AnyMoLe: Any Character Motion In-betweening Leveraging Video Diffusion Models

Supplementary Material

Overview

In this supplementary material, we present implementation details in Section A. Section B presents the architecture of scene-specific joint estimator model. Section C presents additional results. Visual results and additional comparisons are presented in the supplementary video. Our code will be publicly released upon acceptance.

A. Implementation Details

A.1 ICAdapt

As described in the main paper, to train D_{adp}, images are rendered at 30 fps using four different views ($N = 4$) of context frames. From these four videos, we sample 16-image clips at 30 fps, 15 fps, and 10 fps using frame intervals of 1, 2, and 3, respectively. This fps information is passed through the fps embedding layer of D_{adp} as a control parameter for video speed. The dataset gathering process is outlined in Algorithm 1. Using these clips, each consisting of 16 images, ICAdapt is conducted for 500 steps with a learning rate of 1e-5, which requires approximately one and a half hours on an Nvidia A6000 GPU. During the ICAdapt process, the video captioning model [4] is used once, with the same text prompt applied throughout the entire process, because the animation is played within a single scene.

A.2 Scene-specific joint estimator

Scene-specific joint estimator $\mathcal{E}_{scene}$ is trained using three views ($N = 3$) in addition to the rendered keyframes with the weight $w = 3$. As shown in Figure 5 of the main paper, 2D and 3D features are extracted from rendered images. For 2D features, DINOv2 [3] is used. Recently, Darcet et al. [1] showed that training DINOv2 with register tokens can reduce high-norm outliers, leading to improved stabilization. Thus we employed them for 2D features. For 3D-aware features, we employed another DINOv2 variant Fit3D [5], which is also DINOv2 varient, finetuned at 3D scenes. The training was conducted for 3500 steps, which required around 3.5 hours on an Nvidia A6000 GPU. The detailed model architecture of Scene-specific joint estimator will be presented in Section B.

A.3 Two-stage Inference

Two-stage inference is conducted to generate video frames that follow the given context and produce smooth motion. This process is written in Algorithm 2. In the first stage, a coarse video at 5 fps is generated, with each inference covering a three second segment because D_{adp} outputs 16 frames at a time. In the second stage, a fine video at 15 fps is generated, with each inference covering one second segment. The process ensures both contextual alignment and smooth motion. We conducted an experiment to measure inference time. As shown in Table 1, two-stage inference has O(N) time complexity for input keyframes.

Algorithm 1 Gathering video clips from context frames

Require: N: Number of views (e.g., $N = 4$)
Require: N_{fpv}: Total number of frames per view
Require: k: Number of frames per clip (e.g., $k = 16$)
Require: S: List of frame intervals for sampling (e.g., $S = [1, 2, 3]$)
Ensure: $Clips$: A list of sampled clips

Initialize $Clips \leftarrow []$
for $v \leftarrow 1$ to N **do** ▷ Iterate through all views
  for all $s \in S$ **do** ▷ Iterate through all sampling
    for $start \leftarrow 0$ to $N_{fpv} - k \cdot s$ **do**
      $clip \leftarrow [start + i \cdot s$ **for** $i \in \{0, 1, ..., k-1\}]$
      $Clips \leftarrow Clips \cup \{clip\}$
    end for
  end for
end for
return $Clips$

Table 1. Two-stage inference time for 15fps videos.

Generated frames	15	30	45	60	75	90
Inference time (Minutes)	2.43	3.58	4.78	5.97	7.13	8.08

A.3 Motion video mimicking

Motion video mimicking is performed to generate motion that follows the generated video. We use joint loss, image loss, and regularization loss, as described in Eq. 3 of the main paper. The entire optimization process is conducted in a single view ($N = 1$), over 100 steps per sequence. This process is performed in batches to achieve faster optimization. For an 8-second video with 2 seconds of context frames, the batch size is 6, and the optimization process is repeated 14 times to fill the in-between frames.

B. Model Architecture

Our scene-specific joint estimator, $\mathcal{E}_{scene}$, consists of a feature merger, heatmap decoder, and depth MLP. As shown in Figure 1, our feature merger consists of convolutional blocks and concatenation. The output of the feature merger,

Algorithm 2 Two-Stage Video Generation with Context Guidance

Require: D_{adp}: Fine-tuned video diffusion model
Require: I_0, I_N: First and last keyframes of the target video
Require: $Total$: Total time in seconds for the video to generate
Require: I_m: Context frames for guidance, where $M_1 \leq m \leq M_2$, $M_2 < N$
Ensure: Final video with interpolated smooth motion

Stage 1: Coarse Video Generation with Contextual Interpolation
for $t_{vid} \in \{0, 1, \ldots, Total - 3\}$ **do** ▷ Iterate over each second (overlap occurs)
  Initialize noisy latent z_{n_t}
  for $t \leftarrow T$ to 0 **do** ▷ Backward diffusion process
    Denoise $z_{n_{t+1}}$ to obtain z_{n_t}
    if conditioning **then**
      Replace parts of z_{n_t} with $\mathcal{E}_{vae}(I_m) + \epsilon_t$, where I_m are guidance frames
      Update $z_{n_t} \leftarrow z'_{n_t}$ ▷ Refine latents using guided inpainting
    end if
  end for
  Decode z_{n_0} to obtain coarse video I_n
  Update I_m to include I_n for future iterations
end for
Generate a coarse video with sparse frames: $\{I_0, I_1, \ldots, I_{N-1}, I_N\}$
Stage 2: Fine Video Generation with Higher Frame Rate
for $t_{vid} \in \{0, 1, \ldots, Total - 1\}$ (smaller intervals) **do** ▷ Generate finer frames
  Use D_{adp} with keyframes and generated frames from Stage 1 as guidance
  Apply the same guided inpainting technique as in Stage 1
  Generate intermediate frames between keyframes and coarse frames
end for
Combine all frames to produce a high-frame-rate video
return Final video

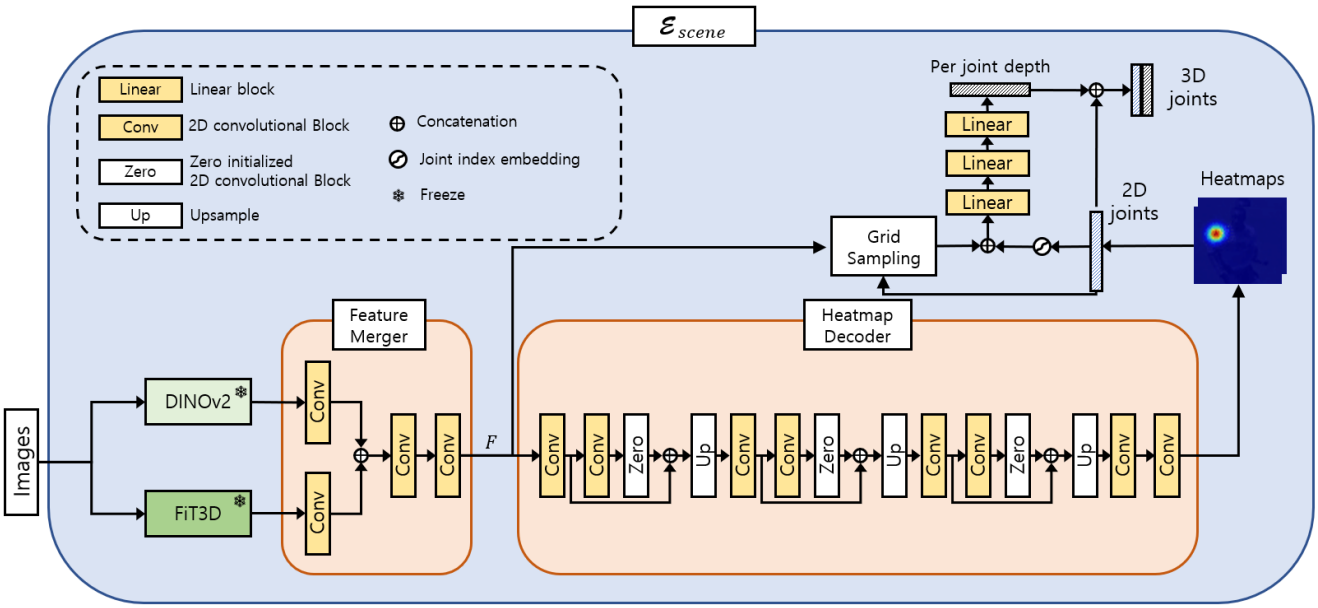

Figure 1. Architecture of scene specific joint estimator.

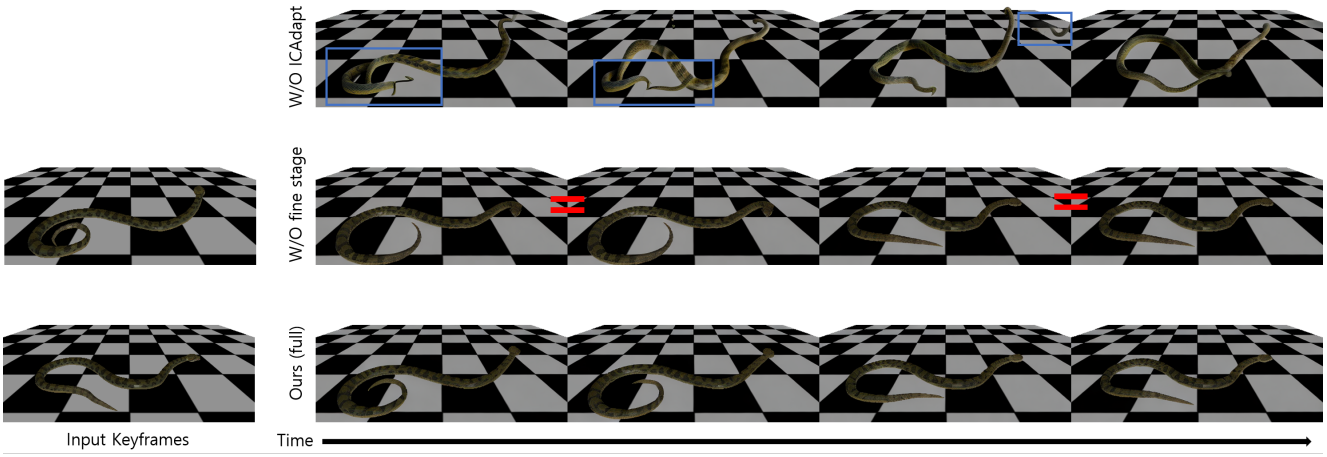

Figure 2. Additional results of the ablation study on video generation.

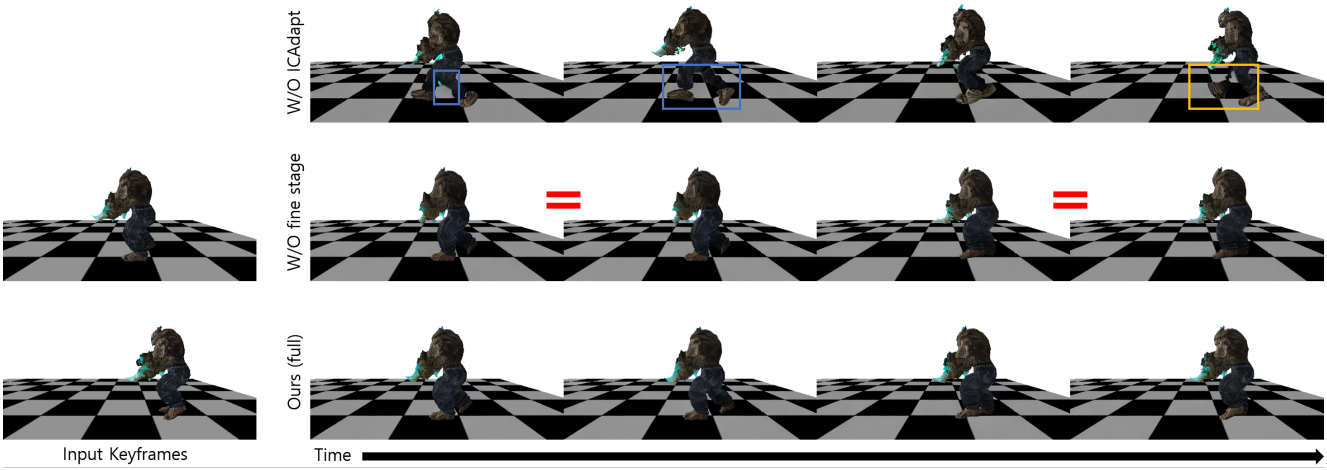

Figure 3. Additional results of the ablation study on video generation.

F, serves as input for both 2D joints and per-joint depth estimation. The heatmap decoder consists of convolutional blocks with residual layers [2]. Here, we use a zero-initialized convolutional block for the residual connection.

The heatmap decoder outputs heatmaps with the channel size equal to the number of joints, $H \in \mathbb{R}^{B\times N_j\times w_h\times h_h}$, where B is the size of the batch, N_j is the number of channels, which corresponds to the number of joints, and w_h and h_h correspond to width and height, respectively. The heatmap becomes the position of the estimated 2D joint positions, $J_{2D} \in \mathbb{R}^{B\times N_j\times 2}$. We concatenate joint index embedding, $e_{joint} \in \mathbb{R}^{B\times N_j\times 1}$, which is analogous to positional embedding, to the 2D joint to provide an additional cue for estimating the depth of each joint. Afterwards, as described in the main paper, we sample the feature $f_{x,y} \in \mathbb{R}^{B\times 1\times 1\times C}$ for each joint from $F \in \mathbb{R}^{B\times w_F\times h_F\times C}$. Here, w_F and h_F are the width and height of feature F, and C is the number of channels. Because $f_{x,y}$ is sampled for each joint, we concatenate and reshape these per-joint features to be $f_{processed} \in \mathbb{R}^{BN_j\times C}$. We reshape the 2D joint position and its joint index embedding to be per-joint values, $J_{processed2D} \in \mathbb{R}^{BN_j\times 3}$. We concatenate all of these sampled features, 2D joint positions, and index embeddings and pass them through the Depth MLP to estimate the per-joint depth values.

C. Additional Results

We provide additional quantitative values computed by the HL2Q metric with varying thresholds: 1/4, 1/2, 3/4, and full in Table 2. The "full" threshold corresponds to the original L2Q, while the "1/2" threshold indicates what we used as

Table 2. Quantitative results of H2Q variants and L2Q. Best scores are denoted in bold.

Character	Humanoid				Non-humanoid			
Methods	HL2Q (1/4)↓	HL2Q↓	HL2Q (3/4)↓	L2Q↓	HL2Q(1/4)↓	HL2Q↓	HL2Q (3/4)↓	L2Q↓
Ours	**0.0019**	**0.0015**	**0.0014**	**0.0011**	**0.0016**	**0.0019**	**0.0024**	**0.0021**
SinMDM	0.1526	0.0971	0.0636	0.0386	0.3664	0.2465	0.1979	0.1710
SinMDM*	0.1539	0.0981	0.0645	0.0390	0.3667	0.2467	0.1982	0.1713
TST	0.0028	0.0028	0.0069	0.0106	-	-	-	-
ERQ-QV	0.0031	0.0028	0.0064	0.0109	-	-	-	-

HL2Q in the main paper. Our method produced the best results across all variants of HL2Q and L2Q. For humanoid characters, HL2Q values decreased as the filtering threshold became larger, indicating that the rotations were more precise at the near-leaf joints. However, no distinct pattern or difference was observed for non-humanoid characters.

For additional qualitative results, we first present the results of further visual ablation studies on video generation. As shown in Figures 2 and 3, our method generated videos with smooth motion and no visible artifacts. Additionally, we visualized the results of Anymole on motion in-betweening in Figures 4 and 5.

References

[1] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. *arXiv preprint arXiv:2309.16588*, 2023. 1

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3

[3] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. 2021. 1

[4] Keunwoo Yu, Zheyuan Zhang, Fengyuan Hu, Shane Storks, and Joyce Chai. Eliciting in-context learning in vision-language models for videos through curated data distributional properties. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 20416–20431, 2024. 1

[5] Yuanwen Yue, Anurag Das, Francis Engelmann, Siyu Tang, and Jan Eric Lenssen. Improving 2d feature representations by 3d-aware fine-tuning. In *European Conference on Computer Vision*, pages 57–74. Springer, 2025. 1

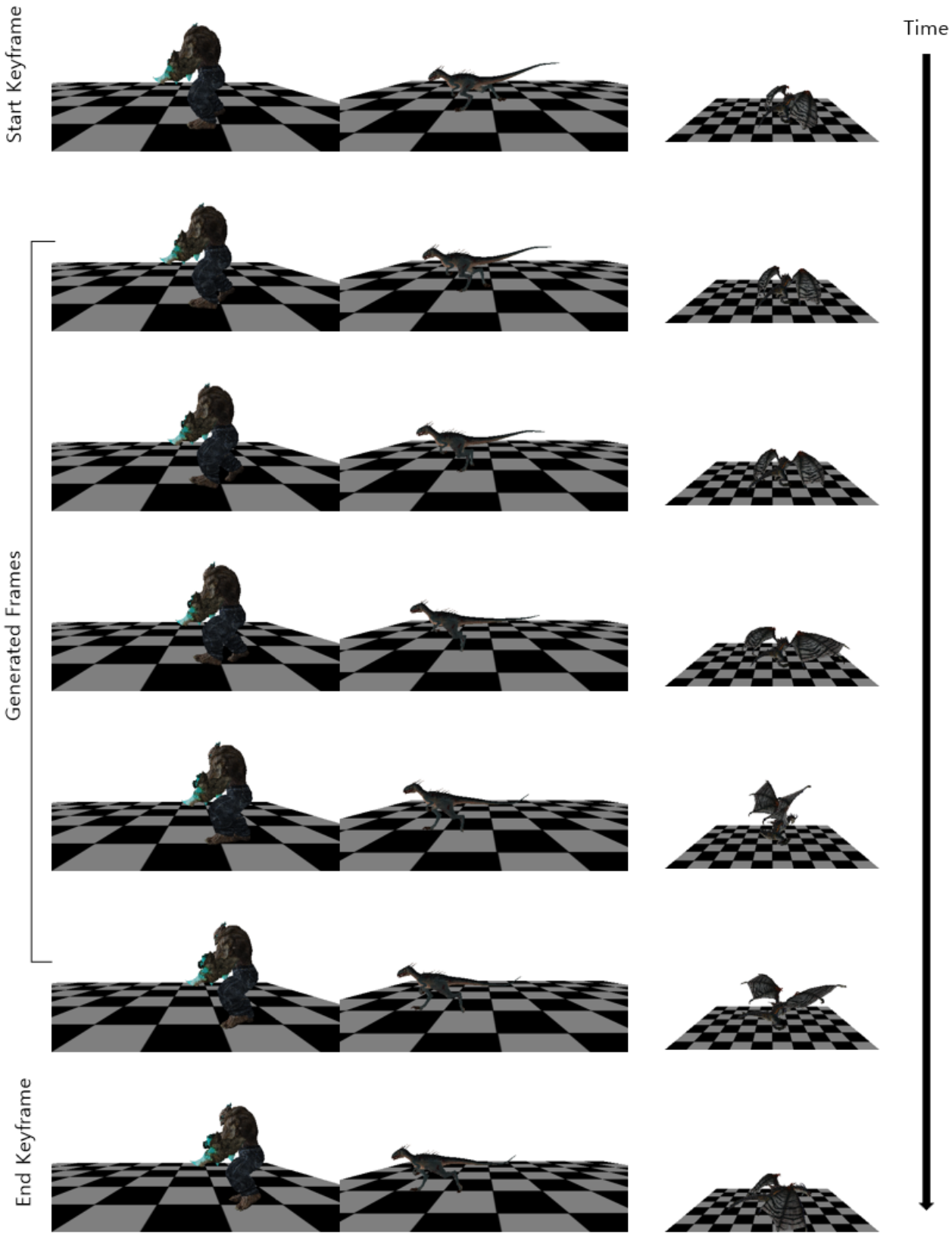

Figure 4. Visual results of generated in-between motion frames.

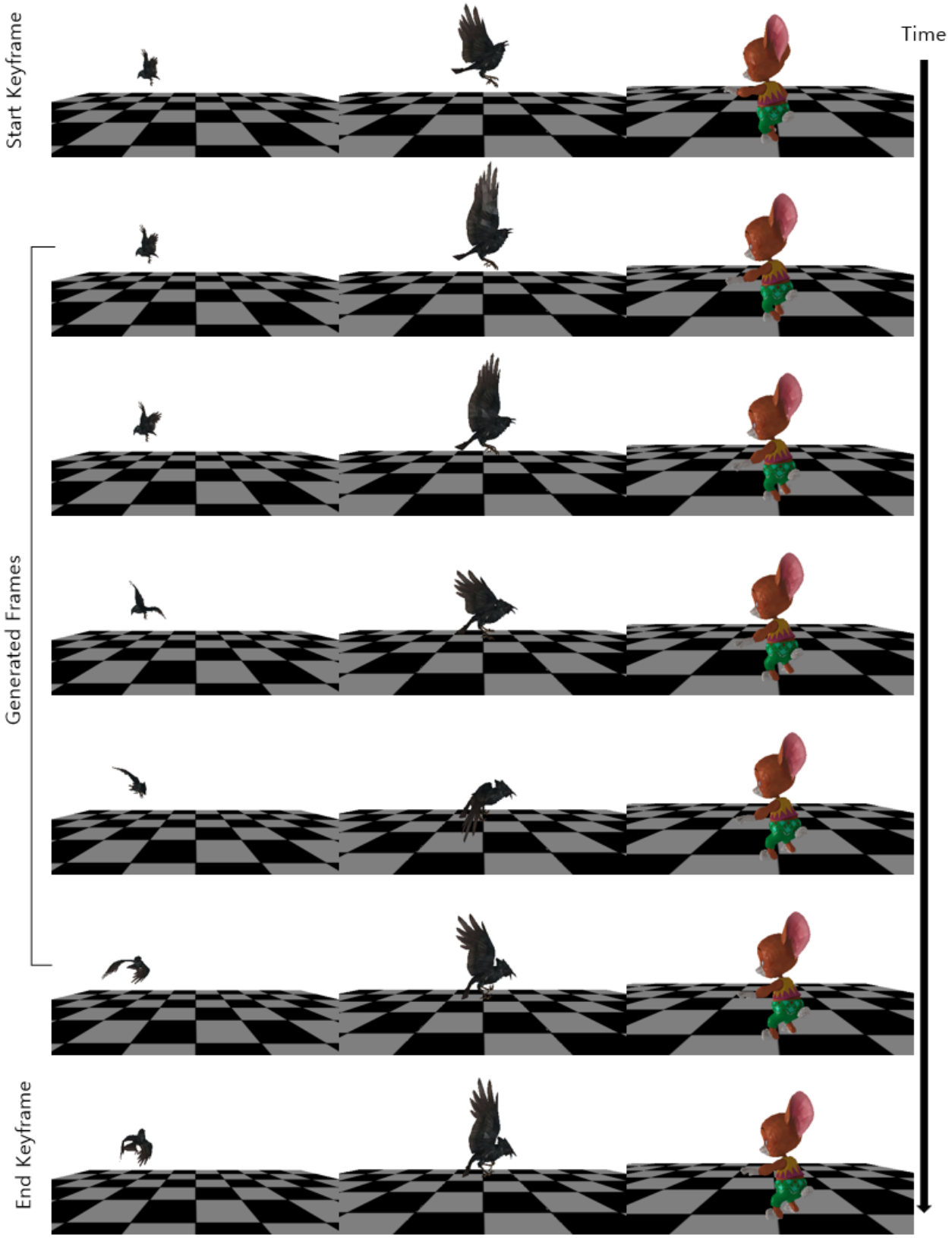

Figure 5. Visual results of generated in-between motion frames.