Physics-guided Shape-from-Template: Monocular Video Perception through Neural Surrogate Models

David Stotko Nils Wandel Reinhard Klein

University of Bonn

Abstract

3D reconstruction of dynamic scenes is a long-standing problem in computer graphics and increasingly difficult the less information is available. Shape-from-Template (SfT) methods aim to reconstruct a template-based geometry from RGB images or video sequences, often leveraging just a single monocular camera without depth information, such as regular smartphone recordings. Unfortunately, existing reconstruction methods are either unphysical and noisy or slow in optimization. To solve this problem, we propose a novel SfT reconstruction algorithm for cloth using a pre-trained neural surrogate model that is fast to evaluate, stable, and produces smooth reconstructions due to a regularizing physics simulation. Differentiable rendering of the simulated mesh enables pixel-wise comparisons between the reconstruction and a target video sequence that can be used for a gradient-based optimization procedure to extract not only shape information but also physical parameters such as stretching, shearing, or bending stiffness of the cloth. This allows to retain a precise, stable, and smooth reconstructed geometry while reducing the runtime by a factor of 400–500 compared to ϕ-SfT, a state-of-the-art physics-based SfT approach.

1. Introduction

Shape-from-template (SfT) methods are a practical solution to many reconstruction tasks without the need for expensive hardware setups for capturing scenes. It is possible to reconstruct dynamic geometry from image-based sources like RGB video sequences where neither depth information nor multiple perspectives are given. Nonetheless, some kind of template is provided that represents the object's state at the beginning of the optimization. This strongly constrains the object's size and hence its distance to the camera. However, as each point of the underlying mesh is able to move, there are several hundred or thousand degrees of freedom in moving the object and changing its appearance.

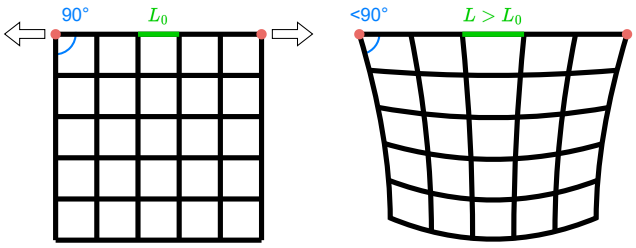

Figure 1. Behavior of stretchable objects like cloth when two anchor points are pulled apart from each other. Neither distances nor angles must remain constant under these deformations.

Most current techniques rely on deformation models to limit the degrees of freedom to e.g. isometric or conformal deformations [2]. In reality, this behavior is only partially fulfilled, as stretchable objects like cloth do not conserve distances or angles. This is easy to see when we look at the sketch in Figure 1 where two points are pulled away from each other, causing a piece of cloth to stretch and shear over the affected area. Such effects are always happening due to anchor points that control the overall acceleration caused by gravity, wind, or other forces. Therefore, it is more realistic to model deformations by performing a physical simulation of the dynamics [20]. Moreover, a stable simulation guarantees a smooth and high-quality reconstruction.

Our approach adopts the general scheme of using a differentiable physics simulation to restrict the object's movement and a differentiable renderer which together allow for a pixel-wise comparison to the target and corresponding gradient-based optimization of input parameters [20]. In contrast to previous methods, we replace the classical cloth simulation [26, 29] with a physics-based neural network. Furthermore, we use nvdiffrast [23] as a fast and differentiable rasterizer together with optimizable uv-coordinates for texture mapping. Our goal with these modifications is to reduce the computation time of the optimization process drastically from many hours to only a few minutes per scene while retaining comparable accuracy for the reconstructed mesh. The source code is available at https://github.com/DavidStotko/Physics-guided-Shape-from-Template.

2. Related Work

Geometry reconstruction The Reconstruction of deformable objects is a difficult task and there are several categories of algorithms that create 3D objects based on the amount of information given and which assumptions are made. Non-rigid structure from motion (NRSfM) algorithms reconstruct deforming geometry like human faces [5, 32], cloth, and similar thin objects [12, 41] or arbitrary deforming objects [42, 51] captured by a static camera. They often incorporate deformation models for isometry, conformality, and other properties or let them be learned by neural networks. SfT additionally makes use of a template (e.g. initial geometry) and/or texture [2, 38]. One approach is the analytic solution of isometric or conformal SfT [2, 6] by solving the corresponding PDEs. Other techniques involve neural networks to learn the shape deformations and reconstruct the geometry [11, 34]. The task of reconstructing geometry might also come with further challenges like occlusions or sparse textures [30]. For a more detailed overview in non-rigid 3D reconstruction, we refer to a recent state-of-the-art report [45].

However, these methods usually lack in physical regularization and realism, as they only apply a deformation model that does not cover the full dynamics. To bypass this problem, a physical simulation can be used to capture deformations and realistic movement at the same time [20]. Similarly, our approach also makes use of a physical simulation but employs a fast physics-based neural network instead of a computationally more expensive classical simulation.

Physics simulation Physical simulations are a valuable tool for a wide variety of tasks and several simulators are available for physical tasks in general [8, 17, 27] and cloth in particular [25, 26, 29, 35]. The idea of constraining the motion to a differentiable high-quality physical simulation that is specified by its physical parameters has already been successfully established in different tasks like the reconstruction of humans, animals and objects [28, 46, 48, 49, 52] and estimating cloth parameters [13, 20, 44]. The downside of these regularizing simulations is their time-consuming complexity which slows down corresponding applications.

A different approach towards differentiable simulations are neural networks, as they are easy to differentiate by construction. For example, neural networks were already successful in tasks like simulating movement trajectories or estimating physical parameters [15, 16]. Extensive research is performed for simulating clothes on human bodies [3, 4, 14, 39, 40] or loose fabrics [21, 24, 31].

To this end, often, supervised learning techniques are employed for which a training dataset is created using classical simulations in order to know the underlying ground truth parameters. This, again, includes the simulation of diverse physical systems [10, 33] and estimation of cloth parameters [9, 19, 36, 50]. Unfortunately, generating a large and high-quality training dataset with traditional physical solvers is computationally expensive [40].

Thus, recent physics-based approaches allow neural networks to learn cloth dynamics solely on the underlying equations of motion without the need for ground truth data [3, 14, 40]. However, to the best of our knowledge, such unsupervised neural surrogate models have not yet been employed to tackle Shape-from-Template tasks.

3. Method

Our method consists of three differentiable parts as shown in Figure 2 in order to infer the shape and physical parameters of cloth in a monocular video sequence. First, physical parameters (Y, S, B), external forces $\vec{F}_{\text{ext}}$ as well as an initial template mesh $\vec{x}_0 = \vec{x}(t_0)$ are fed into a pre-trained physics-based neural cloth model in order to unroll the cloth simulation in time and compute the subsequent cloth shapes $\vec{x}_1, \vec{x}_2, ..., \vec{x}_n$. Second, the updated cloth shapes are used by a differentiable renderer to create a sequence of textured images from a given camera position using an optimized uv-map. The last step compares the rendered images with target frames of a video sequence and backpropagates gradients of a corresponding image loss to update the physical cloth parameters, external forces, and the uv-map.

3.1. Neural Cloth Model

We use a neural surrogate model to obtain fast and differentiable simulations of cloth dynamics. In the following section, we present the underlying physical model, the numerical scheme that is accelerated by our neural network, the networks architecture and a physics-based training strategy.

3.1.1 Physical Cloth Model

Equations of Motion The simulated cloth is represented by a mesh with vertex positions $\vec{x}(t)$, velocities $\vec{v}(t)$, and accelerations $\vec{a}(t)$. The dynamics are described by Newton's second law of motion

$$M\vec{a} = M\frac{\partial^2 \vec{x}}{\partial t^2} = \vec{F}(\vec{x}, \vec{v}) \tag{1}$$

which relates the vertex acceleration $\vec{a}$ and mass M to the force $\vec{F}(\vec{x}, \vec{v})$ acting on it. The force is a superposition of internal cloth forces $\vec{F}_{\text{int}}$ (e.g. stretching and bending), as well as external contributions $\vec{F}_{\text{ext}}$ from gravity and wind.

3.1.2 Implicit Numerical Integration Scheme

Our neural cloth model accelerates a standard numerical integration scheme [1] to solve the equations of motion. To this end, the simulation is modeled in discrete time steps of constant length Δt. Hence, we only compute positions, velocities, accelerations, and forces at discrete times, e.g.

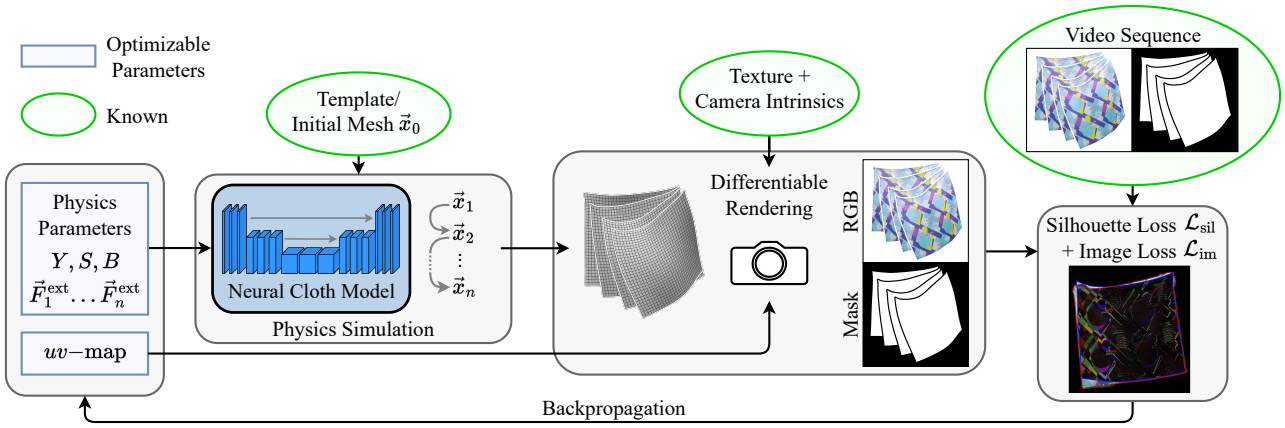

Figure 2. Overview of the optimization loop. A given initial mesh is physically simulated for several time steps by a neural network using physical parameters for stretching Y, shearing S, bending B and external forces $\vec{F}_{\text{ext}}$. The resulting meshes are converted into RGB images and masks by a differentiable renderer together with the known camera intrinsics, texture and an optimizable uv-map. In the end, the renderings are compared to the target video sequence by computing pixel-wise loss functions. Gradients of these losses with respect to the optimizable parameters lead to a successively refined physical simulation and reconstruction.

$\vec{x}_n = \vec{x}(n\Delta t)$. In order to get a stable update scheme, we separate Equation (1) into two first-order differential equations for $\vec{v}$ and $\vec{x}$ and apply the backward Euler method [1]:

$$\vec{v}_{n+1} = \vec{v}_n + \Delta t M^{-1} \vec{F}_{n+1} = \vec{v}_n + \Delta t \vec{a}_{n+1} \quad (2)$$

$$\vec{x}_{n+1} = \vec{x}_n + \Delta t \vec{v}_{n+1} = \vec{x}_n + \Delta t \vec{v}_n + \Delta t^2 \vec{a}_{n+1} \quad (3)$$

Unfortunately, solving this implicit scheme numerically is computationally expensive - even more so, if we want to compute gradients with respect to the solution. Semi-implicit or explicit integration schemes are known for instabilities when simulating the stiff equations of cloth [1] and lose their benefit in runtime when used with tiny time steps. Thus, we decided to train a neural network that learns how to solve for the acceleration $\vec{a}_{n+1}$ based on the current positions, velocities, forces, and cloth parameters. This way, we obtain fast simulations that are naturally differentiable via backpropagation through time.

3.1.3 Network Architecture

The neural cloth model (see Figure 3) is based on a convolutional neural network architecture that maps a rectangular grid of vertex positions $\vec{x}_n$ and velocities $\vec{v}_n$ together with cloth parameters Y, S, B and external forces $\vec{F}_n^{\text{ext}}$ to accelerations $\vec{a}_{n+1}$ in order to update $\vec{v}_{n+1}$ and $\vec{x}_{n+1}$ according to Equations (2) and (3). The CNN block makes use of a U-Net [37] implemented by [18]. We use a gating mechanism that allows the CNN to directly pass $\vec{F}_n^{\text{ext}}$ to $\vec{a}_{n+1}$ which is useful if the cloth is in free fall. If the cloth hangs in a static equilibrium state, $\vec{a}_{n+1} = 0$ and both gates can be closed.

The rectangular grid representation of the cloth was chosen since this structure is similar to woven fabric when simulating cloth at yarn-level [7, 13]. Furthermore, grid computations can be implemented more efficiently on GPUs in comparison to arbitrary graph representations, since no sparse adjacency matrix multiplications are required.

Normalizations In order to resolve ambiguities, we introduce special units of measurement in which our network operates. As the motion of vertices is only affected by the ratio of force and mass, $\vec{a} = M^{-1}\vec{F}$, we fix the mass of interior vertices to be the identity $M = I$ (border and corner vertices are lighter by a factor of 2 and 4 respectively). Moreover, we scale space and time such that edge rest lengths $L_0^{ij} = 1$ (see Equation (7)) and the simulation time steps $\Delta t = 1$.

3.1.4 Physics-based Training Loss $\mathcal{L}_{\text{cloth}}$

The network learns the dynamics of cloth in a self-supervised manner by minimizing a physics-based loss function $\mathcal{L}_{\text{cloth}}$ similar to [3, 40]. This way, we avoid the need of ground truth data from computationally expensive simulators or other sources. The loss function

$$\mathcal{L}_{\text{cloth}} = E_{\text{int}} + \mathcal{L}_{\text{ext}} + \mathcal{L}_{\text{inert}} \quad (4)$$

consists of an internal energy term E_{int} as well as loss terms that reward accelerations in the direction of external forces $\mathcal{L}_{\text{ext}}$ and penalize sudden changes in momentum $\mathcal{L}_{\text{inert}}$.

Internal Energy E_{int} and Forces $\vec{F}_{\text{int}}$

Our cloth model considers a combination

$$E_{\text{int}} = E_Y + E_S + E_B \quad (5)$$

of stretching E_Y, shearing E_S, and bending E_B components for the total internal energy. However, different material models can easily be installed by using appropriate

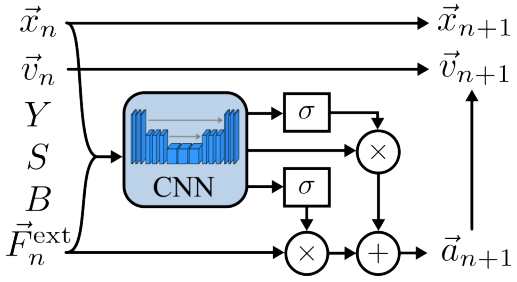

Figure 3. Architecture of the neural cloth model. Detailed explanations are provided in Section 3.1.3.

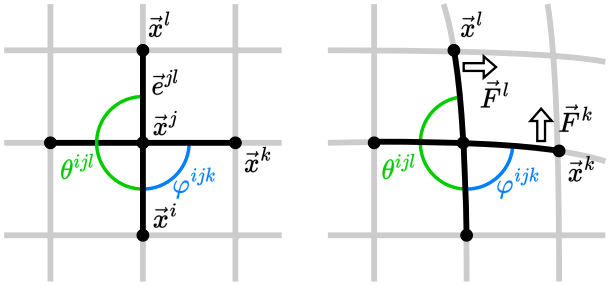

Figure 4. Angles for shearing and bending energies. Straight and corner connections are treated independently to separate in-plane and out-of-plane deformations.

energy functions. The negative gradient of this internal energy term E_{int} with respect to the vertex positions $\vec{x}$ results in forces $\vec{F}_{\text{int}}$:

$$\vec{F}_{\text{int}} = -\frac{\partial E_{\text{int}}}{\partial \vec{x}} \tag{6}$$

Stretching To constrain the length of edges $\vec{e}^{ij}$ between two neighboring vertices $\vec{x}^i$ and $\vec{x}^j$, we penalize deviations from the rest length L_0^{ij} by a Hookean energy term [3]

$$E_Y = \frac{1}{2} Y \left(\|\vec{e}^{ij}\| - L_0^{ij} \right)^2 . \tag{7}$$

The constant Y describes the stretching stiffness, i.e. a weight that determines how hard it is for the system to deviate from the minimal energy state.

Shearing Resistance against angular displacements is taken into account by shearing and bending forces acting on the angle between neighboring edges. A potential can be modeled using the squared difference between the current angle and the target angle [7, 43]. The in-plane angles are restricted by the shearing force between a pair of edges $(\vec{e}^{ik}, \vec{e}^{kj})$, one of which is pointing in warp direction and the other one in weft direction (see Figure 4). For an enclosed angle φ^{ijk}, the energy

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

is scaled by the shearing stiffness S and the target angle φ_0^{ijk} can be set to $\pi/2$ for a planar fabric [7].

Bending A bending loss is calculated analogously for two consecutive edges with both pointing in the same direction (either warp or weft) as depicted in Figure 4. Nonetheless, a bending stiffness B is used to decouple the optimization of both terms such that the bending energy reads

$$E_B = \frac{1}{2} B \left(\theta^{ikl} - \theta_0^{ikl} \right)^2 \tag{9}$$

with rest angles θ_0^{ijk} [40, 43]. In the case of a planar fabric we have $\theta_0^{ijk} \in \{0, \pi\}$, depending on the edge directions. We assume the parameters (Y, S, B) to be the same for all edges and angles in the cloth.

External Forces $\vec{F}_{\text{ext}}$

An arbitrary force field $\vec{F}_{\text{ext}}$ might not be conservative and, thus, cannot be modeled based on the gradient of a potential. However, we can define a loss term $\mathcal{L}_{\text{ext}}$ as

$$\mathcal{L}_{\text{ext}} = -\Delta t^2 \langle \vec{F}_{\text{ext}}, \vec{a} \rangle . \tag{10}$$

$\vec{F}_{\text{ext}}$ can be seen as the effective external forces that act on the grid nodes and result from the superposition of gravity, wind and other forces and may vary over space and time. In principle it is possible for these force vectors to include all internal forces as well but this highly reduces the regularization such that learning almost arbitrary realistic dynamics becomes significantly harder [20].

Inertia

If the neural network only minimizes the previous loss expressions, the cloth will move to the equilibrium state instantly as no contribution restricts how quickly the vertices move. To change that, an inertia term of the form

$$\mathcal{L}_{\text{inert}} = \frac{1}{2} (\Delta t)^2 \langle \vec{a}, M\vec{a} \rangle \tag{11}$$

is added that penalizes momentum changes of the grid [40].

Equations of Motion

By training the neural cloth model, it aims to minimize the loss function $\mathcal{L}_{\text{cloth}}$ (Equation 4) causing its gradient with respect to the network's outputs $\vec{a}_{n+1}$ to vanish:

$$\frac{\partial \mathcal{L}_{\text{cloth}}}{\partial \vec{a}_{n+1}} = \underbrace{\frac{\partial E_{\text{int}}}{\partial \vec{x}_{n+1}}}_{-\vec{F}_{\text{int}}} \underbrace{\frac{\partial \vec{x}_{n+1}}{\partial \vec{a}_{n+1}}}_{\Delta t^2} + \underbrace{\frac{\partial \mathcal{L}_{\text{ext}}}{\partial \vec{a}_{n+1}}}_{-\Delta t^2 \vec{F}_{\text{ext}}} + \underbrace{\frac{\partial \mathcal{L}_{\text{inert}}}{\partial \vec{a}_{n+1}}}_{+\Delta t^2 M \vec{a}_{n+1}} \stackrel{!}{=} 0 \tag{12}$$

Dividing this equation by Δt^2 leads to an implicit scheme for Newton's second law of motion (Equation 1):

$$M\vec{a}_{n+1} = \vec{F}_{\text{int}} + \vec{F}_{\text{ext}} \tag{13}$$

This way, the neural network can learn stable dynamics of cloth directly from the physics-based loss that ensures physical plausibility and no ground truth data is required.

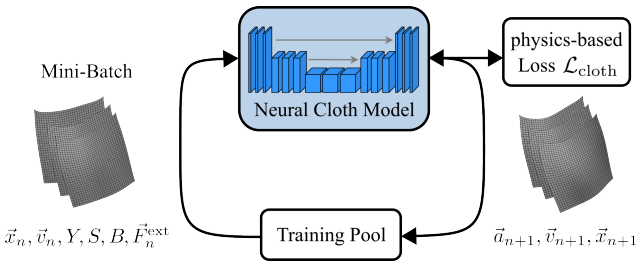

Figure 5. Training cycle of our neural cloth model. Details are provided in Section 3.1.5.

3.1.5 Training Cycle

To train the neural cloth model, we make use of a training cycle similar to [14, 47] (see Figure 5). First, we create a training pool of 5000 cloth states by initializing 32x32 grids at random resting poses ($E_{\text{int}}(\vec{x}_0) = 0, \vec{v}_0 = 0$) with random stretching $Y \in [10, 10000]$, shearing $S \in [0.01, 10]$ and bending $B \in [0.001, 10]$ parameters. After that, we draw a random mini-batch of size 300 containing cloth states $(\vec{x}_n, \vec{v}_n)$, stiffness parameters (Y, S, B), random external forces $\vec{F}_n^{\text{ext}}$ (e.g. gravity and wind), as well as boundary conditions that fasten the top left and right corners of the cloth (see red anchor points in Figure 1). Next, we feed the mini-batch into the neural cloth model in order to predict the accelerations $\vec{a}_{n+1}$ for the next timestep. This allows to compute $\vec{v}_{n+1}$ and $\vec{x}_{n+1}$ based on Equations (2) and (3) and, thus, to evaluate the physics-based loss (Equation 4). Since (Y, S, B) and therefore $\mathcal{L}_{\text{cloth}}$ can vary drastically, we rescale the loss for every batch sample to equal 1. We use the Adam optimizer [22] (the learning rate starts at 10^{-3} and is decreased by $\times 0.2$ after 25 and 50 epochs) to optimize the model.Finally, we feed the predictions of the neural surrogate model back into the training pool to fill it with a larger variety and more realistic training samples. If the physics-based loss of a sample becomes too high, we reset the corresponding cloth sample to a new random initial pose to avoid diverged shapes during the training. Additionally, we randomly reset samples of the training pool from time to time to further increase the variability of training data.

By repeating the training cycle for 100 epochs (approx. 10 hours on a Nvidia GeForce RTX 2080), we obtain a fast and stable neural surrogate model to simulate cloth dynamics that can be used for a wide variety of different stiffness, shearing, bending parameters and offers efficient gradient computations based on backpropagation through time.

3.2. Differentiable Rendering

3.2.1 RGB-Images and Silhouettes

Synthetic images are created using the differentiable rasterizing functionality of nvdiffrast [23]. This requires vertex positions with uv-coordinates and normal vectors, face indices, and a texture file from which everything is known due to an initial template and the physical simulation. The rasterized image is then used for the optimization step. Besides the RGB image, a monochromatic mask is saved as well to optimize the visible silhouette, see Equation (16).

3.2.2 Texture Mapping

The texture for rendering the cloth is extracted from the first frame of a video sequence which might contain deformations or occlusions caused by folds. In order to reduce mismatches in the rendered images for later frames, we optimize the uv-coordinates of each vertex using the information of all video frames. We intentionally decided against optimizing the texture directly because imperfect cloth motion will result in blurry textures and contributions of the (black) background. The effect of this optimization will be analyzed in Section 4.4.

3.3. Shape-from-Template Optimization Loop

Since each part of our pipeline (see Figure 2) is differentiable, gradients can be propagated throughout the entire optimization via backpropagation through time. This way, the cloth shape, its physical parameters, and external forces can be estimated by minimizing the difference between rendered and real-world video frames with gradient descent.

3.3.1 Shape-from-Template Loss

To optimize the shape of the cloth template, we minimize a loss that combines an image loss $\mathcal{L}_{\text{im}}$, silhouette loss $\mathcal{L}_{\text{sil}}$ and a regularization term for external forces R_T:

$$\mathcal{L}_{\text{SfT}} = \mathcal{L}_{\text{im}} + \mathcal{L}_{\text{sil}} + R_T \tag{14}$$

Image Loss

The image loss averages the pixel-wise differences between the ground truth RGB video frames $\vec{I}_i$ and the generated images $\hat{\vec{I}}_i$ from the simulation:

$$\mathcal{L}_{\text{im}} = \frac{1}{N_{\text{p}}} \sum_{i=1}^{N_{\text{p}}} \|\hat{\vec{I}}_i - \vec{I}_i\|_1 \tag{15}$$

Silhouette Loss

As the dataset provides masks for each frame, we also create a silhouette loss. Therefore, the masks of the ground-truth M and rendered image $\hat{M}$ are blurred by a Gaussian kernel G with standard deviation $\sigma = 7$ pixels [20] resulting in smooth gradients for the pixel-wise difference. Again, the corresponding loss function is calculated as the mean of the pixel-wise differences of the blurred masks:

$$\mathcal{L}_{\text{sil}} = \frac{1}{N_{\text{p}}} \sum_{i=1}^{N_{\text{p}}} \|G(\hat{M}_i, \sigma) - G(M_i, \sigma)\| \tag{16}$$

Regularization of External Forces

We separate the external forces $\vec{F}_{\text{ext}} = \vec{w} + \vec{T}$ into a constant part $\vec{w}$ (e.g. gravity and constant wind) and a spatially and temporarily varying part $\vec{T}$ (e.g. turbulent wind). This way, we are able to assign distinct learning rates for $\vec{w}$ and $\vec{T}$ and focus regularization on $\vec{T}$ to keep external forces stable and smooth. To this end, we penalize the turbulent part T_t^{ij} acting at time step t on a vertex with grid indices i, j for its length as well as its temporal and spatial changes:

$$R_T = \alpha \sum_{t,i,j} \|\vec{T}_t^{ij}\|^2 + \beta \sum_{t,i,j} \|\vec{T}_{(t+1)}^{ij} - \vec{T}_t^{ij}\|^2 + \gamma \sum_{t,i,j} \|\vec{T}_t^{(i+1)j} - \vec{T}_t^{ij}\|^2 + \|\vec{T}_t^{i(j+1)} - \vec{T}_t^{ij}\|^2 \quad (17)$$

In our optimization, we set $\alpha = \gamma = 10^{-2}$ and $\beta = 10^{-3}$.

3.3.2 Optimization

Our optimization loop starts with a subset of the first video frames, similar to ϕ-SfT [20], and successively adds frames depending on the number of optimization cycles. This procedure ensures that new dynamics are introduced by new frames only when the previous states match the desired movement. In our case, we start the optimization with the first 10 frames of a video and add the next frame every 5 optimization cycles until the whole video is considered.

At the beginning of the optimization loop, the model starts with the same physical parameters for all scenes. The stiffness parameters (Y, S, B) begin at values $(3000, 8, 0.5)$ and their learning rates are set to $(50, 0.1, 0.01)$. In order to ensure stable simulations, the stiffness parameters are bounded by minimal values $(10, 0.01, 10^{-5})$. External forces are initialized by the gravitational field $\vec{g}$ only, i.e. $\vec{w} = \vec{g}$, without additional wind or turbulences ($\vec{T} = 0$). The learning rates are set to $0.05 \cdot \|\vec{g}\|$ and 0.001 for constant and turbulent force components respectively. During optimization, the component along the direction of gravity is fixed such that wind can only appear in horizontal directions. The uv-coordinates are initialized via the known template mesh and updated with a learning rate of $2 \cdot 10^{-4}$.

4. Evaluation

We evaluate our method on the ϕ-SfT dataset [20] which provides several video sequences of real fabrics with diverse movement. The templates are adjusted to match the requirements of our network, i.e. the geometry is remeshed such that it is represented by a regular 32×32 grid. We perform a qualitative analysis of the 3D meshes and compare the rendered images to the target images. For a quantitative result, we evaluate the precision of the reconstruction as well as the time to converge and compare it with results of ϕ-SfT [20], as it is the state-of-the-art SfT method that uses a physical simulation to regularize the object's movement. Moreover, we investigate the stability of our algorithm by optimizing for thousands of iterations. Finally, we perform an ablation study that shows the importance of various features and describe the limitations of our method.

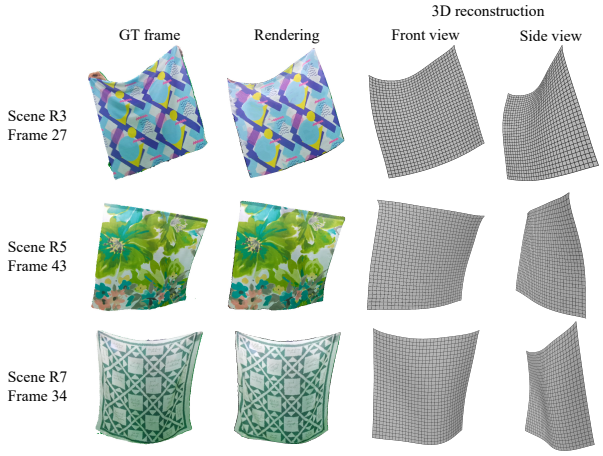

Figure 6. Our approach follows the expected movement in the video and produces smooth 3D geometry for the cloth. Diverse dynamics from manual movement and wind are captured well.

4.1. Qualitative Evaluation

We evaluate the performance of our method on a dataset of masked images and silhouettes containing several scenes with diverse movements. First, we look at the reconstructions of our approach qualitatively. Figure 6 depicts the ground truth frame, the rendered image and two views of the reconstructed mesh for late frames in three example scenes. Generating novel views is no problem due to the physics simulation that generates realistic 3D geometry. Especially scenes R3 and R5 contain intricate movement due to folding and manual movement. Nonetheless, our algorithm is able to follow this movement and produce a smooth and stable 3D reconstruction; see our supplementary video for more details. This behavior is not always the case for SfT and NRSfM methods since assumptions like isometry or conformality do not suffice to prevent the geometry from having an unpleasant and noisy-looking surface [20, 32, 51]. The authors of ϕ-SfT already showed that other SfT and NRSfM methods usually do not produce smooth and realistic reconstructions and thus a physical simulation is needed to achieve pleasing results [20]. For this reason, we focus on comparing our method to ϕ-SfT.

4.2. Quantitative Evaluation

We measure the precision and runtime of our method and compare it to the performance of ϕ-SfT [20]. All evaluations are performed on an Nvidia A100 GPU and an AMD Epyc 7713 CPU.

Method	R3	R4	R5	R7	R8	R9
ϕ-SfT	**7.9**	**10.3**	14.4	9.1	**3.7**	**2.9**
Ours	12.5	14.5	**11.7**	**6.9**	10.1	8.6
Ratio $\frac{C_{\text{Ours}}}{C_{\phi-\text{SfT}}}$	1.59	1.41	0.81	0.76	2.70	2.94

Table 1. Quantitative comparison using the symmetric Chamfer distance C. All values are multiplied by 10^4 for readability.

Method	R3	R4	R5	R7	R8	R9
ϕ-SfT	1204	1453	1152	1065	1186	1157
Ours	**3.07**	**2.48**	**3.03**	**2.58**	**2.55**	**2.47**
Speedup	393	585	380	412	465	469

Table 2. Runtime comparison between ϕ-SfT [20] and our method. All numbers represent the runtime for the optimization loop until convergence in minutes.

The dataset provides different scenes with pseudo ground truth data in form of a target point cloud T per frame. Therefore, we create a reconstruction point cloud R by sampling the mesh uniformly (each mesh triangle weighted by its area) with the same number of points as the target point cloud. The reconstruction quality is then measured by evaluating the symmetric Chamfer distance:

$$C(R,T) = \frac{1}{|R|}\sum_{\vec{r}\in R}\min_{\vec{t}\in T}\|\vec{r}-\vec{t}\|_2^2 + \frac{1}{|T|}\sum_{\vec{t}\in T}\min_{\vec{r}\in R}\|\vec{r}-\vec{t}\|_2^2 \tag{18}$$

We use the reference implementation of ϕ-SfT with default parameters (e.g. 300 optimization cycles) and compare it to our approach after 250 cycles. Unfortunately, the optimization of scenes R1, R2 and R6 did not finish with this ϕ-SfT implementation and thus they are discarded for evaluation. Table 1 shows the evaluated Chamfer distance between the reconstruction and the pseudo ground truth point cloud on real scenes of the ϕ-SfT dataset. Both algorithms perform with comparable quality and always produce high-quality results, although ϕ-SfT matches the target better in 4 out of 6 examples. Especially in scenes R8 and R9 the results of ϕ-SfT are very close to the expectations. Nonetheless, our neural model performs better in scenes R5 and R7.

The main goal of our approach and its difference to ϕ-SfT [20] lies in the runtime. Due to the neural network, we are able to perform the physics simulation much faster than a classical algorithm could do. Furthermore, nvdiffrast [23] is significantly faster in rendering images than Pytorch3D. Table 2 shows the runtime of both approaches for all scenes in minutes. In summary, ϕ-SfT needs between 17:45 h and 24:12 h for optimizing a single scene. Compared to that, our method only needs between 2.5 and 3 minutes per scene. This is a speedup of a factor between 380 and 585. The time for training our physics-based network is not included because this needs to be done only once in advance and the same network is then reused for all scenes.

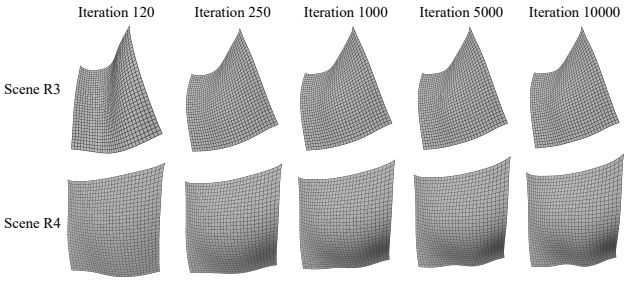

Figure 7. Stability test for R3 and R4 with significant movement. We show a side view of the reconstructed mesh at 120 (before convergence), 250 (regular evaluation), 1000, 5000 and 10000 epochs.

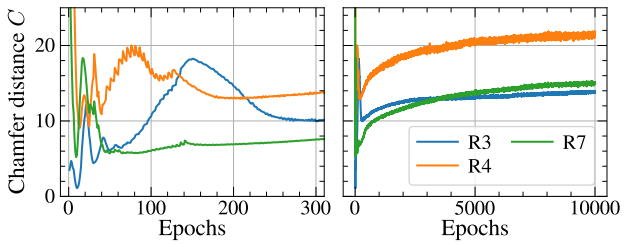

Figure 8. Variation in Chamfer distance C during optimization epochs. After 400 epochs, overfitting worsens the reconstruction quality but does not affect the stability of the overall optimization. The remaining scenes experience very similar behavior.

4.3. Stability

We also investigate the stability of our approach by running the optimization loop 40 times longer than usual. Figure 7 depicts reconstructed meshes from two scenes with diverse movement at different epochs during the optimization loop. It can be seen that our reconstructed mesh still follows the desired movement from the video and does not change significantly after the regular 250 epochs are reached. Moreover, Figure 8 shows how the Chamfer distance C varies during the epochs. Overfitting worsens the Chamfer distance when our model optimizes for too long but does not affect the stability. Compared to that, when optimizing for too long, ϕ-SfT [20] often suffers from instabilities that greatly decrease the reconstruction quality or even lead to a completely different movement than expected.

4.4. Ablation Study

To analyze the importance of each feature in our approach, we perform an ablative study in which we remove one part at a time and compare the results to the full model. Table 3 comprises variants that miss either the silhouette loss $\mathcal{L}_{\text{sil}}$, turbulent forces T_t^{ij}, their regularization R_T, the uv-optimization, or the successive optimization scheme.

Ablation	R3	R4	R5	R7	R8	R9
$\mathcal{L}_{\text{sil}}$	13.8	14.7	12.2	7.1	10.7	8.7
R_T	12.6	14.8	11.9	7.0	10.4	**8.5**
T_t^{ij}	21.0	45.9	172.4	61.9	31.5	14.7
uv-map	13.3	24.1	14.5	61.9	20.1	30.9
Suc.-opt.	**9.7**	19.0	12.4	19.0	13.8	13.6
Full	12.5	**14.5**	**11.7**	**6.9**	**10.1**	8.6

Table 3. Reconstruction quality when one feature in our algorithm is removed. Small variations can also occur due to randomness.

The silhouette loss only slightly improves the results in all scenes. Unsurprisingly, the largest difference is present in scene R3 where one of the two holding points is moving significantly and causing the silhouette to change much more than in other scenes. Removing the regularization R_T also does not change the quality significantly within the given number of iterations. The simulating network itself processes the external forces and regularizes their effect due to convolutional layers but we want to control them explicitly (also regarding the long-time behavior). However, when turbulent forces are neglected completely, large mismatches are caused. These turbulent forces are not only used to create local deformations that only affect a few vertices but also for moving the anchor points. Especially in scene R5 the anchor points create the majority of the dynamics. The uv-map optimization has a large effect on the reconstruction precision because the texture is taken from the first RGB frame. This might include perspective effects and occlusions that introduce a bias into the rendering (see Section 3.2.2). Finally, we omit the successive optimization scheme, i.e. we optimize with all frames from the beginning. Again, we observe a decreasing reconstruction quality in all scenes except R3.

4.5. Limitations

Fine wrinkles Figure 9 shows examples in which our network is not able to simulate fine wrinkles and reconstruct corresponding details. This is visible in folds (scene R3) and high-frequency movement from wind turbulences (R4). Possible solutions are a higher grid resolution or different physical energy terms for shearing and bending forces.

uv-mapping The ablation study shows that optimizing the uv-map significantly improves the precision of the reconstructed mesh. However, the ability of warping the texture by need also opens the possibility of creating distorted texture mappings. Especially small details and geometric patterns can suffer from that effect. Three examples are shown in Figure 10 depicting the optimized cloth of scenes R3, R7 and R9. The cloth textures in scenes R3 and R7 contain several straight lines that get smeared or shifted due to the movement. Nevertheless, most parts of the uv-map barely changed even without a regularization such that intricate patterns or words (scene R7) are still intact. These Issues could be solved by regularizing the uv-coordinates or by optimizing the texture directly. Other methods do not optimize the uv-map at all and keep potential deformations from the first frame which leads to worse reconstructions.

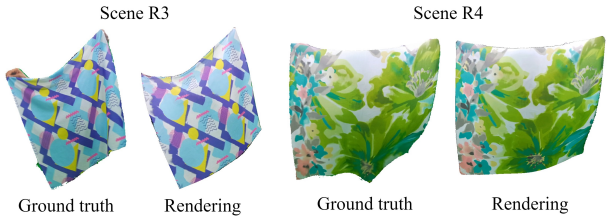

Figure 9. Our network does not capture high-curvature details like sharp folds very well. Such effects occur due to manual movement in scene R3 or wind in R4.



Figure 10. Optimizing the uv-map leads to some artifacts (red boxes) in the texture which are most noticeable when the texture contains regular geometric structures like straight lines or dots.

5. Conclusion

We presented a novel Shape-from-Template method that reconstructs the 3D geometry of a piece of cloth together with physical parameters for stretching, shearing, and bending based on a single monocular RGB video sequence and a template mesh. We employ a physics-based neural network that enables fast and stable physical simulation without the need of costly classical simulation methods. This simulation regularizes possible dynamics of the 3D geometry for the optimization process. Our method was compared to the state-of-the-art physics-based ϕ-SfT method and achieved comparable results with a speedup of 400–500 times.

Acknowledgements

This work has been funded by the DFG project KL 1142/11-2 (DFG Research Unit FOR 2535 Anticipating Human Behavior), by the Federal Ministry of Education and Research of Germany and the state of North-Rhine Westphalia as part of the Lamarr-Institute for Machine Learning and Artificial Intelligence and the InVirtuo 4.0 project, and additionally by the Federal Ministry of Education and Research under grant no. 01IS22094E WEST-AI.

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. 2, 3

[2] Adrien Bartoli, Yan Gérard, François Chadebecq, Toby Collins, and Daniel Pizarro. Shape-from-template. *IEEE TPAMI*, 37(10):2099–2118, 2015. 1, 2

[3] Hugo Bertiche, Meysam Madadi, and Sergio Escalera. Pbns: Physically based neural simulator for unsupervised garment pose space deformation. *CoRR*, abs/2012.11310, 2020. 2, 3, 4

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

[5] C. Bregler, A. Hertzmann, and H. Biermann. Recovering non-rigid 3d shape from image streams. In *CVPR*, pages 690–696 vol.2, 2000. 2

[6] David Casillas-Perez, Daniel Pizarro, David Fuentes-Jimenez, Manuel Mazo, and Adrien Bartoli. The isowarp: the template-based visual geometry of isometric surfaces. *IJCV*, 129(7):2194–2222, 2021. 2

[7] Gabriel Cirio, Jorge Lopez-Moreno, David Miraut, and Miguel A. Otaduy. Yarn-level simulation of woven cloth. *ACM TOG*, 33(6), 2014. 3, 4

[8] Tao Du, Kui Wu, Pingchuan Ma, Sebastien Wah, Andrew Spielberg, Daniela Rus, and Wojciech Matusik. Diffpd: Differentiable projective dynamics. *ACM TOG*, 41(2), 2021. 2

[9] Xudong Feng, Wenchao Huang, Weiwei Xu, and Huamin Wang. Learning-based bending stiffness parameter estimation by a drape tester. *ACM TOG*, 41(6), 2022. 2

[10] Meire Fortunato, Tobias Pfaff, Peter Wirnsberger, Alexander Pritzel, and Peter Battaglia. Multiscale meshgraphnets. In *ICML Workshop*, 2022. 2

[11] 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

[12] 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*, pages 51–72. Springer International Publishing, 2018. 2

[13] D Gong, Z Zhu, A Bulpitt, and H Wang. Fine-grained differentiable physics: a yarn-level model for fabrics. In *ICLR*, 2022. 2, 3

[14] Artur Grigorev, Michael J. Black, and Otmar Hilliges. Hood: Hierarchical graphs for generalized modelling of clothing dynamics. In *CVPR*, pages 16965–16974, 2023. 2, 5

[15] Eric Heiden, David Millard, Erwin Coumans, Yizhou Sheng, and Gaurav S. Sukhatme. Neuralsim: Augmenting differentiable simulators with neural networks. In *IEEE ICRA*, pages 9474–9481, 2021. 2

[16] Florian Hofherr, Lukas Koestler, Florian Bernard, and Daniel Cremers. Neural implicit representations for physical parameter inference from a single video. In *WACV*, pages 2093–2103, 2023. 2

[17] Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Fredo Durand. Difftaichi: Differentiable programming for physical simulation. In *ICLR*, 2020. 2

[18] Pavel Iakubovskii. Segmentation models. https://github.com/qubvel/segmentation_models, 2019. 3

[19] Eunjung Ju and Myung Geol Choi. Estimating cloth simulation parameters from a static drape using neural networks. *IEEE Access*, 8:195113–195121, 2020. 2

[20] 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. 1, 2, 4, 5, 6, 7

[21] Navami Kairanda, Marc Habermann, Christian Theobalt, and Vladislav Golyanik. Neuralclothsim: Neural deformation fields meet the kirchhoff-love thin shell theory. *arXiv:2308.12970*, 2023. 2

[22] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2014. 5

[23] 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. 1, 5, 7

[24] Tae Min Lee, Young-Jin Oh, and In-Kwon Lee. Efficient cloth simulation using miniature cloth and upscaling deep neural networks. *CoRR*, 2019. 2

[25] 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

[26] Junbang Liang, Ming Lin, and Vladlen Koltun. Differentiable cloth simulation for inverse problems. In *NeurIPS*. Curran Associates, Inc., 2019. 1, 2

[27] Miles Macklin. Warp: A high-performance python framework for gpu simulation and graphics. https://github.com/nvidia/warp, 2022. NVIDIA GPU Technology Conference (GTC). 2

[28] Mariem Mezghanni, Théo Bodrito, Malika Boulkenafed, and Maks Ovsjanikov. Physical simulation layer for accurate 3d modeling. In *CVPR*, pages 13514–13523, 2022. 2

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

[30] 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

[31] Young Jin Oh, Tae Min Lee, and In-Kwon Lee. Hierarchical cloth simulation using deep neural networks. In *CGI*, page 139–146. ACM, 2018. 2

[32] Shaifali Parashar, Mathieu Salzmann, and Pascal Fua. Local non-rigid structure-from-motion from diffeomorphic mappings. In *CVPR*, 2020. 2, 6

[33] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based simulation with graph networks. In *ICLR*, 2020. 2

[34] Albert Pumarola, Antonio Agudo, Lorenzo Porzi, Alberto Sanfeliu, Vincent Lepetit, and Francesc Moreno-Noguer.

Geometry-aware network for non-rigid shape prediction from a single view. In *CVPR*, 2018. 2

[35] Yi-Ling Qiao, Junbang Liang, Vladlen Koltun, and Ming C Lin. Scalable differentiable physics for learning and control. In *ICML*, pages 7847–7856, 2020. 2

[36] Carlos Rodriguez-Pardo, Melania Prieto-Martin, Dan Casas, and Elena Garces. How will it drape like? capturing fabric mechanics from depth images. *Computer Graphics Forum*, 42(2):149–160, 2023. 2

[37] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241. Springer International Publishing, 2015. 3

[38] Mathieu Salzmann, Vincent Lepetit, and Pascal Fua. Deformable surface tracking ambiguities. In *CVPR*, pages 1–8, 2007. 2

[39] 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*, pages 11763–11773, 2021. 2

[40] Igor Santesteban, Miguel A. Otaduy, and Dan Casas. Snug: Self-supervised neural dynamic garments. In *CVPR*, pages 8140–8150, 2022. 2, 3, 4

[41] Soshi Shimada, Vladislav Golyanik, Christian Theobalt, and Didier Stricker. Ismo-gan: Adversarial learning for monocular non-rigid 3d reconstruction. In *CVPRW*, 2019. 2

[42] Vikramjit Sidhu, Edgar Tretschk, Vladislav Golyanik, Antonio Agudo, and Christian Theobalt. Neural dense non-rigid structure from motion with latent space constraints. In *ECCV*, 2020. 2

[43] John M. Sullivan. *Curves of Finite Total Curvature*, pages 137–161. Birkhäuser Basel, Basel, 2008. 4

[44] Priya Sundaresan, Rika Antonova, and Jeannette Bohgl. Diffcloud: Real-to-sim from point clouds with differentiable simulation and rendering of deformable objects. In *IROS*, pages 10828–10835, 2022. 2

[45] Edith Tretschk, Navami Kairanda, Mallikarjun B R, Rishabh Dabral, Adam Kortylewski, Bernhard Egger, Marc Habermann, Pascal Fua, Christian Theobalt, and Vladislav Golyanik. State of the art in dense monocular non-rigid 3d reconstruction. *Comput. Graph. Forum*, 42(2):485–520, 2023. 2

[46] Shashank Tripathi, Lea Müller, Chun-Hao P. Huang, Omid Taheri, Michael J. Black, and Dimitrios Tzionas. 3d human pose estimation via intuitive physics. In *CVPR*, pages 4713–4725, 2023. 2

[47] Nils Wandel, Michael Weinmann, and Reinhard Klein. Unsupervised deep learning of incompressible fluid dynamics. *CoRR*, abs/2006.08762, 2020. 5

[48] Sirui Xu, Zhengyuan Li, Yu-Xiong Wang, and Liang-Yan Gui. Interdiff: Generating 3d human-object interactions with physics-informed diffusion. In *ICCV*, pages 14928–14940, 2023. 2

[49] Gengshan Yang, Shuo Yang, John Z. Zhang, Zachary Manchester, and Deva Ramanan. Ppr: Physically plausible reconstruction from monocular videos. In *ICCV*, pages 3914–3924, 2023. 2

[50] Shan Yang, Junbang Liang, and Ming C. Lin. Learning-based cloth material recovery from video. In *ICCV*, 2017. 2

[51] Rui Yu, Chris Russell, Neill D. F. Campbell, and Lourdes Agapito. Direct, dense, and deformable: Template-based non-rigid 3d reconstruction from rgb video. In *ICCV*, 2015. 2, 6

[52] Ye Yuan, Jiaming Song, Umar Iqbal, Arash Vahdat, and Jan Kautz. Physdiff: Physics-guided human motion diffusion model. In *ICCV*, pages 16010–16021, 2023. 2