ReWeaver: Towards Simulation-Ready and Topology-Accurate Garment Reconstruction

Supplementary Material

A. Topology Refinement

To obtain clean sewing-structure topology from the raw predictions, we apply a refinement pipeline that combines reliability-based filtering, 3D curve consolidation, and 2D loop analysis. This removes redundant or inconsistent curves and recovers stable, closed panel boundaries.

We begin by filtering patch predictions using a relatively high threshold $\epsilon_p = 0.7$. Since patch validity is already highly reliable ($\text{Acc}_p = 0.9210$), this single cutoff cleanly separates valid from invalid patch queries.

Curve validity and patch–curve adjacencies, in contrast, are filtered with more permissive thresholds, $\epsilon_c = 0.5$ and $\epsilon_{\text{adj}} = 0.5$. Empirically, these thresholds avoid false negatives but may retain redundant or overlapping curve predictions. The following refinement steps address this over-retention.

Duplicate Curve Merge We first merge curves that represent nearly the identical geometry. Two curves C_a and C_b are considered duplicates if their bidirectional Chamfer distance is small, i.e.,

$$\text{CD}(C_a \to C_b) < 0.03 \quad \text{and} \quad \text{CD}(C_b \to C_a) < 0.03.$$

For each such pair, we retain the curve with the higher predicted validity probability and transfer the discarded curve's adjacency relations to it.

Sub-curve Removal If two curves share the same adjacency pattern, and one is geometrically contained inside the other, we treat the shorter one as a spurious "sub-curve." Formally, if

$$\text{CD}(C_{\text{sub}} \to C_{\text{main}}) < 0.04,$$

we discard C_{sub}. This removes small fragments arising from local over-segmentation.

2D loop pruning For each panel, let $\{E_{ij}\}_{j=1}^{N}$ denote the incident 2D edges predicted from the associated 3D curves. Our goal is to keep only the set of edges that best forms a closed loop.

We first consider all possible orderings and orientations of the edges to form a closed boundary:

$$[E'_{ij_k}] = (E'_{ij_1}, E'_{ij_2}, \ldots, E'_{ij_N}).$$

For any ordering, we define the loop cost as

$$C([E'_{ij_k}]) = \sum_{k=1}^{N} \left\| E'_{ij_k}[-1] \; - \; E'_{ij_{k+1}}[0] \right\|_2^2, \qquad j_{N+1} = j_1. \tag{8}$$

The optimal cost for the unordered edge set is then

$$C(\{E_{ij}\}_{j=1}^{N}) = \min_{[E'_{ij_k}]} C([E'_{ij_k}]). \tag{9}$$

For any edge E_{ik} in the panel, we compute the loop cost after removing it:

$$C\Big(\{E_{ij}\}_{j=1}^{N} \setminus \{E_{ik}\}\Big).$$

If removal reduces the optimal cost by a sufficient margin,

$$C\Big(\{E_{ij}\}_{j=1}^{N} \setminus \{E_{ik}\}\Big) \; < \; C\Big(\{E_{ij}\}_{j=1}^{N}\Big) - \epsilon, \tag{10}$$

where $\epsilon = 10^{-9}$, then E_{ik} is deemed inconsistent with the panel loop and is pruned. When multiple edges satisfy the condition, we remove the one that yields the greatest cost reduction. The pruning is repeated until convergence.

Ablation Analysis

To isolate the effect of each refinement step, we conduct a cumulative ablation study in which the following rules are progressively enabled:

1. threshold-based filtering only;
2. thresholding + 2D loop pruning;
3. thresholding + loop pruning + sub-curve removal;
4. thresholding + loop pruning + sub-curve removal + duplicate merging (full refinement).

The results in Table 3 show that each refinement stage contributes positively to edge accuracy and panel IoU. Because topology refinement has negligible influence on the Chamfer Distance of patches and curves, we omit those metrics here.

B. 2D Geometry Refinement

Since our model does not explicitly enforce perfect edge-to-edge connectivity, the predicted panel boundaries may contain small gaps and misalignments. To obtain visually smoother and more aesthetically pleasing panels, we apply a geometry refinement stage. We first reorder and flip the edges as described in Section A, and then perform the following two geometry refinement steps:

Table 3. **More ablation on topology refinement.** Progressive rules further improve edge accuracy and panel IoU and slightly reduce edge Chamfer Distance CD_e.

Method	$Acc_e \uparrow$	$CD_e \downarrow$	IoU↑
Threshold only	0.5361	0.0416	0.7775
+ 2D loop pruning	0.6045	0.0444	0.7584
+ sub-curve removal	0.6269	0.0418	0.7940
+ duplicate merging	**0.7175**	**0.0391**	**0.8221**

Replace bad edges. Consider an ordered closed loop of N edges $\{E_{ij}\}$ for panel i, where each edge $E_{ij} \in \mathbb{R}^{M_E \times 2}$ is represented by M_E sampled 2D points. For each edge E_{ij}, we denote s_j and e_j as the start and end point, and similarly e_{j-1} and s_{j+1} from its neighbors (indices modulo N). We define the local connection gaps as:

$$\text{gap}_1(j) = \left\|s_j - e_{j-1}\right\|_2, \quad \text{gap}_2(j) = \left\|e_j - s_{j+1}\right\|_2,$$

and normalize them by the edge length $\ell_j = \|e_j - s_j\|_2$. Edges with a large normalized connection error,

$$\frac{\text{gap}_1(j)}{\ell_j} + \frac{\text{gap}_2(j)}{\ell_j} > \tau_{\text{gap}}, \tag{11}$$

are marked as "bad", where $\tau_{\text{gap}} = 3.0$. For each bad edge E_{ij}, we discard its original geometry and replace it with a straight segment that linearly interpolates between e_{i-1} and s_{i+1}, resampled into M_E points. This operation preserves the loop topology while smoothing out locally inconsistent or noisy edge predictions.

Align edges to joint midpoints. After replacing obviously bad edges, we further refine the geometry by snapping all edges to a consistent set of joint targets along the loop. For a panel i with an ordered closed loop of edges $\{E_{ij}\}_{j=1}^{N}$, each sampled as $E_{ij} \in \mathbb{R}^{M_E \times 2}$, we let e_j and s_j denote the end point and start point of edge E_{ij}, respectively. For the joint between E_{ij} and $E_{i,\,j+1}$ (indices taken modulo N), we define a target vertex

$$v_j = \frac{e_j + s_{j+1}}{2}, \tag{12}$$

i.e., the midpoint of the two incident edge endpoints. This yields a set of N joint targets $\{v_j\}_{j=1}^{N}$ distributed along the loop.

For each edge E_{ij}, we estimate a 2D similarity transform

$$T_{ij}(p) = s_{ij} R_{ij} p + t_{ij},$$

that aligns its endpoints to the corresponding joint targets:

$$T_{ij}(s_j) = v_{j-1}, \qquad T_{ij}(e_j) = v_j.$$

Here $s_{ij} \in \mathbb{R}$ denotes the scale factor, $R_{ij} \in \mathbb{R}^{2\times 2}$ is a 2D rotation matrix, and $t_{ij} \in \mathbb{R}^2$ is the translation vector. Such a solution clearly exists and is unique in the two-dimensional space. The scale parameter is clamped to a reasonable range to avoid degenerate transforms. After solving for T_{ij}, we apply it to all M_E sampled points of E_{ij} to obtain the refined edge $\tilde{E}_{ij}$. This alignment step enforces consistent edge-to-edge joints, removes small gaps or overlaps, and preserves the overall shape of each edge.

C. Experiment Details

More training details. We train our model for 50 epochs on 8×H200 GPUs, which takes about 120 hours in total. We use the AdamW optimizer with a unified weight decay of 1×10^{-4}, and set the base learning rates of the visual encoder, 3D prediction head, and 2D prediction head to 5×10^{-5}, 1×10^{-5}, and 5×10^{-5}, respectively. All three modules share the same iteration-wise schedule: a short linear warmup of 1 epochs that ramps the learning rate from a very small value to its base value, followed by cosine annealing down to a small floor of 5×10^{-6}. We use a total batch size of 128 (16 per GPU across 8 GPUs).

Visual Encoder. We adopt a lightweight DINO-based image encoder with 22M parameters, followed by a stack of 12 intra-frame and inter-frame attention layers. We use the DINOv2[37] backbone with a patch size of 14, and its weights are kept fully trainable during training. Inside the visual encoder, the token dimension is fixed to 384. The outputs of the last intra-frame attention layer and the last inter-frame attention layer are concatenated to form a 768-dimensional feature, which is used as the input to the subsequent modules. We resize all input images to a resolution of 518×518 and apply per-pixel normalization by subtracting the dataset mean and dividing by the dataset standard deviation.

3D Curve and Patch Prediction. We set the numbers of curve and patch queries to 200 and 70, respectively—about twice the maximum values in the dataset—to provide a safe margin for more complex or augmented cases while keeping the number of empty queries and the computational cost reasonably low. We assign a weight of 1 to all BCE losses and a weight of 300 to all geometric losses. We train curve/patch validity with a weighted binary cross-entropy loss: the positive class (valid curve/patch) has weight 1, and the negative class (empty query/patch) weight is chosen from the empirical ratio between valid curves and empty queries, scaled by a global factor so that both contribute roughly equally to the loss. We use a bipath transformer decoder with 12 layers, where each layer applies 1) per-path self-attention; 2) cross-attention between the two paths con-

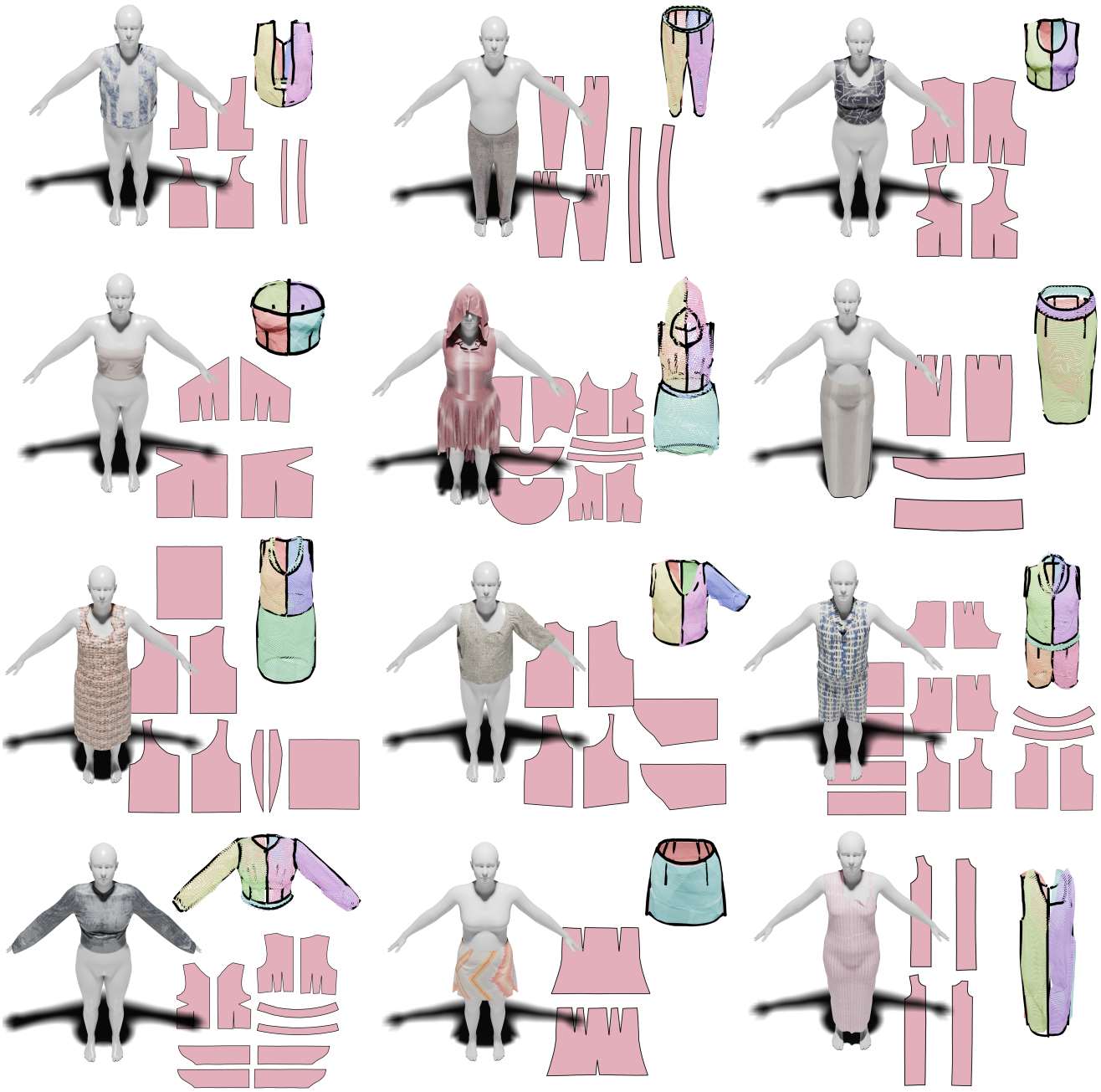

Figure 8. Visualization of more results

ditioned on primitive-type embeddings; 3) cross-attention to the image features; and 4) a final feed-forward network.

2D Pattern Prediction. We assign a weight of 300 to the edge geometry loss and 1×10^{-2} to the scale loss. Each edge is represented by 50 points that are uniformly sampled along its arc length. For every panel, we normalize its 2D coordinates to the range $[-1, 1]$ by subtracting the panel-wise mean and dividing by the maximum absolute coordinate value (scale). This normalization improves the numerical stability of the predictions and largely removes the ambiguity caused by global translations. The 2D pattern prediction module takes curve and patch features together with the patch–curve adjacency matrix, and first groups the corresponding tokens into panel-wise sets according to their adjacency relations. The panel-wise edge tokens are then pro-

cessed by 12 stacked transformer-style layers: each layer applies self-attention among edges of the same panel, followed by cross-attention where edge tokens attend to the corresponding panel token (edges as queries, panel token as keys/values), and finally a feed-forward network update, yielding the edge geometry and a scalar scale prediction for each panel.

D. More Results

The results of our experiments on various styles, such as strapless dresses, pencil skirts, and asymmetrical tops, are shown in Figure 8.

E. Garment Simulation

To dress the target human model, 2D garment panels are first positioned in 3D space based on their corresponding point cloud data. These panels are then imported into Marvelous Designer [34] along with the body model, where their placement is manually refined to ensure alignment with the respective body regions. Furthermore, the sewing relationships are established using the predictions from our model, which identifies and stitches together the edges corresponding to the same 3D curves. Subsequently, we employ the software's built-in physics engine with default material parameters to simulate the draping process. This wraps the panels around the body under collision constraints, achieving a final dressed state that conforms to the target pose. The associated project files will be released in our code repository.

F. Discussion

Notably, compared to prior methods that predict structured pattern descriptions including seam connectivity [14, 36], our approach is inherently more flexible and potentially more generalizable to topology. Since we operate directly on panels and their 2D edge geometry rather than committing to a fixed Structured pattern representation, our method can in principle accommodate garments with additional components (e.g., sewn-on pockets) or complex topologies where seams are not strictly one-to-one at the edge level, without any architectural changes. A thorough empirical study of this generalization capability, especially on such challenging cases, is left for future work.

Unlike most previous works that focus on either 3D garment geometry or 2D panel layouts in isolation, our method reconstructs both representations jointly, which naturally enables further joint optimization to refine them simultaneously. Beyond this, one could extend our framework with additional prediction heads or differentiable cloth simulation to also estimate physical material parameters and human shape/pose from multi-view images, using the coupled 2D–3D predictions as constraints. This would allow reconstruction of more complete and better aligned 3D assets from real-world image, and we believe this constitutes a particularly interesting direction for future work.

G. Limitations

A key limitation lies in the scarcity of high-quality 3D garments with complex topology and photorealistic textures. This limits the model's generalization ability on out-of-distribution and real-world data. Future work could introduce greater variations in camera parameters, human shapes and poses, and backgrounds to generate more diverse synthetic data, and incorporate real-world data during training to narrow the sim-to-real gap.