

DNF: Unconditional 4D Generation with Dictionary-based Neural Fields

Supplementary Material

Abstract

In this supplementary file, we provide additional detail about our network architecture (Section 1), along with further elaboration on implementation details (Section 2). We also refer to reader to the supplemental video for further qualitative results of our DNF for 4D synthesis.

1. Network Architecture Details

1.1. Dictionary Decoder

With a pre-trained shape and motion MLP, we first conduct SVD to each linear layer of the MLP and compress the matrices $U \in \mathbb{R}^{J \times J}$, $\Sigma \in \mathbb{R}^{J \times F}$ and $V \in \mathbb{R}^{F \times F}$ to $U_k \in \mathbb{R}^{J \times k}$, $V_k \in \mathbb{R}^{F \times k}$ and $\Sigma_k \in \mathbb{R}^{k \times k}$. For each layer in the MLP, we then use two linear layers $N_U \in \mathbb{R}^{J \times k}$ and $N_V \in \mathbb{R}^{F \times k}$ to play the role as U and V , replacing the original linear layer $N \in \mathbb{R}^{J \times F}$. To extend the dictionary, we use another two linear layers $N_{U_r} \in \mathbb{R}^{J \times rk}$ and $N_{V_r} \in \mathbb{R}^{F \times rk}$ to learn the residual. During the fine-tuning, we freeze the parameters of N_U and N_V and optimize N_{U_r} , N_{V_r} and σ .

1.2. Shape Diffusion

For each shape feature, consisting of nine $(L + 1)$ vectors (one original latent code and eight coefficient vectors corresponding to eight MLP layers), we naturally split them into nine tokens. Each token is projected to the same dimension, set to 1280 in our implementation. The projected tokens are then summed with positional encoding vectors corresponding to their positions and fed into a transformer decoder. The transformer decoder, composed of 32 self-attention layers, predicts the denoised tokens.

1.3. Motion Diffusion

The overall architecture of motion diffusion is similar to that of shape diffusion but operates on a sequence of motions with t frames as input. The t motion features are concatenated along an additional time dimension, and a positional encoding is added in this dimension to ensure the correct order of the generated motions. Similarly, we project these $(t \times L)$ tokens to an inner dimension and add positional encoding vectors based on their token positions. In the motion diffusion model, each layer of the transformer decoder contains three attention layers:

1. A **spatial self-attention layer** to aggregate tokens within each frame,

2. A **condition cross-attention layer** to incorporate shape conditions, and
3. A **temporal self-attention layer** to aggregate tokens from the same position across different frames (e.g., motion codes of different frames).

In the sampling stage, our motion diffusion is capable of generating sequences longer than t frames through diffusion out-painting with a sliding window. We first generate a t -frame sequence, using the last k frames as the context, and let the diffusion model in-paint the following $(t - k)$ frames, and iteratively repeat this process.

To be more specific, given the motion features $\{\theta_m^k\}$ of the last k frames, we append $(t - k)$ vectors, $\{\theta_m^{(t-k)}\}$, which are initialized as random noise of the same size as the motion features. The goal is to denoise $\{\theta_m^{(t-k)}\}$ using the context provided by $\{\theta_m^k\}$.

For each denoising time step d , we aim to denoise $\{\theta_m^{(t-k)}\}_d$ into $\{\theta_m^{(t-k)}\}_{d-1}$. To achieve this, we first apply a d -step diffusion process to $\{\theta_m^k\}$, obtaining a noised version, $\{\theta_m^k\}_d$, which is then concatenated with $\{\theta_m^{(t-k)}\}_d$. Subsequently, our motion diffusion model denoises the combined vectors, producing $\{\theta_m^{(t-k)}\}_{d-1}$ using a DDIM sampler.

In practice, our diffusion model is trained to generate 6-frame motions and uses the last 2 frames as context to in-paint the subsequent 4 frames, thus extending the generated motion sequence.

2. Implementation Details

2.1. Data processing

Shape space. For each shape identity in the train dataset, we sample 200k points on the given mesh. We then calculate its grid SDF with resolution equals to 256, sampling 50k points uniformly within the unit bounding box and 150k random near-surface points within a distance of 0.02 from the surface of the shape.

Pose space. Following previous work [1], we sample 200k surface points on each shape identity and store the barycentric weights for each sampled point at the same time. Each point is then randomly disturbed with a small noise $\mathcal{N}(0, \Sigma^2)$ along the normal direction of the corresponding triangle in the mesh, with $\Sigma \in \mathbb{R}^3$ a diagonal covariance matrix with entries $\Sigma_{ii} = \sigma$. Then, for each t -th deforming shape for the identity, we compute corresponding points by using the same barycentric weights and the noise to sample the deformed mesh. In our experi-

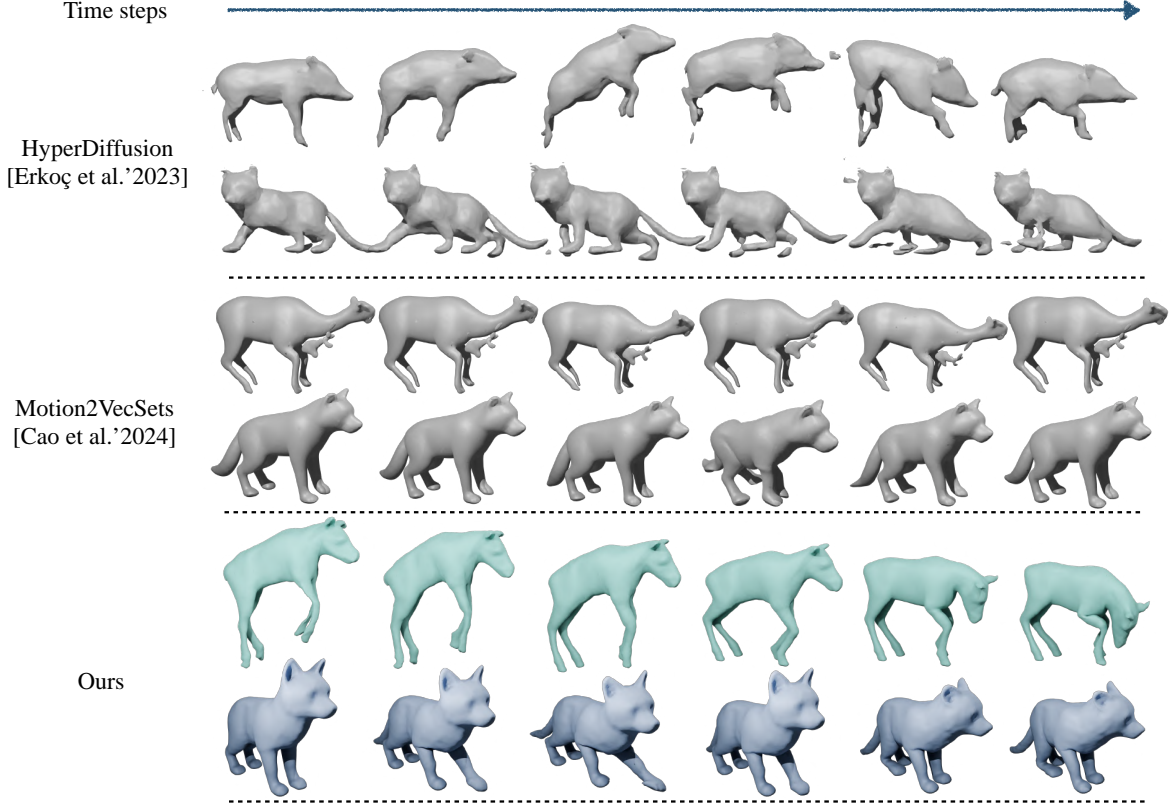


Figure 1. Our generations and its NN baseline generated samples.

ments, we sample 50% surface points ($\sigma = 0$) and 50% with $\sigma = 0.002$.

2.2. Data augmentation

When training the motion diffusion model, we apply data augmentation techniques to enhance the model’s robustness. Specifically, for each motion subsequence, we reverse the frame order to create a new training sample, which significantly improves the continuity of the generated motions. Additionally, we distribute the shape condition \mathcal{S} using a few-step diffusion process, defined as

$$\mathcal{S}_t = f(\mathcal{S}_{t-1}, \epsilon_t), \quad \text{for } t = 1, \dots, T,$$

where f represents the diffusion forward function, ϵ_t is the added noise, and T is the total number of steps. Here, we choose T randomly from the range $[0, 50]$.

3. Additional Results

3.1. Visual comparison with state of the art.

Fig. 1 compares a generation of DNF from Fig. 4 and its nearest neighbors retrieved from baseline generated samples.

References

- [1] Pablo Palafox, Aljaž Božič, Justus Thies, Matthias Nießner, and Angela Dai. Npms: Neural parametric models for 3d deformable shapes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12695–12705, 2021. 1