Pop-Out Motion: 3D-Aware Image Deformation via Learning the Shape Laplacian — Supplementary Material

Jihyun Lee*[1] Minhyuk Sung*[1] Hyunjin Kim[1] Tae-Kyun Kim[1,2]
[1] KAIST [2] Imperial College London

In this supplementary section, we first provide short motion videos generated using our method in Section S.1. We then report the additional results of our ablation study in Section S.2. In Section S.3, we provide implementation details for the proposed framework. Lastly, we report more qualitative deformation results in Section S.4.

S.1. Motion Videos

We provide brief descriptions for our video attachment that includes short motion animations created by our image deformation method. Our video, which is available at https://youtu.be/gHxwHxIZiuM, contains animations of three Mixamo [6] characters – *Michelle*, *Mousey*, and *Ortiz* – in various motions. We also compare our video results to those created using all the alternative methods discussed in Section 4 of the main paper.

To create the animations, we deform the rendered images of each Mixamo [6] character with manually-selected 32 control point handles and their target positions. Note that we use the same experimental setups as in the Mixamo experiment in the main paper (please refer to Section 4). In the video, our method is shown to produce 3D-aware image deformations that can express various motions of each character. In addition, our method can generate more plausible deformations in comparison to the alternative methods, which often result in undesired artifacts.

S.2. Ablation Study

S.2.1 Learning *Deformation Weights*

We compare our method of learning the shape Laplacian to that of directly learning the deformation weights (i.e., bounded biharmonic weights [7]). To this end, we implement a network that can infer the handle-based deformation weights given a 3D reconstruction represented as a point cloud $\mathcal{P} = \{\mathbf{p}_i\}_{i=1\cdots n}$ and a set of user-defined control points $\{\mathbf{h}_j\}_{j=1,\dots,m}$[1]. The network consists of three modules: Feature Extraction Module, Control Points Embedding Module, and Weight Regression Module. Feature Extraction Module maps each point $\mathbf{p}_i$ in the input point cloud $\mathcal{P}$ to a feature vector $\mathbf{f}_i$. We use Point Transformer [15] architecture as in our original framework. Control Points Embedding Module maps a set of control points $\{\mathbf{h}_j\}_{j=1,\dots,m}$ to a context vector $\mathbf{c}$ that encodes information about the user handle selection. We use a variant of PointNet [11] for the module architecture, since Control Points Embedding Module is desired to extract a feature that is permutation-invariant to the input control points. Specifically, we use a shared multilayer perceptron (MLP) network composed of three layers (each of them followed by batch normalization, LeakyReLU, and dropout) and a max pooling layer to aggregate information from all the control points. Next, Weight Regression Module regresses a deformation weight $\mathbf{w}_{j,i}$ associated with the handle $\mathbf{h}_j$ at point $\mathbf{p}_i$, given the following concatenated feature vector:

$$\mathbf{g}_{ji} = [\mathbf{f}_{\mathbf{h}_j}; \mathbf{c}; \mathbf{f}_i], \tag{1}$$

where $\mathbf{f}_{\mathbf{h}_j}$ denotes a feature vector at the j-th point handle (i.e., $\mathbf{h}_j$) that is produced by the Feature Extraction Module. To implement this module, we use the same architecture as that of Cotangent Laplacian Prediction Module, excluding KNN-Based Point Pair Sampling (KPS) and symmetric feature aggregation components. To train our framework, we use L1 losses to inject the direct supervisions for the predicted deformation weights and the intermediate weights, which is analogous to W in our original framework (please refer to Section 3.2 in the main paper). In our experimental setting, we specifically train our network to estimate deformation weights for *16 control point handles* that are sampled via *farthest point sampling* to match our test scenario.

As shown in Table S1, our original approach of learning the shape Laplacian yields more accurate deformation results compared to that of directly learning the deformation weights. As learning deformation weights is dependent on the control handle selection, we also empirically observed that the deformation weight prediction does not generalize well to a set of control handles whose distribution is different from those of the training examples (e.g., when 16 control points are selected via *random sampling*).

* equal contributions

[1] Among the various types of control handles, we consider *point* handles in this experiment.

Table S1. **Learning the deformation weights vs. learning the shape Laplacian.** All experimental setups are the same as in the DFAUST [3] experiments in the main paper (please refer to Section 4.1). We consider 16 point handles selected via farthest point sampling for both network training and test.

Metric	Learning $\mathbf{w}$	Learning A (Ours)
Weight L1 ($\times 100$) $\downarrow$	9.08	**2.10**
Shape CD ($\times 100$) $\downarrow$	8.38	**1.81**
Shape HD ($\times 0.1$) $\downarrow$	0.44	**0.42**

S.2.2 Qualitative Comparisons

We additionally provide the qualitative results of our ablation study discussed in Tables 4 and S1. In Figure S1, columns 3, 4, 5 and 6 correspond to the ablation study presented in Table 4 in the main paper. Specifically, $-$KPS and $-\alpha$ indicate settings where KPS or α is removed from our Laplacian Learning Network, respectively. EM Only and AD Only denote settings where γ_1 and γ_2 functions in our Cotangent Laplacian Prediction Module are both instantiated as element-wise multiplication and absolute difference, respectively. The column 7 corresponds to the ablation study presented in Table S1 in this supplementary document, where Learning $\mathbf{w}$ denotes the method to directly learn the handle-based deformation weights. Overall, we can observe that the originally proposed method yields the most plausible deformation results than those of the compared settings.

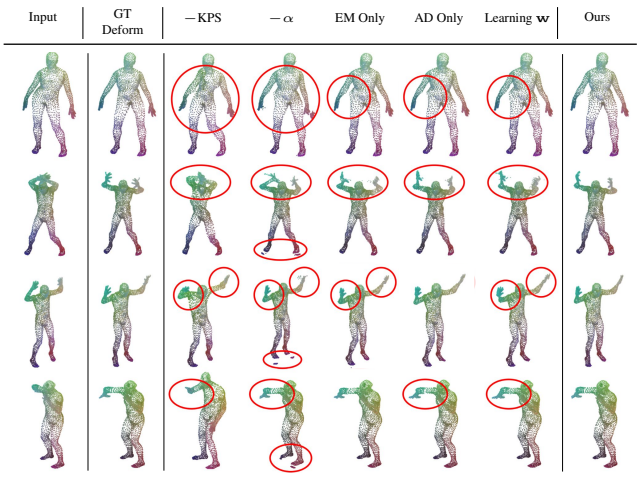

Figure S1. **Qualitative results of our ablation study (best viewed with 200% zoom-in).** Input denotes the source shape and GT Deform indicates the ground truth deformed shape that is computed using the ground truth shape intrinsics. All experimental setups are the same as in the experiments on DFAUST [3] dataset in Section 4.1. We use 16 FPS point handles for each source shape.

S.3. Implementation Details

We now report more details of our implementation.

S.3.1 3D Reconstruction Network

As mentioned in the main paper, we use PIFu [12] to reconstruct an intermediate 3D point cloud from the input image as a prerequisite step to our method. Along with the main PIFu module that learns an implicit function to reconstruct a 3D object geometry, we also train Tex-PIFu module that additionally infers RGB values at given 3D positions of the object surface. Since Tex-PIFu can estimate texture for the object parts that was originally *occluded* in the input image, it allows our method produce image deformations that can *disclose* such occluded parts. When training PIFu and Tex-PIFu, we set the number of epochs as 15 and 10, respectively. We use a batch size of 10 for both modules. Other training setups are the same as those of the original PIFu framework (refer to [12] for more details).

S.3.2 Network Training

The three modules in our Laplacian Learning Network (i.e., Feature Extraction Module, Cotangent Laplacian Prediction Module, and Inverse Mass Prediction Module) are trained in a joint manner. We use a batch size of 8 and set an initial learning rate to 0.1 with a polynomial learning rate decay schedule. Other training details are the same as in the original Point Transformer segmentation network (refer to [15] for more details).

S.3.3 Metric Computation

In the experiments on DFAUST [3] dataset (please refer to Section 4.1 in the paper), we consider three evaluation metrics: (1) L1 distance between the ground truth and the predicted deformation weights, (2) Chamfer distance and (3) Hausdorff distance between the ground truth and the predicted deformed point clouds. Since the ground truth 3D mesh corresponding to each point cloud is available in DFAUST dataset, we first obtain the *ground truth* deformation weights computed using the ground truth topology of the volume mesh generated using [5]. In the same manner, we compute the *ground truth* deformed shape using the ground truth deformation weights and the specified control handle configurations. Since our method models a deformation based on a control handle manipulation, we need to specify the initial handle positions and their transformations to generate the deformed shapes for evaluation. To this end, we first sample the initial point handles (with the number of handles specified in Table 1) via farthest point sampling. Given the source shape and the sampled control points, we retrieve the positions of the semantically aligned points in other shapes in the test set by using the shape correspondences provided in DFAUST dataset. We then use them as target control point positions. In our experiment, we randomly generate 4 different deformations for each test shape and use them to evaluate our deformed shape quality.

S.4. Additional Qualitative Results

S.4.1 Visualization of Learned Deformation Weights

We visualize our deformation weights learned on the point clouds obtained from DFAUST [3] dataset. We show our weights in comparison to those directly computed from the alternative methods (i.e., PSR [8], APSS [4], BPA [2], DeepSDF [10], DGP [14], MIER [9], PCDLap [1], and NMLap [13]). In Figure S2, we show that our method can obtain more accurate deformation weights than the compared scenarios. Especially for the regions around the hands (indicated by red rectangles), our deformation weights are smoothly propagated from the selected control point on the fingertip to *intrinsically* nearby points on the same hand. On the contrary, the compared settings cannot properly distinguish between the two hands. In the same figure, we additionally include the examples of the deformed shapes computed using the shown deformation weights. Our method can produce more plausible shape deformation compared to the other methods.

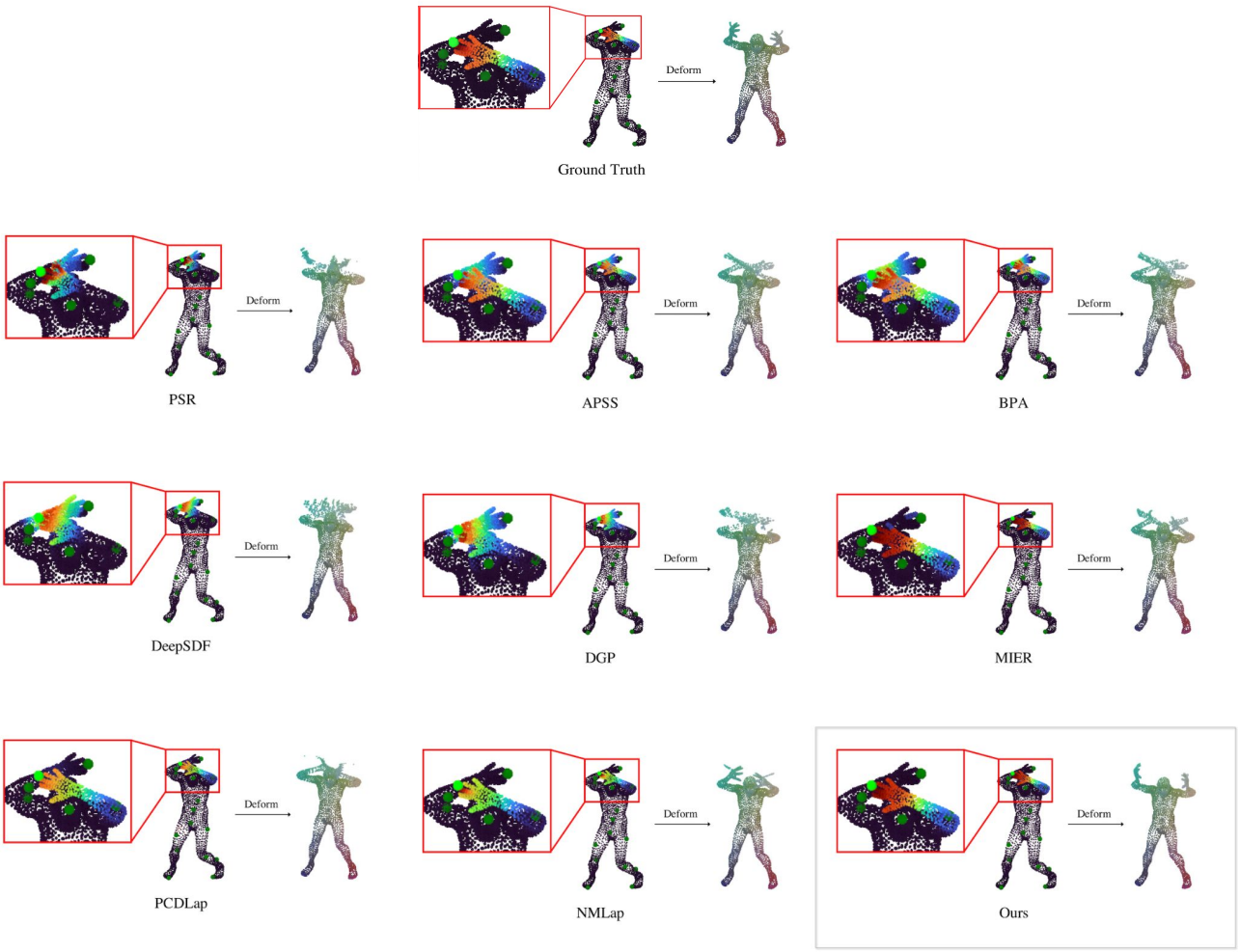

Figure S2. **Visualization of Learned Deformation Weights.** Light green spheres indicate the point handles associated with the shown deformation weights, while dark green spheres are the other point handles. The weight intensity is visualized via color coding (i.e., red for high intensity and blue for low intensity). All experimental setups are the same as in the DFAUST [3] experiment in the main paper (please refer to Section 4.1).

S.4.2 More Comparison Results

We also provide more qualitative results in comparison to the alternative methods (i.e., PSR [8], APSS [4], BPA [2], DeepSDF [10], DGP [14], MIER [9], PCDLap [1], and NMLap [13]) discussed in Section 4 in the main paper. Specifically, these examples are sampled from our video attachment (please refer to Section S.1), which contains the image deformation results of three Mixamo [6] characters – *Michelle*, *Mousey*, and *Ortiz*.

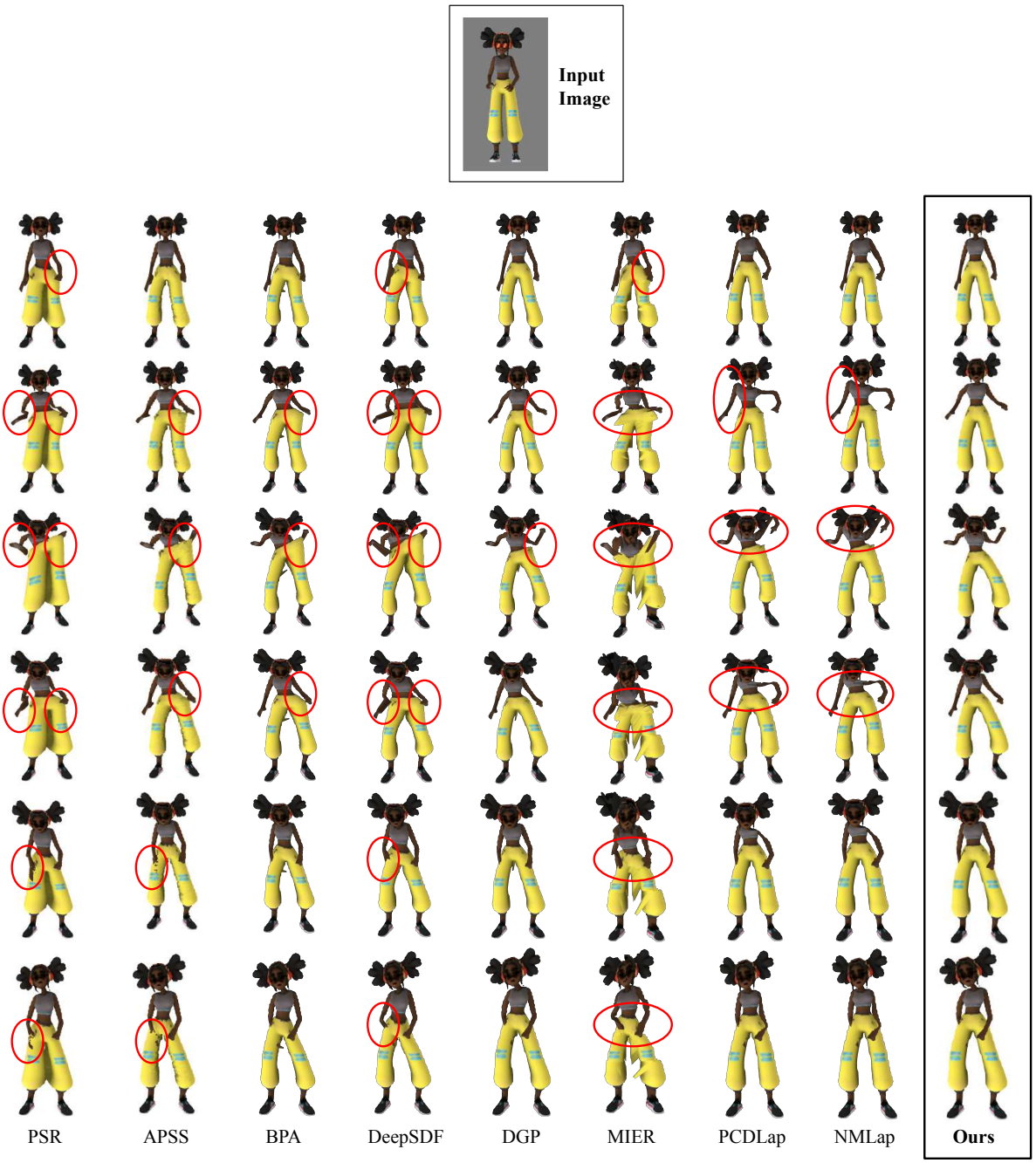

Figure S3. **Comparison results on Mixamo [6] *Michelle* images.** Red circles indicate the image parts with visual artifacts.

Figure S4. **Comparison results on Mixamo [6] *Mousey* images.** Red circles indicate the image parts with visual artifacts.

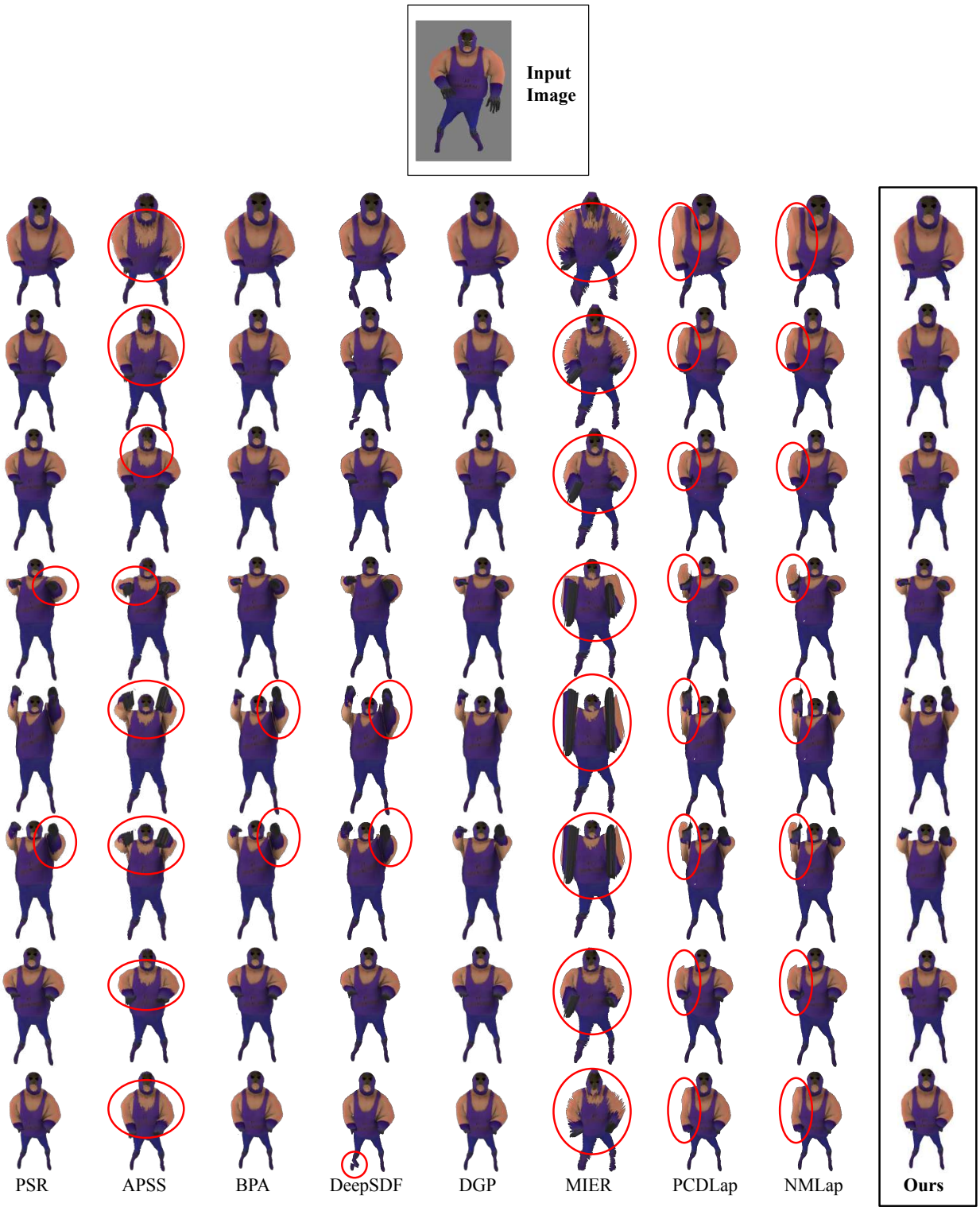

Figure S5. **Comparison results on Mixamo [6] *Ortiz* images.** Red circles indicate the image parts with visual artifacts.

References

[1] Mikhail Belkin, Jian Sun, and Yusu Wang. Constructing laplace operator from point clouds in $\mathbb{R}^d$. In *SODA*, 2009. 3, 4

[2] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Claudio Silva, and Gabriel Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE TVCG*, 1999. 3, 4

[3] Federica Bogo, Javier Romero, Gerard Pons-Moll, and Michael J. Black. Dynamic FAUST: Registering human bodies in motion. In *CVPR*, 2017. 2, 3

[4] Gaël Guennebaud and Markus Gross. Algebraic point set surfaces. In *SIGGRAPH*, 2007. 3, 4

[5] Yixin Hu, Teseo Schneider, Bolun Wang, Denis Zorin, and Daniele Panozzo. Fast tetrahedral meshing in the wild. In *SIGGRAPH*, 2020. 2

[6] Adobe Systems Inc. Mixamo. https://www.mixamo.com. 1, 4, 5, 6

[7] Alec Jacobson, Ilya Baran, Jovan Popovic, and Olga Sorkine. Bounded biharmonic weights for real-time deformation. In *SIGGRAPH*, 2011. 1

[8] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM TOG*, 2013. 3, 4

[9] Minghua Liu, Xiaoshuai Zhang, and Hao Su. Meshing point clouds with predicted intrinsic-extrinsic ratio guidance. In *ECCV*, 2020. 3, 4

[10] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019. 3, 4

[11] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, pages 652–660, 2017. 1

[12] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. PIFu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *ICCV*, 2019. 2

[13] Nicholas Sharp and Keenan Crane. A laplacian for nonmanifold triangle meshes. In *SGP*, 2020. 3, 4

[14] Francis Williams, Teseo Schneider, Claudio Silva, Denis Zorin, Joan Bruna, and Daniele Panozzo. Deep geometric prior for surface reconstruction. In *CVPR*, 2019. 3, 4

[15] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *ICCV*, 2021. 1, 2