Choreographing a World of Dynamic Objects

– Supplementary Material –

A. Implementation Details

A.1. Pipeline Implementation Details

The 3D assets used in our experiments are downloaded from Sketchfab [1] and BlenderKit [3], and we construct the static scene snapshots in Blender [4]. Rendering of 3D-GS [8] for both mesh-based initialization and 4D optimization is performed using `gsplat` [16]. We adopt the Wan 2.2 (14B) image-to-video model [13] as our video generation model. All training is conducted at a resolution of 832×464 (the default for Wan 2.2), and deformation sequences of 41 frames are optimized.

Control points are initialized based on the center points of an occupancy grid. Specifically, for each object, we first compute a signed distance field (SDF) $\phi_i(\mathbf{x})$ from its given mesh. We then extract the set of voxel centers within the object as $\mathcal{I}_i = \{\mathbf{x} \mid \phi_i(\mathbf{x}) \leq 0, \mathbf{x} \in V_s\}$, where V_s denotes all voxel center points in a grid with voxel size s. Finally, we apply farthest point sampling followed by K-means clustering on $\mathcal{I}_i$ to determine the positions $\mathbf{p}_k$ of the control points. We further initialize the scale in each control point's covariance matrix $\mathbf{\Sigma}_k$ as the average distance to its three nearest neighboring control points, and set the initial rotation to the identity. For stable optimization, we keep $\mathbf{p}_k$ fixed and only optimize $\mathbf{\Sigma}_k$ during training. In the training of the deformations, we additionally introduce a split training schedule: at a iteration 100, we reinitialize all deformations after 30 to the deformation at 30, which further facilitates stable learning for later frames.

We use the log-linear learning rate schedule adopted in 3D-GS. The learning rate for the deformations stored in the Fenwick tree decays from 0.006 to 0.00006. The learning rate for the scales of the control points follows the same decay (from 0.006 to 0.00006), while the learning rate for rotations decays from 0.003 to 0.00003. The CFG [6] scale is linearly decayed from 25 to 12. The weight for the temporal regularization loss is decayed from 9.6 to 1.6, and the weight for the spatial regularization loss is decayed from 3000 to 300. The voxel size s used for extracting the uniformly distributed point cloud in temporal regularization and for initializing control points is automatically determined via binary search such that the number of voxel centers near the surface satisfies $|\mathcal{S}_i| \approx 7500$. Each asset is trained for 2,000 iterations with a batch size of 4, requiring approximately 17.7 hours on an NVIDIA H200 GPU.

A.2. Robot Manipulation Implementation Details

For the objects used to generate dense object flow, we directly scanned the real objects in the "pick banana" and "lower lamp" cases and fed the scans into our pipeline. For the other cases, due to challenges in accurately scanning the objects, we instead measured their length statistics and created digital cousins with matching dimensions in Blender before inputting them into our pipeline.

A.3. Baseline Implementation Details

For Animate3D [7] and AnimateAnyMesh [14], we merge all objects in the scene into a single mesh and directly input it into their pipelines. For MotionDreamer [12], we follow their setup and use Neural Jacobian Fields (NJF) [2] as the animation model, training a separate NJF for each object. For robust 4D reconstruction of videos sampled from TrajectoryCrafter [17], we use a coarse set of control points with a Fenwick-tree–based deformation sequence as the 4D representation. We additionally apply both temporal and spatial regularization losses during optimization.

B. Derivation of SDS for Rectified Flow Models

When sampling noise levels τ uniformly from $\mathcal{U}(0,1)$, the training loss of a Rectified Flow (RF) model [5, 10] is:

$$\mathcal{L}_{\text{RF}}(\theta; \mathbf{z}, \mathbf{y}) = \mathbb{E}_{\tau \sim \mathcal{U}(0,1),\, \epsilon} \left[w(\tau) \left\| \hat{v}(\mathbf{z}_\tau; \tau, \mathbf{y}) - (\epsilon - \mathbf{z}) \right\|^2 \right], \tag{1}$$

where $\epsilon \sim \mathcal{N}(0, I)$ and $\mathbf{z}_\tau = (1-\tau)\mathbf{z} + \tau\epsilon$ is the linearly interpolated latent.

Taking the derivative of $\mathcal{L}_{\text{RF}}$ with respect to $\mathbf{z}$ yields:

$$\nabla_{\mathbf{z}} \mathcal{L}_{\text{RF}}(\theta; \mathbf{z}, \mathbf{y}) = \mathbb{E}_{\tau \sim \mathcal{U}(0,1),\, \epsilon} \Big[w(\tau) \left(\hat{v}(\mathbf{z}_\tau; \tau, \mathbf{y}) - (\epsilon - \mathbf{z}) \right) \cdot \left(\frac{\partial \hat{v}(\mathbf{z}_\tau; \tau, \mathbf{y})}{\partial \mathbf{z}} + I \right) \Big]. \tag{2}$$

Following the derivation style of Score Distillation Sampling (SDS) [11], we omit the term that backpropagates

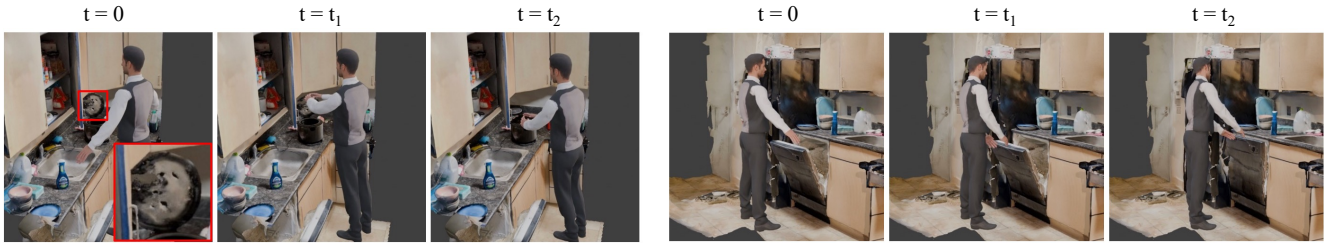


Figure 1. **Additional real-world animation results.** We present two additional examples of animating scanned real-world scenes. Our method generalizes well to both cases, even when the input mesh is only partially scanned.

Table 1. **Comparison on robotic manipulation.** We compare our method with scene flows derived from video. The results show that our method achieves a higher success rate.

Method | Scene	Lower lamp	Pick banana	Close laptop
Video flow	1/8	5/8	2/8
Ours	**6/8**	**8/8**	**7/8**

through the RF model, $\frac{\partial \hat{v}(\mathbf{z}_\tau;\tau,\mathbf{y})}{\partial \mathbf{z}}$, and apply the chain rule from $\mathbf{z}$ back to the 4D representation parameters θ. This gives the RF-SDS gradient used in the main text:

$$\nabla_\theta \mathcal{L}_{\text{RFSDS}}(\theta;\mathbf{z},\mathbf{y}) = \mathbb{E}_{\tau,\epsilon}\left[w(\tau)\left(\hat{v}(\mathbf{z}_\tau;\tau,\mathbf{y}) - (\epsilon - \mathbf{z})\right)\frac{\partial \mathbf{z}}{\partial \theta}\right]. \quad (3)$$

C. More Experiment Results

In this section, we present additional experimental results for our method.

C.1. Additional Real-world Results

We scan two additional real-world scenes using an iPhone and apply our pipeline to generate 4D motions of the objects within them. As shown in Figure 1, our method generalizes well to these scans, even when the input geometry is only partially captured, as highlighted by the red rectangles.

C.2. Additional Robotics Experiments

We compare our manipulation pipeline against an ablated pipeline using naive scene flows derived from video (by generating a single video and estimating 3D point tracking), while keeping the planner identical. As shown in Table 1, our pipeline consistently outperforms this baseline in success rate. The baseline often fails due to hallucinations from video models, depth-estimation errors, or heavy occlusion. In contrast, our explicit 3D representation yields physically-plausible scene flows for robot execution.

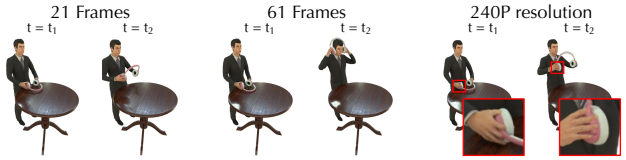


Figure 3. **Visual results of our method under different settings.** Reducing the number of frames leads to incomplete motion, while lowering the training resolution speeds up optimization but results in loss of fine-grained motion details (highlighted in red).

Table 2. **Training time under different settings.** Runtime scales approximately linearly with the number of frames and quadratically with image resolution.

Setting	21 frames	61 frames	240P	Original
Training time (h)	8.0	23.5	4.2	17.7

C.3. Analysis on Resolution and Number of Frames

We evaluate our method under different training resolutions and numbers of frames. The corresponding training times are reported in Table 2, and qualitative results are shown in Figure 3. We observe that the training time scales approximately linearly with the number of frames. However, reducing the number of frames leads to incomplete motion. Furthermore, decreasing the training resolution significantly reduces training time, approximately following a quadratic relationship with image size. While the resulting animations preserve large-scale motion, fine-grained motion details are lost, as highlighted by the red rectangle.

C.4. Comparison on Single Mesh Animation

We further compare our method with baselines on the task of single-object mesh animation. The set of baselines follows the main paper: Animate3D [7], AnimateAnyMesh [14], MotionDreamer [12], and 4D reconstruction from videos generated by TrajectoryCrafter [17]. We evaluate all methods on five prompts: "The lid of a chest is closing", "A lamp is lowering its head", "The blades of a pair of scissors cross together", "A tiger is sitting down", "A tiger is walking".

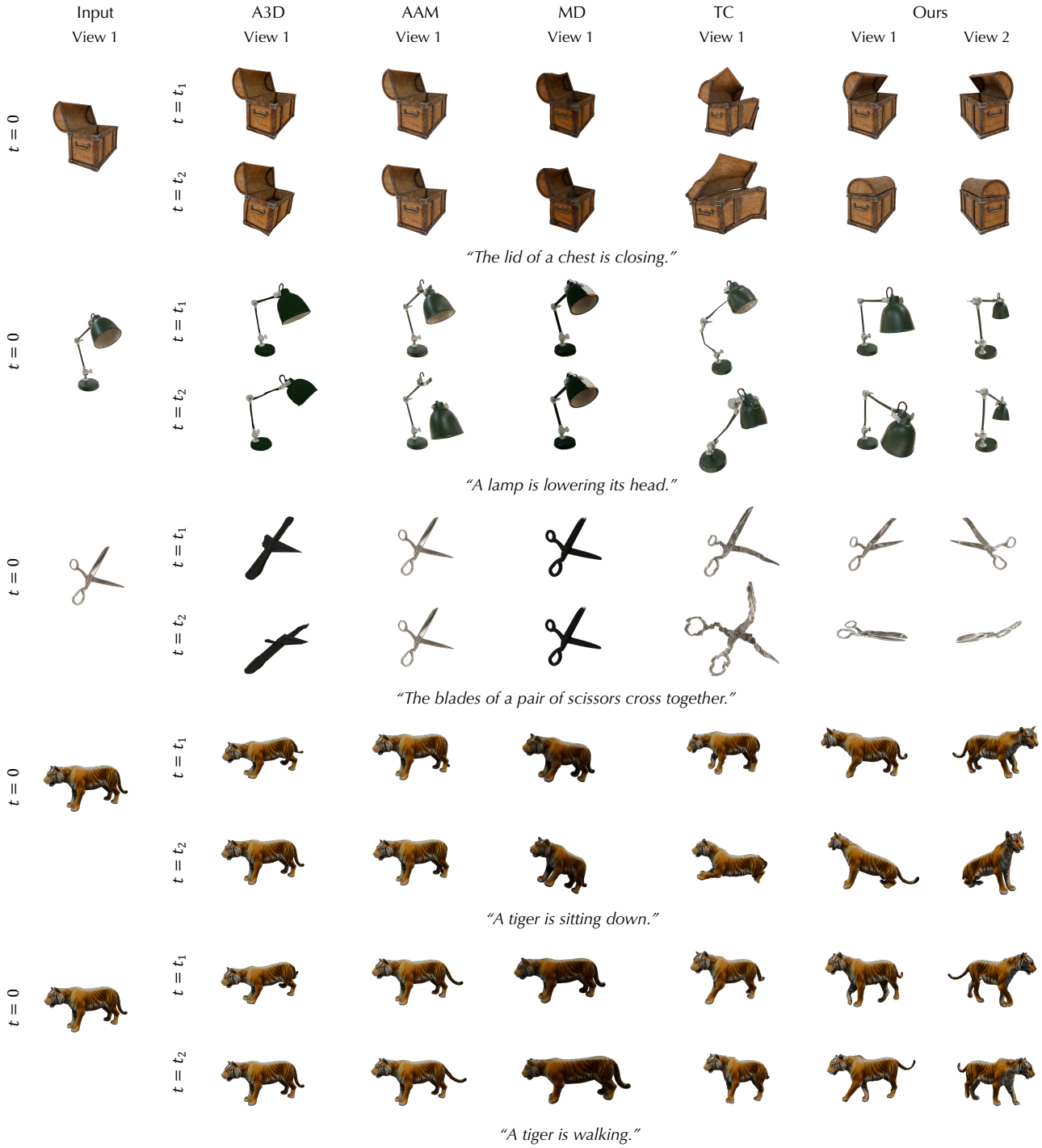


Figure 2. **Qualitative comparisons on single mesh animation.** We compare our approach with several mesh animation methods. Our method produces results that better align with the given prompts and exhibit more natural motion. In the figure, A3D refers to Animate3D [7], AAM denotes AnimateAnyMesh [14], MD represents MotionDreamer [12], and TC corresponds to 4D reconstruction results from videos generated by TrajectoryCrafter [17].

Qualitative results are shown in Figure 2. Our method consistently achieves better prompt alignment and produces more natural motion than existing approaches. For quantitative evaluation, we conducted a user study with 50 participants comparing our results against all baselines: **89.6%** of participants rated our method highest in prompt alignment,

and **84%** rated it highest in motion realism. These results further indicate the strength of our approach relative to existing methods. The full results are provided in Table 4.

C.5. Failure Cases

Our failure cases mainly arise from two factors: (1) limitations of the underlying video generative model, and (2) the inability to handle objects that do not exist in the static snapshot but appear later in the motion sequence. Examples are shown in Figure 4. Our failure cases mainly arise from two factors: (1) limitations of the underlying video generative model, and (2) the inability to handle objects that do not exist in the static snapshot but appear later in the motion sequence. Examples are shown in Figure 4. We elaborate on them below.

Video Generative Model Limitation. Because our approach distills from a pretrained video generation model, its capabilities are inherently linked to those of the underlying model. If the generator cannot synthesize videos aligning with the prompt, our 4D optimization receives misleading gradients. In such cases, our method cannot generate the correct motion. This is shown in the first row of Figure 4, where the video model repeatedly fails to sample videos consistent with the prompt, leading our method to produce incorrect motion.

Inability to Handle Newly Appearing Objects. Another limitation of our method is that it cannot handle objects that do not exist in the initial static snapshot. Our 4D representation only deforms the geometry present at the start, so any object that should appear later in the sequence cannot be created. When the prompt involves new objects entering the scene, the supervision asks for motion that the system cannot produce. In these cases, the optimization either omits the requested effect or yields incomplete motion, as illustrated in the second row of Figure 4, where no liquid appears because the system cannot introduce new geometry.

C.6. Full Table for User Study

In Table 3 and Table 4, we provide the complete user study results, including the number of participants who preferred each method for each scene. Across all scenes, our method receives the highest preference in both prompt alignment and motion realism.

D. Limitation and Future Work

Although our method can generate dynamic scenes with highly realistic interactions among multiple objects, there remain several limitations that point to promising directions for future work. For the failure cases described in Sec. C.5, those arising from limitations of the underlying video generative model may be alleviated as video generation technology continues to improve. For failures caused by newly appearing objects that are not present in the initial static scene, a potential solution is to incorporate a module capable of generating new geometry during the optimization process.

Apart from the failure cases, another limitation of our method is its extensive training time. In our observations, a substantial portion of the runtime is spent backpropagating through the VAE [9]. A promising future direction is to develop a distillation strategy that avoids backpropagating through the VAE entirely. This may be feasible because our objective is to generate motion rather than RGB appearance, suggesting that full VAE gradients may not be strictly necessary for effective motion supervision.

E. User Study Template

We provide screenshots of the user study interface in Figure 5 and Figure 6. Participants were asked to select the best, second-best, and third-best results among five methods. From left to right and top to bottom, the corresponding methods are: Animate3D [7], AnimateAnyMesh [14], MotionDreamer [12] using DynamiCrafter [15], MotionDreamer using Wan 2.2 [13], 4D reconstruction from videos generated by TrajectoryCrafter [17], and our method.

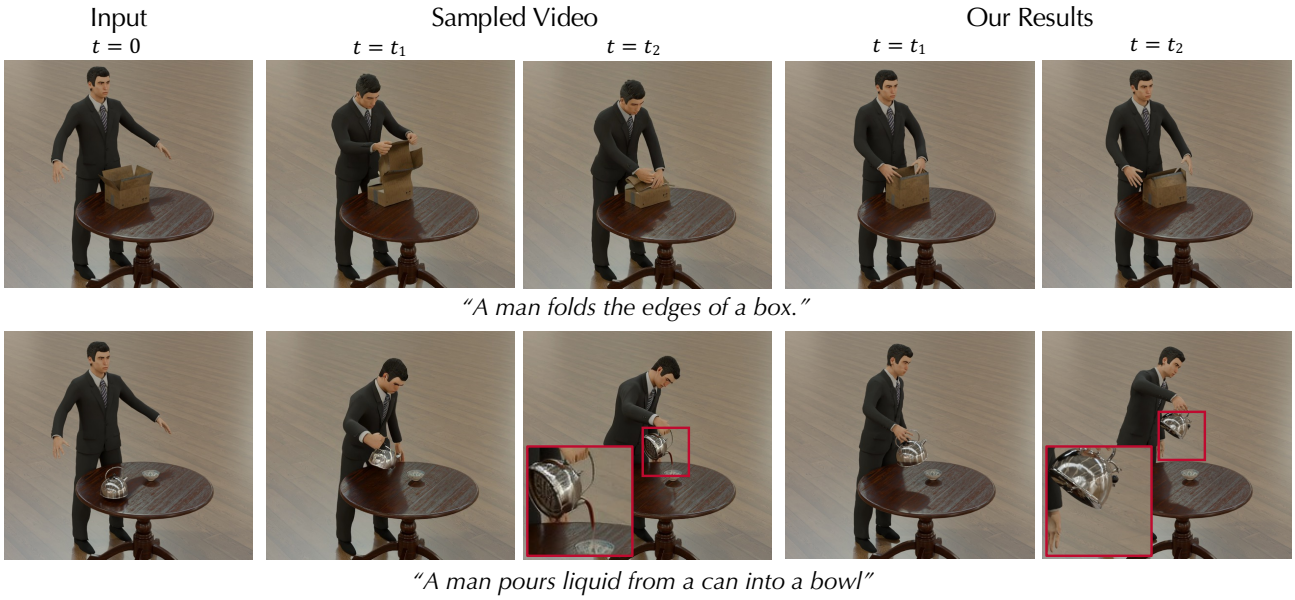


Figure 4. **Failure Cases.** The failure in the first row is due to limitations of the video generative model: it cannot produce motion that matches the prompt, as evidenced by its inability to sample videos aligned with the described action. The failure in the second row arises because our method cannot generate objects that were not present in the initial static scene. As a result, no liquid can appear when prompted, since the system cannot generate newly emerging objects.

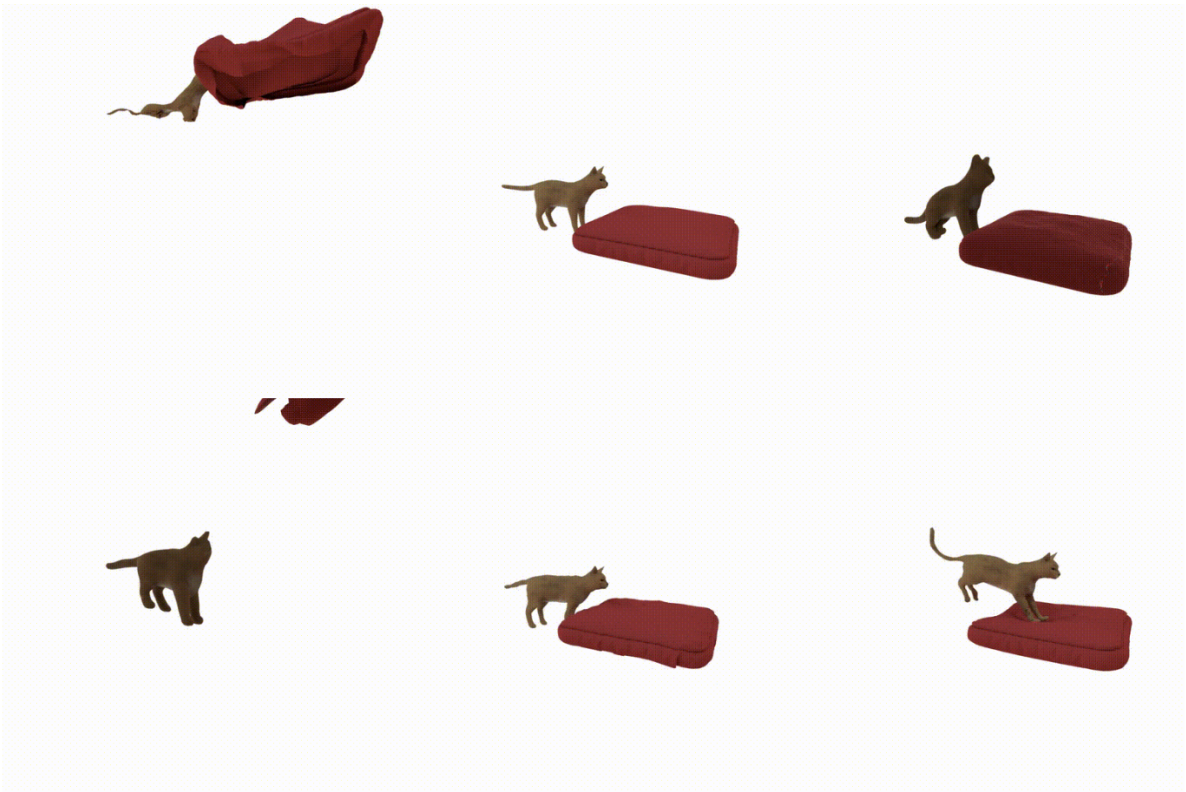
Table 3. Raw results of the user study on generating scene-level 4D motion. We show the number of vote from each participant on which option they consider the best under certain metric.

Metric	Object	Animate3D	AnimateAnyMesh	MotionDreamer (Orig)	MotionDreamer (Wan)	TC	Ours	Total
Prompt Alignment	Cat	0	2	0	1	0	96	99
	Dog	0	3	0	1	7	88	99
	Hugging	1	0	0	0	1	97	99
	Robot	0	0	1	1	2	95	99
	Sea Lion	1	0	1	2	43	52	99
	Brick	0	1	1	0	4	93	99
	Avg	0.3333	1	0.5	0.8333	9.5	86.8333	99
Motion Realism	Cat	0	0	1	1	2	95	99
	Dog	1	2	1	0	11	84	99
	Hugging	0	0	0	0	3	96	99
	Robot	0	1	1	1	1	95	99
	Sea Lion	1	0	2	0	37	59	99
	Brick	1	0	0	0	8	90	99
	Avg	0.5	0.5	0.8333	0.3333	10.3333	86.5	99

Metric	Object	Animate3D	AnimateAnyMesh	MotionDreamer (Orig)	MotionDreamer (Wan)	TC	Ours	Total
Prompt Alignment	Chest	2	0	1	0	0	47	50
	Lamp	0	1	2	1	0	46	50
	Scissors	1	0	1	1	0	47	50
	Sitting	4	1	1	0	5	39	50
	Walking	1	0	3	0	1	45	50
	Avg (raw)	1.6	0.4	1.6	0.4	1.2	44.8	50
Motion Realism	Chest	2	0	1	1	1	45	50
	Lamp	5	0	2	0	0	43	50
	Scissors	0	1	7	0	1	41	50
	Sitting	5	1	2	0	6	36	50
	Walking	2	2	1	0	0	45	50
	Avg (raw)	2.8	0.8	2.6	0.2	1.6	42	50

Table 4. User study results for quantitative comparison on single-object 4D motion generation.

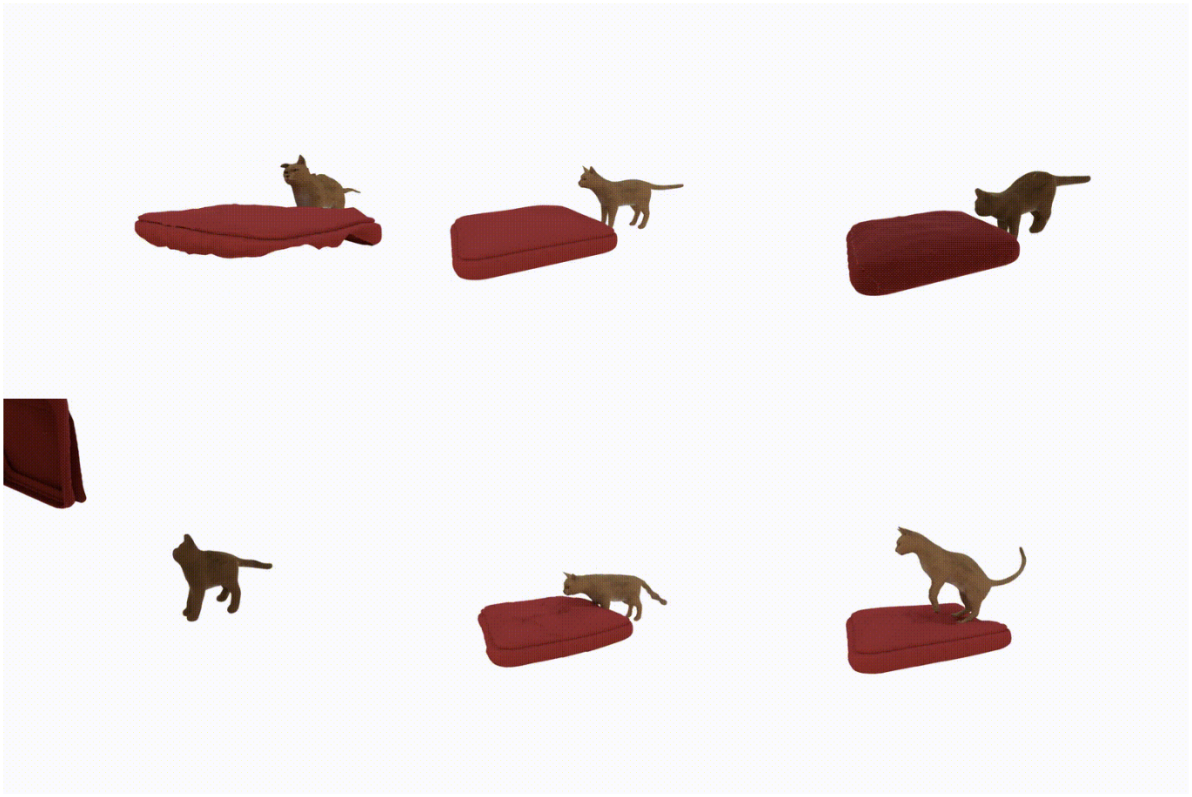
Which video best shows the concept of *a cat jumping on a cushion*? Please judge based on how well each one shows this concept. *



	Top Left	Top Middle	Top Right	Bottom Left	Bottom Mid...	Bottom Right
Best	○	○	○	○	○	○
Second Best	○	○	○	○	○	○
Third Best	○	○	○	○	○	○

Figure 5. **Screenshot of the user study question on Prompt Alignment.**

Which video looks like it has the most realistic and substantial movement? Please judge the videos based on this question. *



	Top Left	Top Middle	Top Right	Bottom Left	Bottom Mid...	Bottom Right
Best	○	○	○	○	○	○
Second Best	○	○	○	○	○	○
Third Best	○	○	○	○	○	○

Figure 6. **Screenshot of the user study question on Motion Realism.**

References

[1] Sketchfab. https://sketchfab.com. Accessed: 2025-11-18. 1

[2] Noam Aigerman, Kunal Gupta, Vladimir G. Kim, Siddhartha Chaudhuri, Jun Saito, and Thibault Groueix. Neural jacobian fields: Learning intrinsic mappings of arbitrary meshes. *ACM Transactions on Graphics (SIGGRAPH 2022)*, 2022. 1

[3] BlenderKit. Blenderkit online asset library. https://www.blenderkit.com. Accessed: 2025-11-18. 1

[4] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. 1

[5] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Proceedings of the 41st International Conference on Machine Learning*, pages 12606–12633, 2024. 1

[6] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. 1

[7] Yanqin Jiang, Chaohui Yu, Chenjie Cao, Fan Wang, Weiming Hu, and Jin Gao. Animate3d: Animating any 3d model with multi-view video diffusion. *Advances in Neural Information Processing Systems*, 37:125879–125906, 2024. 1, 2, 3, 4

[8] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023. 1

[9] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014. 4

[10] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023. 1

[11] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *ICLR*. OpenReview.net, 2023. 1

[12] Lukas Uzolas, Elmar Eisemann, and Petr Kellnhofer. Motiondreamer: Exploring semantic video diffusion features for zero-shot 3d mesh animation. In *2025 International Conference on 3D Vision (3DV)*, pages 893–904. IEEE, 2025. 1, 2, 3, 4

[13] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025. 1, 4

[14] Zijie Wu, Chaohui Yu, Fan Wang, and Xiang Bai. Animateanymesh: A feed-forward 4d foundation model for text-driven universal mesh animation. pages 13557–13568, 2025. 1, 2, 3, 4

[15] Jinbo Xing, Menghan Xia, Yong Zhang, Haoxin Chen, Wangbo Yu, Hanyuan Liu, Gongye Liu, Xintao Wang, Ying Shan, and Tien-Tsin Wong. Dynamicrafter: Animating open-domain images with video diffusion priors. In *European Conference on Computer Vision*, pages 399–417. Springer, 2024. 4

[16] Vickie Ye, Ruilong Li, Justin Kerr, Matias Turkulainen, Brent Yi, Zhuoyang Pan, Otto Seiskari, Jianbo Ye, Jeffrey Hu, Matthew Tancik, and Angjoo Kanazawa. gsplat: An open-source library for gaussian splatting. *Journal of Machine Learning Research*, 26(34):1–17, 2025. 1

[17] Mark Yu, Wenbo Hu, Jinbo Xing, and Ying Shan. Trajectorycrafter: Redirecting camera trajectory for monocular videos via diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 100–111, 2025. 1, 2, 3, 4