Learning 4D Embodied World Models

Supplementary Material

A. Implementation Details

A.1. Data Annotation Details

For data generation, we use the Marigold [4] DDIM Scheduler, performing 100 denoising steps in total. During the last 30 steps, optical flow guide inference was incorporated. The threshold for background mask computation was set to 0.5, and the gradient descent weight was configured as 2000. The pseudocode for the guidance stage is as follows:

Algorithm 1 Denoising With Optical Flow Consistency

Parameters: w ▷ Gradient descent weight
Inputs:
- $\mathbf{z}_t^i$: Noisy depth/normal latent variable at timestep t
- $\mathbf{v}^i$: Latent encoding of the video frame $\mathcal{V}^i$
- $\mathcal{M}^i, \mathcal{M}^{i-1}$: Static region masks for the current and previous frames
- $\mathcal{Z}^{i-1}$: Depth/normal annotation from the previous frame
- t: Current timestep in the denoising process

Outputs: $\mathbf{z}_{t-1}^i$: Noisy images at timestep $t-1$
function DENOISINGONESTEP($\mathbf{z_t^i}$)
  $\mathbf{z}_t^i$: require grads $\leftarrow$ True
  $\mathbf{z} = \epsilon_\theta(\mathbf{v}^i, \mathbf{z}_t^i, t)$ ▷ First denoise
  $\mathcal{L} = ||\text{Decoder}(z) \circ (\mathcal{M}^i \cap \mathcal{M}^{i-1}) - \mathcal{Z}^{i-1} \circ (\mathcal{M}^i \cap \mathcal{M}^{i-1})||^2$ ▷ By Eq.2
  $\mathcal{L}$.backward() ▷ To get the value of $\nabla_{\mathbf{z}_t^i}\mathcal{L}$
  $\mathbf{z}_t^i \leftarrow \mathbf{z}_t^i - w\nabla_{\mathbf{z}_t^i}\mathcal{L}$
  $\mathbf{z}_{t-1}^i = \epsilon_\theta(\mathbf{v}^i, \mathbf{z}_t^i, t)$ ▷ Second denoise
end function

Figure 1 illustrates the denoising process within the diffusion model, which is repeated at each timestep. Each step consists of three stages: **First Denoise**, where the noisy latent variable is passed through the Marigold for an initial forward; **Loss Backward**, where the loss is computed using static region masks and depth from the previous frame, followed by gradient-based updates to refine the latent variable; and **Second Denoise**, where the adjusted latent variable is further processed to produce a refined output. This iterative process ensures consistency across timesteps while improving depth/normal estimation.

A.2. Video Diffusion Model Details

Our RGB-DN video diffusion model's detailed training protocol is shown in Figure 2. For the RGB-DN video latent, we add noise to all frames except the first one, condition it on text instructions, and predict the added noise. We trained our

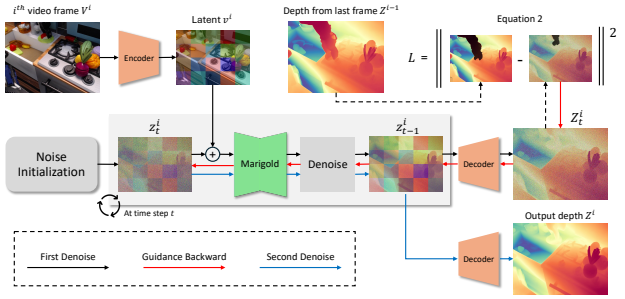

Figure 1. **Data generation pipeline**, comprising first denoise, guidance backward, and second denoise stages to produce refined depth/normal estimations.

video diffusion model using the STDiT3-XL/2 architecture, fine-tuned on OpenSora-v1.2 [10], employing 6 × 8 V100 GPUs. The model processes videos with 16 frames, using gradient checkpointing to optimize memory usage. We set a batch size of 2 for acceleration, used bf16 precision, and applied the ZeRO2 [7] optimization plugin. Additionally, we leveraged a T5 text encoder [6] for conditioning. For sampling, we use the RFlow scheduler with logit-normal sampling across 30 steps and set a classifier-free guidance scale of 7.0.

The training spanned 40,000 iterations with an initial learning rate 1e-4, 1.0 gradient clipping, and a 1,000-step warmup. The optimizer incorporated Adam with epsilon set to 1e-15, and an exponential moving average (EMA [5]) decay of 0.99 was used to stabilize training.

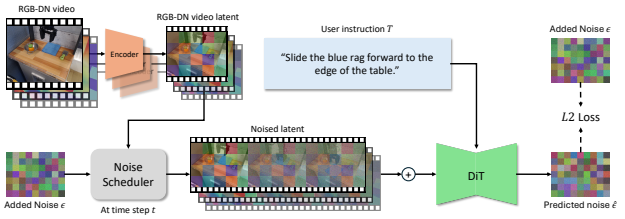

Figure 2. Training Pipeline of the RGB-DN Video Diffusion Model

A.3. 4D Meshes Generation

The parameters for the loss term in Eq.12 are set differently for the RT-1 [2] and RLBench [3] datasets, as shown in the table below:

In Figure 4, we present a visualization of the 3D robotic scene reconstruction optimized using our proposed method in the BridgeV2 [8] dataset. After estimating the depth and normal with the estimator, we refine the outputs to reconstruct the scene accurately. The figure includes untextured

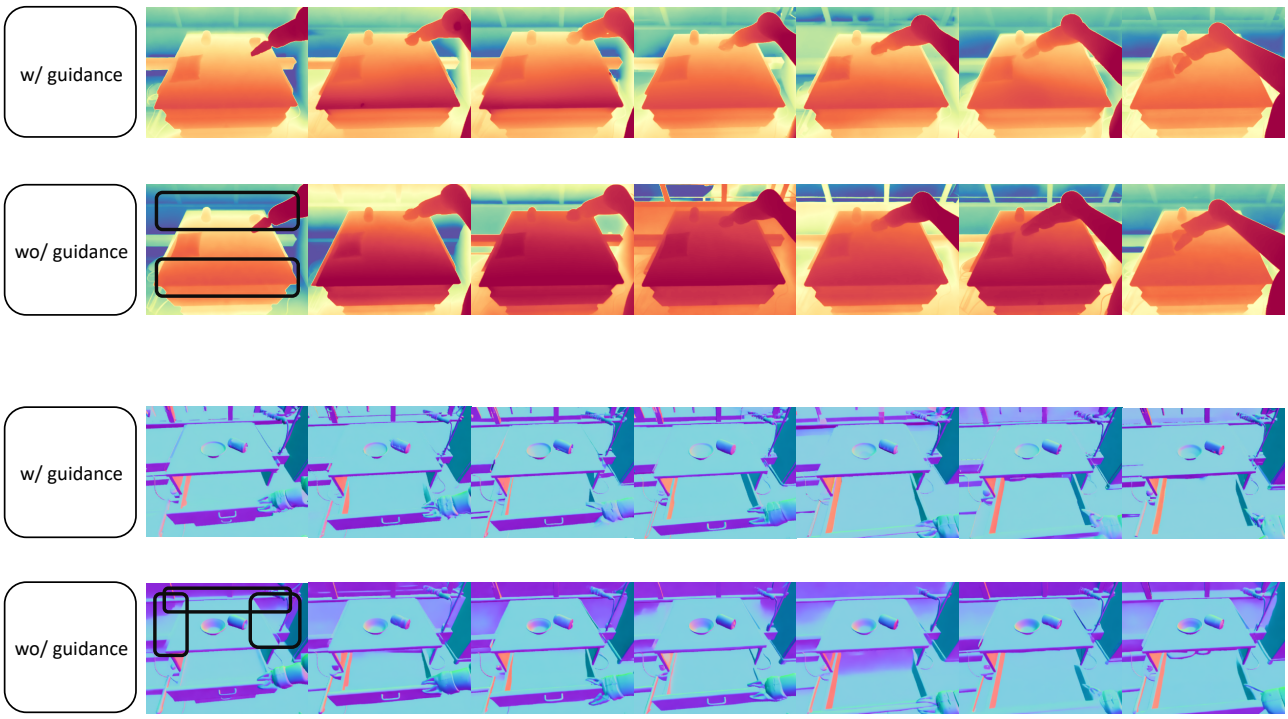

Figure 3. Qualitative Results of Data Generation Methods with and without Guidance, where the black boxes highlight areas of inconsistency.

Dataset	λ_d	λ_b	λ_{g1}	λ_{g2}
RT-1	20	200	20	20
RLBench	20	200	2	2

Table 1. Loss Term Parameters for RT-1 and RLBench Datasets

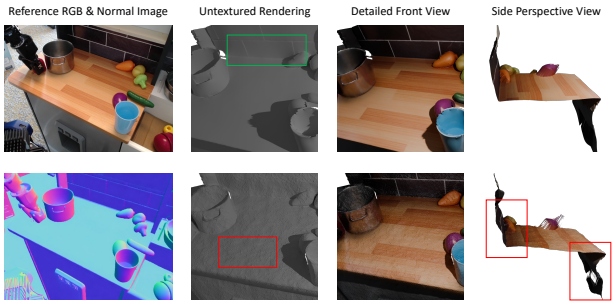

Figure 4. Visualization of the optimized 3D robotic scene reconstruction using our method. The untextured renderings show enhanced detail (green box) and improved surface smoothness (red box). The side perspective view highlights accurate shape and geometry optimization, including the perpendicular alignment of the wall and table (red boxes).

rendering and texture-rendered views, where the wall textures are significantly enhanced due to normal optimization. The side perspective view shows the improved shape and geometry reconstruction. Notably, the wall and table surfaces are well-aligned, appearing perpendicular to each other, further validating the effectiveness of our optimization process in capturing accurate spatial relationships.

A.4. Implementation Details for Robotics Planning

For the RLBench training, we adopted the same architecture and methods as our video diffusion model, with the primary difference being that we used 13 frames and fine-tuned the model on the RT-1 dataset.

For the action prediction stage, we first filter out the background and floor from the data, focusing only on the points of the table and the objects manipulated by the robotic arm, and then sample 1024 points from the filtered point cloud. In our inverse dynamic model, the PointNet extracts features from this point cloud, concatenated with the instruction's language embedding, and passed into a 4-layer MLP, finally outputting the 7DoF actions.

B. More Experiments

B.1. Data Annotation

In this section, we first compare our data generation method with 3D-VLA [9]. They use ZoeDepth [1] for depth map estimation and directly map them into 3D space. The com-

Pretrained	Channel	RGB FVD ↑	RGB SSIM ↑	RGB PSNR ↑	Depth AbsRel ↓	Depth δ_1 ↑	Depth δ_2 ↑	Normal Mean ↓	Normal Median ↓	Normal 11.25° ↑	Consistency Edge-sim ↑
✗	✗	**60.32**	56.44	16.02	37.65	30.33	71.35	128.57	139.50	0.0	50.52
✓	✗	55.04	56.32	16.36	34.39	34.59	67.34	128.77	139.61	0.0	51.62
✗	✓	54.11	65.90	19.28	18.40	65.02	91.20	12.94	7.58	25.02	53.39
✓	✓	56.82	**66.32**	**19.64**	**15.35**	**77.54**	**93.62**	**11.06**	**5.93**	**43.59**	**57.26**

Table 2. Ablation study of video prediction task on RLBench simulation. "Pretrained" ✓ indicates fine-tuning models pre-trained on the RT-1 dataset. "Channel" ✓ indicates concatenating RGB, depth, and normal maps along the channel dimension, while × refers to concatenation along the width.

parison results, shown in Figure 5, evaluate the quality of point cloud generation for both methods, with cubes replacing vertices for rendering. Our generated data demonstrates higher realism, while 3D-VLA exhibits noticeable shape distortion.

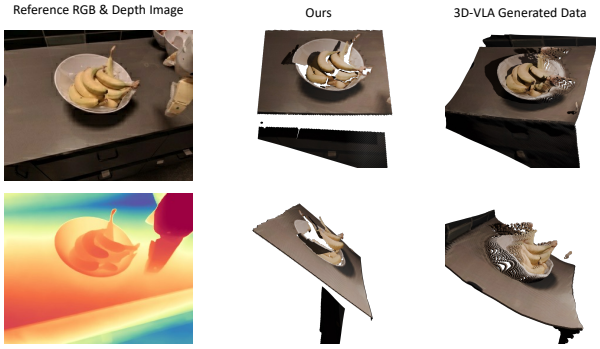

Figure 5. Comparison of point cloud generation quality between our method and 3D-VLA

We also show an ablation study comparing data generation methods with and without guidance. As shown in Figure 3, we focus on the qualitative results to evaluate the impact of guidance. The black boxes in the figure highlight areas of inconsistency, demonstrating that the Guided Marigold method achieves superior frame-to-frame coherence compared to the unguided Marigold-LCM. This emphasizes the importance of guidance in producing consistent dynamic scenes.

B.2. RGB-DN Prediction

We conducted an ablation study of the model framework using the RLBench simulation environment, as it provides ground truth depth values, making the evaluation more convincing. Since RLBench does not natively offer normal map outputs, we employed guided Marigold to generate normal annotations. The metrics used in this table are consistent with those described in the main text, ensuring a direct comparison with the results reported therein.

As shown in Table 2, concatenating RGB, depth, and normal maps along the width dimension performs poorly, particularly on normal map-related metrics, indicating that this approach is ineffective for normal map processing. In contrast, concatenating these inputs along the channel dimension significantly improves performance across all tasks. Furthermore, fine-tuning pre-trained models on the RT-1 dataset further enhances results. Combining pre-trained models and channel-wise concatenation yields the best overall performance.

B.3. More Qualitative Results

Figure 7 showcases qualitative results from the RT-1 dataset, illustrating our approach's capabilities in generating multi-modal visual representations. Our video diffusion model directly produces the RGB, depth, and normal maps, while the meshes are rendered from the reconstructed outputs. These results highlight the robustness of our method across different visual modalities. *Additional video results* can be found in the supplementary materials folder for further analysis and evaluation.

B.4. Action Planning

One potential application of our generated mesh is to extract action trajectories directly. As illustrated in Figure 6, we track the robotic arm in the video to capture its motion path. This trajectory is subsequently lifted into 3D space, enabling the reconstruction of the robot arm's action trajectory. The red line in the visualization represents the captured action trajectory.

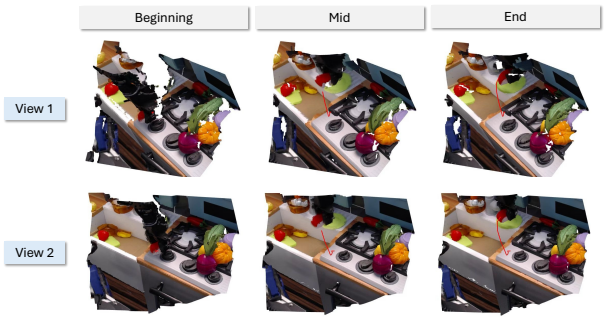

Figure 6. Tracking and visualization of robotic arm action trajectories on the Bridge dataset

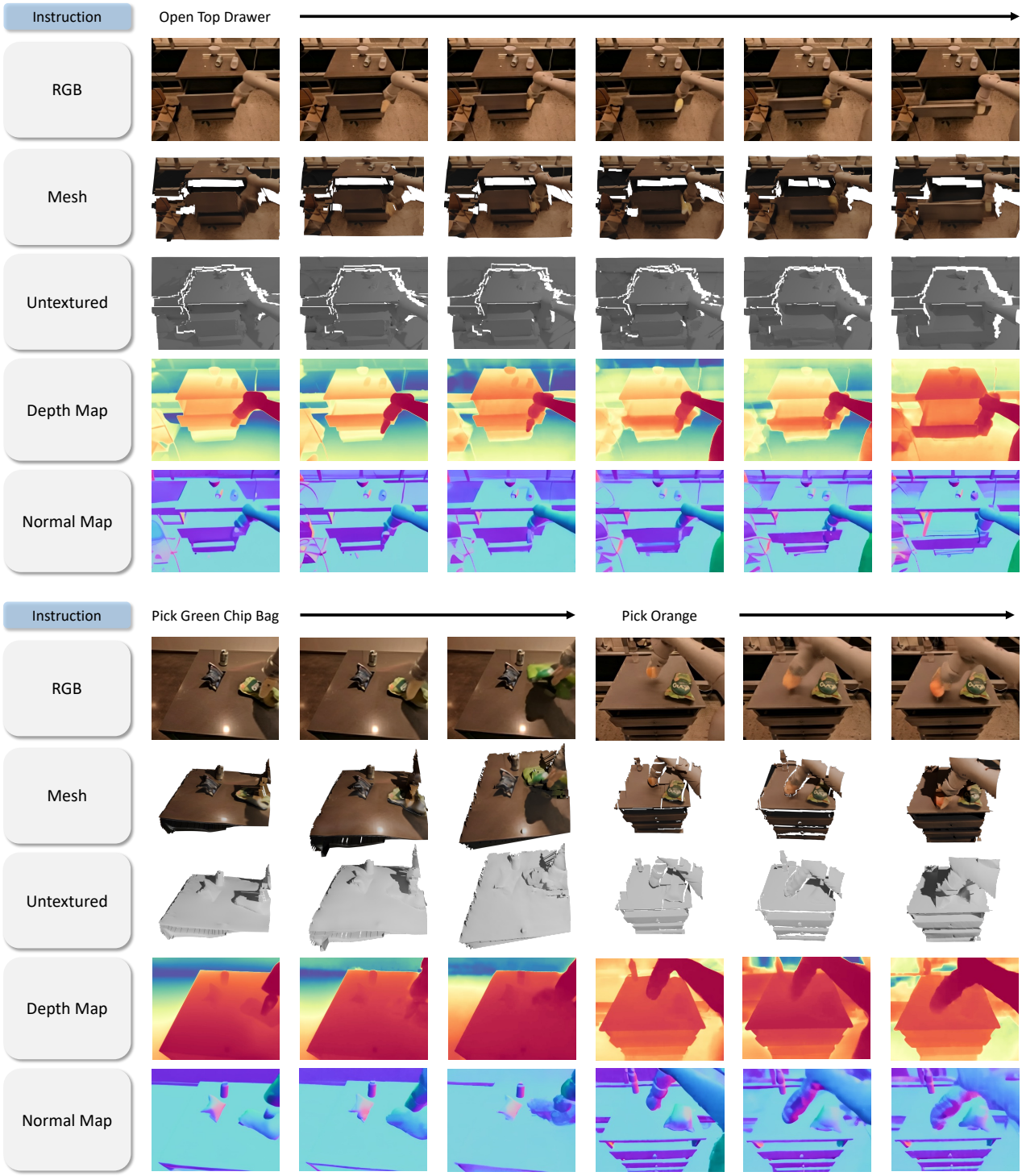

Figure 7. More qualitative results

References

[1] Shariq Farooq Bhat, Reiner Birkl, Diana Wofk, Peter Wonka, and Matthias Müller. Zoedepth: Zero-shot transfer by combining relative and metric depth. *arXiv preprint arXiv:2302.12288*, 2023. 2

[2] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022. 1

[3] Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J. Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 2020. 1

[4] Bingxin Ke, Anton Obukhov, Shengyu Huang, Nando Metzger, Rodrigo Caye Daudt, and Konrad Schindler. Repurposing diffusion-based image generators for monocular depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9492–9502, 2024. 1

[5] Frank Klinker. Exponential moving average versus moving exponential average. *Mathematische Semesterberichte*, 58: 97–107, 2011. 1

[6] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020. 1

[7] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE, 2020. 1

[8] Homer Walke, Kevin Black, Abraham Lee, Moo Jin Kim, Max Du, Chongyi Zheng, Tony Zhao, Philippe Hansen-Estruch, Quan Vuong, Andre He, Vivek Myers, Kuan Fang, Chelsea Finn, and Sergey Levine. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning (CoRL)*, 2023. 1

[9] Haoyu Zhen, Xiaowen Qiu, Peihao Chen, Jincheng Yang, Xin Yan, Yilun Du, Yining Hong, and Chuang Gan. 3d-vla: A 3d vision-language-action generative world model. *arXiv preprint arXiv:2403.09631*, 2024. 2

[10] Zangwei Zheng, Xiangyu Peng, Tianji Yang, Chenhui Shen, Shenggui Li, Hongxin Liu, Yukun Zhou, Tianyi Li, and Yang You. Open-sora: Democratizing efficient video production for all, 2024. 1