

# ChatGarment: Garment Estimation, Generation and Editing via Large Language Models

## Supplementary Material

In the supplementary material, we include additional details on training and evaluation, as well as ablation studies and qualitative visualizations.

### S1. More Implementation Details

#### S1.1. GarmentCodeRC

**Garment Sewing Pattern.** We improve the JSON-format sewing pattern configurations provided by GarmentCode [25, 26]. The original GarmentCode JSON configuration is a fixed-length file containing the same entries for all garments. We optimize this by adding new features, automatically removing irrelevant settings during garment construction (*e.g.*, omitting skirt-related parameters for upper-body garments) and normalizing floating-point values to  $[0, 1]$ . Here are two GarmentCodeRC JSON files for a skirt and a pair of pants:

```
1 {
2   "meta": {
3     "upper": "None",
4     "wb": "FittedWB",
5     "bottom": "PencilSkirt"
6   },
7   "waistband": {
8     "waist": 0.501,
9     "width": 0.205,
10    "height": 5
11  },
12  "pencil-skirt": {
13    "length": 0.365,
14    "rise": 0.988,
15    "flare": 0.577,
16    "low_angle": 5,
17    "front_slit": 0.010,
18    "back_slit": 0.009,
19    "left_slit": 0.001,
20    "right_slit": 0.001,
21    "style_side_cut": "Sun"
22  }
23 } % JSON for a pencil skirt
```

```
1 {
2   "meta": {
3     "upper": "None",
4     "wb": "None",
5     "bottom": "Pants"
6   },
7   "pants": {
8     "length": 0.203,
9     "width": 0.062,
10    "flare": 0.516,
11    "rise": 0.816,
12    "cuff": {"type": "None"}
13  }
14 } % JSON for a pair of pants
```

**Outfit Sewing Pattern.** From real images, we see that people often wear multiple garments, like a T-shirt and pants. In these cases, we combine them into a single outfit represented as a new JSON dictionary. If the subject wears one upper and one lower garment, the model will return in this format:

```
1 {
2   "upperbody_garment": {
3     (upper garment sewing pattern)
4   },
5   "lowerbody_garment": {
6     (lower garment sewing pattern)
7   }
8 }
```

Otherwise, if the subject wears a single whole-body garment (*e.g.*, dresses or jumpsuits), the model will return in the following format:

```
1 {
2   "wholebody_garment": {
3     (wholebody garment sewing pattern)
4   }
5 }
```

#### S1.2. ChatGarment Training Data

**Training Data Overview.** The training data consists of four parts: garment reconstruction data (35%), garment description data (15%), garment editing data (15%), and visual instruction tuning data (35%).

- **Garment Reconstruction Data:** Includes 20,000 simulated garments with images rendered by Blender and text labels generated by GPT-4o. During training, text labels and images are omitted from the input with a 25% probability respectively.
- **Garment Description Data:** Contains 38,000 SHHQ images with text descriptions generated by GPT-4o.
- **Garment Editing Data:** Comprises 20,000 garments generated following the rules in section 3.2.
- **Visual Instruction Tuning Data:** Utilizes LLaVA-v1.5-mix665k dataset<sup>1</sup>.

**Training Data Generation.** To create text descriptions for the garments in our training dataset, we render front and back images of the garments and then query GPT-4o to generate descriptions for the images. We use the prompts in Tab. S6 to generate descriptions for each garment part, and use the prompts in Tab. S7 to generate descriptions for all visible garment parts in the image. Additionally, we provide several examples from our dataset, including low-level and

<sup>1</sup>liuhaotian/LLaVA-Instruct-150K

high-level garment descriptions, as well as garment editing descriptions; see in Fig. S1.

As described in Sec. 3, we construct question and answer pairs to finetune a multimodal LLM. Specifically, we build two datasets: an image-reconstruction dataset and a sewing pattern editing dataset. Text-based generation data is derived by removing the images from the image-reconstruction dataset. Detailed question lists of these datasets are illustrated in Tabs. S1 to S4 respectively. Example textual answers are shown in Tab. S5, where [Sewing pattern without floats] refers to a JSON configuration where all float values are replaced with “0”. Since the projection layer is used to calculate numeric values in the sewing pattern, it is unnecessary to output these numeric values directly in the textual answers. Replacing them with “0” simplifies the training process.

- “<image> Can you estimate the outfit sewing pattern code in the image?”
- “<image> Please estimate the outfit sewing pattern code.”
- “<image> Provide the sewing pattern codes for the garments according to the image.”
- “<image> What is the sewing pattern codes for the outfit shown in the image?”
- “<image> Could you tell the outfit sewing pattern codes for the garments?”

Table S1. **Example questions for image reconstruction.** <image> is the placeholder token of the input image.

- “<image> Can you estimate the outfit sewing pattern code based on the image and the Json-format garment geometry description? [Garment descriptions]”
- “<image> Please estimate the outfit sewing pattern code based on the image and the garment geometry descriptions in Json format. [Garment descriptions]”
- “<image> Provide the sewing pattern codes for the garments according to the image and the Json-format garment geometry description. [Garment descriptions]”
- “<image> What is the sewing pattern codes for the outfit according to the image and the Json-format garment geometry description? [Garment descriptions]”
- “<image> Could you tell the outfit sewing pattern codes for the garments based on the image and the garment geometry descriptions in Json format? [Garment descriptions]”

Table S2. **Example questions for text-guided image reconstruction.** [Garment descriptions] refers to garment descriptions.

- “Can you estimate the outfit sewing pattern code based on the Json-format garment geometry description? [Garment descriptions]”
- “Please estimate the outfit sewing pattern code based on the garment geometry descriptions in Json format. [Garment descriptions]”
- “Provide the sewing pattern codes for the garments according to the image and the Json-format garment geometry description. [Garment descriptions]”
- “What is the sewing pattern codes for the outfit according to the Json-format garment geometry description? [Garment descriptions]”
- “Could you tell the outfit sewing pattern codes for the garments based on the garment geometry descriptions in Json format? [Garment descriptions]”

Table S3. **Example questions for text-based garment generation.** [Garment descriptions] refers to garment descriptions.

- “Adjust the old sewing pattern according to the text descriptions. The old garment sewing pattern is: [Old sewing pattern]. And the text descriptions are: [Text descriptions].”
- “Adjust the old sewing pattern [Old sewing pattern] according to the text descriptions [Text descriptions] without modifying other parts.”
- “Here is an old garment sewing pattern: [Old sewing pattern]. Modify the pattern to align with the text descriptions: [Text descriptions] without changing other parts.”
- “Update the old sewing pattern following the text descriptions: [Text descriptions]. The old garment sewing pattern is: [Old sewing pattern].”

Table S4. **Instructions for garment editing.** [Old sewing pattern] refers to the initial garment sewing pattern to be edited, and [Text descriptions] refers to the editing instructions.

- “[Sewing pattern without floats].”
- “It is [Sewing pattern without floats].”
- “Sure, it is [Sewing pattern without floats].”
- “The sewing pattern is [Sewing pattern without floats].”
- “The estimated sewing pattern is [Sewing pattern without floats].”

Table S5. **Example textual answers for reconstruction, generation, and editing.** [Sewing pattern without floats] refers to a simplified sewing pattern in which numeric values are replaced with “0”.

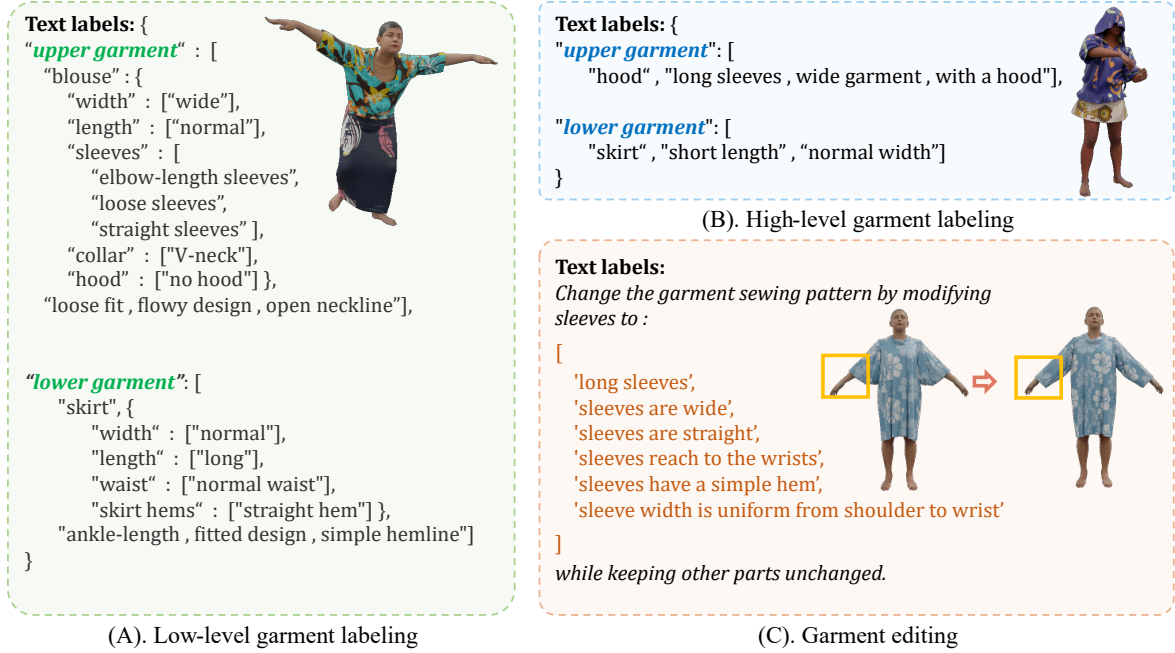


Figure S1. Dataset samples include low-level, high-level, and garment editing descriptions.

"I will provide an image of a human model wearing the [garment name]. The top two subfigures show the front and back views of the model (from left to right), while the bottom two subfigures show the zoomed-in view of the front and back views of the [garment name]. Please ONLY focus on the [part name] on the [garment name].

Please describe the geometries and structures of the [part name] on the [garment name] according to the image. Strictly avoid mentioning other garment parts. Strictly avoid mentioning color, texture, seams, and material.

Return a Json LIST of several phrases, each describing a geometric feature of the [part name], in the Json list format: [geometry feature 1, geometry feature 2, geometry feature 3, ...]"

Table S6. GPT-4o prompts for generating garment part labels in an image. [garment name] refers to the name of the garment and [part name] refers to the name of the garment part.

### S1.3. Training Details

We use LLaVA-1.5V-7B [34] as our base VLM, integrating CLIP for vision encoding and Llama 2 [55], fine-tuned on conversational data, as the LLM backbone. We freeze the vision encoder and projection layers while finetune the LLM using LoRA [18]. The sewing pattern projection layer is a two-layer (5120 x 76) MLP. The model is trained for 40 epochs, 500 steps per epoch, using the AdamW optimizer [22] with a learning rate of 1e-4. Training is done with a batch size of 4 per device on 4 NVIDIA H100 GPUs.

### S1.4. Inference Details

For image-based reconstruction, we apply the Chain-of-Thoughts (CoT) [58] for ChatGarment. Specifically, we first prompt ChatGarment to generate a detailed, JSON-format text description of the outfit in the image. The generated text descriptions are combined with the input image to estimate the final garment JSON configuration. Please see CoT prompts in Tab. S8.

### S1.5. Rule-based Simulation Control

ChatGarment demonstrates garment estimation and editing capabilities. The next step for artists is often simulating realistic garment movement. While existing tools need precise physical parameters to achieve desired deformations, we develop a rule-based method to derive material-specific parameters from text or images. This leverages LLM reasoning to map garment characteristics to simulation inputs.

We use C-IPC [29] as the simulator because of its strong capability in dealing with complex interactions between the human body and garments. C-IPC requires several physical parameters like density, stretching stiffness, bending stiffness, and thickness, which are specific to the simulator and not directly derived from real-world measurements. To bridge this gap, we propose a hierarchical mapping approach. This involves initially matching inferred material properties to predefined material classes, followed by parameter refinement within each class. For initialization, we prompt GPT-4o to identify the closest material match from a set

I will provide one image of a human model wearing several garments. Describe the outer layer garments the models are wearing. In each image, the model may wear one upper garment and one lower garment, or the model may wear a single wholebody garment. Avoid describing extra accessories such as the scarves, socks, watch, badges, and etc. We have known that the model wears [garment types].

For each garment, you should generate THREE strings.

In the first string, describe the garment type (If THE SUBJECT HAS NAME, INCLUDE ITS NAME FIRST!);

Example phrases for the first string: "hood", "T-shirt", "jacket", "tuxedo", etc.

In the second string, describe the structures of the garment (DO NOT INCLUDE ANY INFO ABOUT THE HUMAN MODEL AND THE COLOR OF THE GARMENT) in the format of a dict.

Select the keys from the following list: ['width', 'length', 'sleeves', 'pant legs', 'waist', 'skirt hems', 'skirt hems', 'collar', 'hood', 'waist', ... ]

In the value of the dict, please use several different short phrases in a list with the following tips:

Describe the width of the garment: wide, normal, narrow, etc.

Describe the length of the garment: long, normal, short, etc.

Describe the length and width of the sleeves: long, normal, short, tight, loose sleeveless, etc.

Describe the detailed struture of the sleeves. Example: "asymmetrical sleeves", "straight sleeves", "puff sleeves", "three-quater sleeves", "accordion sleeves", etc.

Describe the length and width of the legs of trousers: long, normal, short, tight, loose legs, etc.

Describe the detailed struture of the pant legs. Example: "asymmetrical legs", "straight legs", "flared legs", "cropped legs", "cuffed legs", etc.

Describe the length and width of the dress: long, normal, short, tight, loose, etc.

Describe the detailed struture of the skirt hems. Example: "straight hem", "A-line hem", "pleated hem", "pencil hem", "slit hem", etc.

Describe the detailed struture of the neck or collar. Example: "crew neck", "V-neck", "turtle neck", "collarless", etc.

Describe the detailed struture of the hood. Example: "normal hood", "cape hood", "cowl hood", etc.

An example of the dict description for a T-shirt is:

'width': ['wide'], 'length': ['normal'], 'sleeves': ['elbow-length sleeves', 'tight sleeves', 'accordion sleeves'], 'collar': ['crew neck'], 'hood': ['no hood']

An example of the dict description for a skirt is:

'width': ['wide'], 'length': ['knee-length'], 'waist': ['high waist'], 'skirt hems': ['pencil hem', 'pleated hem']

In the third string, describe the extra detailed structures of the garment (DO NOT INCLUDE ANY INFO ABOUT THE HUMAN MODEL AND THE COLOR OR PATTERN OF THE GARMENT) that are missing in the second string using several different short phrases split by ','. Example phrases for the third string: "pleated skirt", "high-waist", "zipper closure", "frayed hem", "mid-rise waist", etc.

Please strictly avoid mentioning color, texture, and material.

In the image, if the model is wearing one upper garment and one lower garment, return the results in the following format: "upper garment": [upper garment type, upper garment geometric features, extra features], "lower garment": [lower garment type, lower garment geometric features, extra features]. Otherwise, the model is wearing a single wholebody garment , return the results in the following format: "wholebody garment": [wholebody garment type, wholebody garment geometric features, extra features]. Only return the JSON dictionary in the above format with a length of 1 or 2."

Table S7. **GPT-4o prompts for generating labels for all visible garment parts in an image.** [garment types] refers to the types of the garments the model is wearing.

of predefined material classes (see Tab. S9). The physics parameters for the target material are initially set based on the matched material, after which we further refine specific parameters that significantly impact the simulation behavior.

Our analysis demonstrates that four primary parameters, *memBE* (stretching stiffness), *bendE* (bending stiffness), *density*, and *thickness*, show strong correlations with the high-level descriptors: *rigid/soft*, *heavy/light*, *wrinkle/smooth*, and *perceived thickness*. Moreover, LLM can effectively compare high-level material performance rather than directly

estimating precise parameter values. Based on these correlations, each physical parameter is decoupled and individually mapped to its respective descriptor. We then ask GPT-4o to assign scores ranging from 1 to 10 for these high-level descriptors. These scores are used to adjust the corresponding physical parameters based on the score differences between the target material and the initial matched material, as described by the following equations:

- “<image> Can you describe the garment outfits in in image in the Json format?”
- “<image> Can you estimate the outfit sewing pattern code based the image and the Json format garment geometry description? [Garment descriptions]”

Table S8. **GPT-4o CoT prompts for generating sewing patterns from images.** [Garment descriptions] refers to the textual descriptions of garments generated from the first question.

- “<image> Can you infer the garment material from the provided image input ? ”
- “<existing material list> Based on your inference, can you identify the material from the provided list that most closely matches the inferred physical properties ? ”

Table S9. **GPT-4o prompts for generating materials from images.** <existing material list> refers to the list of predefined materials. These prompts are used to initially infer the garment material from the existing material list and the provided image.

Methods	Dress4D		CLOSE	
	CD (↓)	F-Score (↑)	CD (↓)	F-Score (↑)
LLaVA-13B	3.73	<b>0.78</b>	<b>2.54</b>	0.784
LLaVA-7B	<b>3.06</b>	<b>0.78</b>	2.94	<b>0.790</b>

Table S10. **Ablation study: effect of multimodal LLM backbones.** Models utilizing LLaVA-7B and LLaVA-13B backbones demonstrate comparable performance on the two datasets.

$$\log \text{memb} = \alpha_m \Delta_{\text{soft}} \cdot \log \text{memb}_{\text{base}} \quad (2)$$

$$\log \text{bendE} = \alpha_b \Delta_{\text{light}} \cdot \log \text{bendE}_{\text{base}} \quad (3)$$

$$\text{density} = \alpha_d \Delta_{\text{smooth}} \cdot \text{density}_{\text{base}} \quad (4)$$

$$\text{thickness} = \alpha_t \Delta_{\text{thickness}} \cdot \text{thickness}_{\text{base}} \quad (5)$$

where  $\Delta_*$  denotes the score differences derived from the inferred descriptors, allowing for refined adjustments of each parameter based on the closest matched material.

## S2. Ablation Study Details

**Multimodal LLM backbones.** As shown in Tab. S10, the LLaVA-7B and LLaVA-13B models achieve comparable results. For efficiency, we use the LLaVA-7B model for the other experiments in our paper.

**Training Data.** To assess the impact of part-level garment description datasets, we train a model (ChatGarment\*) exclusively on general garment descriptions. For image-based reconstruction, we continue to use the Chain-of-Thoughts [58] approach, prompting the model with a text description of the given garment as the first step. As shown in Tab. S11,

Methods	Dress4D		CLOSE	
	CD (↓)	F-Score (↑)	CD (↓)	F-Score (↑)
ChatGarment *	4.04	<b>0.79</b>	4.06	0.76
ChatGarment	<b>3.06</b>	0.78	<b>2.94</b>	<b>0.79</b>

Table S11. **Ablation analysis of different training datasets.** ChatGarment \* is only trained on high-level garment description datasets and exhibits poorer image reconstruction performance.

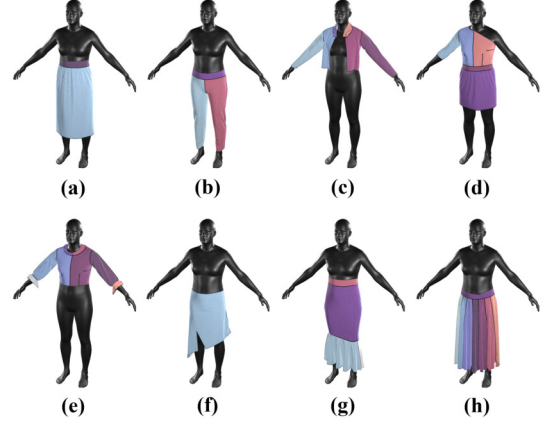


Figure S2. **Examples of GarmentCodeRC garments.** The collection includes a high-waisted skirt (a), fitted pant legs (b), an open-front jacket (c), and various complex designs of dresses, shirts and skirts (d-h).

the absence of part-level description datasets adversely affects image reconstruction results. In the Dress4D dataset [57], ChatGarment\* exhibits a worse Chamfer distance but a slightly higher F-Score. In the CLOSE dataset [2], ChatGarment\* performs worse on both metrics.

## S3. More Results

### S3.1. GarmentCodeRC

GarmentCode [25, 26] is an expressive DSL that can model complex garments with geometric details, including various cuts, frills, and pleats. Built upon GarmentCode, our proposed GarmentCodeRC further enhances support for open-front jackets, high-waisted skirts, and fitted pant legs. Examples of GarmentCodeRC garments are shown in Fig. S2.

### S3.2. Text-based Generation

We provide qualitative examples of text-based garment reconstruction results in Fig. S3, using the same prompt format as DressCode [16]. Compared to DressCode, ChatGarment accurately generates garments with correct lengths, widths, and detailed features. In contrast, DressCode occasionally produces incorrect garment types, inaccurate sizes, and missing details.



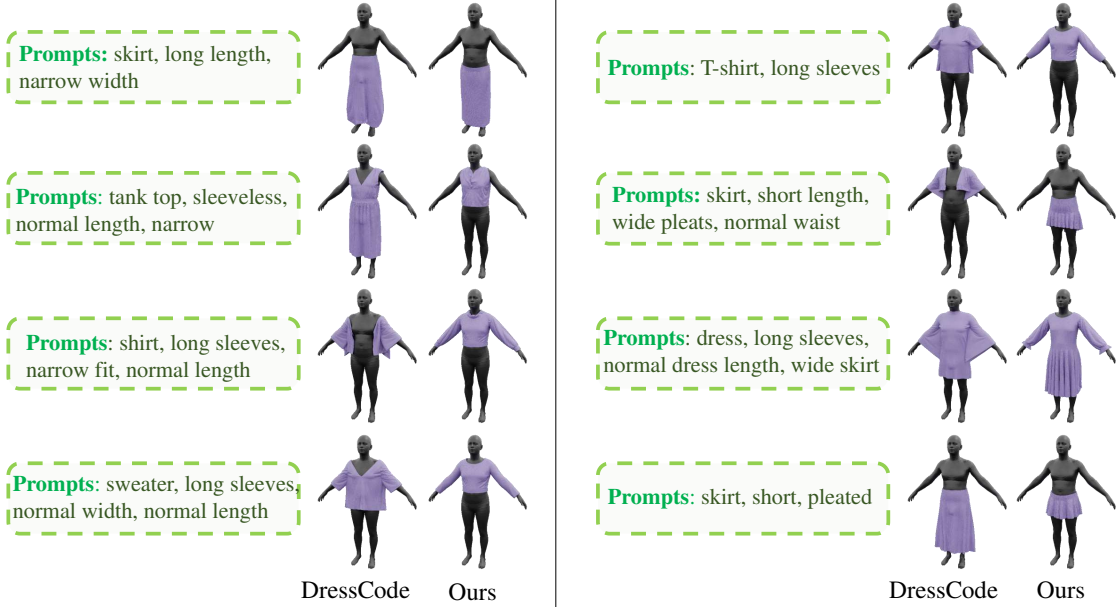


Figure S3. **Text-based generation results.** ChatGarment follows the instruction more accurately, generating more precise details (types, sleeves, length, etc.) compared to DressCode [16].

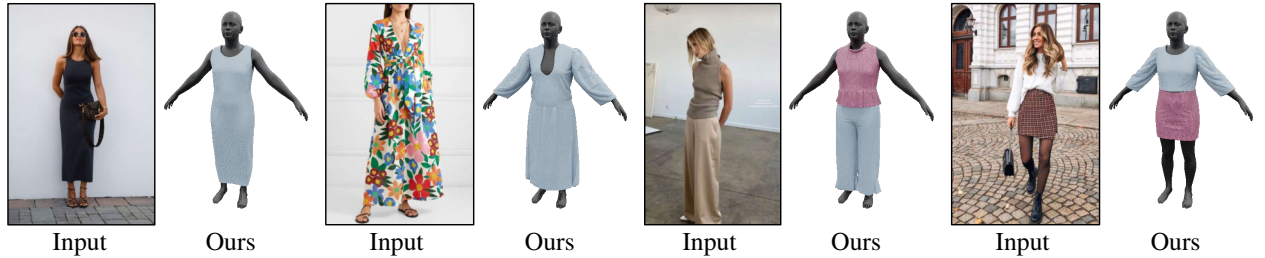


Figure S4. **Single-turn Image-based Garment Reconstruction.** ChatGarment generates valid garments directly from the input images.

### S3.3. Single-turn Image-based Reconstruction

In our experiment, we apply the Chain-of-Thought [58] method for optimized performance. However, ChatGarment also supports direct image-based reconstruction in a single-turn conversation. In this setup, ChatGarment is prompted to generate the garment JSON file directly from the input image. Qualitative examples are provided in Fig. S4.

### S3.4. Rule-based Simulation Control

We present qualitative examples of rule-based simulation control in Fig. S5. The simulation parameters are aligned with the material characteristics in the input image as described in Sec. S1.5. Leveraging the high-level descriptors in our rule-based approach, we can also modify the simulation behavior to make the garment deform like other materials. For instance, decreasing the stiffness (Stiffness↓) results in a softer garment with more pronounced wrinkles and larger deformations under the same motion. Conversely, increas-

ing the stiffness (Stiffness↑) produces a garment with rigid material properties, making it less prone to stretching.

### S3.5. Speed analysis of ChatGarment

We analyze garment reconstruction time on an A100 GPU. The process consists of three main stages: LLM decoding (12.1s), GarmentCode generation (3.5s), and sewing pattern stitching (33.9s). The primary bottleneck is the Warp-based sewing pattern stitching [25, 41] stage.

## S4. Failure Cases and Future Work

As shown in Fig. S6, ChatGarment occasionally struggles to edit specific garment parts without affecting other areas. For example, when adjusting the length of a skirt as requested, slight unintended changes may occur in the upper-body T-shirt. Additionally, in image-based garment reconstruction, it may fail to capture intricate details. While it can accurately identify the garment type as a skirt and estimate its

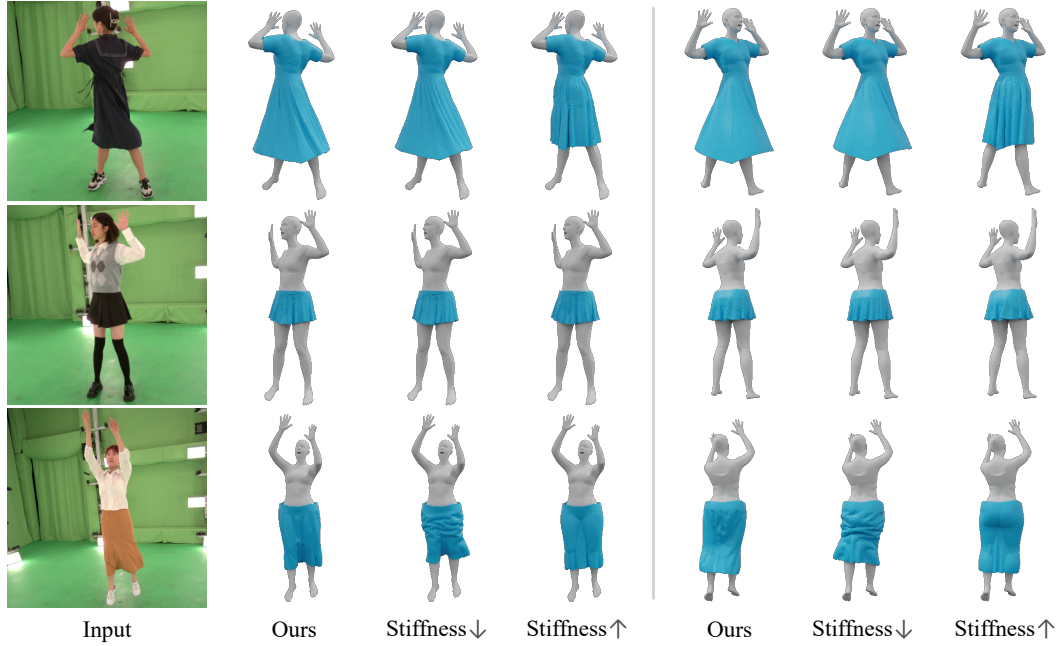


Figure S5. **Rule-based Simulation Control.** We apply our rule-based method to estimate the simulation parameters corresponding to the input images. This approach also allows control over different physical deformation behaviors, such as those of soft materials like silk (Stiffness↓) and rigid materials like denim (Stiffness↑).

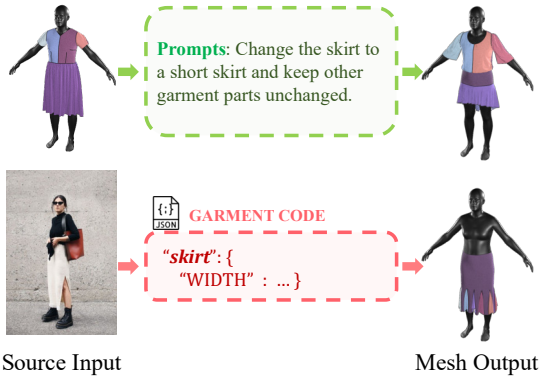


Figure S6. **Failure cases of sewing pattern editing and reconstruction.** ChatGarment might change the irrelevant garment parts (TOP: collars and sleeves). And ChatGarment occasionally misinterprets the garment details (BOTTOM: skirt style).

for sewing patterns, which could enhance both the diversity of generated garments and the precision of garment editing. And hallucinations could be reduced via Retrieval-augmented Generation (RAG), in-context Learning (ICL), or LLM post-training.

length, it may misinterpret finer details, such as mistaking the bottom style of the skirt for pleats. These inaccuracies can be attributed to LLM hallucinations. Additionally, although GarmentCodeRC can model complex garments with geometric details, including various cuts, frills, and pleats, as shown in Fig. S2, it cannot model some specific details such as zippers and pockets.

Future improvements to our method could involve developing a more advanced programming parametric model