Motion 3-to-4: 3D Motion Reconstruction for 4D Synthesis

Supplementary Material

A. Implementation Details

A.1. Training and Inference Details

Network Architecture. Our network architecture consists of two primary components: a shape encoder and a motion latent block. For the shape encoder, we adopt the architecture from 3DShape2VecSet [99], which provides strong representational capacity and high compression efficiency. Specifically, we employ a point cloud encoder to extract features from the canonical shape representation. The encoder takes $N = 4096$ sampled points as input. First, the point coordinates are projected through a Fourier feature embedding layer to obtain position embeddings. These embeddings are then concatenated with surface normals and color attributes, and subsequently projected to the model dimension $d = 768$ via a linear layer. To aggregate point features into a compact representation, we employ a QK-Norm cross-attention block with 64 learnable query tokens. The resulting tokens are further processed through 4 transformer layers with self-attention mechanisms to capture inter-point relationships, ultimately producing a semantically rich canonical shape representation $\mathbf{Z} \in \mathbb{R}^{64\times768}$.

For video encoding, we adopt a frozen DINOv2-ViT-B/14 model [56] to extract spatial features from each frame. Input videos are resized to 224×224 resolution and processed frame-by-frame through the DINOv2 encoder, which extracts 256 patch tokens per frame with dimension 768. For a video with T frames, we obtain image tokens $\mathbf{V} \in \mathbb{R}^{T\times256\times768}$. These tokens are combined with fixed positional embeddings generated using sinusoidal encoding over both temporal and spatial dimensions. To further aggregate spatio-temporal information from the video, we process these embeddings through Motion 3-to-4 blocks. Following the design of VGGT [75], we adopt an alternating global-frame attention architecture consisting of 16 layers (8 global and 8 frame). Both global and frame attention layers use QK-normalization. This alternating design effectively captures both spatial and temporal dependencies while maintaining computational efficiency.

To predict per-point motion trajectories, we extract 64 motion tokens from the processed video sequence. For each point in $M = 4096$ output point clouds, we construct a query using its position, normal, and color through the same embedding layer used in the shape encoder. We then apply a QK-Norm cross-attention layer where the per-point queries attend to the extracted motion tokens from the corresponding frame. This cross-attention mechanism produces per-point features $\mathbf{F} \in \mathbb{R}^{M\times768}$ that encode motion information. Finally, a two-layer MLP with GELU activation decodes these features into motion trajectories $\mathbf{X}_t \in \mathbb{R}^{M\times3}$ for time t. During inference, when we need to animate mesh vertices, we process them in chunks of 4096 to maintain memory efficiency. For temporal processing, we adopt a chunk size of 256 frames. When dealing with videos exceeding 256 frames, we use a sliding window approach with a stride of 255 frames, where each window consists of the first frame concatenated with 255 subsequent frames to maintain temporal consistency.

Training Configuration. We employ the AdamW optimizer with learning rate $\eta = 4\times10^{-4}$, $\beta_1 = 0.9$, $\beta_2 = 0.95$, and weight decay 0.05. The learning rate follows a cosine annealing schedule with 1000 warm-up steps. We train for 60,000 parameter update steps with gradient clipping at norm 1.0. To reduce memory consumption and accelerate training, we apply gradient checkpointing, FlashAttention-v2 in the xFormers library, and a mixed precision strategy with BF16.

Training Data and Strategy. For training data, we use videos at 256×256 resolution with black backgrounds. The point clouds are uniformly sampled on the mesh surface, and crucially, we sample points at consistent barycentric coordinates within each face across all frames. This ensures temporal correspondence between points across different frames, enabling us to track each point's trajectory and supervise the model using MSE loss.

For the training strategy, we train on 12-frame sequences and apply temporal data augmentation. Specifically, we randomly select a starting frame and then sample 12 consecutive frames with stride intervals of 1, 2, or 4 frames. This augmentation strategy enables the model to handle different initial poses and learn to predict larger motion displacements across varying temporal scales.

Inference with video. For cases where only video input is provided without any mesh, we leverage Hunyuan 2.0 [110] to generate a mesh based on the first frame. It is worth noting that the directly generated mesh is non-watertight and includes texture. To address the watertight issue, we apply a vertex mapping technique to convert it into a watertight mesh while preserving the original texture, which is essential for subsequent video-driven animation. For cases where a mesh is already provided, we directly drive the mesh using generated or existing videos.

A.2. Evaluation Details

We utilize the official release code for evaluation. For our held-out dataset, we use the front view as the input view and the remaining three orthogonal views for evaluation.

We exclude the front view from evaluation because including it would be unfair to generation-based methods, since L4GM [60] can perform lossless reconstruction of the input view (i.e., the front view).

In the Table 2, "OOM" refers to "out of memory." For the GVFD [100], the official released code does not provide scripts for processing long sequences; it only includes scripts for single-video inference. Consequently, when the sequence length exceeds 128 frames, our machine encounters the out-of-memory (OOM) problem, preventing us from obtaining quantitative results for this method.

A.3. Ablation Study

We conduct an ablation study to explore different model architecture choices. Due to limited training resources, we train each model variant for 30,000 steps under identical settings. To evaluate the trajectory prediction capability of each variant, we report the mean squared error (MSE) of point trajectories over time on the held-out dataset in Table 4, which provides a more direct measure of the model's ability to track accurate motion trajectories.

Frame Attn	*Global Attn*	*Ref Token*	Rec MSE ↓
✗	✓	✓	0.0055
✓	✗	✓	0.0033
✓	✓	✗	0.0021
✓	✓	✓	**0.0018**

Table 4. Ablation studies of the modules of Motion 3-to-4. Rec MSE denotes the squared error averaged across time steps within the $[-0.5, 0.5]$ bounding box.

B. More Results

We provide several categories of additional visualization results below. We also include a local video webpage in the supplementary materials to better present the results.

B.1. More Results from Synthesis Video

As illustrated in Figure 8, our model demonstrates strong generalization capability across diverse test cases from synthetic videos, achieving high-quality results that showcase its ability to handle various object types and motion patterns.

B.2. More Results from Real-World Video

As illustrated in Figure 9, despite being trained exclusively on synthetic data, our model generalizes well to real-world videos. This demonstrates the model's robustness and its ability to bridge the synthetic-to-real domain gap, highlighting the effectiveness of our method.

B.3. Results of Existing 3D Model

As illustrated in Figure 10, our model has strong practical value as it can extend existing static 3D assets to dynamic 4D content. This capability enables flexible application scenarios, such as animating existing 3D models from various sources, thereby significantly broadening the potential use cases of our method.

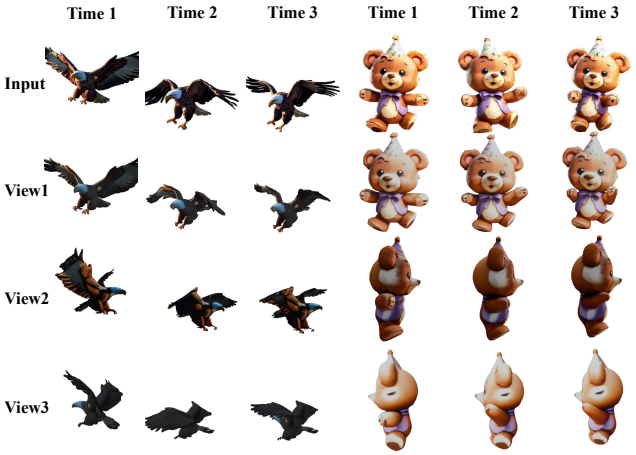

Figure 8. Additional visual results from synthesis video.

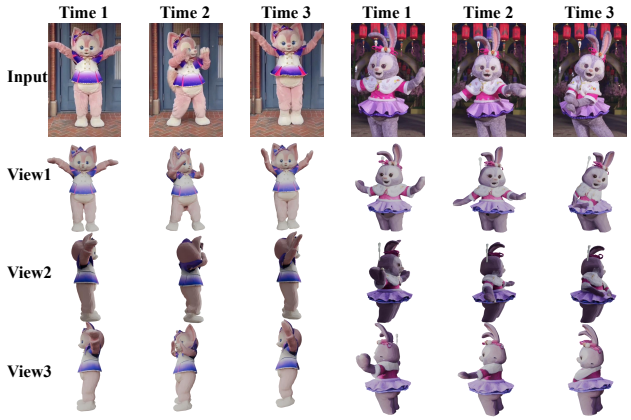

Figure 9. Additional visual results from real-world video.

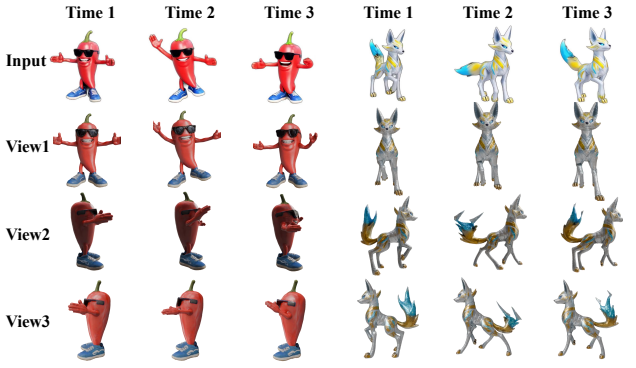

Figure 10. Results of existing 3D model condition on generated video.

B.4. Visual Comparison on Consist4D

The Consist4D dataset [29] provides input views from various angles that differ from the frontal views in our held-out dataset, which better demonstrates the model's robust-

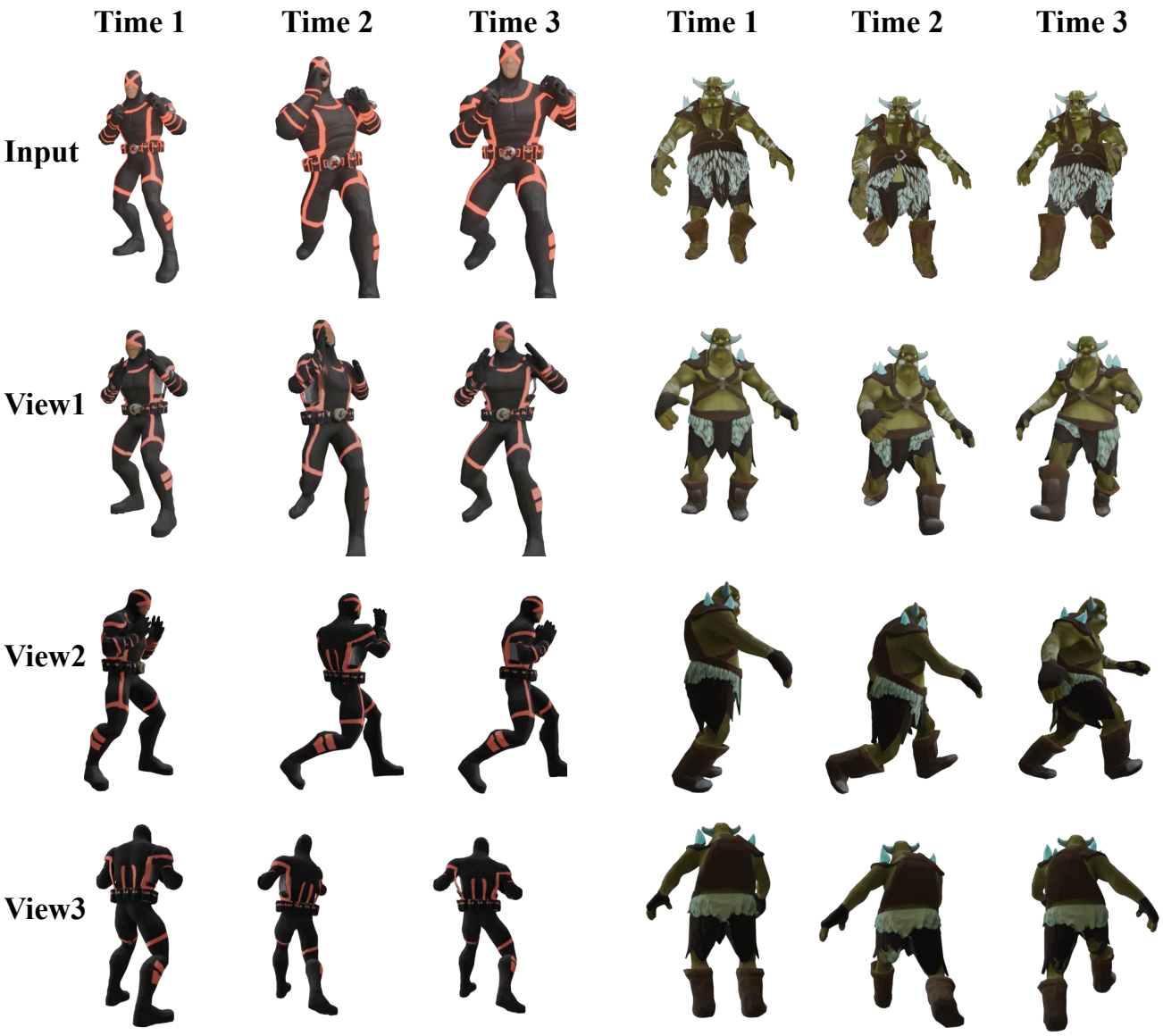

Figure 11. More results from the held-out objaverse dataset.

ness to different viewpoints. This is especially relevant for method L4GM, which is designed to work well under orthogonal views. When the input view is not orthogonal, L4GM's multi-view diffusion module fails to generate consistent results across views, leading to severe ghosting artifacts as shown in Figure 12.

B.5. More Results from the Held-Out Dataset

As illustrated in Figure 11, we present additional results on our held-out dataset to demonstrate that our model achieves superior visual fidelity and temporal motion coherence. These results further validate the effectiveness of our approach in generating high-quality 4D content with both realistic appearance and consistent motion over time.

Figure 12. Visual comparison with SOTA methods on Consist4D.