

# AIpparel: A Multimodal Foundation Model for Digital Garments – Supplementary Materials

## Contents

S1. Details on GarmentCodeData-Multimodal (GCD-MM)	1
S1.1. Text Description Generation	1
S1.2. Generation of Editing Data Sample	4
S1.3. Sewing Pattern Statistics	5
S2. Implementation Details on AIpparel	6
S2.1. Network Architecture	6
S2.2. Training Details	7
S3. Additional Results And Experiment Details	7
S3.1. Sewing Pattern Prediction from Images	7
S3.2. Sewing Pattern Prediction from Texts	8
S3.3. Sewing Pattern Prediction from Multimodal Input	8
S3.4. Sewing Pattern Editing	11
S3.5. Ablation Study	13
S3.6. Draping Details	14
S3.7. Human Study	14
S4. Discussion	14
S4.1. More Discussion on Limitations and Future Work	14
S4.2. Further Social Impacts	16

## S1. Details on GarmentCodeData-Multimodal (GCD-MM)

We expand on the data curation process of GCD-MM. Specifically, in Sec. [S1.1](#), we detail the specifics of generating text descriptions for the sewing patterns. In Sec. [S1.2](#), we elaborate on how we generate the edited sewing patterns and their associated editing instructions. Lastly, in Sec. [S1.3](#), we show statistics comparing sewing patterns in GCD-MM and sewing patterns in previous datasets used by DressCode [\[3\]](#) and SewFormer [\[12\]](#).

### S1.1. Text Description Generation

We generate two types of sewing pattern descriptions for each garment in GCD-MM. The first is a detailed natural language description of the sewing pattern, while the second outlines a suitable occasion for wearing the garment. For captioning purposes, we use a standardized body type corresponding to the mean shape and pose derived from SMPL.

Obtaining pattern descriptions happens in two steps. First, we generate keywords describing the simulated garments using the design parametrization of each garment. Generated based on GarmentCode [\[7\]](#), each garment is characterized by a set of continuous and categorical parameters. We generate descriptions for each garment using the following rules:



**DressCode:** **jacket**; short sleeves; **with a hood**; fitted garment

**Ours:** An upper-body garment; both sleeves; short sleeves; with lapels

**DressCode:** **trousers**; long length; wide fit; front slit; high waist

**Ours:** A maxi skirt; narrow waistband; skirt with front slit; skirt with back slit; skirt with side slit

Figure S1. **Comparison between our Short Captions and DressCodes’**. This figure shows the short captions created by DressCode and our method for two different garments. DressCode produces keywords that do not align with the garment (red).

- **Categorical parameters:** We assign the categorical label when appropriate. For instance, a `godet skirt` is classified as such. Some categorical parameters do not suffice - a `shirt` can signify anything from a crop top to a dress. For these instances, we add additional checks consulting additional parameters.
- **Continuous parameters:** We define thresholds and assign different qualitative labels for garments above and below them. Parameters such as `sleeve length` or `collar width` are obvious examples.
- **Dependent parameters:** Most parameters have no impact on the final garment, as they only become relevant when certain categorical parameters are set. We design rules that consider these edge cases. Only when a `godet skirt` is set, does the `num inserts` become relevant. We include all relevant dependent parameters that have a structural effect on the garment.

Similar to DressCode, we first generate a garment type description and a collection of keywords that contain the specific description based on our rule-based approach. Note that each rule can contribute several keywords. See Figure S1 for the examples.

In the second stage, we use these generated keywords in combination with a render of the front and back of the garment to prompt GPT-4o. We construct the prompt such that GPT-4o objectively describe the garment using the characteristic features of the garment provided by the generated keywords and renders. In addition, we include instructions to focus on information crucial for our learning problems, such as panel connectivity and stitching patterns, while ignoring irrelevant information, such as colors or interpretations.

The following is the system prompt that we used:

You are a fashion expert tasked with providing concise and neutral descriptions of garments based on the provided textual information. Your descriptions should focus on specific stitching details and how different panels are connected (such as seam placements and stitching patterns), as well as any distinctive characteristics and design elements of the garment. When describing the garment’s appearance, use precise and concrete language, avoiding generic phrases or broad descriptions. Do not mention that seams are visible; instead, describe where seams or panels are located to indicate construction details. Do not include any impressions, subjective interpretations, or unobservable aspects. Avoid mentioning colors or any references to images. Keep the descriptions brief and to the point, avoiding unnecessary words. Use only the information provided.

Here we present the user prompt:

Please generate a concise and neutral description of a garment, focusing on specific stitching details, how different panels are connected, and including any distinctive characteristics and design elements, based on the following information:

- **Title**: {title}

- **Description**: {description}

Provide a brief description that emphasizes stitching and construction details (such as seam placements, panel connections, and stitching patterns), along with precise visual observations about the garment’s appearance, including style, silhouette, length, and any unique design features or distinctive characteristics. Avoid using generic phrases or broad descriptions; instead, provide specific details about the garment’s features. Do not mention that seams are visible; instead, describe where seams or panels are located to indicate construction details. Do not include any impressions, subjective interpretations, or unobservable aspects. Avoid mentioning colors or any references to images. Keep the description succinct and avoid unnecessary words. Use only the information provided.

The second type of caption describes an occasion for which a garment is suitable. In this prompt, we ask the model not to pay attention to the garment’s colors which only highlight different panels and are not semantically relevant. Instead, we ask it to focus on the shape and description. We use the same information as before to prompt GPT-4o. This is the system prompt:

You are an expert in fashion design and garment analysis. When provided with images of garments and their metadata, focus solely on their shape and stitching. Note that different colors in the images represent different panels of the garment and are not indicative of style or color choices. Ignore colors and any visible seams meant only for stitching information. The metadata includes a title and a description, which is a list of short attributes; use these to inform your understanding. Based on this information, provide only a detailed, but concise, description of a single occasion where the given garment would be appropriate to wear. Do not include any other information in your response.

and here is the user prompt:

Given the following garment’s metadata and images (remember that colors and seams are only for panel representation and stitching information), please provide only a detailed, but concise, description of a single occasion where this garment can be worn. Do not include any other information in your response.

Here is the metadata:

Title: {title}

Description: {description}

**Effect of GPT version in caption quality.** While GPT-4o potentially increases the accuracy of generated captions, the in-context knowledge about various design parameters crucially helps the model to generate captions more faithful to the garment design. Fig. S2 shows the same captions in Fig. S1 of supp, generated instead using GPT-4V. Notice that DressCode’s caption contains severe flaws (in red) due to inaccurate in-context prompting. Ours do not have these flaws because we prompt GPT-4V with design-parameter-inspired content. We will update Fig.S1 to include this example in the revision.

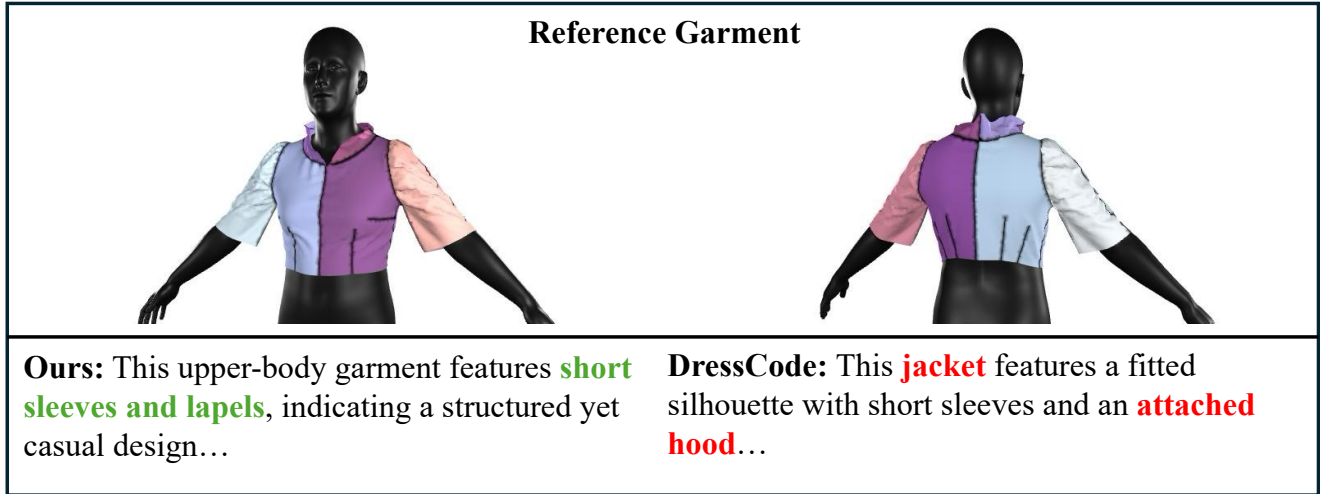


Figure S2. Captions generated using GPT-4V.

## S1.2. Generation of Editing Data Sample

To generate paired garments representing before-and-after edits, we use design parameters from the GCD dataset and systematically apply one of five pre-defined transformation rules. The modified design parameters are then converted into garments using GarmentCode [7].

Each garment from GCD is first evaluated to determine which transformation rules are applicable. One rule is then randomly selected and applied. Due to limitations in GarmentCode’s design space, not all edited design parameters can be converted into sewing patterns. As a result, GCD-MM comprises 120k garment pairs that are successfully generated from the 130k garments in GCD, while approximately 10k garments remain unpaired.

The transformation rules include adjustments to garment lengths (sleeves, pants, skirts), collar type changes, modifications to garment symmetry, toggling the presence of hoods, and structural edits to style elements (e.g., changing the number of inserts in godet skirts). Each rule takes the existing design parameters as input and applies a targeted change. For instance, length adjustments alter sleeves, pants, or skirts by 50% of their initial length, constrained by the maximum length specified in GarmentCode. Similarly, collar types are randomly reassigned from a predefined set, garment symmetry is toggled, and hoods are added or removed.

These rules are designed for three key reasons: (1) they produce clear and concise edits that can be succinctly summarized; (2) they encompass varying levels of editing complexity, from minor panel length adjustments to major structural modifications involving new panels and altered stitching; and (3) for all garments in the dataset, at least one rule can always be applied.

To document each transformation, we generate descriptive sentences for the edited garments using a rule-based approach. Here are a set of examples:

Godet skirt:     *"Increase the number of inserts in the skirt by \$x."*  
Pants:            *"Make the pants longer."*  
Shirt:            *"Switch the collar type from \$currCollar to \$newCollar."*

In total, the defined rules enable 52 distinct, describable modifications, ensuring a diverse and well-documented dataset of garment editing pairs.



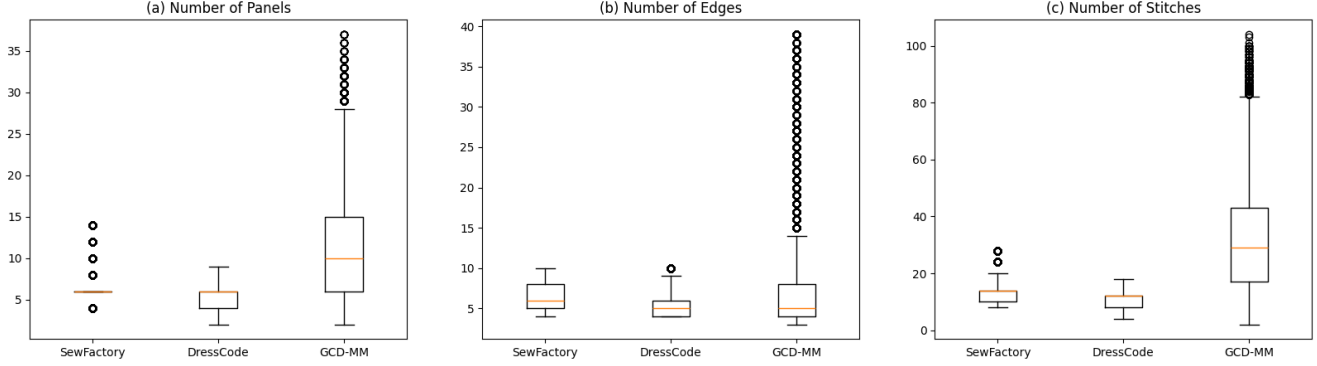


Figure S3. **Dataset statistics comparisons.** Notice that GCD-MM in general contains larger variations in the number of panels, edges, and stitches in the sewing patterns. This poses additional challenges in designing a sewing pattern generation method with GCD-MM.

	SewFactory [12]	DressCode’s Dataset [3, 5]	GCD-MM
Edge Types	L, QB	L, QB	L, QB, CB, A
Number of Sewing Patterns	13700	20292	127629

Table S1. **Dataset Statistics Comparison.** L=Line, QB=Quadratic Beziér, CB=Cubic Beziér, A=Arc. GCD-MM shows a larger variation in both numbers of panels, edges, and stitches than previous sewing pattern datasets. For Panel, edge, and stitching statistics, refer to Figure S3.

### S1.3. Sewing Pattern Statistics

GCD-MM uses sewing patterns fitted on a default body from the GarmentCodeData (GCD) dataset [8], which are procedurally generated sewing patterns using the programming abstraction of GarmentCode [7]. Compared with the sewing patterns used by SewFormer [12] and DressCode [3], GCD contains more complicated and diverse sewing patterns. For detailed documentation and comparison with existing datasets and procedural sewing pattern generators, please refer to GarmentCodeData [8]. Here, we briefly show some statistics comparing these different datasets.

GCD exhibits more diverse and detailed garment feature variations than the previous dataset, including fitted garments, correct sleeve shapes, more collar types, more skirt types, cuffs, and asymmetric features (tops, asymmetric skirt cuts). All of these characteristics make sewing patterns from GCD more complicated than existing sewing pattern datasets.

Comparatively, datasets used by SewFormer [12] and DressCode [3] are procedurally generated sewing patterns from an older programming abstraction [5]. While this programming abstraction can also generate sewing patterns for the types of garments described above, all its variations are from changes in the vertex and control point positions while fixing the number of panels, edges, and stitches the same. This constraint significantly limits the variations exhibited in the datasets used by SewFormer and DressCode. Figure S4 showcases randomly sampled sewing patterns as well as their draped renderings from GCD and sewing patterns used by SewFormer and DressCode. We see that sewing patterns from GCD are generally more complex and diverse than the previous dataset. Table S1 and Figure S3 show a statistical comparison in terms of the number of edges, panels, stitches, and edge types between sewing patterns in GCD-MM, SewFactory [12], and dataset used by DressCode [3]. Notice that comparatively sewing patterns in GCD-MM exhibit the largest variation in all of the statistics, demonstrating the difficulty of the dataset. In particular, because of this difficulty gap, previous methods such as SewFormer and DressCode exhibit poor performance despite fine-tuning their network on GCD-MM. See Section S3 for details.

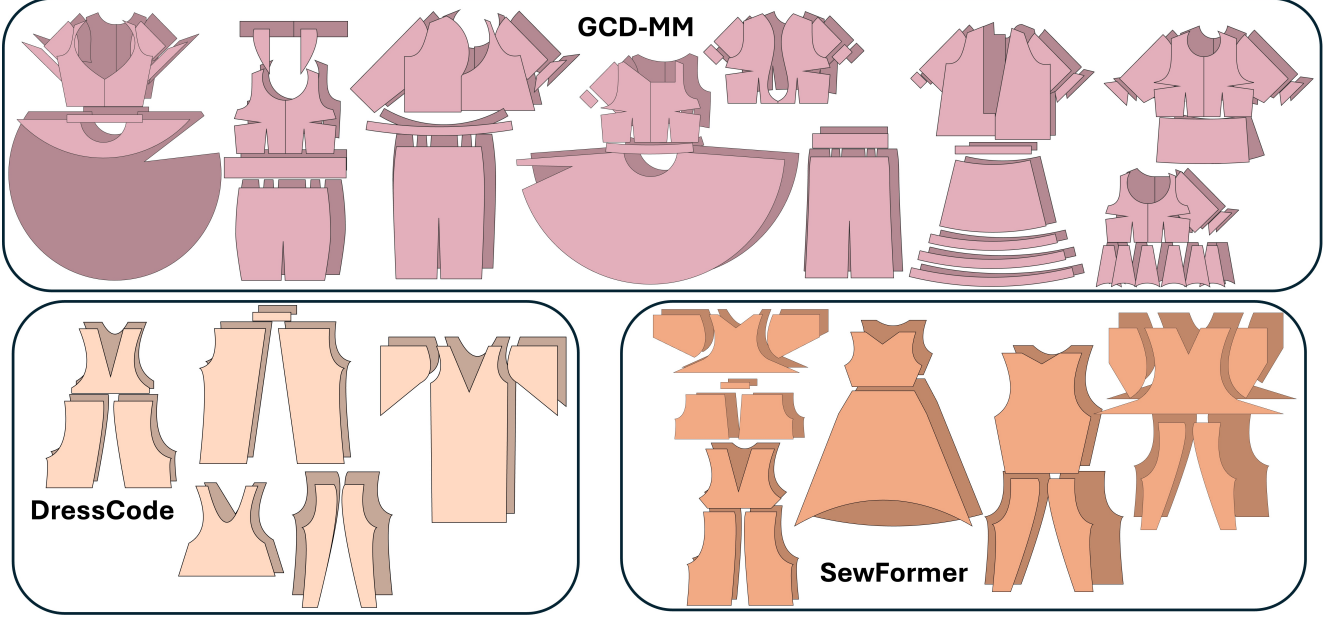


Figure S4. **Visualization of Sewing Patterns.** Random sewing pattern samples from the datasets used by AIppearl and the baselines are visualized. Notice that compared to prior works, GCD-MM exhibits more complex sewing patterns in general.

## S2. Implementation Details on AIppearl

We include details about the network architecture and training hyperparameters of AIppearl.

### S2.1. Network Architecture

As described in Section 3 of the main paper, AIppearl is built on top of LLaVA-1.5 7B [11]. Therefore, the majority of the network, except for the newly added regression heads  $g_{\theta}^{(e)}, g_{\theta'}^{(R)}$ , and the positional embedding projection layers  $h_{\varphi}^{(e)}, h_{\varphi'}^{(R)}$ , are identical to LLaVA-1.5 7B. For completeness, we only summarize key parameter values we used here. Please refer to their paper for architectural details. LLaVA-1.5 7B fine-tunes LLaMA 2 [16] with a vision encoder on a visual question–answer dataset. Specifically, it has a context length of 4049 and a hidden dimension of 4096. Its language model is a 32-layer transformer with 32 head attention layers. Its vision encoder is CLIP [14]. Each image is converted into 255 clip tokens before getting projected into the language model’s embedding space using a custom projector.

To extend LLaVA-1.5 7B for sewing pattern prediction, we expand the vocabulary of the model to include the special tokens defined in Section 3.2 of the main paper. In total, this results in 122 additional tokens added to the vocabulary of LLaVA-1.5 7B. Each of the tokens is initialized to be the average embedding from the existing vocabulary.

Besides additional vocabulary, we also add two additional regression heads  $g_{\theta}^{(e)}, g_{\theta'}^{(R)}$ , and the positional embedding projection layers  $h_{\varphi}^{(e)}, h_{\varphi'}^{(R)}$  to the architecture described above. As described in Section 3 of the main paper, the regression heads will take the output hidden embedding from the language transformer to regress vertex and control point positions using  $g_{\theta}^{(e)}$  and the transformation with  $g_{\theta}^{(R)}$ . Specifically, both of the regression heads are two-layer perceptrons with ReLU non-linearity. Both heads map the 4096-dimensional output embedding to the parameter space. For  $g_{\theta}^{(e)}$ , the output dimension is 8, representing vertex and control points in different channels. Specifically, the first two channels as vertex regression, mapping to the second endpoint of the associated edge. The next four are used for control points to the quadratic and cubic Bézier curves. Finally, if the associated edge is an arc, the last two channels are used to map to an additional point on the arc besides the two endpoints.

During training, only the associated channels for each edge are supervised and the unused channels are masked out for back propagation. With the same architecture,  $g_{\theta}^{(R)}$  has an output dimension of 7, with the first 3 being the translation and the last four being the rotation represented in quaternion.

Finally, the positional embedding layers are also two-layer perceptions with ReLU non-linearity.  $h_{\varphi}^{(e)}$  maps the 2-dimensional vertex coordinate to a 4096-dimensional hidden embedding. The output is then added to that edge type token’s vocabulary embedding before inputting through the language transformer. Similarly,  $h_{\varphi}^{(R)}$  maps the 7-dimensional transformation for each panel to a 4096-dimensional hidden embedding. Then the output is added to the vocabulary embedding of the transformation token  $\langle R \rangle$ .

Both the regression heads and the positional embedding projection layers are initialized to have zero weights in the final layer so that the output before fine-tuning is unaltered.

## S2.2. Training Details

Alpparel is trained for a total of 12,750 steps with a total batch size of 320, and a learning rate of 0.00005 with cosine learning rate decay to zero in 15,000 steps. We also warm-start the fine-tuning from zero learning rate to the default in the first 100 steps. We use  $\lambda = 0.1$  to balance the regression losses and the cross-entropy loss in Equation (2) of the main paper. We use DeepSpeed ZeRO Stage 2 [15] to parallelize the training on  $8 \times H100$  GPUs. The entire training took around 312 H100 GPU hours. We train on all modalities in our GCD-MM jointly. Specifically, we include four different modalities from GCD-MM: *text*  $\rightarrow$  *sewing pattern*, *image*  $\rightarrow$  *sewing pattern*, *text and image*  $\rightarrow$  *sewing pattern*, and *sewing pattern and editing instruction*  $\rightarrow$  *edited sewing pattern*. During each training step, the batch is formed by randomly sampling each of the four modalities with a preset sampling ratio. Specifically, we sample images, texts, image+text, and editing data with the ratio of 3:2:4:1. We randomly split our dataset into 90%, 5%, and 5% for training, validation, and testing. All of our qualitative results are samples from the testing split. While previous works [3, 6, 12] use relative coordinates to represent the control point coordinate, we use absolute coordinates to represent the additional edge parameters. Prior to training, we normalize vertex coordinates and transformation using the global mean and standard deviation computed from all sewing patterns in GCD-MM. Additionally, for input to the positional embedding projection layers, we discretize the input into 256 discrete values ranging between  $\pm 4$  standard deviation values for robustness during generation.

## S3. Additional Results And Experiment Details

We detail the experiment setup and baselines for the result section (Section 5 of the main paper). Further, we also include additional ablation results and qualitative comparisons.

### S3.1. Sewing Pattern Prediction from Images

**Setup & Baseline Details.** We will describe the image-to-garment prediction experiment showcased in Section 4.1 of the main paper in detail. We will also report comparisons on two datasets: GCD-MM and SewFactory.

For GCD-MM, we use our model trained with multimodal data described in Section S2.2 to evaluate the qualitative and quantitative results showcased in Table 2 and Figure 3 of the main paper. To compare with SewFormer [12], we adapt its pre-trained model for sewing pattern prediction on GCD-MM. Specifically, we expand the per-panel query embedding from its default number of 23 to 75 to accommodate all the different panel classes present in GCD-MM. We initialize the newly added panel query embeddings as the average embedding from the pre-trained weights. Similarly, we expand the per-edge embedding from 14 to 39. Furthermore, because GCD-MM contains cubic Bézier curves and arcs, which the SewFactory dataset does not have, we also extend per-edge parameterization from using four channels (2+2: endpoint + optional quadratic Bézier control points) to seven channels (2+4+1: endpoint, control point parameters, arc flag). Specifically, the arc flag takes a value of 0 or 1, indicating if the edge is an arc. If the arc flag is 1, the first two control points would take a value equal to the relative coordinate of the third point on the arc. If the arc flag is zero, then the four channels will be the relative

coordinates of the two control points in the Bézier curve. We keep the network architecture the same except for the above modifications. We fine-tune the pre-trained SewFormer model adapted as above for a total of 16 epochs on the same training split AIpparel is trained on, using a learning rate of 0.00005 and a batch size of 8 on 2×Quadro RTX 8000 GPUs. Except for these, we use the default hyperparameters provided by SewFormer. The validation loss no longer increases after 16 epochs, so we stop the training and use it for comparison.

For comparison on SewFactory, we use the pre-trained SewFormer model as our baseline. However, because the SewFormer authors did not release their train and test split, we show a comparison on a custom test set for this experiment. Specifically, we first train AIpparel on SewFactory data, with a different random split, from scratch for a total of 3750 steps on 8×A100 GPUs using the same hyperparameter settings as described in Sec. S2.2. Then, we evaluate our model on the custom test set. In this way, we ensure a fair comparison with the baseline as the test set should contain a mixture of training and testing examples for both methods.

**Additional Qualitative Visualization.** Figure S5 showcases additional image-to-garment prediction result comparisons to the SewFormer baseline in both the GCD-MM (left) and SewFactory (right) datasets. Our model in general predicts more correct sewing patterns following the guidance of the input image than SewFormer.

**Sewing Pattern Prediction from In-the-wild MultiModal Inputs.** While AIpparel is trained on procedurally generated sewing patterns and annotations, it is able to generalize the in-the-wild input due to the large-scale data it trains on, as well as the world-level knowledge that it inherits from the large multimodal model. Fig. S7 showcases our model’s sewing pattern prediction from an in-the-wild image with GPT-generated text descriptions.

### S3.2. Sewing Pattern Prediction from Texts

We showcase additional text-to-sewing pattern generation visualization from AIpparel in Fig. S6. Notice that our method is able to output correct sewing patterns from long, detailed text descriptions. Moreover, our generated sewing patterns also closely follow the key characteristics described in the text input.

### S3.3. Sewing Pattern Prediction from Multimodal Input

**Setup.** For our multimodal evaluation, we utilize 20 samples for each of the following modality combinations: (1) image, (2) text, (3) image + text, (4) occasion, and (5) editing. These samples are generated following the procedure outlined in Section S1. To ensure proper testing, these test samples are entirely distinct from the training and validation sets used in other experiments.

To benchmark our method, we compare it against two state-of-the-art baselines: SewFormer and DressCode. SewFormer processes image-based inputs, while DressCode is designed for text-based inputs. Since these baselines are limited to specific modalities, we convert multimodal inputs into formats compatible with their architectures. For SewFormer, we use DALL-E 2 to generate a single 512x512 image from non-image inputs using tailored prompts. For DressCode, we convert inputs into keyword-based formats with GPT-4o.

The evaluation of our method and these baselines is conducted using Garment Accuracy, a metric defined as the product of Panel Accuracy and Edge Accuracy, which quantifies the percentage of garments reconstructed with the correct number of panels and edges. Additionally, we measure the squared distance between the predicted and ground-truth vertex positions to assess the geometric accuracy of the reconstructions.

**Baselines.** To generate an image input from a non-image modality, we use DALL-E 2 to produce a single 512x512 image. The prompt used for generation always begins with:

Create an image of a single garment worn by a mannequin. The mannequin should be front-facing and in t-pose.

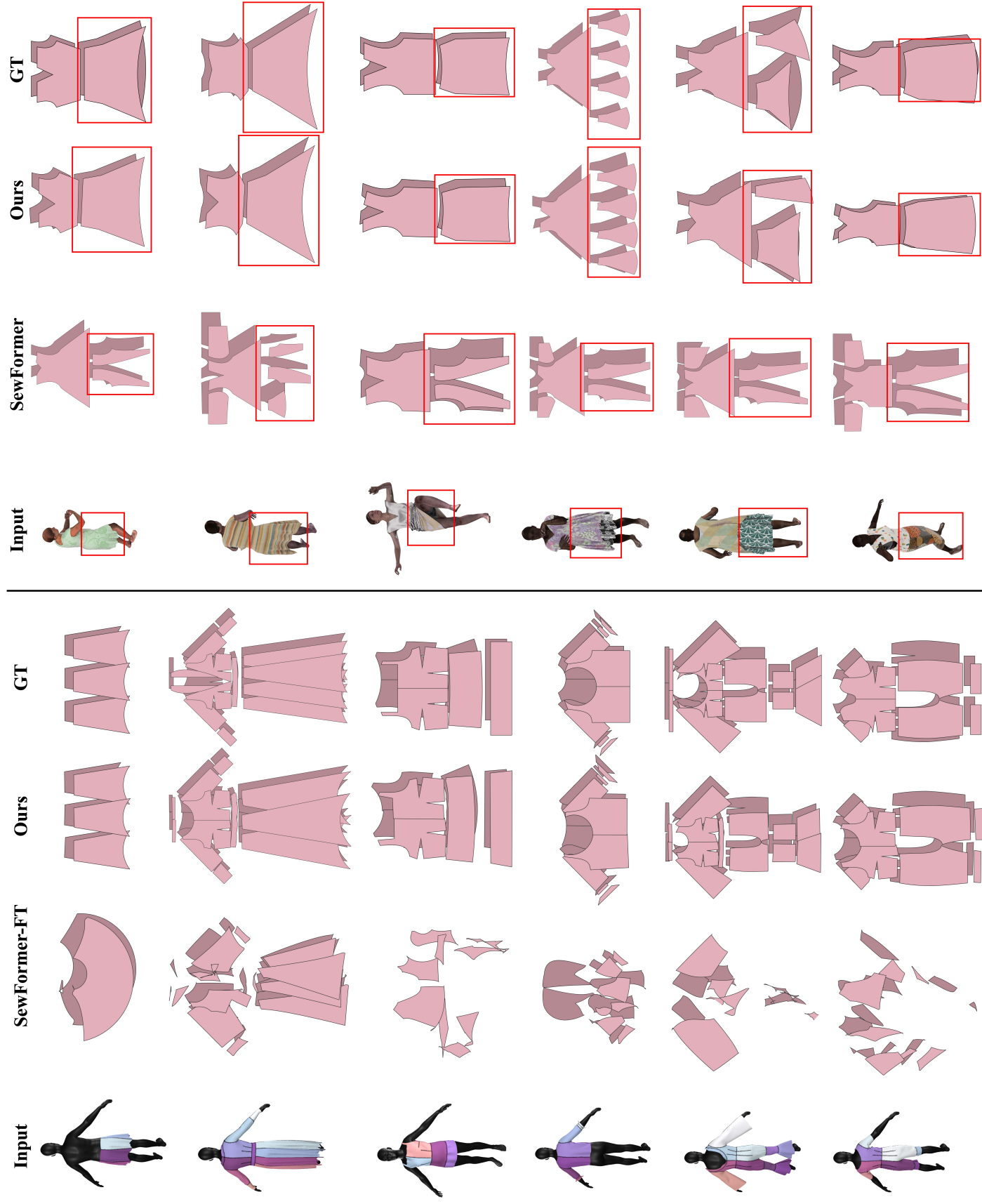


Figure S5. Additional Image to Sewing Pattern Visualizations.



## Input Text

The garment is a **sleeveless hoodie** with a **deep cut neckline**. The hood is attached to the main body with seams running along the back neckline and shoulder areas. **The front panel includes a deep neckline, with seams attaching it to the side and shoulder panels.** The side panels extend under the arms and connect to the back panel, which comprises multiple sections joined with vertical seams. **The silhouette is fitted**, accentuating the upper torso, while the hood adds a distinctive layered look to the upper garment.

The **dress** is a **mini-length garment** featuring a **sleeveless** design with a **short square neckline** and a short square back. It includes a **wide waistband** positioned at a **high rise**. The front and back panels connect seamlessly at the shoulders and sides, with the waistband acting as a horizontal dividing panel, emphasizing the high-rise waistline. **The overall silhouette is fitted**, with the bodice and skirt sections distinctly separated by the wide waistband, ensuring a structured and defined shape.

The **jumpsuit** features a **maxi silhouette with long sleeves**. It has an **extended neckline creating a deep cut at the front**. The garment is constructed with multiple panels joined throughout, including horizontal and vertical connections. **The sleeves are attached at the shoulder seams and feature a wide, loose fit.** **The lower part of the jumpsuit includes flared panels** that add volume to the hem. The design displays a combination of fitted and loose sections, enhancing the overall silhouette with distinctive seam placements.

The **dress** features an **asymmetric design with a single right long sleeve**. The **neckline is also asymmetric**, providing a unique contour to the upper section. Panels are carefully constructed to achieve the asymmetric top, with precise seams joining the right sleeve to the body. **The silhouette is elongated with a clean, straight cut extending to full length.** Distinctive characteristics include the single sleeve and the non-traditional neckline, producing a visually striking and modern appearance.

The **jumpsuit** features a **maxi silhouette** with **long sleeves** and a **short v-neckline** in both the front and back. **The front panel is divided by a central seam running from the neckline to the waist**, where it meets the **waistband seam**. The bodice and lower body panels are connected at the waistband, with additional **vertical seams along the torso sides extending to the hem**. The back contains a yoke panel that connects to the lower body panels with vertical seams. The sleeves are attached to the bodice with shoulder seams extending to the cuffs.

## Prediction

## GT

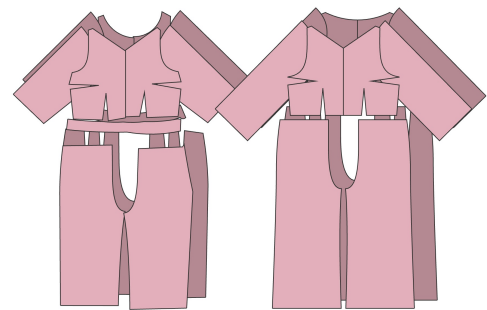
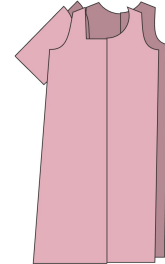
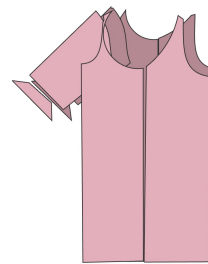
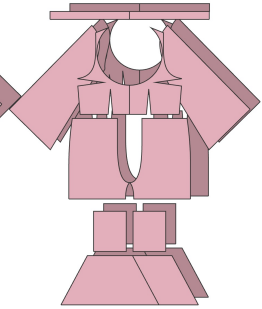
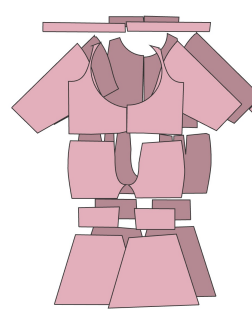
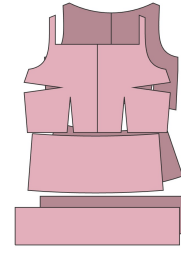
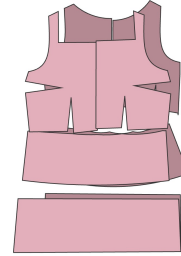
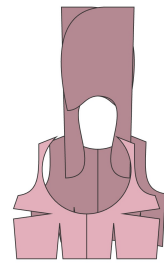
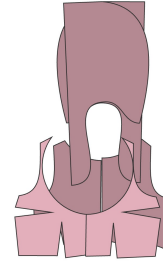


Figure S6. **Text-conditioned sewing pattern generation.** AIpparel generates accurate sewing patterns closely following the text descriptions. Notice that the characteristics described in the bolded phrases all appear in the generated sewing patterns.

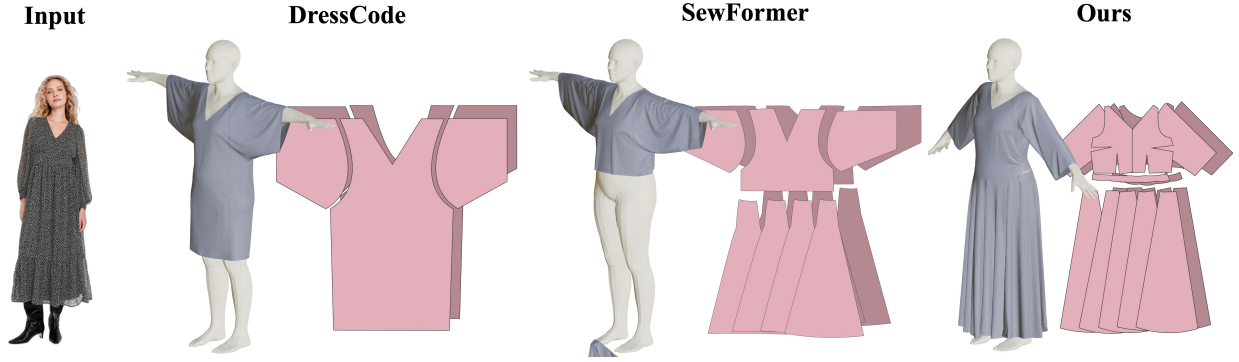


Figure S7. **In-the-wild Image to Garment Example.** Our model is able to predict a sewing pattern more aligned with the input image compared to the baselines. Notice that SewFormer did not drape correctly, resulting in a missing bottom.

The prompt is tailored to each input modality by appending the following continuations.

- **Text:** Make sure that the garment follows this description: + text
- **Occasion:** Make sure the garment suits the following occasion: + text
- **Editing:** Make sure the garment looks like if this edit + edit + was applied to the garment.

Similarly, to convert any input modality into a keyword-based format compatible with DressCode, we design distinct prompts based on the modality. Each prompt is constructed as a concatenation of the following starting phrase:

Describe the garment in a list of comma separated keywords. Give a maximum of 5 keywords.

and a modality specific continuation:

- **Text:** Make sure that the garment follows this description: + text
- **Image:** Make sure that the garment looks like this image.
- **Text + Image:** Make sure that the garment looks like this image and follows this description: + text.
- **Occasion:** Make sure the garment suits the following occasion: + text
- **Editing:** Make sure the garment looks like if this edit + edit + was applied to the garment.

### S3.4. Sewing Pattern Editing

We detail the baseline methods we used for Table 4 and Figure 5 in the main paper. Using existing models, we extend SewFormer and DressCode to translate the sewing pattern and editing instructions to their input domains. Specifically, for SewFormer, we take the editing instruction and rendered image from GCD-MM and translate the rendering image using a pre-trained InstructPix2Pix [1] with the editing instruction as input. The output from InstructPix2Pix is a garment image generated based on the editing instructions and the input rendering. With this input image, we query the SewFormer-FT baseline to obtain the final sewing pattern. For DressCode, we use GPT4V to translate the editing instructions and rendered image into short keywords that describe the edited garment. This is then used to query the pre-trained DressCode and obtain the sewing pattern. The text prompt we use for querying GPT4V is the following:



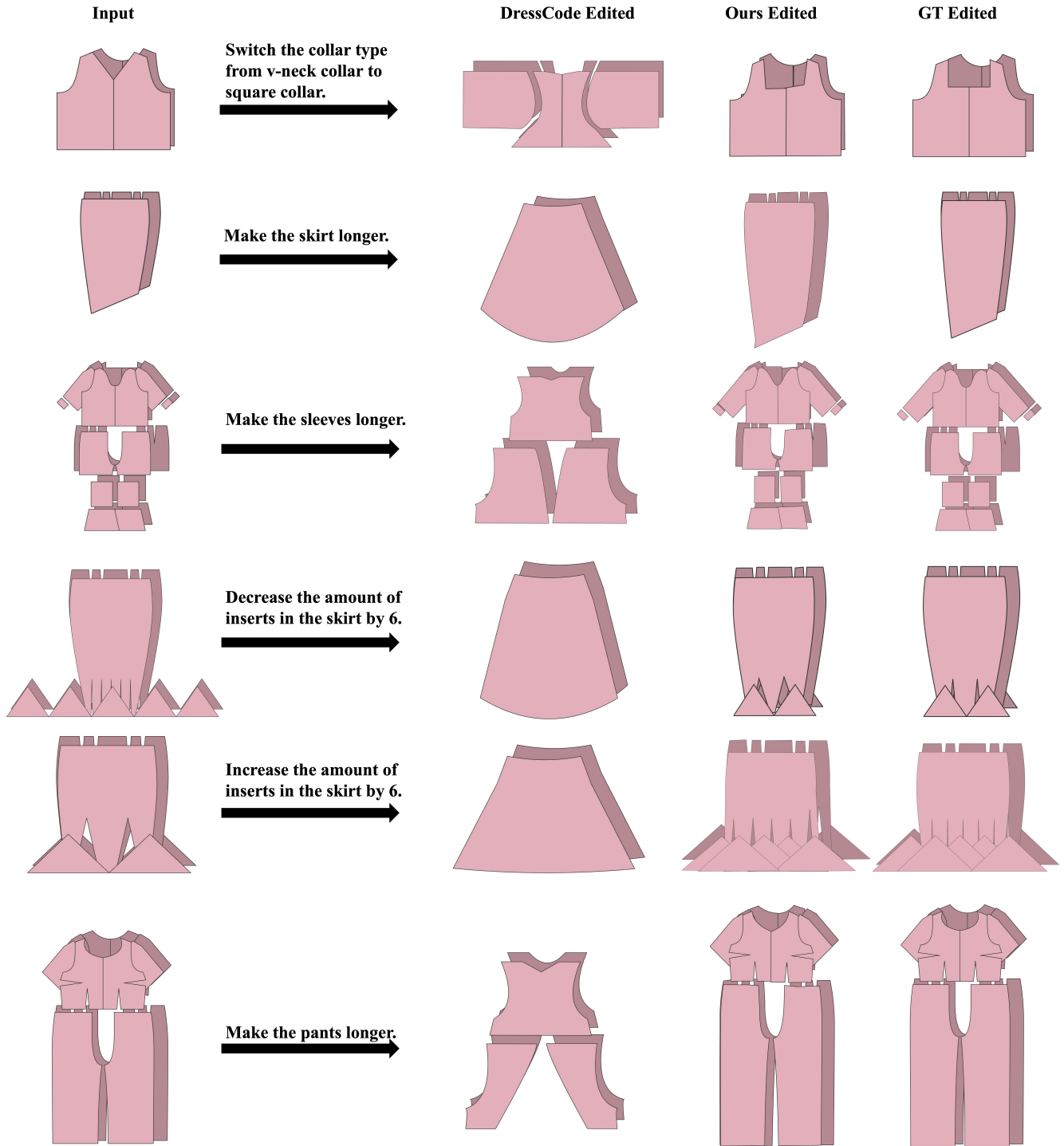


Figure S8. **Additional visualization for sewing pattern editing.** The task is to predict a sewing pattern that closely matches the input sewing pattern while following the editing instructions (text above the arrow). Notice that despite the diverse kinds of editing instructions we give, our methods can output sewing patterns that closely follow the instructions and the input sewing pattern. In the meanwhile, the baseline cannot achieve a similar effect because it takes only takes in text as input, losing structural details.

Method	Panel L2 ( $\downarrow$ )	#Panel Acc ( $\uparrow$ )	#Edge Acc ( $\uparrow$ )	Rot L2 ( $\downarrow$ )	Transl L2 ( $\downarrow$ )	#Stitch Acc ( $\uparrow$ )
LoRA	13.7	31.6	45.4	.020	5.1	.088
<b>AIpparel</b>	<b>5.4</b>	<b>85.2</b>	<b>82.7</b>	<b>.020</b>	<b>2.7</b>	<b>77.2</b>

Table S2. **Ablation Study: Fine-tuning Comparison.** The scores are reported on the image-to-garment prediction tasks on GCD-MM dataset. The metrics indicate that full model fine-tuning significantly outperforms LoRA fine-tuning, allowing the base model to better adapt to sewing pattern understanding.

Method	Panel L2 ( $\downarrow$ )	#Panel Acc ( $\uparrow$ )	#Edge Acc ( $\uparrow$ )	Rot L2 ( $\downarrow$ )	Transl L2 ( $\downarrow$ )	#Stitch Acc ( $\uparrow$ )
6 layers	5.93	83.6	81.0	.008	2.9	74.3
5 layers	6.10	84.2	80.7	0.010	2.8	73.4
4 layers	5.94	83.2	81.3	0.011	3.0	74.7
3 layers	5.92	83.7	80.9	0.010	2.9	73.7
<b>2 layers</b>	<b>5.4</b>	<b>85.2</b>	<b>82.7</b>	<b>.020</b>	<b>2.7</b>	<b>77.2</b>

Table S3. **Ablation Study: Number of Layers in Regression Heads.** The scores are reported on the image-to-garment prediction tasks on GCD-MM dataset.

You are given a list of attributes describing a garment. Your task is to modify the list according to an editing instruction provided.

To accomplish this: 1. If the attribute related to the instruction already exists in the description, locate and modify it to reflect the new information. 2. If the attribute is not present, add a new entry to the description that fulfills the instruction. 3. Ensure that no other attributes are altered unless necessary for consistency or clarity following the modification.

Once the changes are complete, return the list of attributes, without any additional information.

We evaluate this task using the test split of GCD-MM, containing approximately 6,000 editing samples.

**Additional Qualitative Visualization** Figure S8 shows additional visualization of the editing tasks as shown in Fig. 5 of the main paper. Notice that our model is able to correctly edit the sewing pattern with a diverse set of instructions.

### S3.5. Ablation Study

**Setup & Baseline Details.** Table 5 in the main paper shows an ablation study on our proposed tokenization scheme in Section 3.2 of the main paper. As described in Section 5.4, we use text-to-image as our ablation task to conduct an equal comparison of our model with DressCode [3]’s pre-trained model. Furthermore, we swap our tokenizer into DressCode’s model, to ensure an equal comparison. We also do the same for the configuration, *Ours w.o. reg.*, which uses the proposed sewing pattern tokenization scheme without the usage regression heads. We train both models from scratch with a learning rate of 0.0006 and a total batch size of 512 on 2×Quadro RTX 8000 GPUs, for a total of 30,400 steps until convergence.

**Additional Ablation Study.** Table S2 shows a qualitative comparison studying the effectiveness of full model fine-tuning versus LoRA [4] fine-tuning. The table reports reconstruction metrics on the image-to-garment prediction task on GCD-MM dataset. For the LoRA model, we use rank 8 and only fine-tune the query and key projection layers following previous works [2, 9]. The model is trained with the same hyperparameter settings

described in Section S2.2 for 8250 steps. The metrics indicate that the full fine-tuning model significantly outperforms the LoRA fine-tuned version, indicating that fine-tuning all weights in the language transformer is essential for understanding sewing patterns.

**Additional Qualitative Visualization.** Figure S9 shows additional visualizations for the ablation study in Section 4.4 of the main paper. Notice that our model in general demonstrates better sewing pattern prediction ability than DressCode. This can be seen in the pants prediction in the second and third rows of the figure, where DressCode does not predict the correct sewing pattern.

### S3.6. Draping Details

We use the draping pipeline provided by GarmentCode [7] for converting sewing patterns to a 3D mesh of the garment draped on a standard female SMPL model in A-pose. Specifically, the draping process consists of creating the boxed mesh and using Nvidia-Warp [13] for cloth simulation. To obtain the garment in arbitrary poses and in a motion sequence, we follow the simulation pipeline provided by PhysAvatar [17], which uses Codimensional Incremental Potential Contact (C-IPC) [10] simulation for cloth simulation. For simulation details, please refer to Zheng et al. [17]. Finally, the simulated mesh sequence is imported to Blender for texturing and rendering.

### S3.7. Human Study

We conducted a user study to compare sewing patterns generated from multimodal inputs using AIpparel with those using baselines. Specifically, we deployed 10 multiple-choice questions asking which garment better aligns with the input prompts while maintaining realism. The questions contain a combination of sewing patterns generated from images, texts, and editions of existing sewing patterns from different methods. We collected responses from 73 participants and Fig. S10 shows the favorability comparison for each modes of generation. AIpparel is more favorable in all modes, aligning with our quantitative and qualitative results.

## S4. Discussion

We expand our discussion in Section 6 of the main paper to include further limitations, future work, and social impact.

### S4.1. More Discussion on Limitations and Future Work

Due to computational resource constraints, we only train AIpparel on part of the GCD data, and AIpparel outputs a single modality, sewing pattern. As the community gets more computing resources, we are excited to see works extending our methods to larger datasets with richer annotations. It is an interesting direction to further scale up AIpparel to study the emergence of abilities like few-shot or in-context generalization to novel garment generation tasks or perform chain-of-thoughts to achieve a complex garment design. It is also an interesting direction to study how to further enlarge sewing pattern datasets with more variations and more annotations. For example, reflecting realistic variations of fabric properties can enable more accurate sewing pattern prediction.

**Bias and comprehensiveness of GCD-MM.** AIpparel can inherit the bias from the sewing pattern dataset used to create GCD-MM. In fact, GarmentCodeData [8] discusses such biases in its limitation section including only sewing patterns fitted to statistical models computed from a pool of healthy European and North American adults, hence limiting the size variations within the sewing patterns of GCD. However, we note that our data curation pipeline outlined in the paper can be used for other sources. By applying our pipeline to other, less biased, and more comprehensive sewing pattern datasets, we can still improve their quality by creating annotations for the sewing patterns.

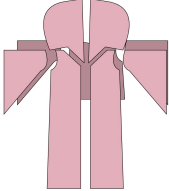
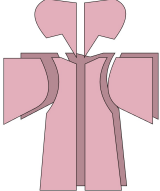
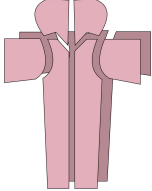
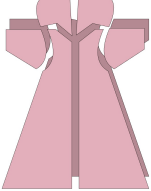
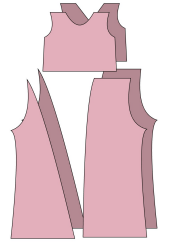
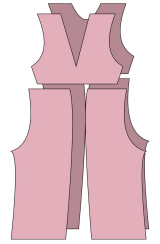
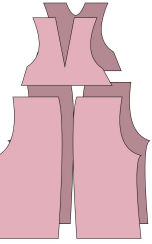
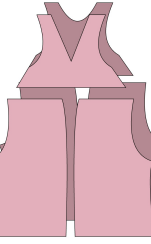
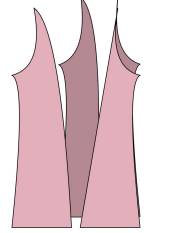
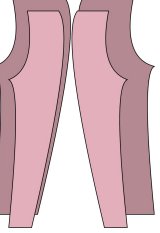
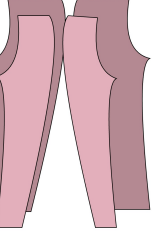
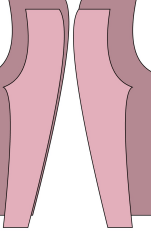
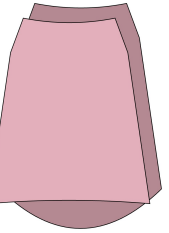
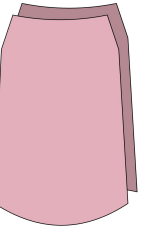
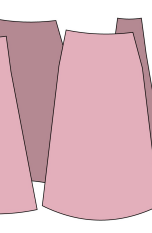
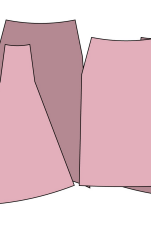
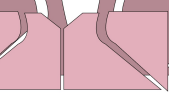



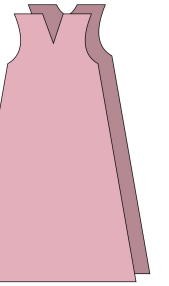
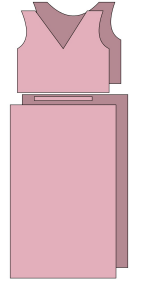
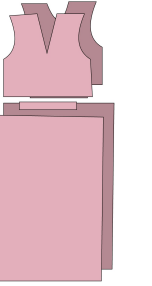
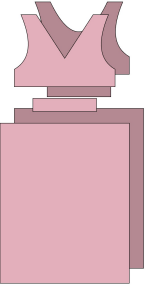
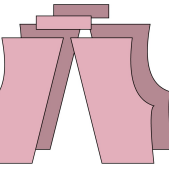
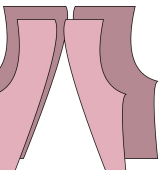
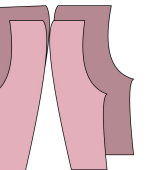
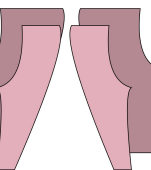
Input	DressCode	Ours w.o. Reg	Ours	GT
coat, with a hood, wide sleeves, long.				
jumpsuit, sleeveless, deep collar, long length, fitted waist, loose-fitting.				
trousers, long legs, high-waisted.				
skirt, normal waist, long length.				
Blouse, long sleeves, waist-length, loose-fitting, wide cuffs.				
sleeveless dress, sleeveless, deep collar, ankle-length, fitted waist.				
cycling shorts, no sleeves, tight garment, knee-length.				

Figure S9. Additional Visualizations for Ablation Study.

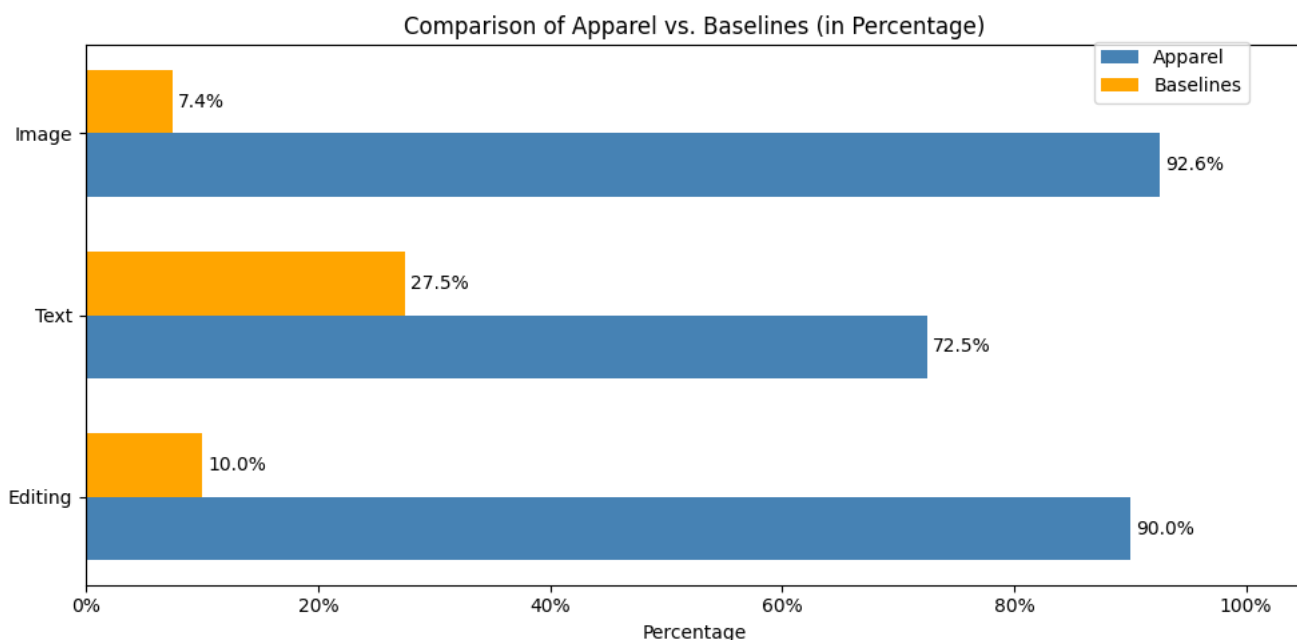


Figure S10. User Favorability Comparison of Apparel vs. Baselines for Multimodal Generation.

#### S4.2. Further Social Impacts

Besides the concerns of hallucination and bias that we inherit from our base model, LLaVA, we also acknowledge that our generated sewing patterns might not produce suitable garments for all communities, due to the limited body type and style selections within the data we trained on. It is important to study how to improve our method and dataset annotation on more diverse sewing patterns and body types in the future.

Another potential risk of our work is the potential bias we inherit from foundation models in our annotation generation process. Because we use large models such as GPT-4V for data generation, existing biases in these models will also be included in our generated annotations. However, because the prompts we used (see Section S1.1) encourage the model to generate descriptions based on the given images and keyword phrases, we did not find any immediate systematic bias present in our annotations.

## References

- [1] Tim Brooks, Aleksander Holynski, and Alexei A. Efros. Instructpix2pix: Learning to follow image editing instructions. In *CVPR*, 2023. 11
- [2] Yao Feng, Jing Lin, Sai Kumar Dwivedi, Yu Sun, Priyanka Patel, and Michael J. Black. Chatpose: Chatting about 3d human pose. In *CVPR*, 2024. 13
- [3] Kai He, Kaixin Yao, Qixuan Zhang, Jingyi Yu, Lingjie Liu, and Lan Xu. Dresscode: Autoregressively sewing and generating garments from text guidance, 2024. 1, 5, 7, 13
- [4] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. 13
- [5] Maria Korosteleva and Sung-Hee Lee. Generating datasets of 3d garments with sewing patterns. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021. 5
- [6] Maria Korosteleva and Sung-Hee Lee. Neurtailor: Reconstructing sewing pattern structures from 3d point clouds of garments. *ACM Trans. Graph.*, 41(4), 2022. 7
- [7] Maria Korosteleva and Olga Sorkine-Hornung. Garmentcode: Programming parametric sewing patterns. *ACM Transactions on Graphics (TOG)*, 42(6):1–15, 2023. 1, 4, 5, 14
- [8] Maria Korosteleva, Timur Levent Kesdogan, Fabian Kemper, Stephan Wenninger, Jasmin Koller, Yuhan Zhang, Mario Botsch, and Olga Sorkine-Hornung. Garmentcodedata: A dataset of 3d made-to-measure garments with sewing patterns, 2024. 5, 14
- [9] Xin Lai, Zhuotao Tian, Yukang Chen, Yanwei Li, Yuhui Yuan, Shu Liu, and Jiaya Jia. Lisa: Reasoning segmentation via large language model. *arXiv preprint arXiv:2308.00692*, 2023. 13
- [10] Minchen Li, Danny M. Kaufman, and Chenfanfu Jiang. Codimensional incremental potential contact. *ACM Trans. Graph. (SIGGRAPH)*, 40(4), 2021. 14
- [11] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023. 6
- [12] Lijuan Liu, Xiangyu Xu, Zhijie Lin, Jiabin Liang, and Shuicheng Yan. Towards garment sewing pattern reconstruction from a single image. *ACM Transactions on Graphics (SIGGRAPH Asia)*, 2023. 1, 5, 7
- [13] Miles Macklin. Warp: A high-performance python framework for gpu simulation and graphics. <https://github.com/nvidia/warp>, 2022. NVIDIA GPU Technology Conference (GTC). 14
- [14] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. 6
- [15] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, page 3505–3506, New York, NY, USA, 2020. Association for Computing Machinery. 7
- [16] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. 6
- [17] Yang Zheng, Qingqing Zhao, Guandao Yang, Wang Yifan, Donglai Xiang, Florian Dubost, Dmitry Lagun, Thabo Beeler, Federico Tombari, Leonidas Guibas, and Gordon Wetzstein. Physavatar: Learning the physics of dressed 3d avatars from visual observations. *arxiv*, 2024. 14