

PhysSkin: Real-Time and Generalizable Physics-Based Animation via Self-Supervised Neural Skinning Supplementary Material

Yuanhang Lei¹ Tao Cheng¹ Xingxuan Li¹ Boming Zhao¹
Siyuan Huang² Ruizhen Hu³ Peter Yichen Chen⁴ Hujun Bao¹ Zhaopeng Cui^{1†}
¹State Key Laboratory of CAD&CG, Zhejiang University ²BIGAI
³Shenzhen University ⁴University of British Columbia

A. Background

A.1. Continuum Mechanics

The object deformation is governed by the principles of continuum mechanics. Constitutive models describe how materials respond to mechanical loading, defining the stress–strain relationships that determine the governing equations. In our framework, we consider two representative models—linear elasticity and Neo-Hookean elasticity—which together enable a smooth transition from small-strain to large-strain behaviors during training.

Linear Elasticity. The simplest practical constitutive model is linear elasticity, which assumes small deformations and linear stress–strain relationships. The corresponding strain energy density function is defined as:

$$\psi(\mathbf{F}) = \mu, \epsilon : \epsilon + \frac{\lambda}{2}, \text{tr}^2(\epsilon), \quad (1)$$

where $\epsilon = \frac{1}{2}(\mathbf{F}^T + \mathbf{F}) - \mathbf{I}$ is the small-strain tensor, and μ, λ are the Lamé coefficients related to the material parameters of Young’s modulus E and Poisson’s ratio ν_p as:

$$\mu = \frac{E}{2(1 + \nu_p)}, \quad \lambda = \frac{E\nu_p}{(1 + \nu_p)(1 - 2\nu_p)}, \quad (2)$$

The first term represents shear deformation energy, while the second accounts for volumetric deformation.

Neo-Hookean Elasticity. To capture nonlinear large-deformation effects, we further adopt the Neo-Hookean constitutive model, whose strain energy density function is defined as:

$$\psi(\mathbf{F}) = \frac{\mu}{2} \sum_i [(\mathbf{F}^T \mathbf{F})_{ii} - 1] - \mu \log(J) + \frac{\lambda}{2} \log^2(J), \quad (3)$$

$$\mathbf{P}(\mathbf{F}) = \frac{\partial \psi}{\partial \mathbf{F}} = \mu(\mathbf{F} - \mathbf{F}^T) + \lambda \log(J) \mathbf{F}^{-T},$$

where ψ denotes the hyperelastic strain energy density, \mathbf{P} is the first Piola–Kirchhoff stress, \mathbf{F} is the deformation gradient encoding local stretch, rotation, and shear, and

$J = \det(\mathbf{F})$ is the Jacobian determinant representing volumetric change.

Progressive Model Interpolation. During the physics-informed learning process, the potential energy term derived from $\psi(\mathbf{F})$ serves as the loss function. To improve numerical stability and avoid divergence at early training stages, we gradually interpolate between the linear and Neo-Hookean models:

$$\psi_t(\mathbf{F}) = (1 - \alpha_t) \cdot \psi_{\text{linear}}(\mathbf{F}) + \alpha_t \cdot \psi_{\text{neo}}(\mathbf{F}), \quad \alpha_t \in [0, 1], \quad (4)$$

where α_t increases smoothly over training iterations. This progressive transition stabilizes the learning of deformation gradients while preserving physical consistency as the model adapts to larger nonlinearity.

A.2. Full-Space FEM Simulator

Our method focuses on real-time subspace physics-based animation, where the underlying dynamics follow the implicit time integration scheme of the Finite Element Method (FEM). Although simulation is performed in a reduced subspace for efficiency, the formulation of motion and energy remains consistent with the standard FEM discretization. Below, we summarize the core equations of the FEM-based implicit integrator that serve as the foundation for our simulator.

Implicit Time Integration. We denote the nodal positions and velocities at the i -th time step as $\mathbf{x}_i \in \mathbb{R}^{3n}$ and $\mathbf{v}_i \in \mathbb{R}^{3n}$, respectively. Given a time step size h , the implicit update scheme is written as:

$$\begin{aligned} \mathbf{x}_{i+1} &= \mathbf{x}_i + h \mathbf{v}_{i+1}, \\ \mathbf{v}_{i+1} &= \mathbf{v}_i + h \mathbf{M}^{-1} [\mathbf{f}_{\text{int}}(\mathbf{x}_{i+1}) + \mathbf{f}_{\text{ext}}], \end{aligned} \quad (5)$$

where $\mathbf{M} \in \mathbb{R}^{3n \times 3n}$ is the lumped mass matrix, \mathbf{f}_{int} denotes the internal elastic force, and \mathbf{f}_{ext} represents external forces such as gravity or user interaction. Substituting \mathbf{v}_{i+1} into \mathbf{x}_{i+1} leads to a nonlinear system:

$$\frac{1}{h^2} \mathbf{M}(\mathbf{x}_{i+1} - \mathbf{y}_i) - \mathbf{f}_{\text{int}}(\mathbf{x}_{i+1}) = 0, \quad (6)$$

where $\mathbf{y}_i = \mathbf{x}_i + h\mathbf{v}_i + h^2\mathbf{M}^{-1}\mathbf{f}_{\text{ext}}$ is evaluated at the beginning of each step. For clarity, we drop the time index and write:

$$\frac{1}{h^2}\mathbf{M}(\mathbf{x} - \mathbf{y}) - \mathbf{f}_{\text{int}}(\mathbf{x}) = 0, \quad (7)$$

Energy Formulation. Eq. (7) can be equivalently derived by minimizing:

$$g(\mathbf{x}) = \frac{1}{2h^2}(\mathbf{x} - \mathbf{y})^\top \mathbf{M}(\mathbf{x} - \mathbf{y}) + E(\mathbf{x}), \quad (8)$$

where $E(\mathbf{x})$ is the potential energy defined by the chosen constitutive model (e.g., linear or Neo-Hookean elasticity). The internal force is given by $\mathbf{f}_{\text{int}} = -\nabla E(\mathbf{x})$, and the gradient of Eq. (8) is:

$$\nabla g(\mathbf{x}) = \frac{1}{h^2}\mathbf{M}(\mathbf{x} - \mathbf{y}) + \nabla E(\mathbf{x}), \quad (9)$$

Numerical Solution. At each time step, Eq. (9) is solved using Newton’s method. Let \mathbf{x}^k denote the current estimate. The update rule is formulated as:

$$\nabla^2 g(\mathbf{x}^k), \Delta \mathbf{x} = -\nabla g(\mathbf{x}^k), \quad \mathbf{x}^{k+1} = \mathbf{x}^k + \Delta \mathbf{x}, \quad (10)$$

Here, $\nabla^2 g(\mathbf{x})$ corresponds to the effective stiffness matrix and may be regularized to maintain positive definiteness. For large-scale problems, we can leverage the sparsity of \mathbf{M} and use preconditioned conjugate gradient (PCG) solvers to efficiently solve the linearized system.

Subspace Integration. Our skinning fields autoencoder outputs the corresponding object-specific skinning eigenmodes [1], which are used for subspace physics-based animation. The above FEM-style implicit integration forms the physical foundation of our subspace physics-based animation system.

A.3. 3D Gaussian Representation

In the main paper, we also demonstrate that our method can seamlessly animate static 3D Gaussian Splatting (3DGS) [9] models. 3DGS employs a substantial number of explicit 3D Gaussians to represent a static 3D scene. Each 3D Gaussian G is defined by a full covariance matrix Σ and a center location μ :

$$G(x) = e^{-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)}. \quad (11)$$

For differentiable rendering optimization, 3DGS decomposes Σ into scaling matrix S and rotation matrix R : $\Sigma = RSS^\top R^\top$, where S and R are stored by a 3D vector s and a quaternion q respectively. To project these 3D Gaussians to 2D image, given a viewing transformation W , we obtain the 2D covariance matrix Σ' and 2D center location μ' as:

$$\Sigma' = JW\Sigma W^\top J^\top, \quad \mu' = JW\mu, \quad (12)$$

where J is the Jacobian of the affine approximation of the projective transformation. Then we can use the neural point-based α -blending to render the color C of each pixel with N ordered 3D Gaussians:

$$C = \sum_{i \in N} T_i c_i \alpha_i, \quad (13)$$

where T_i, α_i are calculated as:

$$T_i = \prod_{j=1}^{i-1} (1 - \alpha_j), \quad \alpha_i = o_i e^{-\frac{1}{2}(x-\mu')^\top \Sigma'^{-1}(x-\mu')}. \quad (14)$$

Here, o_i is the opacity of the 3D Gaussian. Therefore, the 3D scene can be represented by the parameter set P of 3D Gaussians, where $P = \{G_i : \mu_i, q_i, s_i, c_i, o_i\}$.

B. Details on Datasets

RigNet Dataset. We adopt the RigNet [13] dataset (ModelsResource-RigNetv1), which contains 2,703 rigged 3D models in FBX format. Following the official split, we use 80% of the models (2,163) for training, 10% (270) for validation, and 10% (270) for testing. The dataset provides detailed rigging information, including joint hierarchies, skinning weights, remeshed meshes, and precomputed geometric attributes. In our work, we only utilize the mesh geometry in OBJ format for both training and evaluation. The remaining auxiliary data (e.g., rigging information, remeshed meshes, and precomputed supervision) are used solely for fine-tuning the baseline models. Our training, validation, and testing splits are kept identical to those of the original RigNet dataset to ensure fair comparison.

ShapeNet Dataset. We also utilize the ShapeNetCore [2] dataset, a richly annotated subset of ShapeNet [2] containing approximately 51,300 unique 3D models across 55 common object categories. Each model in ShapeNetCore is linked to a corresponding synset in WordNet 3.0. In our experiments, we use 70% of the models for training and the remaining 30% for testing. Since the ShapeNet dataset does not provide annotated skinning annotations, we do not fine-tune any of the feed-forward neural skinning baselines on this dataset.

Details on Data Process. We first convert all mesh files in OBJ format into watertight meshes to prevent surface holes or disconnected components that may cause failures during volumetric sampling or lead to degraded rendering quality. Specifically, we adopt the open-source implementation of the robust watertight manifold generation method proposed by Huang et al. [7, 8], which performs mesh repair and closure to ensure that each mesh forms a closed manifold with consistent face orientations. This preprocessing step guarantees well-defined inside–outside regions and improves the robustness and accuracy of subsequent geometric analysis. For surface point sampling, we employ the Sharp Edge Sampling (SES) [4] method to better capture fine-grained geometric details of the objects, for more implementation details

regarding the surface point sampling process, please refer to the Dora paper [4]. For interior point sampling, thanks to the watertight property of the meshes, our ray-tracing-based sampling procedure [11], as described in the main paper, produces reliable and uniformly distributed samples within the enclosed volume. Finally, we use the transformer-based point cloud encoder Michelangelo [14] to extract a highly compressed latent set \mathbf{F}_s , since our method does not require further optimization of the 3D shape encoder, we directly obtain the latent set for each object from the pretrained Michelangelo model, this design effectively eliminates the computational overhead of shape encoding during the overall model optimization.

C. Details on Baseline Methods

We compare our method with the following baselines: Simplicits [10], RigNet [13], M-I-A [6], Anymate [5], and Puppeteer [12] on ShapeNet [2] and RigNet [13] datasets. We will introduce the specific details of the experimental comparisons. Simplicits train a separate neural network for each object to output skinning weights through self-supervised learning. Other baselines are current state-of-the-art supervised neural rigging and skinning methods that output 3D object joints, bones, and skinning weights in a feed-forward manner.

Simplicits. Simplicits train a separate neural network for each object to output skinning weights through self-supervised learning. For each training batch, we randomly sample 1000 cubature points from the candidate point set for potential energy computation, following the same strategy as previous work [3, 10]. In each batch, we randomly sample 1024 subspace coordinates \mathbf{z} for training. The learning rate is linearly increased to $5e-4$ within the first 1% of the 30K training iterations (warm-up), and then gradually decreased using the cosine decay schedule until reaching the minimum value of $5e-5$. The hyperparameter settings are identical to those of our model for a fair comparison.

RigNet. RigNet provides pretrained model trained on the RigNet [13] dataset, making additional fine-tuning unnecessary. Since the ShapeNet [2] dataset does not contain annotated skinning data, none of the feed-forward neural skinning baselines are fine-tuned on ShapeNet.

M-I-A. We fine-tune the pretrained M-I-A model on the RigNet training data for 300 epochs, which is consistent with the number of training epochs used for our model.

Anymate. For Anymate, we fine-tune the pretrained models on the RigNet dataset for 300 epochs for the joint model, 10 epochs for the connection model, and 130 epochs for the skinning model. We observe that excessive fine-tuning leads to overfitting and degrades the model’s generalization ability. Therefore, we report the best-performing fine-tuned version in the main paper.

Puppeteer. Since Puppeteer only provides inference codes

for the pretrained model and does not include the training codes, we do not perform any fine-tuning for this model.

D. Rotation Equivariant of Skinning Eigenmodes

Using skinning eigenmodes [1] as the subspace basis naturally exhibits *rotation equivariance*. In contrast to traditional affine subspace formulations that may not span global rotations, skinning eigenmodes are both rotation spanning and closed under rotations.

Formally, given a rotation $R \in SO(d)$, the output of linear blend skinning under a global rotation is equivalent to applying the same rotation to all input transformations:

$$R \sum_{b=1}^m w_{ib} T_b X_i = \sum_{b=1}^m w_{ib} (R T_b) X_i, \quad (15)$$

which ensures that any rotation of the output can be produced by an equivalent rotation of the input transformations. This property guarantees that the resulting subspace simulation is rotation equivariant.

Consequently, the skinning eigenmodes maintain rotational consistency within the learned deformation subspace. By default, the first skinning eigenmode is a constant field, corresponding to uniform rigid motion.

Notation.

- $R \in SO(d)$: a rotation matrix in d -dimensional space, representing a rigid rotation.
- m : the number of skinning transformations (typically corresponding to the number of bones or joints).
- b : index of a bone or transformation in the rig.
- i : index of a vertex on the mesh.
- w_{ib} : skinning weight that defines the influence of the b -th bone on vertex i .
- $T_b \in \mathbb{R}^{d \times d}$: transformation (e.g., rotation + translation) associated with the b -th bone.
- $X_i \in \mathbb{R}^d$: rest-pose position of vertex i .

Eq. (15) demonstrates that the skinning subspace is closed under rotations. FastCodey [1] further proves that a subspace simulation is rotation equivariant if and only if the subspace is closed under rotations.

E. More Skinning Fields Results

In Fig. A, Fig. B, Fig. C and Fig. D, we present more qualitative results on unseen 3D shapes from the RigNet [13] and ShapeNet [2] test set, further demonstrating the strong generalization ability of our method across diverse 3D shapes. Moreover, our framework exhibits a clear advantage in memory efficiency: its memory usage remains constant regardless of the number of input shapes, while methods such as Simplicits [10] require memory that scales linearly with the dataset size.

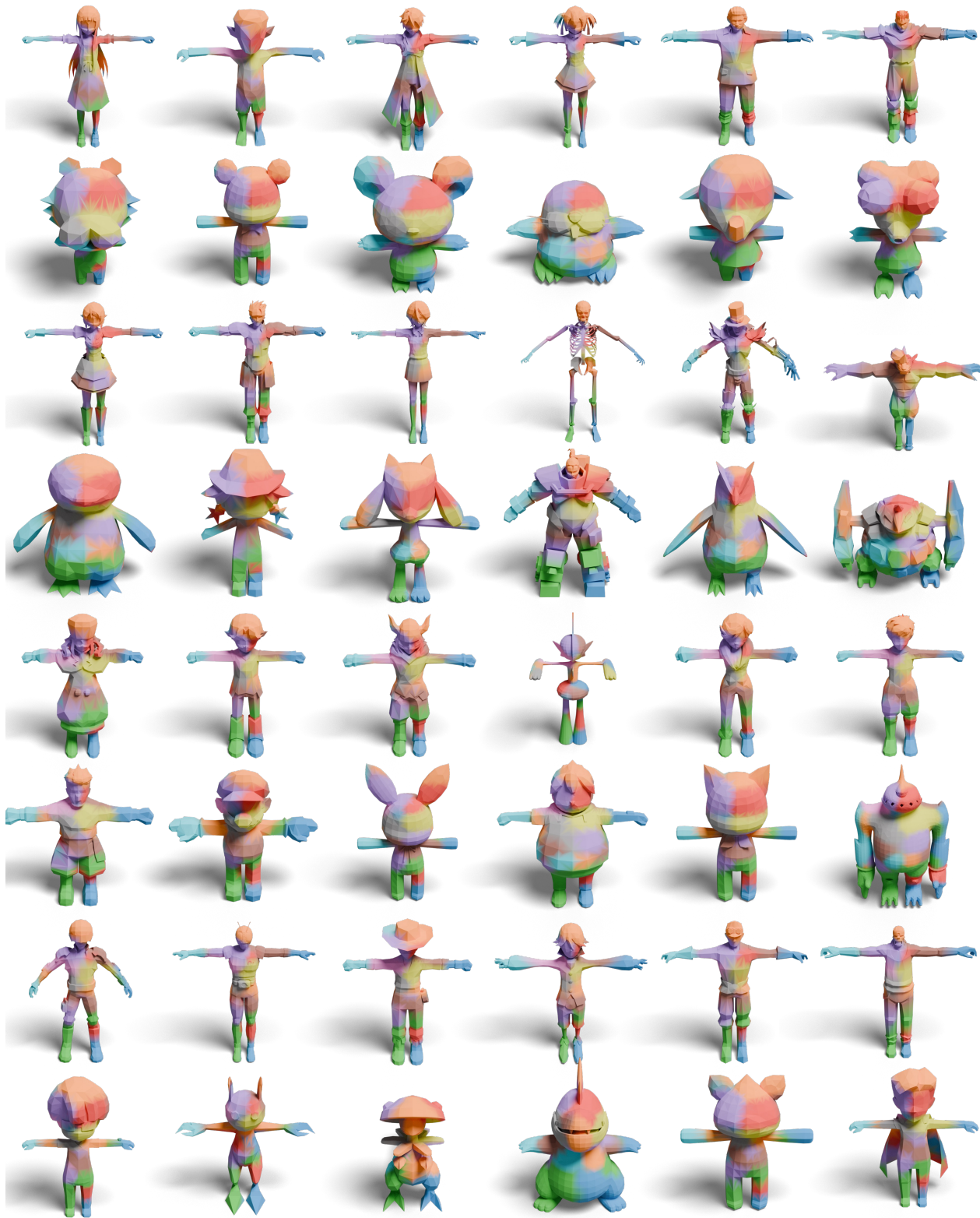


Figure A. Qualitative results on RigNet [13] test set. We visualize blended skinning fields to demonstrate our method’s generalization to diverse 3D shapes.

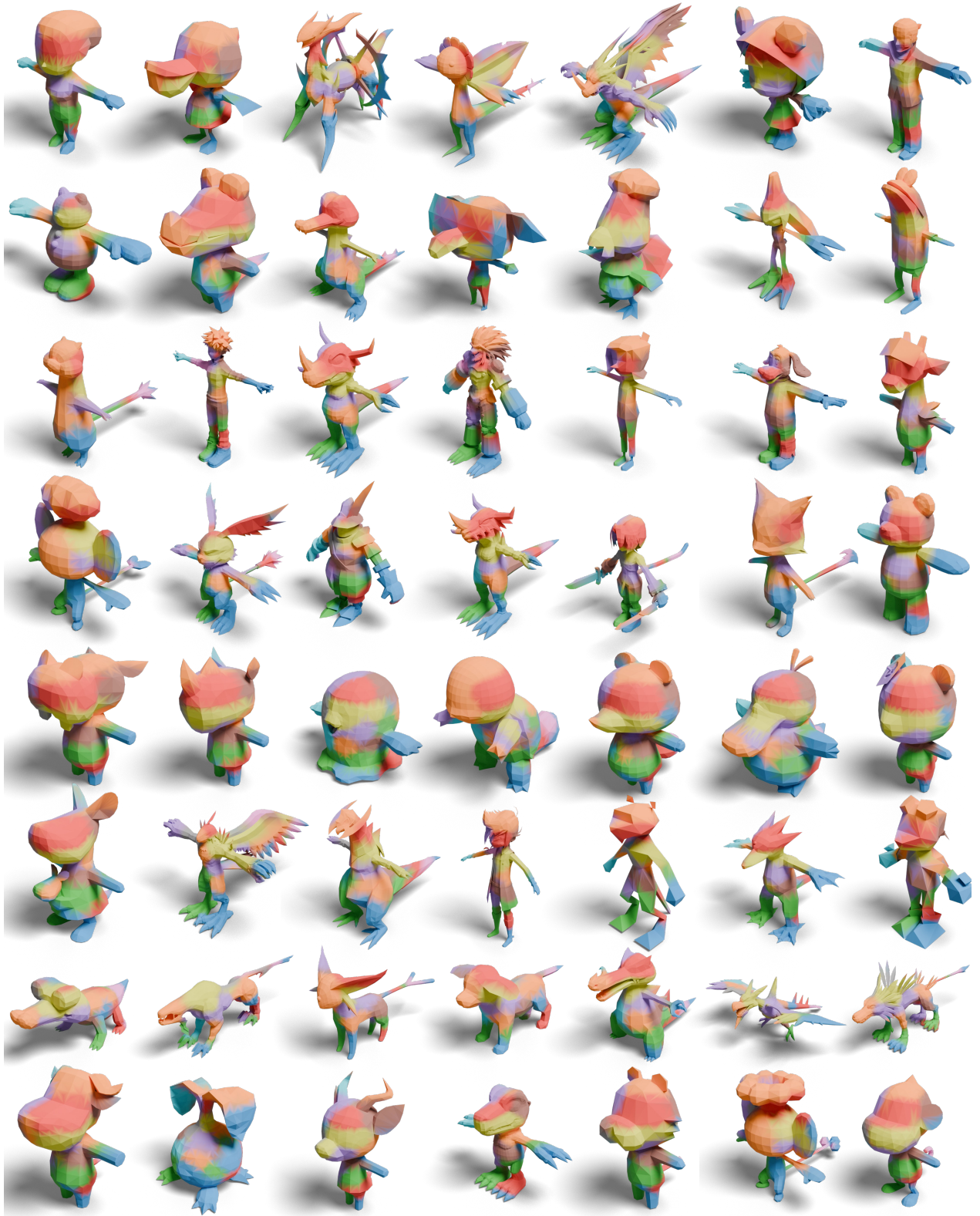


Figure B. Qualitative results on RigNet [13] test set. We visualize blended skinning fields to demonstrate our method’s generalization to diverse 3D shapes.

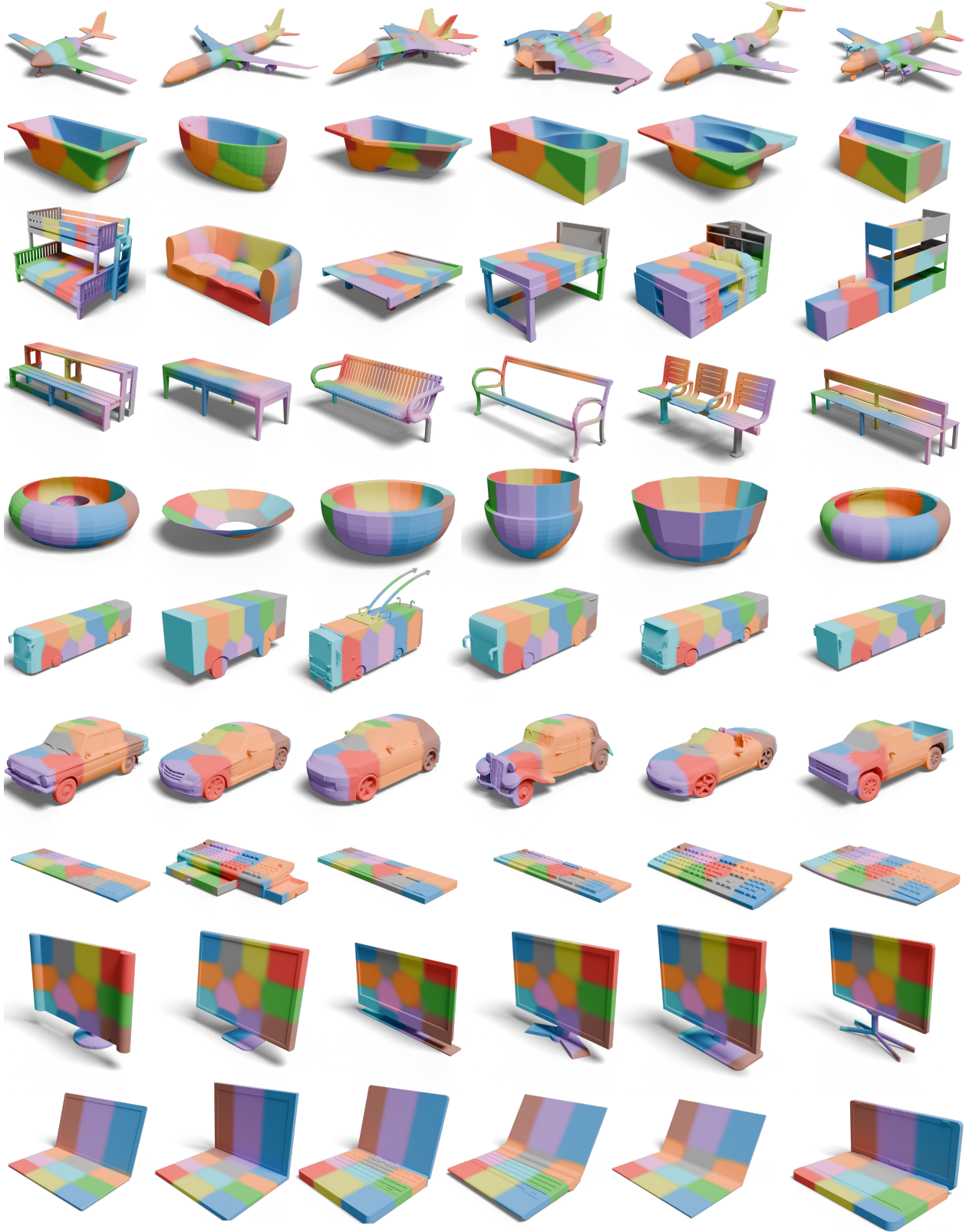


Figure C. Qualitative results on ShapeNet [2] test set. We visualize blended skinning fields to demonstrate our method’s generalization to diverse 3D shapes.

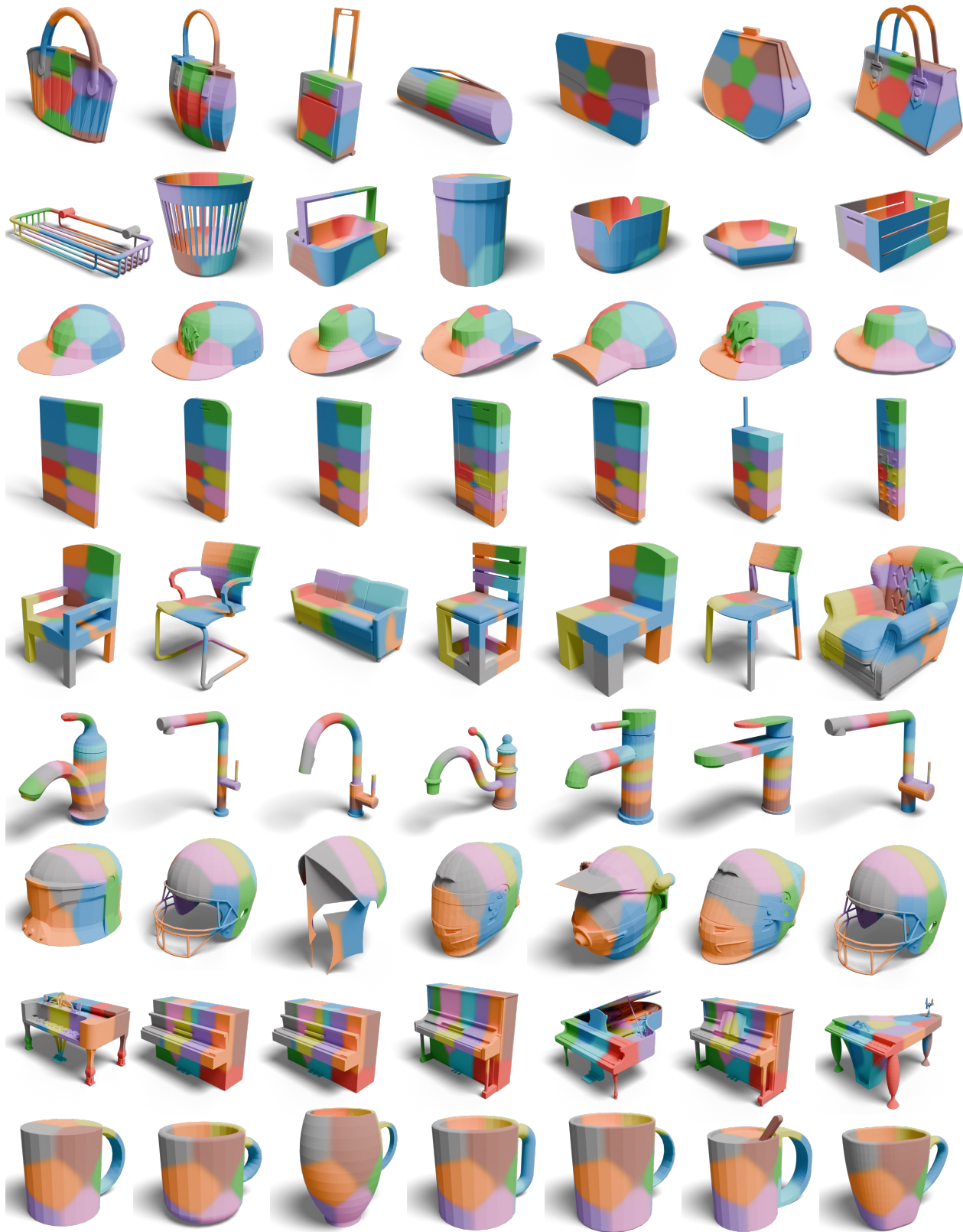


Figure D. Qualitative results on ShapeNet [2] test set. We visualize blended skinning fields to demonstrate our method’s generalization to diverse 3D shapes.

References

- [1] Otman Benchekroun, Jiayi Eris Zhang, Siddhartha Chaudhuri, Eitan Grinspun, Yi Zhou, and Alec Jacobson. Fast complementary dynamics via skinning eigenmodes. *Proc. of SIGGRAPH*, 2023. 2, 3
- [2] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 2, 3, 6, 7
- [3] Yue Chang, Peter Yichen Chen, Zhecheng Wang, Maurizio M Chieramonte, Kevin Carlberg, and Eitan Grinspun. Licrom: Linear-subspace continuous reduced order modeling with neural fields. In *Proc. of SIGGRAPH Asia*, pages 1–12, 2023. 3
- [4] Rui Chen, Jianfeng Zhang, Yixun Liang, Guan Luo, Weiyu Li, Jiarui Liu, Xiu Li, Xiaoxiao Long, Jiashi Feng, and Ping Tan. Dora: Sampling and benchmarking for 3d shape variational auto-encoders. In *Proc. of CVPR*, pages 16251–16261, 2025. 2, 3
- [5] Yufan Deng, Yuhao Zhang, Chen Geng, Shangzhe Wu, and Jiajun Wu. Anymate: A dataset and baselines for learning 3d object rigging. In *Proc. of SIGGRAPH*, pages 1–10, 2025. 3
- [6] Zhiyang Guo, Jinxu Xiang, Kai Ma, Wengang Zhou, Houqiang Li, and Ran Zhang. Make-it-animatable: An efficient framework for authoring animation-ready 3d characters. In *Proc. of CVPR*, pages 10783–10792, 2025. 3
- [7] Jingwei Huang, Hao Su, and Leonidas Guibas. Robust watertight manifold surface generation method for shapenet models. *arXiv preprint arXiv:1802.01698*, 2018. 2
- [8] Jingwei Huang, Yichao Zhou, and Leonidas Guibas. Manifoldplus: A robust and scalable watertight manifold surface generation method for triangle soups. *arXiv preprint arXiv:2005.11621*, 2020. 2
- [9] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM TOG*, 42(4):139–1, 2023. 2
- [10] Vismay Modi, Nicholas Sharp, Or Perel, Shinjiro Sueda, and David IW Levin. Simplicits: Mesh-free, geometry-agnostic elastic simulation. *ACM TOG*, 43(4):1–11, 2024. 3
- [11] Nicolas Moenne-Loccoz, Ashkan Mirzaei, Or Perel, Riccardo de Lutio, Janick Martinez Esturo, Gavriel State, Sanja Fidler, Nicholas Sharp, and Zan Gojcic. 3d gaussian ray tracing: Fast tracing of particle scenes. *ACM TOG*, 2024. 3
- [12] Chaoyue Song, Xiu Li, Fan Yang, Zhongcong Xu, Jiacheng Wei, Fayao Liu, Jiashi Feng, Guosheng Lin, and Jianfeng Zhang. Puppeteer: Rig and animate your 3d models. In *Proc. of NeurIPS*, 2025. 3
- [13] Zhan Xu, Yang Zhou, Evangelos Kalogerakis, Chris Landreth, and Karan Singh. Rignet: Neural rigging for articulated characters. *Proc. of SIGGRAPH*, 2020. 2, 3, 4, 5
- [14] Zibo Zhao, Wen Liu, Xin Chen, Xianfang Zeng, Rui Wang, Pei Cheng, Bin Fu, Tao Chen, Gang Yu, and Shenghua Gao. Michelangelo: Conditional 3d shape generation based on shape-image-text aligned latent representation. In *Proc. of NeurIPS*, 36:73969–73982, 2023. 3