A. Additional Qualitative Results

In this section, we present additional qualitative video results on the following: 1) Short Trajectories: We compare IRASim with baseline methods using short trajectories from RT-1, Bridge, and Language-Table. We also provide additional qualitative results of IRASim on RoboNet; 2) Long Trajectories: We compare IRASim with baseline methods in the long trajectories setting; 3) Scaling: We compare different sizes of IRASim; 4) Robustness to Physically Implausible Trajectories: We show that IRASim can handle physically implausible trajectories.

A.1. Video Generation of Short Trajectories

Qualitative results are illustrated in Fig. 2 and Fig. 5. Fig. 2 demonstrate that IRASim-Frame-Ada surpasses other methods in aligning frames with actions and modeling the interaction between robots and objects. For RoboNet dataset, we follow Wu et al. [16] and use two frames as context for prediction. Fig. 5 illustrates that IRASim is capable of simulating the manipulation of flexible objects, such as dragging clothes.

In terms of the number of context frames, we conduct an additional experiment on Bridge dataset and used 2 frames as context. The performance change is minor: the PNSR of using 1 context frame and 2 context frames are both 25. We hypothesize that the input trajectory itself contains sufficient information about velocity. Thus, including more context frames does not bring about significant improvement.

A.2. Video Generation of Long Trajectories

Qualitative results are illustrated in Fig. 3. IRASim-Frame-Ada generates consistent and long-horizon videos, accurately simulating the entire trajectory. Additionally, IRASim-Frame-Ada maintains its superior performance in frame-action alignment and robot-object interaction as observed in the short trajectory setting.

A.3. Scaling

Qualitative results are shown in Fig. 4. IRASim-Frame-Ada consistently improves the quality of the generated video in terms of reality and accuracy with the increase of model size.

A.4. Robustness to Physically Implausible Trajectories

We perform experiments on rolling out a physically implausible trajectory. In particular, we input a trajectory that commands the robot to move downward even after it touches the table. Physically, the robot cannot penetrate the table and thus will remain on the table even if the input control commands it to move down. We input this trajectory to IRASim to evaluate its performance in handling physically implausible trajectories. As shown in Fig. 6, IRASim can generate physically accurate videos where the robot stays on the table.

B. Datasets

Dataset Statistics. We provide details on the four publicly available robot manipulation datasets: RT-1 [2], Bridge [15], Language-Table [11] and RoboNet [4]. A summary of the dataset statistics is presented in Table 2. For RT-1, Bridge and Language-Table, each training sample consists of a 4-second video clip containing 16 frames, extracted from an episode with a continuous sliding window. For testing and validation, frames are sampled at 16-frame intervals to reduce the number of evaluation videos and, consequently, lower evaluation costs. The original resolution for RT-1 is 256×320, for Bridge it is 480×640, and for Language-Table it is 360×640. To ensure efficient training, we resize the Bridge videos to a resolution of 256×320 and the Language-Table videos to 288×512. For RoboNet, we follow Wu et al. [16] and use 2 frames as context to predict the next 10 frames at a resolution of 256×256. Note that the mentioned "our own dataset" in Sec. 4.4 is similar in size to RT-1, and the action space is the same.

Action Space. Different datasets have different action spaces. In RT-1 and Bridge, a robot arm with a gripper moves in the 3D space to perform manipulation which interacts with objects in the scene. The action spaces of RT-1 and Bridge consist of 1) 6-DoF arm actions in 3D space, $T \in SE(3)$, and 2) continuous gripper actions, $g \in [0, 1]$. In Language-Table, a robot arm moves in a 2D plane to move blocks with a cylindrical end-effector. The action space of Language-Table is 2-DoF translation in 2D space, $p \in R^2$. We convert the arm action of all datasets to relative delta actions. Specifically, we specify the action of RT-1 and Bridge with a 7-dim vector, i.e., $a = [\Delta x, \Delta y, \Delta z, \Delta\alpha, \Delta\beta, \Delta\gamma, g]$ where Δx, Δy, and Δz are the delta XYZ position; $\Delta\alpha$, $\Delta\beta$, and $\Delta\gamma$ are the delta Euler angles; g indicates the gripper joint-angle position in the next step. For Language-Table, we specify the action with a 2-dim vector, i.e., $a = [\Delta x, \Delta y]$ which indicates the delta position in the xy-plane. RoboNet is a large-scale robot manipulation dataset featuring 7 robot platforms with varying action spaces (2, 4, or 5 dimensions). Following Dasari et al. [4], to unify the data, a 5-dimensional vector is used to represent a universal action space, padding zeros for missing dimensions. This vector represents delta XYZ position, delta yaw angle, and gripper joint-angle value: $a = [\Delta x, \Delta y, \Delta z, \Delta\gamma, g]$. For instance, if a robot doesn't control the z-axis, Δz is set to 0.

C. IRASim Model Details

In this section, we introduce more details about two types of trajectory condition methods in Sec. 3.3: *Video-Level Condition* and *Frame-Level Condition*.

Table 1. Quantitative Results of Long-Trajectory Video Generation.

	RT-1		Bridge		Language-Table	
	Latent L2 ↓	PSNR ↑	Latent L2 ↓	PSNR ↑	Latent L2 ↓	PSNR ↑
LVDM	0.2567	23.573	0.2534	21.792	<u>0.1776</u>	<u>26.215</u>
IRASim-Video-Ada	<u>0.2519</u>	<u>23.984</u>	<u>0.2385</u>	<u>22.868</u>	0.2112	22.551
IRASim-Frame-Ada	**0.2408**	**24.615**	**0.2306**	**23.260**	**0.1730**	**26.773**

Table 2. Dataset Statistics. An "episode" is a single trial where the robot completes a task. A "sample" is a clip from an episode. "-" indicates that we follow previous work and do not use a validation set.

Datasets	**RT1**		**Bridge**		**Language-Table**		**RoboNet**	
Data Split	**Episode**	**Sample**	**Episode**	**Sample**	**Episode**	**Sample**	**Episode**	**Sample**
Train	82,069	2,314,893	25,460	482,701	170,256	1,483,133	162,161	2,540,500
Validation	2,167	4,810	1,737	2,905	4,446	5,119	-	-
Test	2,167	4,799	1,738	2,946	4,562	5,243	256	407

C.1. Video-Level Conditioning

In video-level condition (Fig. 2(b)), we first obtain the conditioning embedding $\mathbf{c}_{ST}$ by adding the diffusion timestep embedding to the trajectory embedding. We then use $\mathbf{c}_{ST}$ to regress the scale parameters γ and α, as well as the shift parameters β. Specifically, the computation of the spatial block is as follows:

$$\mathbf{x} = \mathbf{x} + (1 + \alpha_1) \times \text{MHA}(\gamma_1 \times \text{LayerNorm}(\mathbf{x}) + \beta_1) \tag{1}$$

$$\mathbf{x} = \mathbf{x} + (1 + \alpha_2) \times \text{FFN}(\gamma_2 \times \text{LayerNorm}(\mathbf{x}) + \beta_2) \tag{2}$$

where $\mathbf{x}$, with a shape of (N, P, D), denotes the token embeddings. $\mathbf{x}$ is reshaped as (P, N, D) before entering the temporal block. The computation of the temporal block is:

$$\mathbf{x} = \mathbf{x} + (1 + \alpha_3) \times \text{MHA}(\gamma_3 \times \text{LayerNorm}(\mathbf{x}) + \beta_3) \tag{3}$$

$$\mathbf{x} = \mathbf{x} + (1 + \alpha_4) \times \text{FFN}(\gamma_4 \times \text{LayerNorm}(\mathbf{x}) + \beta_4) \tag{4}$$

Note that layer normalization is performed before scaling and shifting.

C.2. Frame-Level Condition

In frame-level condition (Fig. 2(c)), spatial attention blocks and temporal attention blocks are conditioned differently. The derivation of the conditioning embedding for temporal attention blocks $\mathbf{c}_T$ is the same as in video-level condition, where we add the diffusion timestep embedding to the trajectory embedding. Different frames are conditioned differently in spatial attention blocks. We denote the conditioning embedding of spatial attention blocks for the i-th frame as $\mathbf{c}_S^i$. To derive $\mathbf{c}_S^i$, the i-th action in the trajectory is first encoded to an embedding through a linear layer. The diffusion timestep embedding is then added to the encoded embedding to obtain $\mathbf{c}_S^i$. We use $\mathbf{c}_S^1, \ldots, \mathbf{c}_S^N$ and $\mathbf{c}_T$ to regress the corresponding scale parameters γ and α, as well as the shift parameters β. While the computation of the temporal blocks is the same as the video-level condition (Eq. 3 and 4), the computation of spatial blocks is different:

$$\mathbf{x}^i = \mathbf{x}^i + (1 + \alpha_1^i) \times \text{MHA}(\gamma_1^i \times \text{LayerNorm}(\mathbf{x}^i + \beta_1^i)), \tag{5}$$

$$\mathbf{x}^i = \mathbf{x}^i + (1 + \alpha_2^i) \times \text{FFN}(\gamma_2^i \times \text{LayerNorm}(\mathbf{x}^i + \beta_2^i)). \tag{6}$$

where $\alpha_1^i, \gamma_1^i, \beta_1^i, \alpha_2^i, \gamma_2^i, \beta_2^i$ denote the scale and shift parameters for the i-th frame. They are regressed from $\mathbf{c}_S^i$.

D. Baselines Details

In this section, we detail the baseline implementation. For VDM [8], we leverage the implementation provided in [1], which utilizes a 3D U-Net architecture for controllable video generation. We use only the model component from this code and keep the training setting consistent with IRASim. LVDM [7] employs the same model architecture as VDM. It performs diffusion in the latent space while VDM performs diffusion in the pixel space. We use an MLP to encode the trajectory into an embedding. It is then concatenated with the embedding of the diffusion timestep to form the conditioning embedding. This is similar to the original methods in the paper where the text embedding is concatenated with the diffusion timestep embedding to form the conditioning embedding. The initial frame condition method of VDM

[1] https://github.com/lucidrains/video-diffusion-pytorch

and LVDM is the same as IRASim as described in Sec. 3.3. LVDM and IRASim share the same VAE model and training setting. Given that the resolution of Language-Table [11] is up to 288×512, we resize the video to 144×256 in the training of VDM to make the computational cost affordable. During evaluation, we resize the generated video back to 288×512 for comparison with other methods. For RT-1 and Bridge, the training of VDM is performed at a resolution of 256×320. The training hyperparameters for VDM and LVDM are shown in Tab. 3 and 4. More training hyperparameters that share with IRASim can be found in Tab. 5.

We also briefly introduce the baseline details of iVideoGPT [16] and MaskViT [6]. Both of them use VQ-GAN [5] as the image tokenizer and require additional finetuning it on RoboNet, while IRASim employs the VAE encoder from SDXL [14] without the need for extra finetuning. Their parameter sizes are 436M and 228M, respectively. Moreover, iVideoGPT undergoes extensive pre-training on OpenX-Embodiment [12], whereas IRASim achieves better video prediction performance with training only on RoboNet.

E. Training Details

For all models, we use AdamW [9] for training. We use a constant learning rate of 1e-4 and train for 300k steps with a batch size of 64. The gradient clipping is set to 0.1. We found the training of IRASim very stable – no loss spikes were observed even without gradient clipping. However, loss spikes often occur in LVDM and VDM when gradient clipping is not used. Following Peebles and Xie [13], we utilize the Exponential Moving Average (EMA) technique with a decay of 0.9999. All other hyperparameters are set the same as Peebles and Xie [13]. Tab. 5 lists further hyperparameters. All models are trained from scratch. We utilize PNDM [10] with 50 sampling steps for efficient video generation during evaluation. IRASim generates a 16-frame video with a duration of approximately 4 seconds, requiring only 30 seconds on an A100 GPU using 8GB of memory. Although there is still significant room for latency improvement, our method features high throughput and is memory-friendly during inference.

For scaling results in Fig. 4, the configurations of four different sizes of IRASim are shown in Tab. 6. We study the scale performance of IRASim-Frame-Ada since it performs best.

The information about computing resources for training our IRASim is provided in Tab. 7.

F. Evaluation Details

We introduce the evaluation details in this section.

Evaluation Metrics. Latent L2 loss and PSNR measure the L2 distance between the predicted video and the ground-truth video in the latent space and pixel space, respectively. SSIM evaluates the similarity between videos in terms of image brightness, contrast, and structure. FID and FVD assess video quality by analyzing the similarity of video feature distributions.

Evaluation Setup. We evaluate the video quality generated by IRASim and the baselines under two settings: short trajectories and long trajectories. In the short trajectory setting, the input consists of one initial frame and a short trajectory containing 15 actions, resulting in the generation of 15 subsequent frames. These short trajectories are sampled from episodes using a sliding window with an interval of 16. In the long trajectory setting, the input comprises one initial frame and a complete long trajectory, with the output being the generated subsequent frames. The average lengths of the long trajectories are 42.5, 33.4, and 23.7 frames for RT-1, Bridge, and Language-Table, respectively. These lengths also represent the average number of frames for the generated long videos, which are produced in an autoregressive manner, as detailed in Sec. 4.1. The statistics of the generated short and long videos used for evaluation are presented in Tab. 2.

Metric Calculation. In all metric calculations, we ignore the initial frame and only evaluate the quality of the generated frames. For PSNR and SSIM, we refer to skimage [2] for calculation. For FID and FVD, we refer to [3] and [4] for calculation, splitting the generated videos into frames and using their codebases to compute the FID and FVD values. However, we do not calculate FID and FVD metrics for long videos because we find that these metrics do not reflect human preferences well, even in the short trajectory setting. This could be because FID and FVD essentially calculate the similarity between the distributions of two datasets, whereas the *trajectory-to-video* task is a reconstruction task, making reconstruction loss a more suitable evaluation metric.

G. Real-Robot Model-based Planning Details

In this section, we detail the real-robot model-based planning experiment. The experiment demonstrates that IRASim can effectively plan trajectories to finish manipulation tasks by generating the outcomes of executing different candidate trajectories.

Experiment Setup. We follow Babaeizadeh et al. [1] to set up this experiment. We implement a model-based policy to show the usefulness of IRASim. Our policy consists of a sampling-based planner, a cost function, and IRASim as

[2] https://scikit-image.org/docs/stable/api/skimage.metrics.html
[3] https://github.com/mseitzer/pytorch-fid
[4] https://github.com/universome/stylegan-v

Table 3. Hyperparameters for VDM.

Hyperparameter	Value
Base channels	64
Channel multipliers	1,2,4,8
Num attention heads	8
Attention head dimension	32
Conditioning embedding dimension	768
Input channels	3
Parameters	40M

Table 4. Hyperparameters for LVDM.

Hyperparameter	Value
Base channels	288
Channel multipliers	1,2,4,8
Num attention heads	8
Attention head dimension	32
Conditioning embedding dimension	768
Input channels	3
Parameters	687M

Table 5. Hyperparameters for training IRASim.

Hyperparameter	Value
Layers	28
Hidden size	1152
Num attention heads	16
Patch size	2
Input channels	4
Dropout	0.1
Optimizer	AdamW($\beta = 0.9, \beta = 0.999$)
Learning rate	0.0001
Batch size	64
Gradient clip	0.1
Training steps	3000000
EMA	0.9999
Weight decay	0.0
Prediction target	ϵ
Parameters	679M

the dynamic function. We first train IRASim with our own real robot dataset. The input of our policy includes the initial image, the initial position of the end-effector, and a goal image to indicate the task. The output is a predicted trajectory. We use a simple sampling-based planner to generate candidate trajectories. The planner samples 50 individual points from a circle centered on the initial end-effector position and then generates a trajectory between the initial position and each sampled point, resulting in 50 different candidate trajectories. We input the initial image and each trajectory to IRASim to generate the video of executing each trajectory. We use a cost function to calculate the similarity between each predicted video and the goal image. We experiment with 2 cost functions: 1) mean squared error (MSE) and 2) cosine similarity of the feature extracted from ResNet50. We execute the top 5 trajectories with the lowest cost (i.e., the predicted video most similar to the goal image) in the real world and calculate the average success rate. The experiment is repeated three times for each task.

Results Qualitative results are shown in Fig. 7. Quantitative results are shown in Tab. 5. We compare our method with a baseline that randomly picks a trajectory from the 50 candidates. The results show that using IRASim significantly increases the success rate compared to the random baseline.

Discussion About Cost Function. We also explore how different cost functions impact the model's performance. We find that the MSE cost function is generally superior to the ResNet cost function. But the MSE cost function is not always perfect; sometimes it selects incorrect prediction videos, leading to task failure. This suggests that we need to explore better cost functions in future work, considering that the success rate is influenced by both the accuracy of video prediction and the accuracy of the cost function. A suboptimal cost function could affect the evaluation of the video prediction model, as also mentioned by iVideoGPT [16] and VLMPC [18].

Discussion About Sample Policy. Although we use a simple sampling-based planner as the sample policy in this experiment, we note that IRASim can be combined with any policy that has trajectory sampling capabilities (i.e., action chunk techniques [3, 17]). The performance and range of tasks that IRASim can handle could be further enhanced by adopting a more advanced policy [3, 17], which is capable of generating more precise and complex trajectories.

H. Human Preference Evaluation

Five participants took part in the human evaluation. For each participant, we randomly sampled 10 ground-truth video clips from the test set for each of the 3 datasets. And for each video clip, we juxtapose the predictions of IRASim-Frame-Ada with those of VDM, LVDM, and IRASim-Video-Ada (Fig. 7). Thus, a participant evaluated 90 pairs of video clips. Note that the orders of the juxtaposition are random for different clips. See the caption of Fig. 7 for more details. We compare the results of all evaluated video clips and calculate the win, tie, and loss rates. The screenshot of the GUI used

in the human evaluation is shown in Fig. 7. The full text of the instruction given to participants is as follows:

Evaluation Instructions

You are asked to choose the more realistic and accurate video from two generated videos (shown above). The ground-truth video is given as a reference (shown below). Please carefully examine the given videos. If you can find a significant difference between the two generated videos, you may choose which one is better immediately. If not, please replay the videos more times. If you are still not able to find differences, you may choose the "similar" option. Please do not guess. Your decision needs solid evidence.

Table 6. Model Sizes. We use IRASim as an abbreviation of IRASim-Frame-Ada for brevity.

Model	Layers	Hidden size	Num attention heads	Parameters
IRASim-S	12	384	6	33M
IRASim-B	12	768	12	132M
IRASim-L	24	1024	16	461M
IRASim-XL	28	1152	16	679M

Table 7. Compution resources for training IRASim.

Dataset	Concurrent GPUs	GPU Hours	GPU type
RT-1	32	2381	A800 (40 GB)
Bridge	32	2371	A800 (40 GB)
Lanaguge-Table	32	2369	A100 (80 GB)

Figure 1. **Qualitative Results of Long-Trajectory Video Generation.** We show video generation of IRASim with long trajectories on the test set of RT-1, Bridge, and Language-Table. Ground truths are in blue boxes. Predictions are in orange boxes. The given historical frames are in green boxes. See the project page for full videos.

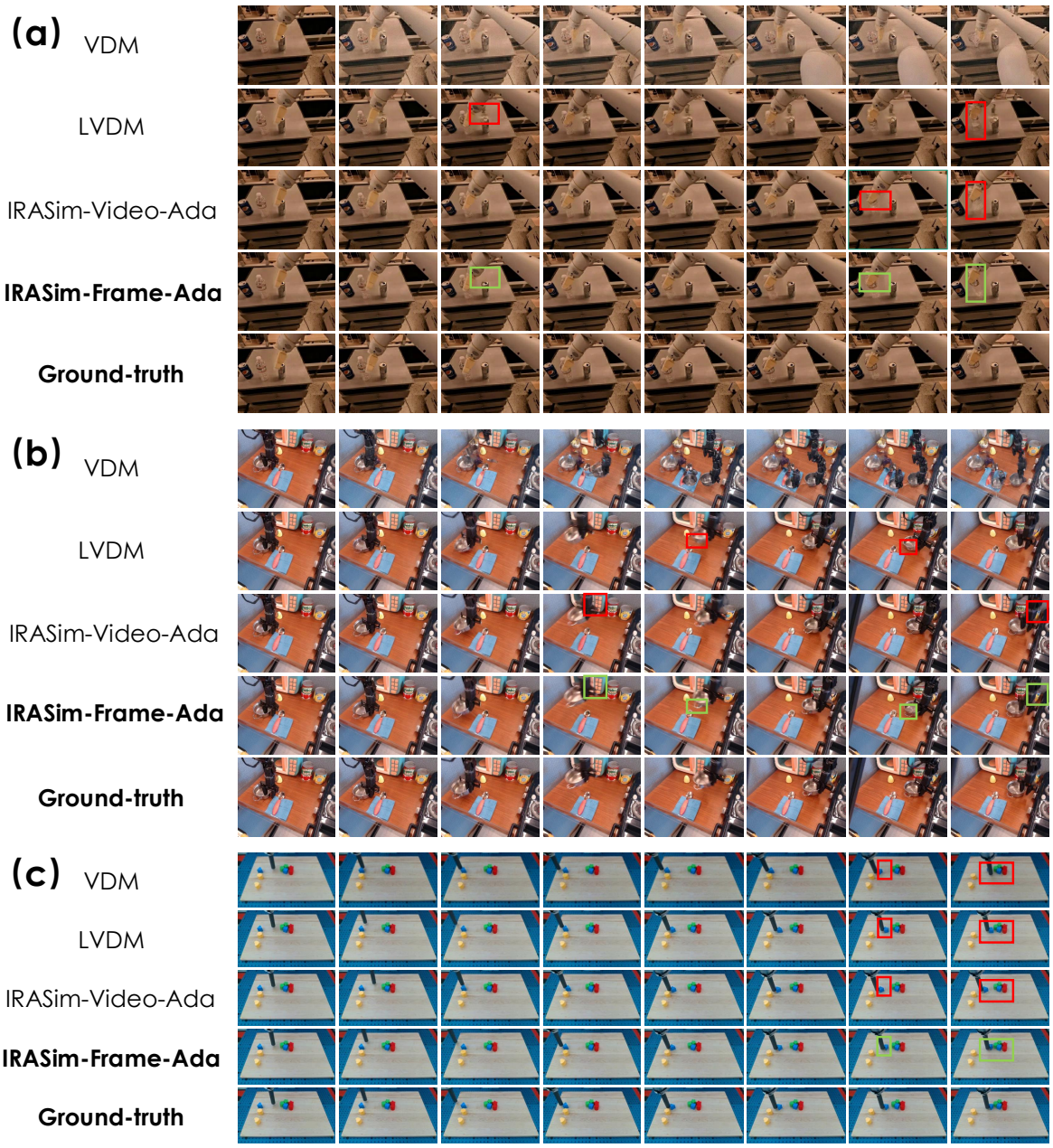

Figure 2. **Additional Qualitative Results on Video Generation of Short Trajectories.** We compare the results of different methods on (a) RT-1, (b) Bridge, and (c) Language-Table. Differences between IRASim-Frame-Ada and other methods are highlighted in green and red boxes.

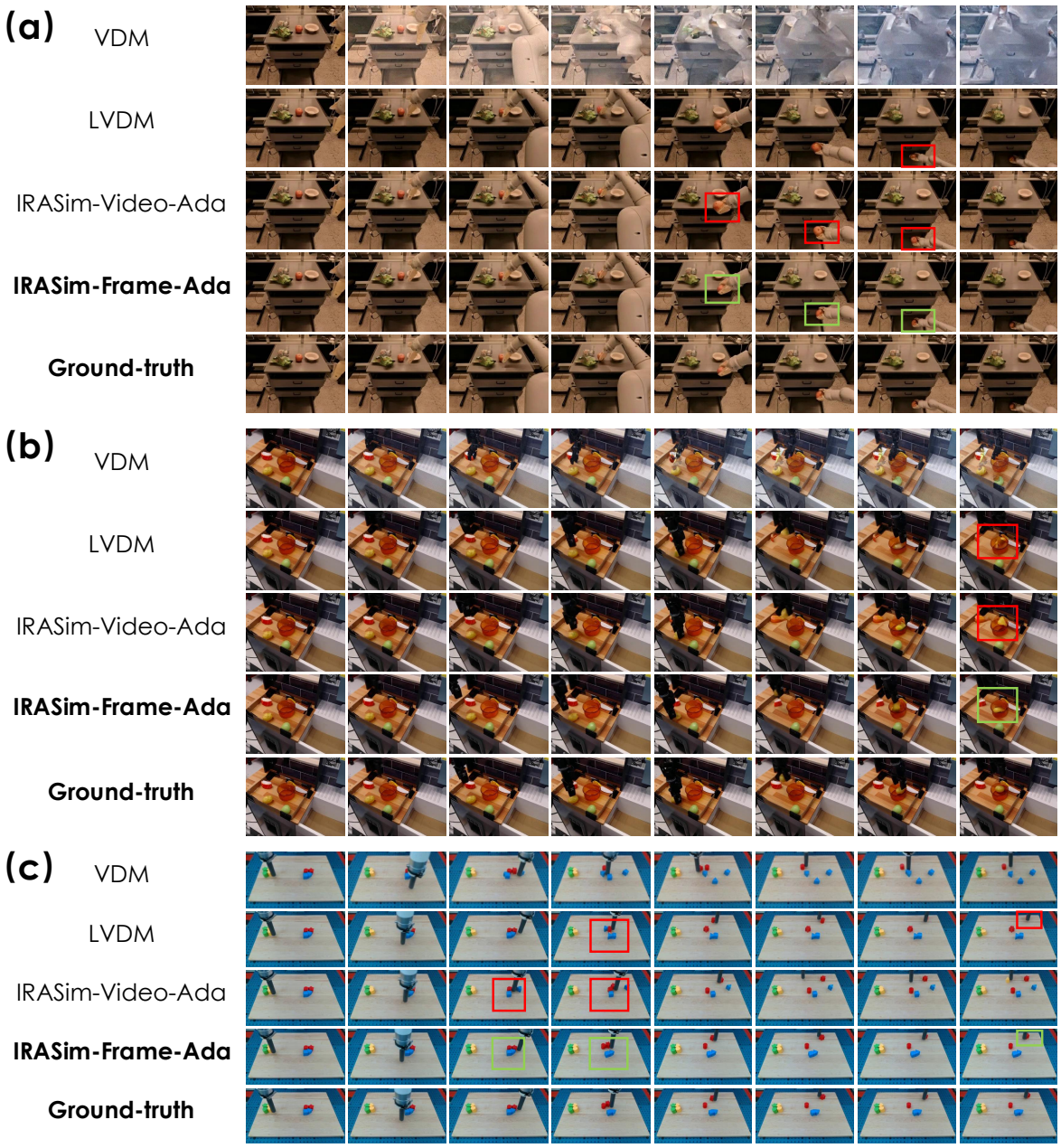

Figure 3. **Additional Qualitative Results on Video Generation of Long Trajectories.** We compare the results of different methods on (a) RT-1, (b) Bridge, and (c) Language-Table. Differences between IRASim-Frame-Ada and other methods are highlighted in green and red boxes. Note that the input trajectory is the entire trajectory of an episode.

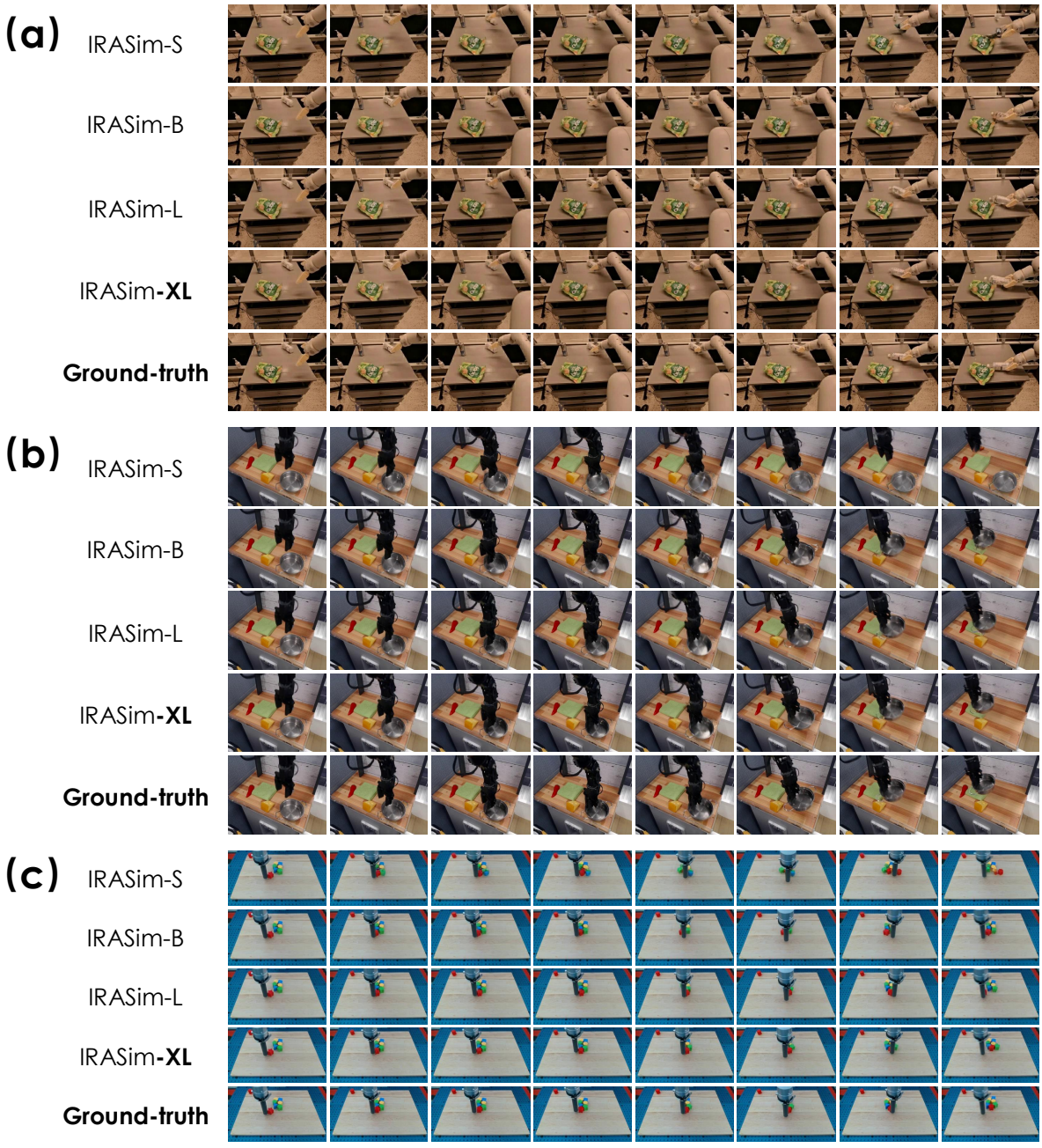

Figure 4. **Additional Qualitative Results on Scaling.** We compare the results of IRASim-Frame-Ada with different model sizes on (a) RT-1, (b) Bridge, and (c) Language-Table.

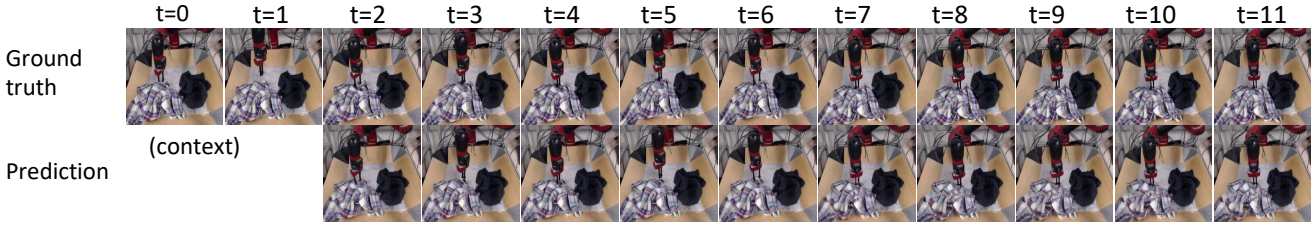

Figure 5. Quantitative results of IRASim-Frame-Ada on the RoboNet dataset. The robot is dragging the clothes, indicating that IRASim is capable of simulating the deformation of flexible objects.

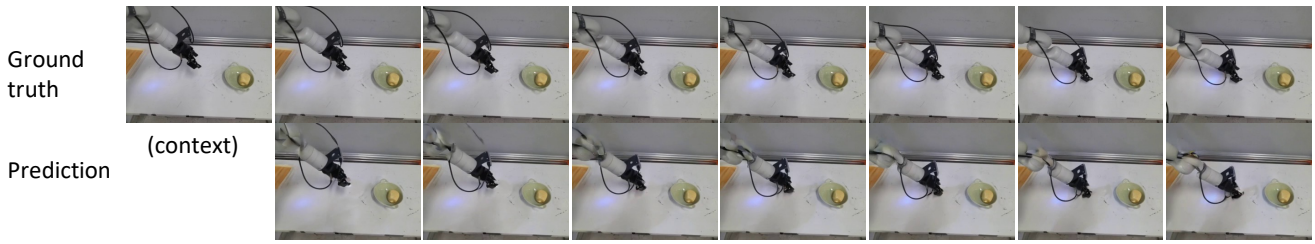

Figure 6. Quantitative results show that IRASim is robust to physically implausible trajectories. We control the robot to poke at the table and record the command trajectory, which is very dangerous as it could damage the robot. As a result, the robotic arm is blocked by the table. We find that executing the same trajectory in IRASim yields similar results, rather than the robotic arm passing through the table. This indicates that IRASim has a certain understanding of the physical laws of the real world.

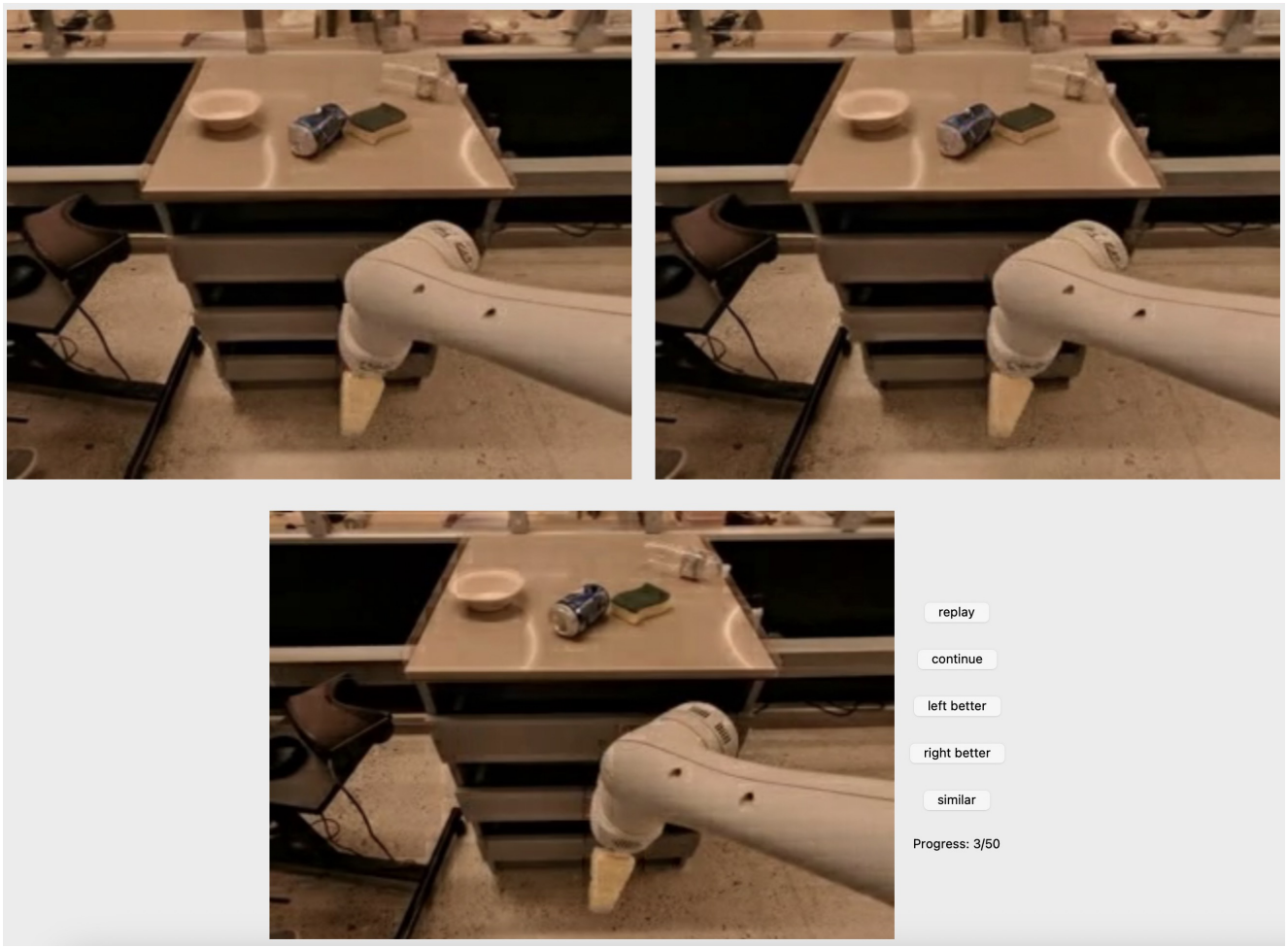

Figure 7. **Screenshot of the GUI in Human Preference Evaluation.** The two videos in the upper row are generated by IRASim-Frame-Ada and a comparing method, arranged in a **random** left-right order. The video in the lower row is the ground-truth video.

References

[1] Mohammad Babaeizadeh, Mohammad Taghi Saffar, Suraj Nair, Sergey Levine, Chelsea Finn, and Dumitru Erhan. Fitvid: Overfitting in pixel-level video prediction, 2021. 3

[2] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-1: Robotics transformer for real-world control at scale, 2023. 1

[3] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In Proceedings of Robotics: Science and Systems (RSS), 2023. 4

[4] Sudeep Dasari, Frederik Ebert, Stephen Tian, Suraj Nair, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, Sergey Levine, and Chelsea Finn. Robonet: Large-scale multi-robot learning. In Proceedings of the Conference on Robot Learning, pages 885–897. PMLR, 2020. 1

[5] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 12873–12883, 2021. 3

[6] Agrim Gupta, Stephen Tian, Yunzhi Zhang, Jiajun Wu, Roberto Martín-Martín, and Li Fei-Fei. Maskvit: Masked visual pre-training for video prediction. In ICLR, 2023. 3

[7] Yingqing He, Tianyu Yang, Yong Zhang, Ying Shan, and Qifeng Chen. Latent video diffusion models for high-fidelity long video generation, 2023. 2

[8] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. Advances in Neural Information Processing Systems, 35: 8633–8646, 2022. 2

[9] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015. 3

[10] Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. In International Conference on Learning Representations, 2022. 3

[11] Corey Lynch, Ayzaan Wahid, Jonathan Tompson, Tianli Ding, James Betker, Robert Baruch, Travis Armstrong, and Pete Florence. Interactive language: Talking to robots in real time. IEEE Robotics and Automation Letters, 2023. 1, 3

[12] OpenX-Embodiment. Open X-Embodiment: Robotic learning datasets and RT-X models. https://arxiv.org/abs/2310.08864, 2023. 3

[13] William Peebles and Saining Xie. Scalable diffusion models with transformers. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 4195–4205, 2023. 3

[14] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis, 2023. 3

[15] Homer Rich Walke, Kevin Black, Tony Z Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, et al. Bridgedata v2: A dataset for robot learning at scale. In Conference on Robot Learning, pages 1723–1736. PMLR, 2023. 1

[16] Jialong Wu, Shaofeng Yin, Ningya Feng, Xu He, Dong Li, Jianye Hao, and Mingsheng Long. ivideogpt: Interactive videogpts are scalable world models, 2024. 1, 3, 4

[17] Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware. In Proceedings of Robotics: Science and Systems, Daegu, Republic of Korea, 2023. 4

[18] Wentao Zhao, Jiaming Chen, Ziyu Meng, Donghui Mao, Ran Song, and Wei Zhang. Vlmpc: Vision-language model predictive control for robotic manipulation, 2024. 4