GarmentoPIA: Generating Garment Pattern Models using Intelligent Agents

Mingi Yeom, Rimsoo Shin, Sung-Hee Lee
Graduate School of Culture Technology, KAIST
Daejeon, South Korea
mingiyeom11@gmail.com, {flyingtiger, sunghee.lee}@kaist.ac.kr

Abstract

A parametric garment pattern model is widely used to generate 2D and 3D garment representations in reconstruction and simulation tasks. However, existing models are often difficult to adapt for specific purposes due to limited variations in base pattern shapes and the rigid, hardcoded handling of body and garment measurements. To address these challenges, we propose GarmentoPIA, a system that automatically generates parametric garment pattern models from garment drafting literature using an LLM-based intelligent agent module. With GarmentoPIA, users can select garment drafting references and generate a wide range of pattern models that incorporate all tailoring parameters specified in the source material. To ensure robustness across diverse literature, the system employs a prompt self-refinement mechanism that iteratively updates its instructions during model generation. Furthermore, GarmentoPIA introduces a garment domain-specific language (DSL), composed of explicit function calls corresponding to drafting components, to produce models independent of LLMs—enhancing usability and accessibility. We validate the effectiveness of our approach through extensive quantitative and qualitative evaluations.

Keywords: Garment pattern modeling, prompt refinement, domain-specific language, large language model.

1. Introduction

The fundamental structure of a garment is its 2D sewing pattern, a collection of flat panels stitched together to create a 3D form. As such, these patterns serve as the essential blueprint for both physical and virtual garments. From a computational perspective, a pattern is usually represented as a parametric model. In this model, its fundamental class (*e.g.*, a skirt, pants, or bodice) is formed by a geometric structure of vertices and edges, which are in turn controlled by a set of semantic parameters (*e.g.*, 'Hip Depth' or 'Back Skirt Length'). These parameters are typically derived from both human body measurements and the garment measurements specified in the design.

Driven by growing interest in high-quality garment modeling, researchers have developed various parametric garment pattern models. For example, [20] supports basic garments like T-shirts, skirts, and pants, while [22] incorporates additional design elements such as waistbands, cuffs, and darts. Further, [19] models garment deformation as a function of body shape, making these models powerful priors for garment estimation and simulation.

Despite these advances, most parametric models are manually designed and therefore limited in scope. They are often configured to represent only a specific style within each class, a significant constraint given that garment styles vary significantly across individual aesthetics, cultural contexts, and historical periods. Manually crafting pattern models to accommodate this diversity is therefore time-consuming and labor-intensive.

To overcome these limitations, we propose GarmentoPIA—a garment pattern model generator that uses intelligent agents built on large language models (LLMs) to extract and operationalize drafting knowledge from existing garment drafting literature (Figure 2). Moreover, our intelligent agentic approach dramatically reduces the need for manual intervention when designing diverse and complex garments. Figure 1 shows the garment patterns generated by models produced by GarmentoPIA, referencing existing pattern drafting books.

GarmentoPIA's key feature is its ability to generate each garment pattern model as an LLM-independent executable program. Once generated, these models can be repeatedly used with new inputs, such as body measurements, without incurring the cost and latency of commercial LLM services. This makes GarmentoPIA a highly scalable and accessible solution for both research and design applications.

We employ GPT [1] as the backbone of our system due to its general-purpose capabilities and widespread availability. Among the two major strategies for adapting LLMs to domain-specific tasks, parameter fine-tuning [12] and prompt refinement [38], we adopt the latter for its robustness across datasets of varying size. While vision-language models (VLMs) are another option, their utility is limited by

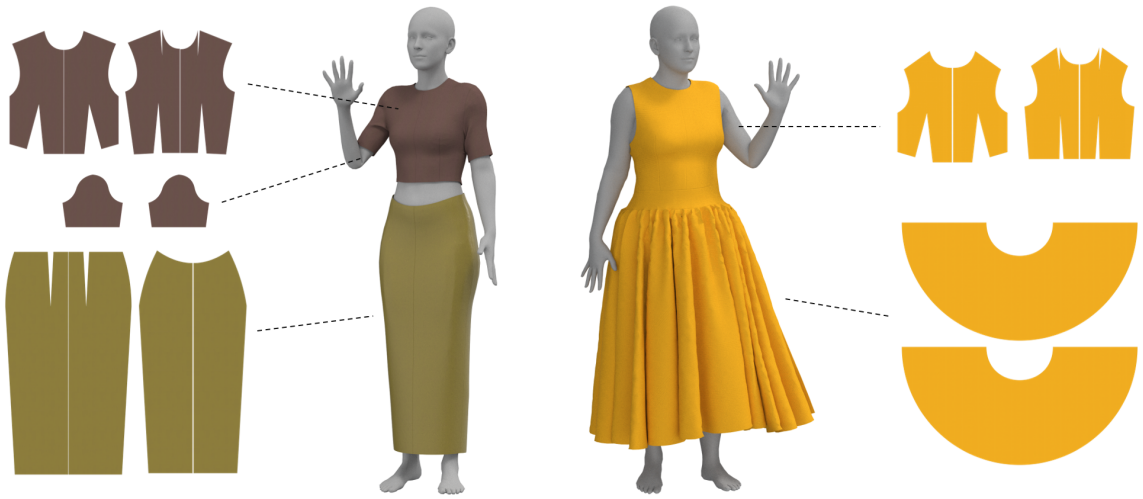

Figure 1. Garment patterns produced by the garment pattern models created by GarmentoPIA with reference to a pattern design book [34] as well as their simulated results.

the nature of garment drafting literature, which tends to rely on illustrations rather than numerical data. To ensure clarity and reproducibility, we base our system on language-centric literature such as [34], which provides step-by-step drafting instructions suitable for structured extraction.

A major challenge lies in translating such textual literature into formalized pattern models. Drafting instructions often assume prior domain knowledge, use ambiguous terms, or repurpose techniques across garment types (*e.g.*, adapting methods for skirt to pants). Consequently, simply feeding raw text to an LLM and prompting it to generate a pattern model does not yield viable results. Instead, we devise a structured, multi-step pipeline tailored for this task.

This challenge is addressed by a structured, three-stage pipeline anchored by the Garment Component Generation Module. An initial preprocessing stage refines the raw drafting literature, resolving ambiguities and normalizing the text into a machine-readable format. The core module then employs SQL-based intelligent agents and Retrieval-Augmented Generation (RAG) to parse these instructions, populating a structured database with pattern components—the points, lines, and constants (measurements) that constitute a 2D pattern— defined in a domain-specific language (DSL). To ensure robustness, this module features a prompt self-refinement mechanism, where the agents iteratively review the generated components and refine the guiding prompts to enhance the translation accuracy of subsequent steps. Finally, a postprocessing stage validates the database for logical consistency and synthesizes the components into a coherent, executable program.

GarmentoPIA offers a systematic and extensible framework for generating garment pattern models with fidelity and stylistic diversity. By allowing users to specify the source literature for pattern drafting, GarmentoPIA can produce models that are personalized while encompassing a wide spectrum of tailoring-related parameters. This approach has a potential to significantly reduce the technical and resource barriers traditionally associated with high-quality garment modeling.

To summarize, our contributions are as follows:

- We propose the first parametric garment pattern model generator from the garment drafting literature.
- The Garment Component Generation Module enhances its performance through self-prompt refinements.
- The generated garment model supports all garment-tailoring parameters and the body measurements specified in the garment drafting literature.

Additional resources and details are provided in the project page at https://github.com/MingiYeom/GarmentoPIA/.

2. Related Work

2.1. Garment Modeling

The automated modeling of garments has been approached through several distinct paradigms. Initial efforts concentrated on direct reconstruction, aiming to infer 3D geometry, 2D sewing patterns, or implicit surface functions directly from sparse inputs like a single image or depth scan [3, 5, 6, 16, 37, 45]. However, unlike natural objects, garments are man-made artifacts with well-defined, classifiable structures. Capitalizing on this inherent property, [20] introduced a foundational 2D parametric garment

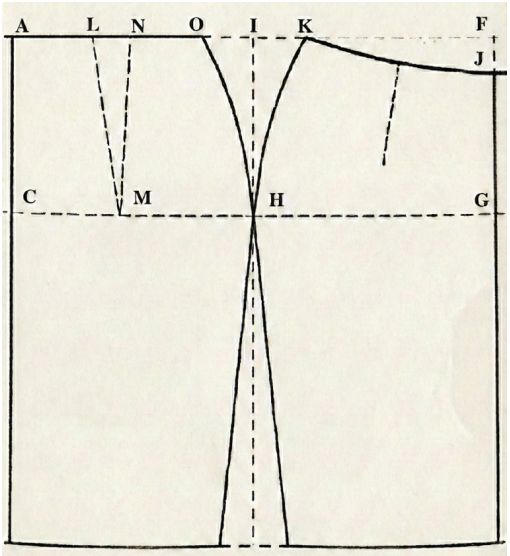

Figure 2. Front and back panel draft and explanation of the Two-Pieced Skirt Sloper pattern, redrawn based on [34] with original text retained. Instructions for symmetric parts are omitted per standard practice.

model and an accompanying large-scale dataset. This contribution catalyzed a shift in the field; many subsequent methods [4, 21, 26, 27, 29] have since adopted this parametric model as strong prior knowledge to improve accuracy and robustness. By operating on these parameterized patterns, these approaches afford greater control, enabling the synthesis of realistic and editable virtual garments from a wide array of inputs, including textual descriptions and conversational cues. However, existing parametric methods [19, 20, 22] share several limitations. Primarily, they rely heavily on predefined parameters and require programming skills, such as creating garments through JSON formats, limiting designers' ability to express their creativity. Furthermore, parametric models frequently fall short in accurately capturing intricate garment details such as darts, pleats, curves, and decorative elements, resulting in simplified patterns that do not align with industrial production standards. To lower the technical barrier for designers and move towards more natural language interaction, recent multimodal systems such as AIpparel [30], GarmentImage [41], Design2GarmentCode [48] and WordRobe [40] generate or edit sewing patterns from short prompts (*e.g.*, 'make it longer'). While these works rely on predefined parametric templates (GarmentCode template) for editing, GarmentoPIA derives centimeter-accurate DSL programs from drafting texts, enabling generation of templates with explicit geometric constraints, such as "extend rightwards with a distance of 1/2 × Front Hip" and "reduce Crotch Length with a distance of Crotch Depth - 1/4-inch".

GarmentoPIA addresses these limitations by introducing a novel garment modeling framework leveraging LLMs. By directly interpreting textual instructions from clothing-making literature, our method greatly increases the diversity of automatically generated patterns. With the use of LLMs, our framework offers designers limitless creative possibilities by enabling them to produce a wide variety of clothing without the need for programming knowledge. Additionally, our technique captures garment details by interpreting design elements and complex geometric configurations directly from detailed textual descriptions.

2.2. LLM-based Intelligent Agent

LLM-based intelligent agents integrate reasoning, tool use, and self-correction to transform static models into adaptive problem solvers. [46] created the synergy of chain-of-thought with external actions by allowing a model to plan, query an API, and update its plan in a single prompt cycle, which resulted in a reduction in hallucinations. [38] pushed further, showing that agents can store free-form "verbal rewards" after each trial and reuse them to reach without gradient updates. This pattern was formalized into a multi-agent conversation framework by Autogen [44], which showed that collaborative teams of specialized LLMs can outperform single-agent baselines in code analysis, chess, and math. Together, these studies demonstrate that agents are highly skilled in selective tool invocation, planning, and iterative error recovery; skills that are necessary for parsing dependent draft instructions. Recent Symbolic-LLM hybrid frameworks [31, 32] have demonstrated that augmenting LLMs with external symbolic solvers significantly enhances logical consistency and mathematical precision. However, generic agents cannot ensure geometric validity because they work over untyped text buffers. By connecting agents to a schema-aware SQL store of Points, Lines, and Constants, GarmentoPIA closes this gap and guarantees that every action generates deterministic, geometric components. The same agents use reflected critiques to fix flawed equations and RAG to retrieve previous steps. As a result, our system provides the accuracy required by industrial pattern drafting while inheriting the interpretability and robustness of intelligent agents.

Retrieval-Augmented Generation (RAG). RAG provides evidence that significantly reduces hallucinations and enhances factual grounding by integrating a parametric language model with an external, non-parametric memory. By injecting retrieved documents at inference time, the RAG architecture [25] demonstrated that even a small retriever–generator stack could outperform much larger standalone Language Models (LM) on open-domain QA. Fur-

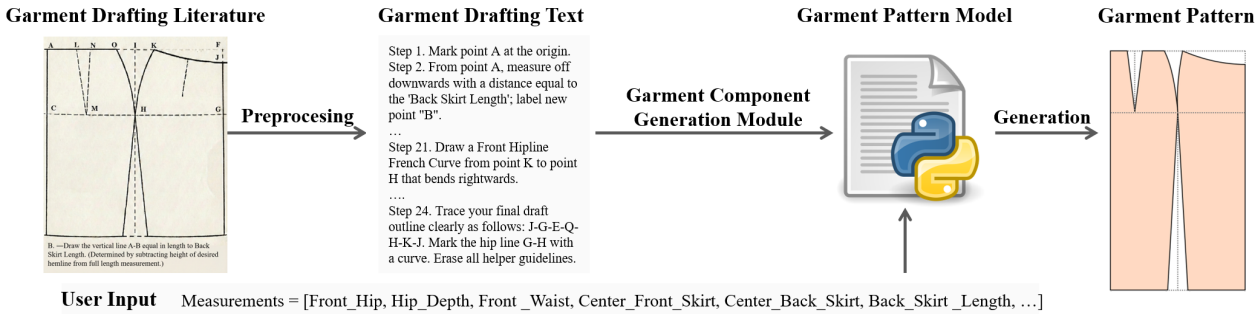

Figure 3. Using the garment drafting text, GarmentoPIA generates a parametric garment pattern model iteratively, and the models generate a 2D garment pattern. The pattern is simulated on the human body based on a garment simulator.

ther studies [10, 17] have shown that integrating retrieval mechanisms significantly enhances model performance and flexibility. Methods such as RAG-Fusion, and Self-RAG [2, 36] as well as Fusion-in-Decoder variants [13, 14, 15], have demonstrated improvements in retrieval-augmented generation by effectively aggregating and refining retrieved content, thereby enhancing coherence and factual accuracy. Due to these strengths, when coupled with a refined dataset derived from garment literature, RAG becomes highly suitable for GarmentoPIA. Additionally, the inherent scalability of RAG ensures our research framework remains adaptable as our garment dataset continues to expand.

Prompt Refinement. Prompt-level interventions are a straightforward way to direct LLM behavior without the need for costly fine-tuning. Iterative Prompt Refinement, which leads users through draft, evaluate, and tweak cycles, consistently raises task scores with only textual changes. Techniques such as Chain-of-Thought prompting and its variants explicitly elicit intermediate reasoning steps, significantly improving arithmetic and commonsense accuracy across models [18, 42, 43, 47]. From the first "automatic prompt optimization" pipelines to the more recent AutoPDL [39], automatic frameworks search the combinatorial space of templates, demonstrations, and agentic patterns to identify high-performing prompts that often match or surpass supervised baselines. Crucially, prompt engineering operates without reference to the training distribution, making it robust to the model being exposed to rare garment terminology or drafting terminology that has existed for a century. These concepts are embodied in GarmentoPIA's updating module, which rewrites unclear instructions into exact DSL calls and prompts the model again whenever geometric checks are unsuccessful.

2.3. Domain Specific Language in Garment Modeling

General-purpose text cannot provide the reusable abstractions and syntactic guarantees that domain-specific languages (DSLs) do. Procedural modeling has long been established in computer graphics for generating complex structures, ranging from recent grammar-based architectural synthesis [11, 35] to neural-aided L-systems for plant modeling [23]. In the fashion domain, the first DSL specifically for sewing patterns was introduced by GarmentCode [22], which allowed designers to programmatically create hierarchical, parameterized clothing by mapping object-oriented constructs to points, curves, and seams. Its component structure allows for semantic edits, such as lengthening a sleeve or changing a waistband, while maintaining pattern validity, which is advantageous for mass customization. Previous parametric-template studies also depicted skirts, pants, and t-shirts as low-dimensional parameter spaces, showing how fit and silhouette are controlled by a small number of measurements. GarmentX [9] translate DSL concepts into 3D by encoding clothing with semantically significant slots that can still be edited after simulation. To address the bottleneck of manually authoring DSLs that can translate well across historical pattern books and cultural styles, GarmentoPIA uses agentic RAG to automatically extract DSL equations from garment literature, populating a schema of Points, Lines, and Constants without the need for human coding. The produced scripts combine the generative extensiveness of LLMs with the rigor of DSLs to create a seamless integration with current cloth simulators. Large-scale, diverse clothing modeling is made possible by this combination, which also lays the groundwork for subsequent processes, such as sustainable virtual sampling and automatic fit estimation.

3. Method

3.1. Overview

GarmentoPIA is a system that automatically generates executable garment pattern models from garment drafting literature. It leverages a novel approach using SQL-based intelligent agents to interpret textual descriptions

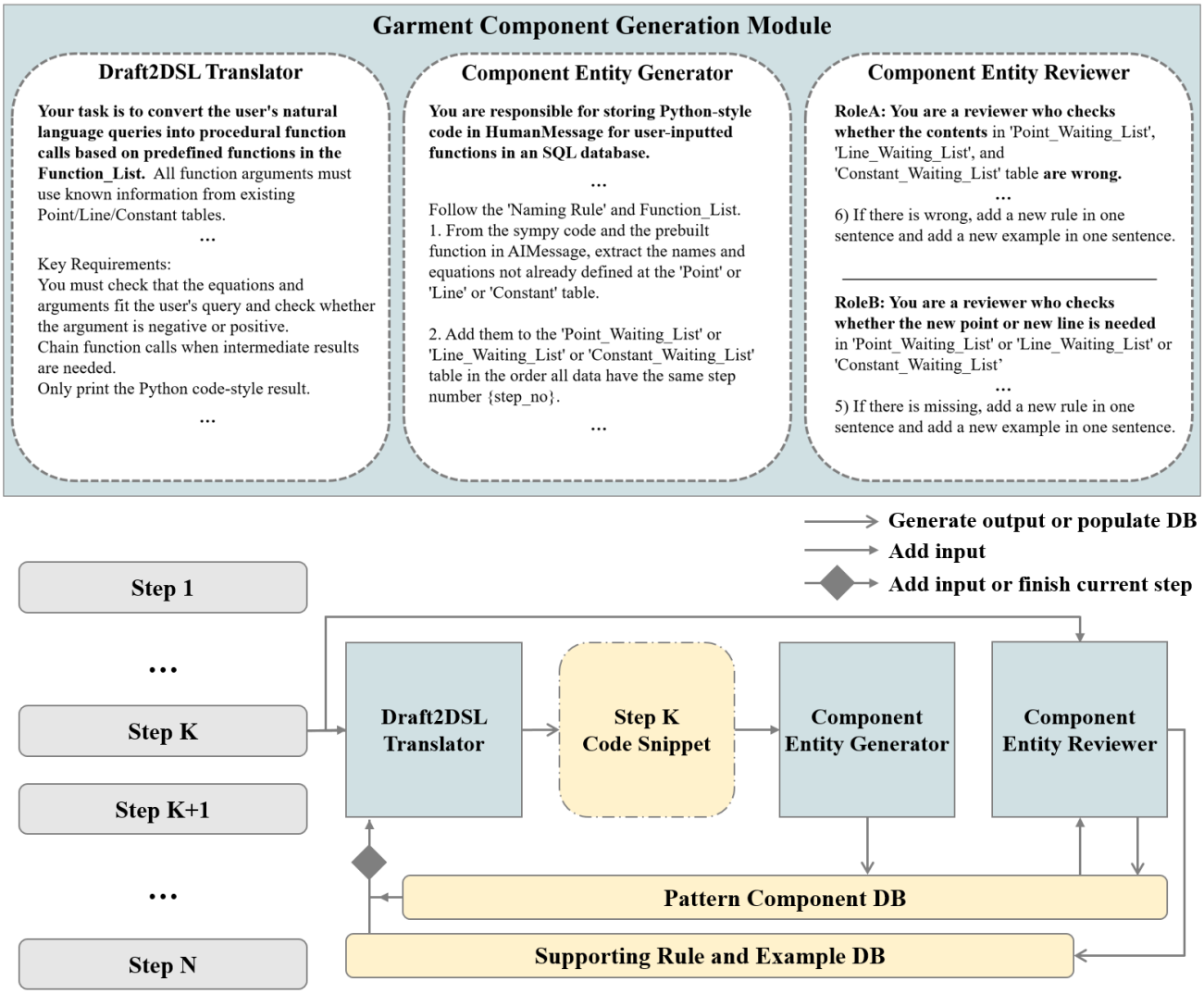

Figure 4. The iterative data processing pipeline of GarmentoPIA, illustrating the instructions for each agent role. Garment Component Generation Module operates on each step $(1 \cdots N)$ of garment drafting text sequentially to populate Pattern Component DB.

and convert them into a structured, machine-readable format. The system is architecturally divided into two primary components: the Garment Component Generation Module, which processes the input text, and the Pattern Component Database, which stores the structured data.

The overall process begins when the module interprets the drafting text and translates it into a DSL designed for garment construction. This structured data is then populated into the Pattern Component DB. Subsequently, a post-processing unit validates the final database entries, corrects syntactical errors, and generates the final output: a Python-based model capable of producing a 2D garment pattern from a set of tailoring parameters. Figure 3 illustrates the complete end-to-end pipeline, from the input drafting text to the final 2D garment pattern.

3.2. Text Data Preprocessing

To collect garment pattern drafting data, we selected Modern Pattern Design [34], a widely used reference in the garment industry. We specifically chose this foundational reference published earlier than more contemporary literature to demonstrate the robustness of our method by proving its capability to handle classic, less structured instructions. For our experiments, we focused on seven pattern categories: bodice with a dart, bodice without darts, arm sleeve, pants, two-piece skirt sloper, full circular skirt, and waistband/cuff. Pants and skirt categories include both front and back components within the same drafting procedure. Following the descriptions in the book, we structured each garment model as a sequence of pattern drafting in-

structions.

However, most pattern drafting instructions in the literature, which combine images and natural language, pose significant challenges for accurate computational interpretation. For instance, drafting instructions for a pants panel may refer to steps in a separate section for skirt drafting, requiring cross-referencing. Illustrations may not precisely correspond to the underlying pattern geometry, making it difficult to match visual elements to specific numerical measurements. Furthermore, ambiguities in the text complicate automated interpretation.

To resolve these issues, we employed a systematic manual preprocessing pipeline to convert unstructured drafting instructions into a formal, machine-interpretable representation. Specifically, we convert converts free-form natural language into a consistent structure. Each instruction is transformed into a standardized five-part format: `[Start Point, Action, Direction, Distance/Condition, Result Point]`. For example, the following instruction: *"B-2—This line equals Full Bodice Length measurement on the chart. Lay square along line A-B with short arm downward pointing to the left and the long arm extending upward, falling upon points B and A. From point B, measure off a distance equal to the Full Bodice Length and mark dot. Label dot 2."* is converted to *"From point B, measure off upwards with a distance equal to the 'Full Bodice Length'; label the new point '2'."* As another example, the source text *"F-G—This line equals the line A-C."* is converted to *"From point F, measure downwards a distance equal to the length of segment A-C to create point G."* with reference to the accompanying image.

3.3. Function Declaration for the DSL

We design a DSL to formalize garment drafting operations with clarity, modularity, and computational robustness. The DSL is inspired by traditional pattern-making tools—most notably, rulers and compasses—which form the foundation of manual garment construction. Depending on the drafting need, these tools appear in specialized forms such as straight edges, circles, ellipses, and French curves. Our DSL reflects this diversity through functionally distinct abstractions, implemented with support from Python's math and sympy libraries [28].

Each DSL function produces a drafting component, such as a point, line segment, or curve. Formally, a drafting operation is defined as a tuple: $d = (d_d, d_o)$. Here, $d_d \in D_d$ represents the function declaration, and $d_o \in D_o$ specifies its execution priority in the drafting sequence.

The declaration set D_d includes functions for both constructing geometric components and finding their intersections. A representative primitive is *make_a_point_by_relative_distance(anchor, dx, dy)*, which defines a new point by applying horizontal (*dx*) and vertical (*dy*) offsets to a source point (*anchor*). To accommodate the full spectrum of drafting requirements, the DSL further extends to complex curved geometries, supporting functions such as *make_a_circle(...)* and *make_a_french_curve(...)*. Each function in D_d includes type hints for all component parameters, enabling syntactic and semantic validation. The usage priority set D_o imposes an execution order based on computational complexity. For example, distance calculations are prioritized as simple primitives, while French curve constructions are treated as higher-order operations due to their geometric intricacy.

French curves are typically left to the discretion of the designer in traditional drafting. To systematize their usage, GarmentoPIA represents them as Bézier curves, parameterized according to garment design literature.

To mitigate the inconsistent naming conventions often generated by LLMs, GarmentoPIA enforces a standardized naming scheme across components and functions. Examples include: *Point<ID>(*e.g.*, PointC), Segment_<StartPointName>_<EndPointName>(*e.g.*, Segment_PointA_Point3)*. This ensures that the Component Entity Reviewer and the postprocessing unit can reliably validate the DSL and resolve errors.

3.4. Pattern Component Database

The Pattern Component DB is implemented using SQL and stores all components required during the garment drafting process. Each table entry consists of three fields:

- Name: The identifier for a point, line, or constant, following the naming conventions in Section 3.3.
- Equation: A numerical value or a function call representing the component's definition.
- Step_Number: The step in the drafting sequence when the component is generated.

The database is organized into two main categories of tables, Final Tables and Waiting List Tables. Final Tables include 'Point', 'Line', and 'Constant' tables that store components whose geometric properties have been fully resolved. In contrast, Waiting List Tables include 'Point_Waiting_List', 'Line_Waiting_List', and 'Constant_Waiting_List', which temporarily hold components that are not yet ready for evaluation due to dependencies on future drafting steps.

During the drafting process, the Component Entity Reviewer monitors the waiting lists and promotes entries to the final tables once the drafting step matches the Step_Number.

At initialization, the DB is pre-populated with essential constants, such as body measurements and garment measurements, which define the parametric basis of the garment model. These constants allow the model to dynami-

cally adapt to user-defined inputs, enabling a fully parametric drafting workflow.

Algorithm 1 Generating the garment model from the DB

Initialize:
Load tables: Points T_p, Lines T_l, Constants T_c, Measurements T_m
Convert each table into a dictionary: D_p, D_l, D_c, D_m
Set predefined dictionary: $D_{\text{predefined}} \leftarrow D_m$
Initialize: `emitting` $\leftarrow \emptyset$, `code_lines` $\leftarrow [\,]$
Import f_{graph} from `graph_ops.py`
function EMIT(name, eq)
    if name $\in D_{\text{predefined}}$ **then**
        return
    end if
    if name $\in$ `emitting` **then**
        raise CycleError
    end if
    `emitting` $\leftarrow$ `emitting` $\cup\{\text{name}\}$
    for token t in `ARG_TOKENS`(eq) **do**
        if $t \in D_{\text{predefined}}$ **then**
            continue
        else if $t \in D_l$ **then**
            EMIT(t, $D_l[t]$)
        else if $t \in D_p$ **then**
            EMIT(t, $D_p[t]$)
        else if $t \in D_c$ **then**
            EMIT(t, $D_c[t]$)
        else
            raise UnknownSymbol(t)
        end if
    end for
    `code_lines.append(f"{name} = {f_graph(eq)}")`
    $D_{\text{predefined}} \leftarrow D_{\text{predefined}} \cup \{\text{name}\}$
    `emitting` $\leftarrow$ `emitting` $\setminus\{\text{name}\}$
end function
for (name, eq) in D_p from T_p **do**
    EMIT(name, eq)
end for
return `ordered_steps.py`

3.5. Garment Component Generation Module

The Garment Component Generation Module parses each step of the garment drafting text sequentially to populate the Pattern Component Database. This process is handled by three intelligent agents, Draft2DSL Translator, Component Entity Generator, and Component Entity Reviewer, each defined by a specific prompt-based role. Detailed prompt structures for each role and the operational workflow of the GarmentoPIA system are illustrated in Figure 4.

The Draft2DSL Translator generates code snippets from individual drafting steps. The Component Entity Generator analyzes these snippets to identify and add new component values to the waiting list. Finally, the Component Entity Reviewer validates the waiting list entries, migrating them to the final database or adding supporting rules and examples.

This modular division into three roles is designed to optimize task-specific performance and manage the context window effectively. All agents interact with the database using SQL queries. Among them, only the Draft2DSL Translator focuses solely on code generation; the other two also contribute to populating the Pattern Component Database.

3.5.1 Draft2DSL Translator

The Draft2DSL Translator processes each drafting step and translates it into a code snippet. It searches the Pattern Component Database using Retrieval-Augmented Generation (RAG) to reference existing component definitions. The resulting code snippet includes function calls from either the DSL (D_d) or the SymPy library. Each prompt given to the translator includes a supporting rule and a step-specific example to guide accurate generation.

3.5.2 Component Entity Generator

The Component Entity Generator takes the translated code snippet and formulates queries to insert new component data into the Pattern Component Database. At this stage, all data is appended to the waiting list, not directly to the final database. The generator also checks for and removes any duplicates that already exist in the database.

3.5.3 Component Entity Reviewer and Prompt Refinement

The Component Entity Reviewer operates in two distinct modes depending on the state of the waiting list. If the list is empty, it analyzes the drafting instruction to identify any components that were missed in this drafting step. If the list contains entries, it validates them, ensuring that they accurately reflect the intent of the drafting instruction.

If discrepancies like missing or incorrect entries are found, the Reviewer triggers a refinement cycle: it updates the relevant supporting rules or examples, clears the waiting list, and re-initiates the process starting from the Draft2DSL Translator. Once the entries are approved, they are migrated from the waiting list to the final tables, and the list is cleared for the next step.

Prompt refinement involves two key types of guidance. One is supporting rules, which apply to the overall drafting step, and the other is supporting examples, which focus on the specific step at hand.

To maintain clarity and conciseness, both rules and examples are constrained to single sentences. Consequently, more complex or ambiguous drafting steps naturally accumulate a richer set of rules and examples over multiple refinement iterations, effectively improving translation accuracy for similar steps in the future. These refined prompts are shared between the Draft2DSL Translator and Component Entity Reviewer, enabling GarmentoPIA to continuously enhance its accuracy without requiring model fine-tuning.

3.5.4 Case Study: Iterative Refinement from Text to Validated DSL

To provide a concrete example showing how GarmentoPIA autonomously rectifies geometric inconsistencies, We trace the error recovery for the instruction: *"From point 3, measure 10 inches downwards; label the new point Q."*

Stage 1: Initial Error (Iteration 0)
The *Draft2DSL Translator* generates flawed code, which the *Component Entity Generator* attempts to register. However, the *Reviewer* halts the process upon detecting an error.

- **Flawed Code:** *PointQ = make_a_point_by_relative_distance(Point2, 0.0, -10.0)*
- **Reviewer Diagnosis:** The *Reviewer* detects the logical mismatch between the instruction ("from point 3") and the code argument (*Point2*), marking it *WRONG*.

Stage 2: Rule Update
The Reviewer updates the *Supporting Rule and Example DB*:

- **Rule:** You must anchor new points to the specified source point.
- **Example:** If the instruction is "From point 3, draw a vertical segment 10-inch downwards; label the new point Q," then set *PointQ = make_a_point_by_relative_distance(Point3, 0.0, -10.0)*.

Stage 3: Refined Generation (Iteration 1)
Retrieving this updated context, the Translator corrects the anchor to *Point3*.

- **Corrected Code:** *PointQ = make_a_point_by_relative_distance(Point3, 0.0, -10.0)*

3.6. Postprocessing and Generated Garment Pattern Model

The postprocessing stage begins with a validation step that systematically verifies the entries in the Pattern Component Database. This process involves checking the syntax and semantics of all function calls, ensuring their arguments and names adhere to predefined rules, such as naming conventions and valid call structures. This validation is applied to each table individually to ensure data integrity.

Following this verification, the postprocessing module resolves the entangled dependency structure within the database. Each drafting step (k^{th}) may reference components defined in earlier steps. To resolve this, the module uses a recursive strategy to identify and order these dependencies. The Postprocessing Unit then converts the final, verified database entries into a structured Python-style garment pattern model. This model encapsulates all necessary geometric and procedural information, including points, lines, constants, function calls, and cutting instructions. The resulting model is a fully self-contained, executable representation of the garment pattern, entirely independent of the LLM and its agent-based generation pipeline. Algorithm 1 provides the pseudocode for the postprocessing to generate a garment pattern model.

4. Implementation Detail

We collect garment drafting literature from Modern Pattern Design [34] and other relevant sources. For our experiments, we use seven types of garment drafting texts: (1) front bodice with a dart, (2) back bodice without darts, (3) arm sleeve, (4) waistband/cuff, (5) two-pieced skirt sloper (front and back), (6) full circular skirt (front and back), and (7) pants (front and back). The drafting texts for skirts and pants describe both front and back panels within a single entry. To account for the stochastic nature of the generation process, we conducted five independent experimental runs for each pattern type.

The GarmentoPIA system is built using LangGraph, with procedural garment pattern modeling powered by the SymPy library [28] in Python. The Pattern Component Database is implemented with SQLite3 for efficient and lightweight data handling. In GarmentoPIA, the Draft2DSL Translator uses the GPT-o4mini API (o4-mini-2025-04-16), while the Component Entity Generator, Component Entity Reviewer, and postprocessing unit leverage the GPT-4.1 API (gpt-4.1-2025-04-14). Our baseline utilizes the GPT-o4mini and GPT-4.1 APIs, chosen for their widespread availability and ease of use. For the garment pattern simulation, we use CLO3D [7].

To demonstrate the framework's versatility and independence from a specific Large Language Model (LLM), we extended our experiments to incorporate Qwen3 Coder Plus

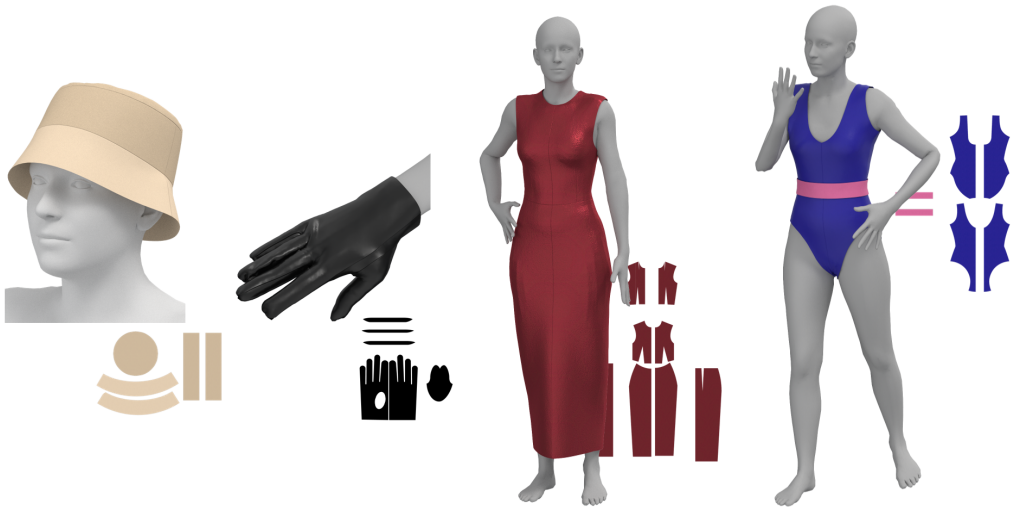

Figure 5. Various patterns produced by GarmentoPIA-generated models from references [8, 34, 24].

API and GPT-5 API as alternative backbone models. The results, detailed in Table 2, compare key performance indicators including success rate and the average time required per garment drafting step (APS).

For the evaluations, garment patterns are rendered from the generated models, a process conditioned on specific body measurements. These measurements are derived from the SMPL-X body model [33], following the procedure detailed in [19]. Each dimension is computed as a geodesic distance between designated start and end vertices, emulating physical measurement with a ruler. In Figure 6, we utilize SMPL-X shapes where only the first shape blend coefficient is set to one, with all others set to zero.

5. Experiment

While several studies have generated garment patterns using existing or custom-defined pattern models, none have addressed the problem of automatically creating garment pattern models themselves. Prior work such as DressCode auto-regressively generates patterns from prompts; however, no method has yet transformed entire drafting textbooks into fully parametric pattern programs, which is the problem we tackles. Therefore, a direct comparison with prior work is not possible. Instead, we evaluate the performance of our system, GarmentoPIA, against a defined baseline. The baseline utilizes an LLM that takes the drafting text, body and garment measurements, and role instructions as prompt input. It also makes use of the SymPy library and Python for computation. The detailed role instruction provided to the baseline LLM is as follows:

```
- Make an SVG format output generator using
#SVG format, #Drafting Text, and #Measurements.
- Create a French Curve based on Bézier curves or
similar methods to produce smooth curves.
- If the number of panels is two or more ...
```

To isolate the contribution of our language design, we introduce GarmentoPIA^{--}, which utilizes our DSL functions within a single-turn prompt, excluding both the multi-agent architecture and the prompt refinement mechanism. This setup verifies whether performance improvements stem from the DSL schema itself or the agentic framework.

All systems generate garment models using the same inputs: body and garment measurements, along with drafting text. We report quantitative performances such as task success rate, error type ratio, and numerical accuracy. We also present qualitative results to demonstrate the quality of the garment models generated by our system.

5.1. Comparing Generation Systems

To ensure the robustness of the system of generating garment models, two primary criteria must be satisfied. First, the integrity of the resulting component database must be verified, specifically by evaluating how accurately the syntax can be represented as executable Python code. Second, it is essential to assess the extent to which the numerical values assigned to the components within the garment model faithfully reflect the information described in the drafting text.

Instances in which the first criterion is not met typically demonstrate as syntax errors. Such errors can be particularly challenging to resolve, as they often require meticulous inspection and repeated attempts to identify the underlying issue. Similarly, numerical errors, which impact the geometry of the generated pattern, present their difficulties. These errors are frequently subtle and may only be detected by the user through careful analysis.

To compare the performance of the GarmentoPIA and baseline systems in handling the DSL and the Python syn-

Table 1. Comparisons of GarmentoPIA and the baseline system. GarmentoPIA^{-} is GarmentoPIA without prompt refinement.

	Task Success Rate			Failure Types		Geometric Accuracy
System	One Panel	Two Panels	All	Syntax Error	Numerical Error	mIoU↑
Baseline	30.0%	0.0%	17.1%	37.1%	45.8%	0.825
GarmentoPIA^{--}	40.0%	20.0%	28.6%	5.7%	65.7%	0.528
GarmentoPIA^{-}	50.0%	33.3%	42.9%	54.2%	2.9%	0.979
GarmentoPIA	90.0%	80.0%	85.7%	4.8%	9.5%	0.975

Table 2. Success Rate of GarmentoPIA with Different LLM Backbones.

	Task Success Rate			Failure Types		Geometric Accuracy	APS
Backbone	One Panel	Two Panels	All	Syntax Error	Numerical Error	mIoU↑	sec↓
GPT-o4mini & GPT-4.1	90.0%	80.0%	85.7%	4.8%	9.5%	0.975	∼224
Qwen3 Coder Plus	70.0%	66.7%	68.6%	31.4%	0.0%	1.000	∼2
GPT-5	84.0%	86.7%	85.0%	15.0%	0.0%	1.000	∼70

tax, we evaluate task success rates between the baseline and GarmentoPIA using body and garment measurements as inputs. Specifically, body measurements are derived from the SMPL-X, and garment measurements are varied according to each pattern type. Task success is defined as the case where the generated pattern SVG matches the manually created reference pattern SVG in all aspects, including one-to-one correspondence between points, identical x and y coordinates, and exactly matching SVG path commands.

Pattern types used in experiment (Section 4) are categorized into two types, One-panel drafting texts, where a single panel (*e.g.*, front bodice) is described, and Two-panel drafting texts, where both front and back panels (*e.g.*, skirts, pants) are described in a single entry. Two-panel texts tend to be more complex, containing more components and drafting steps.

As presented in Table 1, GarmentoPIA^{--} outperforms the standard baseline, validating that our domain-specific language (DSL) is better suited to the garment domain than conventional mathematical libraries. Furthermore, GarmentoPIA^{-} surpasses GarmentoPIA^{--}, suggesting that the proposed foundational structure provides substantial benefits even without prompt refinement. Ultimately, by incorporating prompt refinement, GarmentoPIA achieves superior success rates relative to GarmentoPIA^{-}, particularly in the more challenging task category. These results emphasize that the iterative updates of supporting rules and examples are crucial for facilitating the comprehension and processing of complex drafting instructions.

We categorize the failure cases for each system into syntax and numerical errors, as summarized in Table 1. A high proportion of syntax errors indicates difficulty in using the expected language format. When comparing the baseline with GarmentoPIA^{-}, our system shows a significant reduction in numerical errors. However, the syntax error rate is higher for GarmentoPIA^{-}, suggesting that it struggles more with the custom-defined DSL than the baseline does with standard Python syntax. Nevertheless, our prompt refinement strategy substantially mitigates this issue, reducing the overall error rate to just 4.8%.

To evaluate hidden geometric inaccuracies, we compute the mIoU between the manually annotated reference panels and the corresponding panels generated by each system (Table 1). The results with syntax errors are excluded from this evaluation. The metric quantitatively assesses shape fidelity by measuring numerical deviations in the final garment pattern. Both GarmentoPIA^{-} and GarmentoPIA achieve higher mIoU scores than the baseline.

To verify the system's robustness and LLM-independence, we evaluated GarmentoPIA using backbones from the Qwen3 Coder Plus and GPT-5 (Table 2). The system maintained high success rates across all models, demonstrating that its performance is not tied to a specific LLM. This confirms that our foundational architecture, particularly the domain-specific language (DSL) and prompt refinement strategy, is the principal factor of its success and adaptability.

In summary, the baseline performs the worst, often failing due to missing points or incorrect values, such as syntax errors, skipping steps, or producing reversed geometries. GarmentoPIA^{-}, which disables prompt refinement, seldom assigns incorrect values to the DSL but still suffers from missing points. In addition, GarmentoPIA^{-} identifies issues such as incorrect table assignments and argument mismatches in function calls. Our full system, GarmentoPIA, achieves the highest success rate, resolving these issues through prompt refinement, although it sometimes misses points. Thus, the system's strong performance and its confirmed independence from the underlying LLM backbone validate the robustness of our overall approach.

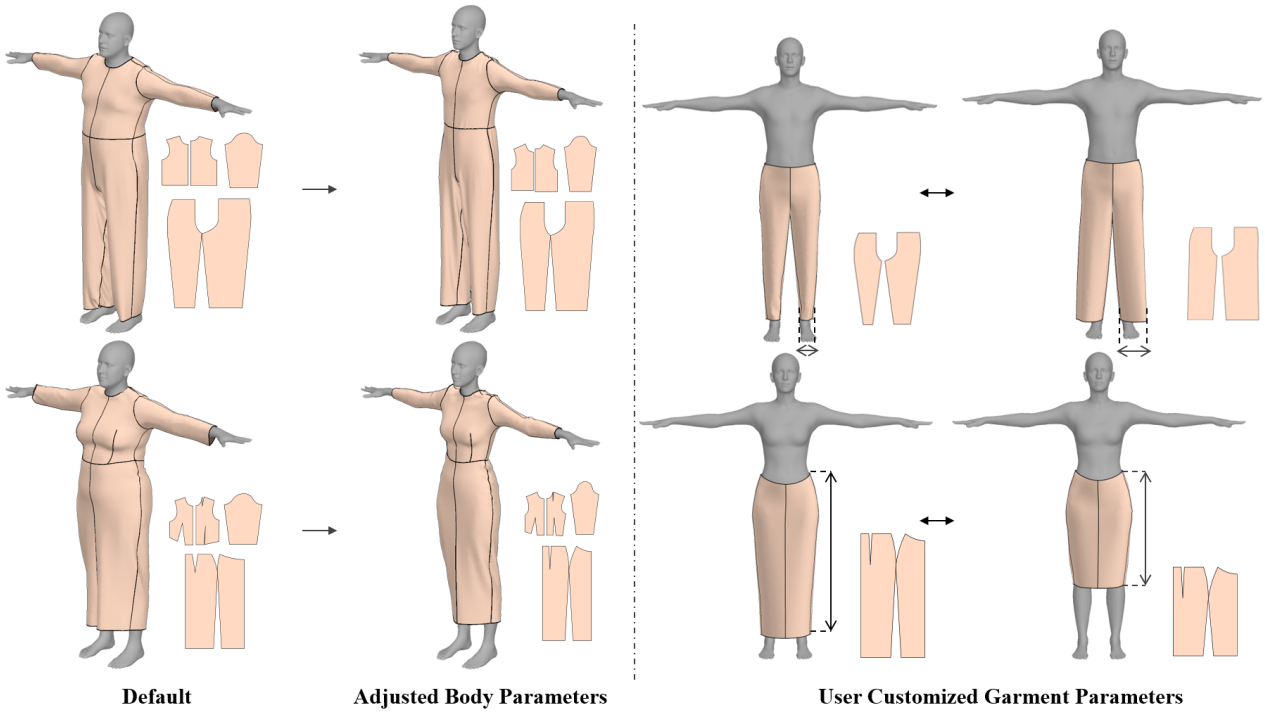

Figure 6. Garment patterns generated using parametric models created by GarmentoPIA. These models take body and garment measurements as input parameters to produce pattern panels for individual body shapes and garment styles.

5.2. Adjusting Body and Garment Measurements on Pattern Generation

We demonstrate the capability of garment pattern models produced by GarmentoPIA to automatically generate garment patterns that adapt to input body measurements. User can also customize garment styles by adjusting specific garment measurements. As shown in Figure 6, the process begins with the SMPL-X template body mesh, where we identify specific vertices corresponding to required body measurements. For each measurement, we calculate the value using either geodesic or Euclidean distance, selecting the metric most suitable for the specific anatomical dimension [34]. To personalize the pattern, we employ an existing method [33] to estimate a body mesh directly from an input image. Since the mesh topology remains invariant despite the shift in vertex positions to represent the estimated body shape, we compute the final dimensions using the same predefined vertex indices on the deformed mesh. Once the garment measurements are adjusted, the garment pattern model generates patterns that reflect both the input body shape and garment specifications.

6. Application

The generated parametric garment model acts as a template to guide existing garment prediction methods in producing the final pattern panels. Subsequently, these panels, which form the garment in the source image, are simulated on a digital human model (Figure 7). Figure 5 is the result of generated patterns from preprocessed diverse garment drafting literature using GarmentoPIA.

Furthermore, the parametric models from GarmentoPIA can be integrated into a pipeline driven by a Large Language Model (LLM) for direct pattern generation from a single image. In this approach, the LLM analyzes the input image to estimate the user's body shape and to identify the garment type. Following the identification, an appropriate parametric garment model is selected from LLM. By inputting body measurements and adjusting garment-specific parameters, 2D garment patterns can be generated based on the prompt refinement.

7. Discussion

While GarmentoPIA demonstrates strong performance in generating garment pattern models from drafting texts, it also presents certain limitations. First, we found that current LLMs cannot accurately interpret garment drafting books, which contain ambiguities and omission of context, as well as cross-referencing across different garment patterns, that could only be understood by people with drafting knowledge. As such, we converted the literature to a

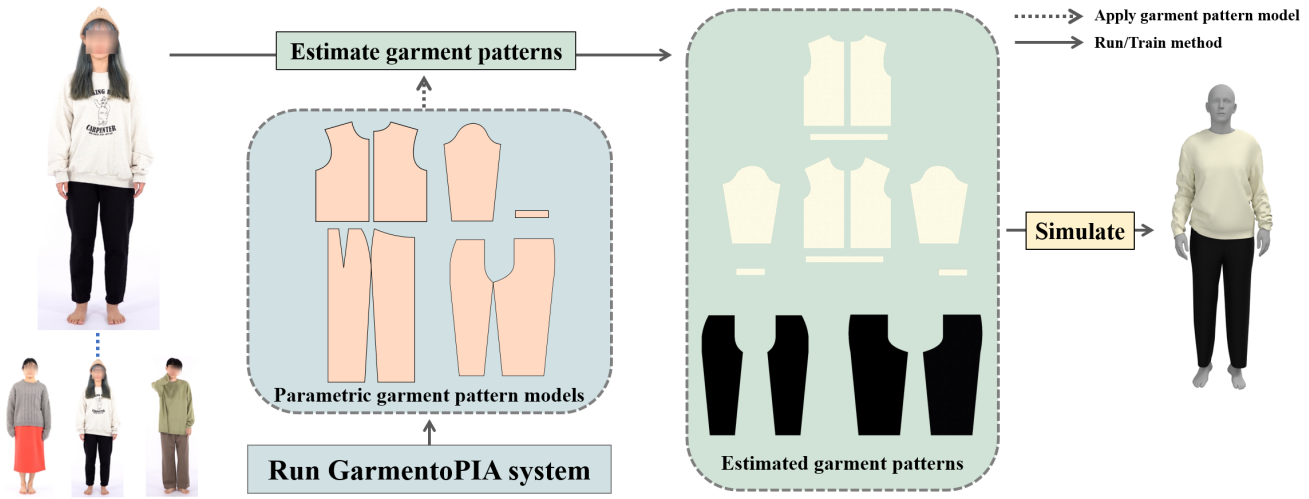

Figure 7. The GarmentoPIA system generates parametric models that enable garment estimation frameworks to estimate 2D pattern panels.

more LLM-comprehensible form in the preprocessing step. Ultimately, the advancement of LLM techniques may overcome this limitation, but in the meantime, future work could develop a semi-automatic translator that makes the preprocessing step more efficient.

Second, the output variability arises during the initial phase of pattern model generation. The stochastic nature of the integrated LLM can introduce foundational inconsistencies at this early stage, which may then propagate through subsequent processes. To enhance the pipeline's end-to-end reliability, future iterations could focus on implementing a deterministic generation source. This would likely involve deploying a self-hosted model where sampling parameters, such as temperature and a fixed random seed, can be explicitly controlled, ensuring consistent and reproducible outputs from the outset.

Third, GarmentoPIA relies on prompt refinement through the accumulation of instruction rules and examples. As this prompt content grows in size to cover more garment types and variations, it begins to approach the token limit of the LLM. This limitation can impact performance, particularly when processing drafting instructions with a large number of sequential steps. In such cases, the system may exhibit occasional errors or omissions. To address these limitations, future work will explore hybrid architectures that integrate symbolic computation and procedural logic alongside LLMs, reducing dependency on large prompts. We also plan to investigate prompt compression strategies and memory-augmented models to handle longer drafting sequences more robustly. Additionally, expanding the system's ability to learn from multimodal data—combining text, diagrams, and examples—could improve its generalization to diverse drafting styles and traditions.

8. Conclusion

In this paper, we presented GarmentoPIA, a novel framework that generates executable garment patterns from textual drafting instructions. Our evaluations demonstrated that GarmentoPIA significantly outperforms baselines by substantially reducing syntax and numerical errors. This success stems from our core contributions: a domain-specific language (DSL) and a self-refining prompt process that systematically translates ambiguous instructions into a formalized program with high geometric fidelity. We validated the framework's robustness and LLM-independence, confirming our architecture is key to its high performance. The resulting executable program enables flexible pattern generation for new measurements, free from the cost and latency of commercial LLM services. By converting traditional knowledge into a reusable digital asset, GarmentoPIA marks a significant step toward a more accessible, automated, and personalized future in fashion design.

Acknowledgement

This work was supported by NRF, MSIT, Korea (RS-2025-23524842) and KOCCA, MCST, Korea (RS-2025-02307327).

References

[1] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 1

[2] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection, 2023. 4

[3] S. Bang, M. Korosteleva, and S.-H. Lee. Estimating garment patterns from static scan data. In *Computer Graphics Forum*, volume 40, pages 273–287. Wiley Online Library, 2021. 2

[4] S. Bian, C. Xu, Y. Xiu, A. Grigorev, Z. Liu, C. Lu, M. J. Black, and Y. Feng. Chatgarment: Garment estimation, generation and editing via large language models. *arXiv preprint arXiv:2412.17811*, 2024. 3

[5] L. Chen, J. Yang, H. Fu, X. Meng, W. Chen, B. Yang, and L. Gao. Implicitpca: Implicitly-proxied parametric encoding for collision-aware garment reconstruction. *Graphical Models*, 129:101195, 2023. 2

[6] X. Chen, B. Zhou, F. Lu, L. Wang, L. Bi, and P. Tan. Garment modeling with a depth camera. *ACM Transactions on Graphics (TOG)*, 34(6):1–12, 2015. 2

[7] CLO Virtual Fashion, Inc. Clo 3d, 2025. 8

[8] DRCOS. Gloves sewing patterns — drcos patterns & how to make, 2025. Accessed: 2025-06-07. 9

[9] J. Guo, J. Chen, W. Chen, Z. Sun, L. Li, B. Zhao, L. Zhu, X. Wang, and Q. Liu. Garmentx: Autoregressive parametric representations for high-fidelity 3d garment generation. *arXiv preprint arXiv:2504.20409*, 2025. 4

[10] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M. Chang. REALM: retrieval-augmented language model pre-training. *CoRR*, abs/2002.08909, 2020. 4

[11] N. Houska, C. Lau, and M. Specht. Recompose grammars for procedural architecture. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–9, 2024. 4

[12] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022. 1

[13] G. Izacard and E. Grave. Distilling knowledge from reader to retriever for question answering, 2020. 4

[14] G. Izacard and E. Grave. Leveraging passage retrieval with generative models for open domain question answering, 2020. 4

[15] G. Izacard, F. Petroni, L. Hosseini, N. D. Cao, S. Riedel, and E. Grave. A memory efficient baseline for open domain question answering, 2020. 4

[16] B. Jiang, J. Zhang, Y. Hong, J. Luo, L. Liu, and H. Bao. Bcnet: Learning body and cloth shape from a single image. In *European Conference on Computer Vision*, pages 18–35. Springer, 2020. 2

[17] V. Karpukhin, B. Oguz, S. Min, L. Wu, S. Edunov, D. Chen, and W. Yih. Dense passage retrieval for open-domain question answering. *CoRR*, abs/2004.04906, 2020. 4

[18] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa. Large language models are zero-shot reasoners, 2023. 4

[19] M. Korosteleva, T. L. Kesdogan, F. Kemper, S. Wenninger, J. Koller, Y. Zhang, M. Botsch, and O. Sorkine-Hornung. Garmentcodedata: A dataset of 3d made-to-measure garments with sewing patterns. In *European Conference on Computer Vision*, pages 110–127. Springer, 2024. 1, 3, 9

[20] M. Korosteleva and S.-H. Lee. Generating datasets of 3d garments with sewing patterns. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021. 1, 2, 3

[21] M. Korosteleva and S.-H. Lee. Neuraltailor: Reconstructing sewing pattern structures from 3d point clouds of garments. *ACM Transactions on Graphics (TOG)*, 41(4):1–16, 2022. 3

[22] M. Korosteleva and O. Sorkine-Hornung. Garmentcode: Programming parametric sewing patterns. *ACM Transactions on Graphics (TOG)*, 42(6):1–15, 2023. 1, 3, 4

[23] J. J. Lee, B. Li, and B. Benes. Latent l-systems: Transformer-based tree generator. *ACM Transactions on Graphics*, 43(1):1–16, 2023. 4

[24] N. Lee and MoodFabrics. The oahu swim suit free sewing pattern, May 22 2025. Accessed: 2025-06-07. 9

[25] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020. 3

[26] X. Li, C. Yu, W. Du, Y. Jiang, T. Xie, Y. Chen, Y. Yang, and C. Jiang. Dress-1-to-3: Single image to simulation-ready 3d outfit with diffusion prior and differentiable physics, 2025. 3

[27] L. Liu, X. Xu, Z. Lin, J. Liang, and S. Yan. Towards garment sewing pattern reconstruction from a single image. *ACM Transactions on Graphics (TOG)*, 42(6):1–15, 2023. 3

[28] A. Meurer, C. P. Smith, M. Paprocki, O. Čertík, S. B. Kirpichev, M. Rocklin, A. Kumar, S. Ivanov, J. K. Moore, S. Singh, et al. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, 2017. 6, 8

[29] D. Morelli, M. Fincato, M. Cornia, F. Landi, F. Cesari, and R. Cucchiara. Dress code: High-resolution multi-category virtual try-on. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2231–2235, 2022. 3

[30] K. Nakayama, J. Ackermann, T. L. Kesdogan, Y. Zheng, M. Korosteleva, O. Sorkine-Hornung, L. J. Guibas, G. Yang, and G. Wetzstein. Aipparel: A multimodal foundation model for digital garments. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 8138–8149, 2025. 3

[31] T. Olausson, A. Gu, B. Lipkin, C. Zhang, A. Solar-Lezama, J. Tenenbaum, and R. Levy. Linc: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5153–5176, 2023. 3

[32] L. Pan, A. Albalak, X. Wang, and W. Wang. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3806–3824, 2023. 3

[33] G. Pavlakos, V. Choutas, N. Ghorbani, T. Bolkart, A. A. Osman, D. Tzionas, and M. J. Black. Expressive body capture: 3d hands, face, and body from a single image. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10975–10985, 2019. 9, 11

[34] H. Pepin. *Modern Pattern Design: The Complete Guide to the Creation of Patterns as a Means of Designing Smart Wearing Apparel*. Funk & Wagnalls Company, 1942. 2, 3, 5, 8, 9, 11

[35] A. Plocharski, J. Swidzinski, J. Porter-Sobieraj, and P. Musialski. Façaid: A transformer model for neuro-symbolic facade reconstruction. In *SIGGRAPH Asia 2024 Conference Papers*, pages 1–11, 2024. 4

[36] Z. Rackauckas. Rag-fusion: A new take on retrieval augmented generation. *International Journal on Natural Language Computing*, 13(1):37–47, Feb. 2024. 4

[37] S. Saito, Z. Huang, R. Natsume, S. Morishima, A. Kanazawa, and H. Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2304–2314, 2019. 2

[38] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023. 1, 3

[39] C. Spiess, M. Vaziri, L. Mandel, and M. Hirzel. Autopdl: Automatic prompt optimization for llm agents. *arXiv preprint arXiv:2504.04365*, 2025. 4

[40] A. Srivastava, P. Manu, A. Raj, V. Jampani, and A. Sharma. Wordrobe: Text-guided generation of textured 3d garments. In *European Conference on Computer Vision*, pages 458–475. Springer, 2024. 3

[41] Y. Tatsukawa, A. Qi, I.-C. Shen, and T. Igarashi. Garmentimage: Raster encoding of garment sewing patterns with diverse topologies. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, pages 1–11, 2025. 3

[42] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. H. Chi, Q. Le, and D. Zhou. Chain of thought prompting elicits reasoning in large language models. *CoRR*, abs/2201.11903, 2022. 4

[43] D. Wu, J. Zhang, and X. Huang. Chain of thought prompting elicits knowledge augmentation, 2023. 4

[44] Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023. 3

[45] Z. Xiong, D. Du, Y. Wu, J. Dong, D. Kang, L. Bao, and X. Han. Pifu for the real world: A self-supervised framework to reconstruct dressed human from single-view images. In *International Conference on Computational Visual Media*, pages 3–23. Springer, 2024. 2

[46] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023. 3

[47] Z. Zhang, A. Zhang, M. Li, and A. Smola. Automatic chain of thought prompting in large language models, 2022. 4

[48] F. Zhou, R. Liu, C. Liu, G. He, Y.-L. Li, X. Jin, and H. Wang. Design2garmentcode: Turning design concepts to tangible garments through program synthesis. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 23712–23722, 2025. 3