Patch-based Representation and Learning for Efficient Deformation Modeling

Ruochen Chen   Thuy Tran   Shaifali Parashar

CNRS, École Centrale de Lyon, INSA Lyon, Université Claude Bernard Lyon 1, LIRIS, UMR5205, France

{ruochen.chen, dinh-vinh-thuy.tran, shaifali.parashar}@liris.cnrs.fr

a) General pipeline for mesh processing   b) PolyFit: a patch-based representation   c) PolyFit-based 3D manipulation
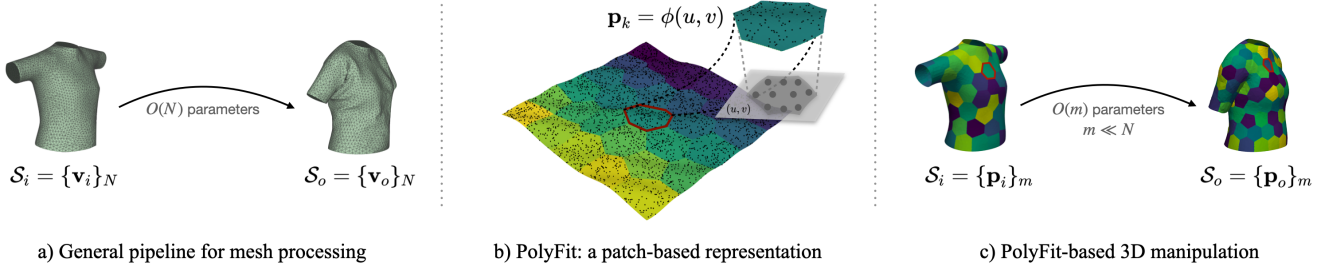
Figure 1. **Patch-based representation and learning.** Against conventional per-vertex parameterizations of mesh deformation (a), we present PolyFit (b) which obtains a simplified patch-wise representation by fitting jet functions with limited parameters; consequently simplifying the transformations to modification of patch parameters only (c) and reducing the computational overhead by a large margin.

Abstract

In this paper, we present a patch-based representation of surfaces, PolyFit, which is obtained by fitting jet functions locally on surface patches. Such a representation can be learned efficiently in a supervised fashion from both analytic functions and real data. Once learned, it can be generalized to various types of surfaces. Using PolyFit, the surfaces can be efficiently deformed by updating a compact set of jet coefficients rather than optimizing per-vertex degrees of freedom for many downstream tasks in computer vision and graphics. We demonstrate the capabilities of our proposed methodologies with two applications: 1) Shape-from-template (SfT): where the goal is to deform the input 3D template of an object as seen in image/video. Using PolyFit, we adopt test-time optimization that delivers competitive accuracy while being markedly faster than offline physics-based solvers, and outperforms recent physics-guided neural simulators in accuracy at modest additional runtime. 2) Garment draping. We train a self-supervised, mesh- and garment-agnostic model that generalizes across resolutions and garment types, delivering up to an order-of-magnitude faster inference than strong baselines.

1. Introduction

3D surface deformation is central to many computer vision and graphics applications such as animation [23], video editing [52] and medical imaging [33], to name but a few. A common practice is to discretize deformable surfaces as meshes and formulate deformations with per-vertex unknowns, either optimized or predicted. Even when driven by reduced controls or regularizers, the underlying degrees of freedom scale with mesh resolution, which makes optimization and inference costly and hinders cross-resolution generalization. Another possibility is to use parametric representations such as splines [10] or NURBS [58] to reduce the number of parameters to be estimated; however, such techniques are practical only for simpler geometries. As the geometries grow more complex, the number of parameters required for accurate representation usually explodes; thus defeating the purpose of using a parametric representation as a low-dimensional deformation state. Although modern learning-based representations such as AtlasNet [24] and other neural representations [38, 55, 75, 77] alleviate some limitations, their large parameter counts and heavy training typically hinder their usage in surface deformation pipelines that require a compact, controllable state.

In this paper, we present a patch-wise, jet-based representation of surfaces which allows an efficient surface deformation with significantly reduced number of parameters to be estimated. Our proposed representation *PolyFit* di-

vides the surfaces into small patches that are represented by simple jet functions, as seen in Figure 1. It is learned in a supervised fashion using analytic functions which are cheap to generate. If needed, its accuracy can be further improved by fine-tuning with a small number of samples of a given object type. To deform surfaces, we directly update the patch-wise n-jet coefficients, thereby limiting the number of parameters to be estimated which leads to efficient processing of the deformations. We showcase the efficiency of our proposed methodology with two well-known problems in computer vision and graphics. First, we propose *PolySfT*: a learning-free, polynomial fitting approach to solve SfT [4, 21, 31, 62, 69] where the goal is to deform a given 3D template as seen in the images. We show that it performs much faster with competitive accuracy than the existing best-performing approaches. Second, we propose *OneFit*: a self-supervised polynomial fitting methodology to learn the draping of the garment. Existing methodologies [9, 15, 65] produce mesh-specific or garment-specific solutions. A few exceptions are [23, 70, 71], which can handle multiple garments at various mesh resolutions. However, [70, 71] are not designed to generate temporally consistent garment deformations as their training process does not incorporate temporal data. In contrast, OneFit is temporally coherent, mesh- and garment-agnostic. Trained on a single garment, OneFit is able to handle different inter-class and intra-class garment variations. Moreover, due to its compact representation, it trains faster than existing methods and provides up to an order-of-magnitude faster inference than strong baselines.

2. Related Work

Surface representation. Parametric chart methods such as AtlasNet [24] learn continuous maps from a 2D domain to 3D surfaces, often trained in a supervised manner on large shape corpora; follow-ups [5, 19] improve efficiency by focusing on local charts. Implicit fields (e.g., SDF/UDF, implicit neural surfaces) [38, 55, 75] model geometry with high capacity but typically require substantial data and compute, and generalization outside the training distribution can be limited. Jets have also been used for local surface fitting [14] and differential estimation on point sets [6] (e.g., normal/curvature via polynomial jets) and, more recently, neural Jacobian-field [2] approaches predict intrinsic Jacobians by supervisedly learning mesh-to-mesh mappings (suitable only for registration or style transfer tasks). Our use of jets differs in both goal and machinery: we make patch-wise n-jet coefficients the state variables of deformation with closed-form derivatives, and drive either a learning-free, test-time inverse optimization (PolySfT) or a self-supervised draping model (OneFit). Unlike [6], which uses a PointNet [60]-based weighting module to modulate the importance of neighbors to infer point-wise differen-

tial quantities, PolyFit performs patch-wise fitting, focusing on efficient deformation modeling rather than per-point geometric property estimation. Unlike [2, 24], PolyFit is trained with lightweight synthetic analytic patches (and a small number of garment patches), is local by construction, and generalizes to arbitrary digital 3D surface deformations.

Shape-from-template. Given a known registration between the texture of the template and the images, [4, 11, 16, 57, 62] compute a unique 3D shape observed in the image, assuming that the object deforms isometrically in a geodesic-preserving fashion, like a piece of paper. Based on this foundational work, [18, 49, 50, 67] developed efficient real-time applications. Further advances have been made to incorporate other deformation models such as conformality [4, 54], equiareality [12, 54], elasticity [1, 28, 43–45], ARAP [53] and diffeomorphism [80]. A major shortcoming of these methods is their inability to handle severe occlusions and capture sharp movements due to the unavailability of high-confidence point correspondences in these scenarios. [20–22, 59, 68] extend SfT to use supervised learning from ground truth data. Recently, [39] proposed a weakly supervised SfT posing manageable constraints on the input sequence to contain some of the already seen shapes during training. However, all these methods severely degrade under challenging scenarios mentioned above.

Alternatively, [31] used physics-based simulation of thin-shell objects [35, 47] to deform a 3D template to match input images. [69] used self-supervised learning of physics-based thin-shell simulations [15, 65] to learn a neural cloth model which operates significantly faster than [31] although with a degraded performance. Our proposed PolySfT uses PolyFit to represent templates with polynomials. It deforms the template by modifying the polynomial parameters to match images in a test-time optimization manner, which is significantly more time and memory efficient than offline physics-based simulations where various physical forces are explicitly manipulated.

Garment Draping. Traditional garment simulation methods rely on computationally expensive but accurate physically based cloth simulation [3, 17, 37, 41, 47, 48]. Advances have been made to reduce the computational complexity of cloth simulation by approximating gradients [29, 34] for fast computation or adding 3D priors [27] such as point clouds of clothed humans. However, these advances compromise reconstruction quality and make the deployment impractical for virtual try-on systems. Unlike traditional approaches, modern learning-based methods yield fast inference. Most methods [7, 25, 26, 40, 51, 56, 63, 64, 66, 74, 76, 78] incorporate a supervised learning approach by using PBS-generated data to learn the relative garment positions with respect to the body. The data generation process is slow and labor intensive which limits the applicability of these methods. Recently, [8, 9, 15, 23, 65] pro-
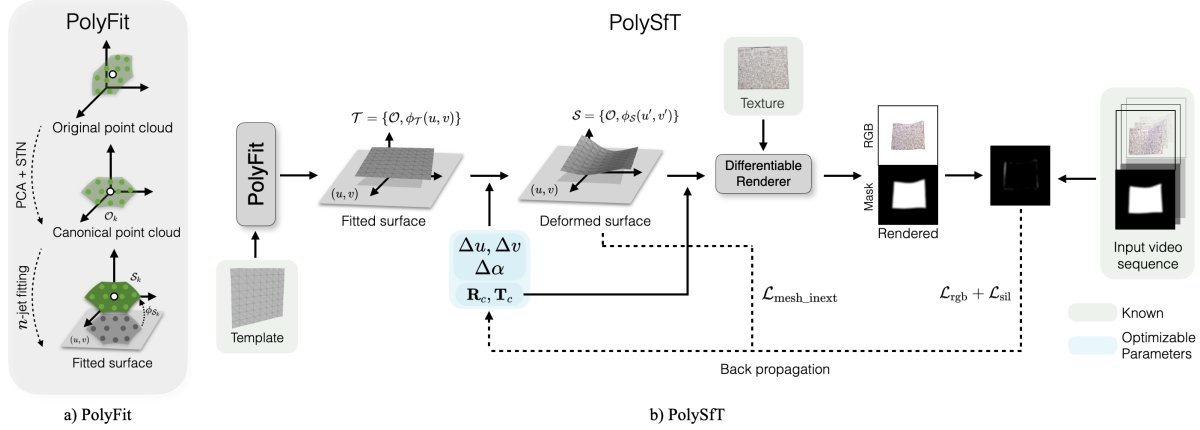
Figure 2. a) **PolyFit**. It orients the input patch to improve bijectivity with the uv-plane and obtains an analytic representation using n-jet fitting. b) **PolySfT**. Using PolyFit, the input 3D template is deformed to match input images by estimating the offsets of fitted jet coefficients $\Delta\alpha$, uv displacements Δuv, as well as a rigid transformation $(\mathbf{R}_c, \mathbf{T}_c)$.

posed self-supervised learning of garment deformations by converting the physical constraints into optimizable losses to estimate garment positions. Most of these methods learn a mesh-specific model which needs to be retrained for slight changes in the garment topology. To our best knowledge, [23] is the only exception that uses graph neural networks to learn temporally coherent drapings of several garment meshes. However, the performance decreases while draping meshes with significantly different resolutions from the training. In contrast, OneFit transforms garment patches into functions using PolyFit, in order to learn drapings of several garments at all possible mesh resolutions in a self-supervised manner similar to [15, 65]. Moreover, as compared to mesh-based methods, OneFit is less prone to cloth self-intersections.

3. PolyFit

PolyFit allows a jet-based representation on surfaces described with points/meshes. It is possible to represent the entire surface $\mathcal{S}$ with a single function or it can be subdivided using Approximated Centroidal Voronoi Diagrams (ACVD) clustering [72], which efficiently constructs uniform tessellations of a given surface area, into desired number of patches. Each patch k is passed into PolyFit which computes orientation $\mathcal{O}_k = (s_k, \mathbf{R}_k, \mathbf{T}_k)$ and a parametric n-jet function $\phi_{\mathcal{S}_k}(u, v)$ with respect to a canonical uv space, as seen in Figure 2(a). Given a set of 3D points $\mathbf{p}_k$ on $\mathcal{S}_k$, PolyFit yields a smooth representation $\mathcal{S}_k := \{\mathcal{O}_k, \phi_{\mathcal{S}_k}(u, v)\}$ such that $\mathbf{p}_k = s_k \mathbf{R}_k \phi_{\mathcal{S}_k}(u, v) + \mathbf{T}_k$. Depending on its orientation, projecting $\mathcal{S}_k$ onto a local 2D frame may produce foldovers (overlaps in (u, v)), so the surface is no longer a single-valued height graph, breaking the bijectivity of $\phi_{\mathcal{S}_k}$. To mitigate this issue, we lever-age Principal Component Analysis (PCA) to transform each patch into a canonical space of maximally planar patch representations, which empirically reduces such degeneracies. We then use a rigid Spatial Transformer Network (STN) [30] parameterized by unit quaternions to refine the orientation and promote a near-bijective height-graph parameterization before n-jet fitting (see Figure 10 in the supplement for an illustrative example).

Following the explicit representation of surfaces in terms of height function, $z(u, v)$, from a canonical uv space, an nth order truncated Taylor expansion of z (also known as n-jet), is given by $z(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{i} \alpha_{i-j,j} u^{i-j} v^j$. The combinations of (α, n) allow an analytic representation of various non-trivial geometries, whose n^{th} order derivatives can be computed precisely. Moreover, given sufficient point samples, $z(u, v)$ can be obtained by fitting an nth order jet in a least squares sense [13]. Therefore, canonical representation of surfaces, in which every point is parameterized by a diffeomorphism $\phi_{\mathcal{S}_k} : (u, v) \mapsto (u, v, z(u, v))^{\top}$, can be oriented using $\mathcal{O}_k = \{s_k, \mathbf{R}_k, \mathbf{T}_k\}$ to fit any smooth surface patch embedded in $\mathbb{R}^3$.

4. PolySfT

PolySfT (see Figure 2(b)) leverages PolyFit to efficiently deform a textured 3D template to match the input images, thereby recovering the 3D shape observed in video. We consider a single-patch representation of the template $\mathcal{T} = \{\mathcal{O}, \phi_{\mathcal{T}}(u, v)\}$, where $\mathcal{O} = (s, \mathbf{R}, \mathbf{T})$ and $\phi_{\mathcal{T}} = (u, v, \sum_{i=0}^{n} \sum_{j=0}^{i} \alpha_{i-j,j} u^{i-j} v^j)^{\top}$. It is deformed using the offset jet coefficients $\Delta\alpha$, uv displacements $(\Delta u, \Delta v)$ as well as a global rotation, $\mathbf{R}_c$ and translation $\mathbf{T}_c$ related to rigid motion of the object. Assuming camera intrinsics are known (a common assumption in SfT), the re-

sulting meshes are converted into RGB and mask images using a differentiable renderer [32]. These renderings are compared to the target input video sequence by computing pixel-wise RGB and silhouette losses similar to [31]. The gradients of the losses are used to refine the optimizable parameters. The reconstructed surface is thus modeled by $\mathbf{R}_c \phi_{\mathcal{S}}(u + \Delta u, v + \Delta v; \alpha + \Delta \alpha) + \mathbf{T}_c$ followed by transformation given by $\mathcal{O}$, to bring the reconstruction from canonical orientation to real one.

Losses. To guide the surface reconstruction, we employ a combination of photometric and geometric losses. Specifically, we adopt an RGB and silhouette loss as defined in [31] to align the reconstructed mesh with the observed imagery. In addition, we apply mesh inextensibility loss to enforce edge-preserving constraints between the deformed and template mesh, $\mathcal{M}_{\mathcal{P}}$ and $\mathcal{M}_{\mathcal{T}}$ respectively:

$$\mathcal{L}_{\text{mesh\_inext}} = k_{\text{mi}} \sum_{i=1}^{n_{\text{edge}}} (e_i(\mathcal{M}_{\mathcal{P}}) - e_i(\mathcal{M}_{\mathcal{T}}))^2 \quad (1)$$

where $e_i(\cdot)$ denotes i-th edge length. Furthermore, we introduce a temporal consistency loss defined as follows, which promotes temporal smoothness across frames:

$$\mathcal{L}_{\text{tc}} = k_{\text{tc}} \frac{1}{W-1} \sum_{t=s}^{s+W-2} \left(\|\delta \alpha_t\|^2 + \|\delta u v_t\|^2 \right). \quad (2)$$

where $\delta \alpha_t = \Delta \alpha_{t+1} - \Delta \alpha_t$, $\delta u v_t = \Delta u v_{t+1} - \Delta u v_t$, and W is the window size.

5. OneFit

OneFit (see Figure 3) leverages PolyFit to efficiently simulate garment deformations using self-supervised learning. The template garment $\mathcal{T}$ is divided into patches using [72] which are passed into PolyFit to obtain a smooth patch representation, $\mathcal{T}_k := \{\mathcal{O}_k, \phi_{\mathcal{T}_k}(u, v)\}$.

A *garment patch embedding*, $\mathbf{Z}_{\mathcal{T}_k}$, is generated by passing $\mathcal{T}_k$ along with its positional encoding into the encoder, an MLP with skip connections. The positional encoding, as described in [46], is applied to each patch to incorporate its center position and its relative offsets from body joints.

A *body embedding*, $\mathbf{Z}_{\mathcal{B}}$ is obtained as a concatenation of dynamic and static encoding. To describe joint orientation relative to the parent joint, we follow [9] and adopt 6D descriptors [79] concatenated with a unit vector with the unposed direction of gravity. This allows to alleviate the discontinuities in the rotation space presented in axis-angle representation. For the structure of the static and dynamic encoder, we adhere to the framework in [9]. The global body pose, $\mathcal{B}(\beta, \theta, \vec{v})$ encapsulates the body shape (β), the current body pose (θ), and the global velocity of the root joint ($\vec{v}$).

Given $\mathcal{B}(\beta, \theta, \vec{v})$ and $\mathcal{T}$, the network first computes the garment patch and body embeddings, $\mathbf{Z}_{\mathcal{T}_k}$ and $\mathbf{Z}_{\mathcal{B}}$ respec-

tively. They are then concatenated and fed into a decoder (details in Section 7.5 of the supplement) as $\mathbf{Z} = \text{concatenate}(\mathbf{Z}_{\mathcal{T}_k}, \mathbf{Z}_{\mathcal{B}})$ to predict the patch deformations, $\mathcal{S}_k := \{\mathcal{O}_k, \phi_{\mathcal{S}_k}(u, v)\}$. The garment deformations are learned by enforcing the physical equilibrium of forces and geometric consistency of template and deformed surface patches posed on the desired body after skinning. This enables a self-supervised, mesh-agnostic, garment-agnostic learning of the deformations.

Geometric Deformation Modeling. As seen in Figure 3, $\mathcal{T}_k$ is deformed to $\mathcal{S}_k$. Upon skinning with $\psi_{\mathcal{S}_k}$, we obtain $\mathcal{P}_k$ posed on body $\mathcal{B}_t$. We impose patch deformations to be isometric and enforce the preservation of their first fundamental form in terms of local metric tensors at $\mathcal{T}_k$, $\mathbf{g}_{\mathcal{T}_k} = \mathbf{J}_{\phi_{\mathcal{T}_k}}^\top \mathbf{J}_{\phi_{\mathcal{T}_k}}$ and $\mathcal{P}_k$, $\mathbf{g}_{\mathcal{P}_k} = \mathbf{J}_{\phi_{\mathcal{S}_k}}^\top \mathbf{J}_{\psi_{\mathcal{S}_k}}^\top \mathbf{J}_{\psi_{\mathcal{S}_k}} \mathbf{J}_{\phi_{\mathcal{S}_k}}$. $\mathbf{J}_{\phi_{\mathcal{T}_k}}$ and $\mathbf{J}_{\phi_{\mathcal{S}_k}}$ can be expressed analytically from the parametric representation obtained in PolyFit. $\mathbf{J}_{\psi_{\mathcal{S}_k}}$ can be analytically calculated from the LBS skinning function [36].

We impose geometric restrictions on the patch boundaries to maintain consistency. Like [15], we allow local stretchings to avoid collisions. We impose following losses:

1) Collision. It penalizes penetration between the body and the garment. For each point, it is given by

$$\mathcal{L}_{\text{collision}} = k_c \sum_{\text{points}} d_c^2, \quad (3)$$

where $d_c = \max(\epsilon - d(x), 0)$ quantifies the degree of interpenetration. $d(x)$ is the signed distance between the garment vertex and the body surface, and ϵ is a small positive constant introduced to enhance stability.

2) Inextensibility. To preserve geodesics between the template and draped garment, it enforces metric tensor similarity. It is computed as

$$\mathcal{L}_{\text{inext}} = k_i \frac{1}{KM} \sum_{\mathcal{T}_k \in \mathcal{T}} \sum_{\mathbf{x} \in \mathcal{T}_k} |k_{\text{ext}} \mathbf{g}_{\mathcal{T}_k}(\mathbf{x}) - \mathbf{g}_{\mathcal{P}_k}(\mathbf{x})| \quad (4)$$

M is the number of points in each patch and K is total number of patches. $k_{\text{ext}} = 1 + \min(d_c, 0.01) \min(e, 100)$, where e is the current epoch. We first allow network to stabilize and then enforce inextensibility.

3) Boundary. It enforces the connectivity between adjacent patches and is defined as follows:

$$\mathcal{L}_{\text{boundary}} = \frac{1}{M_b} \sum_{(i,j) \in \mathcal{B}} \sum_{\text{points}} k_b \|\mathbf{x}_i - \mathbf{x}_j\|^2 + k_{\text{bn}} \left(1 - \cos(\theta_n)\right)^2 \quad (5)$$

where $\mathbf{x}_i$ and $\mathbf{x}_j$ denote boundary points on the adjacent patch of index i and j, M_b denotes the total number of adjacent points between all pairs of patches. $\cos(\theta_n)$ represents the cosine similarity between the normals of the n-th pair
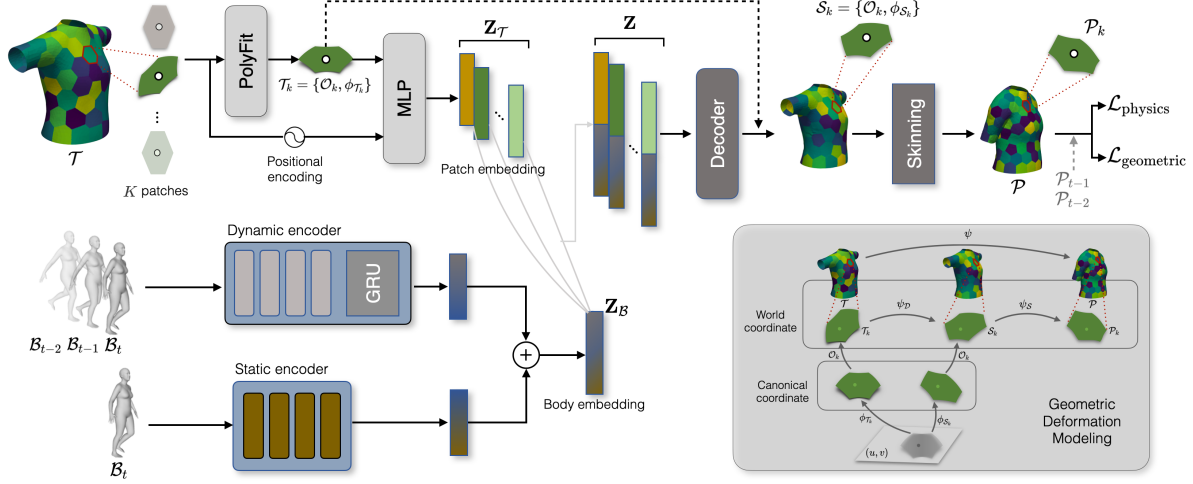
Figure 3. **OneFit.** It deforms $\mathcal{T}$ isometrically to obtain $\mathcal{P}$ posed on body $\mathcal{B}_t$ by forcing patch boundary consistency, avoiding collisions and maintaining physical equilibria.

of adjacent points. It penalizes deviations from perfect parallelism, thus promoting smoother transitions at the boundaries. Overall, the geometric losses are given by

$$\mathcal{L}_{\text{geometric}} = \mathcal{L}_{\text{inext}} + \mathcal{L}_{\text{collision}} + \mathcal{L}_{\text{boundary}} + \mathcal{L}_{\text{mesh\_inext}} \quad (6)$$

$\mathcal{L}_{\text{mesh\_inext}}$, defined in Eq. (1), and $\mathcal{L}_{\text{inext}}$ impose the geodesic preservation constraints at zeroth and first order respectively. It allows the garment deformations to be isometric while taking local body-garment collisions into account.

Physics-based deformation modeling. The physics-based losses incorporate effect of inertia and gravitational forces. The implementation is similar to [15] except losses are defined on points instead of mesh vertices.

1) Gravity. It incorporates gravity by minimizing the potential energy of the garment, given by

$$\mathcal{L}_{\text{gravity}} = \sum_{\text{vertices}} -mg^\top \mathbf{x}, \quad (7)$$

where m is the particle mass and g is the gravitational acceleration.

2) Inertia. It incorporates the inertia loss as proposed in [65]. It is given by

$$\mathcal{L}_{\text{inertia}} = \sum_{\text{vertices}} \frac{1}{2\Delta t^2} m(\mathbf{x}^{[t]} - \mathbf{x}^{[t-1]} - \Delta t v^{[t-1]})^2, \quad (8)$$

where Δt is the simulation time step, $\mathbf{x}^{[t]}$ and $\mathbf{x}^{[t-1]}$ specify the particle's position at times t and $t-1$, respectively.

Overall, physics-based losses are

$$\mathcal{L}_{\text{physics}} = \mathcal{L}_{\text{inertia}} + \mathcal{L}_{\text{gravity}} \quad (9)$$

Together, the losses are given by

$$\mathcal{L} = \mathcal{L}_{\text{physics}} + \mathcal{L}_{\text{geometric}} \quad (10)$$

6. Experiments

6.1. PolyFit

We trained PolyFit on point clouds sampled from regular explicit functions (4-jets, trigonometric, Gaussian and Bessels) and on patches sampled from garment meshes in CLOTH3D [7] dataset, details in Section 7.1 of the supplement. The training time is about 2 hours.

Note that DeepFit [6] and NJF [2] are not directly comparable for our setting. DeepFit performs point-wise jet fitting to estimate local differentials and does not provide a compact patch-level state that can be driven as control variables for surface deformation. NJF presumes a training set of source–target maps and a global latent code to learn piecewise-linear mesh mappings, supervision that is unavailable in our scenario.

We therefore compare PolyFit against AtlasNet [24], a learned multi-chart surface generator, for patch fitting. AtlasNet encodes an input point cloud with PointNet [60] and decodes a latent code through K chart decoders (MLPs), each mapping a 2D parametric domain to a 3D patch; the union of all K patches forms the reconstruction. We train the autoencoder variant of AtlasNet with $K \in \{5, 25, 125\}$ on 5,000 CLOTH3D garments covering diverse types, and evaluate on six garment templates from [65]. For evaluation, we follow the AtlasNet protocol and compute the symmetric Chamfer distance between the reconstruction (concatenating points from all K charts) and 10,000 points uniformly sampled on the ground-truth template. For both methods, the point clouds are normalized before computing the metric. We observe that varying K yields only minor differences in Chamfer distance, so we report the average across K in Table 1 and provide the full table in Table 8

of the supplement. As summarized in Table 1, PolyFit consistently attains lower Chamfer distance, demonstrating accurate fitting with analytic, patch-wise representation. For ablation studies on jet order, the jet-regressor backbone and the training distribution, see Section 7.2 of the supplement.

	Tshirt	Dress	Tank	Top	Shorts	Pants
AtlasNet	0.517	1.070	0.962	0.464	1.509	0.938
PolyFit (Ours)	**0.229**	**0.168**	**0.268**	**0.092**	**0.372**	**0.237**

Table 1. Chamfer Distance (multiplied by 10^3) for patch fitting on six garment templates.

6.2. PolySfT

We use adaptive window optimization ($W=3$, patience 100, period 500; see Section 7.3 in the supplement for details). The loss coefficients k_{mi} and k_{tc} are set to 0.1 and 0.05, respectively. We use Adam optimizer with learning rates 10^{-3} for $(\Delta u, \Delta v)$, 10^{-2} for $\Delta \alpha$, and 10^{-4} for both $\mathbf{R}_c$ and $\mathbf{T}_c$.

We evaluate PolySfT on two real datasets: Kinect-Paper (193 images with ground truth) [73] and Paper-Bend (71 images without ground truth) [61]. Table 2 compares quantitative results on Kinect-Paper against traditional SfT (SFT) [4] and supervised SfT methods [20, 21]; PolySfT attains consistently lower errors than these baselines. We did not compute results for ϕ-SfT [31] due to prohibitive runtime and memory requirements on long sequences. [69] is designed to work with square meshes only and is therefore not applicable to the Kinect-Paper template. Selected reconstructed frames are shown in Figure 4. The code for TD-SfT [20] is not publicly available, so we cannot show visual results. Additional qualitative results on Paper-Bend are shown in Figure 12 of the supplement.

	SFT [4]	**DeepSfT** [21]	**TD-SfT** [20]	**PolySfT**
RMSE (mm)	6.17	6.97	3.37	**2.59**

Table 2. **Kinect-Paper dataset**. Depth RMSE is reported in mm.

In addition, we evaluated PolySfT on synthetic dataset provided by [31], which comprises four sequences (S1-S4) of cloth deformations, each containing between 45 and 50 frames. We compute the 3D error e_{3D} and the average per-vertex angular error e_n in degrees, following the definition given in the supplement of [31]. Table 3 shows that our method outperforms PGSfT and SFT on all sequences. It outperforms ϕ-SfT on S1 and S4, and achieves comparative results on S2 and S3 sequences. Visual comparisons with state-of-the-art methods are shown in Figure 15.

We report wall-clock per-frame optimization time averaged over entire sequences. On an NVIDIA V100

	S1		**S2**		**S3**		**S4**	
	e_{3D}	e_n	e_{3D}	e_n	e_{3D}	e_n	e_{3D}	e_n
PGSfT [69]	0.0298	7.780	0.0448	8.770	0.0823	21.058	0.0919	6.885
ϕ-**SfT** [31]	0.0420	11.860	**0.0230**	10.620	0.0330	**9.120**	0.0050	2.610
SfT [4]	0.0328	7.275	0.0483	7.683	0.0481	14.607	0.0232	5.165
PolySfT	**0.0234**	**6.337**	<u>0.0298</u>	**4.815**	**0.0266**	<u>10.222</u>	**0.0026**	**0.475**

Table 3. Results on the ϕ-SfT synthetic dataset, comparing e_{3D} and e_n errors across methods for sequences S1 to S4.[†]

GPU, PolySfT runs at $\sim$10s/frame. This is $\sim$270$\times$ faster than ϕ-SfT ($\sim$2,800s/frame) and about 2$\times$ slower than PGSfT ($\sim$5s/frame). Despite this gap to PGSfT, PolySfT achieves accurate reconstructions while remaining substantially faster and far more memory-efficient than ϕ-SfT.
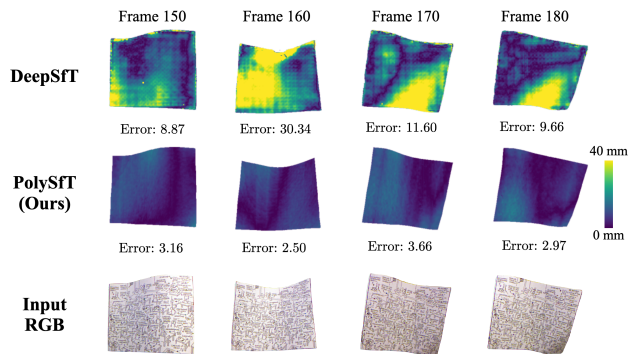

Figure 4. Error map comparison between DeepSfT and Ours on example frames from the Kinect-Paper dataset.

We assess PolySfT's stability by running the optimization beyond 300 iterations, which is our usual checkpoint. We observed the results to stabilize, thus reliably tracking the intended motion, with no visual changes beyond the typical iteration threshold. More details in Section 7.4 of the supplement.

6.3. OneFit

We trained OneFit on a set of 6 standard garment templates (tshirt, dress, pants, shorts, long-sleeve top and tank) used in [65]. We utilize the human motion sequences from the AMASS dataset (60 seq., 10K poses) [42]. We then validate the resulting models on unseen garment meshes from CLOTH3D [7], where the garment preprocessing steps are described in Section 7.6 of the supplement.

We set the adaptive batch size according to the number of patches of the garment. The learning rate begins at 10^{-3} for the first 10 epochs and then reduces to 10^{-4}. We set $k_b = 5e3$, $k_{mi} = 2$, $k_g = 1$, $k_c = 1$, and $k_i = 0.5$. These parameters are fixed for all garments across all experiments.

[†]Best and second-best results are in bold and with underline, respectively.

Figure 5. Single garment OneFit under garment intra-class variations. Trained on a Tank top (in green), OneFit is able to drape tank tops of different styles.

We compare OneFit with state-of-the-art self-supervised methods: GAPS [15], SNUG [65], NCS [9] and HOOD [23]. Except for HOOD, all these methods train mesh-specific models of a single garment. HOOD trains a mesh-based model, but can train a unified model for multiple garments. OneFit trains a mesh-independent model and can learn either a single or a multiple garment network. Furthermore, we can finetune an existing model to a specific garment; thus avoiding from-scratch training. Being mesh-independent, it can generalize to various mesh resolutions. Figure 14 in the supplement shows the scalability of OneFit towards various mesh resolutions with a similar inference time. SNUG requires a post-processing to remove garment-body collision artifacts. GAPS learns a body-specific model; thus no post-processing is required. One-Fit does not require collision post-processing while dealing with garments and bodies in the training dataset or while dealing with garments which cover the garment-body interactions similar to the training data. However, while dealing with unseen garments, for example trying to drape a full-sleeve tshirt from a model trained on half-sleeve tshirt, some collision artifacts are observed, which can be removed with collision post-processing.

OneFit as a single garment model. We test its generalization capabilities. Figure 5 shows the results of OneFit trained on a Tank top. While it drapes well on the trained garment, it generalizes well to the garments of similar style without a post-processing. We also test the generalization capabilities of OneFit towards garment inter-class variations. Figure 6 (top) shows results of OneFit trained on a dress and tested on various garments. Since it learns garment deformations from small patches, it basically learns localized garment-body interactions which are generally extensible to various garments. Hence, we see a decent drape on tshirt and tank tops. The only artifacts that appear over these garment are due to collisions. Since the network is learned on a dress which does not have arms, it is not trained to be aware of the garment-body interactions in this region which makes the collision artifacts inevitable. A simple post-processing can remove these artifacts. The interesting results in Figure 6 (top) are with pants and shorts which are

tightly wrapped to the body as opposed to dress. Besides the collision artifacts, some deformation artifacts are also visible within the area between the legs. Since dress is a loose garment, the network does not witness tightly-bound garment-body interactions between the legs and produces artifacts.
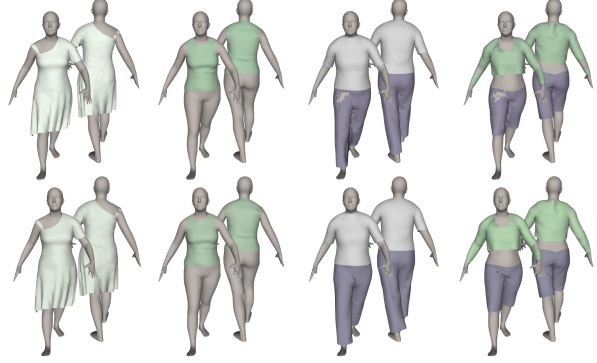

Figure 6. Top: OneFit trained using Dress. Bottom: OneFit trained using a collection of 6 garments.

Loose garments are known to be challenging for most garment draping methods. Figure 7 shows that OneFit trained on dress is close to GAPS, the best performing method in this case. All other methods yield noticeable artifacts.
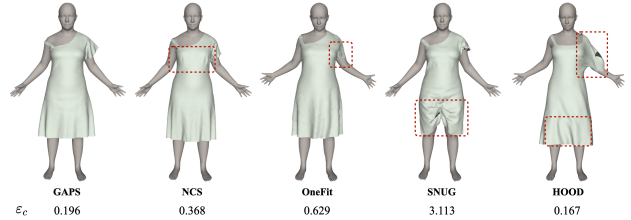

GAPS	NCS	OneFit	SNUG	HOOD
ε_c 0.196	0.368	0.629	3.113	0.167

Figure 7. SOTA comparison for OneFit trained on dress. The results on GAPS are reported after post-processing.[†]

OneFit as a multiple garment model. We trained OneFit jointly on all six garments: tshirt, dress, pants, shorts, long-sleeve top and tank top in order to cover a wide range of body-garment interactions. Table 4 shows that the ε_c (% of vertices under collisions) has drastically reduced as compared to OneFit trained only on dress. We also see that training on multiple garments improves the generalizability of OneFit. Figure 6 (bottom) shows better garment drapings on pants and shorts; which demonstrated deformation artifacts in Figure 6 (top) under a single garment OneFit. Figure 8 shows that OneFit is on par with GAPS,

---

[†]Poses may differ slightly across methods due to differences in the SMPL implementations used by each method.

Model	T-shirt	Dress	Tank	Top	Shorts	Pants
OneFit (Dress)	0.330	0.840	2.834	10.033	6.271	2.389
OneFit (6 garments)	0.422	0.756	0.481	1.592	1.749	1.194

Table 4. ε_c for various garments. Training on multiple garments improves OneFit's generalizability without requiring any post-processing.

Model	ε_c	Training time
OneFit (6 garments)	2.397	8h
OneFit (6 garments) + finetuning	1.982	1h
OneFit (jumpsuit)	1.845	3h

Table 5. Fine-tuning vs training OneFit on jumpsuit.

the best performing method in this case. Figure 9 demonstrates the garment-agnostic nature and capability of OneFit to handle extreme poses, including a high-kick dress and a cobra-pose tank-top.
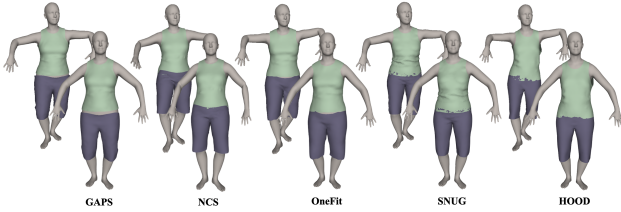

Figure 8. SOTA comparison on tight garments. The results on GAPS are reported after post-processing.[†]
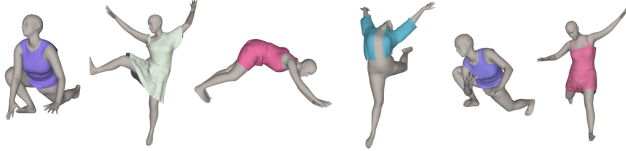

Figure 9. Draping results on challenging poses with diverse garments.

Finetuning OneFit. Once learned, OneFit can be finetuned to a new garment. Table 5 compares OneFit trained on multiple garments to drape a new garment, jumpsuit. Almost 2.5% vertices are observed to be under collision which are brought down to less than 2% by finetuning this model on jumpsuit for an hour. Training OneFit from scratch achieves a similar performance with $3\times$ more computation.

Ablation study. Table 6 shows ablation study on various losses, using a tank top as the test garment. Besides ε_c, we also compute ε_a and ε_e reporting average per-point % area and edge length change. Losses $\mathcal{L}_{\text{mesh\_inext}}$ and $\mathcal{L}_{\text{inext}}$ control the stretchability of garment through zeroth-order (point-based) and first-order (normal-based) metrics; they are both required to minimize size variations while avoiding collisions. $\mathcal{L}_{\text{col}}$ reduces the amount of body-garment collisions.

Model	ε_e	ε_a	ε_c
OneFit	7.828	13.020	0.227
no $\mathcal{L}_{\text{mesh\_inext}}$	13.175	24.011	0.263
no $\mathcal{L}_{\text{inext}}$	7.739	12.760	1.641
no $\mathcal{L}_{\text{col}}$	8.004	13.373	0.387

Table 6. Ablation of OneFit training losses.

	SNUG	**HOOD**	**GAPS**	**OneFit**
Train	2 h	10 h	2-6 h	2-8 h
Runtime	32.4 ms	125.5 ms	5.12 ms	0.48 ms

Table 7. Timing performance.

Timing Comparison. Table 7 reports the timing performance. The mesh specific methods take less time to train but cannot generalize to different topologies. SNUG takes 2 hours to converge for tight garments with less than 10k vertices. GAPS takes 2 hours in the same setting and up to 6 hours for looser garments like dresses. HOOD reports ~10h. OneFit takes 8 hours for training a multiple garment model on 4 NVIDIA A100 GPUs. For runtime, we evaluate on a 2,175-frame CMU sequence. HOOD takes the longest runtime. SNUG takes less inference time but is slower than GAPS due to additional collision post-processing. OneFit achieves the fastest runtime; the optional post-processing step (required only while draping garments out of training set) adds only ~3-4ms per frame.

7. Conclusions

We introduced PolyFit, a patch-based representation that deforms surfaces via a compact set of jet coefficients. We demonstrated its utility in two applications: PolySfT, a test-time optimization that estimates jet coefficients and local uv shifts so that differentiable renderings match the input images, and OneFit, a self-supervised, mesh- and garment-agnostic neural garment simulation model that generalizes across resolutions and garment types. These results highlight the promise of polynomial, patch-wise representations for efficient deformation modeling/learning.

Limitations. Each patch in PolyFit is encoded as a single-valued height function; extreme wrinkles, large bulges, or self-occlusions may violate this assumption. We plan to adopt adaptive, curvature/visibility-aware partitioning and enable higher-order jets on demand to preserve fine details. Additionally, seam artifacts at patch boundaries may occur under large deformations and are mitigated via a lightweight Laplacian smoothing applied along the boundaries. In future work, we will develop more effective boundary control that removes this post-processing.

References

[1] Antonio Agudo and Francesc Moreno-Noguer. Simultaneous pose and non-rigid shape with particle dynamics. In *CVPR*, 2015. 2

[2] Noam Aigerman, Kunal Gupta, Vladimir G Kim, Siddhartha Chaudhuri, Jun Saito, and Thibault Groueix. Neural jacobian fields: Learning intrinsic mappings of arbitrary meshes, 2022. 2, 5

[3] David Baraff and Andrew Witkin. Large steps in cloth simulation. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, page 43–54, 1998. 2

[4] Adrien Bartoli, Yan Gérard, Francois Chadebecq, Toby Collins, and Daniel Pizarro. Shape-from-template. *IEEE TPAMI*, 37(10):2099–2118, 2015. 2, 6

[5] Jan Bednarik, Shaifali Parashar, Erhan Gundogdu, Mathieu Salzmann, and Pascal Fua. Shape reconstruction by learning differentiable surface representations. In *CVPR*, 2020. 2

[6] Yizhak Ben-Shabat and Stephen Gould. Deepfit: 3d surface fitting via neural network weighted least squares. In *ECCV*, 2020. 2, 5, 1

[7] Hugo Bertiche, Meysam Madadi, and Sergio Escalera. Cloth3d: Clothed 3d humans. In *ECCV*, 2020. 2, 5, 6, 1

[8] Hugo Bertiche, Meysam Madadi, and Sergio Escalera. Pbns: Physically based neural simulation for unsupervised garment pose space deformation. *ACM TOG*, 40(6), 2021. 2

[9] Hugo Bertiche, Meysam Madadi, and Sergio Escalera. Neural cloth simulation. *ACM TOG*, 41(6), 2022. 2, 4, 7

[10] F.L. Bookstein. Principal warps: thin-plate splines and the decomposition of deformations. *IEEE TPAMI*, 11(6):567–585, 1989. 1

[11] F. Brunet, A. Bartoli, and R.I. Hartley. Monocular template-based 3d surface reconstruction: Convex inextensible and nonconvex isometric methods. *Computer Vision and Image Understanding*, 125:138–154, 2014. 2

[12] David Casillas-Perez, Daniel Pizarro, David Fuentes-Jimenez, Manuel Mazo, and Adrien Bartoli. Equiareal shape-from-template. *Journal of Mathematical Imaging and Vision*, 61(5):607–626, 2014. 2

[13] Frédéric Cazals and Marc Pouget. Estimating Differential Quantities Using Polynomial Fitting of Osculating Jets. In *Eurographics Symposium on Geometry Processing*, 2003. 3

[14] F. Cazals and M. Pouget. Estimating differential quantities using polynomial fitting of osculating jets. *Computer Aided Geometric Design*, 22(2):121–146, 2005. 2

[15] Ruochen Chen, Shaifali Parashar, and Liming Chen. Gaps: Geometry-aware, physics-based, self-supervised neural garment draping. In *International Conference on 3D Vision (3DV)*, 2024. 2, 3, 4, 5, 7

[16] Ajad Chhatkuli, Daniel Pizarro, Adrien Bartoli, and Toby Collins. A stable analytical framework for isometric shape-from-template by surface integration. *IEEE TPAMI*, 39(5):833–850, 2016. 2

[17] Gabriel Cirio, Jorge Lopez-Moreno, David Miraut, and Miguel A. Otaduy. Yarn-level simulation of woven cloth. *ACM TOG*, 33(6), 2014. 2

[18] Toby Collins and Adrien Bartoli. [poster] realtime shape-from-template: System and applications. In *2015 IEEE International Symposium on Mixed and Augmented Reality*, 2015. 2

[19] Zhantao Deng, Jan Bednařík, Mathieu Salzmann, and Pascal Fua. Better patch stitching for parametric surface reconstruction. In *3DV*, 2020. 2

[20] David Fuentes-Jimenez, Daniel Pizarro, David Casillas-Perez, Toby Collins, and Adrien Bartoli. Texture-generic deep shape-from-template. *IEEE Access*, 9:75211–75230, 2021. 2, 6

[21] David Fuentes-Jimenez, Daniel Pizarro, David Casillas-Pérez, Toby Collins, and Adrien Bartoli. Deep shape-from-template: Single-image quasi-isometric deformable registration and reconstruction. *Image and Vision Computing*, 127:104531, 2022. 2, 6

[22] Vladislav Golyanik, Soshi Shimada, Kiran Varanasi, and Didier Stricker. Hdm-net: Monocular non-rigid 3d reconstruction with learned deformation model. In *Virtual Reality and Augmented Reality*, 2018. 2

[23] Artur Grigorev, Bernhard Thomaszewski, Michael J Black, and Otmar Hilliges. HOOD: Hierarchical graphs for generalized modelling of clothing dynamics. In *CVPR*, 2023. 1, 2, 3, 7

[24] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan Russell, and Mathieu Aubry. AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In *CVPR*, 2018. 1, 2, 5

[25] Peng Guan, Loretta Reiss, David A. Hirshberg, Alexander Weiss, and Michael J. Black. Drape: Dressing any person. *ACM TOG*, 31(4), 2012. 2

[26] Erhan Gundogdu, Victor Constantin, Shaifali Parashar, Amrollah Seifoddini, Minh Dang, Mathieu Salzmann, and Pascal Fua. Garnet++: Improving Fast and Accurate Static 3D Cloth Draping by Curvature Loss. *IEEE TPAMI*, 44(1):181–195, 2020. 2

[27] Jingfan Guo, Jie Li, Rahul Narain, and Hyun Soo Park. Inverse simulation: Reconstructing dynamic geometry of clothed humans via optimal control. In *CVPR*, 2021. 2

[28] Nazim Haouchine and Stephane Cotin. Template-based monocular 3d recovery of elastic shapes using lagrangian multipliers. In *CVPR*, 2017. 2

[29] Yuanming Hu, Jiancheng Liu, Andrew Spielberg, Joshua B. Tenenbaum, William T. Freeman, Jiajun Wu, Daniela Rus, and Wojciech Matusik. Chainqueen: A real-time differentiable physical simulator for soft robotics. In *ICRA*, 2019. 2

[30] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *Proceedings of the 29th International Conference on Neural Information Processing Systems - Volume 2*, page 2017–2025, Cambridge, MA, USA, 2015. MIT Press. 3

[31] Navami Kairanda, Edith Tretschk, Mohamed Elgharib, Christian Theobalt, and Vladislav Golyanik. ϕ-sft: Shape-from-template with a physics-based deformation model. In *CVPR*, 2022. 2, 4, 6

[32] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for

high-performance differentiable rendering. *ACM Transactions on Graphics*, 39(6), 2020. 4

[33] Jose Lamarca, Shaifali Parashar, Adrien Bartoli, and JMM Montiel. Defslam: Tracking and mapping of deforming scenes from monocular sequences. *IEEE Transactions on robotics*, 37(1):291–303, 2020. 1

[34] Yifei Li, Tao Du, Kui Wu, Jie Xu, and Wojciech Matusik. Diffcloth: Differentiable cloth simulation with dry frictional contact. *ACM TOG*, 42(1), 2022. 2

[35] Junbang Liang, Ming C. Lin, and Vladlen Koltun. Differentiable cloth simulation for inverse problems. In *NeurIPS*, 2019. 2

[36] Siyou Lin, Hongwen Zhang, Zerong Zheng, Ruizhi Shao, and Yebin Liu. Learning implicit templates for point-based clothed human modeling. In *ECCV*, 2022. 4

[37] Tiantian Liu, Adam W. Bargteil, James F. O'Brien, and Ladislav Kavan. Fast simulation of mass-spring systems. *ACM TOG*, 32(6):1–7, 2013. 2

[38] Xiaoxiao Long, Cheng Lin, Lingjie Liu, Yuan Liu, Peng Wang, Christian Theobalt, Taku Komura, and Wenping Wang. Neuraludf: Learning unsigned distance fields for multi-view reconstruction of surfaces with arbitrary topologies. In *CVPR*, 2023. 1, 2

[39] Sara Luengo-Sanchez, David Fuentes-Jimenez, Cristina Losada-Gutierrez, Daniel Pizarro, and Adrien Bartoli. Weakly-supervised deep shape-from-template. *IEEE Access*, 13:22868–22892, 2025. 2

[40] Zorah Lähner, Daniel Cremers, and Tony Tung. DeepWrinkles: Accurate and Realistic Clothing Modeling. In *ECCV*, 2018. 2

[41] Miles Macklin, Matthias Müller, and Nuttapong Chentanez. XPBD: Position-Based Simulation of Compliant Constrained Dynamics. In *Proceedings of the 9th International Conference on Motion in Games*, 2016. 2

[42] Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. AMASS: Archive of motion capture as surface shapes. In *ICCV*, 2019. 6

[43] Abed Malti and Cédric Herzet. Elastic shape-from-template with spatially sparse deforming forces. In *CVPR*, 2017. 2

[44] Abed Malti, Richard Hartley, Adrien Bartoli, and Jae-Hak Kim. Monocular template-based 3d reconstruction of extensible surfaces with local linear elasticity. In *CVPR*, 2013.

[45] Abed Malti, Adrien Bartoli, and Richard Hartley. A linear least-squares solution to elastic shape-from-template. In *CVPR*, 2015. 2

[46] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 4

[47] Rahul Narain, Armin Samii, and James F. O'Brien. Adaptive anisotropic remeshing for cloth simulation. *ACM TOG*, 31 (6):147:1–10, 2012. 2

[48] Andrew Nealen, Matthias Mueller, Richard Keiser, Eddy Boxerman, and Mark Carlson. Physically Based Deformable Models in Computer Graphics. *Comput. Graph. Forum*, 25 (4), 2006. 2

[49] Dat Tien Ngo, Sanghyuk Park, Anne Jorstad, Alberto Crivellaro, Chang D. Yoo, and Pascal Fua. Dense image registration and deformable surface reconstruction in presence of occlusions and minimal texture. In *ICCV*, 2015. 2

[50] Dat Tien Ngo, Jonas Östlund, and Pascal Fua. Template-based monocular 3d shape recovery using laplacian meshes. *IEEE TPAMI*, 38(1):172–187, 2016. 2

[51] Xiaoyu Pan, Jiaming Mai, Xinwei Jiang, Dongxue Tang, Jingxiang Li, Tianjia Shao, Kun Zhou, Xiaogang Jin, and Dinesh Manocha. Predicting loose-fitting garment deformations using bone-driven motion networks. 2022. 2

[52] Shaifali Parashar and Adrien Bartoli. 3dvfx: 3d video editing using non-rigid structure-from-motion. In *Eurographics*, 2019. 1

[53] Shaifali Parashar, Daniel Pizarro, Adrien Bartoli, and Toby Collins. As-rigid-as-possible volumetric shape-from-template. In *ICCV*, 2015. 2

[54] Shaifali Parashar, Daniel Pizarro, and Adrien Bartoli. Local deformable 3d reconstruction with cartan's connections. *IEEE TPAMI*, 42(12):3011–3026, 2020. 2

[55] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019. 1, 2

[56] Chaitanya Patel, Zhouyingcheng Liao, and Gerard Pons-Moll. TailorNet: Predicting Clothing in 3D as a Function of Human Pose, Shape and Garment Style. In *CVPR*, 2020. 2

[57] Mathieu Perriollat, Richard Hartley, and Adrien Bartoli. Monocular template-based reconstruction of inextensible surfaces. *IJCV*, 95(2):124–137, 2011. 2

[58] L. Piegl. On nurbs: a survey. *IEEE Computer Graphics and Applications*, 11(1):55–71, 1991. 1

[59] A. Pumarola, A. Agudo, L. Porzi, A. Sanfeliu, V. Lepetit, and F. Moreno-Noguer. Geometry-Aware Network for Non-Rigid Shape Prediction from a Single View. In *CVPR*, 2018. 2

[60] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *CVPR*, 2017. 2, 5

[61] Mathieu Salzmann, Richard Hartley, and Pascal Fua. Convex optimization for deformable surface 3-d tracking. In *ICCV*, 2007. 6

[62] Mathieu Salzmann, Francesc Moreno-Noguer, Vincent Lepetit, and Pascal Fua. Closed-form solution to non-rigid 3d surface registration. In *ECCV*, 2008. 2

[63] Igor Santesteban, Miguel A. Otaduy, and Dan Casas. Learning-Based Animation of Clothing for Virtual Try-On. *Comput. Graph. Forum*, 38(2):355–366, 2019. 2

[64] Igor Santesteban, Nils Thuerey, Miguel A Otaduy, and Dan Casas. Self-Supervised Collision Handling via Generative 3D Garment Models for Virtual Try-On. In *CVPR*, 2021. 2

[65] Igor Santesteban, Miguel A Otaduy, and Dan Casas. SNUG: Self-Supervised Neural Dynamic Garments. In *CVPR*, 2022. 2, 3, 5, 6, 7

[66] Yidi Shao, Chen Change Loy, and Bo Dai. Towards multi-layered 3D garments animation. In *ICCV*, 2023. 2

[67] Mohammadreza Shetab-Bushehri, Miguel Aranda, Erol Özgür, Youcef Mezouar, and Adrien Bartoli. Robusft: Robust real-time shape-from-template, a c++ library. *Image and Vision Computing*, 141:104867, 2024. 2

[68] Soshi Shimada, Vladislav Golyanik, Christian Theobalt, and Didier Stricker. IsMo-GAN: Adversarial Learning for Monocular Non-Rigid 3D Reconstruction . In *CVPRW*, 2019. 2

[69] David Stotko, Nils Wandel, and Reinhard Klein. Physics-guided shape-from-template: Monocular video perception through neural surrogate models. In *CVPR*, 2024. 2, 6

[70] Lokender Tiwari and Brojeshwar Bhowmick. Garsim: Particle based neural garment simulator. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 4472–4481, 2023. 2

[71] Lokender Tiwari, Brojeshwar Bhowmick, and Sanjana Sinha. Gensim: Unsupervised generic garment simulator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 4169–4178, 2023. 2

[72] Sébastien Valette and Jean-Marc Chassery. Approximated centroidal voronoi diagrams for uniform polygonal mesh coarsening. *Comput. Graph. Forum*, 23(3):381–389, 2004. 3, 4, 1

[73] Aydin Varol, Appu Shaji, Mathieu Salzmann, and Pascal Fua. Monocular 3d reconstruction of locally textured surfaces. *IEEE TPAMI*, 34(6):1118–1130, 2012. 6

[74] Raquel Vidaurre, Igor Santesteban, Elena Garces, and Dan Casas. Fully Convolutional Graph Neural Networks for Parametric Virtual Try-On. *Comput. Graph. Forum*, 2020. 2

[75] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *NeurIPS*, 2021. 1, 2

[76] Tuanfeng Y. Wang, Tianjia Shao, Kai Fu, and Niloy J. Mitra. Learning an intrinsic garment space for interactive authoring of garment animation. *ACM TOG*, 38(6), 2019. 2

[77] Lei Yang, Yongqing Liang, Xin Li, Congyi Zhang, Guying Lin, Alla Sheffer, Scott Schaefer, John Keyser, and Wenping Wang. Neural parametric surfaces for shape modeling. *CoRR*, abs/2309.09911, 2023. 1

[78] Meng Zhang, Tuanfeng Y. Wang, Duygu Ceylan, and Niloy J. Mitra. Dynamic neural garments. *ACM TOG*, 40 (6), 2021. 2

[79] Yi Zhou, Connelly Barnes, Lu Jingwan, Yang Jimei, and Li Hao. On the continuity of rotation representations in neural networks. In *CVPR*, 2019. 4

[80] Erol Özgür and Adrien Bartoli. Particle-sft: A provably-convergent, fast shape-from-template algorithm. *IJCV*, 123 (2):184–205, 2017. 2