TwoSquared: 4D Generation from 2D Image Pairs

Supplementary Material

https://sangluisme.github.io/TwoSquared/

S0. Contents of Supplementary Materials

The supplementary zip folder contains the following:

- A PDF document providing additional details on our method, including training specifics, and extended visualizations of our results.
- A video folder showcasing animations of our results.
- A link to our project page https://sangluisme.github.io/TwoSquared/.

S1. Training and Inference Details

TwoSquared (ours). Given a pair of images, we use Hunyuan3D [50] for obtaining their 3D reconstruction. It first encodes each image into a latent space and then decodes each code into a Signed Distance Field (SDF) to generate a textureless mesh using marching cubes. Finally, a high-resolution texture is added to the generated mesh using the texture painting module conditioned on the input image. This whole process of obtaining textured meshes takes around 3 minutes per input image. Then we downsample the mesh to around 4000 vertices and use Diff3F [11] to compute the feature for each vertex. This procedure takes 5 minutes per frame. We feed the per-vertex feature to our correspondence-estimation block to get one-to-one correspondences, which takes around 1 minute. Then we train our Velocity Net for 8,000 epochs, which takes *less than 2 minutes*. Therefore, each pair of inputs takes roughly 17 minutes to train. Our velocity field is continuous in time, and inference can happen at arbitrary numbers of steps without retraining. The inferring time to generate a mesh takes *less than 1 second*. To infer the higher frame rates, we suggest using $T' < 2T$ to ensure satisfactory results.

DiffMorpher [64]: The morphing method takes two input images and *text descriptions* to obtain a morphed sequence between source and target, taking 2 minutes to obtain the intermediate images. Then, the generated images of each timestep are fed through the 3D reconstruction network. We remark that 3D reconstruction takes 3 minutes, making this technique particularly slow at inference. In total, *20 minutes* are needed to obtain deformed 3D shapes in 6 timesteps.

DreamMover [43]: The image morphing method takes 5 minutes to interpolate a pair of images. Same as DiffMorpher [64], this method leverages a pre-trained text-to-image diffusion model and also requires a text prompt. The resulting sequence of images is then fed to Hunyuan3D [50], which takes 3 minutes for every frame. In our experiments, it requires *23 minutes* in total for a sequence of 5 timesteps.

V2M4 [6]: Given a single monocular video, V2M4 can work in two ways: (i) using the entire video, where it processes each frame to obtain a 4D animation, or (ii) using only keyframes, where it processes the selected frames and then applies linear interpolation on vertex positions to generate the skipped frames in 3D. To compare our results with V2M4, we use the Consistent4D dataset [17], which consists of monocular videos of 32 frames, as well as selected sequences from 4D DRESS [53]. Following V2M4's preprocessing guidelines, we split the videos into 2D frames and then remove the background. Since their methodology is designed to work with different 3D generative backbones, for a fair comparison, we adopt Hunyuan3D [50], which they claim achieves the best results. Moreover, we chose VGGT [52] for dense stereo reconstruction, as suggested in their work. When V2M4 takes the entire video as input, their method requires approximately 1 hour and 40 minutes. In the keyframe setting with 7 keyframes, it requires about 50 minutes. Given that their methodology uses advanced processing techniques such as 3D mesh generation, dense stereo reconstruction, camera search, shape alignment and registration, and global texture map synthesis and optimization, they require a GPU with at least 40GB of memory. Our method (other than satisfying the requirement for running Hunyuan3D [50] and DIFF3F [11]) only needs 2 to 4 GB of memory.

S2. Multi-pairs Training and Inferring

Training. As shown in Fig. S.1, if more than two keyframes are available, our work can be easily extended to work with multiple pairs. We obtain 3D meshes and per-vertex features, the same as dealing with a single pair to form the training data. Further, we extract the one feature vector per mesh $\hat{\mathcal{F}}_i \in \mathbb{R}^{1\times f}$ from the feature $\mathcal{F}_i$ we obtained using [11]. Then we concatenate the feature vector to the point coordinates and feed to our deformation block., i.e., the training data we use has the form $\{\{\{P_0, \hat{\mathcal{F}}_0\}, \{P_1, \hat{\mathcal{F}}_1\}\}, \{\{P_1, \hat{\mathcal{F}}_1\}, \{P_2, \hat{\mathcal{F}}_2\}\}, \dots\}$.

Inferring. In a multi-pair case, there are two inference methods. The first is to evaluate the trained model per-pair, i.e., start with each generated high-resolution mesh and only deform within the current pair. That is, deform from S_0 until $t = 1$, start from S_1 again, deform until $t = 1$, etc. This approach best preserves the deformation. However, to most effectively stay geometrically consistent, our method also

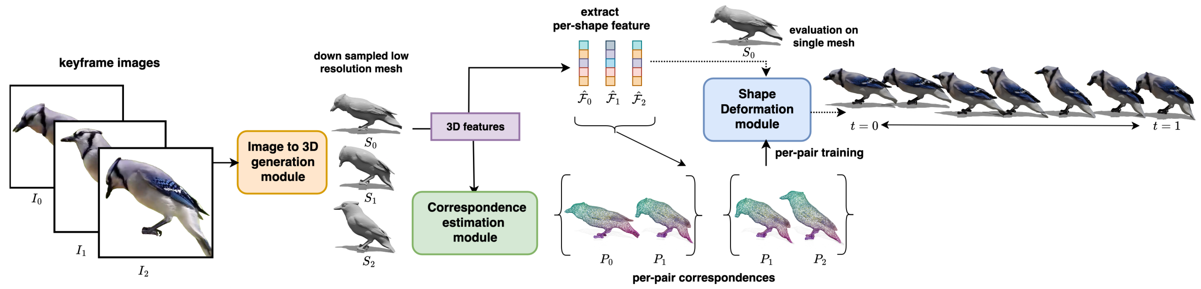
keyframe images
I_0
I_1
I_2
Image to 3D generation module
down sampled low resolution mesh
S_0
S_1
S_2
3D features
Correspondence estimation module
extract per-shape feature
$\hat{\mathcal{F}}_0$ $\hat{\mathcal{F}}_1$ $\hat{\mathcal{F}}_2$
P_0 P_1 P_1 P_2
per-pair correspondences
evaluation on single mesh
S_0
Shape Deformation module
per-pair training
$t=0$
$t=1$

Figure S.1. **Multi-pair Pipeline of TwoSquared**: Given multiple key frames input, we first generate a mesh for each image, then we compute pair-wise sparse correspondences between mesh pairs. During training, we iteratively train over pairs. Different from the single pair case, in the multi-pair case, we concatenate the extracted per-shape feature to the point coordinates and feed it to our deformation block.

supports inferring using the starting mesh S_0. From S_0 deform until $t = 1$ and directly using the deformed mesh to replace S_1 and deform until $t = 1$, hence the geometry and texture of the initial frame are carried consistently throughout the deformation.

S3. Correspondences Estimation Ablation

We propose to downsample the generated mesh, then estimate the correspondences on low-resolution meshes. In this section, we show that downsampled meshes not only accelerate the processing time but also lead to more accurate correspondence estimation and eventually lead to better results. We first show the correspondence quality on Fig. S.2. In Fig. S.3, we obtain the correspondences in high-resolution (HR) mesh (around $20k$ vertices) and sample from high resolution mesh to form the training point cloud, then we evaluate the model on the high resolution meshes to get the textured mesh deformation (first row), we also show the tracked point cloud on the second row. Next, we obtain correspondences on a low-resolution (LR) mesh (around $4k$ vertices) and sample from the low-resolution mesh to form the training point cloud. Then we infer directly on the high-resolution mesh to get the deformed, textured meshes (third row) and the tracked point cloud (fourth row). The results indicate that obtaining correspondences on a low-resolution mesh does not hinder the performance, but even improves the result. Intuitively, the local geometry produced by the generative backbone cannot be reliably matched between the two shapes, while we are more interested in a structural alignment.

S4. More Quantitative Evaluation

For sequences where the ground-truth intermediate shapes are not available, we evaluate the generated results using two geometric measurements. The first is the mesh area standard deviation SAσ, as introduced in the main paper. The second is the geodesic distance deviation $d_{\text{geo}}\sigma$. Since

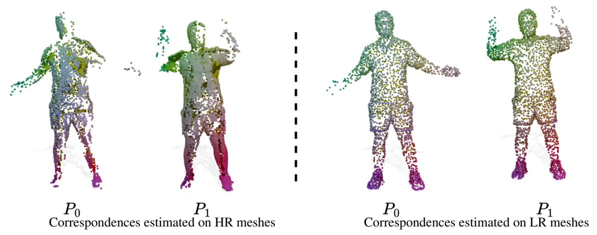
P_0
P_1
Correspondences estimated on HR meshes
P_0
P_1
Correspondences estimated on LR meshes

Figure S.2. **Correspondences visualization:** We visualize correspondences we obtained using a high resolution (HR) mesh (left) and a low resolution (LR) mesh. On high-resolution meshes, the vertices are too clustered, which hinders the functional map optimization step. While on low-resolution (LR) meshes, the vertices are sparser, and it is easier to match similar features.

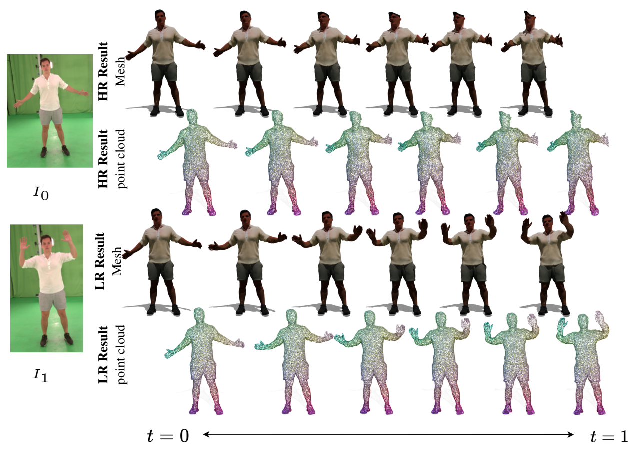

Figure S.3. **Mesh resolution ablation:** We show the qualitative results on correspondences estimated on different resolution meshes. The first two rows are results that we obtain from correspondence on high-resolution (HR) meshes. The last two rows are results that we obtain from correspondence on low-resolution (HR) meshes. Downsampling the mesh to estimate correspondences not only accelerates the process but also improves the correspondence quality that eventually leads to better results.

both methods generate mesh sequences with preserved vertex order, we compute the geodesic distance matrix for

Seq.	Method	time (per pair)	CD ($\times 10^3$) ↓	HD ($\times 10$) ↓	SAσ($\times 10$) ↓
Take19	HR mesh	$\sim$ 12 min	3.802	1.998	**0.196**
	LR mesh	$\sim$ 1 min	**1.451**	**1.996**	0.201
Take2	HR mesh	$\sim$ 12 min	1.359	0.398	0.129
	LR mesh	$\sim$ 1 min	**0.924**	**0.267**	**0.120**

Table S.1. **Quantitative ablation on mesh resolution:** We show the effect of estimating correspondences using different mesh resolutions. Estimating on a low-resolution mesh not only speeds up the process but also improves the results.

each mesh, measure the standard deviation at each vertex across the sequence, and then average over all vertices. The intuition is that under isometric deformations, geodesic distances on the mesh should perserved. Therefore, a smaller $d_{\text{geo}}\sigma$indicates better preservation of geodesic distances throughout the generated sequence. We report the value in Tab. S.2. Note that the results for V2M4 [6] are obtained using the full video input, whereas our results are produced using only sparse keyframes.

Seq.	V2M4 [6]		Ours	
	$d_{\text{geo}}\sigma$($\times 10$) ↓	SAσ($\times 10$) ↓	$d_{\text{geo}}\sigma$($\times 10$) ↓	SAσ($\times 10$) ↓
idling_elephant	**0.139**	1.020	0.156	**0.171**
flying_ironman	0.187	0.234	**0.115**	**0.159**
pecking_blue_jay	**0.143**	0.345	0.153	**0.216**
astronaut	**0.152**	1.272	0.158	**0.292**
triceratops	0.158	0.830	**0.029**	**0.037**
dancing_robot	**0.159**	**0.218**	0.212	0.679

Table S.2. **Quantitative ablation:** We report both the geodesic distance deviation $d_{\text{geo}}\sigma$) and the surface area deviation SAσ. Note that the results for V2M4 [6] are obtained using the full video input, whereas our results are produced using only sparse keyframes. Despite this reduced input, our method achieves superior surface area preservation and comparable performance in geodesic distance deviation.

S5. Morphed Images

We provide additional results to complement the experiments presented in the main paper. As shown in Fig. S.4, DiffMorpher [64] and DreamMover [43] successfully generate high-quality intermediate keyframes that are almost identical to the ground-truth images. We then use these three sequences as inputs to the 3D generation backbone to produce intermediate shapes, which are visualized in Fig. 3. While Fig. S.4 demonstrates that the image morphing methods can create visually plausible keyframes, it also highlights a key limitation: generating 4D sequences by independently conditioning on each keyframe often introduces artifacts and fails to maintain geometric and texture consistency.

We also present the morphed images corresponding to Fig. 7 in the main paper. As shown in Fig. S.5, DiffMorpher [64] failed to interpolate between the two different horse images, leading to subsequent failures in the 3D generation process. DreamMover [43] produces reasonable results by interpolating between images. However, since its task is designed to match the start and end images, it naturally interpolates not only the shape but also the changing texture. While this works well for objects that remain identical, it introduces texture inconsistency when handling two different objects of the same species, ultimately affecting the 3D generation output. This suggests that if 4D generation relies purely on image interpolation, the start and end images must represent the exact same object to maintain consistency. Moreover, a closer examination of DreamMover's results in Fig. S.5 reveals that the horse's movement is not temporally smooth across the morphed images, and the interpolation fails to maintain physical realism. The same situation happens to the cat example as shown in Fig. S.6. See in Fig. S.6 the cat leg's motion is not continuous and smooth.

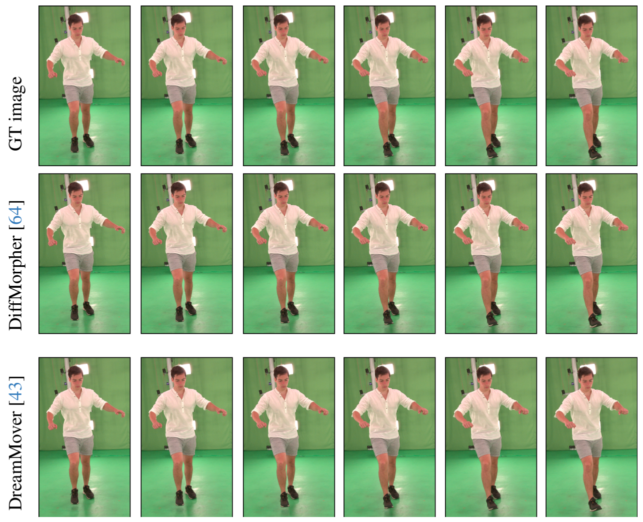

Figure S.4. **Morphing results for 4D-DRESS.** We show the ground truth images and the morphed images using DiffMorpher [64], DreamMover [43] in Fig. 3 in the main paper. These are images that are passed to the 3D generation backbone to generate intermediate shapes in Fig. 3.

S6. Additional Visualizations

In this section, we add more visualization results to demonstrate our method together with our estimated correspondences. Showcase that our method is robust to noisy and incomplete correspondence situations. We also demonstrate that our method can handle different types of objects, including some types that were previously solved using a template, such as human face Fig. S.8, animals Fig. S.6, multi-objects Fig. S.9 and random object Fig. S.7. Compared to the template-based method, our method is more faithful to the real-world details of the objects.

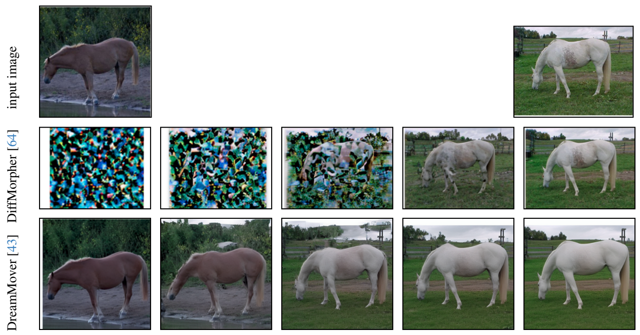

Figure S.5. **Morphing results for horse.** We show the ground truth images and the morphed images using DiffMorpher [64], DreamMover [43] in Fig. 7 in the main paper. These are images that are passed to the 3D generation backbone to generate intermediate shapes. As DiffMorpher heavily failed so the 3D generation steps also failed.

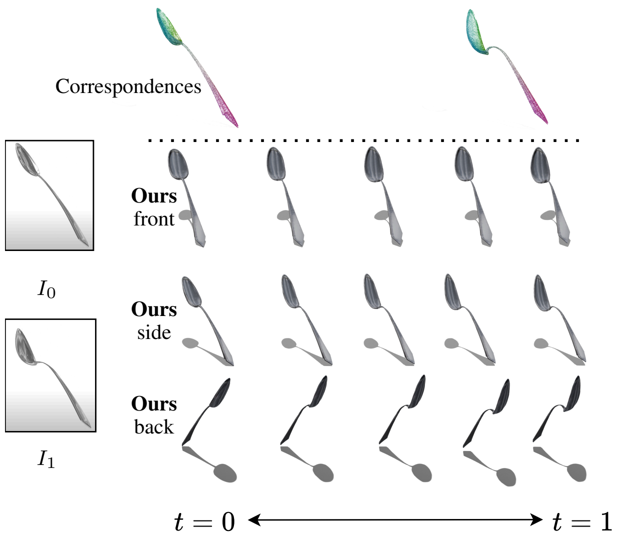

Figure S.7. **Visualization from different angle.** We show our 4D sequences by showing the deformed mesh sequence at different angles.

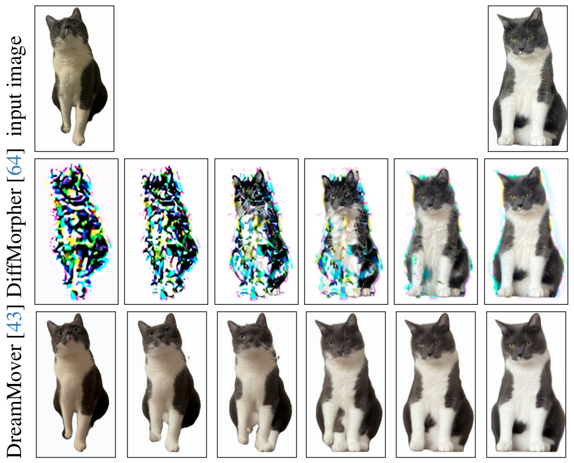

Figure S.6. **Morphing results for cat.** We show the ground truth images and the morphed images using DiffMorpher [64], DreamMover [43] in Fig. 8 in the main paper. These are images that are passed to the 3D generation backbone to generate intermediate shapes.

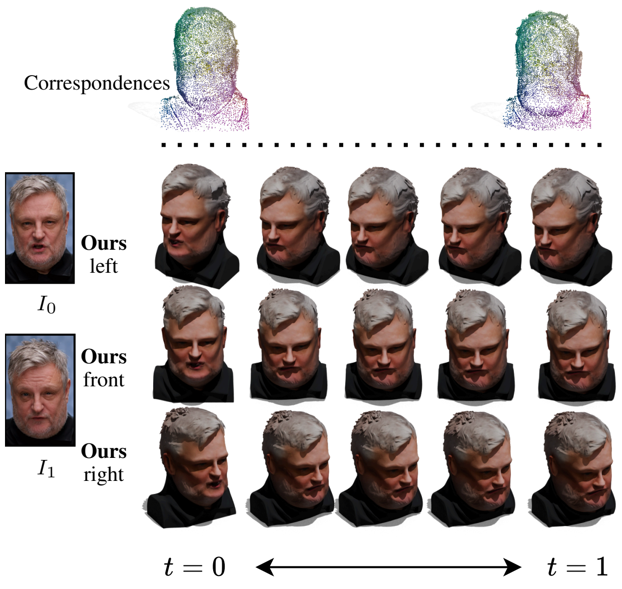

Figure S.8. **Visualization from different angle.** We show our 4D sequences by showing the deformed mesh sequence at different angles. The correspondences on the human face are noisy; however, our method still gets reasonable deformations.

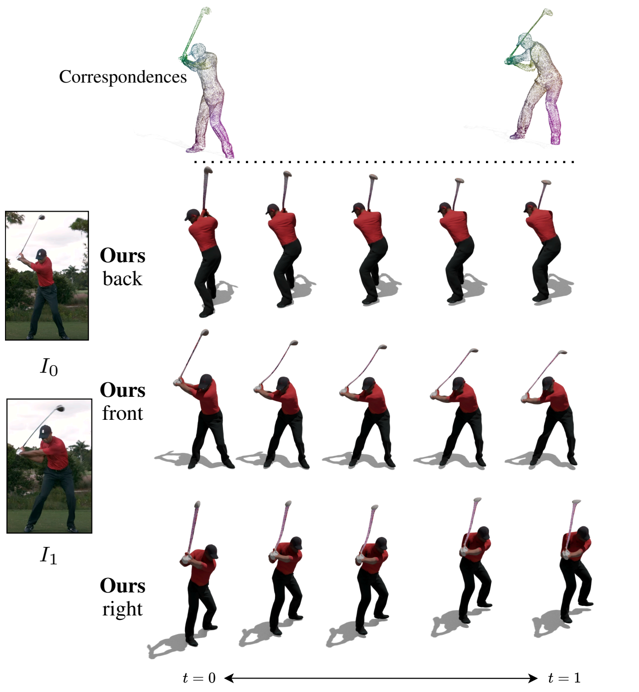

Figure S.9. **Visualization from different angle.** We show our 4D sequences by showing the deformed mesh sequence at different angles. The correspondence after the refinement is smooth and accurate.