Design2GarmentCode: Turning Design Concepts to Tangible Garments Through Program Synthesis

Feng Zhou[1,2*] **Ruiyang Liu**[2,4†] **Chen Liu**[2,4] **Gaofeng He**[2] **Yong-Lu Li**[3] **Xiaogang Jin**[4] **Huamin Wang**[2]

[1]Zhejiang Sci-Tech University [2]Style3D Research [3]Shanghai Jiao Tong University [4]State Key Lab of CAD&CG, Zhejiang University

Abstract

Sewing patterns, the essential blueprints for fabric cutting and tailoring, act as a crucial bridge between design concepts and producible garments. However, existing uni-modal sewing pattern generation models struggle to effectively encode complex design concepts with a multi-modal nature and correlate them with vectorized sewing patterns that possess precise geometric structures and intricate sewing relations. In this work, we propose a novel sewing pattern generation approach ***Design2GarmentCode*** *based on Large Multimodal Models (LMMs), to generate parametric pattern-making programs from multi-modal design concepts. LMM offers an intuitive interface for interpreting diverse design inputs, while pattern-making programs could serve as well-structured and semantically meaningful representations of sewing patterns, and act as a robust bridge connecting the cross-domain pattern-making knowledge embedded in LMMs with vectorized sewing patterns. Experimental results demonstrate that our method can flexibly handle various complex design expressions such as images, textual descriptions, designer sketches, or their combinations, and convert them into size-precise sewing patterns with correct stitches. Compared to previous methods, our approach significantly enhances training efficiency, generation quality, and authoring flexibility. Project page: https://style3d.github.io/design2garmentcode.*

1. Introduction

While generative AI has significantly propelled creativity in fashion design, turning those design ideas into wearable realities remains a formidable challenge. Sewing patterns are the key components to bridge the gap between abstract design ideas and wearable realities. They are foundational blueprints that dictate the precise shapes and dimensions of fabric pieces, essential for assembling garments in both the physical and virtual fashion realms.

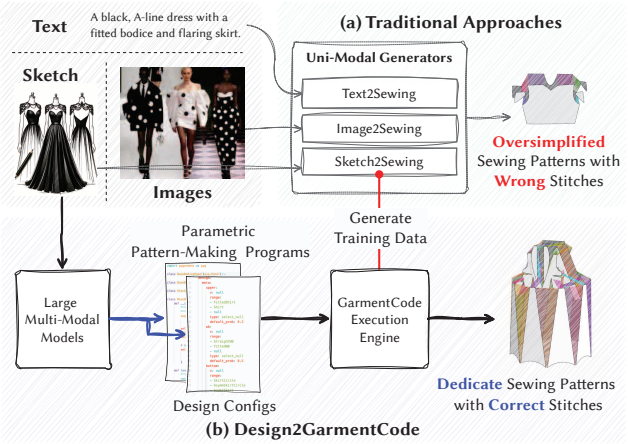


Figure 1. Traditional sewing pattern generation approaches (top) use *uni-modal* models trained on synthetic datasets generated by parametric pattern-making programs (red arrow) to convert text or image prompts into vector-quantized patterns. These methods are resource-intensive and often yield oversimplified patterns with stitching errors. Our approach (bottom) utilizes large pre-trained LMMs to *directly* translate design concepts into parametric programs and configuration files (blue arrow), enabling dedicated, structurally correct pattern generation from multi-modal design inputs within a unified framework.

Traditionally, sewing patterns are drafted manually by professional pattern-makers with years of practice, making the process inefficient, error-prone, and unable to meet the growing demands for refinement and personalization in the fashion market. To this end, parametric pattern-making researches [8, 29, 31] and industrial solutions [1–4] have emerged. These methods formalize the pattern-making process as geometric functions governed by parameters such as body measurements and design features, thereby accelerating the process by enabling pattern makers to generate sewing patterns through parameter adjustments instead of starting from scratch. However, creating these function templates is still complex, requiring not only advanced pattern-making skills but also geometric intuition, mathematical modeling knowledge, and coding abilities to trans-

*Works done during the internship at Style3D Research.
†Corresponding author.

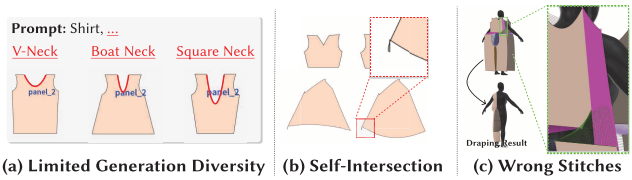


Figure 2. (a) Despite prompts specifying diverse neckline types, DressCode [22] consistently produces only V-neck designs, indicating limited generation diversity. (b) SewFormer [40] often generates sewing patterns with self-intersecting panels, compromising pattern validity. (c) Stitching errors are also prevalent in Sewformer [40], as shown here where a pant side seam is mistakenly stitched to a shirt shoulder seam, resulting in draping failure.

late pattern-making expertise into CAD programs. These technical barriers significantly restrict the widespread adoption of parametric pattern-making solutions.

Recently, several learning-based approaches for sewing pattern generation have been introduced. For instance, NeuralTailor [30] focuses on extracting sewing patterns from unstructured point clouds, DressCode [22] targets text-to-sewing pattern generation, and SewFormer [40] is designed for image-based sewing pattern generation. However, they are generally trained on paired design-sewing pattern data, necessitating large datasets to effectively capture the multimodal nature of design concepts. Furthermore, sewing patterns require centimeter-level precision to ensure proper garment fit, which presents a significant challenge for neural networks that only provide statistical approximations of the true function based on their training data [14, 23, 47]. As a result, these methods frequently generate oversimplified patterns with flawed geometry or stitches, potentially leading to draping failures (Figure 2).

In this paper, we present **Design2GarmentCode**, an innovative approach that leverages the generalization capabilities of vision-language foundation models to achieve multi-modal sewing pattern generation with *minimal computational and data requirements*. Unlike previous methods that directly synthesize vector-quantized patterns, Design2GarmentCode employs LMMs to learn the syntax of parametric pattern-making programs, translating design concepts into *parameters and programs* that can be executed to produce precise and structurally accurate sewing patterns. Design2GarmentCode combines a pre-trained Large Multimodal Model (LMM) as a design interpreter with a finetuned Large Language Model (LLM) as a program synthesizer. Specifically, the LLM is finetuned on code snippets from GarmentCode [31], a domain-specific language for constructing parametric sewing patterns. At runtime, the design interpreter extracts both topological and geometrical information from the design input by responding to a series of questions from the program synthesizer, which then generates garment programs and design configurations following GarmentCode syntax. Our method offers the following major contributions:

- We introduce a novel **modality-agnostic** framework with an intuitive, intelligent interface capable of processing user design intentions across multiple modalities simultaneously by integrating pre-trained LMMs.
- We present the **first** sewing pattern generation approach grounded in **program synthesis**, delivering *fully interpretable*, *geometrically precise*, and *structurally accurate* patterns through a more compact, semantically clear, and LLM-friendly representation.
- Our framework **benefits real-world production** by enabling flexible pattern authoring through natural language or physical feedback, allowing precise customization and efficient *creation of novel garment components*, represented as parametric pattern-making programs.
- Our approach requires only minimal fine-tuning of a pre-trained LLM and the training of a lightweight, text-conditioned transformer decoder, making it **more efficient** than existing vector-quantized sewing pattern generation models trained from scratch while offering *superior generation quality and authoring flexibility*.

2. Related Work

2.1. Garment Modeling with Sewing Patterns

Garment modeling and generation can be broadly classified into two categories: direct 3D garment generation (meshes) and sewing pattern generation, which are later draped onto human bodies via cloth simulation [39, 53, 56] or learning-based techniques [35, 37]. Direct 3D garment generation often relies on differentiable garment representations like unsigned distance fields [65, 67], shells [42], or Gaussian splatting [48]. However, it presents challenges in terms of both geometric accuracy and editability. On one hand, capturing fine garment details like folds and wrinkles necessitates extremely high-resolution 3D representations. On the other hand, editing these generated garments requires a well-defined UV space, and flattening the 3D mesh into developable meshes demands careful consideration from both geometric and statistical perspectives [7, 46].

Sewing pattern generation, by contrast, has been approached through both learning-based and procedural modeling methods. Learning approaches utilize vector-quantized representations of sewing patterns, mapping from unstructured 3D point clouds [7, 30], images [12, 26, 40, 64], or textual descriptions [22] to structured patterns. However, they are highly dependent on the quality and diversity of the training data and often struggle to generalize designs beyond the training domain. Furthermore, generating high-quality 3D garment data requires substantial domain knowledge and the involvement of skilled professionals.

Procedural modeling is an alternative that relies on pre-

defined rules and parameters to generate garment patterns. For instance, GarmentCode [31], a DSL for parametric pattern making, enables precise control over garment design and customization. The GarmentCodeData [32], built on GarmentCode, further illustrates the potential of procedural methods to generate a diverse range of made-to-measure garments, with adaptability to different body shapes. While procedural modeling provides greater control and precision, it typically requires specialized expertise and is less flexible when dealing with novel or unconventional designs.

2.2. LLMs for Program Synthesis

Recent advancements in program synthesis and code generation using large language models (LLMs) have laid essential groundwork for systems like Design2GarmentCode, which generate structured garment code from multi-modal design inputs. Earlier researches like Codex [11] and AlphaCode [36] demonstrated the effectiveness of LLMs in generating complex, task-specific code with high syntax accuracy, showcasing potential in scenarios requiring precise parametric coding. These models [43, 59] highlight how LLMs, when sufficiently trained, can transform natural language inputs into executable code, a capability directly relevant to generating garment codes that follow complex pattern-making syntax. [9, 19, 20, 55, 63] explore multimodal models that integrate visual aids, such as flowcharts and UML diagrams, into LLM training to enhance models' comprehension of complex structures and flow. These models particularly emphasize the need for semantic understanding and adaptability, which are critical in working with domain-specific languages (DSLs) like GarmentCode.

2.3. Neurosymbolic Models

Procedural/symbolic models and learned/neural models have complementary strengths and weaknesses. Neurosymbolic models [47] tend to combine the strengths of both paradigms and propose to generate visual data using symbolic programs augmented with AI/ML techniques. The neurosymbolic pipeline typically includes task specification, program synthesis using a DSL, program execution, and optional neural post-processing for refining results. It has been successfully applied across several areas of computer graphics. In 2D shape modeling, they are used in layout generation [45, 54], engineering sketch creation [16, 44, 49], and vector graphics synthesis by constructing programs that represent geometric shapes and their spatial relations [10, 17]. In 3D shape modeling, they facilitate inferring shape programs from existing 3D models [27, 28, 33, 51] or generating entirely new 3D shapes by training generative models on shape programs [57, 60, 61] or generate generate node graphs that define complex textures and materials [21, 25, 50] following the procedural modeling paradigm. Additionally, neuro-symbolic methods have been employed in human motion prediction [18, 38], reasoning [34, 58, 66] and generation [13, 41, 62], which leveraging visual-language foundation models to extract symbolic representations from visual data, facilitating complex activity reasoning by combining visual cues with symbolic logic [58]. Our method leverages a neurosymbolic approach by instruction-tuning LLMs to generate GarmentCode design configurations and component programs from diverse design concepts, ensuring geometric and structural accuracy.

3. Method

Our goal is to develop a generative model that transforms multi-modal design concepts into precise sewing patterns. This requires understanding diverse inputs and producing patterns with high geometric precision and intricate structures. These requirements present a challenge for conventional models, which require extensive training data and struggle with output precision due to their probabilistic nature. We propose **Design2GarmentCode**, a system leveraging LMMs to **generate parametric pattern-making programs**, or specifically GarmentCode [31]. Design2GarmentCode reduces the need for large datasets utilizing the pre-embedded pattern-making knowledge in LMMs while ensuring output precision with parametric program synthesis. In the following, we first provide an overview of parametric pattern-making programs and GarmentCode syntax, and then describe the detailed design of Design2GarmentCode.

3.1. Parametric Sewing Patterns

Parametric sewing patterns are formally represented as symbolic programs that generate sewing patterns (i.e., 2D CAD sketches) based on body measurements and design configurations. These symbolic programs enhance the efficiency of the pattern-making process by allowing users to draft or modify sewing patterns through semantically meaningful parameters. Mathematically, we can represent a sewing pattern $\mathcal{S}$ as:

$$\mathcal{S} = \langle \mathcal{F}, \mathcal{D}, B \rangle = \cup_{f_i \in \mathcal{F}, d_i \in \mathcal{D}} f_i(d_i, B), \tag{1}$$

where $\mathcal{F}$ is the set of symbolic programs, $\mathcal{D}$ represents design configurations, and B represents body measurements. Each symbolic function $f_i \in \mathcal{F}$ is essentially a series of rule-based 2D draw-calls controlled by its unique set of design configurations $d_i \in \mathcal{D}$ and the body measurements B.

GarmentCode is a domain-specific language (DSL) designed to generate parametric sewing patterns by encapsulating those symbolic programs in a hierarchical, component-oriented manner. In GarmentCode, each symbolic program f_i uses parametric curves to define a garment component, such as sleeves, bodices, or collars. The

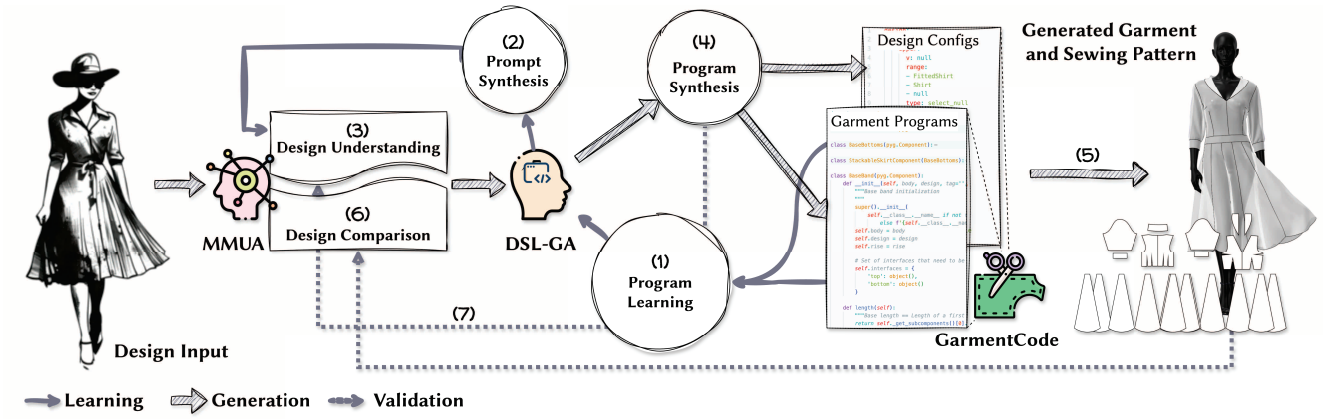


Figure 3. Overview of Dress2GarmentCode. (1) **Program Learning**: we finetune the *DSL Generation Agent (DSL-GA)* using GarmentCode example programs, teaching it the GarmentCode grammar and the semantics of each design parameter. (2) **Prompt Synthesis**: the DSL-GA generates prompts for the *Multi-Modal Understanding Agent (MMUA)* to interpret and extract relevant design features from the input (3). (4) **Program Synthesis**: based on the MMUA's responses, the DSL-GA synthesizes GarmentCode-compliant design configurations and garment programs, which are then executed by the GarmentCode engine to produce sewing patterns and simulated garments (5). To enhance robustness, we incorporate two validation loops: during program synthesis, we employ rule-based validations (7) to ensure the MMUA's outputs are sufficient for generating complete and valid garment programs and design parameters; after the initial generation, the MMUA compares the generated design with the input and suggests modifications to minimize discrepancies.

smallest component is a single panel, and multiple components can be combined through interface functions to create a larger component. In GarmentCode, a complete sewing pattern is specified by topological parameters $\mathcal{D}_T$ (which define the presence and quantity of garment components) and geometrical parameters $\mathcal{D}_G$, which determine the dimensions of each component when combined with body measurements B. As this work primarily focuses on design variations, we use a standard body model throughout all experiments to ensure consistency.

3.2. The Design2GarmentCode System

As illustrated in Figure 3, Design2GarmentCode has three components: **DSL Generation Agent (DSL-GA)**, a fine-tuned LLM responsible for (1) program learning, (2) prompt synthesis, and (4) program synthesis; **Multi-modal Understanding Agent (MMUA)**, a pre-trained LLM that manages design understanding (3) and design comparison (6); and (5) **GarmentCode**, which executes the synthesized programs to generate sewing patterns and 3D garments.

The system workflow begins with **Program Learning** (Sec. 3.2.1), where DSL-GA is finetuned to understand the syntax and semantic meanings of GarmentCode parameters. In **Prompt Synthesis**, DSL-GA creates prompts for MMUA to identify essential design features. These features are then provided to DSL-GA for **Program Synthesis**, where garment programs and design configurations are generated through rule-based parameter validation (Figure 3 (7)) and a learned projector (Sec. 3.2.2). The GarmentCode Execution Engine then produces sewing patterns and draped garment models. Finally, a **Validation** stage compares the generated garment with the original design, allowing MMUA to provide specific correction instructions to DSL-GA for iterative refinement, such as *"make the sleeve longer"*.

3.2.1 Program Learning

During experiments, we found that pre-trained LLMs have some foundational knowledge of pattern drafting. For example, when prompted with "How to draft a basic upper body bodice?", LLMs can produce drafting instructions that align with conventional practices. We use a pre-trained LLM to initialize DSL-GA, however, due to GarmentCode's customized object notations and function logistics, directly prompting DSL-GA to generate GarmentCode programs poses significant challenges [9].

To address these challenges, we propose to align DSL-GA's embedded pattern-making knowledge with the specific syntax and semantics of GarmentCode via LoRA [24] based on fine-tuning. We start by providing the DSL-GA (denoted as Γ) with existing GarmentCode programs $\mathcal{F}$, and instructing it to comment on the functions with detailed pattern-drafting instructions. After manually validating the comments, we get a dataset D paring natural language instructions with GarmentCode implementations:

$$D = \left\{ \left(\Gamma_{cmt}(f_i), f_i \right) \mid f_i \in \mathcal{F} \right\}, \tag{2}$$

where Γ_{cmt} is the instructed DSL-GA for code commenting, f_i is . Similar to [9], we finetune DSL-GA (Γ) on the

dataset D with LoRA [24], aiming for $\Gamma_{ft}(\Gamma_{cmt}(f_i)) \rightarrow f_i$, where Γ_{ft} is the finetuned DSL-GA.

After fine-tuning, the fine-tuned DSL-GA Γ_{ft} gains an understanding of the code structure and parameter semantics in GarmentCode (Figure 9). Therefore, we provide the design configuration $\mathcal{D}$ to Γ_{ft}, prompting it to analyze the semantic meaning of each parameter and generate structured queries. These queries are designed to guide the MMUA in extracting relevant design features from the multi-modal input, enabling Γ_{ft} to generate a comprehensive set of design parameters. The generated prompt P typically starts with analysis instructions, followed by multiple-choice or numerical estimation questions regarding each design parameter d_i. Formally, we have

$$P = \Gamma_{ft}(\mathcal{D}) = \cup_{d_i \in \mathcal{D}} \Gamma_{ft}(d_i) = \cup q_i, \tag{3}$$

where $q_i = \Gamma_{ft}(d_i)$ represents the generated question regarding the i-th design parameter d_i.

3.2.2 Program Synthesis

Initial results showed that MMUA performed significantly better on multiple-choice questions compared to numerical estimation questions. To improve accuracy, we replaced all numerical estimation questions in the initial prompt P with equivalent **multiple-choice questions** with descriptive options such as *"full length"*, *"half length"*, or *"three-quarter length"*. We append a lightweight **projector** Ψ after the finetuned DSL-GA Γ_{ft} to transform these descriptive answers τ_i regarding the design input x into precise geometrical parameters $d_i \in \mathcal{D}$ adhering to GarmentCode [31]:

$$\Psi : \Gamma_{ft}(\cup \tau_i) \rightarrow \mathcal{D}, \text{ where } \tau_i = \textbf{MMUA}(q_i, x). \tag{4}$$

Inspired by DressCode [22], we implement the projector Ψ as text-conditioned decoder-only transformer, where we design a **type-based quantization function Q** to convert the parameter list $\mathcal{D}$ into a token sequence $\mathcal{T} = \{t_1, ..., t_N\}$, where $N = |\mathcal{D}|$ denote the total number of design parameters. The quantization function $\mathbf{Q}$ operates as follows:

$$t_i = \mathbf{Q}(d_i) = \begin{cases} 0/1, & \text{if } d_i \text{ is a boolean variable,} \\ d_i, & \text{if } d_i \text{ is an integer,} \\ \lambda \cdot \textbf{Norm}(d_i), & \text{if } d_i \text{ is a floating number,} \\ \textbf{Index}(d_i, L), & \text{if } d_i \text{ is a selective variable.} \end{cases} \tag{5}$$

where λ is a scaling factor indicating numerical precision. We use $\lambda = 100$ to maintain centimeter-level precision.

As in Eq. 4, we use the finetuned **DSL-GA** Γ_{ft} to encode the answers τ_i from **MMUA** and construct the condition input for Ψ (we use an MLP to match the embedding dimension between $3,072$ in Γ_{ft} and 128 in Ψ). Notably, our token sequence length is fixed to the number of design parameters $|\mathcal{D}| = 122$, regardless of the complexity of the pattern. This fixed-length representation is at least $10\times$ compact than DressCode [22], whose sequence length is $1,500$ and scales with pattern complexity (see Sec. 8 for implementation details).

Method		Text Guided Generation		Image Guided Generation	
		DressCode [22]	Ours	Sewformer [40]	Ours
Quality	SSR	84%	100%	65.33%	94%
	Agreement	7.17%	79.83%	3.33%	88.67%
	Aesthetic	9.50%	68.17%	5.33%	77%
	CLIPScore	0.2386	0.2438	/	/
	F-Score	0.616	/	0.708	0.829
	CD	15.77	/	9.7	2.091
Diversity	# Panels	$5.11_{\pm 1.66}$	$6.92_{\pm 3.10}$	$10.11_{\pm \mathbf{4.42}}$	$\mathbf{11.02}_{\pm 4.18}$
	# Edges	$5.48_{\pm 1.60}$	$6.84_{\pm 3.38}$	$5.79_{\pm 1.71}$	$6.24_{\pm 2.90}$
	# Stitches	$10.06_{\pm 3.24}$	$18.66_{\pm 8.64}$	$15.81_{\pm 5.91}$	$27.9_{\pm 9.83}$

Table 1. Quantitative comparison of our method against state-of-the-art (SOTA) sewing pattern generation techniques in terms of quality and diversity. *SSR* (Simulation Success Rate) indicates the feasibility of simulated garment assembly, while *Agreement* measures alignment with design prompts, and *Aesthetic* evaluates the visual preference of the generated patterns. *CLIPScore* assesses text-image consistency, whereas *Chamfer Distance (CD)* and *F-Score* quantify geometric accuracy. *#Panels*, *#Stitches*, and *#Edges* denote the mean and standard deviation (subscript) of the number of panels, stitches per pattern, and edges per panel.

4. Experiments

4.1. Quantitative Evaluation

We evaluate our proposed method against state-of-the-art sewing pattern generation approaches (DressCode [22] for text-guided and Sewformer [40] for image-guided generation) on **Generation Quality** and **Generation Diversity**.

Generation Quality is evaluated through three metrics: *Simulation Success Rate (SSR)*, *Agreement Score*, and *Aesthetic Score*. The *Simulation Success Rate (SSR)* is calculated as the ratio of successfully simulated garments to the total number of generated sewing patterns, measuring the structural feasibility of the patterns. We prepared a dataset comprising 150 text prompts and 150 test images, covering a wide variety including *tops (78), pants (76), skirts (38), dresses (80), and suits (28)*. For each sample, we generated sewing patterns using both our method and baseline methods, and simulated the patterns using GarmentCode's simulation engine [31, 32] to compute the success rate. The *Agreement and Aesthetic Scores* were derived from a user study involving 30 professional pattern-makers. Each participant is asked to review 50 text and 50 image samples generated by our and the baseline models, and assess their preference based on sewing patterns and simulated garments according to:

- *Agreement*: the degree to which the generated pattern matched the design prompt.
- *Aesthetic Quality*: the visual appeal and structural coher-

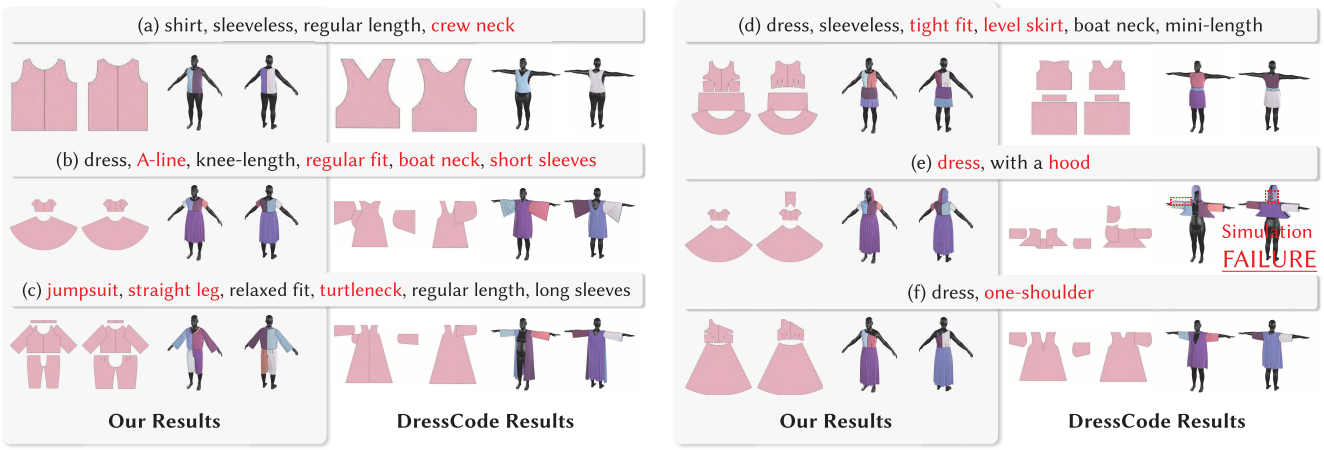


Figure 4. Quality Comparison on Text-Guided Sewing Pattern Generation. For each design, we present the generated pattern using our method (left) alongside DressCode [22] (right), including front and back renderings of the draped garment. We highlight design elements accurately captured by our method but missed by DressCode [22] use red color in the input prompt.

ence of the generated pattern.

For each criterion, participants could express a preference for either our method or the baseline or indicate that both methods were *"comparable"*. We calculate the Agreement and Aesthetic Scores as the percentage of times each option was chosen over the total number of tested samples.

Table 1 presents the results, showing that our method surpasses existing approaches in both SSR and user-evaluated Agreement and Aesthetic scores. For text-guided generation, our model achieves a perfect 100% SSR, notably higher than DressCode's 84%. Additionally, our Agreement score of 79.83% and Aesthetic score of 68.17% far exceed DressCode's respective scores of 7.17% and 9.5%. In image-guided generation, our method attains a 94% SSR, with an Agreement score of 88.67% and an Aesthetic score of 77%, significantly outperforming Sewformer. These enhancements highlight our model's ability to generate sewing patterns that are both structurally precise and visually aligned with the design prompt.

Generation Diversity is evaluated by analyzing the average number of panels (# Panels), edges (# Edges), and stitches (# Stitches) in the generated patterns. For text-guided generation, our method yields more intricate designs, with an average of 6.92 panels, 6.84 edges, and 18.66 stitches per pattern, compared to DressCode's simpler outputs of 5.11 panels, 5.48 edges, and 10.06 stitches. In image-guided generation, our approach also demonstrates superior diversity, producing an average of 11.02 panels, 6.24 edges, and 27.9 stitches per pattern, compared to Sewformer's averages of 10.11 panels, 5.79 edges, and 15.81 stitches. These results emphasize our model's ability to capture and replicate subtle design variations, highlighting its robustness and adaptability across different design inputs.

4.2. Multi-modal Generation Results

Our proposed method demonstrates superior performance across various sewing pattern generation tasks, including text-guided, image-guided, and sketch-based generation.

In text-guided sewing pattern generation (Figure 4), our method accurately captures design details specified in prompts, such as neckline types (*e.g.*, crew neck (a), boat neck (b), turtleneck (c)) and complex structural features like asymmetry (f) and layered skirts (d). In comparison, the baseline model DressCode struggles with limited pattern diversity, often defaulting to simpler shapes like V-neck designs. Additionally, for design descriptions out of its training domain, DressCode frequently generates patterns with incorrect stitching, leading to poor draping results (Figure 4 (e)). Our method could provide structurally sound and visually accurate patterns under a large design variety, showcasing its capability to handle diverse design requests with high fidelity.

For image-guided sewing pattern generation (Figure 5), our model effectively translates detailed visual cues from input images into corresponding sewing patterns. Compared with Sewformer, which often fails to model-specific design elements like cuffs, hoods, and asymmetric features, our approach accurately reproduces these details. Sewformer's results frequently exhibit structural flaws, such as missing or misaligned pattern pieces and extraneous components, resulting in unrealistic garment draping. In contrast, our method maintains structural integrity and captures complex design features, producing patterns that closely align with the source images.

In sketch-based sewing pattern generation (Figure 6), our system seamlessly converts both technical sketches (left)

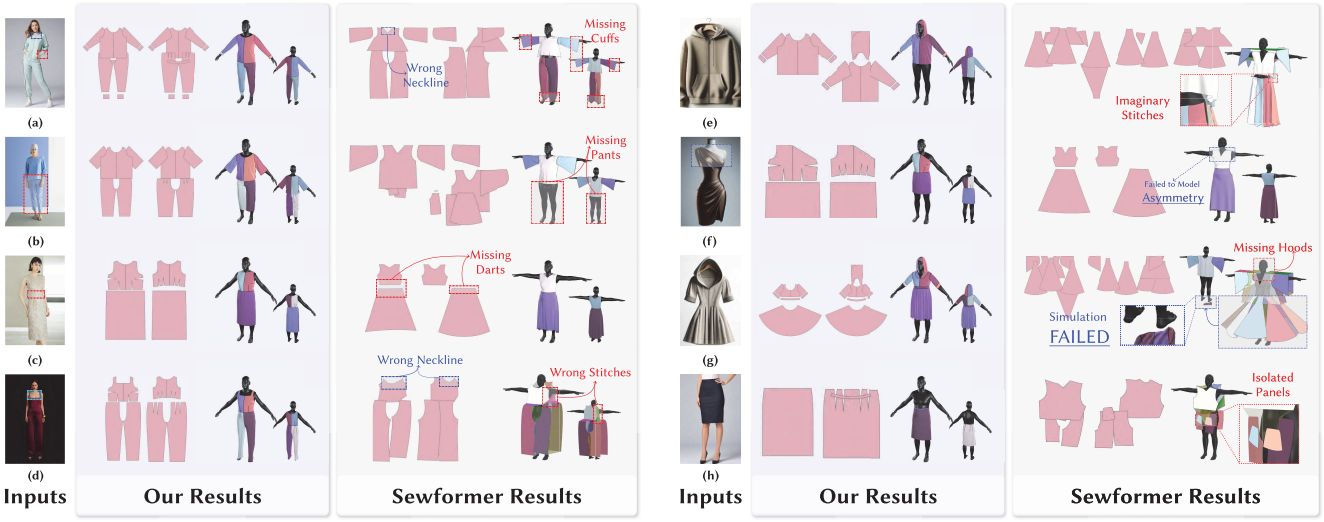


Figure 5. Quality Comparison on Image-Guided Sewing Pattern Generation. We compare our method with Sewformer [40] on Internet-collected fashion photographs (left), and AI-generated design images without human models (right). The results indicate that our method successfully captures design details from diverse styles, producing sewing patterns that accurately reflect neckline (a, d), cuffs (a, e, g), darts (c, d), and asymmetry (f). In contrast, Sewformer's results exhibit several issues, including incorrect necklines (a, d), missing components (b, g), misplaced or imaginary stitches (d, e), and extraneous pattern pieces (h). Additionally, since Sewformer's pattern generation does not account for body shape, garments like skirts and pants frequently appear oversized around the waist, causing them to sag when draped.

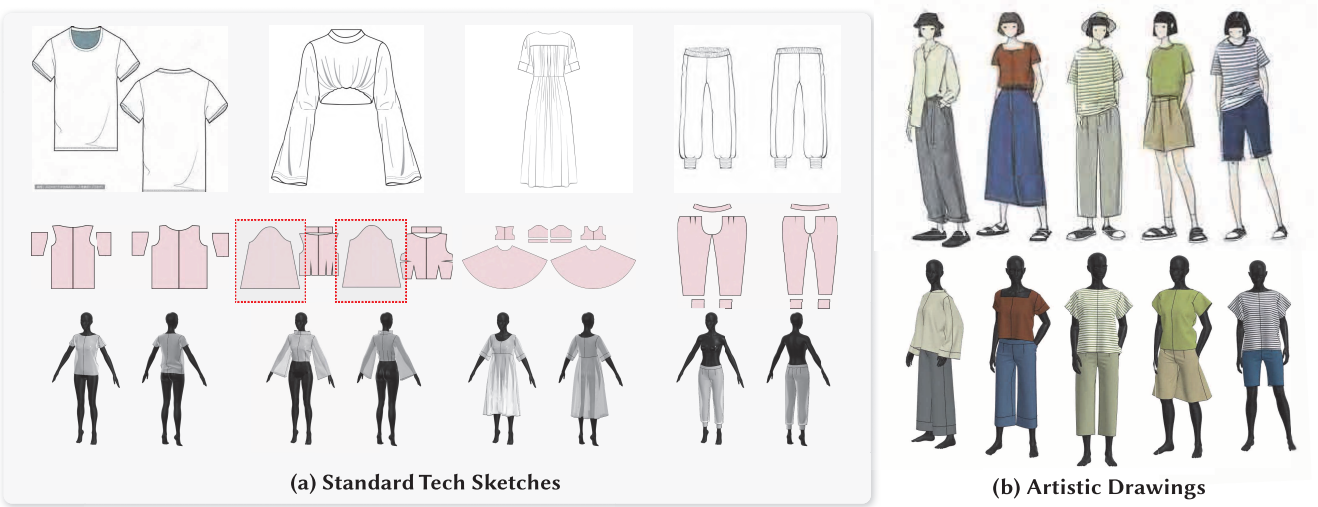


Figure 6. Examples of sketch-based sewing pattern generation. Our method was able to generate high-quality sewing patterns from design sketches under various styles and could integrate seamlessly with industrial fashion design software for (a) pattern editing, i.e. sleeve panels in red boxes are merged from separate front/back sleeve panels; and (b) avatar posture and fabric material editing.

and artistic drawings[1] (right) into high-quality sewing patterns. We also demonstrated that the generated sewing patterns could seamlessly integrate into industrial fashion design software[2]. For example, highlighted sleeve panels in Figure 6 (a) are merged from separate front and back sleeve pieces, while Figure 6 (b) demonstrates avatar posture and fabric material editing.

5. Application

In this section, we explore practical applications enabled by our system that extend beyond basic pattern generation, providing designers with versatile tools for design refinement, integration with physical simulation, and the creation of new garment components.

[1]The drawing is borrowed from the artwork of TWELVEYIN.
[2]We use Style3D Studio [5] for pattern and appearance authoring.

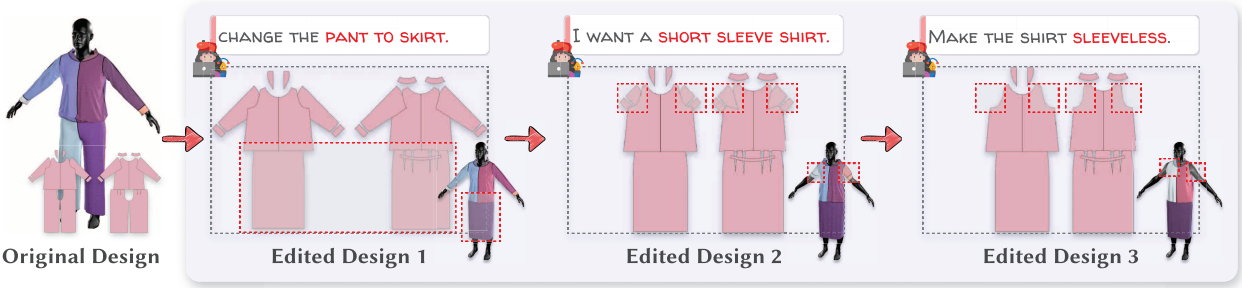


Figure 7. Sewing Pattern Authoring with instructions. Starting from an original design, the system follows user instructions to adjust specific pattern elements. In Edited Design 1, the pants are modified to a skirt based on the command "CHANGE THE PANT TO SKIRT". In Edited Design 2, the sleeves are shortened as requested. Finally, in Edited Design 3, the shirt is made sleeveless in response to the instructions. Note that, each modification accurately applies only to the specified parts, leaving the rest of the design unchanged.

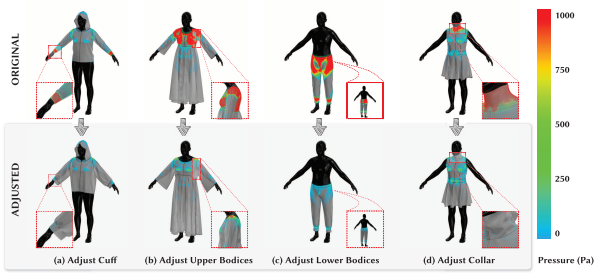


Figure 8. Sewing pattern adjustment based on body pressure measurement. Red regions indicate areas of tight fabric with high body pressure, while blue regions represent looser areas.

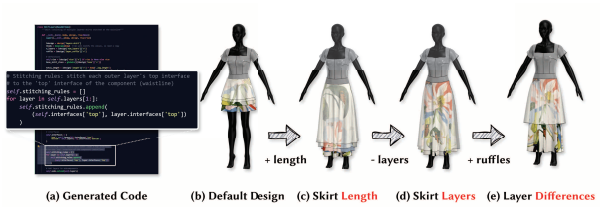


Figure 9. The code for a layered-skirt component generated by our DSL-GA and 3D garment under various design parameters.

Instruction-Based Editing. Our system allows designers to adjust generated sewing patterns through simple, instruction-based edits, utilizing the same refinement process as in our collaborative framework. As illustrated in Figure 7, starting from an original design, the system responds to natural language commands from the user to adjust the sewing pattern. At each step, the modified areas are highlighted in red boxes. From the figure, it is evident that our system can accurately update only the specified parts of the pattern according to the user's instructions while leaving all other parts of the design unchanged.

Physics-Based Editing. Our system's generated sewing patterns integrate seamlessly with professional cloth simulation software, allowing adjustments based on fitness measurements derived from physical simulations. In Figure 8, we demonstrate sewing pattern editing guided by body pressure analysis, including adjustments to the cuff (a), upper bodice (b), lower bodice (c), and collar (d). As shown in the examples, our system accurately identifies areas with excessive tension and adjusts the corresponding sewing patterns to enhance comfort while preserving the overall design.

Generating New Garment Programs. A major challenge in traditional parametric pattern-making is the need to abstract symbolic programs for new sewing patterns, which demands both advanced programming skills and pattern-making expertise. Design2GarmentCode addresses this by correlating GarmentCode grammar with LMMs' embedded pattern-making knowledge, enabling the automatic creation of new garment components. Figure 9 shows a layered-skirt component generated by our DSL-GA, along with 3D garment representations demonstrating different design parameters, such as skirt length (c), number of layers (d), and layer differences such as length difference and ruffling factor (e). The results demonstrate that our system consistently produces high-quality garment components that meet professional standards, while significantly reducing the time and expertise required to create new sewing pattern programs.

6. Conclusion

Design2GarmentCode transforms multi-modal design concepts into precise sewing patterns using LMMs to synthesize parametric programs. It addresses challenges related to data requirements, computation, and the limited precision of neural network-based methods. The experimental results demonstrate the system's ability to capture design details while maintaining structural integrity and geometric precision in generated patterns. Despite these advantages, Design2GarmentCode currently cannot substantially alter GarmentCode's underlying structure and logistics, which impacts generation quality due to inherent limitations in GarmentCode's design and modeling capabilities (Supp. 10).

Acknowledgments

This work was supported by Key R&D Program of Hangzhou (No.2024SDZ1A20). We sincerely thank Fuqi Wang for his valuable support in validation data collection, material modeling and animation.

References

[1] Assyst smart.pattern macro. https://www.assyst.de/en/products/cad/index.html. Accessed: 2024-11-09. 1

[2] Modaris: Smart and speedy patternmaking for fashion experts. https://www.lectra.com/en/fashion/products/modaris. Accessed: 2024-11-09.

[3] Optitex: 2d/3d cad pattern design software. https://optitex.com/products/2d-and-3d-cad-software/. Accessed: 2024-11-09.

[4] Valentina: Open source pattern drafting software. https://valentinaproject.bitbucket.io/index.html. Accessed: 2024-11-09. 1

[5] Style3D Studio. https://www.linctex.com, 2024. 7

[6] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 1

[7] Seungbae Bang, Maria Korosteleva, and Sung-Hee Lee. Estimating garment patterns from static scan data. *Computer Graphics Forum*, 40(6):273–287, 2021. 2

[8] Chen Bao, Yongwei Miao, Bingfei Gu, Kaixuan Liu, and Zhen Liu. 3d interactive garment parametric pattern-making and linkage editing based on constrained contour lines. *International Journal of Clothing Science and Technology*, 33 (5):696–723, 2021. 1

[9] Nastaran Bassamzadeh and Chhaya Methani. A comparative study of dsl code generation: Fine-tuning vs. optimized retrieval augmentation. *arXiv preprint arXiv:2407.02742*, 2024. 3, 4

[10] Alexandre Carlier, Martin Danelljan, Alexandre Alahi, and Radu Timofte. Deepsvg: A hierarchical generative network for vector graphics animation. *Advances in Neural Information Processing Systems*, 33:16351–16361, 2020. 3

[11] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. 3

[12] Xipeng Chen, Guangrun Wang, Dizhong Zhu, Xiaodan Liang, Philip Torr, and Liang Lin. Structure-preserving 3d garment modeling with neural sewing machines. *Advances in Neural Information Processing Systems*, 35:15147–15159, 2022. 2

[13] Xin Chen, Biao Jiang, Wen Liu, Zilong Huang, Bin Fu, Tao Chen, and Gang Yu. Executing your commands via motion diffusion in latent space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18000–18010, 2023. 3

[14] Keyu Duan, Zirui Liu, Peihao Wang, Wenqing Zheng, Kaixiong Zhou, Tianlong Chen, Xia Hu, and Zhangyang Wang. A comprehensive study on large-scale graph training: Benchmarking and rethinking. *Advances in Neural Information Processing Systems*, 35:5376–5389, 2022. 2

[15] Sai Kumar Dwivedi, Yu Sun, Priyanka Patel, Yao Feng, and Michael J. Black. TokenHMR: Advancing human mesh recovery with a tokenized pose representation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 1

[16] Sutherland Ivan Edward. Sketchpad: A man-machine graphical communication system. *PhD Thesis, Massachusetts Institute of Technology*, 1963. 3

[17] Kevin Ellis, Catherine Wong, Maxwell Nye, Mathias Sablé-Meyer, Lucas Morales, Luke Hewitt, Luc Cary, Armando Solar-Lezama, and Joshua B Tenenbaum. Dreamcoder: Bootstrapping inductive program synthesis with wake-sleep library learning. In *Proceedings of the 42nd ACM Sigplan International Conference on Programming Language Design and Implementation*, pages 835–850, 2021. 3

[18] Yao Feng, Jing Lin, Sai Kumar Dwivedi, Yu Sun, Priyanka Patel, and Michael J Black. Chatpose: Chatting about 3d human pose. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2093–2103, 2024. 3

[19] Daniel Fried et al. Incoder: A generative model for code infilling and synthesis. *arXiv preprint arXiv:2204.05999*, 2022. 3

[20] Rinon Gal, Adi Haviv, Yuval Alaluf, Amit H Bermano, Daniel Cohen-Or, and Gal Chechik. Comfygen: Prompt-adaptive workflows for text-to-image generation. *arXiv preprint arXiv:2410.01731*, 2024. 3

[21] Paul Guerrero, Miloš Hašan, Kalyan Sunkavalli, Radomír Měch, Tamy Boubekeur, and Niloy J Mitra. Matformer: A generative model for procedural materials. *arXiv preprint arXiv:2207.01044*, 2022. 3

[22] Kai He, Kaixin Yao, Qixuan Zhang, Jingyi Yu, Lingjie Liu, and Lan Xu. Dresscode: Autoregressively sewing and generating garments from text guidance. *ACM Transactions on Graphics (TOG)*, 43(4):1–13, 2024. 2, 5, 6

[23] Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B Brown, Prafulla Dhariwal, Scott Gray, et al. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*, 2020. 2

[24] Edward J. Hu, Yelong Shen, et al. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. 4, 5

[25] Yiwei Hu, Chengan He, Valentin Deschaintre, Julie Dorsey, and Holly Rushmeier. An inverse procedural modeling pipeline for svbrdf maps. *ACM Transactions on Graphics (TOG)*, 41(2):1–17, 2022. 3

[26] Moon-Hwan Jeong, Dong-Hoon Han, and Hyeong-Seok Ko. Garment capture from a photograph. *Computer Animation and Virtual Worlds*, 26(3-4):291–300, 2015. 2

[27] R Kenny Jones, Homer Walke, and Daniel Ritchie. Plad: Learning to infer shape programs with pseudo-labels and approximate distributions. In *Proceedings of the IEEE/CVF*

Conference on Computer Vision and Pattern Recognition, pages 9871–9880, 2022. 3

[28] R Kenny Jones, Paul Guerrero, Niloy J Mitra, and Daniel Ritchie. Shapecoder: Discovering abstractions for visual programs from unstructured primitives. *ACM Transactions on Graphics (TOG)*, 42(4):1–17, 2023. 3

[29] Yeonghoon Kang, Jihyun Oh, and Sungmin Kim. Development of parametric garment pattern design system. *International Journal of Clothing Science and Technology*, 33(5): 724–739, 2021. 1

[30] Maria Korosteleva and Sung-Hee Lee. Neuraltailor: Reconstructing sewing pattern structures from 3d point clouds of garments. *ACM Transactions on Graphics (TOG)*, 41(4):1–16, 2022. 2

[31] Maria Korosteleva and Olga Sorkine-Hornung. GarmentCode: Programming parametric sewing patterns. *ACM Transaction on Graphics*, 42(6), 2023. 1, 2, 3, 5

[32] 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. In *Computer Vision – ECCV 2024*, 2024. 3, 5, 1

[33] Changjian Li, Hao Pan, Adrien Bousseau, and Niloy J Mitra. Sketch2cad: Sequential cad modeling by sketching in context. *ACM Transactions on Graphics (TOG)*, 39(6):1–14, 2020. 3

[34] Hong Li, Nanxi Li, Yuanjie Chen, Jianbin Zhu, Qinlu Guo, Cewu Lu, and Yong-Lu Li. The labyrinth of links: Navigating the associative maze of multi-modal llms. *arXiv preprint arXiv:2410.01417*, 2024. 3

[35] Ren Li, Benoît Guillard, and Pascal Fua. Isp: Multi-layered garment draping with implicit sewing patterns. *Advances in Neural Information Processing Systems*, 36, 2024. 2

[36] Yingjie Li, Arjun Sekhon, Brandon Labash, et al. Competition-level code generation with alphacode. *Science Advances*, 8(40):eabq4412, 2022. 3

[37] Yifei Li, Hsiao-yu Chen, Egor Larionov, Nikolaos Sarafianos, Wojciech Matusik, and Tuur Stuyck. Diffavatar: Simulation-ready garment optimization with differentiable simulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4368–4378, 2024. 2

[38] Yong-Lu Li, Xiaoqian Wu, Xinpeng Liu, Zehao Wang, Yiming Dou, Yikun Ji, Junyi Zhang, Yixing Li, Xudong Lu, Jingru Tan, et al. From isolated islands to pangea: Unifying semantic space for human action understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16582–16592, 2024. 3

[39] Chen Liu, Weiwei Xu, Yin Yang, and Huamin Wang. Automatic digital garment initialization from sewing patterns. *ACM Transactions on Graphics (TOG)*, 43(4):1–12, 2024. 2

[40] 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. 2, 5, 7

[41] Siqi Liu, Yong-Lu Li, Zhou Fang, Xinpeng Liu, Yang You, and Cewu Lu. Primitive-based 3d human-object interaction modelling and programming. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3711–3719, 2024. 3

[42] Zhen Liu, Yao Feng, Yuliang Xiu, Weiyang Liu, Liam Paull, Michael J. Black, and Bernhard Schölkopf. Ghost on the shell: An expressive representation of general 3d shapes. *arXiv preprint arXiv:2310.15168*, 2023. 2

[43] Erik Nijkamp et al. A conversational approach to code generation using pre-trained language models. *arXiv preprint arXiv:2203.13474*, 2022. 3

[44] Wamiq Para, Shariq Bhat, Paul Guerrero, Tom Kelly, Niloy Mitra, Leonidas J Guibas, and Peter Wonka. Sketchgen: Generating constrained cad sketches. *Advances in Neural Information Processing Systems*, 34:5077–5088, 2021. 3

[45] Wamiq Para, Paul Guerrero, Tom Kelly, Leonidas J Guibas, and Peter Wonka. Generative layout modeling using constraint graphs. In *Proceedings of the IEEE/CVF international Conference on Computer Vision*, pages 6690–6700, 2021. 3

[46] Nico Pietroni, Corentin Dumery, Raphael Falque, Mark Liu, Teresa A Vidal-Calleja, and Olga Sorkine-Hornung. Computational pattern making from 3d garment models. *ACM Transactions on Graphics*, 41(4):157–1, 2022. 2

[47] Daniel Ritchie, Paul Guerrero, R Kenny Jones, Niloy J Mitra, Adriana Schulz, Karl DD Willis, and Jiajun Wu. Neurosymbolic models for computer graphics. *Computer Graphics Forum*, 42(2):545–568, 2023. 2, 3

[48] Boxiang Rong, Artur Grigorev, Wenbo Wang, Michael J Black, Bernhard Thomaszewski, Christina Tsalicoglou, and Otmar Hilliges. Gaussian garments: Reconstructing simulation-ready clothing with photorealistic appearance from multi-view video. *arXiv preprint arXiv:2409.08189*, 2024. 2

[49] Ari Seff, Wenda Zhou, Nick Richardson, and Ryan P Adams. Vitruvion: A generative model of parametric cad sketches. *arXiv preprint arXiv:2109.14124*, 2021. 3

[50] Liang Shi, Beichen Li, Miloš Hašan, Kalyan Sunkavalli, Tamy Boubekeur, Radomir Mech, and Wojciech Matusik. Match: Differentiable material graphs for procedural material capture. *ACM Transactions on Graphics (TOG)*, 39(6): 1–15, 2020. 3

[51] Yonglong Tian, Andrew Luo, Xingyuan Sun, Kevin Ellis, William T Freeman, Joshua B Tenenbaum, and Jiajun Wu. Learning to infer and execute 3d shape programs. *arXiv preprint arXiv:1901.02875*, 2019. 3

[52] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. 1

[53] Huamin Wang. Rule-free sewing pattern adjustment with precision and efficiency. *ACM Transactions on Graphics*, 37 (4):1–13, 2018. 2

[54] Kai Wang, Yu-An Lin, Ben Weissmann, Manolis Savva, Angel X Chang, and Daniel Ritchie. Planit: Planning and instantiating indoor scenes with relation graph and spatial prior networks. *ACM Transactions on Graphics (TOG)*, 38(4):1–15, 2019. 3

[55] Yue Wang, Fangxiang Wan, et al. Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8696–8708, 2021. 3

[56] Zhendong Wang, Longhua Wu, Marco Fratarcangeli, Min Tang, and Huamin Wang. Parallel multigrid for nonlinear cloth simulation. *Computer Graphics Forum*, 37(7):131–141, 2018. 2

[57] Rundi Wu, Chang Xiao, and Changxi Zheng. Deepcad: A deep generative network for computer-aided design models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6772–6782, 2021. 3

[58] Xiaoqian Wu, Yong-Lu Li, Jianhua Sun, and Cewu Lu. Symbol-llm: leverage language models for symbolic system in visual human activity reasoning. *Advances in Neural Information Processing Systems*, 36, 2024. 3

[59] Kevin Xu et al. Polycoder: Multilingual code completion and generation. *arXiv preprint arXiv:2203.10290*, 2022. 3

[60] Xiang Xu, Karl DD Willis, Joseph G Lambourne, Chin-Yi Cheng, Pradeep Kumar Jayaraman, and Yasutaka Furukawa. Skexgen: Autoregressive generation of cad construction sequences with disentangled codebooks. *arXiv preprint arXiv:2207.04632*, 2022. 3

[61] Xiang Xu, Joseph Lambourne, Pradeep Jayaraman, Zhengqing Wang, Karl Willis, and Yasutaka Furukawa. Brepgen: A b-rep generative diffusion model with structured latent geometry. *ACM Transactions on Graphics (TOG)*, 43 (4):1–14, 2024. 3

[62] Xinyu Xu, Yizheng Zhang, Yong-Lu Li, Lei Han, and Cewu Lu. Humanvla: Towards vision-language directed object rearrangement by physical humanoid. *arXiv preprint arXiv:2406.19972*, 2024. 3

[63] Xiangyuan Xue, Zeyu Lu, Di Huang, Wanli Ouyang, and Lei Bai. Genagent: Build collaborative ai systems with automated workflow generation–case studies on comfyui. *arXiv preprint arXiv:2409.01392*, 2024. 3

[64] Shan Yang, Zherong Pan, Tanya Amert, Ke Wang, Licheng Yu, Tamara Berg, and Ming C Lin. Physics-inspired garment recovery from a single-view image. *ACM Transactions on Graphics (TOG)*, 37(5):1–14, 2018. 2

[65] Zhengming Yu, Zhiyang Dou, Xiaoxiao Long, Cheng Lin, Zekun Li, Yuan Liu, Norman Müller, Taku Komura, Marc Habermann, Christian Theobalt, et al. Surf-d: High-quality surface generation for arbitrary topologies using diffusion models. *arXiv preprint arXiv:2311.17050*, 2023. 2

[66] Mingyu Zhang, Jiting Cai, Mingyu Liu, Yue Xu, Cewu Lu, and Yong-Lu Li. Take a step back: Rethinking the two stages in visual reasoning. In *European Conference on Computer Vision*, pages 124–141. Springer, 2025. 3

[67] Junsheng Zhou, Weiqi Zhang, Baorui Ma, Kanle Shi, Yu-Shen Liu, and Zhizhong Han. Udiff: Generating conditional unsigned distance fields with optimal wavelet diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21496–21506, 2024. 2

Design2GarmentCode: Turning Design Concepts to Tangible Garments Through Program Synthesis

Supplementary Material

7. Validations in Design2GarmentCode

7.1. Rule-based Validation

Rule-based validation primarily addresses issues of completeness and hallucination during the MMUA's generation process. With prompts generated by the DSL-GA containing over 100 questions, the MMUA often struggles to provide comprehensive answers in a single attempt. Additionally, due to the inherent hallucination tendencies of LLMs, some responses may fall outside the reasonable parameter range defined by GarmentCode. To mitigate these issues, we compare the MMUA's responses against a predefined complete question space to verify whether all questions have been adequately addressed before program synthesis. Each response is further validated to ensure it falls within GarmentCode's permissible parameter space. Questions with either missing or invalid answers are sent back to the MMUA for re-evaluation, with a maximum of two validation loops to refine the outputs.

7.2. MMUA Design Comparison

During design comparison we ask the MMUA to compare the output design image versus the design input and propose modification suggestions to DSL-GA to edit the generated pattern. Design comparison is especially useful for image-guided generation, where the output design image is rendered from the draped garment mesh under a similar viewpoint to the input image, we use TokenHMR [15] to estimate the camera pose and rough human pose from the input design image. The prompt used for design comparison is given in Figure 10.

Description:

Modify `text_garment_config_full` based on the original image to make the simulation image more closely match the original image.

Background Information:

- **Simulation Image:** An image generated based on the current `text_garment_config_full`. The model has black skin and wears multicolored clothing.
- **Original Image:** The original image provided by the user.
- **Current** `text_garment_config_full`: Text selections made based on the original image, which may contain recognition errors, leading to the generation of the simulation image.
- **Objective:** Ensure the simulation image generated from `text_garment_config_full` aligns more closely with the original image.

Instructions:

Step 1: Understand the Relationship

The simulation image is directly generated from `text_garment_config_full`. Fully understand how they correspond.

Step 2: Compare Images

Compare the simulation image with the original image to identify discrepancies in clothing structure, length, neckline, sleeves, skirts, pants, etc.

Step 3: Modify `text_garment_config_full`

Based on the observed differences, decide if modifications are necessary.

Make adjustments to `text_garment_config_full` to reflect the features of the original image accurately.

Comparison Focus:

Pay close attention to key elements such as clothing structure, length, neckline, sleeves, skirts, and pants.

Figure 10. Prompt for MMUA during design comparison.

8. Implementation Details

We use GPT-4V [6] for **MMUA**, and an instruction tuned version of Llama-3.2-3B for **DSL-GA** (Γ). The following sections contains the detailed explanation for the finetuned DSL-GA (Supp. 8.1), and training details for the Projector Ψ (Supp. 8.2).

8.1. Finetuning DSL-GA Γ_{ft}

To optimize the trade-off between computational cost and generation quality, we use Llama-3.2-3B-Instruct[52] as the base model for DSL-GA, fine-tuned over two epochs with LoRA (rank 16) and a learning rate of 5×10^{-4} on a dataset with 583 hierarchically defined NL-DSL pairs from GarmentCode's public code repository. All code generation experiments were conducted on a single NVIDIA GTX 4090. For multi-modal understanding tasks, GPT-4V was employed as the designated agent.

8.2. Training The Projector Ψ

The projector Ψ is trained on the GarmentCodeData [32] dataset, which comprises approximately 115,000 garment samples draped on a standard A-pose body. We generate initial design descriptions for each sample using GPT-4V or rule-based inverse mapping from the ground truth design parameters for the sample, for example

```
if design.shirt.length.v > 1.0:
    return 'shirt__length__long'
```

The token sequence length is fixed at 122, which is equal to the number of design parameters in GarmentCode. The projection MLP and Transformer decoder are designed with feature dimensions of 128. The MLP consists of 4 intermediate layers, while the Transformer decoder includes 8 layers. Training is conducted using the Adam optimizer with a learning rate of 5×10^{-4}, a batch size of 16, and completed on a single NVIDIA GTX 4090 within 10 hours.

Notably, although we adopt a decoder-only Transformer architecture similar to DressCode, our innovative approach

Figure 11. LMM-based interface for Design2GarmentCode, built upon the original GarmentCode GUI. The chat interface (left) allows users to provide natural language design descriptions or upload reference images or sketches, facilitating multi-modal design parsing into executable GarmentCode programs. We use the original GarmentCode execution engine to turn the generated program into 3D garments.

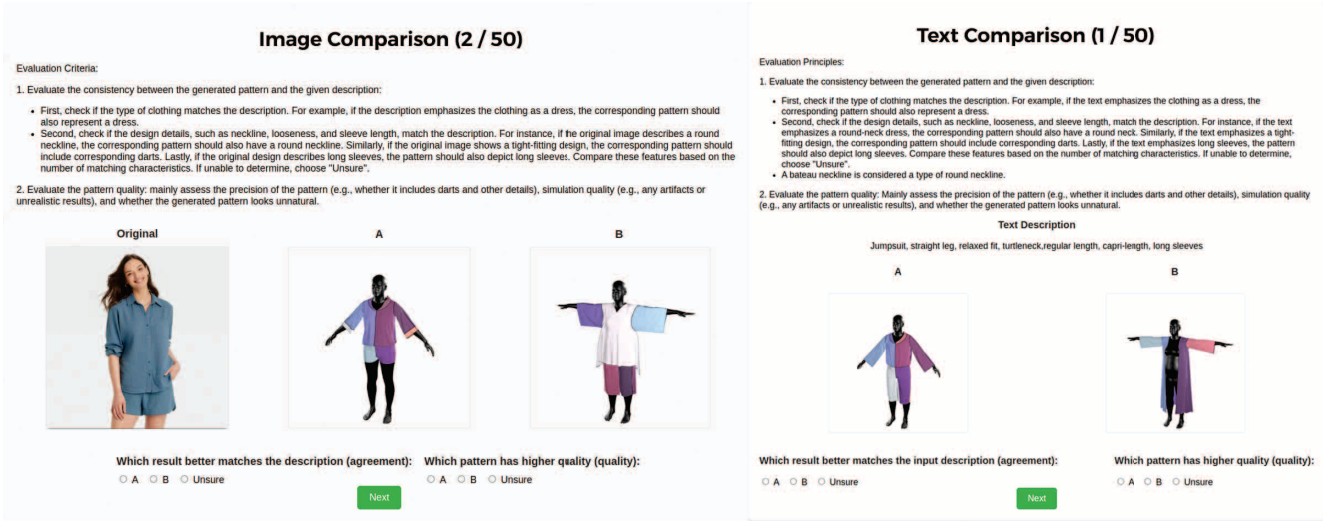


Figure 12. User study interface for evaluating sewing pattern generation quality. For each test input (Original image on the left for image-based evaluation or Text Description on the right for text-based evaluation), participants are presented with the simulation results of sewing patterns generated by Design2GarmentCode and a baseline method. Users are asked to evaluate the results based on two criteria: agreement with the input description and overall sewing pattern quality. If unsure, participants can select the "Unsure" option.

of quantifying sewing patterns through design parameters proves to be significantly more efficient and scalable. Specifically, with DressCode's quantization scheme, the token sequence length is calculated as:

$$L_{seq} = N_p \times (N_e \times L_e + \|R\| + \|T\| + N_e \times \|S\|) + 2$$

where N_p, N_e denotes the maximum number of panels and edges respectively. $\|R\| = 4$ is the length of rota-

Figure 13. Example answers from Llama 3.2 3B when prompted with *"How to draft a basic upper body bodice?"*.

tion quaternions, and $\|T\| = 3$ is the length of 3D translation vector. $\|S\| = 4$ represents the per-edge stitching parameters containing a stitch tag and its existence indicator. L_e represent the length of quantified edge vectors, which might be 6 for cubic bezier curves and 4 for quadratic bezier curves. Using GarmentCodeData as an example, to fully cover GarmentCode's modeling space, the required sequence length under DressCode's method would be $13,951$, with $N_p = N_e = 37, L_e = 6$, which will cost $\approx 1.5h$ to generate a single sewing pattern using DressCode, while our token sequence length is fixed at 122.

8.3. Inference Interface

For more convenient inference, we build an intelligent chat-based interface integrated into the original GarmentCode [31] GUI (Figure 11). The chat interface (left panel) enables users to provide natural language descriptions, upload reference images, or supply design sketches, facilitating multi-modal design parsing into GarmentCode-compliant programs which are then passed to the GarmentCode execution engine. The engine generates sewing patterns and 3D garment simulations (right panel). This interactive interface provides an intuitive environment for creating, editing, and refining sewing patterns, significantly improving accessibility for users without extensive expertise in parametric pattern-making. We provide a recording to demostrate the inference process in demo.mp4.

8.4. User Study Interface

To evaluate the quality of sewing pattern generation, we design a user study interface tailored for comparison (Figure 12). For each test input—either an original image (for image-based evaluation) or a text description (for text-based evaluation)—the interface presents participants with simulated garment results generated by Design2GarmentCode and a baseline method. Participants assess the results based on two criteria: **Agreement**, which measures how well the generated patterns align with the input description, and **Aesthetic**, which evaluates the structural integrity and aesthetic appeal quality of the patterns. An "Unsure" option is available for cases where a clear preference cannot be determined, ensuring unbiased and flexible feedback.

9. LMM Prompting Details

9.1. Pattern Drafting Test

As outlined in Sec. 3.2.1, a key prerequisite for Design2GarmentCode is the presence of embedded pattern-drafting knowledge in pre-trained large models. To assess this capability, we prompted models like O1-preview and LLama 3.2 3B Instruct with the question, "How to draft a basic upper body bodice?". These models produced step-by-step drafting instructions in natural language, including commands such as: "STEP 1: Take Your Measurements," and "STEP 2: Draw the Center Front Line, Draw the Shoulder Line, Draw the Armhole, Draw the Side Seam (Measure the distance from the underbust measurement and divide it by 4. Mark this distance from the armhole point down to the waist. Draw a vertical line to represent the side seam)." Figure 13 showcases sample outputs from Llama-3.2-3B Instruct which we used as baseline for DSL-GA.

9.2. Prompting for MMUA

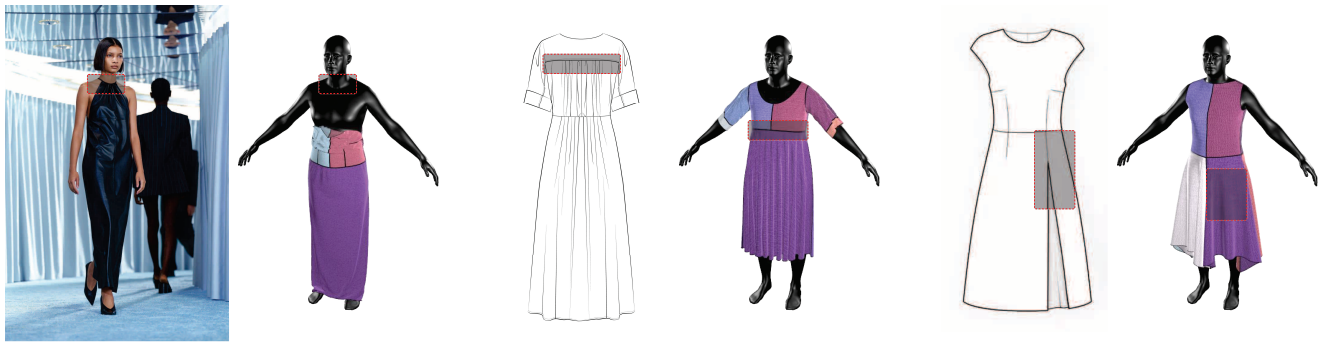
Based on different input design modalities and tasks, we assigned five specific tasks to the MMUA.

Task 1: Identify the image, extract answers for each prompt question based on the image, and combine them to form the recognized clothing information. This task establishes the relationships between parameters and the questions corresponding to each parameter. It serves as the foundation for all subsequent tasks.

Task 2: Generate clothing information based on text. Building on Task 1, this task generates clothing prototype information according to user preferences.

Task 3: Retrieve existing clothing information and modify the clothing design according to the user's ideas.

Task 4: Input stress test images along with the current clothing information from the text space. MMUA interprets the colors in the image as stress levels—red, yellow, or similar colors indicate areas that are too tight. MMUA dynamically adjusts the clothing information to reduce stress.



(a) Thin Structure (Halter-neck) (b) Unconventional Bodices (c) Partial Stitching

Figure 14. Limitations of Design2GarmentCode, including failed to modeling thin structures like halter-neck, unable to model unconventional bodices and stitching relationships are limited to one-to-one mapping.

Task 5: Compare previously generated clothing simulation images, their corresponding clothing information, and the original input image. Identify differences between the simulation and the original image, and dynamically adjust the clothing information to make the final simulation image more closely resemble the original.

As discussed in Sec. 3.2.2, due to the probabilistic nature of LMMs, the MMUA struggles to accurately estimate numerical values in the design configuration. Therefore, we limit the MMUA's task to answering multiple-choice questions, with responses formatted as a selective parameter list. Fig. 15 illustrates example parameters before (`design_cfg_num`) and after (`design_cfg_slc`) modification. The complete prompt will be publicly available with Design2GarmentCode code base.

10. Limitations

A limitation of Design2GarmentCode is its current inability to substantially modify GarmentCode's underlying structure and logic, which impacts the generation quality due to inherent constraints in GarmentCode's design and modeling capabilities. For example, the range of upper garment patterns is limited, making it difficult to model personalized segmentations (Figure 14 (b)). Additionally, for designs like halter necks or strapless tops (Figure 14 (a)), GarmentCode cannot model fine straps, leading to potential simulation failures. These constraints restrict the system's ability to accurately represent certain complex or customized garment designs.

11. Additional Results

In the following, we present additional experimental results in text-, image-, and sketch-guided pattern generation and highlight the versatility and effectiveness of our approach across various modalities.

```
# Numerical parameters
design_cfg_num = [
    # Meta Section
    "meta.upper.v=Shirt",
    "meta.wb.v=FittedWB", # waistband type
    "meta.bottom.v=SkirtLevels",
    "meta.connected.v=False",
    # Waistband Section
    "waistband.waist.v=1.05", # Fitted
    "waistband.width.v=0.2",  # Narrow
    # Fitted Shirt Section
    "fitted_shirt.strapless.v=False",
    # Shirt Section
    "shirt.length.v=1.55", # Long
    ...
    ]

# Selective parameters
design_cfg_slc = [
    # Meta Section
    "meta__upper__Shirt",
    "meta__wb__FittedWB", # waistband type
    "meta__bottom__SkirtLevels",
    "meta__connected__False",
    # Waistband Section
    "waistband__waist__fitted", # Fitted
    "waistband__width__narrow", # Narrow
    # Fitted Shirt Section
    "fitted_shirt__strapless__False",
    # Shirt Section
    "shirt__length__long", # Long
    ...
    ]
```

Figure 15. Example of original design configurations with numerical values and modified design configurations with only selective parameters.Example of original design configurations with numerical values and modified design configurations with only selective parameters.

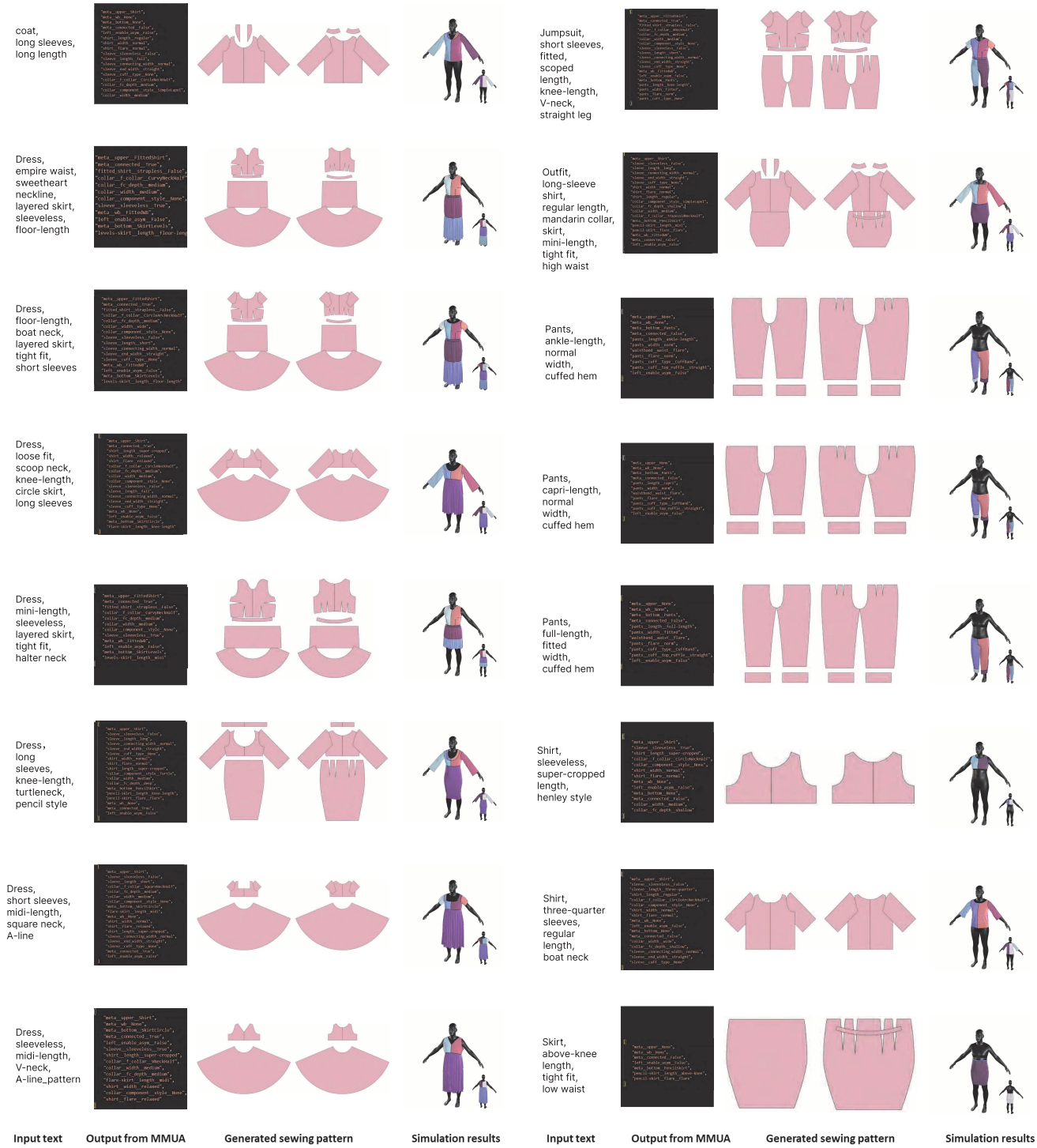


Figure 16. Additional Text-guided generation results. From left to right: input text; output from MMUA; generated sewing pattern; simulation results.

Figure 17. Additional Image-guided generation results. From left to right: input image; output from MMUA; generated sewing pattern; simulation results.

Figure 18. Additional Sketch-guided generation results. From left to right: input sketch; output from MMUA; generated sewing pattern; simulation results.

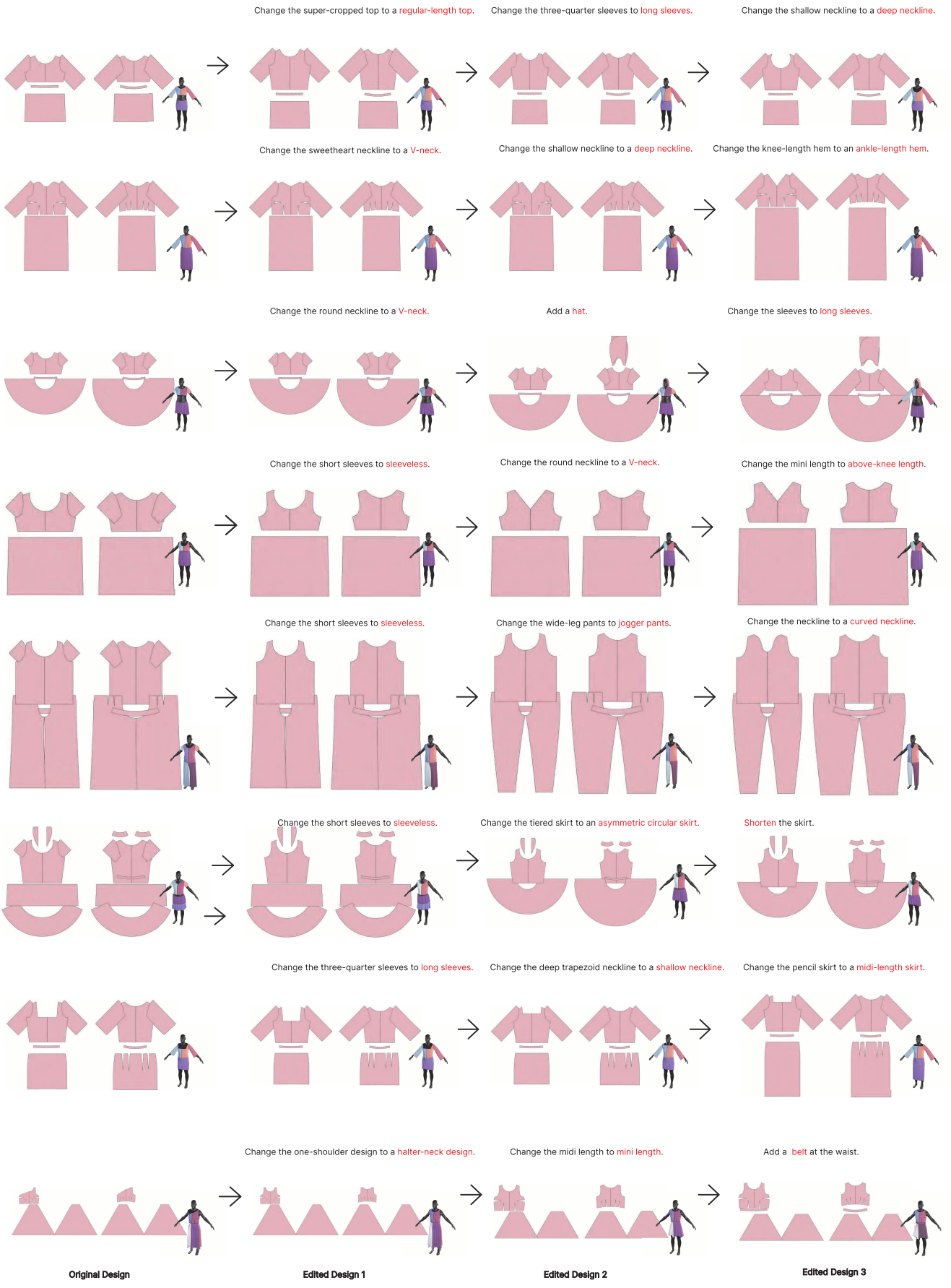


Figure 19. Additional sewing pattern editing results.Starting from the original sewing pattern on the far left, the system applies user instructions to edit the pattern. The left side of each arrow represents the original pattern, while the right side displays the edited result.