V2M4: 4D Mesh Animation Reconstruction from a Single Monocular Video – Appendix –

Jianqi Chen Biao Zhang Xiangjun Tang Peter Wonka
KAUST
{jianqi.chen, biao.zhang, xiangjun.tang, peter.wonka}@kaust.edu.sa

A. Implementation Details

In Sec. 4 of the main paper, we described the design of each component in our framework. These components are sequentially integrated according to the overall pipeline shown in Fig. 2 of the main paper. Below, we provide additional implementation details for each component of our framework.

Camera Search. In the camera search workflow, we set the number of initially sampled large-scale camera poses P_0 to 2000. The yaw and pitch angles are sampled using an equal-area distribution around the sphere. The radius is sampled with steps following a square root distribution within the range [1.0, 5.0], and $\mathrm{lookat}_{x,y,z}$ are uniformly sampled within the range [-1.0, 1.0] for all three axes. We select the top $n = 7$ camera poses, along with the reference video frame, to input to DUSt3R. The scoring function between the rendering view and the reference view is a combination of the DreamSim loss [1] and the foreground mask area loss, with respective weights of 1 and 0.1.

In the utilization of the dense stereo reconstruction model, we obtain the ground truth point clouds of $\widetilde{\mathcal{M}}_t$ during the rasterization process using Nvdiffrast [3]. To speed up optimization and filter out possible outliers in the point clouds, we retain only 5% points of both the ground truth point clouds $\mathrm{PC}_{\mathrm{rend},\{1,\ldots,n\}}$ and the predicted ones $\overline{\mathrm{PC}}_{\mathrm{rend},\{1,\ldots,n\}}$. We then align these two sets of point clouds using Chamfer loss to optimize the global transformation (rotation, scale, and translation) over 2000 iterations. To prevent the alignment from becoming trapped in local minima, such as an object being flipped along its vertical axis, we manually flip the point clouds along the axis starting at the 1000th iteration. We then compare the Chamfer loss before and after flipping to select the best alignment. After aligning the point cloud sets, we optimize the camera pose for 500 iterations using MSE loss between the 3D point clouds of $\mathrm{PC}_{\mathrm{ref}}$ and the 2D pixel positions in $V_{\mathrm{ref},t}$.

We select the top $K = 199$ camera poses from P_0, along with the camera pose from DUSt3R, for the PSO algorithm. We set the number of iterations for PSO to 25. For the subsequent gradient descent optimization, we set the iterations to 300 and use only the MSE loss between the foreground mask area of the rendering view and the reference video frame.

Mesh Appearance Refinement. We enable optimization of the negative condition embedding in TRELLIS. At each inference step of the generative model (conditioned on this embedding), we decode the output into a mesh. By computing the visual alignment between the rendering view of the reconstructed mesh and Gaussian Splats, we backpropagate the gradient through differentiable rendering to the negative condition embedding in TRELLIS. We use a combination of DreamSim, LPIPS, and MSE losses (each with a weight of 1), along with an MSE regularization term weighted at 0.2 to compare with the embedding's initial value, preventing distortion of the reconstructed result. This optimization is applied only during the latter part of the flow model ($0.6 \geq t \geq 0$, where t represents the timestep of the flow model). We begin the optimization at iteration 5 and gradually increase by 1 every 5 timesteps of the flow model.

Topology Consistency. For global alignment, we use Chamfer loss and L1 loss between the rendering views of the two meshes, with equal weighting for both losses. The optimization is performed over 500 iterations, using 20 randomly selected views around the mesh for the rendering loss.

For local alignment, we set the iterations to 1000 and use 50 views for the rendering view L1 loss. In addition to Chamfer loss, we incorporate ARAP loss, Face Area Consistency loss (which penalizes face area changes during mesh deformation), and Edge Length Consistency loss (which penalizes edge length changes). Due to the small values of Face Area Consistency loss and Edge Length Consistency loss, we assign them weights of $1e^6$ and $1e^2$, respectively. All other losses are assigned a weight of 1.

Total	Mesh Gen	Camera Pose Search		
		DUSt3R	PSO	SGD
57.7s	1.5s	13.9s	11.1s	0.5s
Mesh Refine	Mesh Regist	Texture Map Optim	Mesh Interp	4D Asset Convert
14.1s	15.6s	0.3s	0.6s	0.1s

Table 1. Detailed average runtime per frame for each component of our framework.

Texture Consistency. For the first mesh, we select 100 random views around it, and for the subsequent meshes, we use only their rendering views under $\mathcal{C}_{\text{ref}}$. The global texture map is optimized based on all these rendering views over 2500 iterations, using both L1 loss and total variation loss to ensure natural smoothness.

Mesh Interpolation. We set the frame interpolation to 5 for the complex data collected online due to their high FPS, and to 3 for the Consistent4D data, which has a relatively low FPS.

Metric Calculation Settings. For accurate visual similarity evaluation, we use the "ViT-bigG-14" model provided by OpenCLIP [2], trained on the LAION-2B [6] dataset, to calculate the CLIP score. For FVD calculation, we use StyleGAN-V [7]. For the LPIPS metric calculation, we use the VGG model. Since the FVD metric requires input videos to have the same number of frames, for reconstruction results on our additionally collected long animation videos, we split the rendering video into subsequences of 32-frame videos and calculate FVD on all of them. For the final subsequence that has fewer than 32 frames, we exclude it from the calculation.

All experiments in this paper were conducted on a single A100 GPU. Table 1 provides detailed runtime information for each framework component as a supplement to Table 3 in the main paper.

B. Ablation Studies

B.1. Quantitative Ablation Studies

Ablation Study on Different Base 3D Generation Models. In Table 2, we replace our base 3D generator, TRELLIS, with several contemporary models, including Hunyuan3D-2.0 [9], TripoSG [5], and CraftsMan3D [4]. The results demonstrate that using more advanced 3D generators (such as Hunyuan3D-2.0 and TripoSG) leads to corresponding improvements in performance, illustrating the extensibility of our method alongside ongoing advancements in 3D generation techniques.

Ablation of Components and Framework Complexity Analysis. Table 3 presents an ablation study of the key components in our framework, confirming the effectiveness

Dataset	Method	CLIP↑	LPIPS↓	FVD↓	DreamSim↓
Simple	Naïve TRELLIS	0.8905	0.1597	1342.66	0.1282
	Naïve CraftsMan3D	0.9177	0.1702	1185.47	0.0924
	Naïve TripoSG	**0.9259**	0.1534	973.16	0.0657
	Naïve Hunyuan3D 2.0	0.9185	**0.1471**	**851.01**	**0.0647**
	V2M4 (TRELLIS)	0.9259	0.1017	825.59	0.0688
	V2M4 (CraftsMan3D)	0.9212	0.1051	960.63	0.0772
	V2M4 (TripoSG)	**0.9286**	0.0971	758.25	0.0652
	V2M4 (Hunyuan3D 2.0)	**0.9286**	**0.0885**	**691.48**	**0.0634**
Complex	Naïve TRELLIS	0.8887	0.1265	1216.19	0.1492
	Naïve CraftsMan3D	0.9060	0.1350	1162.00	0.1027
	Naïve TripoSG	**0.9201**	0.1404	1023.88	**0.0891**
	Naïve Hunyuan3D 2.0	0.9097	**0.1185**	**1022.81**	0.0962
	V2M4 (TRELLIS)	0.9008	0.0747	666.04	0.1220
	V2M4 (CraftsMan3D)	0.9268	0.0709	558.34	0.1046
	V2M4 (TripoSG)	**0.9359**	0.0608	433.38	**0.0873**
	V2M4 (Hunyuan3D 2.0)	0.9192	**0.0605**	**415.53**	0.0990
All	Naïve TRELLIS	0.8891	0.1352	1014.45	0.1438
	Naïve CraftsMan3D	0.9098	0.1446	970.24	0.0999
	Naïve TripoSG	**0.9222**	0.1436	859.04	**0.0833**
	Naïve Hunyuan3D 2.0	0.9144	**0.1284**	**796.98**	0.0853
	V2M4 (TRELLIS)	0.9073	0.0817	576.73	0.1082
	V2M4 (CraftsMan3D)	0.9270	0.0795	524.00	0.0977
	V2M4 (TripoSG)	**0.9359**	0.0698	401.71	**0.0818**
	V2M4 (Hunyuan3D 2.0)	0.9224	**0.0674**	**377.14**	0.0901

Table 2. Impact of different base 3D generators on the overall performance.

Base Model	Strategy	CLIP↑	LPIPS↓	FVD↓	DreamSim↓
TRELLIS	Final	**0.9259**	0.1017	825.59	0.0688
	w/o PSO	0.8855	0.1158	1090.86	0.1184
	w/o DUSt3R	0.9236	0.1046	**756.29**	0.0713
	w/o SGD	0.9146	0.1269	1086.14	0.0861
	w/o Mesh Refinement	0.9028	0.1103	1023.60	0.1032
	Replace DUSt3R with VGGT	0.9230	**0.1006**	821.22	**0.0631**
CraftsMan3D	Replace DUSt3R with VGGT	0.9207	0.1046	887.53	0.0753
TripoSG	Replace DUSt3R with VGGT	0.9250	0.0960	730.13	0.0757
Hunyuan3D-2.0	Replace DUSt3R with VGGT	**0.9279**	**0.0879**	**727.24**	**0.0636**

Table 3. Ablation study about the impact of key components in our framework.

Parameter	Value	CLIP↑	LPIPS↓	FVD↓	DreamSim↓
DreamSim (Camera Search)	0.1	0.9176	0.1058	**824.20**	0.0784
	1	**0.9259**	**0.1017**	825.59	**0.0688**
	10	0.9234	0.1037	853.18	0.0726
DreamSim (Mesh Refinement)	0.1	0.9129	0.1054	876.41	0.0911
	1	**0.9259**	**0.1017**	**825.59**	**0.0688**
	10	0.9226	0.1078	932.72	0.0750
ARAP	0.1	0.9200	0.1046	**814.31**	0.0727
	1	**0.9259**	**0.1017**	825.59	**0.0688**
	10	0.9258	0.1058	928.54	0.0744

Table 4. Impact of varying DreamSim and ARAP loss weights.

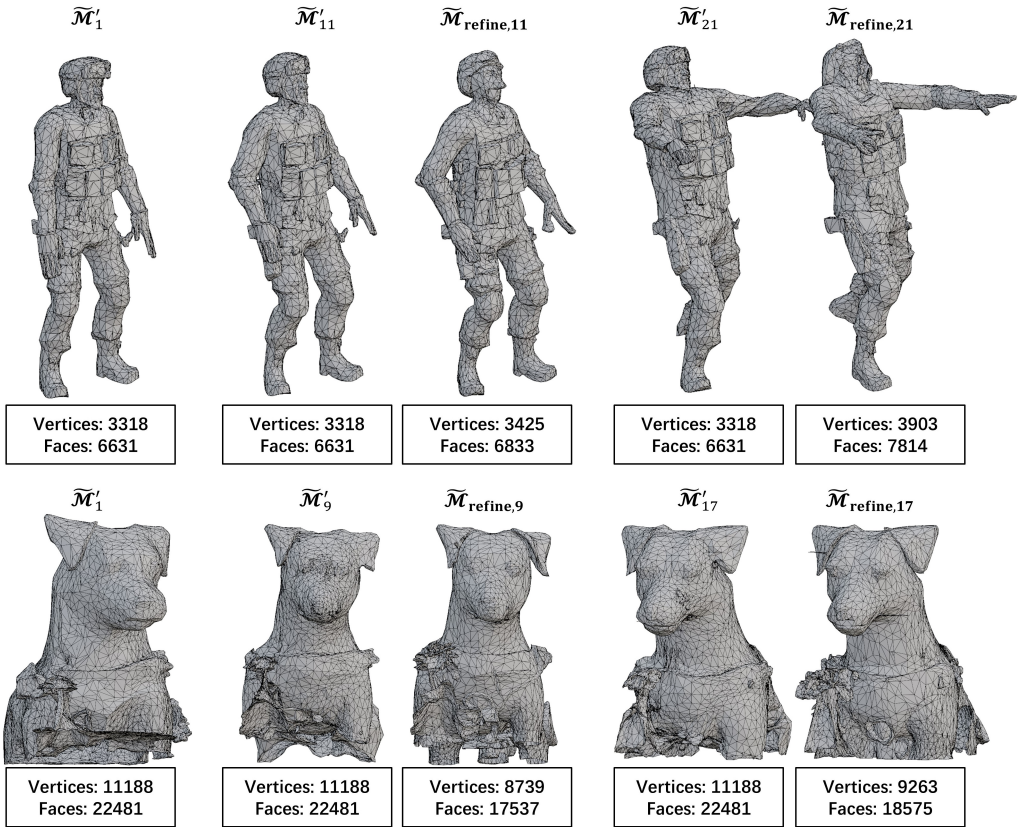

Figure 1. **Topological Consistency Between Meshes.** $\widetilde{\mathcal{M}}'_t$ represents the registered meshes derived from $\widetilde{\mathcal{M}}'_1$, while $\widetilde{\mathcal{M}}_{\text{refine},t}$ denotes the original reconstructed meshes. The number of vertices and faces is displayed for clearer comparison.

Noise	CLIP↑	LPIPS↓	FVD↓	DreamSim↓
Original	0.9259	0.1017	825.59	0.0688
0.1×	0.9289	0.1022	942.99	0.0686
1×	0.9259	0.1077	947.57	0.0771
10×	0.8016	0.1694	2410.03	0.2577

Table 5. Noise impact.

of each procedure. We also evaluate the impact of replacing the default dense stereo reconstruction model DUSt3R with the more advanced VGGT [8], demonstrating additional performance improvements.

Regarding theoretical complexity, our framework exhibits linear time complexity $\mathcal{O}(T)$ with respect to the number of frames T, while maintaining almost constant memory usage $\mathcal{O}(1)$.

Quantitative Parameter Tuning for DreamSim and ARAP. As outlined in Appendix A, during the optimization process, we assign only a loss weight factor for DreamSim and ARAP losses, which are set to 1 by default. Table 4 illustrates the performance impact of varying these parameters.

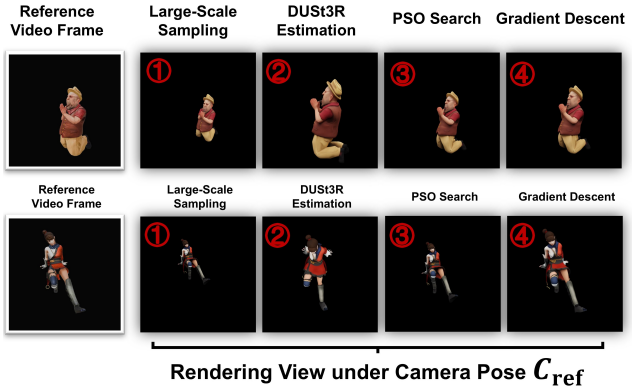

Figure 2. **Performance of the Camera Search Design.** We present intermediate results from each phase of our camera search workflow. The order of the intermediate results is highlighted with red index numbers.

Robustness to Base Mesh Quality. As discussed in the limitations section of the main paper, the quality of the base mesh significantly impacts the results. Table 2 shows that higher-quality meshes produced by advanced 3D generators (e.g., Hunyuan3D-2.0 and TripoSG) yield improved robust-

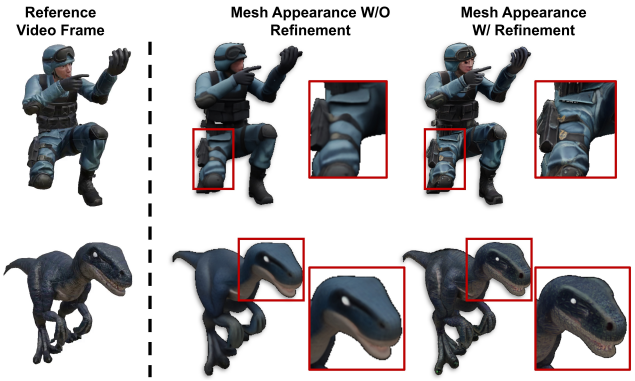

Figure 3. **Performance of the Mesh Appearance Refinement.** We present the mesh appearance before and after applying our mesh appearance refinement technique. Specific parts are enlarged for better comparison. Please zoom in for a clearer view.

ness. Additionally, Table 5 provides a detailed analysis of robustness by evaluating the effects of injecting varying levels of Gaussian noise during mesh generation.

B.2. Qualitative Ablation Studies

Topology Consistency. In Fig. 1, we present both the registered meshes and the original mesh throughout the animation. The original mesh successfully reconstructs to match subsequent timestamp meshes, demonstrating the effectiveness of our design in Sec. 4.3.

Camera Search. In Fig. 2, we display the rendering views under the identified camera poses at different phases of the camera search workflow described in Sec. 4.1. Specifically, we show the top-1 camera view after extensive camera pose sampling, the camera view obtained from DUSt3R estimation, the camera view after the Particle Swarm Optimization (PSO) search, and the final camera view following gradient descent refinement (See details in Algorithm 1 of the main paper). The results demonstrate that our camera search method is both effective and robust, successfully finding the camera pose that aligns with the reference video frame. This alignment subsequently supports accurate mesh reposing.

Mesh Appearance Refinement. In Fig. 3, we present the mesh before and after applying our refinement strategy described in Sec. 4.2. It is evident that the refined mesh exhibits improved texture and is much more aligned with the appearance shown in the reference video frame.

C. Failure Cases

In Fig. 4, we display instances where our method encounters failures, including effects from poor initial 3D mesh

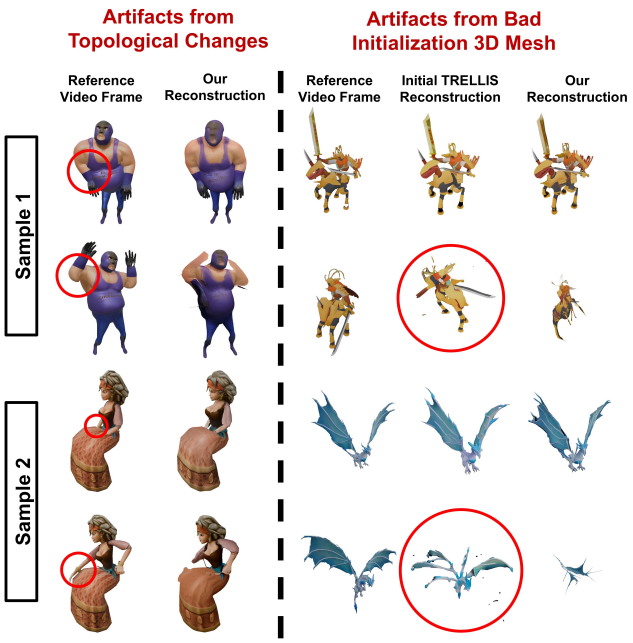

Figure 4. **Failure Cases of Our Method.** We present the limitations of our method in scenarios involving topological changes and poor 3D mesh initialization.

results from TRELLIS and artifacts arising from topology changes during the animation.

D. More Qualitative Results

We display more qualitative results in Fig. 5 and Fig. 6.

References

[1] Stephanie Fu, Netanel Tamir, Shobhita Sundaram, Lucy Chai, Richard Zhang, Tali Dekel, and Phillip Isola. Dreamsim: Learning new dimensions of human visual similarity using synthetic data. *arXiv preprint arXiv:2306.09344*, 2023. 1

[2] Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Brandon Tran, Ludwig Schmidt, Vaishaal Shankar, and Simon Kornblith. Openclip, 2021. 2

[3] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for high-performance differentiable rendering. *ACM Transactions on Graphics*, 39(6), 2020. 1

[4] Weiyu Li, Jiarui Liu, Hongyu Yan, Rui Chen, Yixun Liang, Xuelin Chen, Ping Tan, and Xiaoxiao Long. Craftsman3d: High-fidelity mesh generation with 3d native diffusion and interactive geometry refiner. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5307–5317, 2025. 2

[5] Yangguang Li, Zi-Xin Zou, Zexiang Liu, Dehu Wang, Yuan Liang, Zhipeng Yu, Xingchao Liu, Yuan-Chen Guo, Ding Liang, Wanli Ouyang, et al. Triposg: High-fidelity 3d

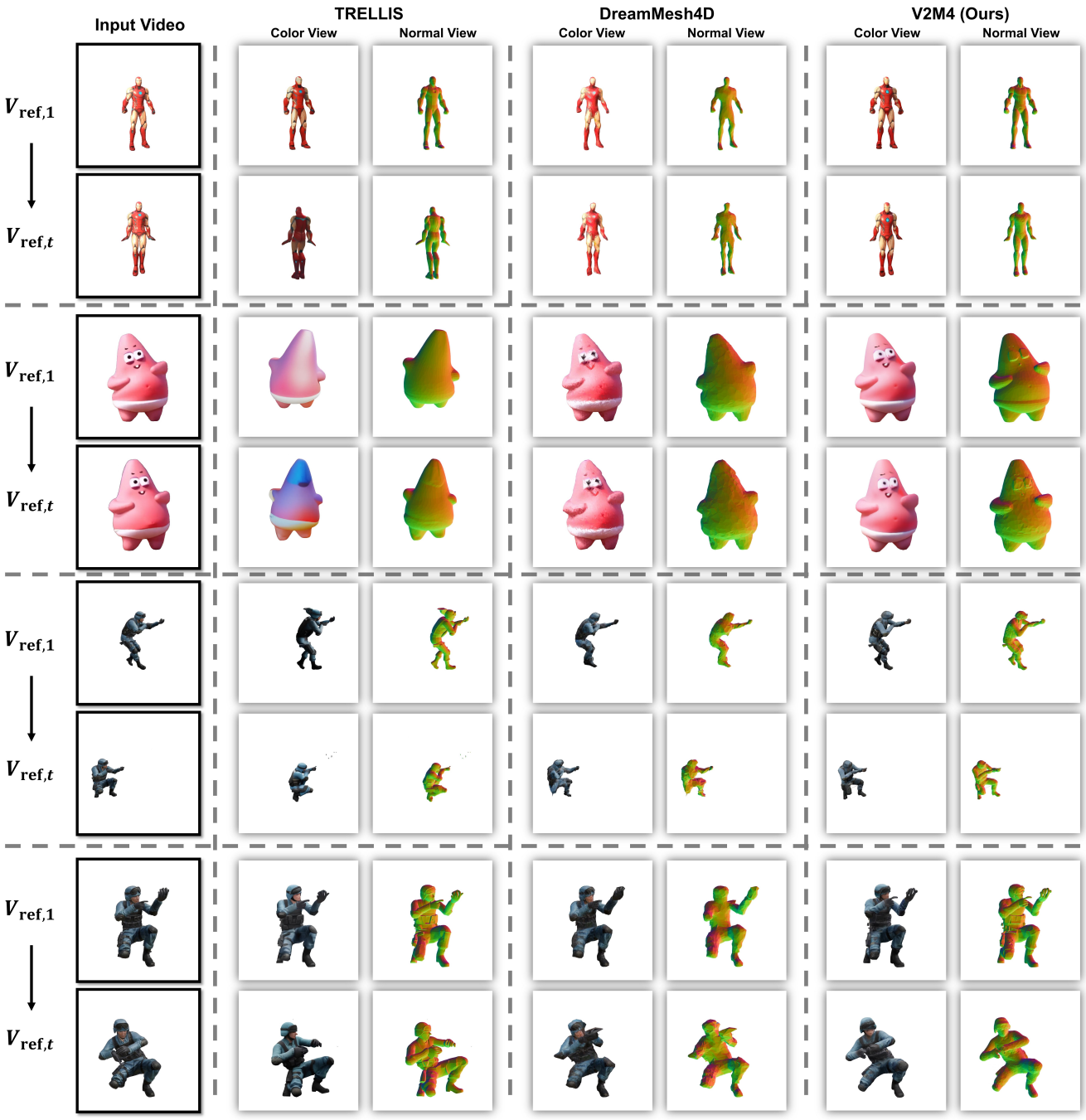

Figure 5. **More Qualitative Comparisons among Different Methods.** We present both the color view and the normal view of the reconstructed meshes. The first two samples are from the Consistent4D dataset, while the others are from our newly collected data. For each sample, two timestamps are shown for quick comparison. Please zoom in for a clearer view.

shape synthesis using large-scale rectified flow models. *arXiv preprint arXiv:2502.06608*, 2025. 2

[6] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in neural information processing systems*, 35:25278–25294, 2022. 2

[7] Ivan Skorokhodov, Sergey Tulyakov, and Mohamed Elhoseiny. Stylegan-v: A continuous video generator with the price, image quality and perks of stylegan2, 2021. 2

[8] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea

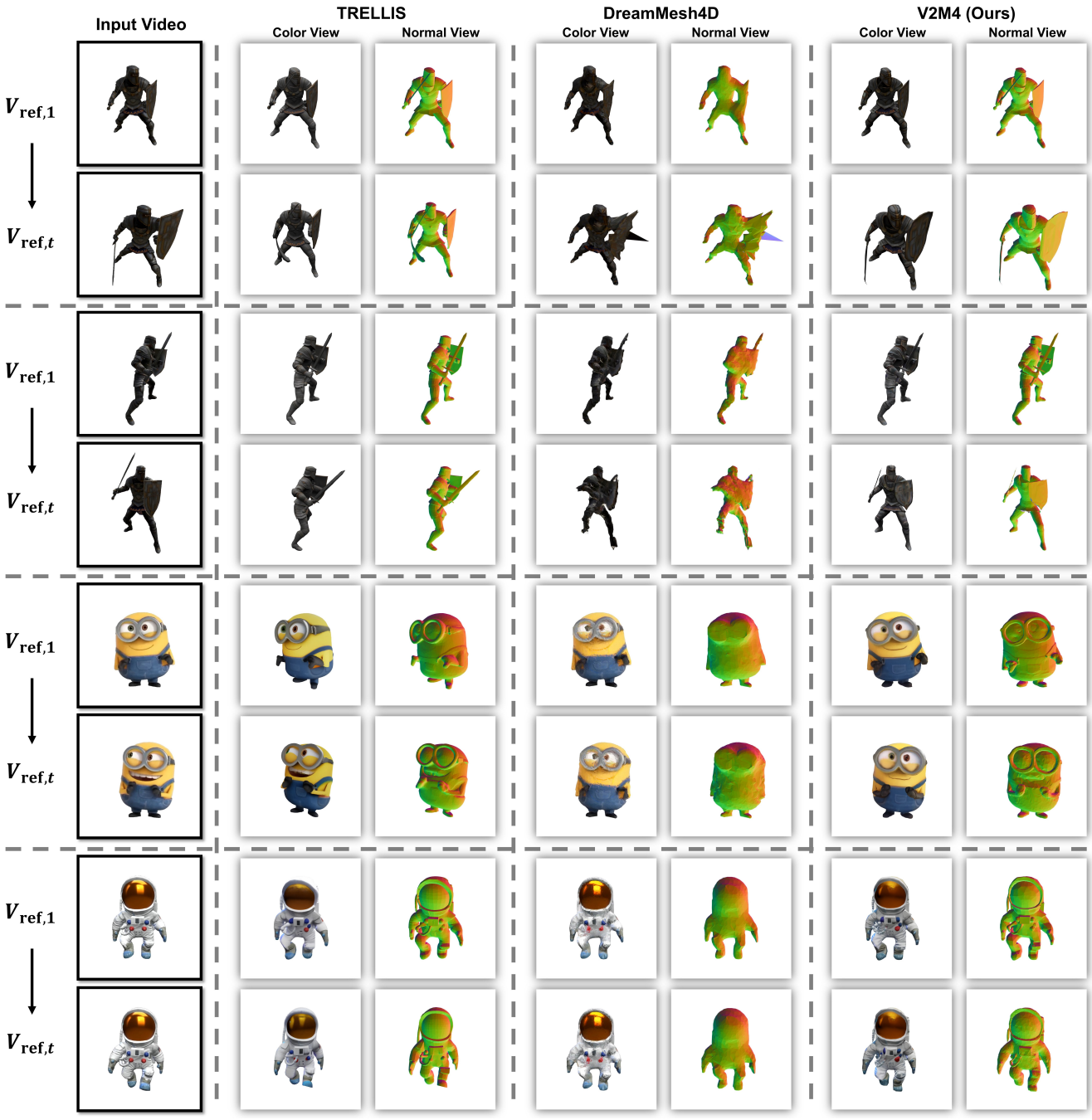

Figure 6. **More Qualitative Comparisons among Different Methods.** We present both the color view and the normal view of the reconstructed meshes. The first two samples are from the Consistent4D dataset, while the others are from our newly collected data. For each sample, two timestamps are shown for quick comparison. Please zoom in for a clearer view.

Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5294–5306, 2025. 3

[9] Zibo Zhao, Zeqiang Lai, Qingxiang Lin, Yunfei Zhao, Haolin Liu, Shuhui Yang, Yifei Feng, Mingxin Yang, Sheng Zhang, Xianghui Yang, et al. Hunyuan3d 2.0: Scaling diffusion models for high resolution textured 3d assets generation. *arXiv preprint arXiv:2501.12202*, 2025. 2