CageNet: A Meta-Framework for Learning on Wild Meshes Supplemental Material

MICHAL EDELSTEIN, Technion - Israel Institute of Technology, Israel
HSUEH-TI DEREK LIU, Roblox & University of British Columbia, Canada
MIRELA BEN-CHEN, Technion - Israel Institute of Technology, Israel

CCS Concepts: • **Computing methodologies** -> **Shape analysis**.

ACM Reference Format:
Michal Edelstein, Hsueh-Ti Derek Liu, and Mirela Ben-Chen. 2025. CageNet: A Meta-Framework for Learning on Wild Meshes Supplemental Material. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH Conference Papers '25), August 10–14, 2025, Vancouver, BC, Canada.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3721238.3730654

1 Hyper Parameters

For both segmentation and skinning weights prediction, we use the basic 4-block DiffusionNet architecture [Sharp et al. 2022], with spectral acceleration and $k = 128$ eigenbasis. We vary the width between the applications, where we use a 64-width NN for the segmentation and a 128-width NN for the skinning weights. We use an ADAM optimizer with an initial initial learning rate of 0.0005 for segmentation and 0.001 for skinning weights. For both applications, we use a batch size of 1, and train for 200 epochs while decaying the learning rate by a factor of 0.5 every 50 epochs, as suggested by Sharp et al. [2022]. We also normalize the cages to the unit sphere, and, when HKS features are used, we utilize DiffusionNet's computation and default parameters – the heat kernel signatures are sampled at 16 values of t logarithmically spaced on [0.01, 1].

2 Skinning Weights Dataset

We evaluate the performance on skinning weights generation on a dataset of 753 artist-created biped characters (see Figure 1) obtained from the Roblox platform, where we hold out 75 meshes as the test set. Meshes in the dataset share the same pose, orientation and skeleton topology.

3 Skinning Weights Symmetry Loss

Let $s \in \mathbb{R}^{n\times k}$ be the ground truth skinning weights matrix, and take s_i to be its i-th column. Further, let $\{\mathcal{S}\}_{i=1}^{N}$ be the set of symmetric vertices, i.e. those which have a close match when reflected around the yz plane. Define $A \in \{0,1\}^{n\times n}$ as the diagonal binary matrix that extracts the symmetric vertices, namely $A_{ii} = 1$ iff $v_i \in \mathcal{S}$. Hence, for a vector h, Ah zeros out all the entries of the non-symmetric vertices. We use A to identify all the joints whose weights have a *large support on symmetric vertices*, by finding all indices $1 \le p \le k$ such that $||As_p||_1 > \delta_s$. Denote this set by $\mathcal{I}$.

Now, take $B \in \{0,1\}^{n\times n}$ to be the binary matrix that maps the symmetric vertices to their match. Thus, $B_{ij} = 1$ iff v_i, v_j are symmetric w.r.t. yz plane. For each $p \in \mathcal{I}$ we check if there exists a joint q, such that the weights s_p, s_q are symmetric on the symmetric vertices. Specifically, for each $p \in \mathcal{I}$ we find $1 \le q \le k$ such that $\frac{\|As_p - Bs_q\|}{N\|s_p\|} < \epsilon_s$. We take $\epsilon_s = 10^{-5}, \delta_s = 30$.

Finally, let $\mathcal{K} = \{(p,q)\}$ be the set of all such pairs. Compute for each pair $l = (p,q)$ and each vertex $1 \le i \le n$ the symmetry error $w(i,l) = \left((A\hat{s}_p)(i) - (B\hat{s}_q)(i)\right)^2$, where $\hat{s}$ are the predicted skinning weights. The symmetry loss is given by:

$$Sym(\hat{s}, s) = \frac{1}{n}\sum_{i=1}^{n}\sqrt{\sum_{l=1}^{|\mathcal{K}|} w(i,l)}.$$

4 CageNet vs. Mesh Repair

Repairing the meshes and then applying a baseline network is problematic for two main reasons. First, meshes may have many variants of issues (non manifold elements, internal components, multiple components), and each such issue will require a different repair method, the choice of which is sometimes manual (and thus not scalable to large datasets). Second, one would still need to map the features between the input mesh and the repaired mesh. Such mapping (e.g. using nearest neighbors) introduces large errors in the functions. For example, for the experiment in Figure 5 in the paper, if we repair the mesh using TetWild [Hu et al. 2018] (obtaining only the exterior surface), and then map the presented function back and forth, we get low accuracy (44.28%) since the internal components are mapped to the exterior surface. Using a cage and barycentric coordinates solves this issue (accuracy 99.98% as we show in Fig. 5), since our setting provides an accurate representation of volumetric functions using a manifold surface. Figure 2 demonstrates this.

5 Segmentation - Additional Results

In Figure 5, we test our segmentation network on wild meshes containing multiple connected components, including a non-manifold one. The segmentation results are consistent with the ground truth of a training mesh shown for reference.

6 Skinning Weights - Additional Results

Quantitative Comparison. Figure 3 shows the average normalized vertex displacement error for each of the animations presented in

Authors' Contact Information: Michal Edelstein, Technion - Israel Institute of Technology, Haifa, Israel, edel.michal@gmail.com; Hsueh-Ti Derek Liu, Roblox & University of British Columbia, Vancouver, Canada, hsuehtil@gmail.com; Mirela Ben-Chen, Technion - Israel Institute of Technology, Haifa, Israel, mirela@cs.technion.ac.il.

SIGGRAPH Conference Papers '25, Vancouver, BC, Canada

ACM ISBN 979-8-4007-1540-2/2025/08
https://doi.org/10.1145/3721238.3730654

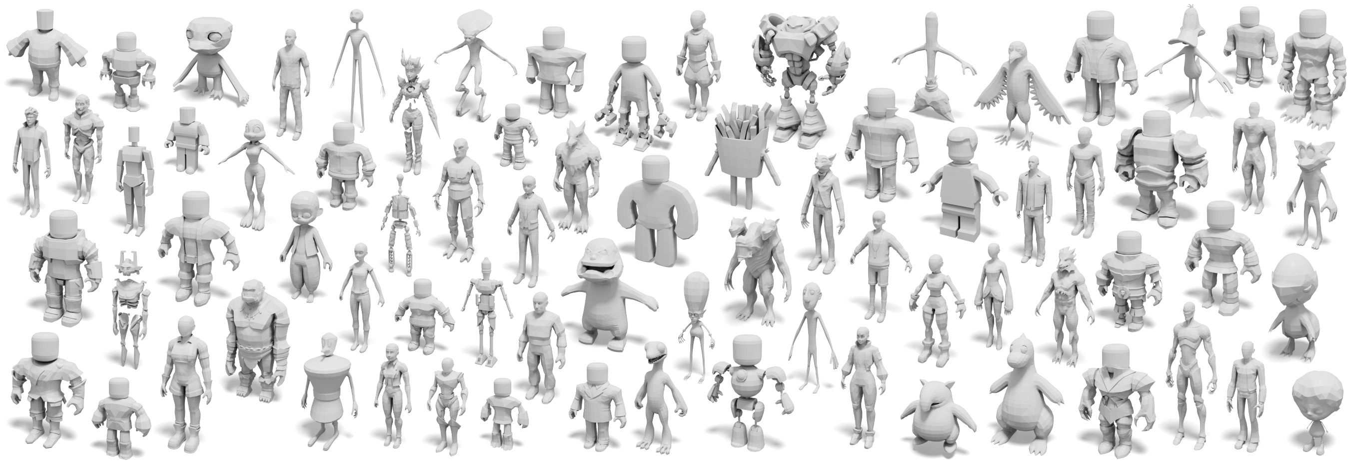

Fig. 1. To evaluate our method on in-the-wild meshes, we collected 753 artist-created characters from the Roblox platform. Here we present samples from the dataset to demonstrate the variety of characters with different complexities.

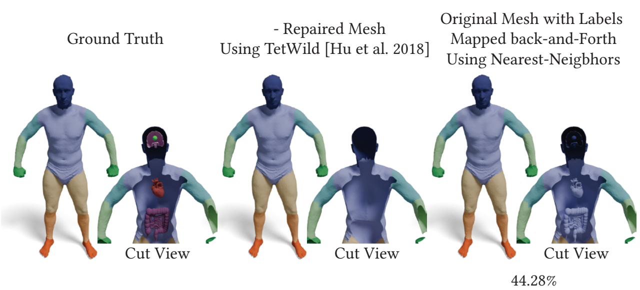

Fig. 2. For a mesh with internal components (left), repairing it using TetWild [Hu et al. 2018] yields a repaired surface (center), such that mapping the labels back and forth from the input to the repaired surface and back leads to a high labeling error (right). Compare with Fig. 5 in the main paper.

the Supplemental video. Note that for all 5 animation sequences, our mean error is lower than the other methods.

Topological Changes. We show our method's performance given topological changes in the cage in Figure 4. Notably, since our training dataset consists of shapes in the A-pose, the neural network was not trained on cages with such topological variations, or the cage offset augmentation, which were unnecessary for this dataset. There, it can be that as the hand grows closer to the body, the error increases. To achieve improved robustness to such variations, the training set would need to incorporate meshes with diverse poses and training should be done with the cage offset augmentation.

More Wild Meshes. We also show additional results on wild meshes in Figure 6. A skeleton was manually created for each shape. The predicted skinning weights align well with the ground truth weights, shown on a reference model from the Roblox dataset. For each model, we include an animation frame to illustrate the effectiveness of the predicted weights.

References

20062020year. 2025. Mania Spider Man Unlimited. https://sketchfab.com/3d-models/mania-spider-man-unlimited-1593574ad8e545e2b4cd623ad9e50429. Accessed: 2025-04-23.

SMF Features Developed From Cheryl Hill. 2025. NASB2 - Raphael. https://sketchfab.com/3d-models/nasb2-raphael-397c3336eb074f609303b03b31e1299e. Accessed: 2025-04-23.

Yixin Hu, Qingnan Zhou, Xifeng Gao, Alec Jacobson, Denis Zorin, and Daniele Panozzo. 2018. Tetrahedral Meshing in the Wild. *ACM Trans. Graph.* 37, 4, Article 60 (July 2018), 14 pages. doi:10.1145/3197517.3201353

Microsoft Corporation. 2025. Windows 3D Model Library. https://apps.microsoft.com/detail/9nblggh42ths?hl=en-US&gl=BM. Accessed: 2025-04-23.

rocklee.ff123. 2025. Kaiju no 8 fortnite. https://sketchfab.com/3d-models/kaiju-no-8-fortnite-097d1f6ef16c4edb9d528b60d7dc7125. Accessed: 2025-04-23.

Nicholas Sharp, Souhaib Attaiki, Keenan Crane, and Maks Ovsjanikov. 2022. DiffusionNet: Discretization Agnostic Learning on Surfaces. *ACM Trans. Graph.* 41, 3 (2022), 27:1–27:16.

Sketchfab. 2012. Sketchfab - 3D models. https://sketchfab.com. Accessed: 2025-04-23.

TurboSquid. 2000. TurboSquid - 3D Models. https://www.turbosquid.com. Accessed: 2025-04-23.

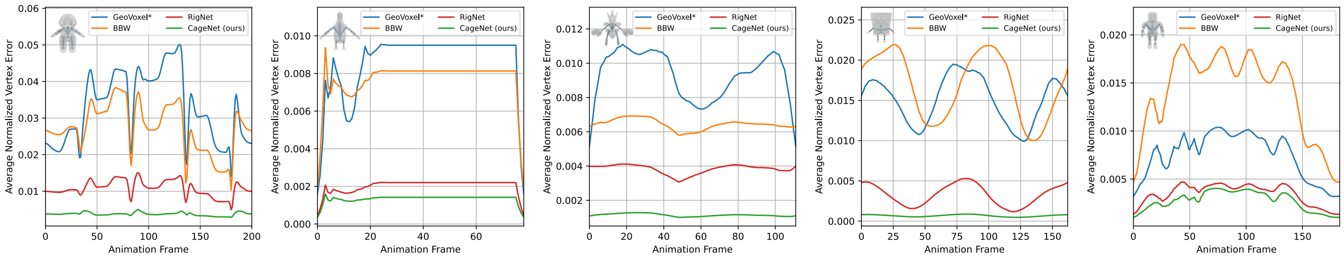

Fig. 3. The average normalized vertex error per frame for all the methods and meshes displayed in the video. Our method has the lowest error across the entire video.

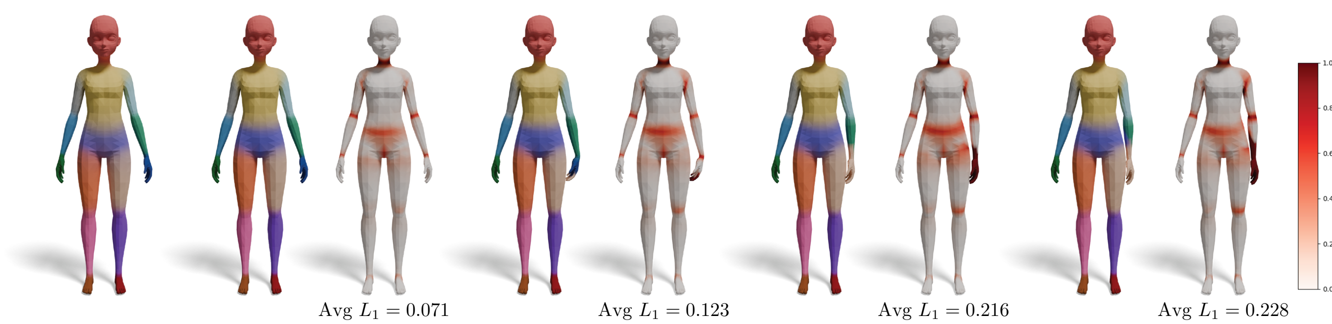

Fig. 4. To demonstrate the effect of topological changes on our skinning results, we deformed a mesh from the test set by moving its arm closer to the body. As the arm approaches, the cage encompasses a larger area of the hand and thigh together, resulting in a larger error in this region.

Fig. 5. Segmentation results on wild meshes. Each mesh contains multiple connected components, visualized using distinct colors (first row), with the corresponding segmentation results shown in the second row. The meshes are taken from the Roblox dataset and from Turbosquid [TurboSquid 2000]

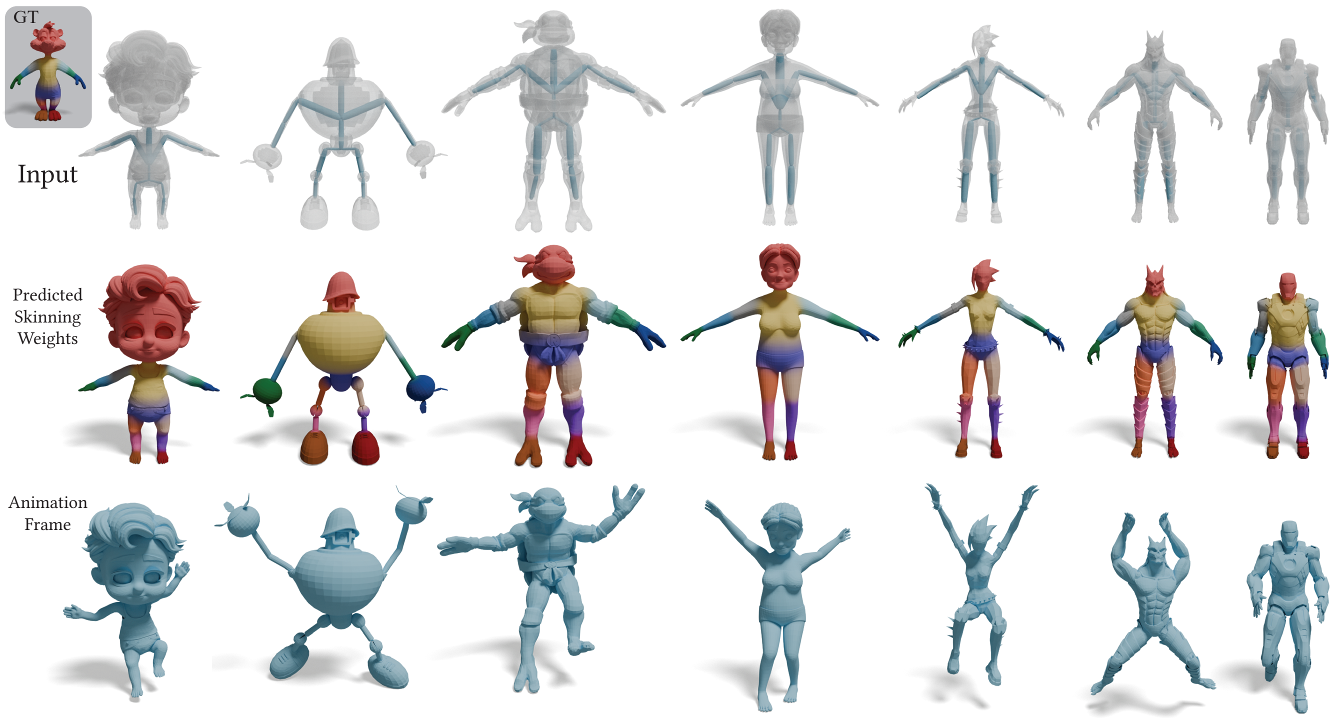

Fig. 6. Skinning weights results on wild meshes containing multiple connected components and non-manifold elements. For each model (top row), we manually created a skeleton consistent with our dataset. We then applied our skinning weight prediction network and show the resulting weights (middle row), as well as one animation frame per model (bottom row). Model sources: Turbosquid [TurboSquid 2000], Windows 3D Library [Microsoft Corporation 2025], Sketchfab [20062020year 2025; Hill 2025; rocklee.ff123 2025; Sketchfab 2012].