

Image-Guided Shape-from-Template Using Mesh Inextensibility Constraints

Supplementary Material

6. Deformation Network

The object deformation at time step t can be represented by $\varphi(\mathbf{x}_0, t; \mathbf{z}_t) = \mathbf{x}_t - \mathbf{x}_0$, for the initial state \mathbf{x}_0 and latent variables \mathbf{z}_t . We can learn such deformation in the following ways:

- (1) **vertex-offset**: consider $\varphi := \Delta \mathbf{x}_t \in \mathbb{R}^{3V}$ as learning variables directly,
- (2) **physics-based**: sequentially predict the per-time-step acceleration \mathbf{a}_t and compute the shape using backward Euler method

$$\begin{aligned}\mathbf{v}_t &= \mathbf{v}_{t-1} + \Delta t \mathbf{a}_t, \\ \mathbf{x}_t &= \mathbf{x}_{t-1} + \Delta t \mathbf{v}_t,\end{aligned}$$

- (3) **shared-weight network**: learn a network $\varphi(\mathbf{x}_0, t; \theta)$ with the shared parameters θ and timestep t as input,
- (4) **frame-wise network** (*ours*): learn a network $\varphi(\mathbf{x}_0; \theta_t)$ with distinct parameters θ_t for each timestep t .

The first idea is intuitive, on which we made an ablation study (see Figure 6) to show that direct estimation may lack smoothness and needs additional regularizers. The state-of-the-art physics-based methods, to which we compare in Section 4, follow the second way. The third approach are widely used in Neural Radiance Fields (NeRF) [29] and Gaussian Splatting (GS) [53], which should optimize for the entire video concurrently so that the input t is informative. In our initial exploration, we modeled the SFT problem as an initial value problem and utilized neural ODE framework [7] to learn the sequence’s velocity field. We found it to be as computationally inefficient as ϕ -SfT. Our method, on the other hand, allows optimizing in a frame-wise strategy described in Section 3.6. This brings the novelty in our work, where we can reconstruct high-quality shapes efficiently.

7. Loss Functions

7.1. Adaptive Data Loss

Rather than using the standard ℓ -p norm for the vision losses as ϕ -SfT [17] or PGSfT [49], we propose to use an adaptive data loss as described in Section 3.3. Its adaptive nature comes from the exponential weighting factor on the pixel difference d , which allows color errors to remain in focus during optimization despite shading variations. Using this loss is particularly beneficial in complex videos with low-textured areas and/or strong deformations. Indeed, the quantitative results in Table 5 showed a significant improvement on complex sequences while slightly degradation (visually indistinguishable) on the simpler ones. Moreover,

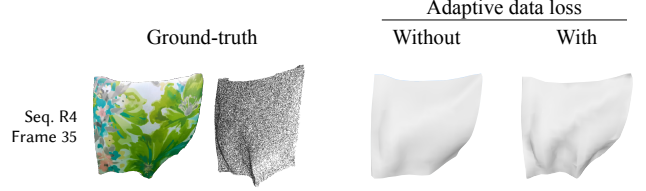


Figure 8. Qualitative comparison on using the adaptive data loss.

Table 7. Ablation study on hyper-parameters of the adaptive data loss. Average chamfer distance ($\times 10^4$) are reported in average of all sequences on ϕ -SfT real dataset.

$\sigma \setminus \alpha$	0.1	1	5	10	15
0.1	5.59	17.91	28.22	28.86	26.73
0.5	18.43	4.85	4.01	4.22	4.76
1	18.89	4.95	3.83	3.91	4.23
2	19.91	4.75	3.93	3.95	4.18
10	27.87	4.81	3.95	3.92	4.23

as illustrated in Figure 8, our method is able to reconstruct very high-detailed wrinkles with the adaptive data loss.

The adaptive data loss is computed with an exponential weighting factor $w(d) = \alpha \exp(d/\sigma)$, where we fix $\alpha = 10$ and $\sigma = 1$. To investigate the robustness of the choice of these hyper-parameters, we evaluate the reconstruction results on ϕ -SfT real dataset with ranges of $\alpha \in \{0.1, 1, 5, 10, 15\}$ and $\sigma \in \{0.1, 0.5, 1, 2, 10\}$, then report the average of all sequences in Table 7. First, we can see that we obtain bad reconstruction when $\sigma = 0.1$. Small σ makes the weight and thus the adaptive data loss exponentially large which can cause optimization to fail. Next, when $\alpha = 0.1$, we also get bad results since the optimizer might lose the importance of vision losses. Finally, when α and σ are bigger, the results are quite robust compared to our default setting.

7.2. Silhouette Loss

We inherited the silhouette loss from the related works ϕ -SfT [17] and PGSfT [49]. Hence, the object masks are required in order to optimize the shapes. In case the ground-truth masks are not given, we can estimate them by using a segmentation network, as shown our pipeline (see Figure 2). The Kinect paper dataset does not contain object masks, and we used SAM 2 in our experiments. We additionally report in Table 8 the impact of using silhouette loss on ϕ -SfT real dataset. In general, we can see that the silhouette loss helps improve the reconstruction performance.

Table 8. Ablation study evaluating the impact of removing silhouette loss terms on our performance. Average chamfer distance ($\times 10^4$) on ϕ -SfT real dataset is reported.

Seq.	No silhouette loss	Default
R1	0.71	0.66
R2	1.04	1.30
R3	4.94	4.49
R4	6.57	8.15
R5	8.68	8.04
R6	3.56	3.37
R7	4.79	4.62
R8	3.33	2.76
R9	2.18	2.12
Avg.	3.97	3.91

7.3. Mesh Inextensibility Loss

We note that the mesh inextensibility loss is essential in our method. Learning shapes without such a regularization, the object shapes can be overstretched out of viewing frame, thus misleading the optimization in subsequent iterations. This is similar to using physics constraints as ϕ -SfT [17] and PGSfT [49], and bypassing the optimization or prior of cloth’s material coefficients. Although classical constraints as-rigid-as-possible (ARAP) [33] or edge preservation can also be imposed considering their computational efficiency, they are still prone to high errors when we have strong deformations or severe occlusion. As suggested by Chen et al. [6], the mesh inextensibility loss relaxes isometry to a weaker form of local rigidity of a point within a neighborhood via preservation of singular values in Equations (7) to (9). We refer to their paper [6] for more discussion on the loss.

The mesh inextensibility loss is originally designed to garment simulation, in which loss variations are not quite large. On the other hand, SfT problem aims to reconstruct various object categories where the inextensibility losses vary a lot. Hence, selecting the weighting factor w_{inext} is essential for our framework to generalize to any object of interest. We observe that the mesh inextensibility loss depends on the mesh scale. Indeed, let us define δ the scale of the object mesh. We imply that the vector $x_j - \bar{x}_i$ also has scale δ . It follows that the covariance matrices in Equations (8) and (9) have the scale δ^2 . Therefore, the mesh inextensibility loss in Equation (7) has scale $\delta^6 = (\delta^2)^3$ due to the determinant computation of 3×3 matrices. In our experiments, we thus fix an adaptive weighting factor $w_{\text{inext}} := \hat{\delta}^{-6}$, where $\hat{\delta}$ approximates the mesh scale by evaluating the median of the edge lengths of the mesh. Since the weighting factors for vision losses are fixed to 1, integrating them into our framework yields an adaptive manner for

reconstructing various 3D objects without extra tuning.

8. Optimization Strategies

We analyze the complexity of the proposed frame-wise optimization strategy in Section 8.1 in comparison with the strategies from physics-based approaches. Additionally, we discuss some possible extensions in Sections 8.2 and 8.3. Depending on the practical applications, future works can develop from these fundamentals to balance between the accuracy and efficiency. We make an ablation study of such strategies on the Kinect Paper dataset in Section 8.4, showing promising results on reconstructing high inter-frame motion.

8.1. Frame-wise Optimization

We compare our proposed optimization strategy in Sec. 3.6 with the two state-of-the-art methods ϕ -SfT [17] and PGSfT [49]. A conspicuous bottleneck for image-guided SfT is rendering and backpropagation through the differentiable renderer, especially when the mesh has a high resolution. We show in the following that our strategy reduces the number of renderings from $O(T^2N)$ to $O(TN)$, where T is the video sequence length and N is the average number of iterations used to optimize a single frame.

Let us first analyze the number of renderings used in ϕ -SfT [17] and PGSfT [49]. These methods rely on the simulation of cloth under Newtonian dynamics. Hence, their proposed optimization strategies are inherently incremental, which requires rendering of prior frames when optimizing the subsequent ones to avoid error accumulation. Assume that the learning process starts with the first frame and gradually incorporates the next frame after N iterations. We can deduce that frame t is rendered $(T - t + 1)N$ times, for $t = 1, \dots, T$, which leads to the total number of renderings

$$\begin{aligned}
\sum_{t=1}^T (T - t + 1)N &= N \left[\sum_{t=1}^T (T + 1) - \sum_{t=1}^T t \right] \\
&= N \left[T(T + 1) - \frac{T(T + 1)}{2} \right] \\
&= N \frac{T(T + 1)}{2} \\
&= O(T^2N).
\end{aligned}$$

On the other hand, our method renders exactly TN times as each frame is optimized independently. Therefore, our frame-wise strategy scales better for long sequences.

It is to be noted that PGSfT [49] leverages physical priors from a pre-trained surrogate model and uses a much smaller number N than ϕ -SfT [17] and ours, in order to speed up the optimization process. In future work, this idea can be combined with our strategy on a specific type of object for real-time reconstruction.

Table 9. Ablation study of various optimization strategies on Kinect Paper dataset. Average depth RMSE and runtime are reported.

	Default	250-iter	Window	Adaptive
RMSE (mm)	4.01	3.99	3.57	3.64
Runtime (mins)	6.04	7.52	50.54	84.37

8.2. Window-wise Optimization

Our strategy for optimizing each frame independently can be extended to optimizing each window of w frames concurrently, where frame-wise strategy is a special case by taking $w = 1$. Using more than one frames allows us to employ temporal smoothness regularizer, e.g. for $w = 3$, we can impose

$$\mathcal{L}_{\text{temporal}}(\mathbf{x}_t, \mathbf{x}_{t+1}, \mathbf{x}_{t+2}) = \ell\left(\mathbf{x}_{t+1}, \frac{\mathbf{x}_t + \mathbf{x}_{t+2}}{2}\right), \quad (10)$$

for ℓ any choice of data loss. This represents the constraint $\mathbf{x}_{t+1} = \frac{\mathbf{x}_t + \mathbf{x}_{t+2}}{2}$ derived from kinematics equations.

8.3. Adaptive Frame-wise Optimization

In our default setting, we used a warmup of 500 iterations and then 200 iterations for each frame to reconstruct the shapes within a video. This setting well balances between accuracy and runtime, as illustrated in our experiments in Section 4. However, this fixed-iteration strategy might fail to capture inter-frame complex and fast deformations as seen at frame 170 in Figure 3 on Kinect Paper dataset. Using more iterations can improve the performance, but the number of iterations vary among sequences and need to be tuned appropriately. Therefore, adaptive stopping criteria for per-frame optimization can be developed. A straightforward approach is to fix a tolerance on the difference of the total loss values between two consecutive iterations.

8.4. Ablation Study

We experiment on Kinect Paper dataset using the following optimization strategies:

- (1) **Default**: the default frame-wise strategy with 500-iteration warmup and 200 iterations per frame,
- (2) **250-iter**: the frame-wise strategy with 500-iteration warmup and 250 iterations per frame,
- (3) **Window**: the window-wise strategy in Section 8.2 500-iteration warmup and 600 iterations for each window of $w = 3$ frames. We also add the temporal loss in Equation (10) for $\ell(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2$.
- (4) **Adaptive**: the adaptive frame-wise strategy in Section 8.3 with tolerance $\tau = 10^{-6}$.

Table 9 shows the depth map root mean square error (RMSE) and runtime of the four strategies. We additionally plot the per-frame RMSE for the entire sequence in

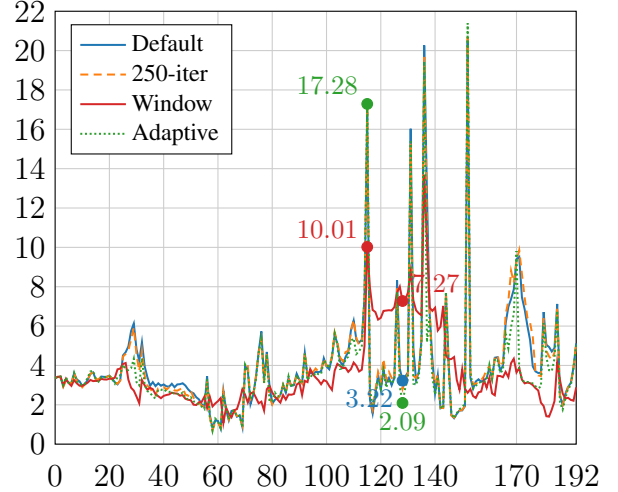


Figure 9. Illustration of depth map RMSE (mm) on the entire Kinect Paper sequence of 193 frames with various optimization strategies.

Figure 9. We can observe that later frames 110-170 are tricky to our method. Using more iterations, **250-iter** improves the accuracy a bit compared to **Default** and thus needs a minute more. However, the overall behavior looks quite similar to **Default**. **Window** can enhance the average accuracy in a temporal smoothness manner on the error curve. This strategy, however, needs more studies on tuning the per-window iterations or window size to achieve better performance. **Adaptive** strategy illustrates its effectiveness of reaching highly accurate reconstruction (e.g. RMSE reduced from 3.22 with **Default** and 7.27 with **Window**, to 2.09 at frame 128) while still struggles with some tricky frames (e.g. RMSE at frame 115 is 17.28 compared to 10.01 with **Window**). Further works can study on the tolerance to improve accuracy, while noticing the costly tradeoff in runtime.