

# Appendices

723

724

## Contents

725

<b>1. Introduction</b>	<b>1</b>	<b>726</b>
<b>2. Related Works</b>	<b>2</b>	<b>727</b>
<b>3. Preliminaries</b>	<b>3</b>	<b>728</b>
3.1. Sewing Pattern . . . . .	3	729
3.2. Geometry Image . . . . .	3	730
<b>4. Methodology</b>	<b>3</b>	<b>731</b>
4.1. PatternMaker . . . . .	4	732
4.2. Garment Geometry Image . . . . .	4	733
4.3. GarmentSewer . . . . .	5	734
<b>5. Experiments</b>	<b>6</b>	<b>735</b>
5.1. Experimental Settings . . . . .	6	736
5.2. Sewing Pattern Generation . . . . .	6	737
5.3. Garment Mesh Generation . . . . .	6	738
5.4. Ablation Studies . . . . .	7	739
<b>6. Conclusion</b>	<b>8</b>	<b>740</b>
<b>A Appendices Overview</b>	<b>12</b>	<b>741</b>
<b>B Garment Geometry Image Preparation &amp; Postprocessing Pipeline</b>	<b>12</b>	<b>742</b>
B.1. Garment Geometry Image Preparation . . . . .	12	743
B.2. Postprocessing pipeline . . . . .	14	744
<b>C Experiments</b>	<b>18</b>	<b>745</b>
C.1. Experiment setup . . . . .	18	746
C.2. Sewing Pattern Generation . . . . .	18	747
C.3. Garment Mesh Generation . . . . .	18	748
C.4. Qualitative Results . . . . .	19	749
<b>D Discussions</b>	<b>20</b>	<b>750</b>

751

## A. Appendices Overview

This appendix provides supplementary materials that support and extend the main paper. It is organized into three sections. In Sec. B, we describe the full procedure for preparing the Garment Geometry Image (GGI) along with the post-processing pipeline that converts a predicted or processed GGI back into a simulation-ready 3D garment mesh. In Sec. C, we provide detailed experimental settings, including dataset preparation, evaluation protocols, and additional qualitative and quantitative results to ensure fair comparison across baselines. In Sec. D, we discuss limitations, broader impact, and potential future directions for garment generation and 3D modeling. Together, these sections offer the technical details needed to reproduce our work and understand the complete scope of the SwiftTailor framework.

## B. Garment Geometry Image Preparation & Postprocessing Pipeline

### B.1. Garment Geometry Image Preparation

Following the overview in Sec. 4.3 of the main paper, we provide the complete procedure for constructing the Garment Geometry Image (GGI). The GGI is formed by three aligned components—geometry, semantic, and stitching images. Before generating these components, we first repack all garment panels into a unified square layout. This packed layout serves as the shared UV template onto which all information is embedded, ensuring alignment across components and enabling downstream tasks such as texture editing.

---

#### Algorithm 1 Layout Packing with Orientation Correction

---

**Input:** Panels  $\mathbf{P}$  in the sewing pattern

**Output:** Packed UV layout  $L_{UV}$

```

1:  $B \leftarrow$  bounding-box sizes of all panels in  $\mathbf{P}$ 
2:  $B_{\text{sorted}} \leftarrow$  sort  $B$  by decreasing height, then width
3: Initialize binary-search range  $[l, r]$  for the target square size
4: while  $l < r$  do
5:    $mid \leftarrow \lfloor (l + r) / 2 \rfloor$ 
6:   Test row-wise placement of  $B_{\text{sorted}}$  inside a square of size  $mid$ 
7:   if all panels fit then
8:      $r \leftarrow mid$ 
9:   else
10:     $l \leftarrow mid + 1$ 
11:   end if
12: end while
13:  $L_{UV} \leftarrow$  final packing of  $B_{\text{sorted}}$  within a square of size  $l$ 
14: for each panel  $P_i$  with bounding box  $b_i$  do
15:   Compute the outward-facing normal of  $P_i$ 
16:   if the panel normal is opposite to the layout normal then
17:     Flip  $P_i$  horizontally within  $b_i$  and update  $L_{UV}$ 
18:   end if
19: end for
20: return  $L_{UV}$ 

```

---

**Layout Packing** To obtain a compact and consistent representation, all panels predicted by the model are arranged within a single square layout. Given a set of panels  $\mathbf{P}$ , we compute the bounding box of each panel and reduce the layout task to packing rectangles into a square of unknown size. A useful monotonic property holds: if all panels fit into a square of side length  $s$ , then they also fit into any larger square  $s' > s$ . This observation allows us to apply binary search to identify the smallest feasible square size, preventing unnecessary blank space that may hinder learning efficiency and waste memory storage. For each candidate size, we test feasibility using a simple row-wise packing heuristic: panels are sorted by height and width, and placed from bottom to top and from left to right. Although simple, this strategy is effective and fast across the GarmentCodeData [17]. The full algorithm is given in Algorithm 1.

A second consideration arises from panel orientation. Flattening all panels into UV space introduces inconsistencies in their outward-facing normals, especially between front- and back-facing panels. To enforce a consistent orientation across the layout, we choose a fixed normal direction for the whole layout and horizontally flip panels if mismatching. This ensures uniform facing direction in the packed UV map and simplifies later steps in remeshing.

775  
776  
777  
778

---

**Algorithm 2** Semantic Image Creation from Packed Layout
 

---

**Input:** Panels  $\mathbf{P}$  with type labels, packed layout  $L_{UV}$ , predefined color map  $\mathcal{C}$  from panel types to unique colors

**Output:** Semantic image  $GGI_{\text{semantic}}$

```

1:  $GGI_{\text{semantic}} \leftarrow$  blank image with the same resolution as  $L_{UV}$ 
2: for each panel  $P_i \in \mathbf{P}$  do
3:    $R_i \leftarrow$  UV region of  $P_i$  in  $L_{UV}$ 
4:    $t_i \leftarrow$  type label of  $P_i$ 
5:    $c_i \leftarrow \mathcal{C}(t_i)$ 
6:   Fill region  $R_i$  in  $GGI_{\text{semantic}}$  with  $c_i$ 
7: end for
8: return  $GGI_{\text{semantic}}$ 

```

---

**Semantic Image** After determining the packed UV layout, we generate the semantic image by assigning each panel a unique color based on its panel type, see Algorithm 2. Using the metadata from the sewing pattern, every panel region in the packed layout is filled with its corresponding color, producing a dense map that encodes panel identity and functional category. This semantic image provides strong structural cues for GarmentSewer, enabling the model to distinguish between panels of similar shapes, such as left and right sleeves or front and back torso pieces, and to preserve correct topology in garments with other components.

779  
780  
781  
782  
783  
784

---

**Algorithm 3** Stitching Image Creation from Packed Layout
 

---

**Input:** Panels  $\mathbf{P}$  with stitched edge pairs  $\mathbf{S}$ , packed layout  $L_{UV}$ , predefined stitch color map  $\mathcal{C}_{\text{stitch}}$

**Output:** Stitching image  $GGI_{\text{stitch}}$

```

1:  $GGI_{\text{stitch}} \leftarrow$  blank image with the same resolution as  $L_{UV}$ 
2: for each stitched pair  $(e_a, e_b) \in \mathbf{S}$  with panels  $(P_i, P_j)$  do
3:    $B_a \leftarrow$  boundary pixels of edge  $e_a$  in  $L_{UV}$ 
4:    $B_b \leftarrow$  boundary pixels of edge  $e_b$  in  $L_{UV}$ 
5:    $k \leftarrow$  stitch identifier of pair  $(e_a, e_b)$ 
6:    $c_k \leftarrow \mathcal{C}_{\text{stitch}}(k)$ 
7:   Color  $B_a$  and  $B_b$  in  $GGI_{\text{stitch}}$  with  $c_k$ 
8: end for
9: return  $GGI_{\text{stitch}}$ 

```

---

**Stitching Image** Simultaneously, we construct the stitching image by encoding all boundary edges that participate in stitching relations. The boundary of each panel is extracted by applying a dilation operation on the packed layout to obtain a one-pixel-wide contour along its edges. For every stitched edge pair, we identify the corresponding boundary pixels and assign a shared stitch-identifier color to both edges, see Algorithm 3. This produces a map in which all edges that must be joined in the final garment share the same color, enabling consistent boundary alignment and merge operations during postprocessing.

785  
786  
787  
788  
789  
790

**Geometry Image** To construct the geometry image, we rasterize the 3D garment surface into the packed UV layout. For each panel, mesh vertices are first mapped to their corresponding UV coordinates, and the 3D positions are written into the geometry image at those pixel locations. Since meshes are typically far sparser than the resolution of the UV grid, this direct rasterization produces incomplete regions. A key challenge in constructing the geometry image is obtaining smooth and reliable values both inside panel and along boundaries. As illustrated in Fig. B.1, relying solely on barycentric interpolation produces visibly jagged boundary artifacts, since the interpolation is restricted to the discrete triangulation and does not align

791  
792  
793  
794  
795  
796

with the true geometric contour of the panel. These irregularities become problematic during training because GarmentSewer employs an edge-sensitive regression loss with higher weights assigned to pixels near boundaries. If the training data contain jagged or discontinuous boundary signals, the model is forced to reproduce these artifacts, which degrades both reconstruction quality and stitching consistency. To mitigate this issue, we adopt a hybrid interpolation scheme: linear interpolation is used along panel boundaries to create smooth, consistent edge values, while barycentric interpolation is applied only within triangle interiors. This produces a dense and smooth geometry field that better reflects the underlying surface and avoids introducing unwanted artifacts into the learning process, see Algorithm 4.

---

**Algorithm 4** Geometry Image Creation from Packed Layout
 

---

**Input:** Panels  $\mathbf{P}$  with mesh vertices and faces, packed layout  $L_{UV}$

**Output:** Geometry image  $GGI_{\text{geo}}$

```

1:  $GGI_{\text{geo}} \leftarrow$  blank image with the same resolution as  $L_{UV}$ 
2: for each panel  $P_i \in \mathbf{P}$  do
3:    $(V_i, F_i) \leftarrow$  vertices and faces of  $P_i$ 
4:    $U_i \leftarrow$  UV coordinates of  $V_i$  from  $L_{UV}$ 
5:
6:   for each vertex  $v \in V_i$  with UV  $u \in U_i$  do ▷ Vertex rasterization
7:      $GGI_{\text{geo}}(u) \leftarrow v$ 
8:   end for
9:
10:   $E_i \leftarrow$  boundary edges of  $P_i$ 
11:  for each edge endpoints  $(v_a, v_b) \in E_i$  do ▷ Edge interpolation
12:     $u_a, u_b \leftarrow$  the corresponding UV coordinates of  $v_a$  and  $v_b$ 
13:    Sample continuous UV coordinates  $\{u_k\}$  along  $u_a$  and  $u_b$ 
14:    for each sampled coordinate  $u_k$  do
15:       $\alpha \leftarrow \frac{\|u_k - u_a\|}{\|u_b - u_a\|}$ 
16:       $GGI_{\text{geo}}(u_k) \leftarrow (1 - \alpha)v_a + \alpha v_b$  ▷ Linear interpolation
17:    end for
18:  end for
19:
20:  for each face  $(v_a, v_b, v_c) \in F_i$  do ▷ Interior barycentric interpolation
21:     $u_a, u_b, u_c \leftarrow$  the corresponding UV coordinates of  $v_a, v_b,$  and  $v_c$ 
22:    Identify all UV coordinates  $\{u_k\}$  inside the triangle of  $(u_a, u_b, u_c)$ 
23:    for each coordinate  $u_k$  do
24:       $GGI_{\text{geo}}(u_k) \leftarrow \text{barycentric\_interpolation}(v_a, v_b, v_c, u_a, u_b, u_c, u_k)$ 
25:    end for
26:  end for
27: end for
28: return  $GGI_{\text{geo}}$ 

```

---

## B.2. Postprocessing pipeline

**Remeshing** Given the predicted geometry image, the first step in the postprocessing pipeline is to recover a valid triangular mesh for each panel. As shown in Fig. B.2, we perform remeshing directly in UV space by scanning the geometry image in a grid-aligned manner. For every  $2 \times 2$  UV cell, we examine the occupancy of its four pixels and generate either one or two triangles depending on how many valid vertices are present. When all four pixels contain valid geometry, we select the diagonal that yields the shorter 3D distance, ensuring a consistent and well-shaped triangulation. The pseudo-code is provided in Algorithm 5. All triangles are constructed with clockwise orientation so that the resulting face normals sharing the same normal with geometry image  $\vec{n}_{GGI_{\text{geo}}}$  and follow a consistent outward normal direction (right side of the garment)  $\vec{n}_{\text{out}}$ , which is crucial for later stitching and rendering. This UV-aligned remeshing produces a dense and topologically clean mesh for each panel without needing a physics-based surface reconstruction step.



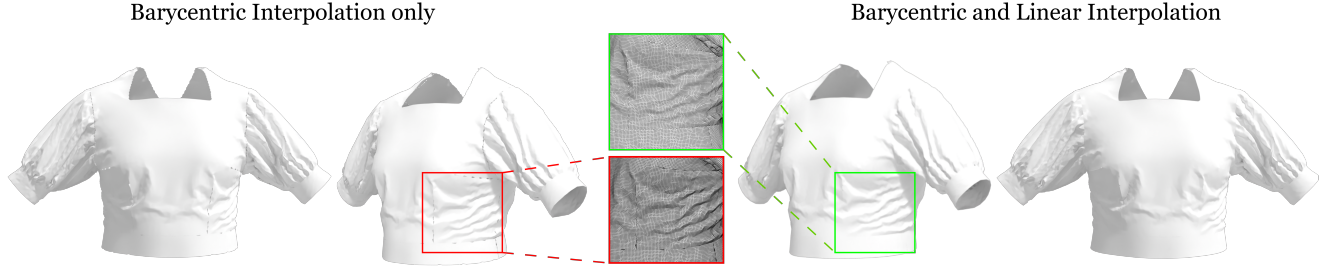


Figure B.1. Effect of interpolation schemes on the geometry image. We first rasterize the mesh into a geometry image using different interpolation strategies and then remesh it back into 3D. Barycentric interpolation alone (*left*) introduces jagged and discontinuous boundary values that deviate from the true panel contour. Our hybrid interpolation (*right*), which applies linear interpolation along panel edges and barycentric interpolation only inside triangle interiors, produces smooth and consistent boundary signals, preventing these artifacts from propagating into GarmentSewer predictions.

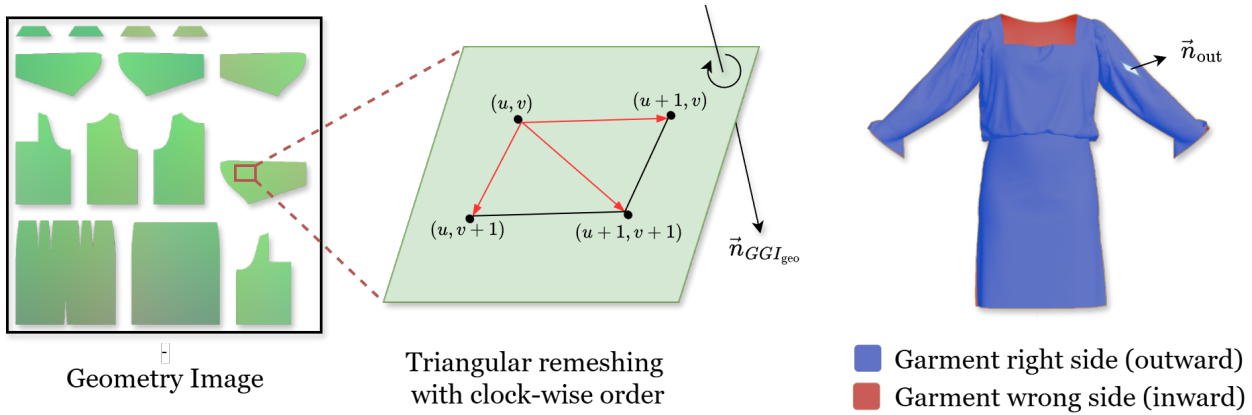


Figure B.2. Remeshing from the geometry image. Starting from the UV-aligned geometry image, we perform local triangular remeshing by examining each  $2 \times 2$  UV cell and generating one or two triangles depending on valid vertex occupancy. When all four vertices are present, the diagonal yielding the shorter 3D distance is selected. All faces are constructed in clockwise order to ensure consistent outward-facing normals across panels.

**Stitching** After remeshing individual panels, we restore global garment connectivity using the stitching image. Fig. B.3 shows the resulting improvement before and after stitching, including zoomed-in wireframe views of the seam regions. Each stitched pair of edges is first extracted from the stitching image and aligned using Dynamic Time Warping to obtain a one-to-one correspondence along their UV boundary curves. The corresponding 3D vertices are then merged through a disjoint-set union, followed by averaging the vertex positions to ensure geometric consistency at the seam. Finally, degenerate faces arising from the merge are removed. This stitching step produces watertight and smoothly connected panel boundaries, removing the discontinuities that would otherwise appear in the initial, panel-wise remeshed output (see Algorithm 6). Together with the remeshing stage, this completes the conversion from the predicted Garment Geometry Image into a coherent and simulation-ready 3D garment mesh.

814  
815  
816  
817  
818  
819  
820  
821  
822

---

**Algorithm 5** Remeshing from Geometry Image

---

**Input:** Geometry image  $GGI_{\text{geo}}$ **Output:** Vertex array  $V$ , face array  $F$ , occupancy map  $O$ , vertex index map  $I$ 

```

1: for each UV coordinate  $u$  do
2:    $O(u) \leftarrow 1$  if  $GGI_{\text{geo}}(u)$  contains a valid 3D vertex, else 0
3:   if  $O(u) = 1$  then
4:      $I(u) \leftarrow$  assign a unique vertex index
5:      $V[I(u)] \leftarrow GGI_{\text{geo}}(u)$ 
6:   end if
7: end for
8:  $F \leftarrow \emptyset$ 
9: for  $x = 0$  to  $H - 2$  do
10:  for  $y = 0$  to  $W - 2$  do
11:     $(x_0, y_0) \leftarrow (x, y)$ ,  $(x_1, y_1) \leftarrow (x + 1, y)$ 
12:     $(x_2, y_2) \leftarrow (x, y + 1)$ ,  $(x_3, y_3) \leftarrow (x + 1, y + 1)$ 
13:     $\mathcal{S} \leftarrow \{(x_0, y_0), (x_1, y_1), (x_2, y_2), (x_3, y_3)\}$ 
14:     $\mathcal{S}_{\text{valid}} \leftarrow \{(x', y') \in \mathcal{S} \mid O(x', y') = 1\}$ 
15:    if  $|\mathcal{S}_{\text{valid}}| < 3$  then
16:      continue
17:    else if  $|\mathcal{S}_{\text{valid}}| = 3$  then ▷ One triangle
18:      Let  $(x_a, y_a), (x_b, y_b), (x_c, y_c)$  be the three valid pixels
19:      Order  $(I(x_a, y_a), I(x_b, y_b), I(x_c, y_c))$  clockwise
20:      Add the triangle to  $F$ 
21:    else ▷ Two triangles
22:       $i_{00} \leftarrow I(x_0, y_0)$ ,  $i_{10} \leftarrow I(x_1, y_1)$ 
23:       $i_{01} \leftarrow I(x_2, y_2)$ ,  $i_{11} \leftarrow I(x_3, y_3)$ 
24:       $d_1 \leftarrow \|V[i_{00}] - V[i_{11}]\|$ 
25:       $d_2 \leftarrow \|V[i_{10}] - V[i_{01}]\|$ 
26:      if  $d_1 \leq d_2$  then
27:        Add faces  $(i_{00}, i_{10}, i_{11})$  and  $(i_{00}, i_{11}, i_{01})$  to  $F$ 
28:      else
29:        Add faces  $(i_{00}, i_{10}, i_{01})$  and  $(i_{10}, i_{11}, i_{01})$  to  $F$ 
30:      end if
31:    end if
32:  end for
33: end for
34: return  $(V, F, O, I)$ 

```

---

**Algorithm 6** Stitching Panels via Dynamic Time Warping and Disjoint Set Union**Input:** Stitching image  $GGI_{\text{stitch}}$ , vertices  $V$ , faces  $F$ , vertex index map  $I$ **Output:** Updated vertices  $V$  and faces  $F$ 

- 1: Extract boundary UV coordinates from  $GGI_{\text{stitch}}$  and group them by stitch identifier
- 2:  $\mathcal{E} \leftarrow \emptyset$
- 3: **for** each stitched edge pair  $(E_a, E_b)$  **do**
- 4:    $\mathcal{C} \leftarrow \text{Dynamic\_Time\_Warping}(E_a, E_b)$
- 5:   **for** each correspondence  $(u_a, u_b) \in \mathcal{C}$  **do**
- 6:     Add  $(I(u_a), I(u_b))$  to  $\mathcal{E}$
- 7:   **end for**
- 8: **end for**
- 9: Initialize DSU over all vertex indices
- 10: **for** each  $(i_a, i_b) \in \mathcal{E}$  **do**
- 11:   Union( $i_a, i_b$ )
- 12: **end for**
- 13: Merge vertices in  $V$  according to DSU representatives by averaging their 3D vertex coordinates
- 14: Update  $F$  by replacing each index with its representative and removing degenerate faces
- 15: **return**  $V, F$

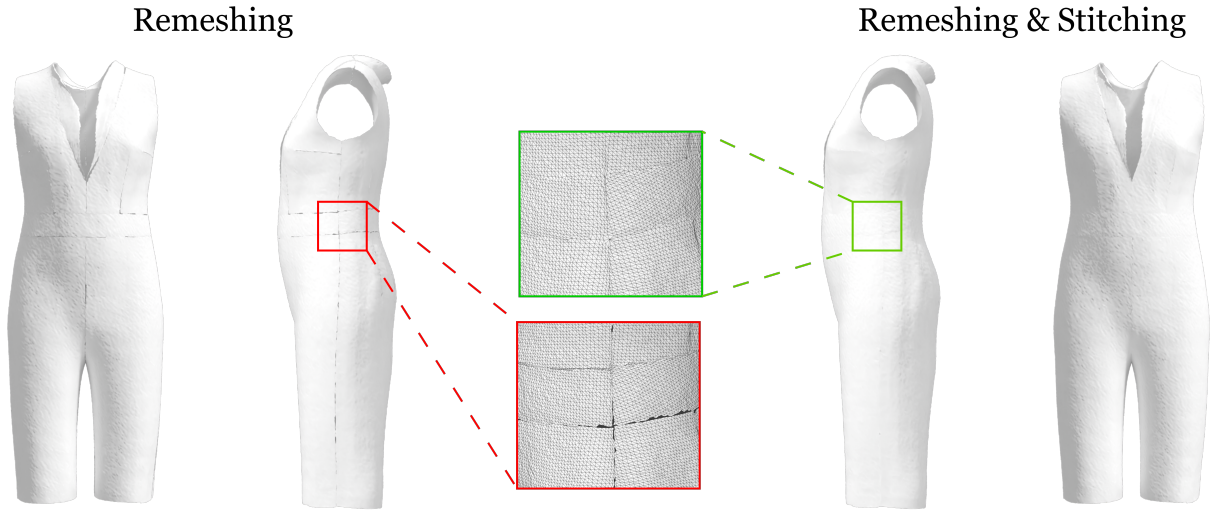


Figure B.3. Stitching results before and after seam alignment. Using the stitching image, boundary edges are paired and aligned via Dynamic Time Warping, followed by vertex merging through a disjoint-set union. The zoomed-in wireframe views highlight how stitching resolves discontinuities and removes gaps between corresponding panel edges, achieving globally coherent garment mesh. The example is conducted on predicted sewing pattern from PatternMaker.

## C. Experiments

### C.1. Experiment setup

**Data Setup** We evaluate all methods on the test split of GCDMM [31], an extended version of GarmentCodeData [17] that includes text prompts and editing instructions. This split contains more than 5,000 samples and is used for assessing sewing pattern generation accuracy. Since computing 3D metrics and running full garment construction is significantly more expensive, we randomly extract a subset of 500 samples from test set for evaluating 3D garment generation. For SewingLDM [26] under the image-conditioned setting, we follow the authors’ instructions and extract garment-only sketches from the front-view image, excluding the SMPL body from the scene.

**Sewing Pattern Generation** In the supplementary, we report sewing pattern generation results using image only inputs. The pattern generation metrics, including panel count accuracy, edge accuracy, and rigid transformation errors, require the model to recover precise structural details of the garment. Text inputs alone do not provide sufficiently strong geometric cues to guide any existing method toward predicting the exact panel layout or edge topology. For this reason, we exclude the text only setting from our pattern generation evaluation.

**Garment Mesh Generation** Our goal in this stage is to measure how reliably each method can convert a predicted sewing pattern into a valid 3D garment mesh. Because predicted patterns are not always directly convertible, each pattern generator is given up to 20 attempts to produce a pattern that successfully reconstructs into a mesh. The first successful attempt is used for evaluation; if no valid reconstruction is obtained within the budget, the result is recorded as an empty point cloud. In tables of this section, we also report the average number of sampling until success. The supplementary further presents results under single condition inputs, including text only and image only settings, to isolate the contribution of each type of conditioning information. This allows us to analyze how well each baseline and our SwiftTailor pipeline perform when restricted to a single source of information.

### C.2. Sewing Pattern Generation

**Image-guided Pattern Generation.** For the image-guided setting, the visual input provides strong cues about panel shapes and garment structure. As shown in Tab. C.1, our method achieves the lowest geometric errors across all continuous parameters and the highest accuracy on discrete structural components. Compared to AIpparel[31], ChatGarment[1], and SewingLDM[26], our model more reliably recovers the correct number of panels, edge configurations, and stitching relations from a single garment image. These improvements highlight the effectiveness of PatternMaker in leveraging image features for fine-grained structural reasoning and producing sewing patterns.

Table C.1. Quantitative results on sewing-pattern generation with image condition only. Best results are shown in **bold**.

Method	Vertex L2 ( $\downarrow$ )	#Panel Acc ( $\uparrow$ )	#Edge Acc ( $\uparrow$ )	Rot L2 ( $\downarrow$ )	Transl L2 ( $\downarrow$ )	#Stitch Acc ( $\uparrow$ )
AIpparel [31]	5.18	89.94	75.76	0.007	2.51	71.04
ChatGarment [1]	16.47	14.05	39.08	0.057	19.99	30.58
SewingLDM [26]	19.41	15.42	42.77	0.107	25.04	28.17
<b>Ours</b>	<b>3.70</b>	<b>91.04</b>	<b>88.96</b>	<b>0.006</b>	<b>1.91</b>	<b>83.69</b>

### C.3. Garment Mesh Generation

**Image-guided 3D Garment Generation.** Our method achieves the lowest MMD and the highest coverage under image-only conditioning in Tab. C.2, showing that it reconstructs 3D garments that are both closer to the reference distribution and more diverse. Compared to pairing PatternMaker with GarmentCode[16], replacing the construction stage with GarmentSewer reduces MMD (from 6.82 to 5.23) and improves COV (from 0.56 to 0.68).

**Text-guided 3D Garment Generation.** Tab. C.3 reports results for text-only conditioning with weaker geometric cues. Our approach still improves over other pipelines in MMD, while maintaining similar coverage as PatternMaker + GarmentCode. The gap between methods is smaller than in the image-only case, which aligns with the difficulty current pattern generators

Table C.2. Quantitative results on mesh generation using image condition only. Best results are shown in **bold**.

Method	MMD ↓	COV ↑	#Sampling ↓
AIpparel[31] + GC[16]	6.95	0.52	3.93
ChatGarment[1] + GC[16]	11.64	0.22	<b>1.31</b>
SewingLDM[26] + GC[16]	10.56	0.37	2.07
<b>PatternMaker + GC[16]</b>	6.82	0.56	2.93
<b>Ours</b>	<b>5.23</b>	<b>0.68</b>	2.93

face when inferring precise panel geometry from textual descriptions. Nevertheless, once a plausible pattern is obtained, the proposed construction stage consistently produces reliable 3D meshes compared to physics engine.

Table C.3. Quantitative results on mesh generation using text condition only. Best results are shown in **bold**.

Method	MMD ↓	COV ↑	#Sampling ↓
AIpparel[31] + GC[16]	8.59	0.39	3.76
ChatGarment[1] + GC[16]	12.89	0.20	<b>1.15</b>
SewingLDM[26] + GC[16]	19.97	0.26	4.80
<b>PatternMaker + GC[16]</b>	8.58	<b>0.43</b>	2.70
<b>Ours</b>	<b>7.80</b>	0.42	2.70

**Modular Exchanges.** Tab. C.4 evaluates modularity by replacing GarmentCode with our GarmentSewer in the second stage. For all pattern generators except ChatGarment [1], plugging in GarmentSewer yields a clear improvement in final mesh quality, with lower MMD and slightly higher COV, while keeping the number of sampling attempts unchanged. This demonstrates that our construction module can be seamlessly integrated into existing pipelines to enhance 3D reconstruction performance without modifying the upstream components. For ChatGarment [1], which outputs coarse attribute-based patterns, the effect remains limited, reflecting upstream representation constraints rather than limitations of the construction stage. Pairing SewingLDM [26] with GarmentSewer also produces a notable drop in MMD compared to SewingLDM [26] + GarmentCode [16], and the full SwiftTailor pipeline combining PatternMaker with GarmentSewer achieves the best overall balance of MMD, COV, and sampling cost among all combinations.

Table C.4. Quantitative results of all combinations between pattern generator and garment constructor on mesh generation using multi-modal inputs (image and text).

Method	MMD ↓	COV ↑	#Sampling ↓
AIpparel [31] + GC [16]	6.94	0.52	4.27
AIpparel [31] + GarmentSewer	6.03	0.63	4.27
ChatGarment [1] + GC [16]	12.27	0.22	1.20
ChatGarment [1] + GarmentSewer	12.56	0.23	1.20
SewingLDM [26] + GC [16]	11.33	0.34	5.87
SewingLDM [26] + GarmentSewer	10.96	0.41	5.87
<b>PatternMaker + GC [16]</b>	6.82	0.54	2.98
<b>SwiftTailor (Ours)</b>	5.31	0.68	2.98

## C.4. Qualitative Results

**Qualitative Comparison between GarmentSewer and GarmentCode.** Fig. D.1 illustrates the differences between GarmentCode[16] and GarmentSewer when reconstructing a 3D garment from the same predicted sewing pattern produced by PatternMaker. GarmentCode[16] follows a rule-based pipeline that places 2D panels around the body and stitches them

through a physics-driven sewing process. This initialization often leads to unfavorable starting states, such as panels intersecting the body or being arranged with incorrect relative orientation. As a result, the subsequent simulation struggles to recover a stable configuration, which can produce collapsed folds, tangled regions, or unrealistic draping.

In contrast, GarmentSewer directly reconstructs a geometry-image representation of the final garment silhouette, providing a stable and coherent 3D initialization before refinement. Because the mesh is already globally consistent at the start, the local relaxation during post-processing only needs to resolve minor geometric adjustments rather than repairing major structural errors. This allows GarmentSewer to preserve the intended panel relationships more faithfully and produce garments with cleaner silhouettes, smoother draping, and more reliable seam alignment. The qualitative differences across diverse garment types in Fig. D.1 highlight that GarmentSewer can avoid the failure modes seen in GarmentCode [16], especially in cases involving asymmetric patterns or complex multi-panel structures.

**More qualitative results** We provide additional qualitative examples produced by our pipeline under multimodal inputs. These results are presented in Fig. D.2.

## D. Discussions

Despite the high-fidelity results and efficient inference enabled by our pipeline, several limitations remain. A key challenge is the absence of high-frequency wrinkles in the reconstructed meshes. This limitation does not arise from the geometry-image representation itself, but from the behavior of GarmentSewer during training. The model naturally learns to smooth out fine geometric variations in order to preserve global structure and ensure stable reconstruction, which results in clean and visually coherent meshes but suppresses subtle wrinkles and fold patterns. While these smooth meshes are suitable for visualization and garment showcasing, restoring realistic high-frequency wrinkles remains an open problem. One promising direction is to apply a lightweight physics-based refinement or learning-based approach to restore wrinkle on top of our stable initialization, enabling wrinkle recovery without relying on full-scale simulation.

Another limitation lies in the robustness of the pipeline under challenging, in-the-wild inputs. PatternMaker and the downstream reconstruction stages are designed around curated datasets with relatively clean observations and well-structured garments. When confronted with complex backgrounds, occlusions, unconventional silhouettes, or garments far outside the training distribution, failure cases become more frequent. Extending the system to handle broader visual variability through stronger vision encoders, data augmentation, or explicit garment parsing will improve reliability in real-world scenarios.

Finally, the modular design of our pipeline creates opportunities for future extensions. Examples include reconstructed garments with material properties or support realistic downstream simulation, or integrating user-driven editing interfaces that operate directly on predicted meshes. These directions broaden the scope from reconstruction toward interactive and controllable garment modeling.



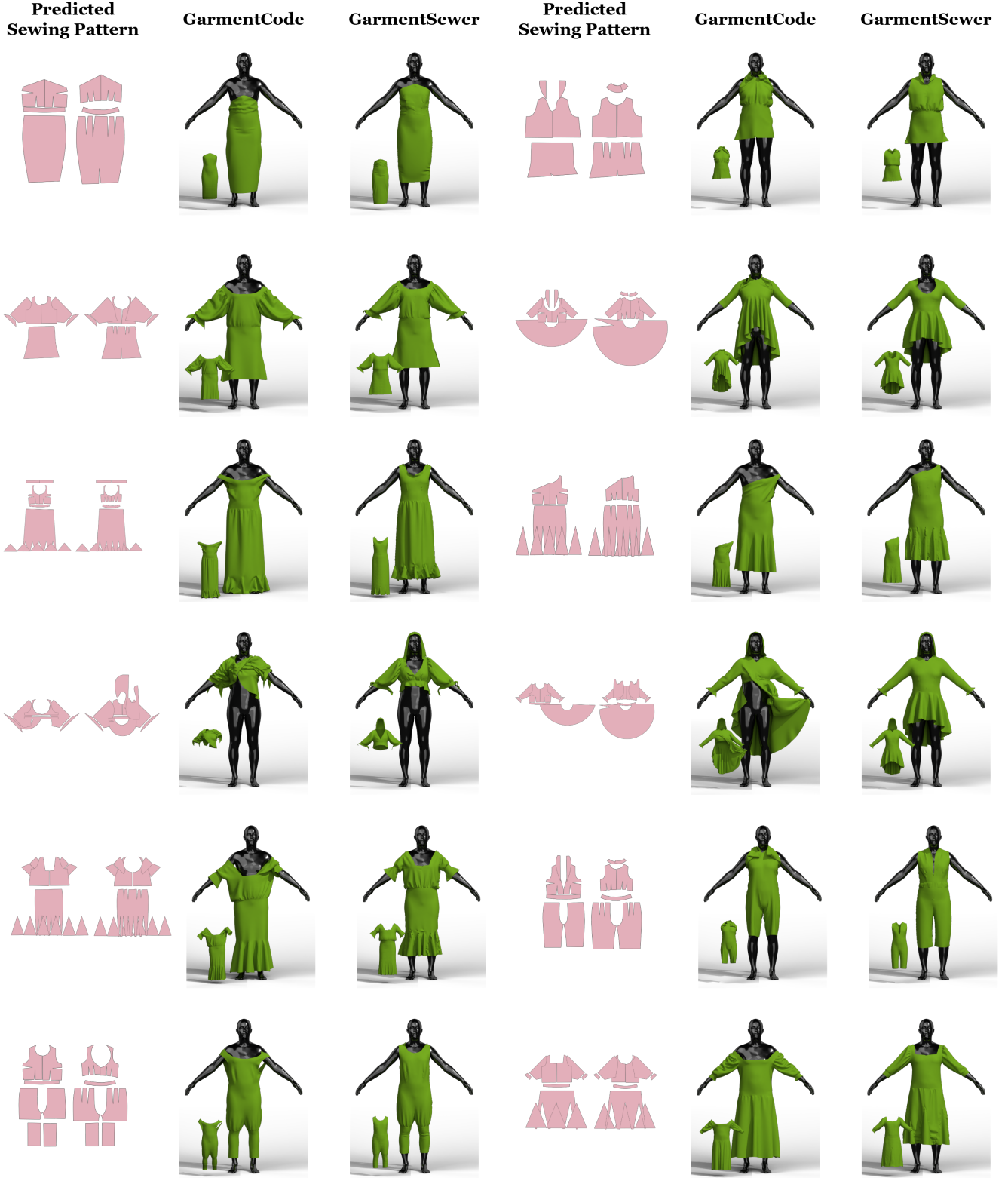


Figure D.1. Qualitative comparison between GarmentCode [16] and GarmentSewer given the same sewing patterns predicted by Pattern-Maker. GarmentSewer produces stable initializations and consistent draping, while GarmentCode[16] often fails due to rule-based panel placement.



Figure D.2. Additional qualitative results from our pipeline. Each example shows the re-draped garment on the SMPL [27] body together with its initial state constructed by GarmentSewer (the smaller mesh on the left). Textures are added to enhance visualization of garment geometry and structure.