

Patch-based Representation and Learning for Efficient Deformation Modeling

Supplementary Material

7.1. PolyFit: Training and Implementation

Training dataset. To support the training of the rotation correction block in PolyFit, we created a dataset consisting of point cloud patches, generated by combining four families of functions, including jet, trigonometric, Gaussian and Bessel. The four families of functions are:

- 1) 4-jet: $f(u, v) = \sum_{i=0}^4 \sum_{j=0}^i \alpha_{i-j,j} u^{i-j} v^j$
- 2) Trigonometric: $T(u, v) = \alpha \cos(\theta \sqrt{u^2 + v^2})$
- 3) Gaussian: $G(u, v) = \alpha \exp\left(-\frac{(u-u_0)^2 + (v-v_0)^2}{2\sigma^2}\right)$
- 4) Bessel: $B(u, v) = \alpha J_0\left(k\sqrt{(u-u_0)^2 + (v-v_0)^2}\right)$

where $\alpha \in [-0.5, 0.5]$, $\theta \in [\pi, 2\pi]$, $\sigma \in [0.5, 1]$ and $k = 5$. Here, J_0 denotes the Bessel function of the first kind of order 0. Using $(u, v) \in [-1, 1]$, we sum the outputs from the four functions and train the PolyFit in a supervised way, by minimizing the height discrepancies between the original and the fitted surface points. We further add patches extracted from CLOTH3D [7] training dataset. The garment meshes are first subdivided four times to achieve a dense mesh. ACVD [72] is applied to the refined mesh, clustering the vertices into k patches according to the surface area. Specifically, the number of patches is given by $\max(100, \min(400, \lfloor \frac{A}{0.008} \rfloor))$, where A denotes the area of the mesh. We extracted 100k patches and computed ground truth normals from their corresponding meshes.

Training details. The batch size is set to 512 and the learning rate is set to 0.001. For every patch, we perform a preprocessing step including normalization, basis extraction and coordinate frame transformation, as depicted in [6]. Figure 10 illustrates the benefit of using STN correction module, which refines the orientation of the given input point cloud and promotes a near-bijective height-graph parameterization before n -jet fitting.

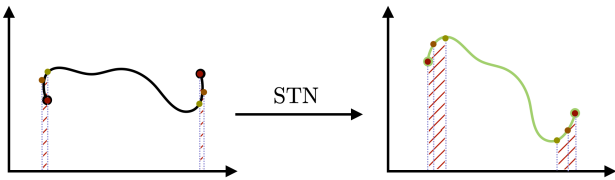


Figure 10. Effect of STN canonicalization. It promotes a near-bijective parameterization (1-D section shown).

7.2. PolyFit: Experiments

Comparison with AtlasNet. We report per-template Chamfer distances for AtlasNet and PolyFit across $K \in$

$\{5, 25, 125\}$ learned charts. For training, we use square (patch) as template type, the number of sampled points is set to 10,000. The point clouds are normalized before computing the metric. As seen in Table 8, varying K leads to only minor CD changes (the total point budget is fixed), while PolyFit attains consistently lower errors on all templates.

	Tshirt	Dress	Tank	Top	Shorts	Pants
AtlasNet ($K = 5$)	0.531	1.039	0.939	0.481	1.508	0.963
AtlasNet ($K = 25$)	0.490	1.060	0.942	0.420	1.471	0.992
AtlasNet ($K = 125$)	0.531	1.111	1.005	0.490	1.547	0.858
PolyFit (Ours)	0.229	0.168	0.268	0.092	0.372	0.237

Table 8. Chamfer Distance (multiplied by 10^3) for patch fitting on six garment templates.

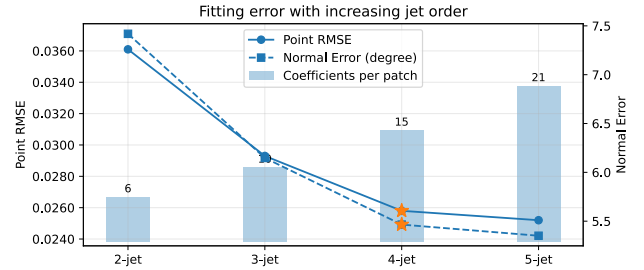


Figure 11. Fitting error with respect to jet order n on CLOTH3D patches.

Ablation on Jet Order n . To evaluate the fitting performance of PolyFit, we use garment patches from the CLOTH3D validation dataset [7]. We compute its performance from metrics including height RMSE and normal loss, measured in degrees. Figure 11 shows the performance of n -jet fitting on the CLOTH3D dataset. This shows that the 4-jet function is capable of fitting point clouds from garment patches effectively. Therefore, we fix $n = 4$, as this setting has been shown to achieve high accuracy on garments with reasonable computational complexity.

Table 9 indicates that the STN module noticeably enhances the model’s fitting accuracy as it re-orientates patches to improve their bijectivity, which leads to better jet-fitting.

	Height RMSE	Normal Diff (°)
with	0.0201	5.274
w/o	0.0259	5.465

Table 9. PolyFit fitting metric, with and w/o STN.

Additional ablation studies. We evaluate PolyFit on patches sampled from the CLOTH3D validation split and compare it with PointNet [60] and DGCNN [76], two point-based networks that we adapt to regress jet coefficients directly from point clouds. As summarized in Table 10, PolyFit delivers lower geometric error (height RMSE and normal difference) and shorter per-patch inference time than these alternatives.

Model	Height RMSE	Normal Diff (°)	Time (ms)
PointNet [60]	0.0309	6.936	0.0754
DGCNN [76]	0.0290	6.406	0.0625
PolyFit	0.0201	5.274	0.0481

Table 10. Comparison with point-based backbones on CLOTH3D validation patches. We report height RMSE, normal-angle error (°), and per-patch inference time (ms).

We further ablate the family of parametric functions used for training. Table 11 shows that increasing the diversity of parametric functions and augmenting the training set with patches extracted from garment meshes both yield additional accuracy gains.

Function used for training	Height RMSE	Normal Diff (°)
Gaussians only	0.0248	5.485
4 Families	0.0239	5.423
4 Families + garment patches	0.0201	5.317

Table 11. Study on different training data for PolyFit.

7.3. PolySfT: Implementation details

Adaptive Window Optimization. We adopt an adaptive sliding-window optimization strategy with a window size of W . Within each window, optimization continues until either the loss fails to improve for a preset number of consecutive iterations (referred to as the **patience threshold**) relative to the current minimum, or the number of iterations exceeds a certain period (referred to as the **frame period**). Once either condition is met, we shift the window forward by one frame and initialize the new frame’s parameters using those from the previous frame. This method promotes temporal consistency and maximizes optimization efficiency.

7.4. PolySfT: Experiments

In addition to the quantitative and qualitative results reported in the main paper, we provide further visual results here. Figure 12 presents additional qualitative results on the Paper-Bend and Kinect-Paper datasets. Renderings of the reconstructed meshes (second column) closely match the input images (first column), resulting in low per-pixel RGB

error maps (third column). Figure 15 shows a comparison with SOTA methods on the synthetic dataset provided by [31].

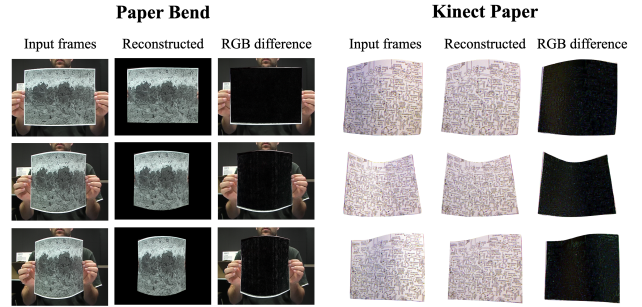


Figure 12. Additional reconstruction results on Paper-Bend and Kinect-Paper.

Stability test. We assess PolySfT’s stability by running the optimization process for many more iterations than usual. Figure 13 displays the reconstructed meshes for two scenes with different motion patterns at 50 iterations, 300 iterations (the average evaluation point), and extended runs at 1000 and 5000 iterations. The results demonstrate that the mesh reliably tracks the intended motion, with only minimal changes beyond the typical iteration threshold. Moreover, initializing from previous frames provides a robust starting point for the current frame. Experiments are conducted on a single NVIDIA V100 GPU.

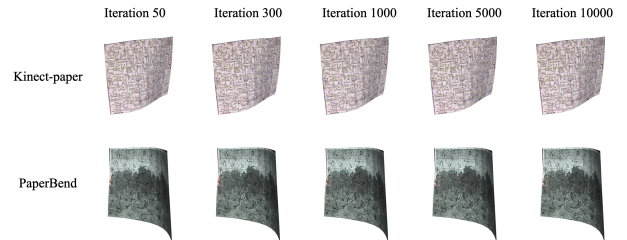


Figure 13. Stability test for Kinect-Paper and Paper-Bend. We show the reconstructed mesh at various iterations.

7.5. OneFit: Network and training details

In this section, we provide details of the OneFit architecture and training setup. In the Dynamic encoder, different from [9], the Gated Recurrent Unit (GRU) layers are initialized with random hidden states. The body feature extractor is implemented using a five-layer multilayer perceptron (MLP) with LeakyReLU activation between the layers. Each layer contains 256 nodes, with the exception of the final layer.

The decoder consists of four fully connected layers, each

with dimensions of 512, 512, 512, and 256, respectively. This is followed by three prediction heads for jet coefficients, translation and scale, each implemented as three fully connected layers with dimensions 128 and 64, ending with a final output layer.

Finally, to maximize parallel computation on GPUs, the batch size for each garment is dynamically determined based on the number of patches using the following equation: $bs = \frac{20,000}{\text{number of patches}}$.

7.6. OneFit: Garment preprocessing

We describe the preprocessing used to align CLOTH3D garments to the average SMPL body in T-pose.

T-pose average shape conversion. In CLOTH3D, garments are posed with legs slightly apart, differing from the standard SMPL T-pose on which skinning weights are defined. In addition, the dataset is fitted on different body shapes. To evaluate garments from CLOTH3D with OneFit, we first preprocess each garment to align it with the average SMPL body in standard T-pose. Specifically, for each garment vertex we query the closest body vertex and displace it according to the difference between the original and standard body shapes. A single iteration of Laplacian surface smoothing is then applied to remove local artifacts. For loose garments such as dresses and skirts, which do not adhere closely to the legs, we only correct the position in terms of shape difference without enforcing pose alignment.

7.7. OneFit: Experiments

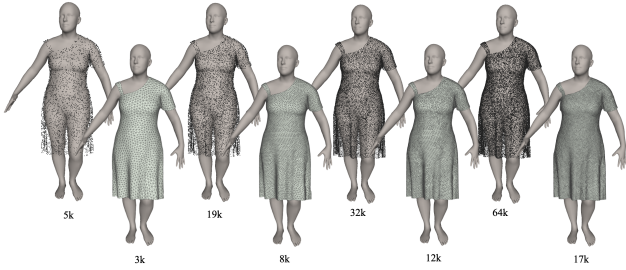


Figure 14. **OneFit** drapings with different mesh resolutions obtained within a similar inference time.

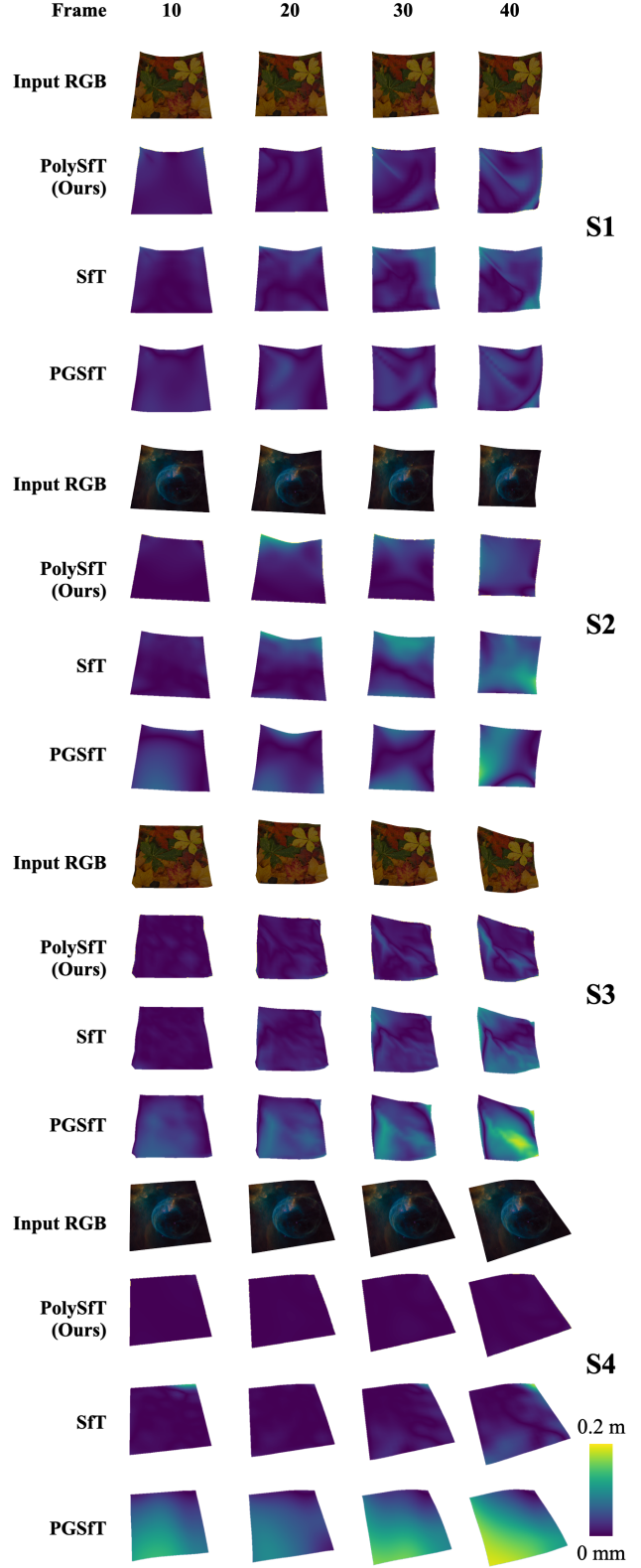


Figure 15. Error map comparison with SOTA methods on ϕ -SfT Synthetic Dataset, showing frames 10, 20, 30, and 40 from left to right.