SAFT: Shape and Appearance of Fabrics from Template via Differentiable Physical Simulations from Monocular Video – Supplementary Material –

David Stotko Reinhard Klein

University of Bonn

This supplementary material contains further information about the physical simulation and the rendering gradients in Section 1. Further implementation details and additional formulas are summarized in Section 2. The complete evaluation of the appearance estimation as one possible downstream task for the 3D reconstruction is given in Section 3.

1. Method

1.1. Numerical Integration Scheme

For simulation, we discretize the time into constant steps of $\Delta t = 5\,\mathrm{ms}$ resulting in all time-dependent quantities to be discretized accordingly, *e.g.* vertex positions $\vec{x}_n = \vec{x}(n\Delta t)$. The update rules when applying the backward Euler schemes to Newton's law are given by

$$\vec{v}_{n+1} = \vec{v}_n + \Delta t \vec{a}_{n+1} = \vec{v}_n + \Delta t M^{-1} \vec{F}_{n+1} \tag{1}$$

$$\vec{x}_{n+1} = \vec{x}_n + \Delta t \vec{v}_{n+1}. \tag{2}$$

However, these rules require information about the time step that we are trying to compute in the first place. This problem can be solved by introducing the differences $\Delta v = \vec{v}_{n+1} - \vec{v}_n$ and $\Delta x = \vec{x}_{n+1} - \vec{x}_n = \Delta t \vec{v}_{n+1}$ and approximating the force by a first-order Taylor series [1]:

$$\begin{aligned} \vec{F}_{n+1} &= \vec{F}(\vec{x}_n + \Delta\vec{x}, \vec{v}_n + \Delta\vec{v}) \\ &\approx \vec{F}(\vec{x}_n, \vec{v}_n) + \frac{\partial \vec{F}(\vec{x}_n, \vec{v}_n)}{\partial \vec{x}} \Delta\vec{x} + \frac{\partial \vec{F}(\vec{x}_n, \vec{v}_n)}{\partial \vec{v}} \Delta\vec{v} \\ &= \vec{F}_n + \Delta t \frac{\partial \vec{F}_n}{\partial \vec{x}} \vec{v}_{n+1} + \frac{\partial \vec{F}_n}{\partial \vec{v}} \Delta\vec{v} \end{aligned} \tag{3}$$

Inserting this approximation into Equation (1) yields the expression

$$M\vec{v}_{n+1} = M\vec{v}_n + \Delta t \vec{F}_n + (\Delta t)^2 \frac{\partial \vec{F}_n}{\partial \vec{x}} \vec{v}_{n+1} + \Delta t \frac{\partial \vec{F}_n}{\partial \vec{v}} \Delta\vec{v} \tag{4}$$

which can be transformed to get

$$\left(M - (\Delta t)^2 \frac{\partial \vec{F}_n}{\partial \vec{x}} - \Delta t \frac{\partial \vec{F}_n}{\partial \vec{v}} \right) \vec{v}_{n+1} = M\vec{v}_n + \Delta t \vec{F}_n + \Delta t \frac{\partial \vec{F}_n}{\partial \vec{v}} \vec{v}_n \tag{5}$$

for the velocity update. This linear system of equations can be solved using only quantities from the current time step.

1.2. Energy Terms

The forces acting on the cloth can be split into internal forces (stretching, bending, shearing) and external forces (gravity, wind, and other contributions). We model the internal forces explicitly by imposing corresponding deformation energies and obtaining the forces using the derivatives $\vec{F} = -\frac{\partial E}{\partial \vec{x}}$. The following paragraphs will describe each contribution in more detail.

Stretching A stretching energy limits the individual changes in length for edges $\vec{e}^{ij}$ between two connected vertices i and j. We impose a Hookean energy term

$$E_Y = \frac{1}{2} Y \left(\|\vec{e}^{ij}\| - L_0^{ij} \right)^2 \tag{6}$$

that penalizes deviations from the rest length L_0^{ij}. The parameter Y is called stretching stiffness or Young's modulus and scales the amount of stress required to create a certain strain.

Bending We also define two resistances against the deformation of angles within the geometry. The bending energy describes the resistance against a curvature relative to the rest state. In the case of a quad mesh, we use pairs $(\vec{e}^{ij}, \vec{e}^{jk})$ of connected edges that form a straight path on the geometry (see Figure 1). For an enclosed angle $\theta^{ijk} = \sphericalangle(\vec{e}^{ij}, \vec{e}^{jk})$ we define the bending energy

$$E_B = \frac{1}{2} B \left(\theta^{ijk} - \theta_0^{ijk} \right)^2 \tag{7}$$

with the rest angle θ_0^{ijk} and a bending stiffness B [2, 11].

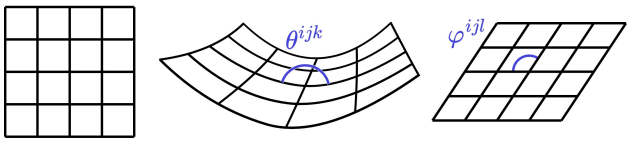


Figure 1. Schematic view of a fabric in its rest state, a bend fabric, and a sheared one.

Shearing Furthermore, an energy term is required that penalizes shearing deformations of the geometry, as these do not necessarily result in stretching or bending of the geometry. In real cloth, such deformations will push different yarns together, resulting in a repulsive force from the collisions. An energy term can again be modeled using the angles $\varphi^{ijl} = \sphericalangle(\vec{e}^{\,ij}, \vec{e}^{\,jl})$ of all remaining pairs of connected edges that were not used for bending (Figure 1). The functional form of the shearing energy

$$E_S = \frac{1}{2} S \left(\varphi^{ijl} - \varphi_0^{ijl}\right)^2 \tag{8}$$

is the same as for the bending energy but contains a different scaling parameter S, the shearing stiffness. It would also easily be possible to allow for independent stiffness parameters for each edge or pair of edges. However, for the sake of simplicity, we opted to use a homogeneous cloth model.

External Forces All other forces acting on the cloth are summarized in external forces $\vec{F}_{\text{ext}}$. In general, these forces are not conservative and, thus, can not be modeled as the gradient of an energy term. Therefore we treat them as arbitrary vectors $\vec{F}_n^i$ for each vertex $i = 1, \ldots, N_{\text{V}}$ at each frame $n = 1, \ldots, N_{\text{f}}$. Two important examples in our scenario are the presence of wind and manipulations by people. Due to their many degrees of freedom, the external forces are able to model the intricate internal cloth forces as well. However, previous work has demonstrated that this approach is much more challenging to optimize and often results in inferior reconstructions [4]. We can observe similar effects, as the forces begin to crumple the fabric together when our regularization terms are omitted, see the ablation study of the main paper (Section 4.1.3). For more control we split the external forces $\vec{F}_n^i = \vec{C} + \vec{D}_n^i$ into a spatially and temporally constant part $\vec{C}$ and a dynamic part $\vec{D}_n^i$.

1.3. Simulation Details

Time Discretization We perform a physical simulation using four constant steps of $\Delta t = 5\,\text{ms}$ between frames to match the $20\,\text{ms}$ video data. For the sake of simplicity, we keep the external forces constant for time steps in between frames, *i.e.* we reuse the same external forces $\vec{F}_n^i$ four times until we switch to the next forces $\vec{F}_{n+1}^i$. Hence, there is no difference between simulation steps and frames other than the simulation error and stability. Individual forces per simulation time step are easy to implement but may require some smoothness regularization to get reasonable interpolation steps due to missing target frames to compare to.

Mass In order to apply Equation (5), we need a mass matrix $M = mI$ associated to each vertex of the cloth. To do so, we define an area density coefficient ρ. We triangulate our template mesh, calculate the mass of each triangle face, and distribute a third of the face mass to each of its vertices. This leads to a relatively homogeneous mass distribution inside the mesh and lower masses at the mesh border. Note that the cloth motion is determined by the acceleration $\vec{a} = M^{-1}\vec{F}$ only and scaling force as well as the mass by the same amount will result in the same motion. Hence, we keep the area density and therefore the masses constant using $\rho = 0.1\,\frac{\text{kg}}{\text{m}^2}$ while optimizing the forces.

Damping We also include an explicit damping term due to the large air resistance present for cloth. We consider the minimalistic damping model

$$\vec{v}_n = \delta\, \vec{v}_n' \tag{9}$$

that scales down the vertex velocities by a factor of δ after the update steps in Equations (2) and (5) are applied. The damping is kept constant at $\delta = 0.9$ as well, since small changes will affect the whole movement and make optimization difficult.

1.4. Orthogonality of Gradients

The gradient of the loss function with respect to some arbitrary (physical) parameter q can be decomposed using the chain-rule. We assume the parameter q to have some influence on a vertex at position $(x\ \ y\ \ z)$ that will be projected on pixel $(p_x\ \ p_y)$ during rendering. An illustration of the projection step is depicted in Figure 2. We now decompose the gradient

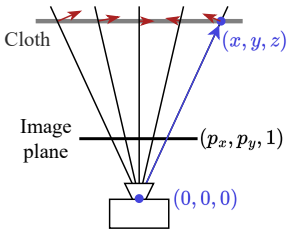


Figure 2. Projection step in camera coordinates.

$$\frac{\partial \mathcal{L}}{\partial q} = \frac{\partial \mathcal{L}}{\partial (p_x\ \ p_y)} \frac{\partial (p_x\ \ p_y)}{\partial (x\ \ y\ \ z)} \frac{\partial (x\ \ y\ \ z)}{\partial q} \tag{10}$$

into three parts denoting contributions for the projection step itself as well as the calculations before and afterwards. Our renderer makes use of the OpenCV camera convention such that the camera intrinsic parameters c_x, c_y, f_x, f_y determine on which pixel a vertex will be projected:

$$\begin{pmatrix} p_x \\ p_y \end{pmatrix} = \begin{pmatrix} \frac{f_x x}{z} + c_x \\ \frac{f_y y}{z} + c_y \end{pmatrix} \tag{11}$$

Its derivative with respect to the vertex coordinates is therefore

$$\frac{\partial (p_x\ \ p_y)}{\partial (x\ \ y\ \ z)} = \begin{pmatrix} \frac{f_x}{z} & 0 & -\frac{f_x x}{z^2} \\ 0 & \frac{f_y}{z} & -\frac{f_y y}{z^2} \end{pmatrix} \tag{12}$$

Similar calculations can be done for other camera conventions as well. An arbitrary image gradient $\frac{\partial \mathcal{L}}{\partial (p_x \;\; p_y)} = (a \;\; b)$ now yields the following 3D gradient for the vertex position:

$$\frac{\partial \mathcal{L}}{\partial (x \;\; y \;\; z)} = \begin{pmatrix} a & b \end{pmatrix} \begin{pmatrix} \frac{f_x}{z} & 0 & -\frac{f_x x}{z^2} \\ 0 & \frac{f_y}{z} & -\frac{f_y y}{z^2} \end{pmatrix} \tag{13}$$

$$= a \begin{pmatrix} \frac{f_x}{z} & 0 & -\frac{f_x x}{z^2} \end{pmatrix} + b \begin{pmatrix} 0 & \frac{f_y}{z} & -\frac{f_y y}{z^2} \end{pmatrix} \tag{14}$$

$$= \begin{pmatrix} a\frac{f_x}{z} & b\frac{f_y}{z} & -a\frac{f_x x}{z^2} - b\frac{f_y y}{z^2} \end{pmatrix} \tag{15}$$

The (unnormalized) viewing direction towards the vertex in camera space is exactly $\vec{d} = (x \;\; y \;\; z)$ and the orthogonality follows directly:

$$\left\langle \frac{\partial \mathcal{L}}{\partial (x \;\; y \;\; z)}, \vec{d} \right\rangle = \begin{pmatrix} a\frac{f_x}{z} & b\frac{f_y}{z} & -a\frac{f_x x}{z^2} - b\frac{f_y y}{z^2} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

$$= 0 \tag{16}$$

As a consequence, the orthogonality of the final gradients $\frac{\partial \mathcal{L}}{\partial q}$ can only be altered by calculations before the projection step happens. In our case, the physical simulation introduces complex interaction between vertices such that the term $\frac{\partial (x \;\; y \;\; z)}{\partial q}$ in Equation (10) breaks the perfect orthogonality. Nonetheless, the loss of information during the projection causes the depth ambiguity problem and heavily influences all gradients with explicit implications for large portions of the gradient computation.

While this property of the rendering gradients is not new, our regularization terms are designed to utilize this knowledge and enhance the gradients in order to improve on 3D reconstruction tasks in monocular setups. While the regularizations are closely related to our specific physical simulation, the presented concepts are applicable in a variety of scenarios.

2. Implementation Details

2.1. Simulation Scheme

As a consequence of the spatial discretization into discrete vertices, the energy terms dependent not only on a single vertex but also on neighboring vertices. In order to get an accurate update step for all vertices at once, we have to include their influence as well. Hence, the derivatives $\frac{\partial \vec{F}_n}{\partial \vec{x}}$ in Equation (5) have to be computed for all vertices participating in the calculation. All derivatives are computed automatically using the automatic differentiation abilities of PyTorch [7] such that only the simple energy functions have to be implemented.

It is possible to infer the physical parameters Y, B, S as well as the external forces acting on the cloth. However, we are only able to infer the net forces that determine the motion of the cloth. Canceling contributions such as gravity and the counteracting holding forces can not be reconstructed individually. Nonetheless, the reconstructed physical parameters and forces can be used to recreate or even change the given dynamics.

Parameter	Init	Min	Max	lr
$\log_{10}(Y)$	$\log_{10}(200)$	1	3	0.02
$\log_{10}(B)$	-3	-4	-2	0.02
$\log_{10}(S)$	-4	-5	-2	0.02
$\vec{C}$	$(0, -1, 0)$	-	-	0.1
$\vec{D}_n^i$	$\vec{0}$	-	-	0.2
Texture T	0.5	0	1	0.05
Diffuse T_D	(0.5)	0	1	$2 \cdot 10^{-4}$
Roughness T_R	(0.5)	10^{-3}	1	$2 \cdot 10^{-4}$
Metallic T_M	(0.5)	0	1	$2 \cdot 10^{-4}$
Environment T_E	1.5	0	-	0.01

Table 1. Initial values, limits, and learning rates of all optimized parameters.

2.2. Stable Gradients

Some functions or derivatives introduce numerically instable or undesired expressions in the forward or backward pass of the optimization. We mitigate this issue by limiting all such values to always maintain some distance to all mathematical singularities. However, there may be dramatic variations in the gradients that remain, and we address this issue through two methods. The first is to clip the gradients to a maximum length of 1000. The second is to employ an automatic gradient clipping algorithm [9] that clips gradients based on their length in previous epochs.

2.3. Initial Values, Limits, and Learning Rates

We always start with the same parameters to reconstruct the cloth. Similarly, some parameters are limited to lie between a minimum and maximum value to ensure a stable simulation. All these values are summarized in Table 1. Note, that we actually optimize the logarithm of the three stiffness parameters (when using SI units), *e.g.* $\log_{10}(Y)$, due to their large range of reasonable values of which we allow at least two orders of magnitude. With our hyperparameters the stiffness parameters do not reach their limits during optimization. The different textures and the environment map are initialized with a uniform gray color. In the case of the material textures (diffuse, roughness, and metallic textures), the decoder starts the optimization with gray textures. Note, that the roughness values have to be strictly positive due to singularities at zero during the rendering.

2.4. Comparison of all evaluated Methods

All three methods use ground truth geometry in the first frame but remeshing introduces minor differences between all of them. ϕ-SfT [4] uses irregular meshes with approximately 400 vertices per scene. They employ arcsim [6] to perform a sophisticated cloth simulation based on continuum mechanics and PyTorch3D [8] for rendering, both of which having high computational costs. PG-SfT [10] needs regular rectangular meshes with $32 \times 32 = 1024$ vertices but does not apply any texture mapping resulting in distorted textures compared to ground truth. The physical simulation is performed by a fast physics-informed neural network that was trained in a self-supervised way. Although the neural network computes the simulation results very quickly, its quality is limited to low-frequency motion due to its simple convolutional architectures. The rendering is done using nvdiffrast [5] for fast and differentiable rendering. Our approach combines the benefit of both predecessors: we employ a fast yet accurate simulation based on a mass-spring system. It produces versatile results, is easy to extend and to implement. For the reconstruction task we use regular meshes with $25 \times 25 = 625$ vertices but arbitrary meshes can be simulated as well. We make use of nvdiffrast as well for fast rendering but also include nvdiffrecmc [3] for high-quality rendering based on ray tracing.

2.5. Additional Formulas

2.5.1. Finite Differences

Within the texture mapping phase, the texture T is smoothed using the regularization $\mathcal{R}_T = \mathrm{FD}(T)$ with

$$\begin{aligned} \mathrm{FD}(T) := \quad & \frac{1}{(w-1)h} \sum_{i=1}^{w-1} \sum_{j=1}^{h} \|T_{i+1,j} - T_{i,j}\|_1 \\ & + \frac{1}{w(h-1)} \sum_{i=1}^{w} \sum_{j=1}^{h-1} \|T_{i,j+1} - T_{i,j}\|_1 \end{aligned} \tag{17}$$

being the formula for the mean of all neighboring finite differences in the texture. The same regularization is employed to smooth the environment map T_E during the appearance estimation phase.

2.5.2. Depth Differences

We are able to render depth images D_R of the reconstructed mesh and compare them with ground truth D_GT. We only consider pixels i, j with depth values within the region of interest (ROI) in which both the ground truth as well as the reconstructed geometry yield valid depth values:

$$\delta(D_\mathrm{GT}, D_\mathrm{R}) = \frac{1}{|\mathrm{ROI}_n|} \sum_{i,j \in \mathrm{ROI}} \|D_{ij,\mathrm{R}} - D_{ij,\mathrm{GT}}\|_2 \tag{18}$$

We report the average depth error for all frames in a scene as the final metric.

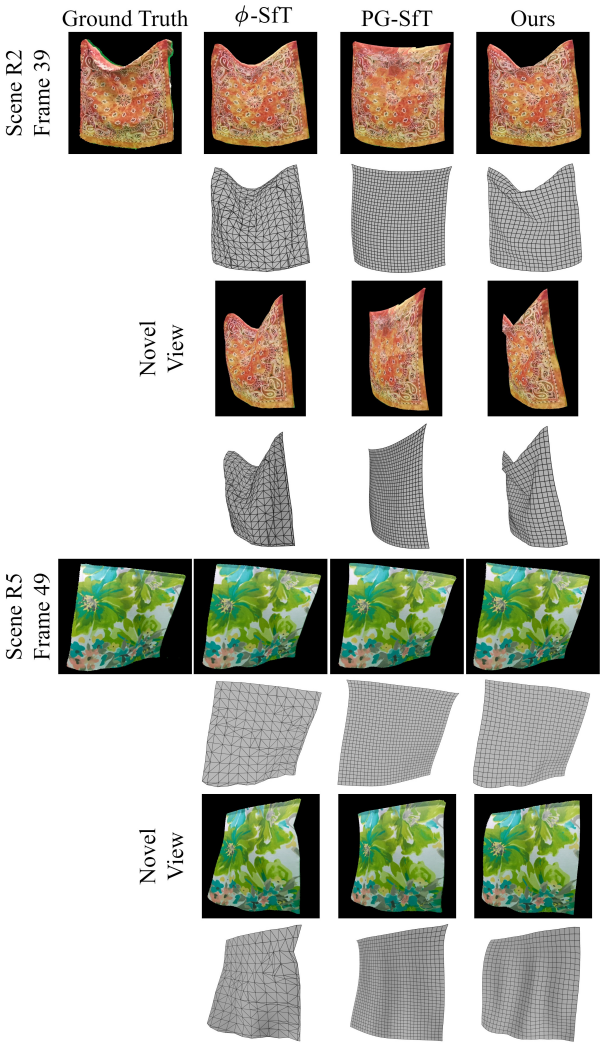


Figure 3. Renderings and mesh views of all methods for two different scenes.

2.5.3. Chamfer Distance

Similar to previous work [4, 10], we use the symmetric Chamfer distance to quantify the quality of our 3D reconstructed geometry.

$$\begin{aligned} \mathbf{CD}_p(P_\mathrm{GT}, P_\mathrm{R}) = \quad & \frac{1}{|P_\mathrm{R}|} \sum_{\vec{r} \in P_\mathrm{R}} \min_{\vec{t} \in P_\mathrm{GT}} \|\vec{r} - \vec{t}\|_2^p \\ & + \frac{1}{|P_\mathrm{GT}|} \sum_{\vec{t} \in P_\mathrm{GT}} \min_{\vec{r} \in P_\mathrm{R}} \|\vec{r} - \vec{t}\|_2^p \end{aligned} \tag{19}$$

The chamfer distance is computed by comparing a ground truth point cloud P_GT from a depth sensor with a sampled point cloud P_R on the reconstructed fabric mesh and averaging these values over all frames in a scene. The point cloud of the reconstructed mesh P_R will always have the same number of points as the ground truth point cloud P_GT. We

use the L_1 Chamfer distance $\mathbf{CD}_1$ as well as the L_2 Chamfer distance $\mathbf{CD}_2$ and compare both of them. The use of linear and quadratic weighting facilitates the identification of outliers.

2.5.4. Point to Surface Distance

We also report the symmetric point-to-surface (p2s) distance between the ground truth point cloud P_{GT} and the reconstructed triangulated surface T_{R}

$$\begin{aligned}\mathrm{p2s}_p(P_{\mathrm{GT}}, T_{\mathrm{R}}) = \quad & \frac{1}{|T_{\mathrm{R}}|} \sum_{\vec{r} \in T_{\mathrm{R}}} \min_{\vec{t} \in P_{\mathrm{GT}}} \mathrm{p2t}(t, r)^p \\ + & \frac{1}{|P_{\mathrm{GT}}|} \sum_{\vec{t} \in P_{\mathrm{GT}}} \min_{\vec{r} \in P_{\mathrm{R}}} \mathrm{p2t}(t, r)^p \end{aligned} \tag{20}$$

using the point-to-triangle distances p2t. Note that in our case the point cloud is the ground truth data and the mesh is the reconstructed surface. Thus, a one-sided metric that averages the minimal distances from each point in the point cloud to the closest point of the mesh surface would not account for overly large reconstructed surfaces. Again, we average the point-to-surface distances for all frames in a scene for the final metric. Similar to the Chamfer distance, we report the L_1 and L_2 p2s distance and compare both of them.

3. Evaluation

3.1. Qualitative Comparison

In addition to the qualitative comparison in the main paper, we present mesh renderings and a novel view for the same scenes in Figure 3 and for all scenes in the supplemental video. The novel view camera is always rotated by $45°$ to the left for all scenes and translated to get a good view on the fabric (with individual translations per scene).

We observe that PG-SfT [10] produces overly smooth geometries that do not capture high-frequency details (*e.g.* in scenes R2, R3, and R4) and distorted textures due to erroneous uv-maps. ϕ-SfT [4] seems to follow the motion well with minor errors. Especially in scene R2 the renderings of ϕ-SfT and our method look similar, however, the novel view and the mesh view reveal significant differences on how the large fold looks like. The fold reconstructed by our method not only looks more realistic, Section 3.2 shows that the 3D error for our method is much lower in the this scenes compared to the other two approaches indicating the correct folding behavior. In scene R5 ϕ-SfT also creates an unrealistic kink close to the top right corner of the mesh which is visible in the novel view in Figure 3 while our method remains smooth as expected for a hanging cloth. Moreover, the supplemental video displays that our method produces visibly the best accordance with the ground truth video almost all the time.

Figure 4 depicts depth values and depth differences with respect to the ground truth depth data for two scenes. The depth values show nicely that PG-SfT does not include any fine wrinkles in the reconstructed mesh despite having the highest mesh resolution. ϕ-SfT and our method both include such details but our results agree better with the ground truth values than ϕ-SfT's results. The supplemental video also points out that our approach yields the best depth values almost all the time.

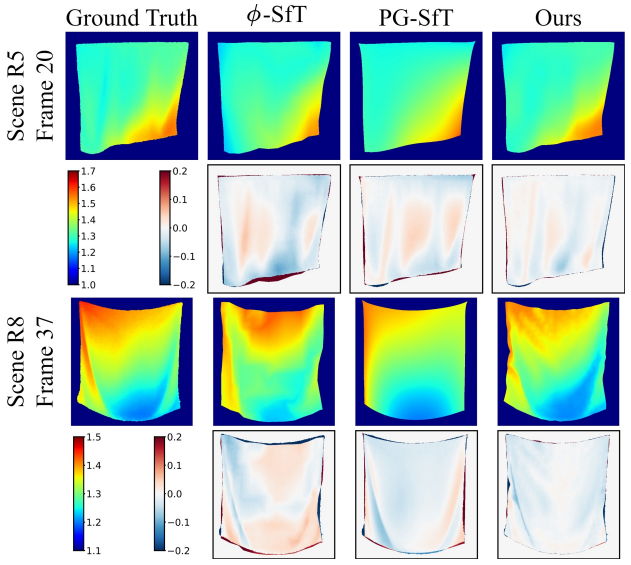


Figure 4. Qualitative comparison between depth images for two example frames of different scenes.

3.2. Quantitative Comparison

We extend our evaluation by comparing several different metrics for each method. First, we evaluate the depth images quantitatively by averaging the mean depth errors of all images per scene (see Section 2.5.2). We only use the depth values where ground truth and reconstruction overlap (*i.e.* there are valid depth values to compare). This excludes small regions in the images in which only one geometry, either the ground truth or the reconstructed one, is present. We summerize the results in Table 2 and observe that our approach beats ϕ-SfT [4] and PG-SfT [10] in all scenes by up to $4.63\times$ and $2.96\times$ respectively. Both baseline methods have similar performance with mean depth errors of $1.67\,\mathrm{cm}$ and $1.83\,\mathrm{cm}$ respectively. We are able to reduce the average errors by a factor of 1.84 compared to ϕ-SfT and 2.01 compared to PG-SfT.

In addition to the L_2 Chamfer distance CD_2, we also report the L_1 Chamfer distance CD_1 in Table 3 (Section 2.5.3). We use the same point cloud as for the L_2 Chamfer distance, that is created by sampling the same number of points as the ground truth point cloud randomly distributed on the reconstructed mesh. The linear weight for each pair of points reduces the contribution of outliers and

Method	R1	R2	R3	R4	R5	R6	R7	R8	R9	Mean
ϕ-SfT [4]	2.50	1.48	1.29	1.99	2.27	1.71	1.66	1.13	1.01	1.67
PG-SfT [10]	1.60	1.13	2.03	2.08	2.00	2.37	1.54	2.00	1.70	1.83
Ours	**0.54**	**0.71**	**0.76**	**1.64**	**1.31**	**0.81**	**1.28**	**0.63**	**0.51**	**0.91**

Table 2. Average depth difference δ for all methods $[\times 10^{-2}\,\mathrm{m}]$.

Method	R1	R2	R3	R4	R5	R6	R7	R8	R9	Mean
ϕ-SfT [4]	2.21	2.63	2.34	3.56	4.31	3.22	3.04	2.14	1.95	2.82
PG-SfT [10]	2.97	2.14	3.69	4.02	3.64	4.38	3.01	3.79	3.24	3.43
Ours	**1.08**	**1.24**	**1.44**	**2.92**	**2.57**	**1.57**	**2.38**	**1.22**	**1.04**	**1.72**

Table 3. Quantitative comparison using the L_1 Chamfer distance $[\times 10^{-2}\,\mathrm{m}]$.

focuses on the mean distance between the point clouds instead. We observe that our approach yields the best result in all scenes again. On average the L_1 Chamfer distance improves by a factor of 1.64 compared to ϕ-SfT and by a factor of 1.99 compared to PG-SfT.

Lastly, we compare the L_1 and L_2 p2s distance for all methods (see Section 2.5.4 for the definitions). This metric does not include any sampling but only considers the closest point on a triangle instead of including the whole interior. In 8 out of 9 scenes our method achieves the best results often beating the other approaches by a factor of more than 2.5 for the L_1 distance and a factor of more than 4 for the L_2 distance. Only in scene R5 PG-SfT performs best. This may come from the uniform motion in this scene which does not expose the weaknesses of their method. However, on average our reconstruction outperform ϕ-SfT [4] and PG-SfT [10] by factors of 3.29 and 2.53 for the L_1 p2s distance and by factors of 3.94 and 3.43 for the L_2 p2s distance.

3.3. Fixed Physical Parameters

We perform a second ablation study that concentrates on the reconstruction quality and stability when the physical material parameters Y, B, and S are not optimized but constant at their initial values. On one hand, these stiffness parameters can not decrease such that the external forces are less likely to become strong enough to crumple the cloth. On the other hand, the optimization loses its ability to adapt to different fabrics automatically. We evaluate the L_2 Chamfer distances in this scenario with and without our regularization terms and collect the results in Table 7. We observe that fixing the physical parameters mainly prevents complete failures without regularizations. Similar to the paper's ablation study, some scenes improve in quality due to the variety in deformations but the full model still reaches the best mean error.

3.4. Real-world Appearance Estimation

We complete the evaluation of the optical material estimation by depicting our results for all real-world scenes in Figure 5a. As a comparison, Figure 5b shows the results when using the imprecisely reconstructed geometry of PG-SfT [10]. We emphasize notable differences between the estimates with red rectangles within the diffuse texture in both figures. Similar characteristics are visible in the roughness and metallic textures. As described in the main paper, the enhanced geometry reconstruction leads to visibly sharper textures in most of the scenes. Especially fine structures like the red striped ovals in scene R3 and the flower stamens in scenes R4 and R5 can not be resolved using the geometry of PG-SfT. The largest difference is visible in the optical material for scene R2. PG-SfT is not able to reconstruct the deformation at all, causing the appearance optimization to fake the appearance by making the fabric highly reflective. The environment maps do not show any significant differences because the dominantly diffuse fabrics only make it possible to recover low-frequency features which are very insensitive to the geometry.

3.5. Synthetic Appearance Estimation

We report the results for the appearance estimation of the synthesized scenes in Figure 6. For all five scenes, we depict the target textures, the estimated textures employing the ground truth motion, and the estimated textures using the reconstructed geometry of our method. The precision of all corresponding geometry reconstructions is given in Table 6. All estimates suffer from the ambiguity that different distributions of color and brightness between diffuse texture and environment map may create the same appearance when observing only limited data, such as a single monocular view. Hence, we mainly compare our results with the ideal case of an estimate based on perfect geometry. The scenes SR2 and SR3 pose a difficult problem to the geometry reconstruction due to their metallic material causing some specular

Method	R1	R2	R3	R4	R5	R6	R7	R8	R9	Mean
ϕ-SfT [4]	7.73	3.96	4.45	12.44	21.87	14.08	11.85	8.25	7.20	10.20
PG-SfT [10]	2.46	5.70	4.94	7.61	**4.52**	20.12	4.05	15.26	5.84	7.83
Ours	**0.70**	**1.42**	**1.58**	**6.80**	6.94	**2.87**	**3.90**	**1.97**	**1.76**	**3.10**

Table 4. L_1 point-to-surface distance $\mathrm{p2s}_1$ between the ground truth point cloud and the reconstructed mesh $[\times 10^{-3}\,\mathrm{m}]$.

Method	R1	R2	R3	R4	R5	R6	R7	R8	R9	Mean
ϕ-SfT [4]	22.65	15.21	9.71	27.38	62.88	30.26	23.78	10.81	8.85	23.50
PG-SfT [10]	3.86	8.24	11.21	17.51	**20.59**	73.12	6.82	30.55	12.38	20.48
Ours	**0.34**	**1.14**	**2.49**	**14.32**	25.26	**2.55**	**4.42**	**1.27**	**1.93**	**5.97**

Table 5. L_2 point-to-surface distance $\mathrm{p2s}_2$ between the ground truth point cloud and the reconstructed mesh $[\times 10^{-5}\,\mathrm{m}^2]$.

Scene	SR1	SR2	SR3	SR4	SR5
CD_2	1.34	1.56	13.7	0.83	0.99

Table 6. 3D Reconstruction quality of the synthesized scenes $[\times 10^{-4}\,\mathrm{m}^2]$.

highlights within the video sequence. As a consequence, the geometry does not fit the target well enough for scene SR3 (see Table 6), resulting in blurred textures at the bottom. Nonetheless, despite little color variations, lots of fine details in the optimized optical material are resolved for scene SR2. The three non-metallic materials (SR1, SR4, and SR5) show very similar results to the estimates that use the ground truth geometry. Especially the two fabric materials in scenes R4 and R5 show sharp textures with very little differences.

References

[1] David Baraff and Andrew Witkin. Large steps in cloth simulation. In *Annual Conference on Computer Graphics and Interactive Techniques*, page 43–54, 1998. 1

[2] Gabriel Cirio, Jorge Lopez-Moreno, David Miraut, and Miguel A. Otaduy. Yarn-level simulation of woven cloth. *ACM TOG*, 33(6), 2014. 1

[3] Jon Hasselgren, Nikolai Hofmann, and Jacob Munkberg. Shape, light, and material decomposition from images using monte carlo rendering and denoising. In *Advances in Neural Information Processing Systems*, pages 22856–22869. Curran Associates, Inc., 2022. 4

[4] Navami Kairanda, Edith Tretschk, Mohamed Elgharib, Christian Theobalt, and Vladislav Golyanik. ϕ-SfT: Shape-from-Template with a Physics-Based Deformation Model. In *CVPR*, pages 3948–3958, 2022. 2, 4, 5, 6, 7

[5] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for high-performance differentiable rendering. *ACM TOG*, 39 (6), 2020. 4

[6] Rahul Narain, Armin Samii, and James F. O'Brien. Adaptive anisotropic remeshing for cloth simulation. *ACM TOG*, 31 (6), 2012. 4

[7] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019. 3

[8] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d, 2020. 4

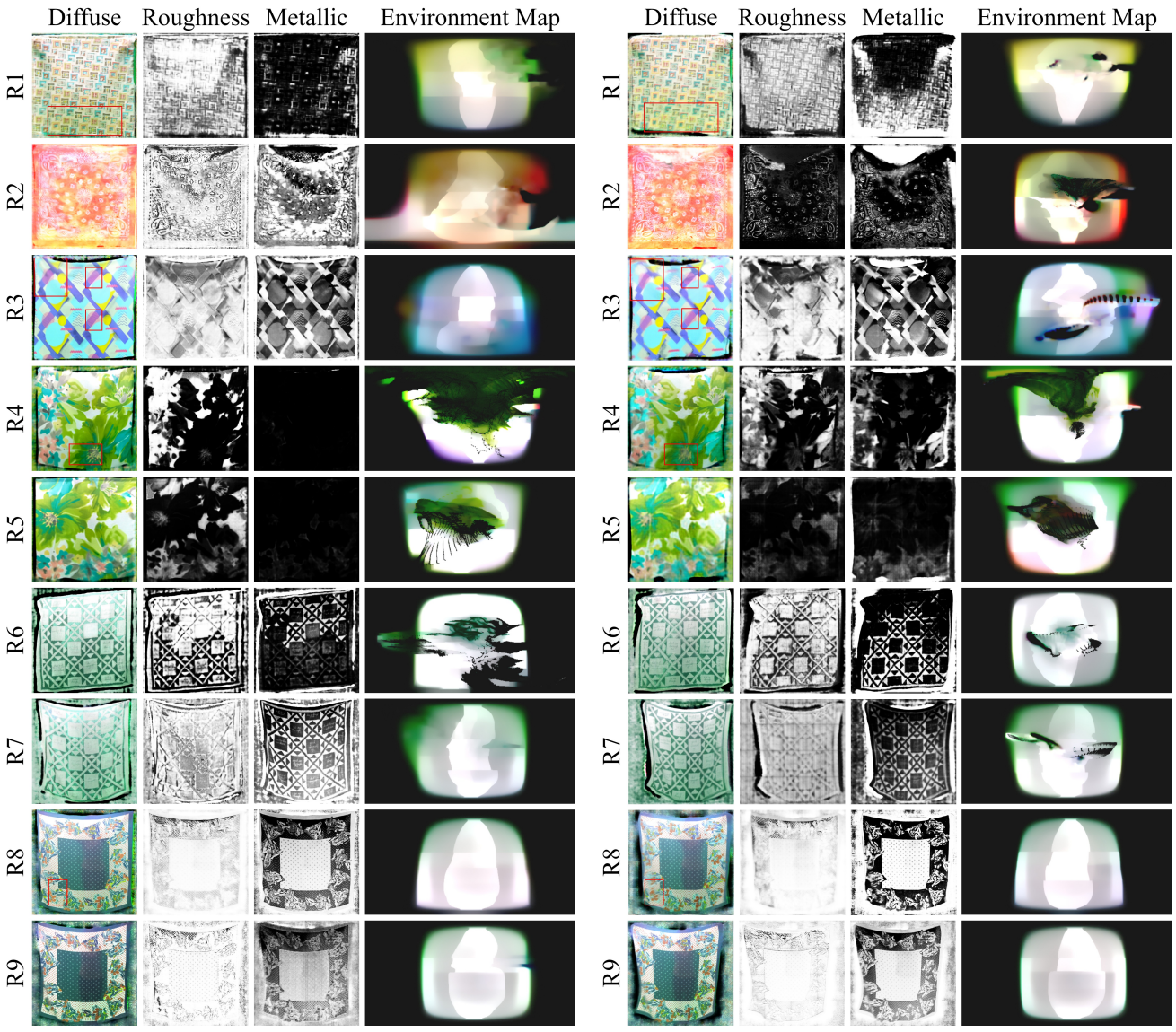
[9] Prem Seetharaman, Gordon Wichern, Bryan Pardo, and Jonathan Le Roux. Autoclip: Adaptive gradient clipping for source separation networks. In *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2020. 3

[10] David Stotko, Nils Wandel, and Reinhard Klein. Physics-guided Shape-from-Template: Monocular Video Perception through Neural Surrogate Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11895–11904, 2024. 4, 5, 6, 7, 8

[11] John M. Sullivan. *Curves of Finite Total Curvature*, pages 137–161. Birkhäuser Basel, Basel, 2008. 1

Experiment	R1	R2	R3	R4	R5	R6	R7	R8	R9	Mean
w/o phys. param.	1.64	4.18	2.31	15.28	7.32	5.24	**3.15**	2.12	1.33	4.73
w/o phys. param. & $\mathcal{R}_F$	5.12	5.47	2.29	14.56	**6.83**	5.45	3.29	2.89	1.18	5.23
w/o phys. param. & $\mathcal{R}_E$	4.77	1.56	4.02	9.79	25.65	3.53	4.73	1.72	1.54	6.37
w/o phys. param. & $\mathcal{R}_F$ & $\mathcal{R}_E$	4.52	4.44	**2.30**	15.70	7.42	5.74	3.24	1.58	1.42	5.15
Full model	**1.01**	**1.55**	2.70	**8.29**	7.12	**2.18**	4.84	**1.23**	**1.04**	**3.33**

Table 7. L_2 Chamfer distances without optimization of physical material parameters compared to the full model $[\times 10^{-4}\,\mathrm{m}^2]$.



(a) Estimation using the geometry of our 3D reconstruction.

(b) Estimation using the geometry of PG-SfT [10].

Figure 5. Estimated appearance parameters of all real scenes. Major differences are highlighted with a red box. Please zoom in for details.

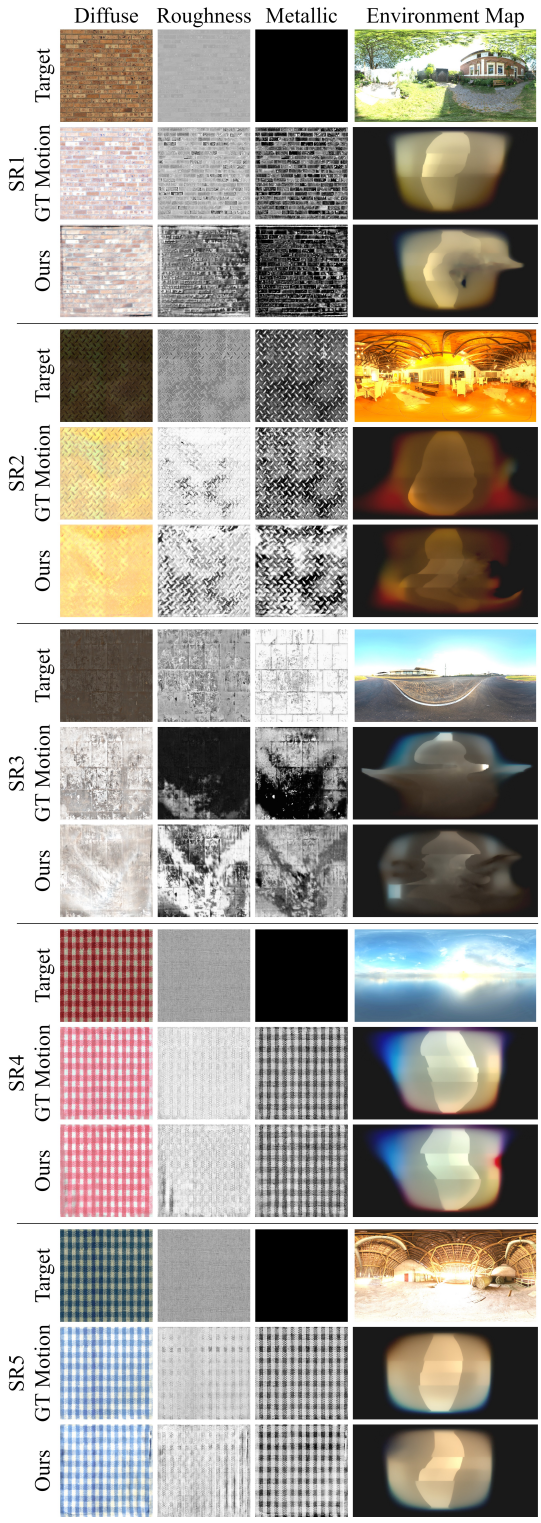


Figure 6. Appearance parameters of all synthesized scenes. The first row depicts the input material textures or the environment map. The second and third row show the estimated appearance maps when using the ground truth geometry or our reconstructed geometry, respectively.