Supplementary Document: High-Fidelity 4D Cloth Capture Pipeline with a Two-Level Pattern

ZIHENG LIU, University of Utah, USA
ANKA CHEN, NVIDIA, USA
SHU CHEN, University of Utah, USA
YIN YANG, University of Utah, USA
CEM YUKSEL, University of Utah, USA
JENNY HAN LIN, University of Utah, USA

ACM Reference Format:
Ziheng Liu, Anka Chen, Shu Chen, Yin Yang, Cem Yuksel, and Jenny Han Lin. 2026. Supplementary Document: High-Fidelity 4D Cloth Capture Pipeline with a Two-Level Pattern. *ACM Trans. Graph.* 45, 4 (July 2026), 4 pages. https://doi.org/10.1145/3811305

1 Neural Network Implementation

This section provides detailed architectural specifications for the three neural networks described in Sec. 3.2: the Marker Localizer, Marker Recognizer, and Crosspoint Localizer.

1.1 Marker Localizer

The Marker Localizer (illustrated in Fig. 4) simultaneously identifies coarse markers and localizes their localization points using a grid-based detection approach. The input image is divided into 8×8 pixel grid cells, where each cell independently predicts marker presence and the positions of five localization points (center and four corners).

Backbone. We employ a ResNet-18 [He et al. 2016] backbone pre-trained on ImageNet, modified with dilated convolutions to preserve spatial resolution. Specifically, layer 2 uses dilation rate $(2, 2)$ and layer 3 uses dilation rate $(4, 4)$, both with stride $(1, 1)$ instead of the standard $(2, 2)$ downsampling. This maintains output resolution at $1/8$ of the input rather than $1/32$.

Detection Head. The 512-channel ResNet features are processed through two convolutional layers: $\text{Conv}_{3\times3}(512\to128)$ followed by $\text{Conv}_{1\times1}(128\to11)$, producing 11 output channels per grid cell:

- Channel 0: Marker presence confidence (sigmoid-activated)
- Channels 1-10: Five localization point positions (x, y) - center, top-left, top-right, bottom-left, bottom-right

All positions are relative to the grid cell's top-left corner. The four corner localization points are used for the homography transformation in the subsequent stages (see Fig. 4c).

Loss Function. The loss combines binary cross-entropy for marker identification and MSE for localization point regression:

$$\mathcal{L} = \lambda \cdot \text{BCE}(c, c^*) + \frac{1}{5}\sum_{k=1}^{5} \text{MSE}(\mathbf{p}_k, \mathbf{p}_k^*) \cdot \mathbb{1}_{c^*>0.5} \tag{1}$$

where c and c^* are the predicted and ground truth marker confidences, $\mathbf{p}_k$ and $\mathbf{p}_k^*$ are the predicted and ground truth positions for the five localization points, $\lambda = 10$ scales the identification loss, and the indicator function ensures position loss is only computed on grid cells containing markers.

Inference. Markers are extracted by thresholding confidence ($c > 0.5$), converting relative positions to absolute coordinates, and applying non-maximum suppression to remove duplicates. For markers near grid boundaries (within 1 pixel), predictions are duplicated to neighboring cells with adjusted offsets to ensure robust detection across cell boundaries.

1.2 Marker Recognizer

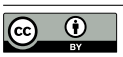
The Marker Recognizer (illustrated in Fig. 4d) performs multi-class classification to identify which specific marker pattern is present in a given image crop. After markers are detected and localized by the Marker Localizer, their corresponding image regions are extracted via homography transformation (see Fig. 4c) and classified to determine marker identity.

Input Preprocessing. For each detected marker, the four corner localization points are used to compute a homography transformation that unwarps the marker region into a standardized square of size 96×96 pixels. This transformation removes perspective distortion and most non-rigid deformations, providing a canonical view with reduced variance.

Backbone. We employ a ResNet-50 [He et al. 2016] backbone pre-trained on ImageNet. Unlike the Marker Localizer, the standard ResNet-50 architecture is used without dilated convolutions, as the input is an unwarped marker crop rather than a full image. Input images are normalized using ImageNet statistics with mean $\mu = [0.485, 0.456, 0.406]$ and standard deviation $\sigma = [0.229, 0.224, 0.225]$.

Classification Head. The 2048-channel ResNet-50 features are converted to class predictions through: $\text{AdaptiveAvgPool}_{2\text{D}}(1\times1) \to$ Flatten $\to$ $\text{Linear}(2048\to N) \to$ Softmax, where N is the number of unique marker IDs in the dataset.

Authors' Contact Information: Ziheng Liu, University of Utah, Salt Lake City, UT, USA, lzh2199@gmail.com; Anka Chen, NVIDIA, Kirkland, USA, ankachan92@gmail.com; Shu Chen, University of Utah, USA, samuel233qaq@gmail.com; Yin Yang, University of Utah, USA, yangzzzy@gmail.com; Cem Yuksel, University of Utah, USA, cem@cemyuksel.com; Jenny Han Lin, University of Utah, USA, jenny.h.lin@utah.edu.



ACM 1557-7368/2026/7-ART
https://doi.org/10.1145/3811305

Loss Function. The network is trained with standard cross-entropy loss:

$$\mathcal{L} = -\sum_{i=1}^{N} y_i^* \log(y_i) \tag{2}$$

where y_i is the predicted probability for marker ID i and y_i^* is the ground truth one-hot label.

1.3 Crosspoint Localizer

The Crosspoint Localizer (illustrated in Fig. 4e) performs dense prediction to localize the 25×25 crosspoints within each unwarped marker image. For each crosspoint, the network predicts a visibility flag and, if visible, its (x, y) coordinates within the 112×112 unwarped marker image. By applying the inverse homography transformation, these coordinates are projected back to the original image space, enabling precise correspondence between image observations and the garment's parametric representation.

Input Representation. Following marker recognition, the network takes two inputs: (1) the 112×112 unwarped marker image, and (2) the reference texture corresponding to the identified marker ID. Both inputs are normalized using ImageNet statistics. This dual-input formulation generalizes better across different marker appearances, as the network does not need to memorize each marker internally.

Backbone. We employ a ResNet-50 backbone pre-trained on ImageNet with dilated convolutions in layers 2 and 3 (dilation rates 2 and 4 respectively) to preserve spatial resolution. The architecture follows an encoder-decoder structure. The reference texture and unwarped image are processed through separate initial convolution layers, then concatenated with intermediate image features and combined via a 1×1 convolution before further encoding through the ResNet layers.

Decoder Head. After encoding, the 2048-channel features are processed through: $\text{ConvTranspose}_{2\times2}(2048\to512) \to \text{Conv}_{4\times4}(512\to64) \to \text{Conv}_{1\times1}(64\to3)$, producing 3 output channels per pixel:

- Channel 0: Crosspoint visibility confidence (sigmoid-activated)
- Channels 1-2: Crosspoint coordinates (x, y) within the unwarped marker image

Loss Function. The loss combines binary cross-entropy for visibility prediction and MSE for coordinate regression:

$$\mathcal{L} = \lambda \cdot \text{BCE}(c, c^*) + \text{MSE}(\mathbf{p}, \mathbf{p}^*) \cdot \mathbb{1}_{c^*>0.5} \tag{3}$$

where c and c^* are predicted and ground truth visibility confidences, $\mathbf{p}$ and $\mathbf{p}^*$ are predicted and ground truth crosspoint coordinates, $\lambda = 100$ scales the visibility loss, and MSE is computed only on visible crosspoints.

Partial Marker Handling. The network also processes partially visible markers near garment boundaries. For these markers, a reasonable homography transformation is estimated using adjacent fully visible markers, allowing crosspoint localization even when not all four localization points are observable.

2 Physics-Based Post-Processing

The triangulated reconstruction contains missing regions due to self-occlusions and may exhibit intersection artifacts from measurement noise. We employ a physics-based post-processing pipeline built on Vertex Block Descent (VBD) [Chen et al. 2024] and Offset Geometric Contact (OGC) [Chen et al. 2025] within the Newton Physics Engine [Newton Contributors 2025] to complete missing regions and remove intersections while preserving the fidelity of captured data.

2.1 Multi-Resolution Hierarchy

We construct a hierarchy of 4 resolution levels to enable efficient VBD simulation on high-resolution garment meshes:

- Level 0 (coarsest): 12× downsampled
- Level 1: 4× downsampled
- Level 2: 2× downsampled
- Level 3 (finest): full-resolution

Downsampling Method. To construct coarser levels from the full-resolution mesh, we employ a hybrid approach that mirrors the full-resolution construction. For the inner mesh, we simply use larger regular grid spacing (e.g., 12mm, 4mm, 2mm instead of 1mm). For the outer boundary mesh, we generate new triangulations with larger triangles and merge them with the coarser inner grid, following the same procedure used for the full-resolution mesh.

Upsampling and Interpolation. After solving at a coarse level, we transfer the solution to the next finer level using barycentric interpolation. For each fine-level vertex, we locate the containing triangle in the coarse mesh and compute its position as a weighted combination of the coarse triangle's vertices using barycentric coordinates.

2.2 Dynamic Rest Shape Computation

The rest shape computation uses the same formulas for both captured and inpainted regions. The key difference lies in the source of the rest shape geometry:

Rest Shape Computation. For the bending and membrane energies, we compute:

- **Rest angles:** The dihedral rest angle $\bar{\theta}$ for each edge is the dihedral angle between two neighboring triangles in the rest geometry.
- **Deformation gradient:** For the membrane (StVK) energy, the deformation gradient is computed as $\mathbf{F} = \mathbf{D}\bar{\mathbf{D}}^{-1}$, where $\mathbf{D} = [\mathbf{x}_1 - \mathbf{x}_0, \mathbf{x}_2 - \mathbf{x}_0]$ is the 3×2 matrix formed by the triangle edges. This is projected to 2D to obtain a 2×2 matrix before computing the deformation gradient with the inverse of $\bar{\mathbf{D}}$ from the rest geometry.

Rest Shape Source. The rest geometry differs between captured and missing regions:

- **Captured regions:** When all vertices of the stencil (hinge for bending, triangle for membrane) are captured, the rest angles and rest matrices are computed from the observed triangulated data of the current frame. This ensures that bending and membrane energies evaluate to near-zero for the captured

geometry, preventing the material model from penalizing observed deformations.

- **Missing regions (first pass):** The rest angles and rest matrices are set to the values from the previous frame: $\bar{\theta} = \theta^{t-1}$ and $\bar{\mathbf{D}} = \mathbf{D}^{t-1}$. This encourages minimal deformation, providing temporal coherence.
- **Missing regions (bidirectional smoothing pass):** After initial forward/backward propagation, we perform a second smoothing pass where the rest angles and rest matrices are averaged from the previous and following frames: $\bar{\theta} = \frac{1}{2}(\theta^{t-1} + \theta^{t+1})$ and $\bar{\mathbf{D}} = \frac{1}{2}(\mathbf{D}^{t-1} + \mathbf{D}^{t+1})$.

2.3 Optimization Parameters

The total energy is a weighted combination of multiple terms:

$$E_{\text{total}} = w_{\text{spring}} E_{\text{spring}} + w_{\text{membrane}} E_{\text{StVK}} + w_{\text{bend}} E_{\text{bending}} + w_{\text{contact}} E_{\text{barrier}}$$

where:

- $w_{\text{spring}} = 1.0 \times 10^6$ (spring energy for captured vertices)
- w_{membrane}: Lamé parameters ($\mu = 1.0 \times 10^2, \lambda = 1.0 \times 10^1$) for the StVK membrane stretching/shearing energy
- $w_{\text{bend}} = 1.0 \times 10^{-1}$ (dihedral bending energy)
- $w_{\text{contact}} = 1.0 \times 10^4$ (barrier contact energy with contact radius 1mm)

The spring weight is set significantly higher than material energies to anchor captured regions to observations.

2.4 Initial Frame Selection

To initialize the sequence in a penetration-free state, we empirically select an appropriate initial frame t_i based on the garment type:

- **Suit, shirt, and pants:** We select a frame close to the T-pose, where the body configuration most closely resembles the default flat garment shape.
- **Skirt:** We choose a frame during the spinning motion when the skirt expands and most vertices are directly recovered from the capture.

Starting from the default penetration-free garment mesh, we progressively deform it to match the captured data at frame t_i. We then propagate in both directions to recover the full sequence.

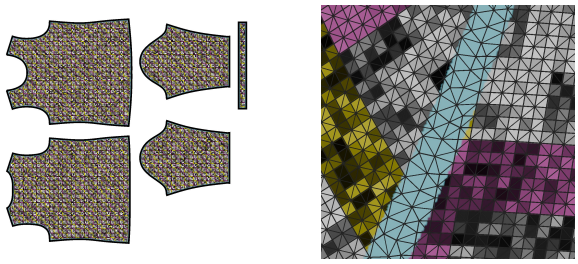
3 Data Generation

All data used to train our models are synthetically generated, as the feature points are so densely distributed in the images that manual annotation is infeasible. This section details our comprehensive data generation pipeline, from garment design to final dataset composition.

3.1 Garment Design and Pattern Creation

The foundation of our synthetic data generation begins with obtaining base 2D garment patterns from online resources. For each garment type (shirt, skirt, suit, and pants), we source standard sewing patterns that define the geometric layout of pattern pieces.

We then overlay our custom marker pattern design onto these base patterns. The resulting composite pattern is sent to a custom fabric printing service, which produces physical printed fabric that is used in our real-world capture setup. Simultaneously, this same pattern image serves as the texture map for our digital template garment as shown in Fig. 1a.



(a) *Shirt Pattern* **(b)** *Triangulation of Shirt.*

Fig. 1. *(a) The shirt pattern is used as both texture image for the template mesh and the actual printing pattern. (b) The inner triangulation matches the noise pattern and the outer triangulation handles sewing.*

The triangle mesh construction employs a hybrid approach where the interior mesh matches the pattern geometry while the boundary mesh ensures correct sewing as shown in Fig. 1b. For the interior regions of each pattern piece, we use a regular triangular grid with 1mm resolution to maintain correspondence with the printed pattern. However, the boundary regions require special handling to ensure that pattern pieces can be correctly sewn together during cloth simulation. We leverage the commercial garment design software Style3D to simulate the garment assembly and generate boundary triangulations that respect sewing constraints and edge alignment between adjacent pieces. The resulting high-resolution mesh is used as the template for 3D reconstruction in our pipeline. For synthetic training data generation, we create a separate low-resolution version of this mesh, as using the full-resolution mesh would be computationally expensive and unnecessary for training purposes.

3.2 Body Motion Generation

To generate diverse and realistic training data, we leverage the AMASS dataset [Mahmood et al. 2019] in combination with the SMPL body model [Loper et al. 2015]. Specifically, we select 5 subjects (05, 12, 49, 85, and 124) from the CMU dataset [Carnegie Mellon University [n. d.]] in AMASS format and use all their motion sequences. These sequences cover a wide range of human activities including walking, running, jumping, dancing, sports, and acrobatic movements, ensuring our trained models generalize across diverse and drastic body poses and movements.

3.3 Cloth Simulation and Rendering

For each body pose frame, we simulate realistic cloth deformation using a high-performance physics-based cloth simulator [Lu et al. 2025]. The simulation uses a timestep of 1/120 seconds, matching the temporal resolution of the CMU motion capture dataset. The simulator accounts for material properties, gravitational forces, and collision constraints between the garment and the SMPL body mesh, ensuring the garment reaches a physically stable configuration on

Fig. 2. *A suit simulated and rendered in a baseball pitching pose.*

the body. We use As-Rigid-As-Possible (ARAP) model [Sorkine and Alexa 2007] for membrane energy and quadratic bending [Bergou et al. 2006] for bending energy. Note that the energy forms (membrane, bending, and contact) differ from those used in our physics-based post-processing, as the two simulators serve different purposes and are built on different frameworks. The simulation parameters are:

- $w_{\text{density}} = 1.0 \times 10^2$
- $w_{\text{membrane}} = 1.0 \times 10^5$ (ARAP membrane stretching/shearing energy)
- $w_{\text{bend}} = 1.0 \times 10^{-2}$ (quadratic bending energy)
- $w_{\text{contact}} = 1.0 \times 10^8$ (penalty contact energy with contact radius 1mm)

Each of the three garments (suit, skirt, and shirt) is simulated on the same set of 4874 motion frames.

The simulated garment meshes are then rendered using Blender to generate photorealistic training images. We use a fixed camera configuration but apply random rotation and translation to the garment to increase viewpoint diversity. For lighting, we randomly select from 10 environment maps for each render. Each simulation frame generates 5 variations with different random transformations and lighting conditions, resulting in 24,370 images per garment (4,874 frames × 5 variations). All datasets are split into training, validation, and test sets with a ratio of 22:2:1. An example frame is shown in Fig. 2.

3.4 Ground Truth Generation

Since the entire generation process is simulated, we have precise access to ground-truth information for all supervision signals required by our neural networks. For each rendered frame, we automatically extract:

Marker Localization Ground Truth: The 3D positions of all marker centers and corner localization points are known from the mesh geometry. These are projected to 2D image coordinates using the render camera parameters, providing exact ground-truth positions for the Marker Localizer training.

Marker Recognition Ground Truth: Each marker is assigned a unique identifier based on its position in the garment pattern topology. The four corner localization points of each marker are used to compute a homography transformation that warps the marker region into a standardized 96×96 pixel image crop for training. This assignment remains consistent across all frames, enabling the Marker Recognizer to learn distinctive visual patterns for each marker ID.

Crosspoint Localization Ground Truth: The 25×25 crosspoint grid on each marker is tracked through the simulation. The four corner localization points are used to compute a homography transformation that warps the marker region into a standardized 112×112 pixel image crop. For each crosspoint, we determine its visibility by rendering depth tests and checking occlusions. Visible crosspoints are projected to coordinates within the warped image crop, while occluded points are marked as invalid. This provides complete supervision for the Crosspoint Localizer including both position and validity labels.

References

Miklos Bergou, Max Wardetzky, David Harmon, Denis Zorin, and Eitan Grinspun. 2006. A quadratic bending model for inextensible surfaces. In *Symposium on Geometry Processing*, Vol. 227. 230.

Carnegie Mellon University. [n. d.]. CMU MoCap Dataset.

Anka He Chen, Jerry Hsu, Ziheng Liu, Miles Macklin, Yin Yang, and Cem Yuksel. 2025. Offset Geometric Contact. *ACM Trans. Graph.* 44, 4, Article 160 (July 2025), 21 pages. doi:10.1145/3731205

Anka He Chen, Ziheng Liu, Yin Yang, and Cem Yuksel. 2024. Vertex Block Descent. *ACM Trans. Graph.* 43, 4 (July 2024), 116:1–116:16. doi:10.1145/3658179

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. 2015. SMPL: A Skinned Multi-Person Linear Model. *ACM Trans. Graph.* 34, 6 (Oct. 2015), 248:1–248:16. doi:10.1145/2816795.2818013

Zixuan Lu, Ziheng Liu, Lei Lan, Huamin Wang, Yuko Ishiwaka, Chenfanfu Jiang, Kui Wu, and Yin Yang. 2025. High-performance CPU Cloth Simulation Using Domain-decomposed Projective Dynamics. *ACM Trans. Graph.* 44, 4, Article 51 (July 2025), 17 pages. doi:10.1145/3731182

Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. 2019. AMASS: Archive of Motion Capture as Surface Shapes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 5442–5451.

Newton Contributors. 2025. *Newton: GPU-accelerated physics simulation for robotics, and simulation research.* Newton a Series of LF Projects, LLC. https://github.com/newton-physics/newton

Olga Sorkine and Marc Alexa. 2007. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, Vol. 4. 109–116.