

Articulated Kinematics Distillation from Video Diffusion Models

Supplementary Material

1. Automatic Skinning Weight Computation

Assigning skinning weights to Gaussian kernels is not a straightforward task. For example, a kernel close to a bone actually should not be affected by it if the shortest segment from the kernel passes the outside region of the object. As a concrete example, Gaussian kernels on one foot of a human should not be influenced by the other foot. While using learnable weights could address this [2, 3, 5], it would undermine the degree-of-freedom (DoF) reduction achieved by the rigging system. To resolve ambiguities in weight assignment, we use a reference mesh that aligns with the geometry of 3DGS to define weights on the mesh surface, and then transfer these weights to the Gaussian kernels. This mesh is typically an output form from text-to-3D frameworks or can be generated using mesh extraction tools. For automatic weight computation on the mesh, we use the widely adopted auto-rigging system Pinocchio [1]. Pinocchio conceptualizes weight computation on a simply connected mesh as a heat diffusion process along the mesh surface, accounting for bone visibilities blocked by other surface parts. Specifically, for the bone b , the weight contribution vector \mathbf{w}^b on vertices is computed by solving the following Poisson equation:

$$-\Delta \mathbf{w}^b + \mathbf{H}^b \mathbf{w}^b = \mathbf{H}^b \mathbf{p}^b. \quad (1)$$

Here, Δ is the cotangent surface Laplacian operator. $\mathbf{p}_j^b = 1$ only if the closest bone to the vertex j is b . \mathbf{H}^b is a diagonal matrix with entries $\mathbf{H}_{jj}^b = c/d_j^2$ only if bone b is visible from vertex j within the mesh, where c is a user-defined constant and d_j is the distance from vertex j to bone b . The visibility is determined by whether the segment from the vertex to its closest point on the bone is completely enclosed within the mesh. This equation can be interpreted as follows: the bone first transfers heat to its visible vertex, and then the heat diffuses along the surface. The resulting heat distribution represents the weight field for that bone. We extend Pinocchio to support noisy mesh inputs that may have multiple components by manually setting the visibility of a connected component to its nearest bone as true if all bones are invisible from that component. This can prevent the system matrix of Eq. (1) from being singular and allows the outlier components to be controlled by their nearest bones.

By concatenating the weight contributions of each bone, we obtain the weight matrix $\mathbf{W} \in \mathbb{R}^{V \times B}$, where V is the number of vertices. Each row of \mathbf{W} represents the skinning weight vector at a given vertex. These weight vectors can be interpolated to arbitrary surface points on the mesh using

barycentric interpolation. Since Gaussian kernels generated during reconstruction are typically located near the geometric surface, we identify weight vectors of Gaussian kernels with their nearest surface points on the mesh.

2. SDS Gradient for V-Prediction Diffusion Models

The video diffusion model in our pipeline, CogVideoX-5B [8], is a v -prediction diffusion model [7], where the model predicts the so-called velocity instead of noise. The diffusion loss for training is

$$\mathcal{L}_{\text{Diff}}(\theta, \mathbf{z}, y) = \mathbb{E}_{t, \epsilon} \left[\frac{1}{1 - \alpha_t} \|\mathbf{z} - \hat{\mathbf{z}}\|_2^2 \right], \quad (2)$$

where \mathbf{z} is the latent code of a video, $\hat{\mathbf{z}} = \sqrt{\alpha_t} \mathbf{z}_t - \sqrt{1 - \alpha_t} \mathbf{v}_\theta(\mathbf{z}_t; t, y)$, $\mathbf{z}_t = \sqrt{\alpha_t} \mathbf{z} + \sqrt{1 - \alpha_t} \epsilon$, and \mathbf{v}_θ is a large transformer. Taking the derivative of $\mathcal{L}_{\text{Diff}}$ w.r.t. \mathbf{z} , we get

$$\begin{aligned} \nabla_{\mathbf{z}} \mathcal{L}_{\text{Diff}} &= \mathbb{E}_{t, \epsilon} \left[\frac{1}{1 - \alpha_t} \left(I - \frac{\partial \hat{\mathbf{z}}}{\partial \mathbf{z}} \right) (\mathbf{z} - \hat{\mathbf{z}}) \right] \\ &= \mathbb{E}_{t, \epsilon} \left[\frac{1}{1 - \alpha_t} \left(I - \frac{\partial \hat{\mathbf{z}}}{\partial \mathbf{z}_t} \frac{\partial \mathbf{z}_t}{\partial \mathbf{z}} - \frac{\partial \hat{\mathbf{z}}}{\partial \mathbf{v}_\theta} \frac{\partial \mathbf{v}_\theta}{\partial \mathbf{z}} \right) (\mathbf{z} - \hat{\mathbf{z}}) \right]. \end{aligned} \quad (3)$$

Here we omit the constant 2 that arise from the derivative of the square function for notation simplicity. Following the SDS gradients for U-Net-based diffusions [6], where terms involving the gradients of U-Nets are omitted, we similarly omit $\frac{\partial \hat{\mathbf{z}}}{\partial \mathbf{v}_\theta} \frac{\partial \mathbf{v}_\theta}{\partial \mathbf{z}}$:

$$\begin{aligned} \nabla_{\mathbf{z}} \mathcal{L}_{\text{SDS}} &= \mathbb{E}_{t, \epsilon} \left[\frac{1}{1 - \alpha_t} (1 - (\sqrt{\alpha_t})^2) (\mathbf{z} - \hat{\mathbf{z}}) \right] \\ &= \mathbb{E}_{t, \epsilon} [\mathbf{z} - \hat{\mathbf{z}}]. \end{aligned} \quad (4)$$

3. Shadow Casting

We can also cast shadows on the ground layer mentioned above to further indicate the spatial relationship between the object and the ground. When a rendering ray intersects with the ground at a point \mathbf{P} , the ground color is weighted by this heuristic shadow intensity: $s(\mathbf{P}) = 1 - s_{\text{max}} \exp(-\beta d(\mathbf{P}))$, where $d(\mathbf{P})$ is the vertical height from the ray-ground intersection point \mathbf{P} to the deformed asset, s_{max} is the maximum level of shadowing to apply, and β is the decay coefficient as the object height above increases. This shadowing approximates a distant, parallel light source positioned vertically above the ground, complemented by diffusive ambient lighting. Incorporating shadows in SDS optimization does not yield substantial improvements in motion synthesis quality, but it can increase the immersiveness when humans

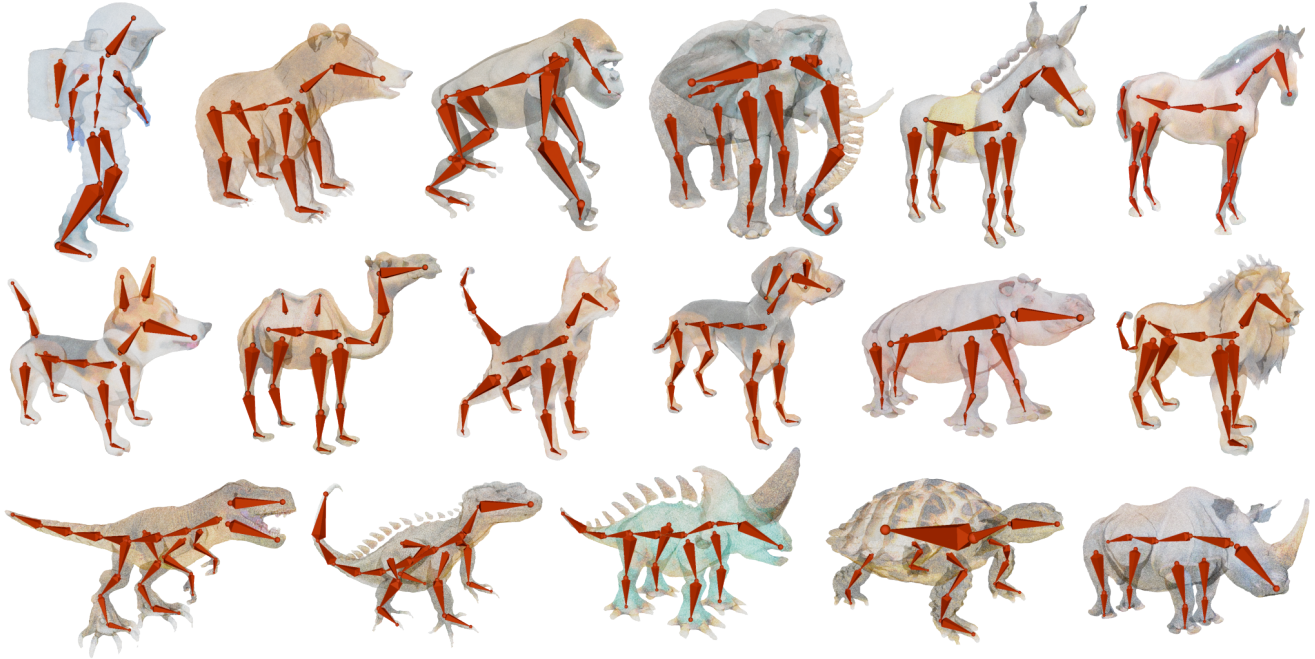


Figure 1. Gallery of skeleton systems from our experiments.

evaluate the generated videos. We leave a more thorough investigation of more advanced rendering techniques such as true global illumination to future work.

4. Implementation Details

Motion Synthesis The video SDS loss is evaluated with $\text{CFG} = 100$. The diffusion time t is sampled uniformly from $[t_{\text{start}}, t_{\text{end}}]$, where $t_{\text{start}} = 0.02$ and t_{end} decrease linearly from 0.98 to 0.5 over the first 5000 optimization iterations. In the training loss, we set $\lambda_1 = 2 \times 10^5$ and $\lambda_2 = 10^7$. A total of 10,000 optimization iterations are performed.

Motion Tracking We use Warp [4] to simulate skeletons as articulated rigid bodies while optimizing the tracking loss and applying gradient clipping with PyTorch. A chunk of simulation substeps is wrapped as a differentiable PyTorch layer using `torch.autograd.Function`. During the forward pass, data from the PyTorch scope is transferred to Warp, and a Warp gradient tape is initialized and stored to capture the computational graph within the Warp scope during forward simulation. In the backward pass, gradients received from the PyTorch scope are transferred to the Warp scope, and the gradient tape backpropagates the gradients in reverse time. Once the required gradients for the layer’s inputs are computed, gradient clipping is applied to them before transferring them back to the PyTorch scope. To ensure that assets remain standing on their own, we introduce

a virtual penalty force to keep the root bones of their articulation trees aligned with their initial upward directions. During training, we set $\lambda_3 = 0.2$ and perform 200 optimization iterations.

5. Skeleton Gallery

Here, we present a gallery of skeleton systems utilized in our experiments, as shown in Fig. 1. All skeletons are manually crafted in Blender. Mesh representations of assets are initially imported into Blender, followed by the embedding of bones based on their biokinematic structure.

6. Additional Motion Diversity Experiments with Artist-Rigged Characters

Our method supports artist-rigged characters with predefined skeleton embeddings and skinning weights. To use these assets, we convert them to 3DGS and transfer the predefined skinning weights to Gaussian kernels. The SDS optimization process remains unchanged. In Fig. 2, we show a fox jumping, and a girl dancing and punching. These assets are all artist-rigged: the fox is from Truebones¹ and the girl is from Mixamo². We note that additional consistent textural descriptions in prompts are crucial for human motion synthesis. On the other hand, we find that whether our method can produce diverse motions largely depends

¹<https://truebones.gumroad.com/1/sk2MC>

²<https://www.mixamo.com/>

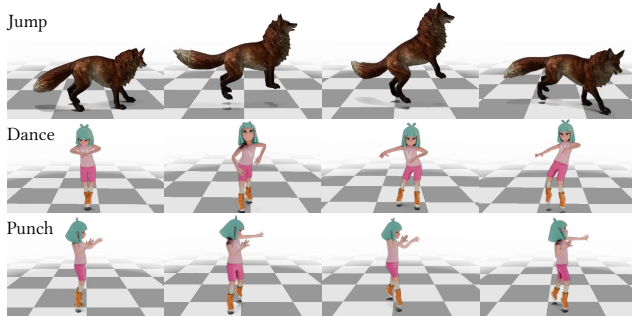


Figure 2. Additional Motion Diversity Experiments

on the ability of the video model to synthesize desired full-body motions. It is difficult for our method to generate animals sitting since CogVideoX struggles with transitions from standing to sitting, as well as precise human motion control such as cartwheeling and raising hands. Additionally, fine-grained motions, such as hand pose variations, are difficult to capture.

References

- [1] Ilya Baran and Jovan Popović. Automatic rigging and animation of 3d characters. *ACM Transactions on graphics (TOG)*, 26(3):72–es, 2007. 1
- [2] Shoukang Hu, Tao Hu, and Ziwei Liu. Gauhuman: Articulated gaussian splatting from monocular human videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20418–20431, 2024. 1
- [3] Muhammed Kocabas, Jen-Hao Rick Chang, James Gabriel, Oncel Tuzel, and Anurag Ranjan. Hugs: Human gaussian splats. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 505–515, 2024. 1
- [4] Miles Macklin. Warp: A high-performance python framework for gpu simulation and graphics. <https://github.com/nvidia/warp>, 2022. NVIDIA GPU Technology Conference (GTC). 2
- [5] Arthur Moreau, Jifei Song, Helisa Dharmo, Richard Shaw, Yiren Zhou, and Eduardo Pérez-Pellitero. Human gaussian splatting: Real-time rendering of animatable avatars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 788–798, 2024. 1
- [6] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *The Eleventh International Conference on Learning Representations*, 2023. 1
- [7] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022. 1
- [8] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. Cogvideox: Text-to-video diffusion models with an expert transformer. *arXiv preprint arXiv:2408.06072*, 2024. 1