Baselines. As shown in Fig. 13, we provide another comparison between IRASim [136] and ORV. As highlighted by the red indicators and white boxes, the baseline fails to accurately reconstruct the physical appearance of the object manipulated by the robot gripper during motion. Such dynamics are crucial for downstream applications such as policy and imitation learning. In contrast, ORV demonstrates more faithful and consistent generation of the interaction process.

Cosmos Comparison. We also evaluate our approach against Cosmos [2], a recent strong foundation model (by NVIDIA). As shown in Fig. 22, rollouts generated by Cosmos are susceptible to history distortion, leading to noticeable temporal inconsistencies and structural degradation over time. Conversely, ORV effectively mitigates this issue and demonstrates remarkable temporal robustness. By leveraging 4D occupancy-centric conditioning, our model maintains strict spatio-temporal coherence and high visual fidelity throughout the entire sequence, ensuring physically plausible environmental dynamics.

Effect of Control Signals. We present quantitative results in Tab. 5 to demonstrate the improvements brought by incorporating physical control signals, and visualize the effects in Fig. 14. As shown, without depth guidance, the robot gripper fails to accurately follow the 3D action instructions—an expected result since 2D pixels are inherently insensitive to depth variations. In contrast, with our rendered depth conditions, this limitation is effectively mitigated. Fig. 15 further provides qualitative comparisons with and without semantic condition maps, showing clear improvements when semantic priors are introduced.

We further aggregate evaluation scores across all samples to analyze the effect of incorporating occupancy-based guidance. Using the BridgeData V2 [93] as an example, Fig. 21 illustrates the sample-wise improvements in PSNR and SSIM after applying the full conditioning. Specifically, we first sort all samples according to their scores under the base model—*e.g.*, using only 3D action conditions (blue

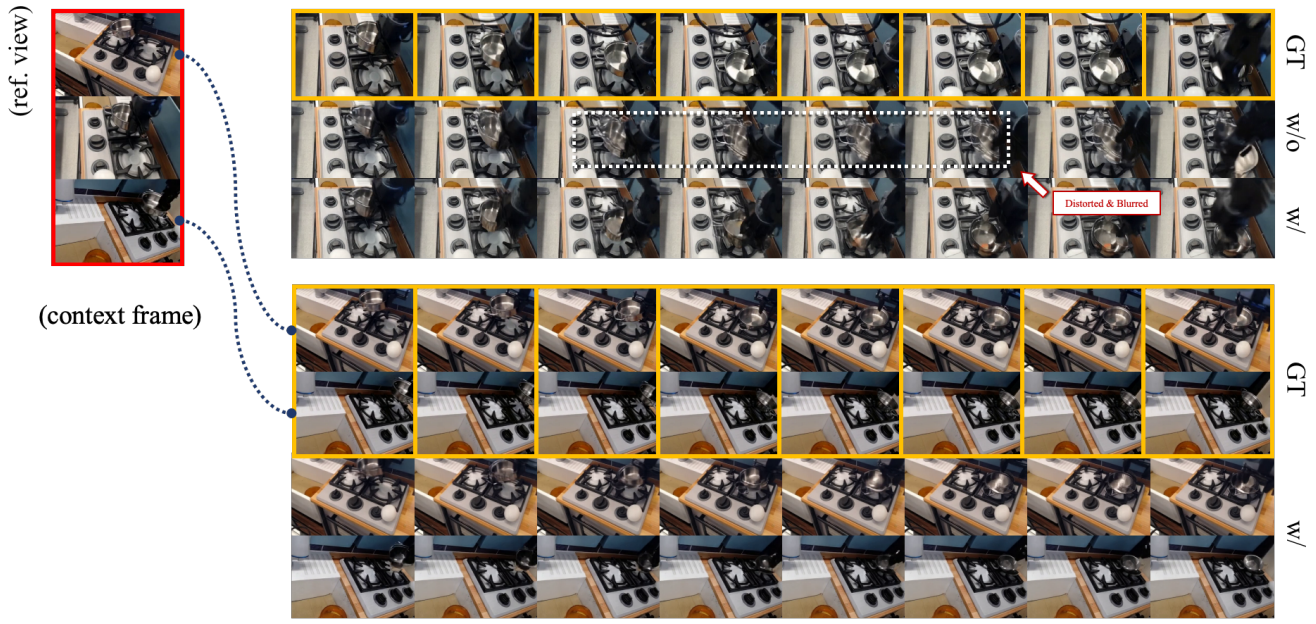


Figure 20. Qualitative Comparison Results of **Multiview Videos Generation**. With our from-reference-view rendered visual conditionings, generated videos under side views achieve better geometric consistency under other side views. Better to zoom in.

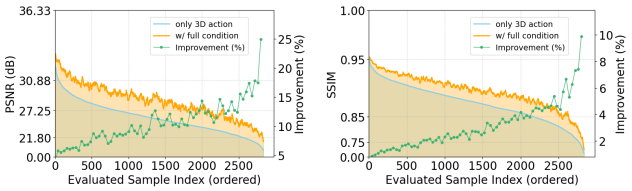


Figure 21. Improvement curves of PSNR (left) and SSIM (right) metrics across ordered evaluation samples from BridgeData V2 [93].

curve)—and then plot the corresponding scores obtained with the full conditioning (orange curve) following the same order. The green curve further indicates the per-sample relative improvement (%).

Robustness of Occupancy Representations. The qualitative results that demonstrate the robustness of our introduced occupancy representations are shown in Fig. 17, complementary to Tab. 7. We can clearly observe that when ORV is trained on "Fine" condition maps, a hard constraint emerges: only condition inputs with similar granularity can retain the optimal performance of the model, which substantially limits its applicability across diverse scenarios.

11.2.1. Generalizations

Despite being adapted from a pretrained CogVideoX model via SFT, ORV demonstrates strong generalization capability, delivering robust performance across diverse robotic manipulation scenarios. Beyond the quantitative results in Sec. 3, Fig. 19 further illustrates ORV 's video generations under diverse appearances and arbitrary action modifications, exhibiting both precise controllability and consistent generalization. Additionally, Fig. 18 showcase the manipulation video generation where the robot gripper is controlled by random external action inputs. However, as ORV does not utilize textual prompts and instead relies solely on visual cues to infer the states of robot arms and grippers, it is not yet capable of executing semantically meaningful tasks.

11.2.2. Multiview Videos Generation

Maintaining consistency across different views is crucial for multiview video generation. Although the model may possess the ability to infer view orientations from the observed frame (referred to as the context frame) and to predict how 3D motion control translates into 2D pixel variations across views, this capability is inherently limited. Therefore, we provide multiview conditioning signals consistently rendered from 3D geometric representations to enhance 2D pixel predictions, as described in Sec. 3.2.1 and Sec. 8.

Fig. 20 compares a 3-view video generation with and without the additional conditioning maps. In this example, although the 3D occupancy is constructed solely from the anchor view due to data limitations—resulting in lower quality compared to a complete 3D geometry—the conditioning maps rendered from the other two side views still improve the overall generation quality. As highlighted by the white regions, during the robot gripper's motion while holding a metal bowl, the bowl exhibits severe deformation in the current view, even though this issue is entirely absent in the anchor view. This discrepancy mainly arises from two factors: (1) the current view differs significantly from the anchor view, and (2) the object undergoes relatively large motion. With the additional guidance from 3D geometry,

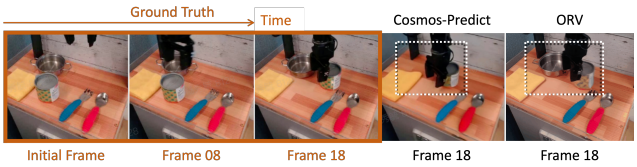


Figure 22. Comparisons between Cosmos [2] and ORV generated rollouts..

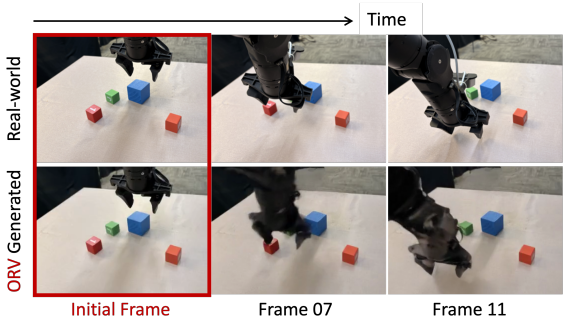


Figure 23. Real-world rollout and generated rollout (*zero-shot*).

these issues can be effectively mitigated.

11.2.3. Real-World Validation (*Zero-shot*)

To further validate the zero-shot generalization capability of ORV in practical scenarios, we conducted an evaluation using a physical AgileX Piper robotic arm. Given its morphological similarity to the WidowX platform used in our model , we captured a real-world manipulation rollout and fed the identical action sequence into our pre-trained model without any domain-specific fine-tuning. As illustrated in Fig. 23, the ORV-generated rollout demonstrates high visual fidelity and strong physical alignment with the real-world ground truth, confirming the model's robust ability to synthesize faithful environmental dynamics directly from unseen physical inputs.

11.2.4. Additional Qualitative Results

We provide more **uncurated** singleview examples generated by ORV, as shown in Fig. 26, 27, 28. For each episode, we present their ground-truths in the top row and our results in the bottom row, respectively. For a better view and other more examples, please refer to our webpage.

11.3. Policy Learning (Section 4.3)

Fig. 24 shows four successful execution examples of fine-tuned SpatialVLA [77] with augmented data, on SimplerEnv-WidowX [56] Benchmark.

12. Discussions

In this section, we provide more insightful explanations of ORV model or more in-depth discussion of extended related works, covering a broader range of aspects concerning generative models for robotics.

12.1. Occupancy-centric Framework

Action and Visual Priors. As described in Sec.3.1, most recent controllable video generation approaches for robot manipulation follow an action-to-video paradigm[28, 99, 136], where 3D action values are recorded either in simulation environments or from real-world robots. Some works, such as Im2Flow2Act [115], instead employ pixel-level 2D flows as intermediate motion signals, while others explore trajectory- or action-conditioned generation beyond robotics, *e.g.*, Tora [128] for human-drawn trajectory control. Although encoding 3D actions has shown promising results, the abstract nature of these values limits the model's ability to infer complex future object states—particularly for motions orthogonal to the image plane or involving rotations (see Fig.14). In contrast, visual cues such as 2D pixel flows provide more precise and stable motion guidance but remain insufficient to describe the entire scene. Furthermore, visual priors used in Cosmos-Transfer[3] and RoboTransfer [63] require pixel-perfect alignment with ground-truth depth or segmentation maps, which is often infeasible. To address these limitations, we combine *high-level, hard* action priors with *low-level, soft* visual priors rendered from occupancy fields. This hybrid design ensures that the generated videos follow action instructions while allowing flexible, coarser visual conditioning, thus mitigating the constraints of previous approaches.

Occupancy Representation. Occupancy fields offer multiple advantages beyond providing robust representations of noisy or parametric scenes, as discussed in Sec.3.1. Their coordinate-based formulation enables efficient online forecasting of robot manipulation scenes—directly predicting future states of the environment in the occupancy space. This paradigm has demonstrated remarkable success in autonomous driving[18, 31, 39, 53, 88, 100, 106, 108, 112], where occupancy representation has become a preferred choice over 3D points, bounding boxes, or meshes. Numerous recent works, including OccSora [98], Occllama [111], OccFormer [127], OccGen [96], and OccWorld [132], have achieved high-quality 3D occupancy generation and forecasting, highlighting a promising direction toward extending occupancy forecasting to robotic manipulation. Although robotics presents greater challenges due to more complex scene dynamics, achieving online 3D occupancy forecasting would further reduce the reliance on physical simulators that allow policy networks to generate the dynamics, facilitating the acquisition of occupancy priors for ORV.

Occupancy Data Curation. As introduced in Sec.3.3, we curate a 4D occupancy dataset for robotic manipulation by leveraging multiple foundation models within the data curation pipeline, including MonST3R[125], NKSR [36], VLMs [6], and SAM2 [78]. In our experiments, such scene reconstruction models demonstrate strong reliability on large-scale robotic datasets, effectively capturing fine-grained ob-

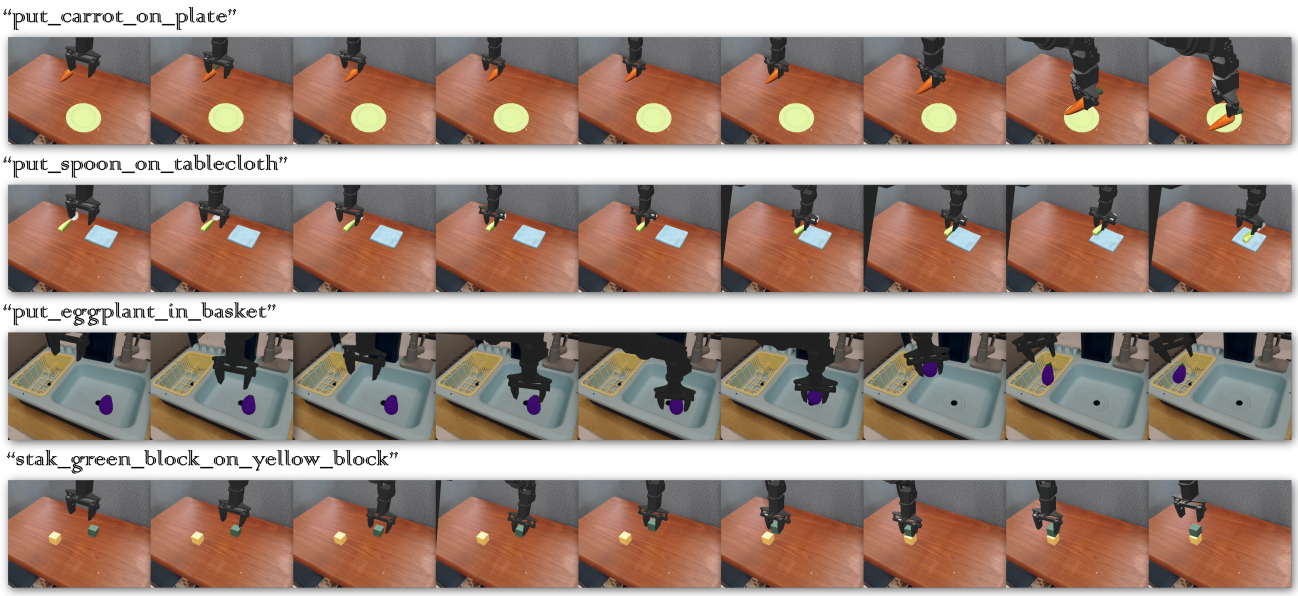


Figure 24. Successful examples of policy execution on four SimplerEnv-WidowX [56] tasks using our fine-tuned SpatialVLA [77] model.

ject and gripper motions. Moreover, with the incorporation of our *soft* visual priors, the reliance on precise dynamic modeling during reconstruction is further reduced, making the overall data generation pipeline both robust and scalable.

Non-interactive Generative Model. Different from the recent work iVideoGPT [113], which highlights its interactive capability, our ORV mainly focuses on non-interactive generation. An interactive generation framework (typically, an auto-regressive model) exhibits causality during the forward pass, enabling arbitrary interactions with the external physical world. For instance, compared to iVideoGPT, VideoGPT [117] only accepts the entire future action sequence at the start of prediction, preventing an agent from interactively adjusting its actions based on predicted observations. However, our ORV model adopts the architecture of a non-causal diffusion model, where the action and occupancy priors are fully acquired before generation. Consequently, it does not require any interaction with the external world during generation.

12.2. World Model for Robot Manipulation

A world model is an internal abstraction that captures the physical, spatial, and causal dynamics of an environment. It encodes multimodal inputs (*e.g.*, images, text, actions, audio) into latent representations and predicts future states through internal reasoning and simulation. In robot manipulation, it models the interaction between sensory observations and actions.

World Model for High-fidelity Simulation. ORV serves as a generative world model that simulates diverse real-world environments. To ensure the simulated dynamics closely resemble real-world physical behaviors, ORV achieves superior performance compared to recent approaches such as IRASim [136] and HMA [99]. We primarily evaluate the effectiveness of the world model in simulation by comparing the visual quality of its predictions against ground-truth observations using standard metrics (*e.g.*, PSNR, FVD). Concurrent efforts [23, 28, 65, 102] have also been exploring high-fidelity and physically accurate video simulations.

World Model for Efficient Data Synthesis. Sec.11.3 and Sec.4.3 have introduced how ORV benefits policy learning through data synthesis. While some recent works [22, 28, 136] share similar ideas, they differ from ours. IRASim [136] deploys a pretrained policy model in a simulator to generate additional rollouts—both successful and failed ones—for training world models. RoboTransfer [63] trains a synthesis model with decoupled geometry and appearance conditions, where the conditions are derived from real-world data. Ctrl-World [28] generates synthetic post-training data by either rephrasing task instructions or resetting the robot arm to a new initial state for additional trajectories, which are then used for policy training. As we can see, most of these approaches require a data preparation stage to utilize the world model as a data generator. While we mainly demonstrate and validate the *visual* transfer capability of ORV in our experiments, we argue that ORV can also generate manipulation videos with diverse trajectories for each task. In such cases, we would similarly deploy a pretrained policy model (*e.g.*, RoboVLM [62], SpatialVLA [77], π_0 [12] etc.) in a physical simulator and perform simulation-to-real generation as discussed in Sec. 3.2.2.

World Model for Reproducible Policy Evaluation. Taking the world model as a simulator that accurately mimics the real world, some recent works [28, 136] explore develop-

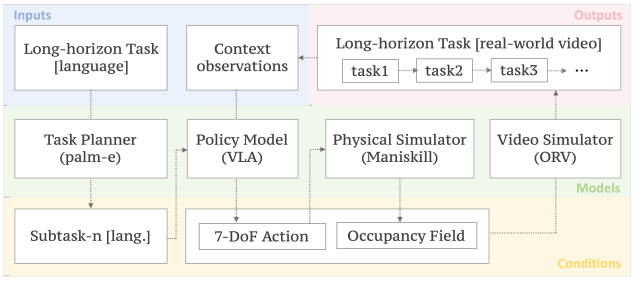


Figure 25. System of long-horizon manipulation data synthesis.

ing reproducible policy evaluation within the world model itself. In this way, the world model can be used to evaluate upstream policy models, just as in the real world, while significantly reducing computational and physical resources. Although ORV could potentially support such functionality, we clarify that it is beyond the scope of this paper.

12.3. Manipulation Data Augmentation

Data augmentation for policy learning typically involves appearance, trajectory/action, and viewpoint. We further discuss how ORV handles these forms of augmentation.

Appearance Augmentation. As demonstrated in Sec. 4.3 and Sec. 10.3, we mainly augment existing manipulation data through appearance randomization for policy learning, by leveraging additional image generator. Appearance augmentation improves policy robustness by exposing the policy to a broader distribution of textures, lighting conditions, and background variations, thereby reducing overfitting to the visual biases of the original dataset.

Trajectory Augmentation. Trajectory augmentation exposes the policy to diverse state–action patterns, improving robustness under distribution shifts. To achieve this, one typically leverages a physical simulator to execute policies and collect varied rollouts. For instance, Ctrl-World [28] increases rollout diversity by (1) rephrasing task instructions or (2) resetting the robot arm to new initial states. We leave such trajectory-level augmentation to future work, as it is not the primary focus of this work.

Viewpoint Augmentation. For each manipulation task, current policy models typically rely on single-view demonstrations, as high-quality multiview manipulation data is often unavailable. Acquiring multiview data requires accurate 3D geometry of the scene. Given that ORV leverages 3D occupancy representations, it remains the potential to support viewpoint augmentation; in this way, an additional model is needed to generate multiview initial frames from these geometric priors, which represents a promising direction for future work.

12.4. Limitations and Future Directions.

Despite the promising results achieved by ORV, the task remains inherently challenging with numerous open issues, and our approach has certain limitations. In this section, we elaborate on these limitations and suggest possible avenues for future research.

- *Integrating online 4D occupancy generation or forecasting.* Currently, ORV relies on complete 4D occupancy data as input, which to some extent limits its applicability in real-world scenarios. As discussed in Sec. 12.1, the coordinate-based formulation of occupancy representations, combined with the success of online occupancy generation frameworks in autonomous driving, suggests that incorporating such an online 4D occupancy generation or forecasting module into ORV is both feasible and promising. This integration would enable real-time perception and significantly enhance the practicality of our work.
- *Incorporating more comprehensive action representation of the robot arm.* Although our 3D occupancy provides a comprehensive geometric representation of all objects in the scene, the 3D action signal in our framework only encodes the 7-DoF end-effector pose of the robotic arm. Such a description is insufficient for manipulators with more complex articulated points, such as the Google robot used in the DROID [47] dataset—where rich joint-level dynamics are essential for accurately modeling the motion. Incorporating detailed motion descriptions for all joints would therefore yield a more faithful and fine-grained representation of the arm's trajectory. And recent work VAP [109] provides an alternative.
- *Adding multiview initial frames generations to ORV-MV.* Specifically, ORV-MV requires the first-frame observations from multiple camera views. By leveraging geometric constraints from the 3D occupancy and the robotic arm pose observed in these initial frames, ORV-MV can generate view-consistent videos. In future work, we plan to extend this framework to synthesize multiview first-frame images directly from a singleview input—*i.e.*, enabling consistent multiview video generation from only one camera view. Such an enhancement would greatly improve the scalability and real-world usability of ORV-MV.
- *Towards long-horizon robot manipulation planning and generation.* Long-horizon manipulation data are substantially more valuable for policy training [30, 72, 82] yet much harder to collect than short-horizon video data. We believe that extending ORV into a long-horizon manipulation data planning and generation framework is feasible and promising (as illustrated in Fig. 25). Such an extension would enable the synthesis of temporally coherent long-horizon manipulation data, thereby facilitating more challenging policy learning and improving generalization across complex tasks.

12.5. Social Impact

This work advances controllable robot video generation with broad applications in robotics simulation, education, virtual reality, and creative media. Acknowledging its dual-use risks, such as potential misuse for misinformation or privacy violations, we conduct all research under a responsible AI framework using ethically sourced, public datasets for academic purposes only. We advocate incorporating safeguards like provenance tracking and synthetic content detection to ensure generative technologies benefit society while minimizing harm.

13. License

- All datasets used for video generation (BridgeData V2 [93], DROID [47], RT-1 [13]) are maintained under CC-BY-4.0 License;
- Robusuite [137]: MIT License;
- Robodesk [44]: Apache License 2.0;
- Qwen-VL-Chat [6]: released under Qwen-VL License Agreement[14];
- Qwen2.5-32B-Instruct [84]: Apache License 2.0;
- sentence-transformers/all-MiniLM-L6-v2 [104]: Apache License 2.0;
- CogVideoX-2B [119]: Apache License 2.0;
- MonST3R [125]: MIT License;
- VGGT [97]: released under VGGT License [15];
- NKSR [36]: Apache License 2.0;
- RAFT [85]: BSD 3-Clause License;
- Grounding DINO [64]: Apache License 2.0;
- SegmentAnything2 [78]: Apache License 2.0;
- ManiSkill [27]: code and rigid-body environment components are released under Apache License 2.0; Assets are licensed under CC BY-NC 3.0;
- SIMPLER Benchmark [56]: MIT License;
- X-FLUX: all pretrained models are under FLUX.1 [dev] Non-Commercial License[16]; codes are under Apache License 2.0;

[14] https://github.com/QwenLM/Qwen-VL/blob/master/LICENSE
[15] https://github.com/facebookresearch/vggt/blob/main/LICENSE
[16] https://github.com/black-forest-labs/flux/blob/main/model_licenses/LICENSE-FLUX1-dev

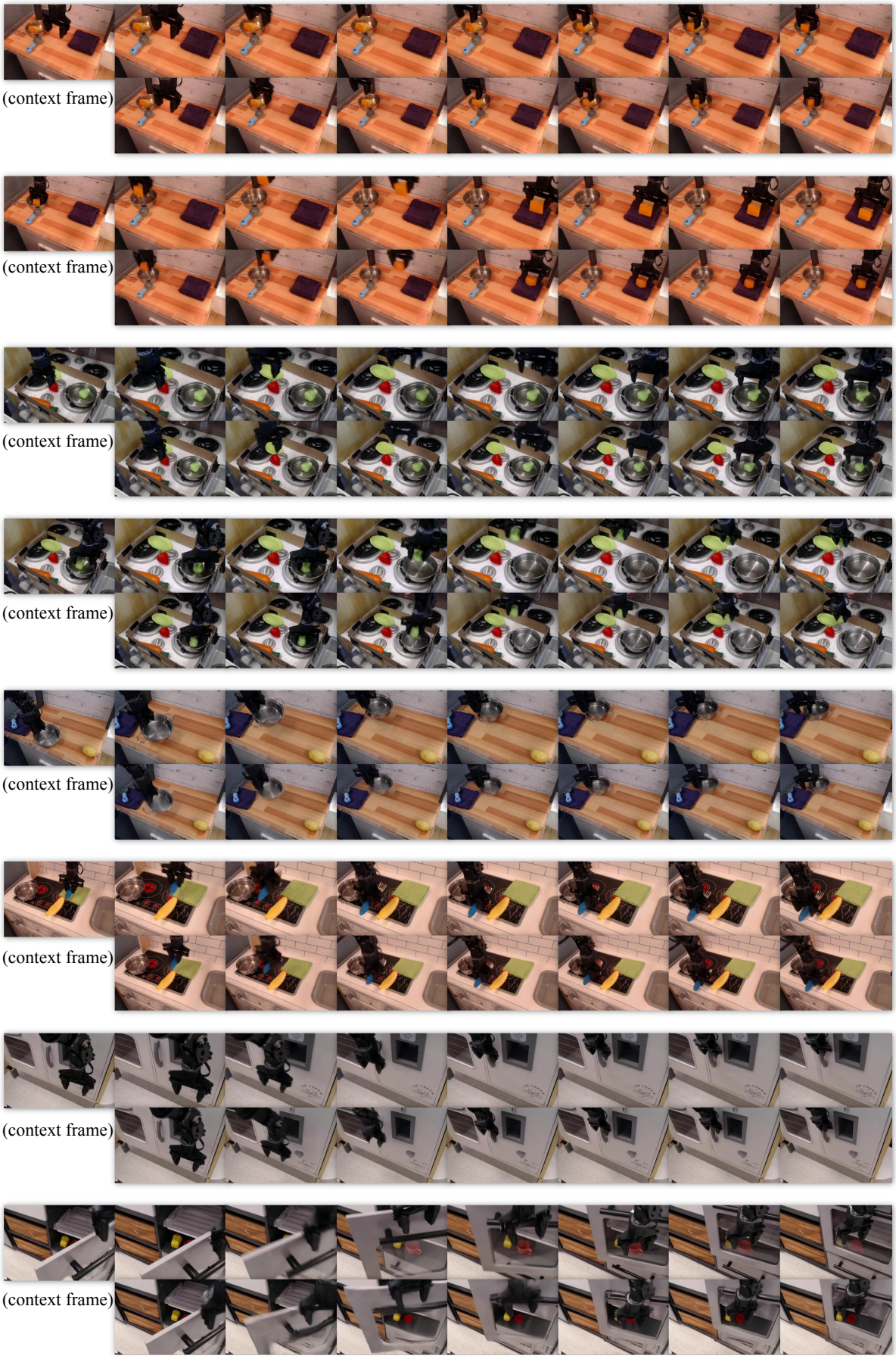


Figure 26. Additional Qualitative Results of ORV #1.

Figure 27. Additional Qualitative Results of ORV #2.

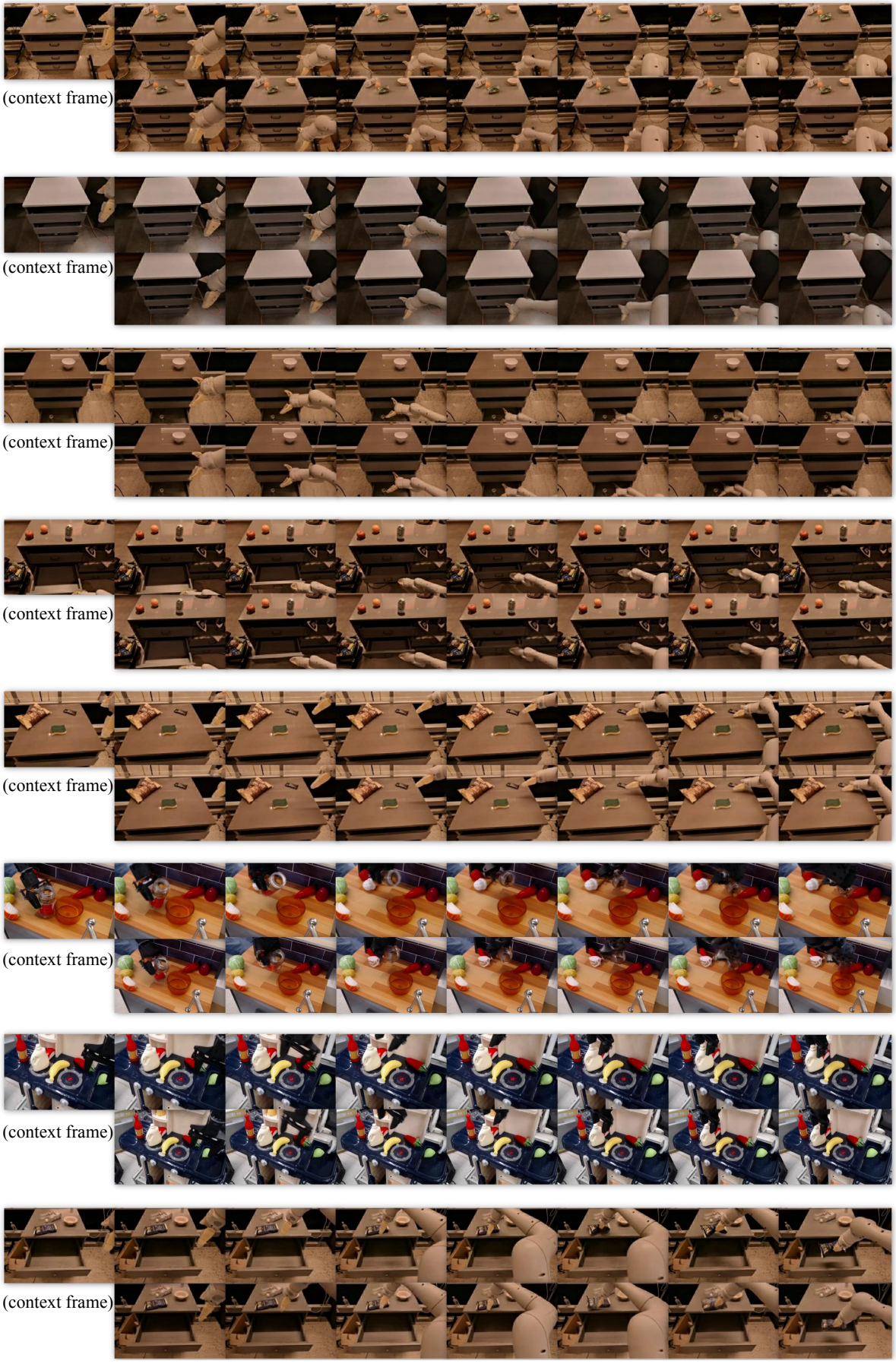


Figure 28. Additional Qualitative Results of ORV #3.