

**Universidad
Rey Juan Carlos**

Escuela Técnica Superior
de Ingeniería Informática

**Máster Universitario en Informática Gráfica, Juegos y
Realidad Virtual**

Curso 2023-2024

Trabajo Fin de Máster

EDICIÓN PARAMÉTRICA DE PATRONES

**Autora: Marta Quintana Portales
Tutor: Jorge Félix López Moreno**

Agradecimientos

Quiero empezar este trabajo dando las gracias a todas las personas que me han acompañado a lo largo de esta aventura y celebrar con ellos que este es el primer paso para poder dedicarme a lo que me gusta.

Gracias a Jorge López Moreno, mi tutor, por el apoyo, las oportunidades y la confianza en mí para la realización de este TFM.

A todo el equipo de SEDDI y diseñadores, en especial a Alejandro Graciano, Harrison Johnson, Iván Alduan y Alex Rodríguez, gracias por vuestros conocimientos, dedicación y disponibilidad.

A los profesores y compañeros que me han enseñado e inspirado en este máster.

A mi familia y amigos, gracias por acompañarme y apoyarme incondicionalmente, en especial a mi hermana. Recuerdo aquel día en que llegué bastante tarde a casa después de un largo día de clases y trabajo, y me encontré una frase escrita en la pizarra de mi habitación. Desde entonces, ha sido mi mantra y, cada vez que pensaba que no podía, esta frase me ha motivado a seguir adelante.

Gracias.

“¡Vamos Marta! Tú puedes con todo lo que te propongas :)"

Resumen

En las últimas décadas, la industria de la moda ha generado un impacto ambiental negativo bastante significativo debido a prácticas insostenibles en la producción y el consumo desmedido de ropa. La moda digital ha surgido como una solución para abordar este problema, permitiendo la creación de prototipos de diseño sin necesidad de desperdiciar materiales físicos. Además, facilita el escalado y ajuste de patrones, reduciendo de manera considerable los residuos y el consumo de tiempo y recursos.

Este trabajo se enmarca en el proyecto de Colaboración Público-Privada TAI-LOR, que busca mejorar las herramientas del *software* de SEDDI Inc para facilitar a los usuarios el cosido y el escalado de prendas digitales. El objetivo principal de este trabajo es desarrollar una herramienta de edición paramétrica que permita la creación y modificación de patrones 2D y 3D con puntos de medida, además de investigar la preservación del estilo de las prendas entre avatares con diferentes medidas antropomórficas y contribuir al avance hacia una industria de la moda más sostenible.

Se diseñó y desarrolló una herramienta de creación y edición de puntos de medida en el *software* de SEDDI Author, permitiendo crear y editar estos puntos de medida dinámicamente. Esta herramienta facilita mediciones para controlar el ajuste y realizar modificaciones basadas en distancias, considerando la orientación del punto de medida y las conexiones entre piezas. Además, se realizó una optimización basada en la deformación de la longitud de las aristas de la malla 3D de la prenda teniendo en cuenta dos puntos de medida, uno horizontal y otro vertical, para preservar el ajuste de la prenda entre diferentes avatares. Los resultados cumplen con las expectativas a nivel local, sin embargo, los parámetros utilizados no son suficientes para hacer un escalado completo de la prenda y se sugiere mejorar este método en futuros trabajos aplicando *grading rules* a las áreas que no dependen de esos dos puntos de medida, como el cuello, los hombros o las mangas.

Este trabajo muestra el potencial de las herramientas digitales, especialmente la edición paramétrica, en la industria de la moda, contribuyendo a la evolución del diseño de moda digital y sirviendo de base para futuras investigaciones.

Palabras clave: Puntos de medida (POM), Escalado de patrones, *Grading rules*, Moda digital, Edición paramétrica, SEDDI Author.

Abstract

In recent decades, the fashion industry has generated a quite significant negative environmental impact due to unsustainable production practices and excessive clothing consumption. Digital fashion has emerged as a solution to address this issue, enabling the creation of design prototypes without the need to waste physical materials. Additionally, it facilitates the scaling and adjustment of patterns, significantly reducing waste, time, and resource consumption.

This work is part of the TaiLOR collaboration project, aimed at enhancing SEDDI Inc software tools to help digital garment sewing and scaling for users. The main objective of this study is to develop a parametric editing tool enabling creation and modification of 2D and 3D patterns with measurement points, alongside investigating garment style preservation across avatars with varying anthropometric measurements, contributing to advancing towards a more sustainable fashion industry.

A tool for creating and editing measurement points was designed and developed within SEDDI Author software, enabling dynamic points of measuser creation and editing. This tool facilitates measurements to control fit and make adjustments based on distances, considering measurement point orientation and piece connections. Additionally, an optimization was performed based on deformation of 3D garment mesh edge lengths between two measurement points, one horizontal and one vertical, to preserve garment fit across different avatars. The results meet local expectations; however, the parameters used are not sufficient for a complete scaling of the garment, and it is suggested to improve this method in future works by applying grading rules to areas that do not depend on these two measurement points, such as the neck, shoulders, or sleeves.

This work demonstrates the potential of digital tools, especially parametric editing, in the fashion industry, contributing to the evolution of digital fashion design and providing a foundation for future research.

Keywords: Measurement points (POM), Pattern grading, Grading rules, Digital fashion, Parametric editing, SEDDI Author.

Índice de contenidos

Índice de tablas	7
Índice de figuras	10
Índice de códigos	11
1. Introducción	12
1.1. Contexto	12
1.1.1. Moda y escalado de patrones	12
1.1.2. Moda Digital	15
1.1.3. Proyecto de colaboración <i>TaiLOR</i>	17
1.2. Objetivos	19
1.3. Requisitos	19
1.4. Metodología	20
1.5. Plan de Trabajo	22
1.6. Estructura del documento	23
2. Estado del arte	24
2.1. <i>Software</i> de diseño de moda y patronaje	24
2.2. Edición paramétrica	26
2.3. Puntos de medida: POM	29
2.4. Preservación del estilo de la prenda	31
2.5. SEDDI	32
3. Análisis y Desarrollo	35
3.1. Curvas de Bézier	36
3.2. Análisis del diseño de la herramienta de edición de patrones en SEDDI Author	38
3.3. Desarrollo de la herramienta de creación de puntos de medida	40
3.3.1. <i>POM TOOL</i>	42
3.3.2. <i>onUpdate</i>	44
3.3.3. <i>addPOMInteriorLine</i> y <i>addPointOfMeasure</i>	46

3.4. Desarrollo de la herramienta de edición paramétrica de patrones en SEDDI Author	49
3.4.1. <i>Parametric Editing Tool</i>	53
3.4.2. Conectividad entre patrones	56
3.5. Problema de optimización: preservación del estilo de la prenda entre avatares	58
3.5.1. Exportación e Importación de datos de la escena	60
3.5.2. Escalado proporcional en 3D	63
3.5.3. Relación entre los puntos de medida y los vértices 2D de la prenda	65
3.5.4. Función de coste y optimización	68
4. Resultados	70
4.1. Tipos de puntos de medida (POM)	70
4.2. Posibles ediciones paramétricas	72
4.3. Preservación del estilo con puntos de medida entre diferentes avatares	81
5. Conclusiones y trabajos futuros	87
5.1. Conclusiones	87
5.2. Trabajos futuros	89
Bibliografía	90

Índice de tablas

4.1. Nombre avatares	82
4.2. Resultados esperados en relación a la referencia <i>WOMAN BASE</i> .	82
4.3. Resultados obtenidos en cm en relación a la referencia <i>WOMAN BASE</i>	84

Índice de figuras

1.1.	Escalado camiseta [1]	14
1.2.	Escalado pantalón [2]	14
1.3.	<i>Software Lectra</i> [3]	16
1.4.	Probador virtual (AR) [4]	16
1.5.	Metaverso DressX [5]	16
1.6.	Proyecto Primrose Adobe [6]	16
1.7.	Gestión de tareas en <i>Trello</i>	20
1.8.	Página web de este TFM	21
1.9.	Diagrama de Gantt	23
2.1.	Adobe Illustrator [7]	25
2.2.	Cameo [8]	25
2.3.	Optitex [9]	25
2.4.	Browzwear [10]	25
2.5.	Configuración de diseño y ajuste de una camiseta en Cameo [11] .	26
2.6.	Posibles ediciones camiseta base del patrón paramétrico en Clo3D	27
2.7.	Posibles ediciones mangas base patrón paramétrico en Clo3D . .	27
2.8.	Edición paramétrica de patrón 2D en Clo3D	28
2.9.	Edición de avatar paramétrico en Browzwear	28
2.10.	Ejemplo de <i>grading rules</i> de una camiseta [12]	29
2.11.	Puntos de medida en una camiseta [13]	30
2.12.	Puntos de medida en un pantalón [13]	30
2.13.	Resultados del artículo donde se puede ver la transferencia de la prenda, preservando el diseño original entre avatares con distintas medidas antropomórficas. Fuente: Laurence Boissieux 2012 . . .	31
2.14.	Material escaneado y digitalizado en SEDDI Textura junto a un patrón diseñado y cosido en SEDDI Author [14]	32
2.15.	Diagrama del flujo de datos en SEDDI AUTHOR	33
2.16.	Objeto <i>Store</i> del estado de <i>Redux</i> en la consola de la aplicación con los datos actualizados	34
3.1.	Curvas de Bézier de distintos órdenes con sus puntos de control [15]	37

3.2.	Patrón camiseta base definido con curvas continuas de Bézier, cada pieza esta formada por un <i>path</i> de Bézier, representado con Python	37
3.3.	Flujo de trabajo del diseño de la herramienta de generación, visualización y edición de puntos de medida	38
3.4.	Flujo de detalles: Herramienta de creación de puntos de medida	39
3.5.	Diseño de la interfaz de usuario propuesto por los diseñadores para la herramienta de creación de POMs y edición paramétrica	39
3.6.	<i>ParametricPanel</i>	41
3.7.	Ejemplo creación de un punto de medida horizontal	48
3.8.	Punto de medida en la interfaz de usuario	49
3.9.	Esquema del flujo del problema de optimización	59
3.10.	Escalado proporcional de la prenda del avatar de referencia al avatar <i>MAN BASE</i> y <i>BIG WOMAN</i>	64
3.11.	Izquierda: intersección horizontal de un vértice y sus vértices correspondientes del borde más cercanos. Derecha: zoom en la intersección donde se indican los pesos w ₀ , w ₁ , w ₂ , w ₃ de cada vecino en el vértice P.	66
3.12.	Izquierda: intersección vertical de un vértice y sus vértices correspondientes del borde más cercanos. Derecha: zoom en la intersección donde se indican los pesos w ₀ , w ₁ , w ₂ , w ₃ de cada vecino en el vértice P.	67
3.13.	Modificación de los vértices 2D de la prenda en función del punto de medida	67
3.14.	Valor del POM optimizado y evolución del error durante la optimización	68
4.1.	Punto de medida horizontal	71
4.2.	Punto de medida vertical	71
4.3.	Punto de medida diagonal	71
4.4.	Posibles puntos de medida horizontales	72
4.5.	Posibles puntos de medida verticales	72
4.6.	Posibles puntos de medida diagonales más comunes	73
4.7.	Ejemplos de otros posibles puntos de medida diagonales	73
4.8.	Puntos de medida que suelen usarse para este tipo de camiseta	74
4.9.	Puntos de medida horizontales modificados de forma paramétrica	74
4.10.	Puntos de medida horizontales en la manga y cuello modificados de forma paramétrica	75
4.11.	Problemas de edición paramétrica con mangas definidas de diferentes formas	76
4.12.	Solución: redefinir tapa de la manga en solo 2 segmentos con la misma longitud	76
4.13.	Puntos de medida verticales modificados paramétricamente	77

4.14. Puntos de medida verticales en la manga y cuello modificados paramétricamente	77
4.15. Puntos de medida diagonales en la apertura manga modificados de forma paramétrica	77
4.16. Ejemplo conexiones horizontales	78
4.17. Ejemplo conexiones verticales	78
4.18. Ejemplo conexiones diagonales	78
4.19. Edición paramétrica en un vestido	79
4.20. Importación patrón de un pantalón	80
4.21. Edición paramétrica de un pantalón	80
4.22. <i>Womens US ASTM Sz M/6 - WOMAN BASE</i> Medidas: 50x60cm	81
4.23. Puntos de medida que queremos calcular para los patrones de los nuevos avatares, POM vertical en rojo y POM horizontal en amarillo	82
4.24. Camiseta 50x60cm para todos los avatares	83
4.25. Resultados patrones ajustados con las medidas correspondientes .	84
4.26. Camisetas con las medidas ajustadas a cada avatar para preservar el ajuste	85

Índice de códigos

3.1.	Clase ParametricPatternProperties	40
3.2.	PomTool	42
3.3.	OnUpdate PomTool	45
3.4.	addPOMInteriorLine	46
3.5.	Acción addPointOfMeasure y su reducer	47
3.6.	Puntos de medida en el estado global	48
3.7.	SliderComponent	50
3.8.	Acción addPointOfMeasureProperty y su reducer	51
3.9.	updatePOMInteriorLines y updatePOMInteriorLine	52
3.10.	ParametricEditingTool	53
3.11.	ParametricEditing	54
3.12.	calcularTransformación y getTransformFn	55
3.13.	ParametricEditing	57
3.14.	Parametric Panel: handleExport y handleImport	60
3.15.	JSON exportado	62
3.16.	Datos procesados en Python	62
3.17.	Calcular distancias mínimas y correspondencias de avatar	63
3.18.	Calcular nuevas posiciones de la prenda	63
3.19.	Métrica de deformación: longitud de las aristas	64
3.20.	Ejemplo de agrandar un POM 7cm	65
3.21.	Función de coste	69

1

Introducción

Este Trabajo Fin de Máster se sitúa dentro del ámbito del diseño asistido por computador. El avance de la tecnología ha abierto un nuevo campo de investigación en la industria textil, permitiendo la creación de soluciones innovadoras y desempeñando un papel cada vez más importante en el proceso creativo, cambiando la manera en que experimentamos y consumimos la moda.

En este primer capítulo se realiza una breve introducción a la industria de la moda y la moda digital, así como al escalado de patrones. Se detalla el proyecto de colaboración *TaiLOR*, que servirá como base para comprender el contexto en el que se desarrolla este trabajo. A continuación, se presentan los objetivos y requisitos del proyecto, seguidos de una descripción de la metodología y el plan de trabajo. Finalmente, se expone la estructura del documento para guiar al lector a través de sus diferentes secciones.

1.1. Contexto

1.1.1. Moda y escalado de patrones

La moda es una expresión cultural y económica que juega un papel fundamental en la sociedad. Va más allá de ser simplemente prendas de vestir; lo que llevamos puesto nos resguarda y protege, pero también representa tendencias y estilos que reflejan quiénes somos o quiénes queremos llegar a ser [16].

En los últimos años, la moda rápida o *fast fashion* ha llevado a un aumento significativo en el consumo de ropa. Impulsado por la globalización y el poder de las redes sociales, este fenómeno está alimentando un consumismo desmedido, así como una sobreproducción y desperdicio innecesarios en la industria textil. Las plataformas como Instagram, TikTok y Pinterest han permitido que las marcas lleguen directamente a los consumidores, creando un sentido de urgencia para seguir las tendencias y comprar constantemente nuevos productos. Este modelo ha disparado la cantidad de ropa que se produce, pero también la que se desecha, generando una serie de problemas ambientales, sociales y éticos bastante graves.

Para fabricar una camiseta de algodón se necesitan 2.700 litros de agua, la cantidad que una persona bebe en dos años y medio [17]. En un estudio de la Unión Europea de 2020, se habla de que se emplearon 9 metros cúbicos de agua, 400 metros cuadrados de tierra y 391 kg de materias primas por cada ciudadano en ropa y calzado [17]. La industria textil es responsable del 20 % de la contaminación mundial del agua potable. Los tintes y la emisión de micro-plásticos por las microfibras que componen los tejidos, son liberadas sobre todo en los primeros lavados y terminan en el mar [17].

La recolección y reciclaje de ropa usada sigue siendo bajo, solo el 1 % se utiliza para fabricar ropa nueva, lo que contribuye a la acumulación de residuos en vertederos e incineración masiva. Todo este proceso desde la fabricación hasta que se desecha, es responsable del 10 % de las emisiones globales de carbono [17].

Para 2030, aunque todavía está en proceso de negociación, Europa aspira a que todos los textiles sean duraderos y reciclables, fabricados en gran medida con fibras recicladas, libres de sustancias peligrosas y producidos con respeto a los derechos sociales y al medio ambiente [18]. Para 2050 se pretende lograr una economía circular en este sector [17].

Una iniciativa clave es orientar el comportamiento de los consumidores hacia una moda más lenta, promoviendo opciones más sostenibles y duraderas. Las nuevas estrategias para hacer frente a este problema van desde desarrollar nuevos modelos de negocio de alquiler de la ropa, venta de ropa de segunda mano, para reutilizar y reciclar lo máximo posible, hasta diseñar los productos de la forma más eficiente posible con el uso de la tecnología.

Una parte importante de este impacto ambiental se produce durante la fase de diseño y fabricación. Para diseñar una prenda se crea un patrón base adaptado perfectamente a una persona o un maniquí con el estilo que se quiera seguir. Sobre esa referencia habitualmente se desarrollan todas las tallas, lo que comúnmente se llama escalado de patrones o *pattern grading*. Este proceso consiste en crecer o decrecer el patrón en los puntos precisos (generalmente las costuras) unos milímetros o centímetros por cada talla. Los patronistas profesionales adaptan las prendas a personas con diferentes proporciones y tipos de cuerpo, donde manualmente se ajusta el patrón para que el diseño de la prenda sea el mismo en las

diferentes tallas como podemos ver en las Figuras 1.1 y 1.2. Aunque existen tablas y medidas, cada fabricante realiza su propio tallaje y el escalado dependerá del diseño particular de cada prenda. Este proceso es iterativo y requiere de ajustes continuos hasta conseguir el diseño deseado para cada una de las tallas.

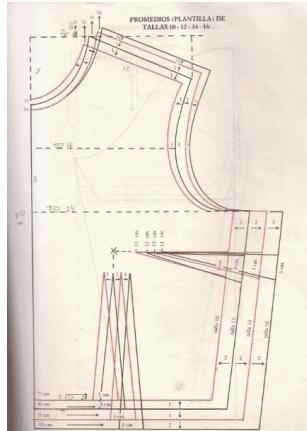


Figura 1.1: Escalado camiseta [1]

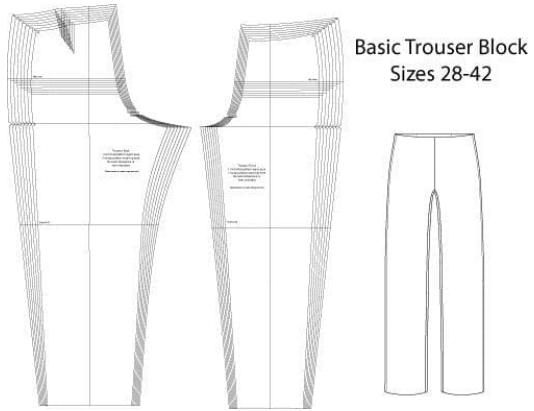


Figura 1.2: Escalado pantalón [2]

En este proceso de escalado se generan muchos descartes y materiales (papel de patrón, piezas escaladas incorrectamente, retales y recortes de tejido) que van directamente a los vertederos. En esta parte de la fabricación es donde la tecnología puede tener un papel fundamental con la utilización de *software CAD*. El *software CAD*, o Diseño Asistido por Computadora, es una herramienta que permite crear modelos y diseños en 2D y 3D de manera digital. Idealmente, con dicho *software*, los diseñadores pueden visualizar de manera realista como va a quedar la prenda, tomar decisiones de diseño más informadas y realizar las iteraciones que harían manualmente de manera digital para luego solamente fabricar las piezas del prototipo final (o incluso evitarlo completamente). De esta forma se puede optimizar el uso de materiales, reducir el desperdicio y mitigar la contaminación asociada con este proceso.

Actualmente, el modelado de prendas de ropa virtuales es una tarea que requiere mucho tiempo y conocimiento del que normalmente carecen los diseñadores y patronistas. Pero cada vez existen más cursos y formaciones en este ámbito para formar profesionales en este sector. Se espera que la incorporación de inteligencia artificial y aprendizaje automático en estos programas facilite a los diseñadores las tareas de diseño, cosido y modelado, así como una mayor automatización y optimización en la producción [19].

Como hemos comentado, la tecnología puede ser un factor clave para el diseño sostenible, para optimizar la producción, así como para la transparencia y trazabilidad de los materiales. La moda digital también está emergiendo con fuerza, permitiendo la creación de prendas virtuales y experiencias de compra *online* más sostenibles e interactivas.

1.1.2. Moda Digital

La moda digital se refiere a la integración de la tecnología en diversos aspectos de la industria de la moda. Podemos agruparlos en cuatro tendencias:

1. Diseño y creación de prototipos digitales.
2. Promoción y negocios *online*.
3. Humano digital y metaverso.
4. Ropa y accesorios con tecnología electrónica inteligente.

Para crear diseños y prototipos digitales se utilizan *software CAD*. Podemos distinguir dos tipos: los *software CAD* 2D que sirven para ilustrar los diseños, visualizar el corte de patrones y crear marcadores, y los *software 3D* que sirven para simular virtualmente las prendas y analizar el ajuste. Existen *software* que combinan ambos, como Lectra, que podemos ver en la Figura 1.3 y que comentaremos más adelante. Estos programas también utilizan algoritmos de inteligencia artificial para mejorar los ajustes, diseños y prototipos.

En cuanto a la promoción y los negocios digitales, podemos hablar del uso de ropa digital en campañas de *marketing* que aparecen en los anuncios o en las redes sociales, cuentan con efectos de realidad aumentada o modelos virtuales para presentaciones y catálogos digitales. El comercio *online* ha experimentado un gran crecimiento, aunque la incorrecta selección de talla y ajuste representa el 70 % de las devoluciones en compras *online* debido a la falta de estandarización en las tallas. Para incentivar las ventas *online*, muchas empresas ofrecen devoluciones gratuitas, pero esto conlleva gastos adicionales por gestión, logística y mantenimiento de *stock* [20]. Para abordar estos problemas, existen diferentes soluciones, como probadores virtuales, sistemas de recomendación de talla basados en tablas de características antropomórficas y medidas corporales. La realidad aumentada (AR) y la tecnología de prueba virtual permiten a los clientes visualizar cómo se verán y quedarán las prendas antes de comprarlas, como podemos ver en la Figura 1.4, lo que reduce las devoluciones y aumenta la satisfacción del cliente. Además, se están utilizando algoritmos de inteligencia artificial (IA) y aprendizaje automático para analizar grandes datos y gestionar el ciclo de vida del producto, así como para personalizar la experiencia de compra y recomendar productos adaptados al estilo y las preferencias individuales de cada cliente [20].

Por otro lado se encuentran los humanos digitales y el metaverso. Se utilizan técnicas de escaneo corporal 3D para crear avatares virtuales para usarlos en animaciones, para probar virtualmente diferentes prendas y accesorios en esos avatares, para representar al jugador en los videojuegos o en desfiles de moda virtuales. En esta sección podemos incluir la ropa digital que se diseña y vende exclusivamente para customizar tu avatar virtual en el metaverso, como los *skins*

en experiencias de realidad virtual o en videojuegos, podemos ver un ejemplo en la Figura 1.5.

Finalmente, también se puede llamar moda digital a la ropa y accesorios que cuentan con tecnología electrónica inteligente, conocidos como *wearables*. Los *wearables* son accesorios o dispositivos electrónicos que se pueden llevar puestos y que están diseñados para realizar diversas funciones. Entre ellos se incluyen los *smartwatches*, camisetas que con sensores monitorizan el estrés, la frecuencia cardíaca, el ritmo respiratorio y la actividad deportiva, pijamas que miden el sueño y alertan a los familiares de caídas, o zapatillas que rastrean las métricas de carrera [21]. También podemos añadir en este grupo la personalización de ropas con LEDs, pantallas modulares flexibles, como podemos ver en la Figura 1.6, o efectos sonoros.



Figura 1.3: Software Lectra [3]



Figura 1.4: Probador virtual (AR) [4]

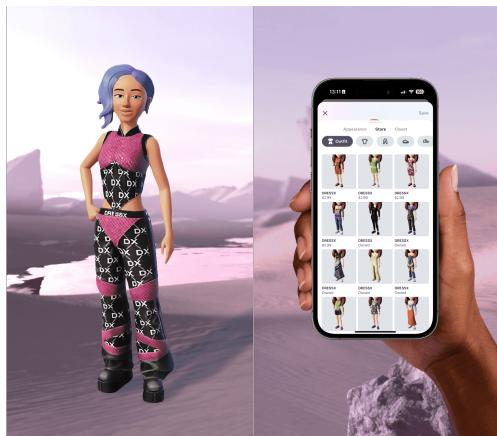


Figura 1.5: Metaverso DressX [5]



Figura 1.6: Proyecto Primrose Adobe [6]

En este trabajo nos vamos a enfocar en la tendencia del diseño y la creación de prototipos digitales.

1.1.3. Proyecto de colaboración *TaiLOR*

TaiLOR es un proyecto para la elaboración rápida de prototipos de prendas de vestir en 3D, utilizando Inteligencia Artificial. También llamado *sastrerIA*, es una colaboración público-privada entre la Universidad Rey Juan Carlos y SEDDI (Desilico), que comenzó su desarrollo en 2022 [22].

Desilico es una empresa formada en 2017 por profesores e investigadores de la Universidad Rey Juan Carlos para el desarrollo de herramientas de simulación y visualización 3D dirigidas al sector textil. En 2018 se asociaron con la empresa estadounidense SEDDI. Desde entonces, SEDDI ha subcontratado proyectos de investigación a la URJC y proporcionado empleos a estudiantes de la Rey Juan Carlos de estudios de grado y máster en informática gráfica, desarrollo de videojuegos, diseño y *marketing* de moda [23].

Este proyecto se está llevando a cabo en la Comunidad de Madrid y ha sido respaldado por el Ministerio de Ciencia e Innovación y la Agencia Estatal de Investigación, a través de los Fondos Next Generation, con número de expediente CPP2021-008842 con un presupuesto de 579.643€ [24].

Se ha identificado un gran problema en la industria de la moda, donde la digitalización en la cadena de suministro aún no se ha adoptado como una tecnología sólida. La industria textil es más insostenible y contaminante de lo necesario debido a los desperdicios en los procesos de fabricación, la sobreproducción, el exceso de muestreo y las altas tasas de devolución de los consumidores. *TaiLOR* propone prácticas industriales más sostenibles, acelerando la comercialización mediante muestras digitales y avatares personalizados.

El objetivo es desarrollar un proceso basado en IA para facilitar la creación de prototipos digitales de prendas 3D. Para ello hay que resolver tres problemas:

- **Datos de material limitados para la digitalización:** Normalmente, un diseñador solo cuenta con una pequeña muestra física de material e información muy limitada sobre las propiedades de fabricación, composición y familia de tejidos. Tampoco tienen acceso a hardware de escaneo especializado. Lo que se pretende en este proyecto es desarrollar una base de datos propia y aprovechar ese conocimiento para construir una IA capaz de utilizar imágenes de dispositivos de escaneo más convencionales para obtener todos los datos mecánicos y ópticos necesarios para simulaciones precisas del material.
- **Cosido 3D de paneles 2D para confeccionar una prenda 3D sobre un avatar maniquí:** Actualmente, ningún *software CAD* 3D disponible es capaz de ayudar al usuario a ensamblar y coser automáticamente la prenda. Este tipo de operaciones requieren la manipulación de piezas en 3D, una tarea compleja y tediosa que requiere un conjunto de habilidades que ge-

neralmente faltan o son escasas en la mayoría de los estudios de diseño de moda. En esta parte del proyecto se planea usar el aprendizaje automático para etiquetar automáticamente piezas 2D y buscar configuraciones de ensamblaje 3D factibles.

- **Escalado de patrón avanzado:** Aunque existen soluciones disponibles para escalar un diseño de patrón dado, generalmente siguen una progresión lineal y no tienen en cuenta las restricciones del cuerpo humano ni la intención de diseño, por ejemplo, la relación entre los puntos de referencia del cuerpo y las partes de la prenda, o la influencia de la caída del material. Se pretende crear un algoritmo de inteligencia artificial para escalar de forma no uniforme las subpartes y/o piezas del patrón 2D.

Estas iniciativas aún se encuentran en fase de desarrollo. La contribución propuesta en este proyecto se centra en el marco del escalado de patrones avanzado. En esta etapa de la investigación de *TaiLOR* buscamos que, una vez diseñada la prenda para un avatar específico, el programa pueda escalar automáticamente el patrón a diferentes tallas, teniendo en cuenta el ajuste y el diseño de la prenda. De esta manera, podremos reducir significativamente los costos de fabricación y material al eliminar la necesidad de realizar el escalado de patrones de forma manual.

Este trabajo fin de máster contribuirá a la investigación para diseñar el sistema de escalado automático. Optamos por realizar un sistema de escalado con un enfoque paramétrico, en el que, a través de diferentes puntos de medida establecidos en la prenda, podremos escalar automáticamente el patrón 2D a diferentes tallas o avatares con distintas medidas corporales. Para ello, ha sido necesario implementar una serie de herramientas de edición paramétrica en el *software* de SEDDI y resolver un problema de optimización para obtener este escalado automático. Estos aspectos se detallarán a lo largo del trabajo.

1.2. Objetivos

El objetivo de este trabajo es diseñar e implementar una herramienta que permita editar un patrón base de forma paramétrica en la plataforma SEDDI Author, e investigar el escalado de patrones desde un punto de vista paramétrico.

Para alcanzar este objetivo, se han definido los siguientes subobjetivos:

- Crear una herramienta en SEDDI Author que permita al usuario crear dinámicamente puntos de medida.
- Crear una herramienta de edición paramétrica en SEDDI Author, para que los puntos de medida puedan modificar la prenda en 2D y en 3D. También se deben tener en cuenta las conexiones entre puntos de medida.
- Diseñar y resolver un problema de optimización para encontrar los valores óptimos de los puntos de medida que preserven el ajuste de la prenda entre avatares de distintas medidas antropomórficas.

1.3. Requisitos

Para cumplir con los objetivos citados anteriormente debemos tener en cuenta los siguientes requisitos:

- La herramienta de edición paramétrica debe ser compatible con el *software* actual de SEDDI Author .
- El patrón base que vamos a utilizar en este trabajo es el patrón de una camiseta recta.
- La herramienta de edición paramétrica debe modificar la longitud de los puntos de medida (POMs) para modificar el patrón 2D y prenda 3D en función de estos POMs. Estos puntos de medida los debe poder definir el usuario en el patrón.
- Cada punto de medida debe contar con un control deslizante y con un valor de entrada para tener la posibilidad de introducir el valor exacto de la longitud del punto de medida en centímetros.
- La herramienta debe permitir activar o desactivar la conectividad entre las piezas de un patrón.

Los requisitos de la herramienta han sido definidos por las necesidades de los diseñadores, la empresa colaboradora y las tendencias del mercado.

1.4. Metodología

Para el desarrollo de este proyecto, se adoptó una metodología ágil [25] que combinó elementos de *Scrum*, *Kanban* y seguimiento continuo, para gestionar eficazmente las tareas y los avances en la implementación de las nuevas herramientas en la plataforma SEDDI Author .

Se llevaron a cabo reuniones semanales por videoconferencia con el tutor y el equipo del proyecto SEDDI *TaiLOR*, tanto en inglés como en español. Estas reuniones, programadas principalmente los martes de 12 a 13h o los lunes a las 10h, se realizaron a través de *Google Meet* [26]. Durante estas sesiones, se discutieron los avances semanales y se establecieron nuevas tareas para abordar en la semana siguiente. Además, se mantuvo la comunicación con el tutor a través de *Microsoft Teams* [27] y con el equipo de SEDDI mediante el *Seddi.Chat* de *Rocket* [28].

Para la gestión de tareas, se utilizó *Trello* [29], una herramienta *Kanban* en línea y gratuita. En *Trello*, se crearon tableros que representaban el flujo de trabajo del proyecto. Las tarjetas en *Trello* representaban las diferentes tareas del proyecto y se movían a través de columnas que representaban su estado, como se puede ver en la Figura 1.7. En particular, la columna “Haciendo” se limitaba a 2 o 3 tareas a la vez. Esta restricción ayudó a mantener un enfoque claro y concentrado en las actividades en curso, facilitando así la visualización clara del progreso de las tareas y la organización personal para realizar este trabajo fin de máster.

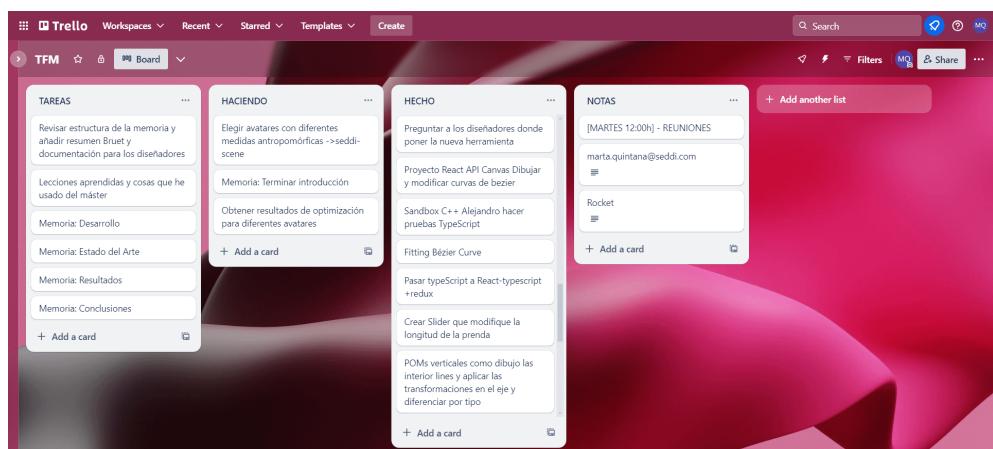


Figura 1.7: Gestión de tareas en *Trello*

El código fuente del proyecto se alojó en GitLab [30] y se utilizó un enfoque de control de versiones Git [31]. Dentro del repositorio de *Author*, se creó una rama específica para este trabajo llamada *parametric-editing-tool*. Esta rama refleja las herramientas implementadas en este proyecto y los cambios realizados hasta

Capítulo 1. Introducción

el momento. En un futuro cercano, si la empresa colaboradora decide que la funcionalidad implementada es la deseada, los desarrolladores de SEDDI revisarán y probarán esta rama y posteriormente será integrada en la rama principal del proyecto para que los clientes se beneficien de estas herramientas. En caso de que se decida que la funcionalidad no es la deseada, esta rama servirá para futuras mejoras o como referencia para posibles modificaciones dentro del proyecto de colaboración *TaiLOR*.

Además de las reuniones semanales, se fijaron otras reuniones por videoconferencia con los diseñadores para diseñar, probar y validar la herramienta de edición paramétrica propuesta.

También se realizaron algunas sesiones de *pair programming* con Alejandro Graciano e Iván Alduan del equipo de SEDDI Author. Estas videollamadas fueron fundamentales para adquirir los conocimientos necesarios y poder trabajar de manera autónoma en el desarrollo de las nuevas herramientas en Author, además de resolver algunas dificultades durante el proceso de implementación.

Se creó una página web tipo blog¹ para un seguimiento continuo del proyecto. Esta página web tipo blog que se muestra en la Figura 1.8, se utilizó para mantener un registro detallado de los progresos y facilitar la comunicación con todos los miembros del equipo sobre el estado del proyecto. En Trello se reflejaban las tareas en curso y su estado actual, mientras que la página web tipo blog proporcionaba una visión más amplia y detallada de las tareas realizadas y progreso general del proyecto.

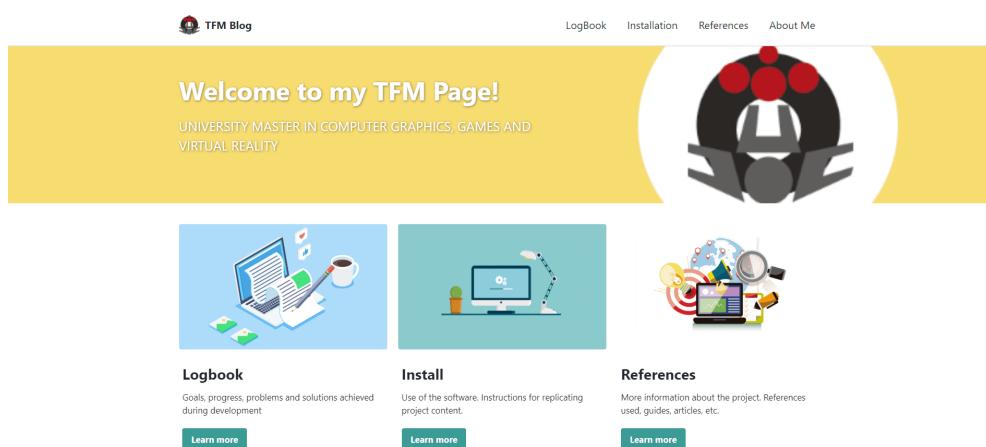


Figura 1.8: Página web de este TFM

Este enfoque combinado de metodologías ágiles nos ha permitido una gestión efectiva del proyecto, proporcionando una organización clara y flexible para mantener un seguimiento detallado del progreso en todo momento.

¹<https://martaquintana.github.io/2023-tfm-migjrv/>

1.5. Plan de Trabajo

El plan de trabajo seguido durante este proyecto se puede dividir en las siguientes etapas:

1. **Investigación y análisis:** en esta fase se llevaron a cabo varias reuniones para analizar y definir el tema del Trabajo Fin de Máster, utilizando como referencia el artículo de Brouet [32].
2. **Curvas de Bézier y nuevas tecnologías:** esta fase inicial consistió en repasar los conocimientos matemáticos sobre curvas de Bézier y realización de ejercicios y tutoriales de distintas herramientas que se utilizan en SEDDI Author. Se desarrolló una pequeña aplicación web utilizando *Node.js* para entender el funcionamiento de *React* con *TypeScript*, *Canvas API* y *Redux*.
3. **Aterrizaje en SEDDI AUTHOR:** se realizó la familiarización con el entorno de desarrollo de SEDDI Author, donde fue necesario solicitar permisos de acceso y realizar la instalación de las herramientas necesarias.
4. **Implementación de las herramientas de edición paramétrica:** desarrollo e implementación de las herramientas de edición paramétrica dentro de SEDDI Author.
5. **Problema de optimización:** diseño y desarrollo del problema de optimización relacionado con la adaptación de los valores de los parámetros de los puntos de medida para preservar el estilo de la prenda entre diferentes morfologías de avatar.
6. **Extracción de resultados:** análisis y extracción de los resultados obtenidos durante el desarrollo de la herramienta de edición paramétrica.

En la Figura 1.9 se muestra el diagrama de Gantt que representa los meses dedicados a cada fase del plan de trabajo. La documentación del blog y la memoria del proyecto se mantuvieron constantemente actualizadas a lo largo del desarrollo, aunque hacia la fase final, el enfoque se centró principalmente en la redacción y actualización de esta memoria.



Figura 1.9: Diagrama de Gantt

1.6. Estructura del documento

La estructura de este trabajo está compuesta por los siguientes apartados:

1. **Introducción:** en este capítulo se hace una breve introducción a la industria de la moda, centrándose en la moda digital y el contexto del proyecto de investigación en el que se encuentra este trabajo. Se establecen los objetivos y la metodología que se ha seguido para alcanzarlos a lo largo del proyecto.
2. **Estado del arte:** en este apartado se mencionan otros paquetes de *software* de patronaje digital y se explica el motivo por el que se ha elegido implementar una herramienta de edición paramétrica en este trabajo.
3. **Análisis y Desarrollo:** se describen las herramientas desarrolladas y cómo se han implementado en el *software* de SEDDI Author. Este apartado se ha dividido en varios subapartados para exponer de una manera más clara y detallada cómo se ha solventado cada fase de esta herramienta.
4. **Resultados:** se analizan y muestran ejemplos visuales de las diferentes herramientas implementadas.
5. **Conclusiones y trabajos futuros:** se presentan las conclusiones obtenidas a partir del trabajo realizado, así como posibles proyectos futuros derivados de este.

2

Estado del arte

En esta sección se presentan las principales herramientas de diseño de moda y patronaje que se utilizan hoy en día y qué tipos de ediciones paramétricas existen. Hablaremos sobre los puntos de medida y cómo conservar el estilo de la prenda entre diferentes tallas. También se muestra una pequeña presentación de SEDDI y SEDDI Author, la aplicación donde se desarrolla este trabajo.

2.1. *Software de diseño de moda y patronaje*

Para optimizar el tiempo de preproducción y fabricación se utiliza *software* CAD 2D y 3D. Estas aplicaciones facilitan el escalado y la creación de diseños de la prenda, aunque se sigue verificando con modelos reales [20]. Las herramientas más utilizadas en el diseño de moda y patronaje varían dependiendo de las necesidades específicas de cada diseñador y empresa. Algunos de los *software* más populares en la industria son:

- **Adobe Illustrator:** es un *software* 2D que se utiliza para la creación de bocetos digitales y diseño de patrones [33], como podemos ver en la Figura 2.1.
- **Cameo:** es un *software* de diseño de patrones de costura que permite a los usuarios crear y modificar patrones de costura en 2D, como se muestra en la Figura 2.2. Destaca por su capacidad para generar patrones en tamaño real, listos para impresión, facilitando el proceso de corte y confección [8].

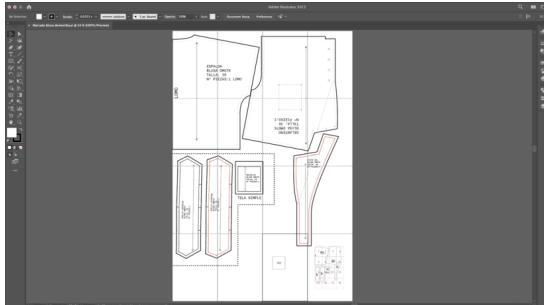


Figura 2.1: Adobe Illustrator [7]

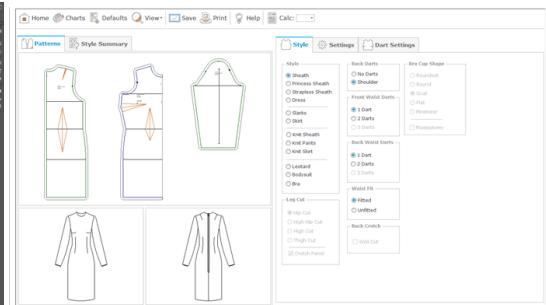


Figura 2.2: Cameo [8]

- **Optitex:** ofrece herramientas para el diseño de patrones tanto en 2D como en 3D para la creación de prototipos virtuales y la simulación de tela. Podemos visualizar un ejemplo en la Figura 2.3. Destaca por su tecnología de digitalización de patrones [9].
- **Lectra y Gerber AccuMark:** *software* industrial asociado a maquinaria textil y especializado en la digitalización de patrones, creación de marcadores y diseño de prototipos digitales [34][35].
- **Browzwear y Clo3D:** ambos son populares para la creación de prototipos virtuales en 3D y la visualización de prendas en entornos tridimensionales en tiempo real. Browzwear [10] suele ser preferido en entornos más profesionales debido a su enfoque en la precisión de simulación de telas y el realismo aparente de las prendas virtuales, como se muestra en la Figura 2.4. En cambio, Clo3D [36] es popular entre diseñadores independientes y estudiantes por su accesibilidad y funcionalidad.
- **Marvelous Designer:** Versión muy popular de Clo3D [36], que está adaptada al *pipeline* de videojuegos o efectos visuales (VFX) [37].

Además de todas estas soluciones *software*, existen otras herramientas de diseño 2D y 3D disponibles en el mercado pero menos extendidas como Style 3D, Shima Shaki, Gerbera 3D, Tuka Tech, Suite DC o C-Design Moda Virtual [38].

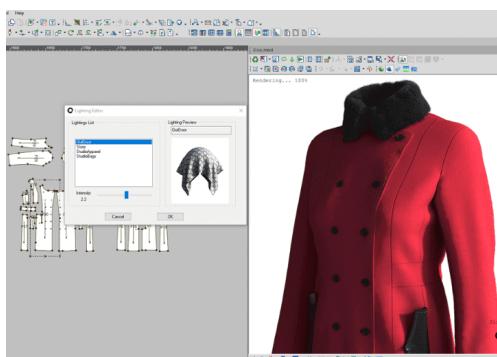


Figura 2.3: Optitex [9]

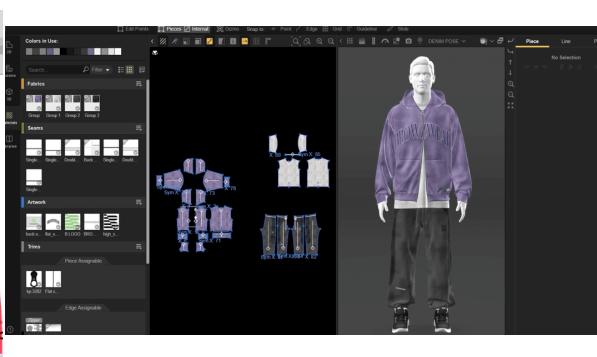


Figura 2.4: Browzwear [10]

2.2. Edición paramétrica

Las herramientas mencionadas en el apartado anterior suelen ofrecer flexibilidad en la edición de patrones y prendas, mediante ajustes manuales o manipulando los puntos de control *directamente en el patrón*. Sin embargo, solo algunas aplicaciones usan parámetros para crear y ajustar de manera indirecta los patrones.

Cameo permite el uso de parámetros para modificar los patrones base. Para una camiseta, los parámetros disponibles incluyen configuraciones de ajuste como la holgura de pecho, de cintura o de cadera, la profundidad del cuello delantero y trasero, el ancho del cuello, la punta del cuello, la forma y posición del escote/cuello. Además como se puede ver en la Figura 2.5, se pueden realizar ajustes generales en la longitud de la camiseta, la profundidad de la sisa, el ajuste de la tripa, la altura del hombro, la longitud del hombro y el punto del hombro entre otros [11].

Este tipo de programas tiene la siguiente limitación: si se edita el patrón desde los *settings*, no se puede editar directamente en el patrón de forma manual ya que se elimina las relaciones semánticas entre sus segmentos y además, los parámetros son específicos para cada tipo de prenda. Para los pantalones, por ejemplo, existen parámetros como la caída de la cintura, la cintura del pantalón, la forma y profundidad de la cadera, la longitud y posición de la pinza, la longitud de la entrepierna, la circunferencia del dobladillo, el ancho de la rodilla o la profundidad de la rodilla.

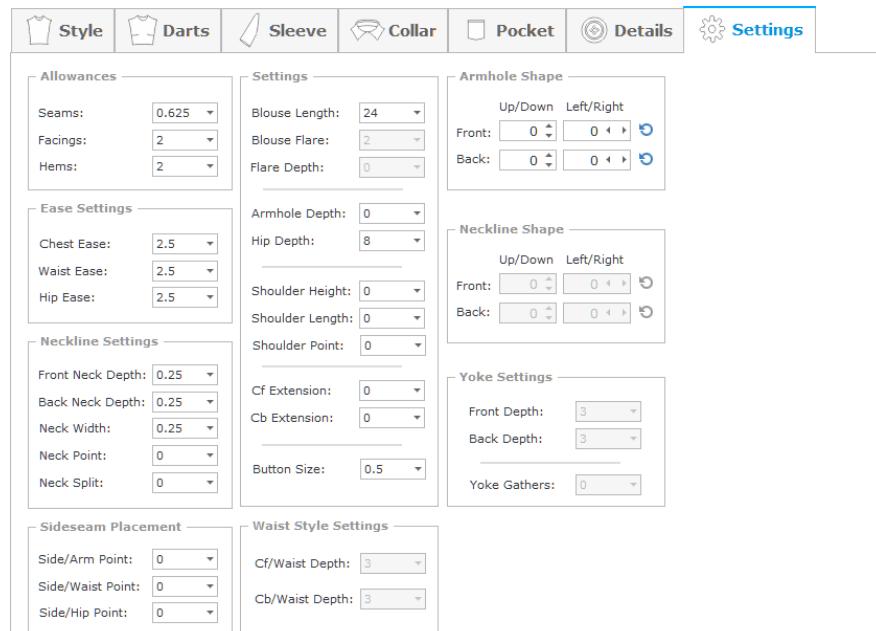


Figura 2.5: Configuración de diseño y ajuste de una camiseta en Cameo [11]

Clo3D ofrece un editor paramétrico de patrones [39] que solamente sirve para crear el patrón de una camiseta con mangas ajustando una serie de medidas específicas. Es decir, ofrece una serie de características que permiten modificar un patrón base antes de crearlo. En el cuerpo, se puede ajustar la longitud de la camiseta, la apertura y profundidad del cuello, el ancho de hombros, la caída o inclinación del hombro, el ancho delantero y trasero, el ancho de pecho (de sisa a sisa), el recorrido de la sisa y el dobladillo. Estas características se pueden observar en la Figura 2.6. En cuanto a las mangas, se puede editar la longitud de la tapa a la apertura, la altura y anchura de la tapa, el ancho de bíceps y la apertura de la manga. Estos ajustes se pueden visualizar en la Figura 2.7. Además, se encuentra la posibilidad de utilizar simetría tanto en el cuerpo como en las mangas.

Este editor tiene la limitación de que una vez creado el patrón, no es posible modificar las medidas. Si se necesita cambiar alguna medida, es necesario volver a crear el patrón desde cero con las nuevas medidas. En la Figura 2.8 se muestra esta herramienta y podemos ver cómo funciona en este vídeo¹.

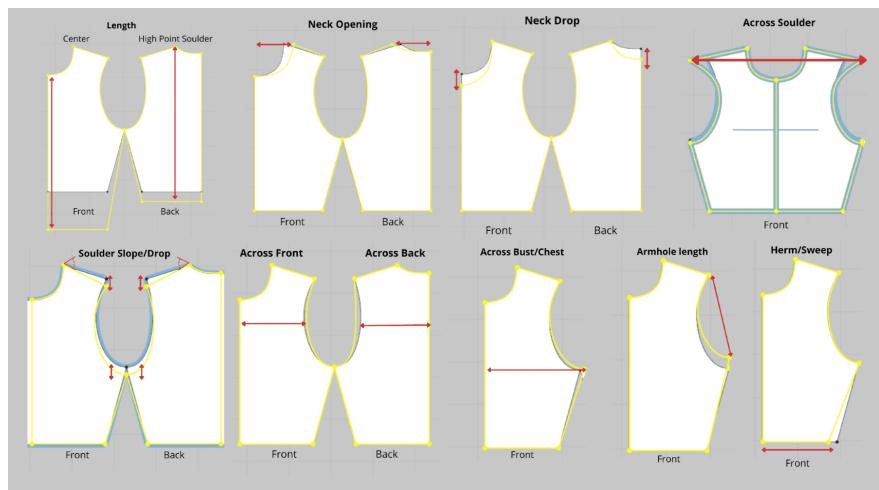


Figura 2.6: Posibles ediciones camiseta base del patrón paramétrico en Clo3D

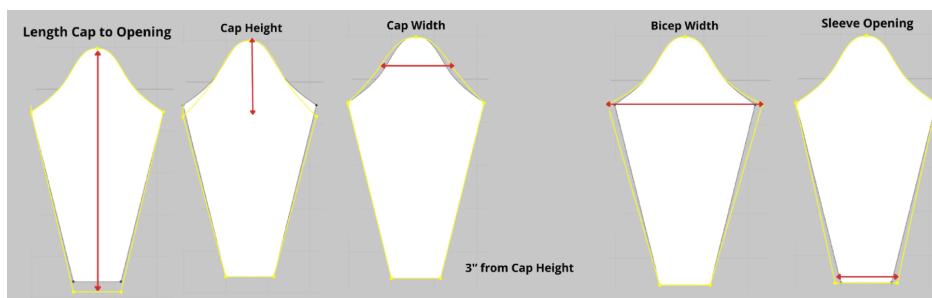


Figura 2.7: Posibles ediciones mangas base patrón paramétrico en Clo3D

¹<https://www.youtube.com/watch?v=0ZV2hKhpcIE&t=139s>

2.2. Edición paramétrica

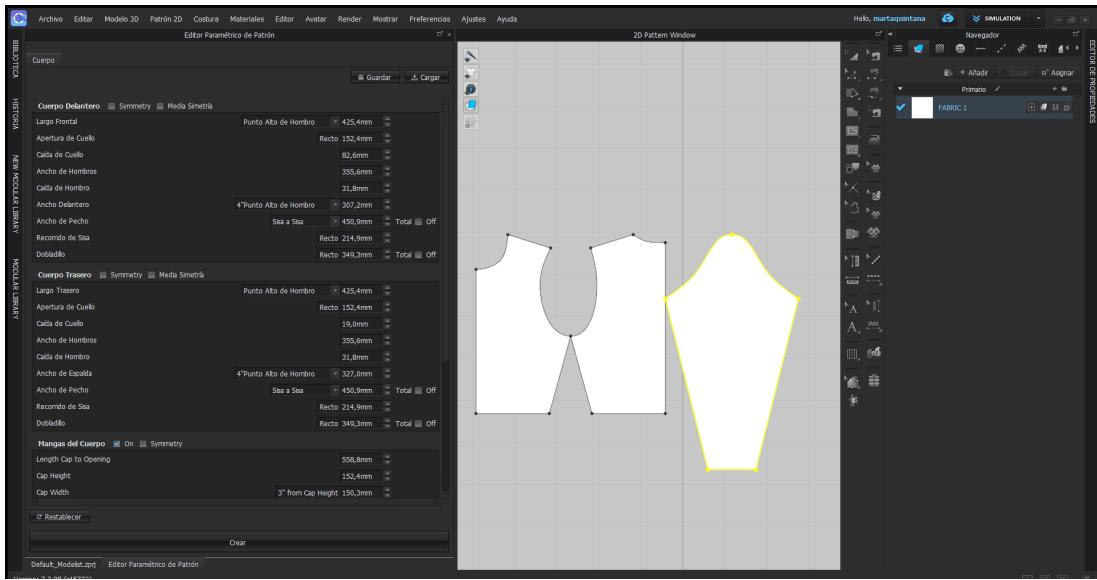


Figura 2.8: Edición paramétrica de patrón 2D en Clo3D

Por otro lado, en lo que respecta a los avatares, la mayoría de estos programas tienen avatares con estructuras básicas fijas. Aunque puedes elegir entre una variedad de animaciones o posiciones predefinidas, y cambiarlo entre un amplio catálogo de formas corporales y tallas, por lo general no puedes ajustar características como la altura del avatar en tiempo real. Sin embargo, existen algunos programas como Browzwear que sí permiten la edición paramétrica del avatar [40]. Esto significa que puedes ajustar ciertos aspectos específicos como la altura, el ancho del pecho, la cintura y el ancho de los brazos y piernas. En la Figura 2.9 se muestran los parámetros que puedes editar.

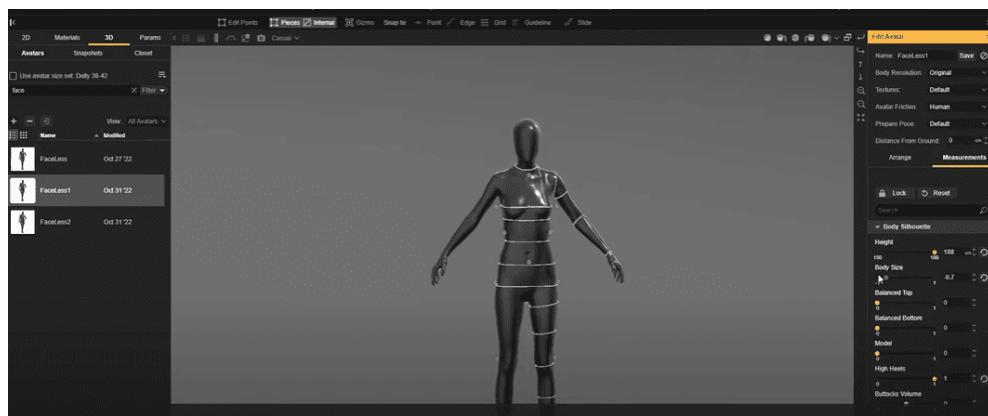


Figura 2.9: Edición de avatar paramétrico en Browzwear

2.3. Puntos de medida: POM

Los puntos de medida o POM (*Points of Measure*), son marcas realizadas sobre un patrón que se utilizan para tomar medidas y asegurar que las prendas se ajusten correctamente. Estas medidas definen una distancia objetivo entre dos puntos específicos, a menudo también se definen con circunferencias completas o medias circunferencias.

Un POM puede ser la distancia entre dos puntos como la longitud desde el hombro hasta el dobladillo, circunferencias completas como el perímetro de la cintura, o medias circunferencias como las medidas de la mitad del contorno.

Estos puntos de medida ayudan a los diseñadores a adaptar los patrones en el proceso de escalado de la prenda, creando las reglas de escalado entre tallas conocidas como *grading rules*. Estas reglas determinan cómo se aumentan o disminuyen las medidas de los POMs en cada talla de la prenda, garantizando que todas las tallas mantengan las proporciones correctas [12]. En la siguiente imagen, Figura 2.10, podemos ver que la talla base es la talla M y las tallas S y L se definen a partir de estos POMs, sumando o restando los centímetros necesarios a la talla base.

	Card Title	ΦS	ΦM	ΦL	
	Chest Circumference	-2 46	0 48	2 50	
	Waist Circumference	-2 45	0 47	2 49	
	Hip Circumference	-2 44	0 46	2 48	
	Total Length of Garment	-1/2 26	0 26 1/2	1/2 27	
	Sleeve Slit	0 5	0 5	0 5	
	Pocket width	0 4 1/2	0 4 1/2	1/4 4 3/4	
	Pocket length	0 5 1/2	0 5 1/2	0 5 1/2	
	Pocket Flap Length	0 1 3/4	0 1 3/4	0 1 3/4	
	Collar Width	0 1 3/4	0 1 3/4	0 1 3/4	
	Neckline Circumference	-1/2 15	0 15 1/2	1/2 16	
	Placket Width	0 1 1/2	0 1 1/2	0 1 1/2	
	Armhole Circumference	-1 20	0 21	1 22	
	Sleeve Length	-1/2 24	0 24 1/2	1/2 25	
	Sleeve Cuff Width	0 3	0 3	0 3	
	Sleeve Hem Circumference	-1/4 5 1/2	0 5 3/4	1/4 6	

Figura 2.10: Ejemplo de *grading rules* de una camiseta [12]

Estas medidas garantizan que las prendas se ajusten correctamente a diferentes tipos de cuerpo. Durante la producción, permiten realizar verificaciones precisas para asegurar que las prendas se han fabricado según las especificaciones

2.3. Puntos de medida: POM

[41]. También ayudan a mantener la consistencia en la fabricación, asegurando que todas las unidades de la producción tengan las mismas dimensiones. En la fabricación a medida, los POMs se utilizan para crear prendas que se ajusten perfectamente a las medidas individuales de los clientes. En las Figuras 2.11 y 2.12 podemos ver ejemplos de puntos de medida.

POM #	Point of Measurement	Point of Measurement Description
1-1	Chest 1" below armhole	Measure 1" down from bottom of armholes, straight across garment from side to side.
1-3	Sweep (Straight)	Measure straight across garment from edge to edge at bottom(for garments with slits - measure above the slits).
1-4	Sweep Mid Band/Rib (Relaxed)	Measure straight across at the center of rib/band side to side.
1-5	Sweep Mid Band/Rib (Extended)	Measure straight across the center of rib/band side to side fully extended.
1-6	Shoulder Slope	Measure from extended imaginary line between HPS, down to the shoulder/armhole point.
1-9	Across Front 8" Below HPS	Measure .8" down from HPS across front from armhole seam to armhole seam.
1-11	Front Length HPS	Measure from HPS straight down to the bottom edge of the garment. Include rib trim/band if applicable.
1-16	Armhole Circumference	Measure along the contour of armhole seam from the bottom of the armhole to the top of the armhole.
1-17	Armhole (Straight)	Measure from shoulder point straight down to the armhole/side seam intersection.
1-26	Upper Arm (Muscle) 1" below AH	Measure 1" below armhole/side seam intersection, straight across the sleeve perpendicular(90 degree angle) to the fold of the sleeve.
1-28	Forearm	Measure . up from sleeve opening, straight across the sleeve perpendicular(90 degree angle) to the fold of the sleeve.
1-29	Sleeve Opening (Relaxed)	Measure from edge to edge along bottom of the sleeve opening.
1-32	Sleeve Cuff Height	Measure from the join seam to the bottom edge.
1-68	Bottom Rib/Band Height	Measure from the join seam to the bottom edge of the rib/band.

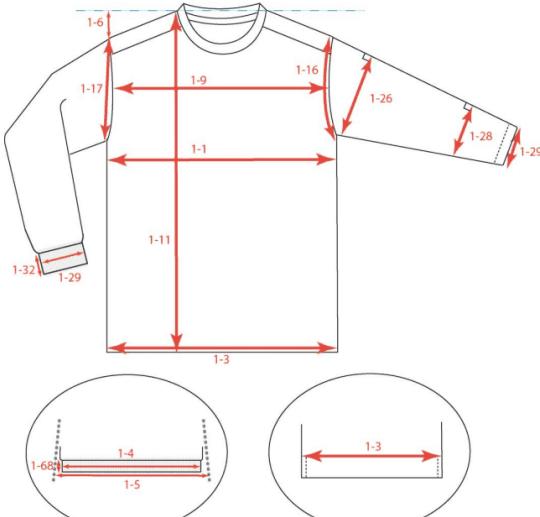


Figura 2.11: Puntos de medida en una camiseta [13]

POM #	Point of Measurement	Point of Measurement Description
2-1	Waist Relaxed	Measure from side to side at the middle of the waistband. For pants without waistband – measure at the top edge of the waist.
2-2	Waist Extended	Measure from side to side at the middle of waistband fully extended.
2-3	Waistband Height	Measure from the top edge of the waistband to the bottom edge of waistband or last row of stitching for garments with an elastic waist.
2-5	Seat (Low Hip) - 9" below waistband	Measure down from the center front and natural fold of the side seams 9" below the waistband seam (or the last row of stitching on a waistband). Then measure across the marked points from the side to the center to the opposite side.
2-6	Front Rise - including waistband	Measure from crotch point to top of waistband, following the contour of the front rise.
2-8	Thigh - 1" below crotch	Measure 1" below crotch point straight across from edge to edge, parallel to the leg opening.
2-9	Knee Width	Measure straight across from edge to edge, parallel to the leg opening, 2" above the half of the inseam length.
2-10	Leg Opening (Relaxed)	Measure from edge to edge.
2-11	Leg Opening (Extended)	Measure from edge to edge with cuff fully extended.
2-12	Inseam	Measure from crotch point, following the seam to the hem edge.
2-21	Front Pocket Opening	Measure from top of pocket opening to bottom of pocket opening following the contour of the opening (from bartack to bartack).
2-43	Cuff Height	Measure at the cuff join seam to the bottom edge of the cuff.

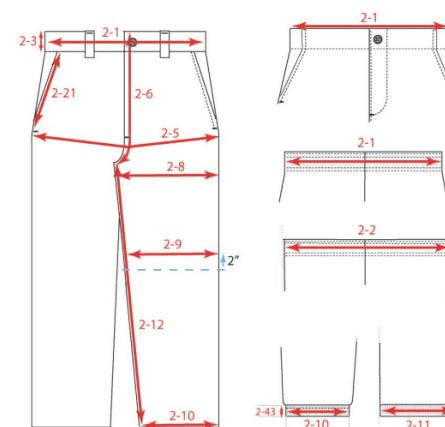


Figura 2.12: Puntos de medida en un pantalón [13]

2.4. Preservación del estilo de la prenda

En este contexto, el estilo se refiere a las proporciones y relaciones del patrón basadas en medidas antropométricas, asegurando un ajuste adecuado y una apariencia deseada en la prenda final. La transferencia del estilo en prendas de vestir presenta un desafío técnico que se encuentra en constante investigación. Los estudios que se enfocan en la transferencia de diseños de una prenda a otra, manteniendo la apariencia y el ajuste original son escasos. Un trabajo destacado en este ámbito es el artículo “*Design Preserving Garment Transfer*” de Brouet (2012) [32]. Este artículo propone un método para la transferencia de prendas de forma automática que preserva el diseño y el estilo entre avatares con diferentes formas corporales. Siguiendo este procedimiento consiguen preservar el estilo de la prenda en los diferentes avatares, como podemos observar en la Figura 2.13.



Figura 2.13: Resultados del artículo donde se puede ver la transferencia de la prenda, preservando el diseño original entre avatares con distintas medidas antropomórficas.
Fuente: Laurence Boissieux 2012

Este proceso se compone de tres pasos principales: calcular nuevas posiciones 3D de la prenda proporcional al nuevo avatar, modificar esas posiciones 3D para que preserven el diseño, y finalmente calcular un nuevo patrón 2D para la prenda 3D final.

Para ello, proponen calcular correspondencias para cada punto de la prenda en la superficie del avatar original, pero de una manera algo sofisticada. En lugar de simplemente tomar el punto más cercano, buscan correspondencias que sean ortogonales a algunos de los huesos del avatar. Una vez calculadas estas relaciones, cada punto de la prenda tiene una correspondencia entre el avatar y un *offset*, que es el vector que une ese punto en el avatar de referencia con el punto correspondiente en la prenda. Entonces, para el nuevo avatar, la nueva posición del punto de la prenda es simplemente el punto correspondiente en el nuevo avatar más el *offset*.

A continuación, realizan una optimización numérica con una función de energía para intentar recuperar el estilo original de la prenda. Esto se basa en criterios de orientación de los nuevos triángulos en comparación con los originales, criterios

de posición con respecto a los huesos asociados a las correspondencias, y criterios basados en evitar colisiones con el nuevo avatar.

Por último, hacen una parametrización de la malla del nuevo diseño en 3D, seguido por ajustes heurísticos en las fronteras del resultado 2D para eliminar ruido, para mejorar las fronteras del diseño 2D y obtener un resultado más limpio.

2.5. SEDDI

SEDDI es una empresa de *software* que ofrece una solución colaborativa de *software CAD 3D* para simulación de prendas desde la nube. Actualmente se utiliza para crear textiles digitales y prendas de vestir de manera eficiente [42][43]. SEDDI cuenta con dos herramientas: SEDDI Textura y SEDDI Author.

SEDDI Textura es una herramienta que permite digitalizar tejidos de forma precisa. Escaneando una muestra de tela con un escáner convencional, son capaces de simular de manera rápida y precisa cómo se comporta el material textil en el mundo real gracias a la red neuronal que utilizan para entrenar el modelo con datos mecánicos y ópticos. Desde la aplicación web puedes simular repeticiones perfectas y físicas realistas. También permite exportar este material para poder utilizarlo en otros programas como Clo3D, SEDDI Author, Browzwear o Substance by Adobe, entre otros [44].

SEDDI Author es una plataforma que permite diseñar, crear y producir prendas digitales con alta precisión técnica. Facilita la colaboración en tiempo real entre usuarios, así como la prueba y simulación de telas (que pueden importarse de SEDDI Textura, Figura 2.14). También permite crear maquetas fotorrealistas para ventas y marketing, así como elaborar prototipos de prendas gráficas. De esta forma, se logra reducir de forma drástica la necesidad de muestras físicas, ya que se lleva a cabo la construcción de las prendas y los tipos de costuras de manera digital [45].

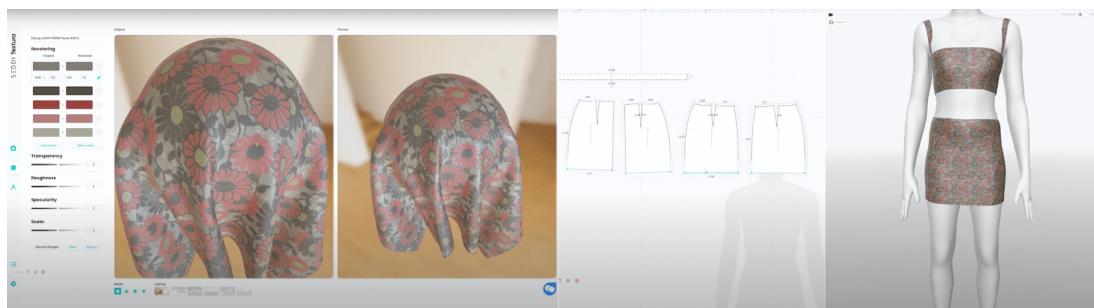


Figura 2.14: Material escaneado y digitalizado en SEDDI Textura junto a un patrón diseñado y cosido en SEDDI Author [44]

SEDDI Author

A diferencia de los *software CAD* mencionados anteriormente, SEDDI Author es la única plataforma que no requiere de instalaciones, todo se ejecuta a través del navegador. Además, facilita a los equipos de trabajo ver, crear y editar una colección completa al mismo tiempo. Otra de las ventajas es que utiliza la tecnología *TrueSeams™*, lo cual permite una gran precisión en la simulación de las costuras y las telas para tomar decisiones de producción en tiempo real, sin tener que realizar muestras físicas [45].

A nivel técnico, esta plataforma utiliza múltiples herramientas y tecnologías para su funcionamiento. El servidor de SEDDI Author está desarrollado en *Node.js*, la interfaz de usuario está implementada con *React*, *TypeScript* y *Redux*, utiliza *Docker* para gestionar y ejecutar algunos servicios y maneja tecnologías gráficas como *Canvas API* y *WebGL*.

React es una biblioteca de *JavaScript* para la construcción de interfaces de usuario interactivas y dinámicas [46]. Se utiliza *React* con *TypeScript*, que es un superconjunto de *JavaScript* que añade tipos estáticos y objetos basados en clases, que hace mejorar la robustez y mantenibilidad del código [47]. En la Figura 2.15 se muestra el diagrama de *Redux* integrado con *React* que utiliza SEDDI AUTHOR, el cual analizaremos a continuación.

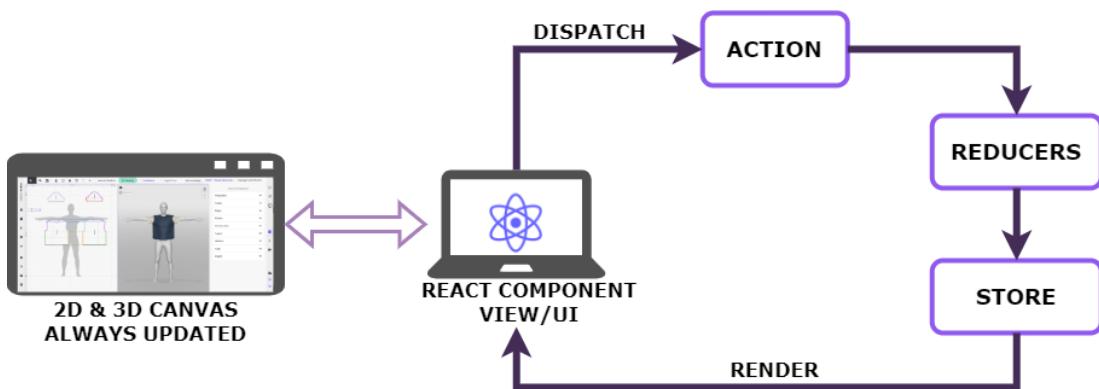


Figura 2.15: Diagrama del flujo de datos en SEDDI AUTHOR

Redux se utiliza para la gestión del estado global de la aplicación [48]. El estado global es un almacén de datos centralizado que contiene toda la información de la aplicación y está disponible desde cualquier lugar dentro de la misma. Para actualizar esta información se utilizan acciones y *reducers*. Las acciones son eventos que provocan cambios, mientras que los *reducers* son funciones encargadas de actualizar el estado en respuesta a estas acciones. El estado global o *Store*, Figura 2.16, almacena todos los datos y permite que diferentes partes de la misma accedan y actualicen este estado adecuadamente, garantizando así que la aplicación

se mantenga sincronizada y consistente en todo momento [49]. Por ejemplo, si un usuario arrastra un nuevo avatar a la escena, la aplicación registrará el evento “arrastrar”, lanzará la acción “actualizar avatar” con un *dispatch* y el reducer cambiará el identificador del avatar actual por el nuevo en el estado global (*Store*) y se actualizarán las visualizaciones correspondientes en la aplicación web.

Dentro de esta interfaz de usuario se utiliza el *Canvas API* de *HTML5* [50] y *WebGL* [51]. *Canvas API* sirve para renderizar los patrones de las prendas en 2D y 3D en el navegador, asegurando de que estén siempre actualizados. Los modelos 3D se benefician de *WebGL* para que el renderizado de las prendas se realice de forma eficiente utilizando la tarjeta gráfica o *GPU*.

El uso de todas estas tecnologías facilita el desarrollo de aplicaciones tan complejas como esta para que sean robustas, escalables y funcionen en cualquier sistema operativo y dispositivo. El principal problema de estas aplicaciones es que necesitas conexión a internet, pero con los últimos avances tecnológicos con las conexiones Wi-Fi, la fibra óptica o el 5G, hacen que la mayoría de personas podamos tener acceso desde cualquier lugar y no sea un gran inconveniente.

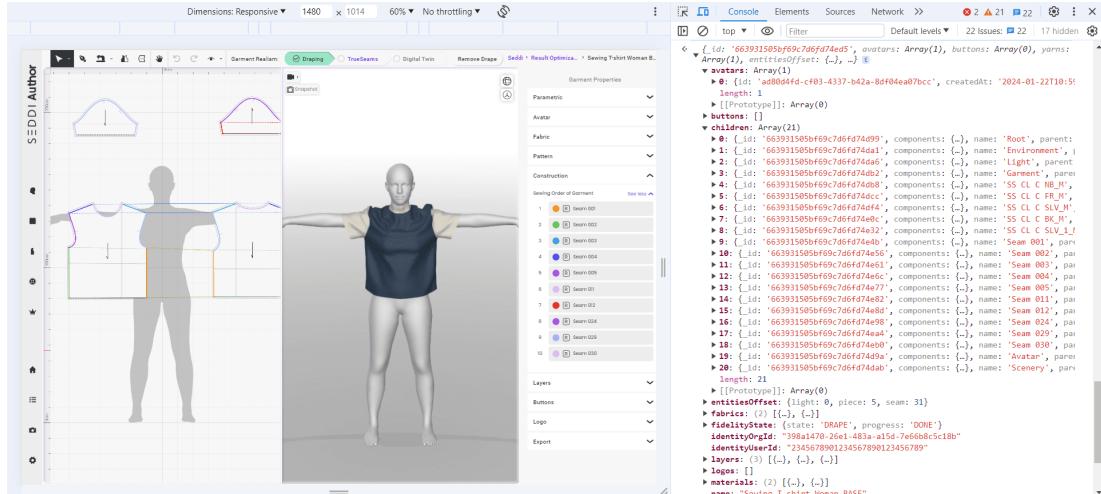


Figura 2.16: Objeto *Store* del estado de *Redux* en la consola de la aplicación con los datos actualizados

3

Análisis y Desarrollo

En este apartado se analiza el diseño y desarrollo de las nuevas herramientas para la creación de puntos de medida y edición paramétrica en SEDDI Author. Además, se plantea y resuelve un problema de optimización para la preservación del estilo de la prenda entre diferentes avatares. Para hacerlo de una manera sencilla, este capítulo se divide en varias secciones. Primero, se introducen las curvas de Bézier, que son la base matemática de los patrones. A continuación, se analiza el diseño de la herramienta y se describe por un lado el desarrollo de la herramienta de creación de POMs y, por otro, la de edición paramétrica, aunque veremos que una depende de la otra. Finalmente, se explora el problema de optimización, teniendo el artículo de Brouet [32] como base para obtener una solución.

En esta sección, se analizarán los fragmentos de código y pseudocódigo responsables de cada funcionalidad en la aplicación. Es importante tener en cuenta que el código proporcionado no es la implementación completa. En este enlace de GitHub¹ se puede acceder una versión más detallada de cada herramienta, aunque el código completo no está disponible públicamente dado que se encuentra en el repositorio oficial de SEDDI Author.

¹<https://github.com/martaquintana/2023-tfm-migjrv/tree/main/author-code>

3.1. Curvas de Bézier

Las curvas de Bézier son representaciones matemáticas polinómicas que se utilizan para modelar formas suaves en el campo de los gráficos por ordenador. Se utilizan para el diseño de fuentes, ilustraciones vectoriales, animaciones, programas CAD y en la interpolación de datos o aproximación de funciones.

Una curva de Bézier se define mediante un conjunto de puntos de control, donde la curva solo pasa por el punto inicial y el final. Los puntos intermedios permiten ajustar la forma y la suavidad de la curva. Se definen con una fórmula cerrada representada por la siguiente ecuación de grado n y no requiere resolver ningún sistema de ecuaciones. El orden o grado n es el número de puntos de control -1 :

$$\mathbf{B}(t) = \sum_{i=0}^n \binom{n}{i} (1-t)^{n-i} t^i \mathbf{P}_i$$

donde \mathbf{P}_i son los puntos de control, $\binom{n}{i}$ es el coeficiente binomial y t es un parámetro que varía de 0 a 1.

Las curvas más utilizadas son las de orden 1, 2 y 3 que podemos ver en la Figura 3.1 y se definen con las siguientes fórmulas:

- **Curva de Bézier Lineal (Orden 1):** está definida por dos puntos, P_0 y P_1 .

$$B(t) = (1-t)P_0 + tP_1, \quad 0 \leq t \leq 1$$

- **Curva de Bézier Cuadrática (Orden 2):** está definida por tres puntos, P_0 , P_1 y P_2 .

$$B(t) = (1-t)^2 P_0 + 2(1-t)tP_1 + t^2 P_2, \quad 0 \leq t \leq 1$$

- **Curva de Bézier Cúbica (Orden 3):** está definida por cuatro puntos, P_0 , P_1 , P_2 y P_3 .

$$B(t) = (1-t)^3 P_0 + 3(1-t)^2 tP_1 + 3(1-t)t^2 P_2 + t^3 P_3, \quad 0 \leq t \leq 1$$

Cuando $t=0$, estamos en el punto inicial y si $t=1$, estamos en el punto final de la curva. Entre estos valores, el parámetro t determina una interpolación lineal entre los puntos de control, donde $(1-t)$ y t actúan como pesos para cada punto de control respectivamente.

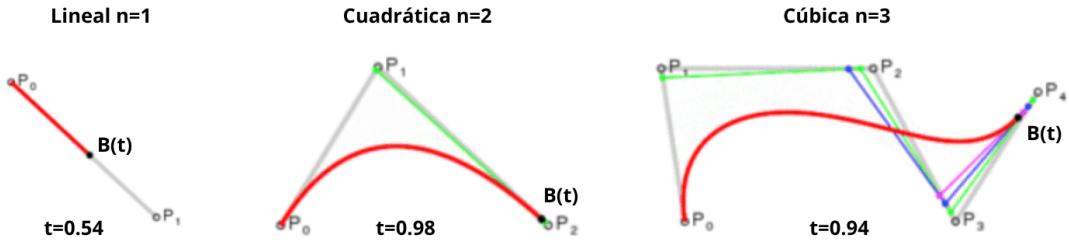


Figura 3.1: Curvas de Bézier de distintos órdenes con sus puntos de control [15]

Cuando el orden del polinomio es alto, los puntos de control son cada vez menos intuitivos y aparecen efectos poco deseables como el fenómeno de Runge, en el que aparecen oscilaciones grandes cerca de los extremos del intervalo de interpolación. Para resolver esto se subdivide en subconjuntos de puntos pequeños y se utilizan curvas de orden bajo, normalmente curvas cúbicas (curvas de orden 3 definidas por 4 puntos).

Al subdividir los intervalos en curvas, las curvas de Bézier pueden unirse de manera continua para formar trayectorias más complejas, conocidas como *paths* de Bézier, que en este caso nos sirven para definir curvas y rectas. En este proyecto y en la plataforma se utilizan las curvas de Bézier lineales, cuadráticas y cúbicas para definir los patrones en *paths* de Bézier como podemos observar en la Figura 3.2, donde cada segmento corresponde a una curva de Bézier.

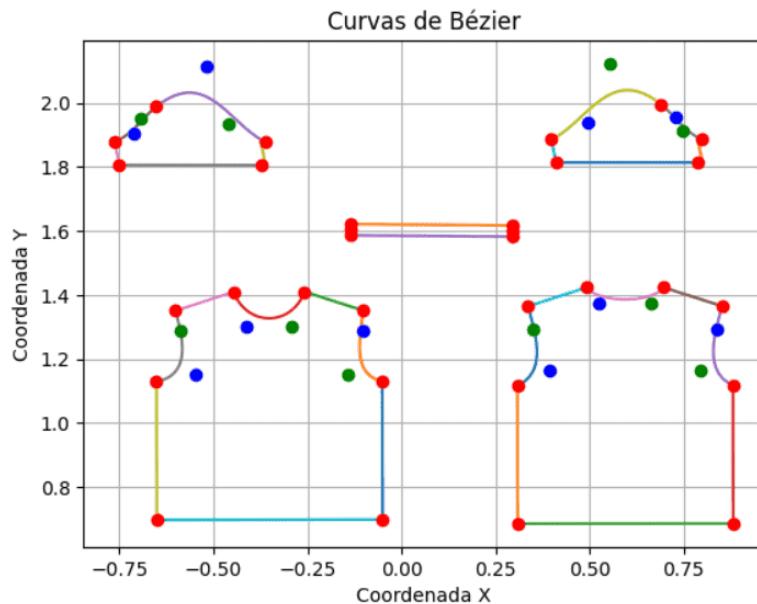


Figura 3.2: Patrón camiseta base definido con curvas continuas de Bézier, cada pieza esta formada por un *path* de Bézier, representado con Python

3.2. Análisis del diseño de la herramienta de edición de patrones en SEDDI Author

En el diseño de la herramienta es importante considerar el concepto de ajuste. El ajuste describe la relación entre una prenda y el cuerpo del usuario. Para mantener la confianza del cliente, las marcas deben preservar sus estándares de ajuste en todos sus productos y temporadas. Estos estándares se incluyen en el paquete técnico como especificaciones con puntos de medida. Actualmente, garantizar el ajuste implica un proceso complejo de envío de especificaciones, creación de patrones y muestras, y verificación manual mediante POMs. Para omitir el muestreo físico, se propone usar esta herramienta digital para medir con precisión el ajuste y poder hacer las modificaciones necesarias en el patrón basadas en distancias.

El diseño de la herramienta se realizó teniendo en cuenta las preferencias y el punto de vista de los diseñadores. Se usó Figma para esquematizar y tomar decisiones de diseño junto con los requerimientos de los usuarios. Los puntos de medida deben servir tanto para modificar la prenda como para tomar medidas.

En el esquema de la Figura 3.3 se muestra el flujo de la herramienta para generar, visualizar y editar puntos de medida, con los comentarios de los diseñadores. Primero vemos que el usuario debe definir los puntos de medida, seleccionando los puntos en el patrón. Una vez creados, aparecerán en las propiedades paramétricas. Al editar un POM, se debe comprobar si se va a escalar la pieza completa o no. Si está activada la opción de escalado completo y tiene *grading rules*, se escala la prenda completa con el valor del POM teniendo en cuenta los *grading rules* en el resto de la prenda. Si no, se debe comprobar si solo se escala el POM seleccionado o si existen conexiones con otros POMs. Si hay conexiones, se escalan todos los POMs afectados y si no hay conexiones, se escala solo el POM editado.

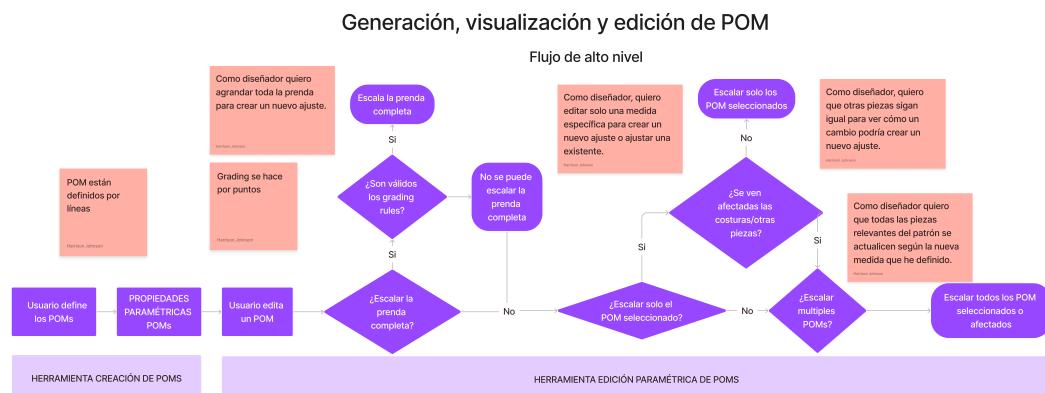


Figura 3.3: Flujo de trabajo del diseño de la herramienta de generación, visualización y edición de puntos de medida

Capítulo 3. Análisis y Desarrollo

La Figura 3.4 detalla el diseño propuesto por Harrison para la herramienta de los puntos de medida y la interfaz de usuario. Finalmente, se optó por el diseño que se presenta en la Figura 3.5. En el diseño de la interfaz de usuario, necesitamos una nueva sección en el panel de propiedades que se llamará “*Measurements*”, que presenta un listado de puntos de medida. Este panel permitirá que los usuarios puedan gestionar las medidas en los patrones 2D, eliminar una medida seleccionada o añadir una nueva. Al hacer clic en “*Add POM*”, el usuario podrá dibujar una línea sobre una pieza del patrón, ajustándose automáticamente al borde de la pieza. Una vez añadidos, se podrán guardar los POMs haciendo *click* en “*Save*”. Los POMs seleccionados se podrán editar mediante entrada de texto o mediante un control deslizante. También contará con una casilla de verificación para “escalar toda la prenda”, que actualizará todos los POMs si existen *grading rules* y el ícono de enlace que permitirá vincular los POMs, donde los usuarios podrán editar y eliminar estos enlaces individualmente.

Flujo de detalles: Herramienta de creación de POMs

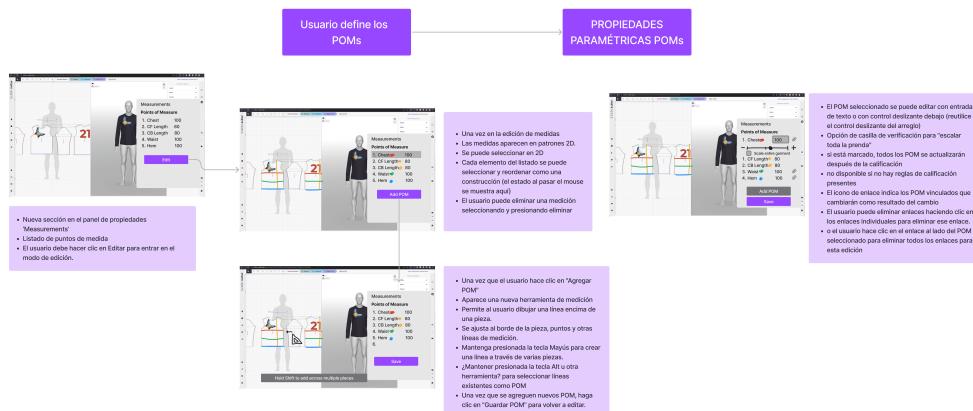


Figura 3.4: Flujo de detalles: Herramienta de creación de puntos de medida

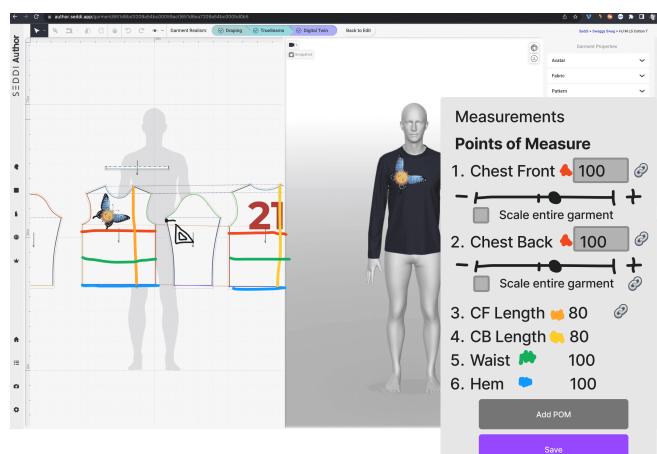


Figura 3.5: Diseño de la interfaz de usuario propuesto por los diseñadores para la herramienta de creación de POMs y edición paramétrica

3.3. Desarrollo de la herramienta de creación de puntos de medida

En este diseño solo se contemplan los puntos de medida que forman parte de las piezas; no se pueden definir puntos fuera de los patrones. Aunque en algunos casos los patronistas utilizan puntos de medida fuera de la prenda, se decidió limitar los puntos de medida para que vayan desde un punto o segmento a otro punto o segmento del borde de la prenda.

Es importante destacar que, aunque se ha diseñado y planificado completamente esta herramienta, no se ha implementando el escalado de la prenda teniendo en cuenta los *grading rules* debido a limitaciones de tiempo y recursos. Pero, se ha establecido una base sólida que puede ser desarrollada y expandida en trabajos futuros.

3.3. Desarrollo de la herramienta de creación de puntos de medida

Aunque la herramienta de creación de puntos de medida y de edición paramétrica están relacionadas, la implementación interna está separada. Sin embargo, ambas comparten el panel de interfaz de usuario que es un componente en *React* llamado “*ParametricPanel*”.

La interfaz de usuario se ha implementado en *React* con *TypeScript*. Para construir el componente “*ParametricPanel*” en Author se ha creado la clase *ParametricPatternProperties*. Esta clase, que se muestra en el código 3.1, es responsable de renderizar en la aplicación web los botones, texto, casillas de verificación y controles deslizantes para gestionar propiedades paramétricas. En este componente se definen funciones para manejar los diferentes eventos como crear, borrar o actualizar los valores de los puntos de medida. El botón de las líneas 15-21 “*Add POM*”, activa la herramienta *TOOLS.POM* que se encarga de la creación de los puntos de medida que vamos a analizar a continuación, y el botón “*Save*” de las líneas 22-28, activa la herramienta de edición paramétrica *TOOLS.PARAMETRIC_EDITION* que analizaremos en el siguiente apartado. Al usar *Redux*, este componente está conectado a diferentes eventos para tener acceso al estado global de la aplicación, línea 34, para lanzar acciones y mantener actualizados los datos del estado. En la Figura 3.6 se visualiza el panel al añadir dos POMs.

```
1 const ParametricPatternProperties = ({ show, scene, pointsOfMeasure }) =>
2 {
3   const { dispatch, getState } = useStore();
4   [...]
5   useEffect(() => {...}, [scene, changedPomIndex, dispatch, pointsOfMeasure]);
6
7   return (
8     <div>
9       <InputGroup name="Mesurements" hr={true}>
10         {pointsOfMeasure.length > 0 &&
11           pointsOfMeasure.map((pom, index) => (
```

Capítulo 3. Análisis y Desarrollo

```

12     <SliderComponent [...] />
13   ))
14 </div>
15 <Button
16   onClick={() => {
17     dispatch(show(setTool(TOOLS.POM)));
18     dispatch(editorSelectTool(TOOLS.POM));
19   }}>
20   Add POM
21 </Button>...
22 <Button
23   onClick={() => {
24     dispatch(show(setTool(TOOLS.PARAMETRIC_EDITION)));
25     dispatch(editorSelectTool(TOOLS.PARAMETRIC_EDITION));
26   }} >
27   Save
28 </Button>
29 </div>...
30 </InputGroup>
31 </div>
32 );
33 };
34 export default connect( {...}, )(ParametricPatternProperties);

```

Código 3.1: Clase ParametricPatternProperties

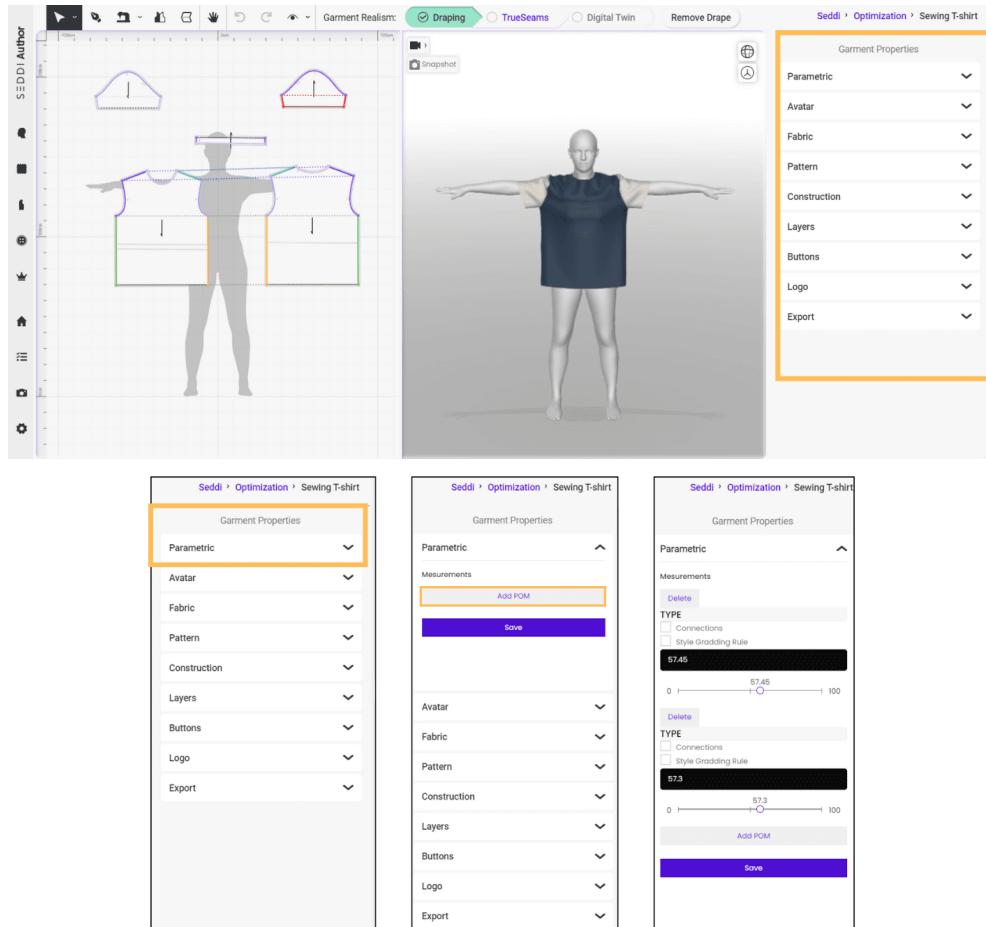


Figura 3.6: *ParametricPanel*

3.3.1. POM TOOL

En este apartado nos centramos en cómo se ha implementado la herramienta de creación de puntos de medida, “*POM TOOL*”, en la aplicación de SEDDI. Al pulsar en el botón “*Add POM*” se lanzan dos eventos: *setTool* y *editorSelectTool*, como vemos en las líneas 18 y 19 del código del apartado anterior. Estos eventos activan una serie de acciones para cambiar de la herramienta de selección que está activa por defecto a la “*POM TOOL*” y así activar el modo de edición en la ventana gráfica y poder crear los puntos de medida en tiempo real.

En el código 3.2 se muestra la clase *PomTool*, que interactúa con el *canvas* 2D de SEDDI (*SceneViewport2*). Para que la herramienta siempre esté actualizada se utilizan las funciones *onAnimationTick* y *onUpdate*.

La función *onAnimationTick* que se muestra en la línea 15 del siguiente código, se encarga de actualizar diferentes elementos de la escena 2D en cada fotograma de la animación de la web, lo que significa que se ejecuta en cada paso de la animación para realizar ajustes dinámicos en la visualización.

Durante la ejecución de *onAnimationTick*, se realizan una serie de tareas importantes para mantener la coherencia y la respuesta en tiempo real de la visualización, lo que permite gestionar las interacciones del usuario. Al detectar los movimientos del cursor, podemos actualizar la posición y el estado de los diferentes elementos presentes en la escena 2D. Dentro de esta función se ejecuta *executeBehaviors*, línea 27, que se encarga de mantener actualizado el estado de la selección de los elementos del *canvas*. Por otro lado, dentro de esta función también se encuentra *getPOMShapes*, líneas 28, 55-73, que se encarga de dibujar la línea y los puntos anaranjados que aparecen cuando seleccionas el primer punto del POM y mueves el ratón. Para facilitar la definición del segundo punto, *getPointWithSnap2*, líneas 41-53, se encarga de ajustar las coordenadas del cursor del usuario para que coincidan con las coordenadas del primer punto del POM en el caso de que esté lo suficientemente cerca de su coordenada *x* o *y*, para poder facilitar el trazado de líneas totalmente rectas para definir tanto POMs horizontales como verticales.

```

1 [...]
2 export class PomTool {
3   constructor({ ... }: ToolOptions) {
4     this.onUpdate({
5       camera,
6       viewport,
7       actions,
8       scene,
9       selection,
10      });
11    }
12    onUpdate(updateArgs: Partial<ToolOptions>) { ... }
13    onAnimationTick(inputArgs: OnAnimationTickArgs) {
14      const {
15        ...
16      }

```

Capítulo 3. Análisis y Desarrollo

```
17     camera: { scale },
18     actions,
19     selection,
20     pieces,
21     scene,
22     state,
23     ...
24 } = this;
25 const { pointer } = inputArgs;
26 const toolState: PatternState = {[...]};
27 this.executeBehaviors(toolState, pointer);
28 this.shapes.push(...this.getPOMShapes(scene, pointer));
29 [...]
30 this.setCursor(toolState);
31 const update = this.needsUpdate;
32 this.needsUpdate = false;
33 return update;
34 }
35
36 private executeBehaviors(toolState: PatternState, pointer: PointerFrame)
37 {
38     ...
39     newSelection = this.selectBehaviour.state.selection;
40 }
41
42 private getPointWithSnap2(pointer, point2, scale): IVector2
43 {
44     const IsSnappingToCoordinateX = Math.abs(pointer.x - point2.x) <= scale;
45     const IsSnappingToCoordinateY = Math.abs(pointer.y - point2.y) <= scale;
46
47     if (IsSnappingToCoordinateX) {
48         point.x = point2.x;
49     }
50     if (IsSnappingToCoordinateY) {
51         point.y = point2.y;
52     }
53     return point;
54 }
55
56 private getPOMShapes(scene: any, pointer: any) {
57     if (this.punto1) { const cursorSnap = this.getPointWithSnap2(...); }
58
59     const shape = new Path2D();
60     shape.moveTo(this.punto1.x, this.punto1.y);
61     shape.lineTo(cursorSnap.x, cursorSnap.y);
62
63     shapes.push(
64     {
65         path: UnitCircle, position: ..., fill: '#FFFFFF', stroke: '#ffa500'
66     },
67     {
68         path: shape, stroke: '#ffa500', with: 2,
69     },
70     {
71         path: UnitCircle, position: ... fill: '#ffa500', stroke: '#ffa500',
72     },
73 );
74
75     return shapes;
76 }
77 [...]
78 }
```

Código 3.2: PomTool

3.3.2. *onUpdate*

La función *onUpdate* se activa en el constructor de *PomTool*. Esta función se ejecuta cuando se detectan cambios en diferentes variables, como modificaciones en la escena o en la cámara, así como cuando se realiza una selección de un elemento.

En el código 3.3 se muestra la funcionalidad implementada en *onUpdate*. Cuando se detecta que ha cambiado algún atributo de la escena se actualiza todo lo necesario. Nos centraremos en el caso en el que se identifica la selección de un elemento, primero se comprueba que la selección no está duplicada, línea 9, que se refiere a que el usuario ha hecho una nueva selección, y posteriormente se evalúa el valor de *hit*. *Hit*, línea 16, es una estructura de datos que contiene información sobre la colisión entre el cursor y el *path* de Bézier de la prenda, obteniendo el punto de impacto exacto.

Cuando haces *click* en un segmento de la prenda para seleccionar el primer punto del POM, se evalúa el punto de impacto *hit* y se identifica la pieza, el *path* de Bézier y el parámetro *t* de la curva que corresponde con ese *hit*. Esto nos sirve para definir el punto de medida en función del patrón de la pieza seleccionada, líneas 32-33. De esta forma obtenemos las coordenadas *x,y* del primer punto de medida.

Para visualizar el POM en el *canvas 2D* se utilizan líneas internas. Como SEDDI ya cuenta con una función que permite dibujarlas, para facilitar el pintado de esas líneas internas (*interiorLines*), las coordenadas del punto se pasan a coordenadas globales del *canvas 2D*, esto se hace porque las coordenadas del punto son en coordenadas locales (del patrón) y para dibujarlas es necesario que estén en coordenadas globales. Tenemos que tener cuidado en varias partes del proyecto de asegurarnos que las operaciones están en el mismo sistema de coordenadas. Se implementó la función *ToGlobalCoordinates*, línea 34, que aplica la transformación de la prenda (rotación y traslación) al punto para obtener su posición en el sistema de coordenadas global y la función *ToLocalCoordinates* que hace la transformación inversa.

Para terminar, se guarda el valor de este punto en *pomPostions*, y cuando se detecte un nuevo segmento o punto seleccionado (segundo punto del POM), se analiza su punto de impacto, se define respecto a la prenda y se guarda igual que para el primero, pero además se lanza la acción *addPOMInteriorLine()*, líneas 39-47, ya que un POM se define con dos puntos. Esta función crea la línea interior y guarda el punto de medida en los datos, esta función se describe con más detalle en el siguiente subapartado.

Por otro lado, si al hacer una nueva selección existen posiciones de puntos de medida previas (el primer punto ya está definido), para obtener el valor correcto del segundo punto se comprueba la posición del cursor *getPointWithSnap2*, línea

13, que si está dentro de la tolerancia definida, ajusta la posición del puntero de acuerdo a si queremos una línea horizontal o vertical perfecta igual que se utiliza al pintar la línea.

Cuando se selecciona un nuevo punto y ya hay dos puntos guardados, ya se ha creado un POM, se inicializan los valores de todos los parámetros de forma que este punto sea el primer punto de un nuevo POM, esto se realiza en las líneas 24-30.

```

1  onUpdate(updateArgs: Partial<ToolOptions>) {
2      const { state, editionMode, entities, scene } = updateArgs;
3      const selectionChanged = this.selection !== state?.selection;
4      const sceneChanged = this.scene !== scene;
5      const pieces = scene ? Garment.Pieces(scene) : [];
6      this.updateToolState(pieces, updateArgs);
7      [...]
8
9      const isDuplicate = points.some( (existingSelection) =>[..]);
10     if (!isDuplicate && this.selection && this.selection.length > 0) {...}
11     if (pomPostions.length > 0) {
12         const pathpos0 = [{ pomPostions[0].x, pomPostions[0].y }];
13         newpoint = this.getPointWithSnap2(sharedPointer.worldspace,...);
14     }
15
16     hit = this.collider.cast(newpoint, hit, tolerance);
17     if (!hit) { hit = undefined; }
18     const path = new Path2(hit.line.path);
19     const to = hit.segment.to;
20     let from = hit.segment.from;
21     ...
22     const path1 = path.split(from, hit.t);
23
24     if (points.length >= 2) {
25         const poms = scene.pointsOfMeasure;
26         points = [];
27         pomPostions = [];
28         hitOffsets = [];
29         this.punto1 = null;
30     }
31
32     const selectionPOM = scene.find((piece) => piece.id === this.selection[0].piece);
33     const pos = selectionPOM...sewLine.path[this.selection[0].nodes.sewLine[0]];
34     const puntoMundo = this.ToGlobalCoordinates( path1.path[1].position,
35     selectionPOM.components.transform2D );
36     pomPostions.push(puntoMundo);
37
38     if (pomPostions.length >= 2) {
39         const positionPOM = [ ... ];
40         actions.addPOMInteriorLine(
41             this.selection[0].piece,
42             positionPOM,
43             false,
44             points,
45             this.unitSystem,
46             hitOffsets
47         );
48     }

```

Código 3.3: OnUpdate PomTool

3.3.3. *addPOMInteriorLine* y *addPointOfMeasure*

Al haber seleccionado dos puntos, se lanza la acción *addPOMInteriorLine*. Esta función dibuja una línea interna gris en el patrón y calcula la medida de la línea en centímetros (*segmentPathLength*, línea 17 del código 3.4), para que se visualice en la ventana gráfica del patrón 2D. A continuación, se estudia la orientación de la línea formada por los puntos de medida y se indica con valores 0, 1 y 2, correspondientes a si la orientación de la línea es horizontal, vertical o diagonal, respectivamente.

```

1 export function addPOMInteriorLine(...)

2 {
3   return (dispatch: AppDispatch, getState: AppGetState) => {
4     const { scene: { data: { children } }, } = getState();
5     const entity = children.find(({ _id }) => _id === entityID);
6     const interiorLines = cloneDeep(entity.components.garmentPiece.interiorLines
7   );
8     const interiorLinesPath = GarmentPieceLine.CreateWithPointTypes({
9       path: SpaceUtils.transformPathToLocalSpace(entity.components.transform2D,
10      path),
11    });
12   const POMNewId = BsonUtils.ID();
13   const interiorLineNewId = BsonUtils.ID();
14   interiorLines.push(...);
15   dispatch(
16     updatePieces(...),
17   );
18   const segmentPath = new Path2(interiorLinesPath.path);
19   const segmentPathLength = Number(SegmentLabelUtils.GetPathLength(segmentPath
20     , unitSystem, 1, SCALES.m)) / 2;
21   ...
22   const dx = Math.abs(p0.x - p1.x);
23   const dy = Math.abs(p0.y - p1.y);
24   if (dx <= tolerance) {
25     orientation = 1;
26   } else if (dy <= tolerance) {
27     orientation = 0;
28   } else {
29     orientation = 2;
30   }
31   const pathPOM = new Path2(entity.components.garmentPiece.sewLine.path);
32   const path0 = pathPOM.split(hit[0].from, hit[0].hit_t);
33   const path1 = pathPOM.split(hit[1].from, hit[1].hit_t);
34   const puntoMundo0 = path0.path[1].position;
35   const puntoMundo1 = path1.path[1].position;
36   let positionPOM = new Path2( [ { position: { x: puntoMundo0.x, ... } } ] );
37   dispatch(
38     addPointOfMeasure(POMNewId, points, {
39       isDrawing: true,
40       sliderValue: segmentPathLength,
41       interiorLineId: interiorLineNewId,
42       garmentPieceId: entity._id,
43       hit: hit,
44       orientation: orientation,
45       position: positionPOM,
46     }),
47   );
48 }

```

Código 3.4: addPOMInteriorLine

Al terminar de crear la línea interior en *addPOMInteriorLine*, se lanza la acción *addPointOfMeasure*, líneas 35-43 del código anterior (3.4), que añade a los datos del estado global de *Redux* toda la información del punto de medida creado en la aplicación. Cabe destacar que, como para la herramienta paramétrica las operaciones son a nivel local, los puntos de medida se guardan en coordenadas locales del patrón. La acción que vemos en el código 3.5: *addPointOfMeasure*, líneas 1-11, utiliza el reducer de los datos *SCENE_ADD_POINT_OF_MEASURE*, líneas 15-38, para crear el POM con todas sus propiedades en el estado global.

```

1 // ACTION
2 export const addPointOfMeasure = (_id, POM, additionalValues = {}) => {
3   return {
4     type: 'SCENE_ADD_POINT_OF_MEASURE',
5     payload: {
6       _id,
7       POM,
8       ...additionalValues,
9     },
10   };
11 };
12
13 // REDUCER
14 [...]
15 case types.SCENE_ADD_POINT_OF_MEASURE: {
16   const payload = action.payload as PointOfMeasurePayload;
17   const existingPointsOfMeasure = Array.isArray(state.pointsOfMeasure)
18     ? state.pointsOfMeasure
19     : [];
20   return {
21     ...state,
22     pointsOfMeasure: [
23       ...existingPointsOfMeasure,
24       {
25         _id: payload._id,
26         POM: payload.POM,
27         isDrawing: payload.isDrawing || false,
28         interiorLineId: payload.interiorLineId || null,
29         type: payload.type || '',
30         connected: payload.connected || false,
31         gradding: payload.gradding || false,
32         sliderValue: payload.sliderValue || 0,
33         garmentPieceId: payload.garmentPieceId || null,
34         hit: payload.hit !== undefined ? payload.hit : 0,
35         orientation: payload.orientation || 0,
36         position: payload.position || null,
37       },
38     ],
39   }
40 }

```

Código 3.5: Acción *addPointOfMeasure* y su reducer

Al actualizar el valor de la variable global de los puntos de medida (*pointsOfMeasure*), se actualiza automáticamente el *parametricPanel* y se crea el *Slider-Component*, que permite la edición paramétrica. Este sería el POM actualizado en el *Store* del estado global:

3.3. Desarrollo de la herramienta de creación de puntos de medida

```

1 pointsOfMeasure:
2 - Array(1):
3   - 0:
4     - POM:
5       - connected: false
6       - garmentPieceId: "6601616cc3a445d4d1536856"
7       - gradding: false
8       - hit:
9         - Array(2):
10        - 0:
11          - from:
12            segment: 2
13            offset: 0
14          - hit_t:
15            segment: 2
16            offset: 0.5201904059767545
17        - 1:
18          - from:
19            segment: 4
20            offset: 0
21          - hit_t:
22            segment: 4
23            offset: 0.485474231038235
24       - interiorLineId: "66676ca6ae1d7f639119ba01"
25       - isDrawing: true
26       - orientation: 0
27       - position: Path2
28         -0:position:{x: -0.14851548157951627, y: 0.2873494540069362}
29         -1: position: {x: -0.14606609306547552, y: -0.28734833836503343}
30       - sliderValue: 57.45
31       - type: ""
32       - _id: "66676ca6ae1d7f639119ba00"

```

Código 3.6: Puntos de medida en el estado global

En la Figura 3.7 se muestra el funcionamiento de la *POM Tool* al seleccionar los dos puntos en el panel frontal de una camiseta y vemos cómo se crea una línea interior gris con el valor de la distancia entre los puntos de medida en centímetros. Esta herramienta permite crear puntos de medida que van desde un punto o segmento de la prenda hasta otro punto o segmento, y visualizar la distancia entre ellos. Además, se facilita la creación de puntos horizontales y verticales exactos, así como crear puntos de medida diagonales.



Figura 3.7: Ejemplo creación de un punto de medida horizontal

3.4. Desarrollo de la herramienta de edición paramétrica de patrones en SEDDI Author

Una vez definidos los POMs con la *POM TOOL*, podemos ver en la Figura 3.8 que se ha creado la línea interna en el patrón y que en las propiedades paramétricas aparece un nuevo componente para poder modificar las propiedades del punto de medida, el *SliderComponent*.

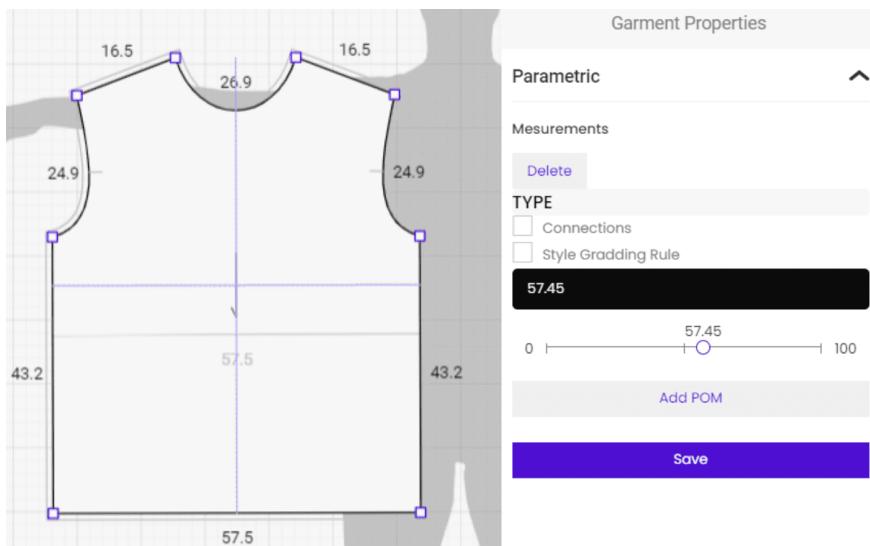


Figura 3.8: Punto de medida en la interfaz de usuario

Este *SliderComponent* permite la edición de los POMs al cambiar el valor de la distancia entre los puntos de medida con un control deslizante o a través de entrada de teclado, eliminar el POM y establecer conexiones entre puntos de medida dependiendo del tipo y la casilla de verificación de conexión. En el *Parametric Panel* se crea un *SliderComponent* por cada POM guardado en los datos del estado global y están identificados por el índice del POM.

Al interactuar con los botones se lanzan diferentes eventos con *useEffect* como vemos en las líneas 12 o 24 del código 3.7. *useEffect* se utiliza para ejecutar código adicional después del renderizado del componente, se ejecuta cuando cambian ciertas dependencias. En este caso, las dependencias son *scene*, *pomIndex*, *sceneRef.current.data.pointsOfMeasure*, *sliderValue*, *inputTypeValue*, *isCheckedConnected*, etc. Si alguna de estas dependencias cambia, se ejecutará nuevamente el código dentro de *useEffect*. Dentro de este evento, se actualizan los estados locales, como el *inputValue*, el *sliderValue* o *isCheckedConnected*, con los valores actuales de las propiedades de la escena, y también se actualiza el estado global con el método *dispatch* para lanzar las acciones *deletePointOfMeasure*, para borrar los puntos de medida, *updatePOMInteriorLines*, para actualizar las líneas internas cuando se cambia la distancia del POM o *updatePointOfMeasureProperty*

para actualizar las propiedades del POM como la conexión con otros POMs, el valor del deslizador o el tipo en el estado global. Esto se hace para mantener sincronizados los valores del estado local (del componente) con los valores de la escena (el estado global de la aplicación en *Redux*).

```

1 const SliderComponent = ({  

2   pomIndex,  

3   initialValue,  

4   scene,  

5   onDelete,  

6   updateSliders,  

7   onPomIndexChange,  

8 }) => {  

9   const dispatch = useDispatch();  

10  ...  

11  useEffect(() => {  

12    ... dispatch(updatePOMInteriorLines(sceneRef.current.data));  

13  }, [  

14    dispatch,  

15    pomIndex,  

16    inputValue,  

17    inputTypeValue,  

18    sliderValue,  

19    isCheckedConnected,  

20    isCheckedGrading,  

21  ]);  

22  

23  useEffect(() => {  

24    ... dispatch(deletePointOfMeasure(pointsOfMeasure[pomIndex].id));  

25  }, [dispatch, pomIndex, sceneRef.current.data.pointsOfMeasure]);  

26  

27  

28  const handleCheckboxChange = (checkboxName) => {  

29    if (checkboxName === 'connected') {  

30      ... dispatch(updatePointOfMeasureProperty(connectedIndex, 'connected', !  

31      isCheckedConnected));  

32    };  

33  const handleInputonBlur = (e) => {  

34    ... dispatch(updatePointOfMeasureProperty(pomIndex, 'sliderValue', newValue))  

35    ;  

36  };  

37  

38  const handleInputTypeonBlur = (e) => {  

39    ... dispatch(updatePointOfMeasureProperty(pomIndex, 'type', e.target.value));  

40  };  

41  

42  const handleSliderChange = (value) => {  

43    ... dispatch(updatePointOfMeasureProperty(pomIndex, 'sliderValue', value));  

44    updateSliders(pomIndex);  

45    onPomIndexChange(pomIndex);  

46  };  

47  

48  const handleDeleteClick = () => {  

49    onDelete(pomIndex);  

50  };  

51  

52  return (  

53    <div>  

54      <Button onClick={handleDeleteClick}>Delete</Button>  

55      <Input  

56        name="type",
```

Capítulo 3. Análisis y Desarrollo

```
57     type='text',
58     inputStyle='settings',
59     placeholder='Type',
60     onBlur={handleInputTypeOnBlur}
61     value={inputTypeValue}
62   />
63   <Checkbox
64     name='connected',
65     label='Connections',
66     isSelected={isCheckedConnected}
67     onCheckboxChange={() => handleCheckboxChange('connected')}
68     disabled={false}
69   />
70   <Checkbox
71     name='grading',
72     label='Style Gradding Rule',
73     isSelected={isCheckedGrading}
74     onCheckboxChange={() => handleCheckboxChange('grading')}
75     disabled={false}
76   />
77   <Input
78     name='name',
79     type='number',
80     placeholder='cm',
81     onBlur={handleInputOnBlur}
82     value={inputValue}
83   />
84   <RangeSlider
85     breakpoints={[0]}
86     min={0}
87     max={100}
88     value={sliderValue}
89     step={0.0001}
90     disabled={false}
91     onChange={handleSliderChange}
92   />
93   </div>
94 );
95 };
```

Código 3.7: SliderComponent

Para actualizar las propiedades se llama a la acción *updatePointOfMeasureProperty*, líneas 1-7 del código 3.8, que indicándole el valor de la propiedad al *reducer UPDATE_POINT_OF_MEASURE_PROPERTY*, líneas 16-24, actualiza solo ese valor en el estado global. Para eliminar los puntos de medida se lanza la acción *deletePointOfMeasure*, líneas 8-12, y el *reducer DELETE_POINT_OF_MEASURE*, líneas 25-30, elimina el punto de medida con el id correspondiente al POM que queremos borrar.

```
1 // ACTIONS
2 export const updatePointOfMeasureProperty = (index, property, value) => {
3   return {
4     type: 'UPDATE_POINT_OF_MEASURE_PROPERTY',
5     payload: { index, property, value },
6   };
7 };
8 export const deletePointOfMeasure = (_id) => {
9   return {
10     type: 'DELETE_POINT_OF_MEASURE',
11     payload: { _id },
12   };
13 }
```

```

12     };
13     [...]
14 // REDUCERS
15 [...]
16 case types.UPDATE_POINT_OF_MEASURE_PROPERTY: {
17     const payload = action.payload as PointOfMeasurePropertyPayload;
18     return {
19         ...state,
20         pointsOfMeasure: state.pointsOfMeasure.map((point, i) =>
21             i === payload.index ? { ...point, [payload.property]: payload.value }
22             : point,
23         ),
24     };
25 }
26 case types.DELETE_POINT_OF_MEASURE: {
27     const payload = action.payload as PointOfMeasureDeletePayload;
28     return {
29         ...state,
30         pointsOfMeasure: state.pointsOfMeasure.filter((pom) => pom._id !==
31             payload._id),
32     };
33 }
```

Código 3.8: Acción addPointOfMeasureProperty y su reducer

Este componente de *React* tiene que estar constantemente actualizándose si se cambia algún parámetro del POM. Al detectar un cambio en la herramienta paramétrica (que se explicará en el siguiente apartado), se llama a *updatePOMInteriorLines*, que itera sobre cada punto de medida de la escena y llama a la función *updatePOMInteriorLine*, que, a su vez, llama a *updatePieces* para actualizar la pieza con los puntos de medida en su nueva posición, asegurando así que la línea interior y su medida se dibujen correctamente. Podemos ver ambas funciones en el código 3.9.

```

1 export function updatePOMInteriorLines(scene: ISceneAppWithPoms) {
2     return (dispatch: AppDispatch) => {
3         if (scene.pointsOfMeasure) {
4             const poms = scene.pointsOfMeasure;
5             for (let i = 0; i < poms.length; i++) {...}
6                 dispatch(updatePOMInteriorLine(
7                     poms[i].POM[0][0].piece,
8                     positionPOM,
9                     false,
10                    poms[i].interiorLineId,
11                ), ...
12            });
13        export function updatePOMInteriorLine(...){}
14        return (dispatch: AppDispatch, getState: AppGetState) => {
15            const { scene: { data: { children }, }, } = getState();
16            if (existingInteriorLine) {...}
17                dispatch(updatePieces(
18                    [{ _id: entity._id,
19                      holeLines: entity.components.garmentPiece.holeLines,
20                      interiorLines: interiorLines,
21                      sewLine: entity.components.garmentPiece.sewLine,
22                      transform2D: entity.components.transform2D,
23                  }, ], false, ), ...
24        });

```

Código 3.9: updatePOMInteriorLines y updatePOMInteriorLine

3.4.1. *Parametric Editing Tool*

Al crear el POM y pulsar el botón de “Save” se activa la herramienta de edición paramétrica “*ParametricEditingTool*”. Esta herramienta también es una clase de *TypeScript* que utiliza *onUpdate* y *onAnimationTick* para mantener actualizada la ventana gráfica según los cambios en el estado y la interacción del usuario.

En este caso, *onUpdate* se encarga de verificar si ha habido cambios relevantes en el estado, como en la *POM TOOL*, como cambios en la selección, en la escena, etc. Y lo más importante, si hay cambios en los puntos de medida (*pomsChanged*), estos cambios se detectan si en el estado global se ha modificado alguna propiedad de los puntos de medida en el *Parametric Panel* (*SliderComponent* de cada *POM*). Si existen cambios se activa la función *ParametricEditing*, línea 15 del código 3.10, para realizar la edición paramétrica correspondiente.

```

1 export class ParametricEditingTool {
2   constructor({...
3   }: ToolOptions) {...}
4   this.onUpdate({
5     camera,
6     viewport,
7     actions,
8     scene,
9     selection,
10    });
11  }
12 onUpdate(updateArgs: Partial<ToolOptions>) {
13   ...
14   if (pomsChanged) {
15     this.ParametricEditing(this.scene);
16   }
17 }
18 onAnimationTick(inputArgs: OnAnimationTickArgs) {
19   const {
20     camera: { scale },
21     actions,
22     selection,
23     pieces,
24   } = this;
25   const { pointer } = inputArgs;
26   ...
27   this.setCursor(toolState);
28   const update = this.needsUpdate;
29   this.needsUpdate = false;
30   return update;
31 }
32 [...]
33 }
```

Código 3.10: ParametricEditingTool

La función *ParametricEditing* que se muestra en el código 3.11 se encarga de ajustar la posición de los puntos de medida y los nodos que componen el patrón en función de la longitud deseada (especificada por el usuario mediante el valor del deslizador o entrada de teclado del POM).

Primero, la función recorre todos los puntos de medida presentes en la escena. Para cada POM, se identifica la pieza de ropa y la línea interior correspondiente, línea 5. Luego, se calcula la diferencia entre la longitud actual de la línea interior y la longitud deseada, líneas 6-9. Si existe una diferencia de longitud, es decir, el valor del POM ha sido modificado, se calculan los desplazamientos necesarios en los ejes *X* e *Y* (*CalcularTransformación*, líneas 12-21). Finalmente, se actualizan los nodos de la curva de Bézier del patrón correspondientes a los POMs con las transformaciones calculadas para reflejar este ajuste. Los nodos pueden representar puntos o segmentos del patrón. Si un *hit* tiene el parámetro *t*, significa que se ha seleccionado un punto a lo largo de la línea del patrón; al modificar el POM que contiene este punto, se ajustará todo el segmento correspondiente. Dependiendo de la orientación del punto de medida, se aplicarán transformaciones a uno (si el POM es vertical) o a dos (si es horizontal o diagonal) nodos que corresponden con el POM. Estas operaciones permiten mover las curvas de Bézier que definen el patrón y modificar automáticamente la malla 2D y 3D que compone la prenda dentro de Author, utilizando la acción *updatePathNodes*.

```

1  private ParametricEditing(scene: ISceneAppWithPoms) {
2      ...
3      if (poms) {
4          for (let i = 0; i < poms.length; i++) {
5              ...
6              if (garmentPieceObject && interiorLineObject) {
7                  const segmentPath = new Path2(interiorLineObject.path);
8                  const longitudActual = GetPathLength(segmentPath, this.unitSystem)/2;
9                  const longitudDeseada = poms[i].sliderValue;
10                 const diferenciaLongitud = longitudActual - longitudDeseada;
11
12                 if (diferenciaLongitud != 0) {
13                     const [transformNodesFn, transformNodesFn2] =
14                         this.calcularTransformacion(
15                             poms[i].orientation,
16                             0.01,
17                             diferenciaLongitud,
18                             garmentPieceObject,
19                             transform,
20                             transform2,
21                             segmentPath,
22                             );
23                     if (poms[i].orientation == 0 || poms[i].orientation == 2) {
24                         this.actions.updatePathNodes(poms[i].POM[0], transformNodesFn);
25                         this.actions.updatePathNodes(poms[i].POM[1], transformNodesFn2);
26                     } else if (poms[i].orientation == 1) {
27                         this.actions.updatePathNodes(poms[i].POM[1], transformNodesFn2);
28                     }
29
30                     if (poms[i].connected) {
31                         [...]
32                         // POMS CONECTADOS
33                     }
34                 }
35             }
36         }
37     }
}

```

Código 3.11: ParametricEditing

Como se ha comentado anteriormente, en el código 3.12 podemos observar la función *CalcularTransformación* que tiene como objetivo calcular la transformación de los nodos del patrón para ajustar la distancia deseada del punto de medida. Dado que la diferencia de longitud debe distribuirse de manera equitativa, primero se calcula el “ajusteMitad” (línea 2), que representa la mitad de la diferencia de longitud. Este ajuste se aplicará a cada punto del POM.

Para los POMs verticales, el ajuste es la diferencia completa (línea 5). Los puntos del POM deben estar definidos de arriba abajo para que las transformación solamente edite las correspondencias del segundo punto. El desplazamiento en el eje *X* es 0 (línea 14) y en el eje *Y* es la diferencia total de longitud. Así, el segundo punto del POM se mueve hacia abajo, alargando la prenda en esa dirección. Para los POMs horizontales, el desplazamiento en el eje *Y* es 0 (línea 12). El primer punto del POM se moverá la mitad de la diferencia hacia la izquierda, el segundo la mitad de la diferencia hacia la derecha. Dependiendo de si la diferencia es positiva o negativa, la prenda se encogerá o se expandirá. Es importante que los puntos del POM horizontales y diagonales estén definidos de izquierda a derecha para que las transformaciones locales tengan sentido. Para los POMs diagonales, el ajuste sigue la dirección del vector director unitario de la línea que define el POM, asegurando que el movimiento mantenga la dirección original de la línea.

Finalmente, se calculan las transformaciones con los desplazamientos en *X* e *Y* usando la función *GetTransform*, líneas 34-35. *GetTransform* asigna cada transformación a los nodos de las curvas de Bézier, incluyendo tanto las posiciones de los puntos como los puntos de control que definen las curvas, líneas 40-58. Estas transformaciones permiten que, dependiendo de la orientación y la diferencia de longitud, los nodos se ajusten adecuadamente para reflejar el cambio en el POM. Al editar los nodos de las curvas de Bézier se modifican la malla 2D y 3D de la prenda automáticamente dentro de Author.

```

1 private calcularTransformacion (...): [NodeTransformFn, NodeTransformFn] {
2     let ajusteMitad: number = (factorConversion * diferenciaLongitud) / 2;
3
4     if (orientacion === 1) {
5         ajusteMitad = factorConversion * diferenciaLongitud ;
6     }
7     let desplazamientoX: number = ajusteMitad ;
8     let desplazamientoY: number = ajusteMitad ;
9
10    if (orientacion === 0) {
11        desplazamientoY = 0;
12    } else if (orientacion === 1) {
13        desplazamientoX = 0;
14    } else if (orientacion === 2) {
15        ...
16        const dx = p2.x - p1.x;
17        const dy = p2.y - p1.y;
18        const dist = Math.sqrt(dx * dx + dy * dy);
19
20        const ux = Math.abs(dx / dist);
21        const uy = dy / dist;
22
23        const direccionOriginal = { x: ux, y: uy };

```

```

24     desplazamientoX = direccionOriginal.x * ajusteMitad;
25     desplazamientoY = direccionOriginal.y * ajusteMitad;
26 }
27 ...
28 sliderInicio.set(-desplazamientoY, -desplazamientoX);
29 transformacionInicio.setTranslation(sliderInicio ...);

30
31 sliderFinal.set(desplazamientoY, desplazamientoX);
32 transformacionFinal.setTranslation(sliderFinal ...);

33
34 const transformacionNodoFnInicio = this.getTransformFn(transformacionInicio)
35 const transformacionNodoFnFinal = this.getTransformFn(transformacionFinal)

36
37 return [transformacionNodoFnInicio, transformacionNodoFnFinal];
38 }

39
40 private getTransformFn(transform: TransformRT2) {
41     const transformNodesFn: NodeTransformFn = (node, prevNode, entity) => {
42         const { components: { transform2D: { rotation } }, ... } = entity;
43         const updatedNode = { ... node };
44         const updatedPrevNode = { ... prevNode };
45         updatedNode.position.x += transform.translation.x;
46         updatedNode.position.y += transform.translation.y;

47         if (updatedNode.control1) {
48             updatedNode.control1.x += transform.translation.x;
49             updatedNode.control1.y += transform.translation.y;
50         }
51         if (updatedPrevNode.control2) {
52             updatedPrevNode.control2.x += transform.translation.x;
53             updatedPrevNode.control2.y += transform.translation.y;
54         }
55     }
56
57     return { node: updatedNode, prevNode: updatedPrevNode };
58 };
59
60     return transformNodesFn;
61 }
62 }
```

Código 3.12: calcularTransformación y getTransformFn

3.4.2. Conectividad entre patrones

Si al modificar el valor de un punto de medida este está conectado a otros puntos de medida, es decir, ambos POMs tienen el mismo tipo, el parámetro de entrada “*Type*”, y está activa la caja de verificación “*Connections*”, también se deberá ajustar la longitud de los POMs conectados.

En *ParametricEditing*, al recorrer cada punto de medida editado, si el valor de “*connected*” es verdadero, línea 6 del código 3.13, es decir, el usuario ha activado la casilla de “*Connections*”, se recorren los demás puntos de medida que tienen el mismo tipo que el POM editado para comprobar qué POMs hay que modificar. La conexión entre los puntos de medida busca mantener una relación proporcional entre sus longitudes en lugar de simplemente igualar sus valores.

Para asegurar que los POM conectados mantengan una relación proporcional con el POM editado, se calcula una proporción basada en la relación entre la longitud deseada actual del POM editado y su longitud actual, línea 5. Esta proporción es la división de la longitud deseada del POM editado entre su longitud actual. A continuación, esta proporción se utiliza para determinar la nueva longitud de los POMs conectados, multiplicando la longitud actual del POM conectado por esta proporción, líneas 13-14. Luego se calcula la nueva transformación, líneas 16-20, y se aplican las transformaciones correspondientes teniendo en cuenta la orientación, líneas 21-26. De esta manera, los POMs conectados se ajustan en relación con los cambios realizados en el POM editado, manteniendo una proporción coherente.

```

1  private ParametricEditing(scene: ISceneAppWithPoms) {
2      if (scene.pointsOfMeasure) {
3          for (let i = 0; i < poms.length; i++) {
4              [...]
5              const proporcion = longitudDeseada / longitudActual;
6              if (poms[i].connected) {
7                  for (let j = 0; j < poms.length; j++) {
8                      if (j !== i && poms[j].type === poms[i].type) {...}
9                      if (garmentPieceObject && interiorLineObject) {
10                          const segmentPath = new Path2(interiorLineObject.path);
11                          const longitudActual2 = GetPathLength(...) / 2;
12                          const longitudDeseada2 = poms[j].sliderValue * proporcion;
13                          const diferenciaLongitud2 =
14                              longitudActual2 - longitudDeseada2;
15                          const [transformNodesFn, transformNodesFn2] =
16                              this.calcularTransformacion(
17                                  poms[j].orientation,
18                                  diferenciaLongitud2,
19                                  ...
20                              );
21                          if (poms[j].orientation === 0 || poms[j].orientation === 2) {
22                              ...updatePathNodes(poms[j].POM[0], transformNodesFn);
23                              ...updatePathNodes(poms[j].POM[1], transformNodesFn2);
24                          } else {
25                              ...updatePathNodes(poms[j].POM[1], transformNodesFn2);
26                          }
27                          ...
28          }

```

Código 3.13: ParametricEditing

En este vídeo² se puede ver cómo funciona tanto la herramienta de creación como de edición de puntos de medida y conexiones, así como las ventajas de utilizar la edición paramétrica frente a una edición manual de la prenda. Como se puede comprobar, la edición paramétrica es más precisa y rápida. En el apartado de resultados se analiza más en detalle el funcionamiento de estas herramientas.

El escalado con *grading rules* propuesto por los diseñadores en el diseño de la herramienta no se ha implementado en esta primera propuesta de edición paramétrica. En su lugar, se optó por realizar el estudio de la preservación del estilo de la prenda entre avatares teniendo en cuenta la edición paramétrica.

²<https://www.youtube.com/watch?v=oJnXdKj0yNE>

3.5. Problema de optimización: preservación del estilo de la prenda entre avatares

El diseño de la optimización se basa en la propuesta de Brouet et al.[32], aunque de una manera más simplificada, prescindiendo de marcadores antropomórficos y sus distancias respecto a componentes específicas de la prenda (P.Ej.:dobladillos). Nuestra hipótesis es que bastará con minimizar mediante una transformación afín del patrón 2D la diferencia con los triángulos de la prenda 3D, deformados por la modificación en el avatar 3D, para escalar la prenda preservando el estilo respecto a dicho POM. Siguiendo unos pasos similares a Brouet et al.[32], en primer lugar, se calculan las nuevas posiciones 3D de la prenda proporcional al nuevo avatar, en segundo lugar, modificamos esas posiciones 3D para que preserven el diseño y finalmente calculamos el nuevo patrón 2D. A continuación, se describe el diseño y la implementación de esta solución para preservar el estilo de la prenda entre diferentes avatares.

Para agilizar esta optimización, esta implementación se llevó a cabo fuera del entorno de *SEDDI Author* en un cuaderno de *Jupyter*, lo que permitió utilizar *Python* desde el editor y combinar el código, el texto y los resultados en un solo documento. En este enlace³ se puede visualizar toda la implementación.

Para este proyecto, se optó por una optimización basada en la deformación de la longitud de las aristas de la malla 3D de la prenda. El problema a resolver consiste en encontrar el valor óptimo de un punto de medida. Tenemos un avatar de referencia al que la prenda le queda perfectamente ajustada y queremos averiguar cuánto debe valer el punto de medida para que la prenda se ajuste de manera similar a un nuevo avatar, es decir, preservar el estilo de esa prenda en avatares con diferentes medidas antropomórficas. En la Figura 3.9 podemos observar las partes en las que se divide el problema de optimización que se van a explicar en este apartado.

³<https://github.com/martaquintana/2023-tfm-migjrv/blob/main/author-code/Optimization/author-tool-optimization.ipynb>

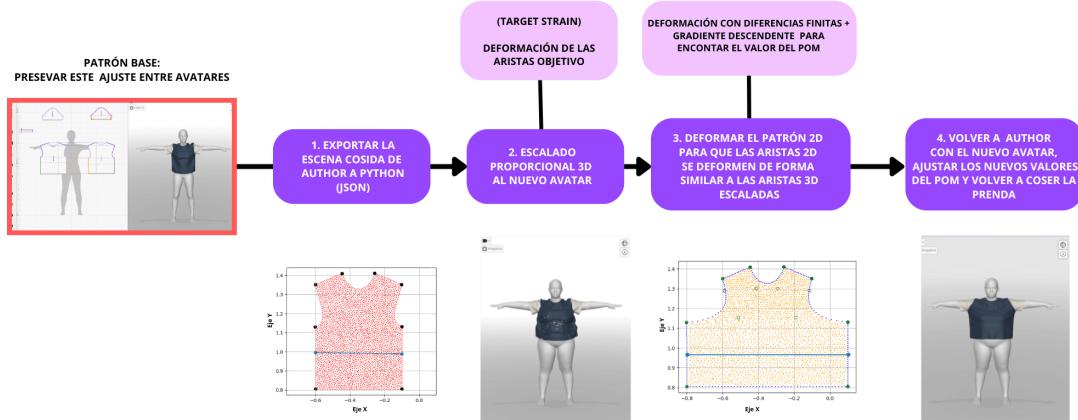


Figura 3.9: Esquema del flujo del problema de optimización

Para ello, primero se extraen los datos de Author en un archivo JSON y se importan a *Python* para analizar y extraer la información de la prenda y el avatar. Con estos datos, se calcula un escalado proporcional de la malla 3D de la prenda en un nuevo avatar. Esto se realiza calculando la distancia entre el avatar de referencia y la malla 3D de la prenda y aplicando esta distancia entre la prenda 3D y el avatar nuevo. Una vez obtenida la prenda 3D en el nuevo avatar, se evalúa cómo han cambiado las aristas de esta malla en comparación con la original. Se calcula el porcentaje de cambio de cada arista, que representa la deformación objetivo que se busca replicar en el patrón 2D. Para determinar el valor óptimo del punto de medida y lograr el ajuste deseado, se define el POM a través de las curvas de Bézier del patrón 2D. Esto permite modificar la malla 2D del patrón al cambiar los puntos de medida y evaluar la función de coste.

Se implementó el método de diferencias finitas para variar gradualmente el valor del POM y encontrar el punto donde converge el error de la función de coste. La función de coste calcula el error cuadrático medio entre la deformación de las aristas de la malla 2D del patrón y la deformación objetivo de la malla 3D de la prenda. De esta manera, se obtiene el valor que se debe ajustar en el POM para preservar el estilo de referencia de la prenda en el nuevo avatar, y así obtener el patrón 2D óptimo para ese nuevo avatar.

Para los resultados obtenidos en el siguiente apartado del proyecto, se definieron dos POMs para establecer el ajuste: uno horizontal y otro vertical, tratados de forma independiente. Se logró determinar el valor óptimo para cada POM en ejecuciones separadas del algoritmo, habría sido interesante estudiar varios POMs simultáneamente para comprender mejor sus interacciones.

A continuación, se detalla el proceso y la implementación de las diferentes partes de este problema de optimización.

3.5.1. Exportación e Importación de datos de la escena

Para extraer los datos del estado global de la aplicación, se añadieron dos nuevos elementos en el *Parametric Panel*: un botón “*Export JSON SCENE*” y un campo de entrada para importar un archivo JSON con las prendas o el avatar modificados. Al hacer *click* en el botón “*Export JSON SCENE*”, se activa una función que gestiona la exportación de los datos del estado global de la escena a un archivo JSON. En este proceso, se asegura que los datos se exporten como *arrays* para que se representen correctamente en el JSON, líneas 3-31 del código 3.14.

Por otro lado, para la importación de un archivo JSON en Author, se implementa un manejador de eventos que procesa el archivo seleccionado por el usuario, línea 35. El contenido del archivo se lee como texto y se analiza para obtener los datos necesarios. Estos datos se convierten en el formato adecuado, como convertir arrays de números en Uint32Array o Float32Array, según sea necesario, líneas 43-51. Después, estos datos se utilizan para actualizar el estado global de la aplicación mediante la llamada a las acciones correspondientes. Por ejemplo, se llama a la acción *updateCreateEntities* en la línea 66, si se necesita actualizar la prenda, o a *dropAvatar*, línea 68, si se agrega un nuevo identificador de avatar. De esta manera, los datos importados del JSON se integran adecuadamente en la aplicación y se actualizan las ventanas gráficas 2D y 3D con los nuevos datos.

```

1 [...]
2 const sceneData = useSelector((state) => state.scene.data);
3 const handleExport = useCallback(() => {
4   const avatar = sceneData.children.find((e) => Boolean(e...avatar));
5   const avatarMesh = {
6     indices: Array.from(avatar.components.mesh.mesh.geometry.indices),
7     positions: Array.from(avatar.components.mesh.mesh.geometry.position),
8   };
9   const pieces = sceneData.children.filter((e) => Boolean(...garmentPiece));
10  const piecesEncoded = pieces.map((e) => {
11    return {
12      components: {
13        garmentPiece: {
14          geometry: {
15            indices: Array.from(...geometry.indices),
16            position2D: Array.from(...geometry.position2D),
17            position3D: Array.from(...geometry.position3D),
18          },
19          sewLine: {
20            sides: e.components.garmentPiece.sewLine.sides.map((s) => ({
21              indices: Array.from(s.indices),
22            })),
23            },
24          },
25        },
26      };
27    });
28  const data = { avatar: avatarMesh, pieces: piecesEncoded, poms: sceneData.
29  pointsOfMeasure };
30  const json = JSON.stringify(data);
31  ...
32 }, [sceneData]);

```

Capítulo 3. Análisis y Desarrollo

```
32
33
34 const handleImport = useCallback(
35   (event) => { ...
36     reader.onload = (e) => {
37       try {
38         const importedState = JSON.parse(e.target.result);
39         const piecesDecoded = importedState.pieces.map((e) => {
40           return {
41             components: {
42               garmentPiece: {
43                 geometry: {
44                   indices: new Uint32Array(...geometry.indices),
45                   position2D: new Float32Array(...geometry.position2D),
46                   position3D: new Float32Array(...geometry.position3D),
47                 },
48                 sewLine: {
49                   sides: e.components.garmentPiece.sewLine.sides.map((s) => ({
50                     indices: new Uint32Array(s.indices),
51                   })),
52                 },
53               },
54             },
55           );
56         });
57         const geometryResult = {
58           pieces: piecesDecoded.map((e) => {
59             return {
60               id: e._id,
61               geometry: e.components.garmentPiece.geometry,
62               sewLine: e.components.garmentPiece.sewLine,
63             };
64           }),
65         };
66         updateCreateEntities({ dispatch, getState, geometryResult, scene });
67         if (importedState.avatar.id) {
68           dispatch(dropAvatar(importedState.avatar.id, false));
69         }
70       } catch (error) {...}
71     };
72     reader.readAsText(file);
73   }, [dispatch, getState]);
74
75 const renderParametricPattern = () => {
76   return (
77     <div>
78       <Button
79         style={...}
80         onClick={() => {
81           handleExport();
82         }} >
83         Export JSON SCENE
84       </Button>
85       <input type='file' accept='.json' onChange={handleImport} />
86
87       <InputGroup name='Mesurements' hr={true}>
88         [...]
89       </InputGroup>
90     </div>
91   );
92 }; ...
93 };
```

Código 3.14: Parametric Panel: handleExport y handleImport

3.5. Problema de optimización: preservación del estilo de la prenda entre avatares

En el código 3.15 se muestra la estructura de JSON exportado. Los datos geométricos que nos interesan exportar del estado global incluyen los índices de los triángulos y las posiciones de los vértices del avatar, líneas 3-4, así como los índices de los triángulos y las posiciones 2D y 3D de los vértices de las piezas del patrón, líneas 13-15. Es importante destacar que tanto la malla 2D como la 3D de las piezas del patrón tienen el mismo número de triángulos. Además, se exportan los datos de los puntos de medida, “*poms*” línea 18.

En este vídeo⁴ se muestra cómo se importan diferentes JSON a SEDDI Author usando esta herramienta.

```
1 {
2   "avatar": {
3     "indices": [...],
4     "positions": [...]
5   },
6   "pieces": {
7     "_id": "662fb24e1e7bd0f81dedb29f",
8     "components": {
9       "garmentPiece": {
10         [...]
11         "geometry": {
12           "indices": [...],
13           "position2D": [...],
14           "position3D": [...]
15         }
16       }
17     }
18   "poms": [
19     {
20       "_id": "662fb3388ba497e6b1aec34e",
21       "POM": [...]
22     }
23   ]
24 }
25 }
```

Código 3.15: JSON exportado

Una vez tenemos los datos en formato JSON, los importamos y procesamos en *Python*. Esto nos permite tener todos los datos listos para realizar la optimización:

```
1 [...]
2 avatar_indices = np.array(avatar['indices'])
3 avatar_positions = np.array(avatar['positions']).reshape(-1, 3) # 3 column
matrix
4
5 pieces_indices = np.array(all_pieces_indices)
6 pieces_positions = np.array(all_pieces_position3d).reshape(-1, 3) # 3 column
matrix
7 pieces_positions2d = np.array(all_pieces_position2d)
8
9 pieces_bezierPath = np.array(all_sewLines)
```

Código 3.16: Datos procesados en Python

⁴<https://www.youtube.com/watch?v=6w9Wm53zq8w>

3.5.2. Escalado proporcional en 3D

Para realizar el escalado proporcional de la prenda 3D para un nuevo avatar, primero tenemos que calcular la distancia entre la prenda 3D original y el avatar de referencia.

En el código 3.17 se calculan las distancias mínimas entre los vértices de la prenda 3D y los del avatar. Se utiliza un bucle para recorrer todos los vértices de la prenda, línea 4, y para cada vértice, se calcula su distancia a todos los vértices del avatar, línea 5. En “ac” (*Avatar Correspondences*), línea 10, almacenamos los índices correspondientes de los vértices del avatar más cercanos a los de la prenda. En las líneas 12-16, calculamos el desplazamiento para cada vértice de la prenda restando las coordenadas del vértice de la prenda con las coordenadas del vértice correspondiente del avatar. Estos desplazamientos resultantes se almacenan en la variable *offsets*.

```

1 min_distances = np.zeros(len(pieces_positions))
2 ac = np.zeros(len(pieces_positions), dtype=int) # Avatar Correspondences
3
4 for i, pos_prenda in enumerate(pieces_positions):
5     dist_vertex = np.linalg.norm(pos_prenda - avatar_positions, axis=1)
6     nearest_indice_vertex = np.argmin(dist_vertex)
7     min_dist = dist_vertex[nearest_indice_vertex]
8
9     min_distances[i] = min_dist
10    ac[i] = nearest_indice_vertex
11
12 offsets = np.zeros_like(pieces_positions)
13
14 for i, indice_avatar in enumerate(ac):
15     offset = np.array(pieces_positions[i]) - np.array(avatar_positions[
16         indice_avatar])
16     offsets[i] = offset

```

Código 3.17: Calcular distancias mínimas y correspondencias de avatar

Después de calcular el vector de desplazamiento, denominado “*offsets*” en el paso anterior, que corresponde con la distancia entre los vértices de la prenda 3D y los vértices más cercanos en el avatar de referencia. Se utilizan estos desplazamientos para calcular las nuevas posiciones de las piezas del patrón 3D en el nuevo avatar, como vemos en el código 3.18. Dado que los avatares en SEDDI tienen la misma estructura de malla, los índices de los vértices en el nuevo avatar coinciden con los del avatar de referencia. Por lo tanto, para cada vértice de la prenda, se suma el desplazamiento, *offset*, correspondiente al vértice del nuevo avatar utilizando el índice obtenido previamente en los *Avatar Correspondences*.

```

1 new_pieces_positions = np.zeros_like(pieces_positions)
2
3 for i, indice_avatar in enumerate(ac):
4     new_pieces_positions[i] = np.array(new_avatar_positions[indice_avatar]) + np.
        array(offsets[i])

```

Código 3.18: Calcular nuevas posiciones de la prenda

3.5. Problema de optimización: preservación del estilo de la prenda entre avatares

Para visualizar que se ha realizado correctamente el escalado proporcional de la prenda, podemos guardar los datos en un nuevo JSON e importarlos a la escena de Author y ver cómo quedaría en el nuevo avatar. En la Figura 3.10 vemos un ejemplo del escalado proporcional de la prenda 3D en dos avatares.



Figura 3.10: Escalado proporcional de la prenda del avatar de referencia al avatar *MAN BASE* y *BIG WOMAN*

Una vez tenemos la malla 3D de la prenda escalada, se calcula nuestra métrica de deformación, en este caso la longitud de las aristas. Para ello se calcula la longitud de cada arista de la malla 3D de la prenda original, y también las longitudes de la malla 3D de la prenda escalada, líneas 34 y 35 del código 3.19. Una vez tenemos esos valores se calcula el *target_stain*, línea 37. Este cálculo representa la deformación de la arista. Concretamente, devuelve un valor que indica cuánto ha cambiado la longitud en términos relativos a la longitud original. Este valor objetivo es lo queremos conseguir en las aristas de la malla 2D de la prenda al mover el POM para que se cumpla el ajuste.

```

1 def compute_edge_lengths(points, indices):
2     edge_lengths = []
3     for i in range(0, len(indices), 3):
4         p1_index, p2_index, p3_index = indices[i], indices[i+1], indices[i+2]
5         p1 = points[p1_index]
6         p2 = points[p2_index]
7         p3 = points[p3_index]
8         edge_ab = np.linalg.norm(p2 - p1)
9         edge_bc = np.linalg.norm(p3 - p2)
10        edge_ca = np.linalg.norm(p1 - p3)
11        edge_lengths.append(edge_ab)
12        edge_lengths.append(edge_bc)
13        edge_lengths.append(edge_ca)
14    return edge_lengths
15
16 def compute_edge_difference/avatar1_lengths, avatar2_lengths):
17     if len(avatar1_lengths) != len(avatar2_lengths):
18         raise ValueError('AVATARS NEED SAME NUMBER OF EDGES.')
19     differences = np.abs(np.array(avatar1_lengths) - np.array(avatar2_lengths))
20     return differences

```

```

21 def compute_edge_strain(current_length, original_length):
22     if original_length == 0:
23         return 0
24     return current_length / original_length - 1
25
26 def compute_total_strain(current_lengths, original_lengths):
27     total_strain = []
28     for current_length, original_length in zip(current_lengths, original_lengths):
29         strain = compute_edge_strain(current_length, original_length)
30         total_strain.append(strain)
31     return np.array(total_strain)
32
33 # COMPUTE EDGE LENGTHS Avatar 1 & Avatar 2
34 avatar1_edge_lengths = compute_edge_lengths(avatar1_3dpoints, pieces_indices)
35 avatar2_edge_lengths = compute_edge_lengths(avatar2_3dpoints, pieces_indices)
36
37 target_strain = compute_total_strain(np.array(avatar2_edge_lengths), np.array(
    avatar1_edge_lengths)).flatten()

```

Código 3.19: Métrica de deformación: longitud de las aristas

3.5.3. Relación entre los puntos de medida y los vértices 2D de la prenda

Una vez tenemos el valor de nuestras aristas objetivo, para poder hacer la optimización tenemos que poder mover los vértices de los triángulos de la malla 2D de la prenda respecto al valor del POM. Vamos a ver como funciona el ejemplo del código 3.20.

```

1 diff = 0.7
2 new_pieces_bezierPath = fbezier(diff)
3 new_boundary_points = fboundary(new_pieces_bezierPath)
4 new_vertex_points = fvertex(new_boundary_points)

```

Código 3.20: Ejemplo de agrandar un POM 7cm

Para poder definir los vértices respecto al POM se realizaron una serie de métodos. Se creó una función *fbezier(diff)*, que sirve para mover las curvas de Bézier del patrón en función del POM, *diff* es el valor que queremos agrandar o encoger el POM. Este valor *diff* es el que queremos calcular en la optimización. Se ha utilizado *fbezier(0.7)* en la Figura 3.13 en el paso 2, que indica que hay que agrandar el POM 7 centímetros, 3.5 centímetros a cada lado. Se aplica una transformación equivalente a la herramienta paramétrica, donde *diff*, sería el ajuste mitad, teniendo en cuenta que en un POM horizontal, el primer punto del POM se mueve la mitad del valor de *diff* a la izquierda y el segundo, la mitad a la derecha teniendo en cuenta el signo de *diff* o si es un POM vertical, solamente variará la posición del segundo punto del POM agrandando o encogiendo el largo de la camiseta el valor entero de *diff*.

Para mover los vértices de la malla 2D estos deben depender de las curvas de Bézier. Para ello, se identificaron los puntos del borde de la malla 2D y se obtu-

vieron las coordenadas de cada punto en función del índice del segmento y el valor de t de las curvas de Bézier. Como estos puntos del borde se definen a través de el índice de segmento y el parámetro t , la función `fboundary(new_pieces_bezierPath)`, recalcula los puntos del borde de la malla 2D con los cambios de las curvas de Bézier del patrón realizados en `fbezier`. Paso 3 de la Figura 3.13.

Para definir los vértices interiores de la malla 2D respecto del POM tenemos que hacer que los vértices internos dependan de los vértices borde de la malla. Para ello, por cada vértice se lanza un rayo en la dirección del POM, se busca el punto que intersecta con las curvas de Bézier y el rayo, y luego se buscan los 4 puntos de los vértices del borde de la malla 2D más cercanos a esa intersección. Para que el vértice interno dependa de estos 4 puntos de los vértices del borde se hace una ponderación por distancia inversa que asigna un peso a cada punto en función de su distancia al vértice.

En la Figura 3.11 se muestra cómo se lanza un rayo en la dirección del POM (en este caso, un POM horizontal) que intersecta con los vértices del borde (puntos de la curva de Bézier). En la gráfica de la derecha, se pueden ver los cuatro puntos más cercanos a ese vértice P (puntos azules y magentas). Cada uno de estos vértices tendrá un peso para el punto P , y cuando se modifiquen estos puntos del borde de la prenda, el vértice aplicará los pesos a los puntos del borde modificados para calcular la nueva posición del punto P . En la Figura 3.12 se muestra el rayo y los puntos más cercanos para un vértice P con un POM vertical. Esto crea una configuración de cuadrilátero para cada vértice, definiendo su posición mediante esta especie de combinación lineal entre coordenadas baricéntricas e interpolación bilineal con pesos basados en distancias.

Siguiendo el ejemplo de la Figura 3.13, este proceso corresponde al Paso 4, en la figura podemos ver todos estos pasos y el siguiente código corresponde con esas operaciones. Al realizar estas correspondencias POM-curva de Bézier, curva de Bézier-bordes, bordes-vértices internos malla 2D, los vértices se modifican con el POM y ya podemos calcular las aristas modificadas 2D para obtener el valor que hay que agrandar o encoger el POM en la optimización.

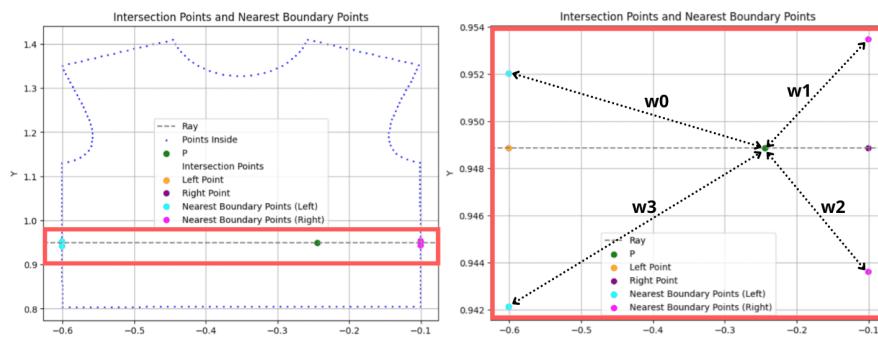


Figura 3.11: Izquierda: intersección horizontal de un vértice P y sus vértices correspondientes del borde más cercanos. Derecha: zoom en la intersección donde se indican los pesos w_0, w_1, w_2, w_3 de cada vecino en el vértice P .

Capítulo 3. Análisis y Desarrollo

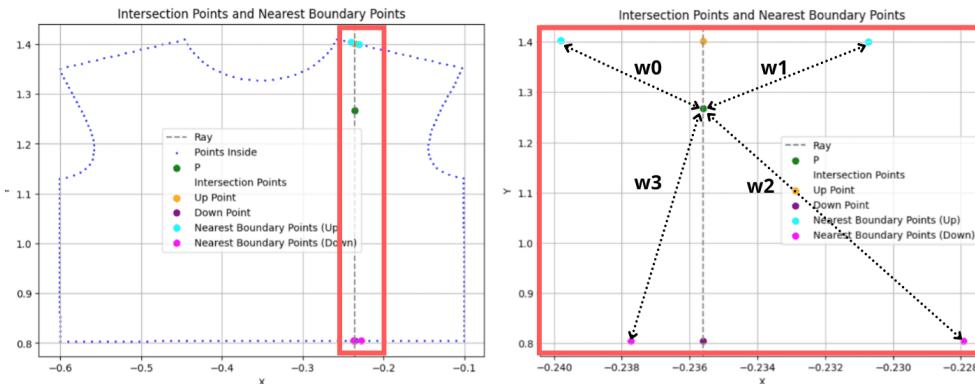


Figura 3.12: Izquierda: intersección vertical de un vértice y sus vértices correspondientes del borde más cercanos. Derecha: zoom en la intersección donde se indican los pesos w_0, w_1, w_2, w_3 de cada vecino en el vértice P .

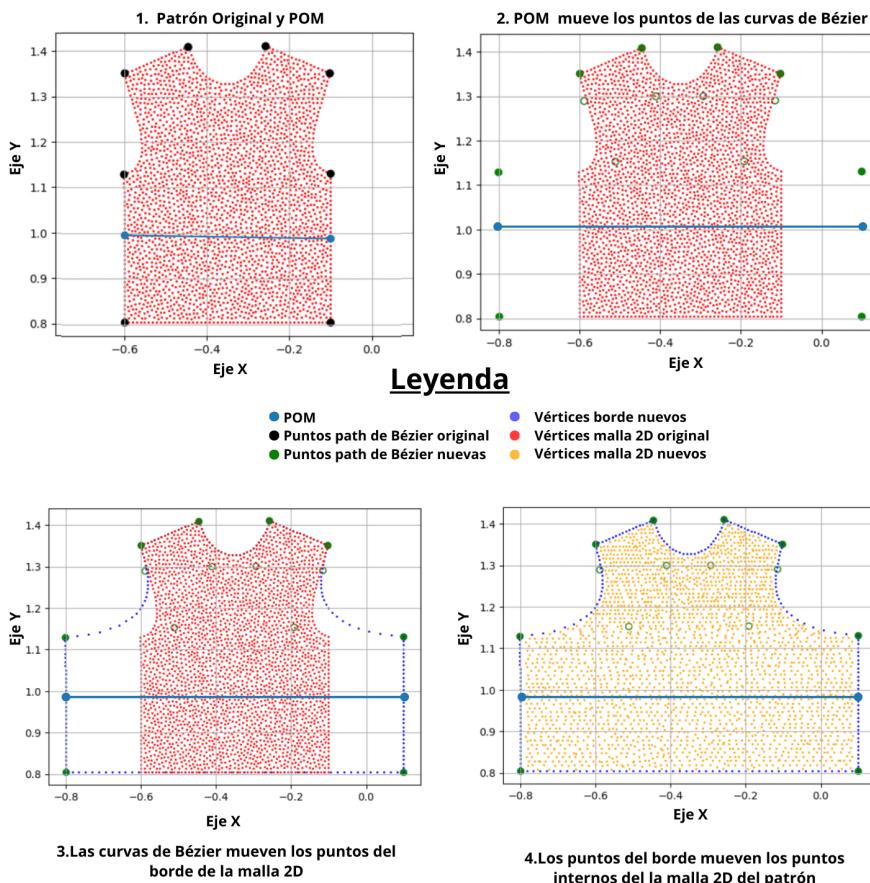


Figura 3.13: Modificación de los vértices 2D de la prenda en función del punto de medida

Esto nos permite modificar el punto de medida siguiendo los mismos criterios que teníamos en la herramienta paramétrica para obtener las nuevas posiciones de las curvas de Bézier y los vértices que componen la malla 2D.

3.5.4. Función de coste y optimización

En este proceso de optimización se ajusta iterativamente el valor del POM para minimizar la función de coste. Al final de este apartado, en el código 3.21, se puede ver la implementación.

Para calcular el nuevo valor del POM y pasárselo a la función de coste se ha utilizando el método de gradiente descendente, para aumentar o disminuir el valor del POM en cada iteración, líneas 22-28. Cuando el proceso iterativo de la optimización llega a 50 iteraciones se alcanza un valor óptimo del POM que minimiza el error entre las deformaciones de la prenda 3D objetivo y la prenda 2D ajustada. La elección de 50 iteraciones en el proceso de optimización se basa en la observación de que, después de este número de iteraciones, los primeros tres dígitos del valor de pérdida convergen, manteniéndose constantes. Nos interesan esos 3 dígitos porque corresponden con los centímetros, por ejemplo un valor de 0.568 corresponde con 5.68 centímetros y 2 decimales son suficientemente precisos para la aplicación de SEDDI.

La función de coste recibe el POM actualizado y calcula la diferencia cuadrática media entre las deformaciones de la prenda 3D y la prenda 2D ajustada por ese POM. Para ello, primero se generan nuevos puntos de control de las curvas de Bézier y bordes de la prenda (mediante las funciones *fbezier* y *fboundary*) y se calculan los nuevos vértices de la prenda (*fvertex*), líneas 4, 5 y 6. Luego, se calculan las longitudes de los segmentos de las aristas de la prenda y se obtiene la deformación total en 2D, línea 8. La diferencia entre esta deformación y la deformación objetivo (*target_strain*, la deformación en 3D) se cuadra, se promedia y se devuelve el coste, líneas 10 y 11. En la Figura 3.14 se puede ver la evolución de la función de coste durante la optimización.

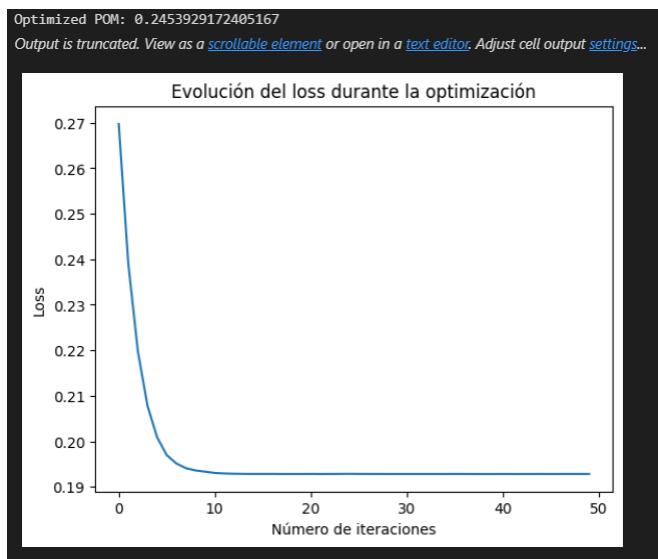


Figura 3.14: Valor del POM optimizado y evolución del error durante la optimización

```

1 original_edge_length = compute_edge_lengths(...)
2
3 def cost_function(optimized_pom, pieces_indices, original_edge_length,
4     target_strain):
5     new_pieces_bezierPath = fbezier(optimized_pom)
6     new_boundary_points = fboundary(new_pieces_bezierPath)
7     new_vertex_points = fvertex(new_boundary_points)
8
9     e_length = compute_edge_lengths(np.concatenate((new_boundary_points,
10        new_vertex_points), axis=0), pieces_indices)
11
12     new_strain = compute_total_strain(e_length, original_edge_length)
13     return np.mean((new_strain - target_strain) ** 2)
14
15 def optimize_pom(...):
16     optimized_pom = initial_pom
17     losses = []
18
19     for _ in range(num_iterations):
20         loss = cost_function(optimized_pom,...)
21         losses.append(loss)
22
23         epsilon = 1e-6
24         perturbation = epsilon
25         loss_positive = cost_function(optimized_pom + perturbation, ...)
26         loss_negative = cost_function(optimized_pom - perturbation, ...)
27         grad = (loss_positive - loss_negative) / (2 * epsilon)
28
29         optimized_pom -= learning_rate * grad
30
31     return optimized_pom, losses
32
33 initial_pom = 0
34 learning_rate = 0.1
35 num_iterations =50
36
37 optimized_pom, losses = optimize_pom(initial_pom, pieces_indices,
38     original_edge_length, target_strain, learning_rate, num_iterations)

```

Código 3.21: Función de coste

Cabe destacar que inicialmente se consideró la opción de emplear tensores con PyTorch para el cálculo del gradiente en el proceso de optimización con el optimizador Adam, pero se presentaron algunas dificultades al intentar implementar esta técnica. Las operaciones de transformación de los vértices al ajustar el punto de medida no están diseñadas específicamente para ser diferenciables, sobre todo la parte de identificar la orientación del POM para aplicar una transformación u otra, lo que generaba problemas al calcular el gradiente. Aunque prácticamente todas las operaciones se consiguieron ejecutar correctamente con tensores en un primer prototipo, por la complejidad de una posible solución completa y el margen temporal de este proyecto, se tomó la decisión de optar por una técnica de diferencias finitas. Aunque este método puede ser más costoso computacionalmente, fue adecuado para abordar las particularidades de nuestro problema y los resultados obtenidos fueron satisfactorios.

4

Resultados

En este apartado se muestran y discuten los resultados obtenidos gracias a las herramientas desarrolladas. Se presenta la creación de los diferentes tipos de puntos de medida, se analizan las posibles modificaciones paramétricas, tanto con conectividad entre diferentes POMs como sin ella y además, se muestran las soluciones obtenidas del problema de optimización para preservar el ajuste de la prenda en diferentes avatares.

4.1. Tipos de puntos de medida (POM)

Los puntos de medida van desde un punto o segmento a otro punto o segmento de la prenda y se representan con líneas que unen esos dos puntos. Los puntos de medida se dividen en tres tipos según su orientación: horizontales, verticales y diagonales. Estos puntos se definen y dibujan sobre el patrón en el *canvas* 2D.

- Horizontales: los puntos de medida horizontales se definen de izquierda a derecha. Se utilizan para medir el ancho de la prenda, como el busto o el contorno de la cintura como se muestra en la Figura 4.1.
- Verticales: los puntos de medida verticales se definen de arriba hacia abajo, ver Figura 4.2. Son empleados para medir el largo de diferentes secciones de la prenda, como la altura de la prenda desde el hombro hasta el dobladillo o la longitud de la pierna en unos pantalones.

Capítulo 4. Resultados

- Diagonales: los puntos de medida diagonales se utilizan para medir partes de la prenda que no son perfectamente horizontales ni verticales, como la abertura de la manga (Figura 4.3) y otras áreas inclinadas. Además, permiten un estiramiento de la prenda de manera desigual en diferentes direcciones que puede ser útil para jugar con el diseño de la prenda.

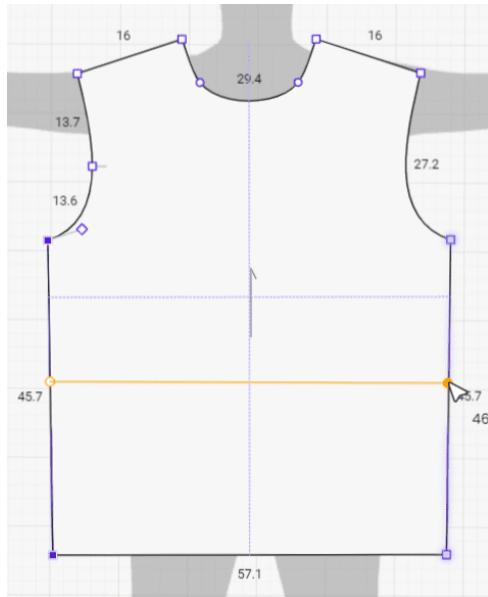


Figura 4.1: Punto de medida horizontal

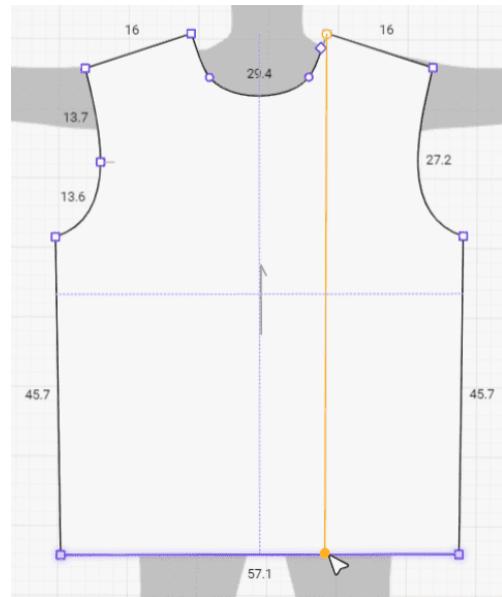


Figura 4.2: Punto de medida vertical

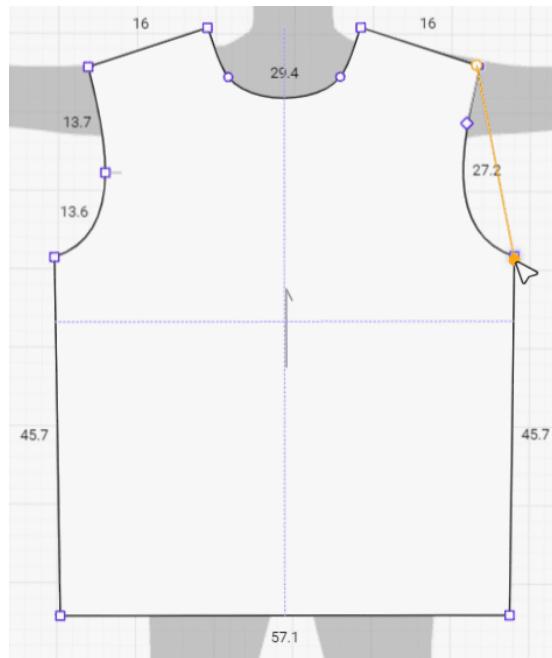


Figura 4.3: Punto de medida diagonal

4.2. Posibles ediciones paramétricas

Las ediciones paramétricas disponibles dependen de los tipos de puntos de medida. Para una mejor visualización de los POMs, se muestran con líneas amarillas los puntos de medida horizontales, rojas los verticales y verdes los diagonales. Según definas la orientación de los POMs podrás editar de diferente manera la prenda:

- Horizontales: los puntos de medida horizontales editan ambos puntos o segmentos que definen el POM. Mueven los puntos en direcciones opuestas para ensanchar o estrechar la prenda. En la Figura 4.4 podemos ver todas las posibles modificaciones en el eje horizontal.

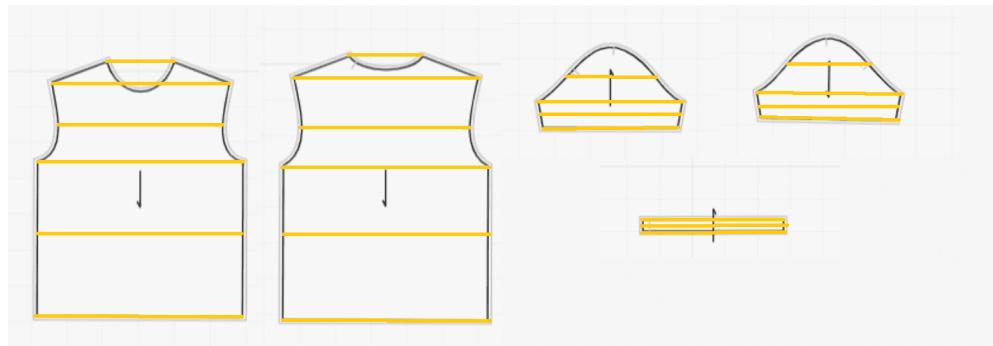


Figura 4.4: Posibles puntos de medida horizontales

- Verticales: los puntos de medida verticales mueven solamente el último punto o segmento definido en el POM, para alargar o encoger la prenda. En la imagen 4.5 podemos ver las diferentes formas de definir estos puntos de medida en el eje vertical. En cualquiera de los casos que vemos en la pieza frontal y trasera de la camiseta, solo se modifica el segmento del dobladillo que hace alargar la prenda.

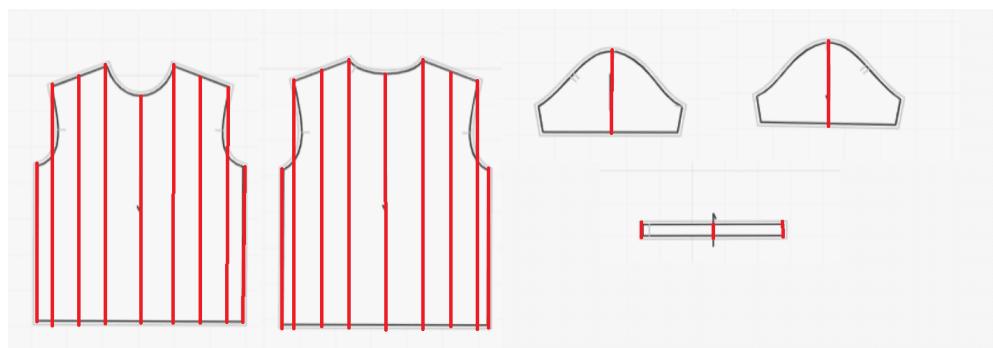


Figura 4.5: Posibles puntos de medida verticales

- Diagonales: los puntos de medida diagonales editan los dos puntos o segmentos del POM y se mueven en dirección opuesta siguiendo el ángulo original. En la Figura 4.6 se muestran los puntos de medida diagonales que nos interesan pero se podrían definir de múltiples maneras como en la Figura 4.7 para hacer deformaciones fuera de lo común pero que podrían ser interesantes para estudiar algún diseño.

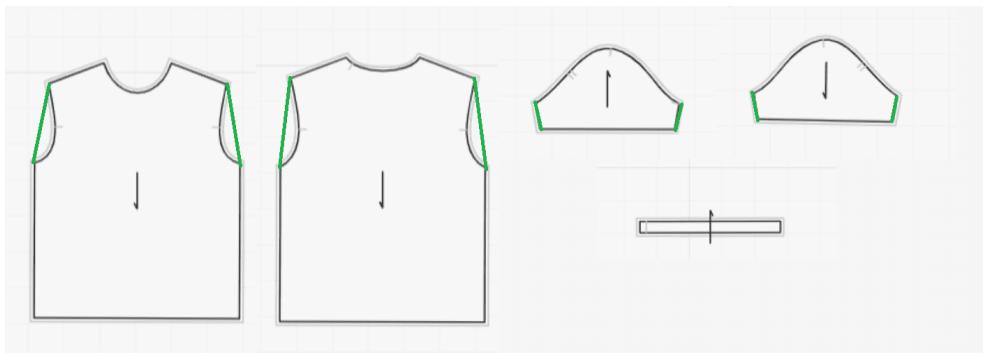


Figura 4.6: Posibles puntos de medida diagonales más comunes

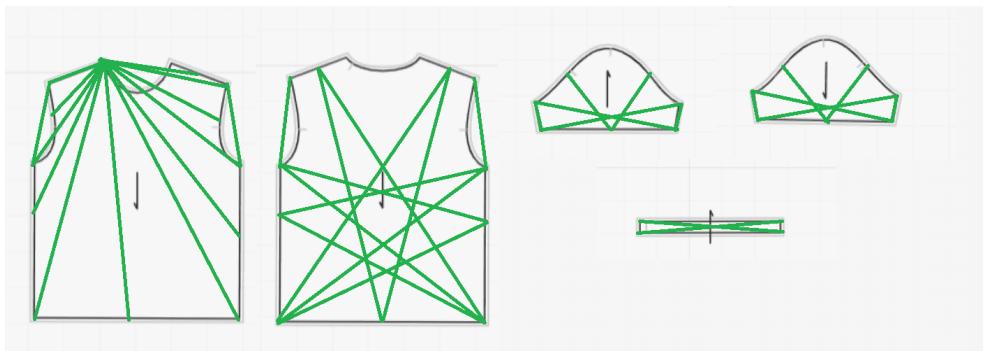


Figura 4.7: Ejemplos de otros posibles puntos de medida diagonales

A continuación se muestran los resultados del patrón 2D al realizar estas modificaciones paramétricas en la prenda base con la herramienta desarrollada. Cabe destacar que al modificar los patrones 2D en SEDDI Author automáticamente se actualiza la pieza 3D pero para simplificar estas visualizaciones solo se van a mostrar las modificaciones en el patrón 2D.

Para nuestra prenda base, los puntos de medida que se han tenido en cuenta son los que se muestran en la Figura 4.8, que corresponden con los puntos de medida que suelen usar los patronistas para definir los puntos de medida en un patrón de una camiseta estándar.

4.2. Posibles ediciones paramétricas

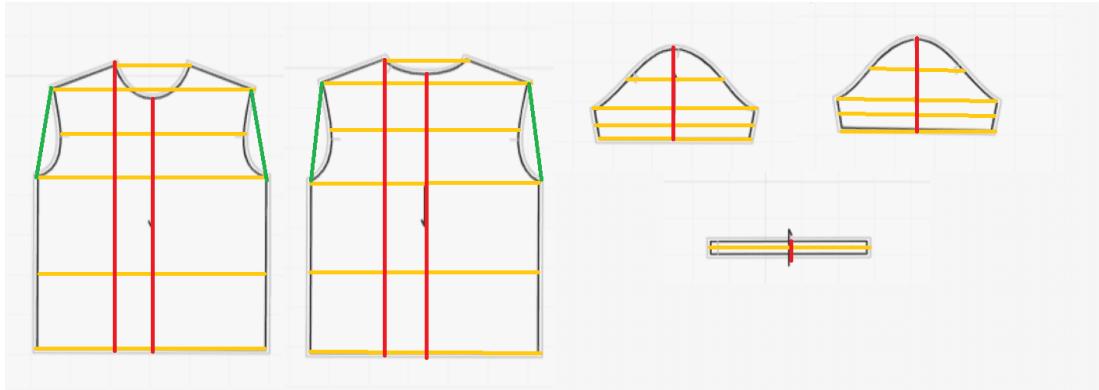


Figura 4.8: Puntos de medida que suelen usarse para este tipo de camiseta

Empezamos mostrando las modificaciones paramétricas horizontales de la pieza frontal del patrón (la pieza trasera se comporta de la misma manera). En las Figuras 4.9 y 4.10 primero podemos observar la pieza original, seguida del POM modificado con una distancia más pequeña y finalmente con un valor más grande.

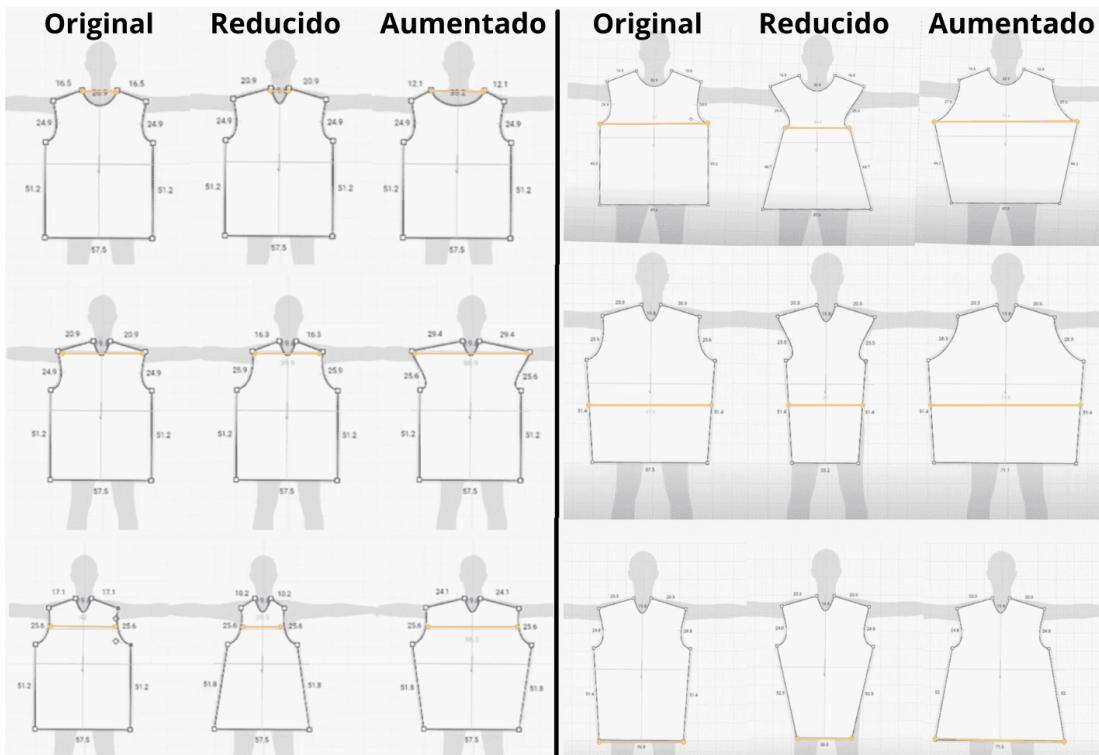


Figura 4.9: Puntos de medida horizontales modificados de forma paramétrica

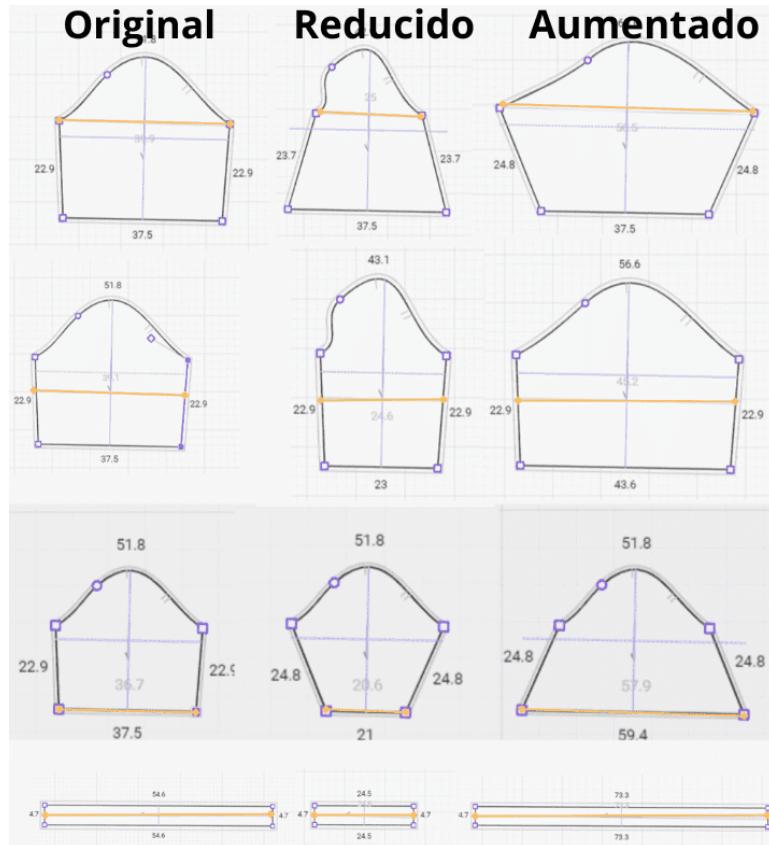


Figura 4.10: Puntos de medida horizontales en la manga y cuello modificados de forma paramétrica

En la Figura 4.10 se muestran las modificaciones horizontales de las mangas. Observamos que, al cambiar la anchura, todos los casos funcionan correctamente, excepto en el primero, al modificar el final o anchura de tapa de la manga, la curva produce algunos efectos no deseados.

En la Figura 4.11 se analizan mangas definidas de diferentes formas. A la izquierda, se presentan dos mangas originales en las que podemos observar que, al agrandar o encoger el POM, la anchura de la tapa de la manga se comporta de manera extraña. Esto ocurre porque la curva de la manga está definida en más de dos segmentos, o si son dos segmentos, no están divididos por la mitad. Esta es una limitación de la herramienta desarrollada, ya que tenemos la restricción de que el POM debe estar definido por dos puntos o dos segmentos. La modificación solo afecta a esos segmentos y no puede afectar a los demás, por lo que no podemos mover todos los segmentos que componen la curva.

Una posible solución para que la manga se comporte de forma correcta, sería redefinir o crear una nueva manga donde la curva esté definida en dos segmentos divididos por la mitad, punto verde de la Figura 4.12. De esta forma podemos mover los dos segmentos correctamente.

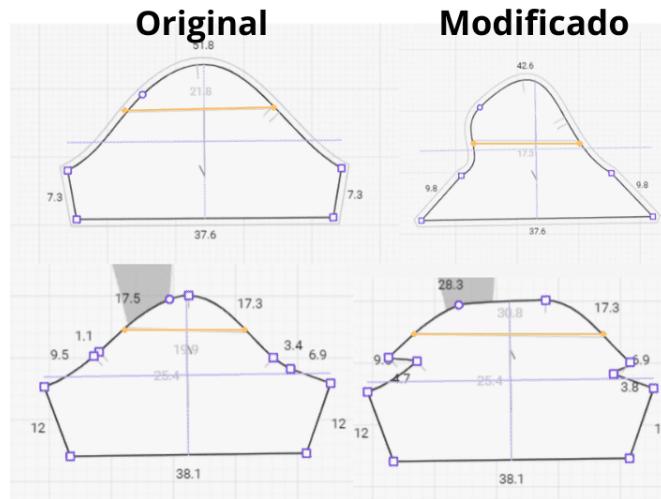


Figura 4.11: Problemas de edición paramétrica con mangas definidas de diferentes formas

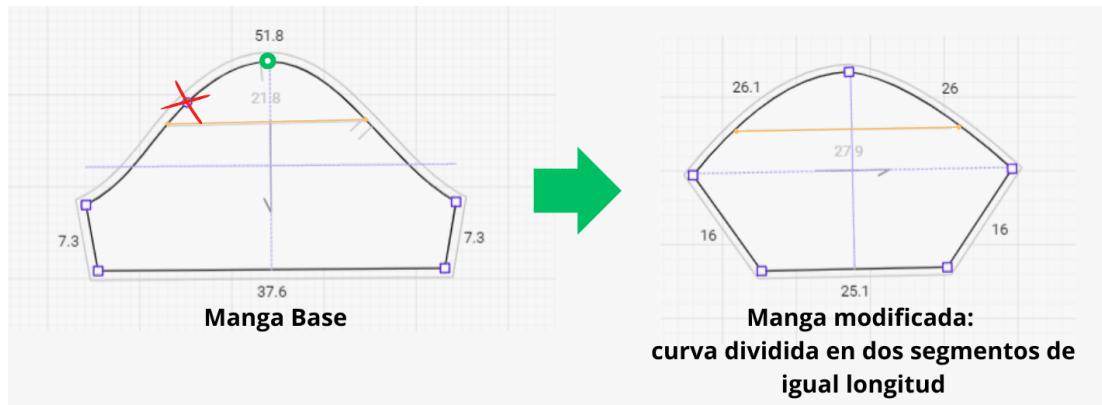


Figura 4.12: Solución: redefinir tapa de la manga en solo 2 segmentos con la misma longitud

Como podemos observar, las curvas y las mangas requieren especial cuidado al definirlas. Es posible que, al hacer modificaciones paramétricas, posteriormente se necesiten ajustes manuales para que, al coser, el ajuste con las demás piezas sea correcto.

A continuación se muestran las modificaciones paramétricas verticales. En la Figura 4.13, el POM se define desde el punto alto del hombro y desde el centro de la camiseta hasta el dobladillo. Se muestra la prenda original, acortada y alargada, y observamos cómo se modifica solo el largo de la camiseta. También podemos ver cómo se realizan las modificaciones verticales de las mangas y el cuello en la Figura 4.14.

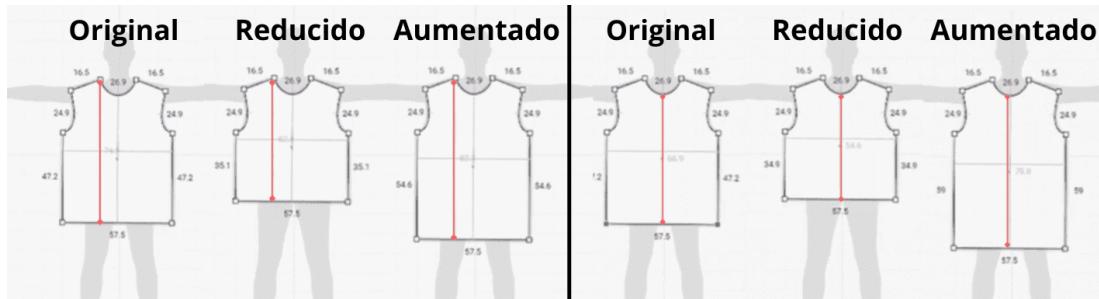


Figura 4.13: Puntos de medida verticales modificados paramétricamente

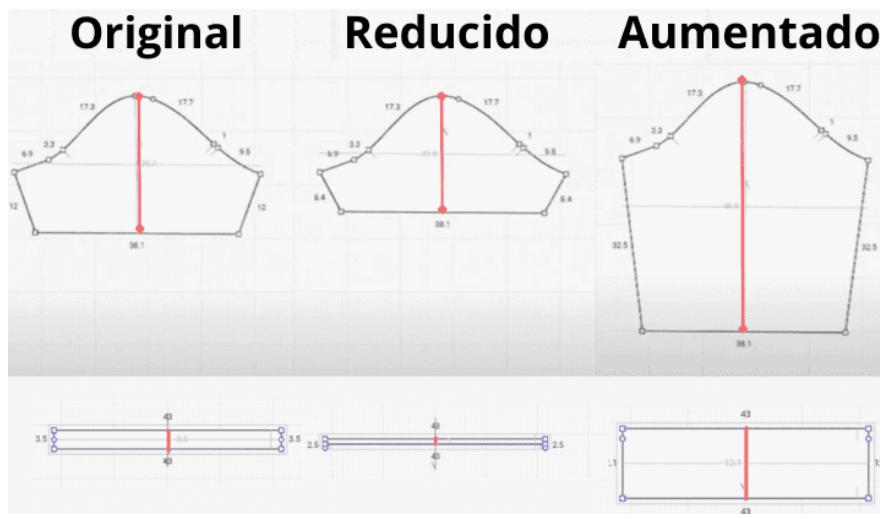


Figura 4.14: Puntos de medida verticales en la manga y cuello modificados paramétricamente

Finalmente tenemos los puntos de medida diagonales, en la Figura 4.15 se muestra un ejemplo de modificación paramétrica diagonal de la apertura de la manga en la pieza frontal del patrón.

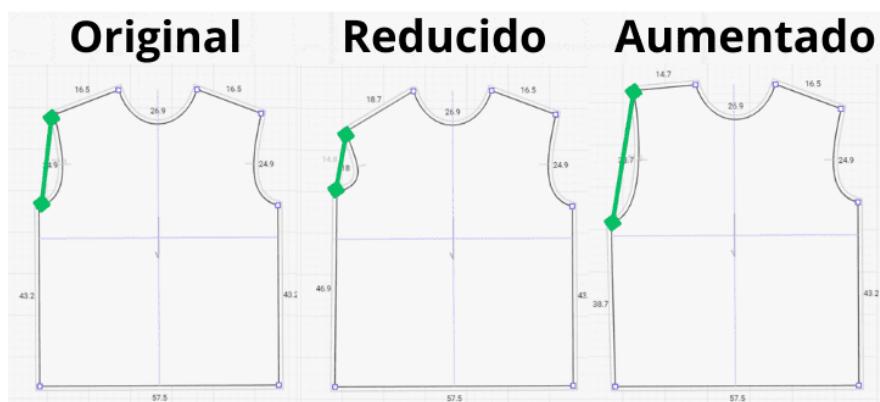


Figura 4.15: Puntos de medida diagonales en la apertura manga modificados de forma paramétrica

4.2. Posibles ediciones paramétricas

Gracias a la herramienta de edición paramétrica, podemos definir conexiones entre todos los casos de puntos de medida mencionados anteriormente. En las siguientes imágenes, se muestran las piezas frontales (*FRONT*) y traseras (*BACK*) del patrón de la camiseta. A la izquierda, se encuentran las posiciones originales, y a la derecha, las posiciones modificadas. Al definir los POMs en las propiedades paramétricas, es importante asegurarse de que tengan el mismo tipo. Con solo modificar una de las medidas, es decir, cambiar el valor de un POM, si tenemos activado el botón *Connections*, todos los POMs con el mismo tipo se moverán proporcionalmente.

En la Figura 4.16 se muestra una conexión horizontal (ancho de la camiseta) entre el panel frontal y el trasero , en la 4.17 se puede ver un ejemplo de conexión vertical (largo de la camiseta) y en la 4.18, una conexión diagonal que afecta a la apertura de todas las mangas solamente modificando un POM.

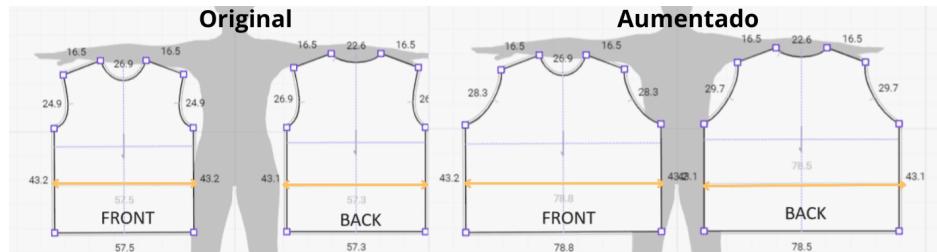


Figura 4.16: Ejemplo conexiones horizontales

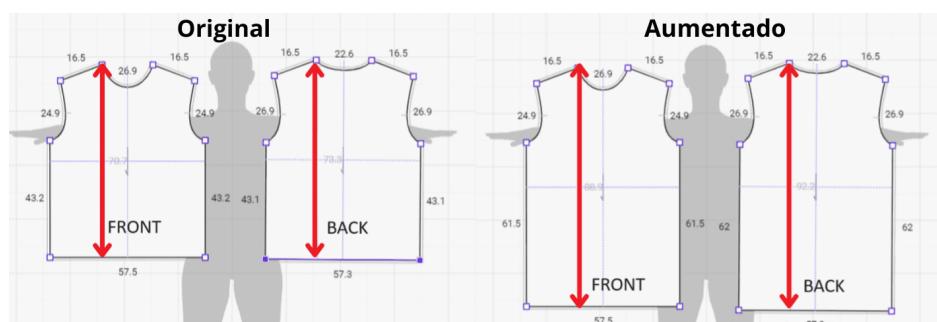


Figura 4.17: Ejemplo conexiones verticales

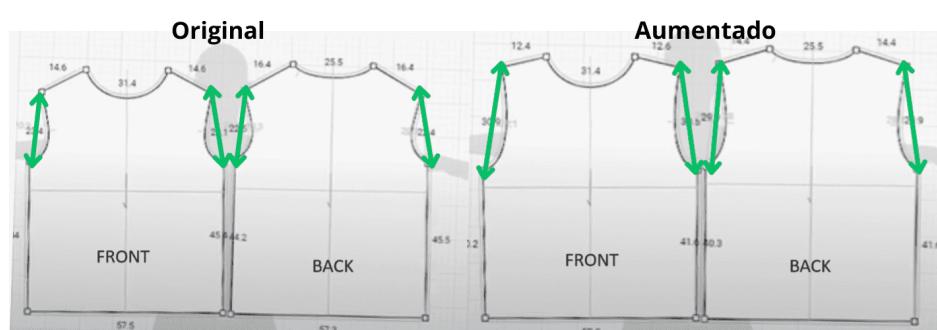


Figura 4.18: Ejemplo conexiones diagonales

Capítulo 4. Resultados

Para completar esta sección de resultados de modificaciones paramétricas y validar la herramienta con otros patrones, también se llevaron a cabo pruebas con patrones de otro tipo de camisetas, vestidos y pantalones.

Una de las limitaciones de la herramienta es que las operaciones se realizan según la orientación del patrón de la camiseta base de Author (*Mens SS T Seddi*). Para definir los puntos de medida y modificar su longitud con la herramienta paramétrica, es necesario importar los patrones en la misma rotación que nuestra camiseta base. Como podemos ver en la Figura 4.19, se ha importado el vestido *PTP1-Dress V2* rotado de forma diferente a la camiseta base, lo que provoca que las operaciones para mover los segmentos del patrón funcionen de manera incorrecta.

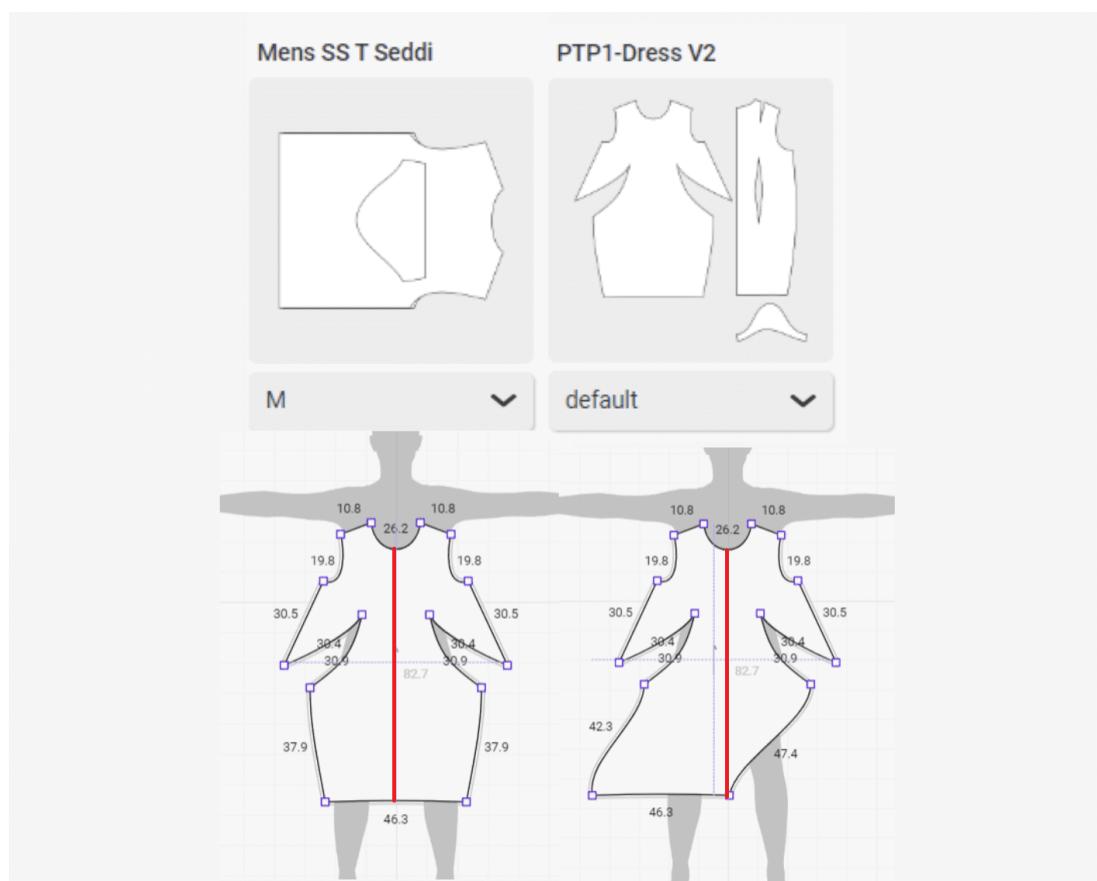


Figura 4.19: Edición paramétrica en un vestido

En cambio, por ejemplo si importamos este patrón de un pantalón con la misma orientación que la camiseta base, ver Figura 4.20, las ediciones paramétricas funcionan correctamente y podemos realizar las ediciones paramétricas que necesitemos como se puede observar en la Figura 4.21.

4.2. Posibles ediciones paramétricas

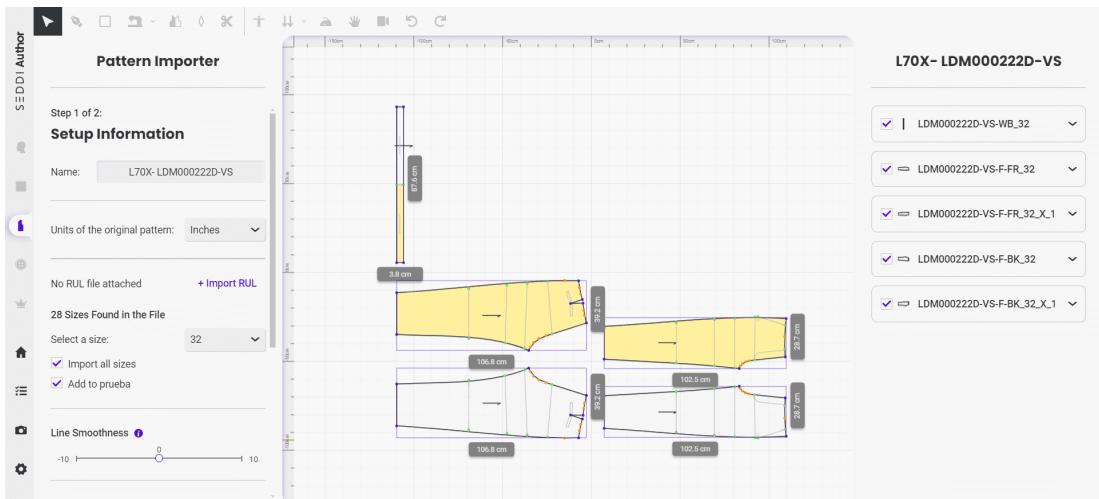


Figura 4.20: Importación patrón de un pantalón

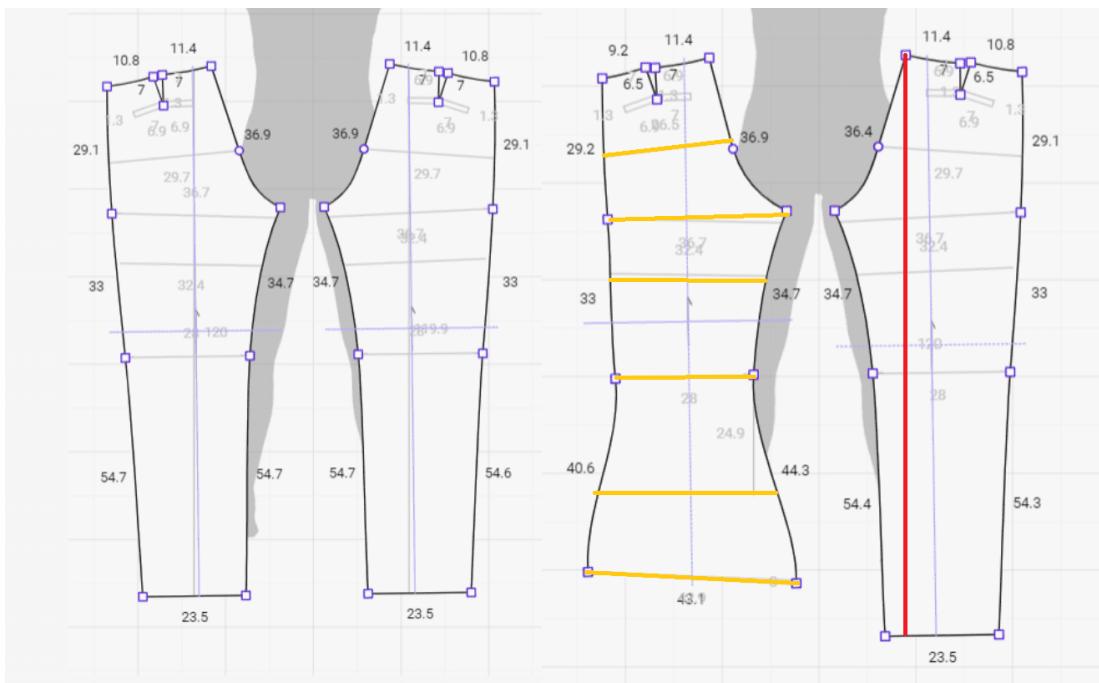


Figura 4.21: Edición paramétrica de un pantalón

Todos estos resultados se pueden visualizar en este video¹, donde se muestra cómo se crean los puntos de medida y cómo se modifican las prendas con la herramienta de edición paramétrica. En este otro video², se puede ver las ediciones paramétricas con las conexiones entre puntos de medida.

¹<https://www.youtube.com/watch?v=o0WfbeXqinA>

²<https://www.youtube.com/watch?v=YYMzGYHQdnE>

Los resultados demuestran que podemos definir los puntos de medida en cualquier parte de la prenda y modificarla de manera paramétrica. No obstante, es importante tener cuidado a la hora de definir las mangas y asegurarnos de que la orientación inicial del patrón sea correcta.

4.3. Preservación del estilo con puntos de medida entre diferentes avatares

Para obtener los resultados de la optimización para preservar el estilo entre avatares con diferentes formas corporales, se ha utilizado el Avatar *Womens US ASTM Sz M/6* como referencia, que llamaremos “*Woman Base*”, como se muestra en la Figura 4.22. Este avatar se encuentra en posición T y la camiseta utilizada es la que ofrece *Author Public* como base en sus patrones. Suponemos que esta camiseta le queda perfecta y es el diseño que queremos preservar. Las medidas que queremos calcular son el ancho a través del cuerpo y el largo desde el hombro hasta el dobladillo, Figura 4.23. Para *Woman Base*, las medidas son 50 cm de ancho y 60 cm de largo.



Figura 4.22: *Womens US ASTM Sz M/6 - WOMAN BASE* Medidas: 50x60cm

4.3. Preservación del estilo con puntos de medida entre diferentes avatares

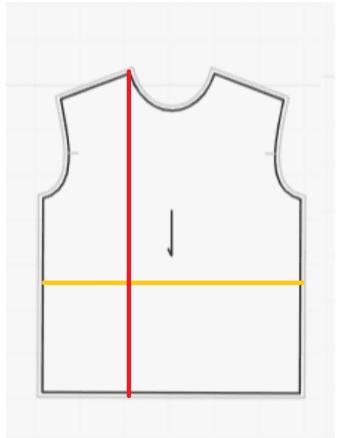


Figura 4.23: Puntos de medida que queremos calcular para los patrones de los nuevos avatares, POM vertical en rojo y POM horizontal en amarillo

Para analizar los resultados, se han utilizado 5 avatares. De esta forma tenemos una perspectiva completa al representar la prenda en avatares con medidas antropomórficas extremas. Para hacerlo más fácil de leer, hemos establecido una nueva nomenclatura a los avatares utilizados, *BASE* para los avatares con medidas estándar, *THIN* para los más delgados y *BIG* para los más robustos como se puede ver en la Tabla 4.1.

Nombre	Avatar de Author
WOMAN BASE	Womens US ASTM Sz M/6
MAN BASE	Mens US ASTM Sz M/40
THIN WOMAN	WOMENS EU Sz 36
THIN MAN	MENS EU Sz 48
BIG WOMAN	WOMENS US ASTM Plus 24 Straight
BIG MAN	MENS US ASTM Big 54

Tabla 4.1: Nombre avatares

Los resultados que esperamos son los siguientes:

AVATAR	POM HORIZONTAL	POM VERTICAL
WOMAN BASE	50cm (que se quede igual)	60cm (que se quede igual)
MAN BASE	Aumente	Aumente
THIN WOMAN	Disminuya o igual	Disminuya o igual
THIN MAN	Aumente (menos que MAN BASE)	Aumente (más que MAN BASE)
BIG WOMAN	Aumente	Aumente
BIG MAN	Aumente	Aumente

Tabla 4.2: Resultados esperados en relación a la referencia *WOMAN BASE*

Capítulo 4. Resultados

En la Figura 4.24 observamos cómo queda la camiseta con los POMs sin modificar, de 50x60 centímetros. Como se aprecia, la prenda queda corta o estrecha en todos los avatares. Por tanto, en la optimización, el objetivo es encontrar los valores óptimos de ancho y largo para que esta camiseta se adapte de manera adecuada a las diferentes formas corporales de los avatares.

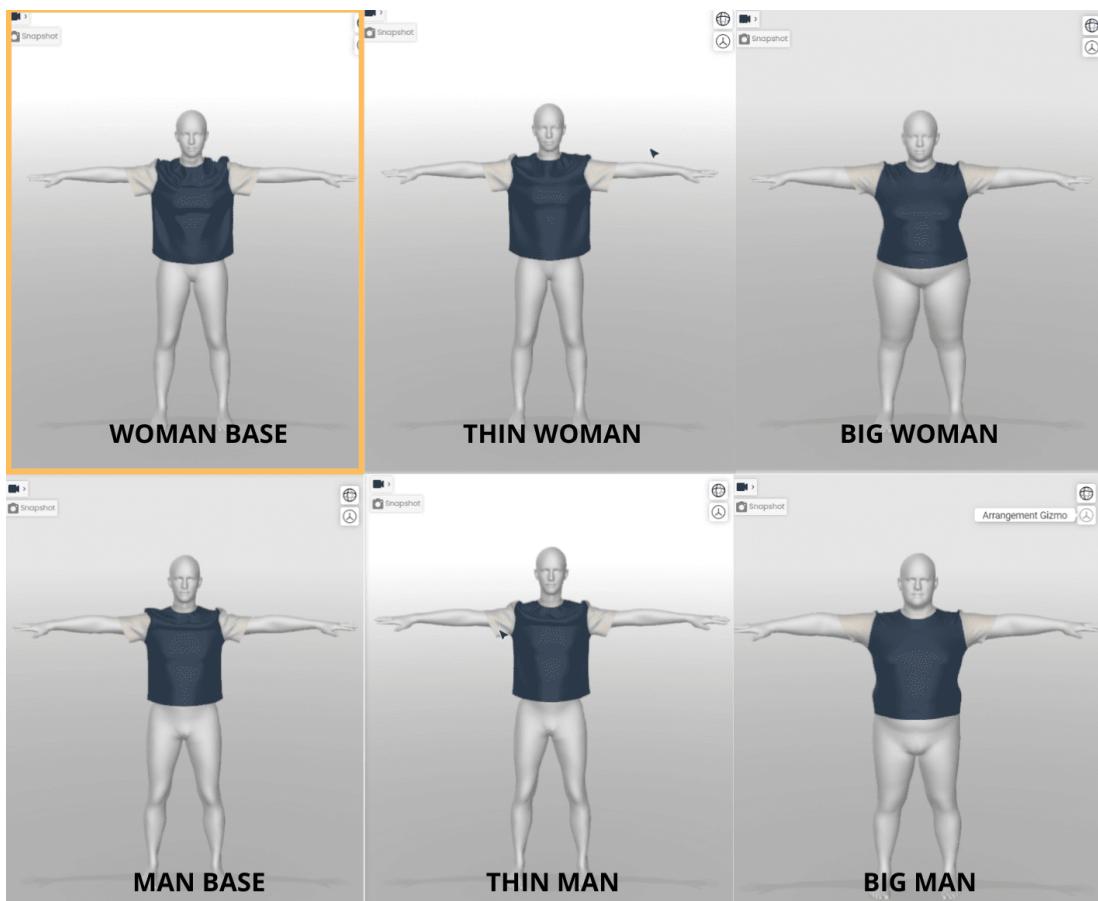


Figura 4.24: Camiseta 50x60cm para todos los avatares

Para obtener los siguientes resultados visuales, una vez se consiguen los valores óptimos de los puntos de medida en Python, con la herramienta de edición paramétrica se modifican los puntos de medida ajustando los valores en SEDDI Author en el patrón 2D tanto para la parte frontal como la parte trasera de la camiseta y automáticamente se actualiza el 3D. Finalmente se cose la prenda para ver el ajuste. En este video³ se muestra el proceso de modificación de puntos de medida y cosido de la prenda con los valores óptimos.

³<https://www.youtube.com/watch?v=KXJe6TCvCG0>

4.3. Preservación del estilo con puntos de medida entre diferentes avatares

En la Tabla 4.3 se muestran los valores obtenidos en la optimización, que son coherentes con los resultados esperados.

AVATAR	POM HORIZONTAL	POM VERTICAL
WOMAN BASE	50 cm	60 cm
MEN BASE	+8.27	+2.13
THIN WOMAN	-1.11	+0.13
THIN MAN	+6.82	+2.2
BIG WOMAN	+24.53	+4.65
BIG MAN	+30.55	+6.38

Tabla 4.3: Resultados obtenidos en cm en relación a la referencia *WOMAN BASE*

En la Figura 4.25 se puede observar los patrones 2D modificados, donde se ha ajustado la anchura y altura de forma paramétrica con los valores de la tabla y en la Figura 4.26 visualiza las piezas 3D ya cosidas con estas medidas óptimas.

En el caso de *MAN BASE*, hay un aumento de 8.27 cm en la medida horizontal y de 2.13 cm en la vertical, lo cual concuerda con lo esperado. Por otro lado, para *THIN WOMAN*, se observan disminuciones en las medidas horizontales, mientras que las medidas verticales se mantienen relativamente estables. En *THIN MAN*, la medida horizontal aumenta, pero no tanto como en *MAN BASE*, debido a que es más delgado. En cambio, para *BIG WOMAN* y *BIG MAN*, se observan aumentos significativos en ambas medidas.

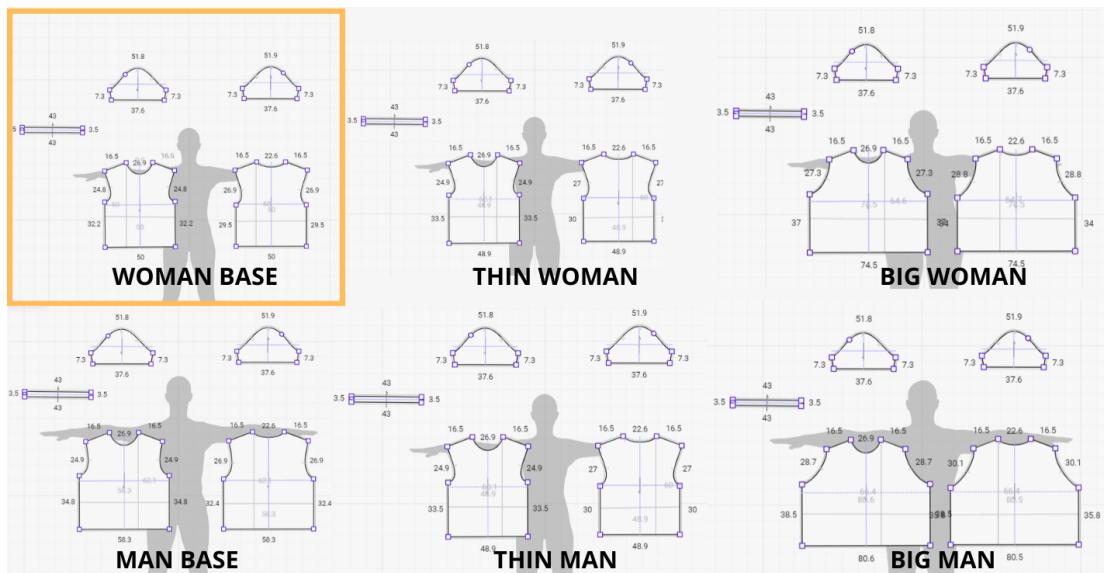


Figura 4.25: Resultados patrones ajustados con las medidas correspondientes

Capítulo 4. Resultados

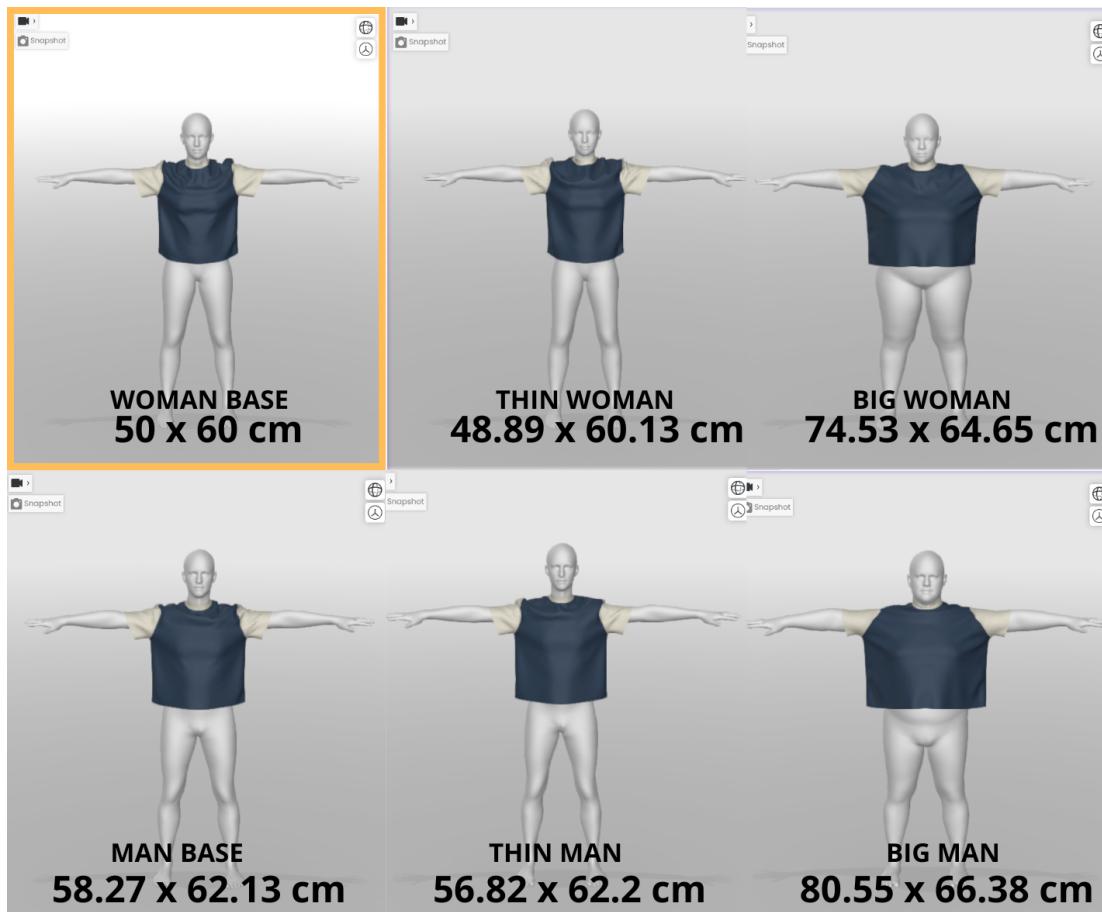


Figura 4.26: Camisetas con las medidas ajustadas a cada avatar para preservar el ajuste

Analizando los resultados de la Figura 4.26, se observa que los cambios son los esperados en relación con la referencia *WOMAN BASE*. El largo de la camiseta queda exactamente en la curva de la cintura de todos los avatares, similar a la referencia, y la anchura también muestra un ajuste correcto, ya que visualmente se respeta la distancia del cuerpo a la prenda de la misma manera que ocurre en el avatar de referencia.

4.3. Preservación del estilo con puntos de medida entre diferentes avatares

Todos estos resultados se compartieron con los diseñadores para validar la herramienta. Sus comentarios fueron positivos sobre la capacidad de crear y editar los puntos de medida, destacando que las ediciones paramétricas propuestas tenían sentido. También consideraron que la herramienta es útil, ya que evita la necesidad de editar manualmente punto por punto el patrón, permitiendo en su lugar realizar ediciones basadas en distancias, de manera similar a como lo harían manualmente los patronistas.

Sin embargo, también identificaron áreas de mejora significativas, sobre todo en la parte de preservación del estilo de la prenda. Aunque el ajuste del patrón hemos visto que se adapta a las nuevas formas corporales, solo tenemos en cuenta el largo y el ancho del cuerpo. Esto provoca que las sisas se agranden, pero las mangas, el cuello y los hombros no se ajustan adecuadamente a estas nuevas proporciones. Estos dos parámetros no son suficientes para ajustar la prenda correctamente. El siguiente desafío es ajustar el cuello, los hombros, la sisa y las mangas para que mantengan las mismas proporciones en el nuevo cuerpo. Para mejorar esto, nos propusieron alternativas como utilizar *grading rules* (reglas de escalado) para que las medidas de las mangas, hombros, cuello y sisa se escalen de manera proporcional a lo establecido por estas reglas, respetando el estilo del diseño, según el ancho y el largo obtenidos en el proceso de optimización. Por limitaciones del alcance de este proyecto, estas mejoras serán abordadas en un próximo artículo académico.

Finalmente, en este vídeo resumen⁴ se muestra cómo funcionan todas las herramientas desarrolladas y los resultados que se presentan en este trabajo.

⁴<https://www.youtube.com/watch?v=N1ugn6eypq8>

5

Conclusiones y trabajos futuros

Para finalizar, en esta sección se comentan las conclusiones a las que se ha llegado en este proyecto, si se han cumplido o no los objetivos, así como los conocimientos adquiridos, posibles mejoras y futuros proyectos a partir de este.

5.1. Conclusiones

El objetivo principal de este trabajo fue diseñar e implementar una herramienta que permitiera editar un patrón base de forma paramétrica en la plataforma SEDDI Author, e investigar el escalado de patrones desde un punto de vista paramétrico. Para comprobar que se han cumplido, procedemos a analizar si se han conseguido o no, los subobjetivos propuestos al inicio del proyecto.

- **Crear una herramienta en SEDDI Author que permita al usuario crear dinámicamente puntos de medida:**

Este subobjetivo se ha cumplido con éxito. La herramienta permite la creación dinámica de puntos de medida, facilitando a los usuarios la creación de puntos de medida horizontales, verticales y diagonales en cualquier parte del patrón y visualizar las medidas de los puntos de medida en centímetros para realizar verificaciones de ajuste.

- **Crear una herramienta de edición paramétrica en SEDDI Author, para que los puntos de medida puedan modificar la prenda en 2D y en 3D. También se deben tener en cuenta las conexiones entre puntos de medida:**

Se diseñó e implementó una herramienta de edición paramétrica en SEDDI Author que permite modificar la prenda tanto en 2D como en 3D, teniendo en cuenta las conexiones entre los puntos de medida. Esto cumple el segundo subobjetivo.

- **Diseñar y resolver un problema de optimización para encontrar los valores óptimos de los puntos de medida que preserven el ajuste de la prenda entre avatares de distintas medidas antropomórficas:**

Se diseñó y resolvió el problema de optimización, cumpliendo con el subobjetivo. Sin embargo, los resultados mostraron que la optimización basada en escalado afín, no es lo suficientemente precisa para preservar el estilo de la prenda en su totalidad. Los ajustes en el largo y ancho no son parámetros suficientes para mantener el ajuste correcto de toda la prenda. Esto indica que, aunque el subobjetivo se ha cumplido técnicamente, la solución necesita una futura mejora para que las prendas se ajusten completamente, teniendo en cuenta otros parámetros y/o basándose en la interpolación de las reglas de escalado (*grading rules*).

Podemos concluir que los requisitos establecidos se han cumplido, dado que la herramienta de edición paramétrica desarrollada es compatible con el software actual de SEDDI Author. Se utilizó el patrón de una camiseta recta como base, y adicionalmente se comprobó que funciona con otros tipos de prenda con la misma orientación que la prenda base. Además, la herramienta permite modificar la longitud de los puntos de medida (POMs) para ajustar el patrón 2D y que se refleje en la prenda 3D. También, cada POM cuenta con un control deslizante y un valor de entrada para introducir la longitud exacta en centímetros y posibilitando la activación o desactivación de la conectividad.

Los dos diseñadores involucrados en la validación de la herramienta, valoraron su potencial y su capacidad para facilitar el proceso de creación y ajuste de patrones. La herramienta es útil porque permite ediciones automáticas basadas en distancias 3D, lo que simplifica la modificación de los patrones en función del cuerpo objetivo. Sin embargo, existen algunas limitaciones como la orientación de las prendas, que debe ser la misma que la del patrón base, y tener cuidado con la definición de los patrones, especialmente al modificar segmentos como las mangas. Aunque la optimización logra una buena preservación del ajuste, no es suficiente para mantener la proporción en todos los aspectos de la prenda.

En conclusión, en general se han cumplido con todos los objetivos propuestos, la herramienta desarrollada permite la creación dinámica de puntos de medida,

la edición paramétrica de los patrones en 2D y 3D, y se ha proporcionado una solución a un problema de optimización para encontrar los valores óptimos de los puntos de medida. Por otra parte y como ya hemos comentado, se ha verificado que el escalado afín de un POM, no es suficiente para preservar el estilo de la prenda a escala global, por lo que se requiere un estudio más detallado de la interpolación de *grading rules* en partes críticas como el cuello, los hombros, las sisas y las mangas.

Para finalizar, este trabajo fin de máster me ha aportado mucho a nivel personal y profesional. Me ha permitido mejorar mi nivel de inglés, participar en un entorno de trabajo profesional y aprender nuevas tecnologías como React, TypeScript, los estados de Redux y PyTorch. Además, ha sido una oportunidad para afianzar mis conocimientos en áreas como las mallas 3D de la asignatura de Fundamentos Matemáticos y Físicos para la Informática Gráfica, las curvas de Bézier que aprendí en Animación por Computador, así como en Visualización y Metodología Aplicada para la redacción de esta memoria.

5.2. Trabajos futuros

Para mejorar la herramienta y su capacidad de preservación del ajuste de la prenda, se propone abordar las siguientes tareas en futuros desarrollos:

- **Extensión de la edición paramétrica a diferentes tipos de prendas y orientaciones:** Desarrollar funcionalidades que permitan editar patrones de cualquier tipo de prenda, independientemente de su orientación inicial.
- **Mejora en la definición de los puntos de medida y almacenamiento permanente en la base de datos de SEDDI:** Implementar la capacidad de definir puntos de medida que consideren varios segmentos de la prenda simultáneamente, para solventar la limitación actual en la edición de mangas. Y mejorar la herramienta para que los puntos de medida creados dinámicamente se guarden automáticamente en la base de datos de SEDDI, garantizando la persistencia de los datos y facilitando su acceso en futuras sesiones de trabajo.
- **Integración de *grading rules* para preservar el estilo de la prenda:** Incorporar funcionalidades que permitan interpolar de forma no lineal las reglas de escalado *grading rules* al ajustar la prenda, asegurando una preservación más precisa del estilo y las proporciones de la prenda entre diferentes tallas.
- **Optimización con tensores para mejorar la velocidad de cálculo y estudiar la dependencia entre puntos de medida:** Vectorización del

algoritmo de optimización con tensores para acelerar el cálculo y mejorar la eficiencia del proceso de ajuste de patrones. Además realizar la optimización teniendo en cuenta más de un punto de medida a la vez.

- **Incorporación de la preservación de ajuste en SEDDI Author:** Integrar la funcionalidad de preservación de ajuste implementada en Python dentro de la plataforma SEDDI Author, permitiendo a los usuarios aplicar este proceso directamente en sus proyectos sin tener que extraer e importar escenas.

Al abordar estas áreas en futuros desarrollos, se espera mejorar significativamente la capacidad y la flexibilidad de la herramienta para satisfacer las necesidades de los diseñadores y optimizar el proceso de ajuste de patrones en SEDDI Author.

Bibliografía

- [1] Irene Tinoco. (2024) Pinterest: Modelado. Consultado el 12 de mayo de 2024. [Online]. Available: <https://www.pinterest.com/pin/14566398791460162/>
- [2] Laura Marsh. (2024) Pinterest: Mens basic trouser pattern. Consultado el 12 de mayo de 2024. [Online]. Available: <https://www.pinterest.es/pin/842595411505309457/>
- [3] Modaes. (2024) Lectra da un paso al frente en industria 4.0 con un sistema para personalizar la producción. [Online]. Available: <https://www.modaes.com/equipamiento/lectra-da-un-paso-al-frente-en-industria-40-con-un-sistema-para-personalizar-la-produccion>
- [4] D. Bastarrica. (2022) Walmart estrena probador virtual para vender más ropa. [Online]. Available: <https://es.digitaltrends.com/realidad-virtual/walmart-probador-virtual/>
- [5] DressX. (2024) Dressx. [Online]. Available: <https://pro.dressx.com/fashion/>
- [6] Futuroprossimo. (2023) Project primrose di adobe: il futuro della moda, abiti interattivi e dinamici. [Online]. Available: <https://es.futuroprossimo.it/2023/10/project-primrose-di-adobe-il-futuro-della-moda-abiti-interattivi-e-dinamici/>
- [7] Domestika. (s.f.) Curso de patronaje con adobe illustrator. Imagen obtenida de la lección de proyecto final del curso de Patronaje con Adobe Illustrator en Domestika. [Online]. Available: https://www.domestika.org/es/courses/5275-patronaje-con-adobe-illustrator/final_project_lessons
- [8] Wild Ginger Software. (2024) Cameo. [Online]. Available: <https://wildginger.com/cameo/default.htm#section1>
- [9] “Optitex.” [Online]. Available: <https://optitex.com/es/>
- [10] Browzwear. (2024) Browzwear. [Online]. Available: <https://browzwear.com/>
- [11] W. Ginger. Patternmaster settings. Consultado el 15 de mayo de 2024. [Online]. Available: <https://www.wildginger.com/gallery/pmsettings.htm>
- [12] Techpacker. Pattern grading in the fashion garment industry. Consultado el 15 de mayo de 2024. [Online]. Available: <https://techpacker.com/blog/design/pattern-grading-in-the-fashion-garment-industry/>
- [13] Apparel and accessories how to measure guide. Consultado el 15 de mayo de 2024. [Online]. Available: <https://www.dxl.com/media/content/vendors/Apparel%20and%20Accessories%20How%20to%20Measure%20Guide.pdf>
- [14] SEDDI Textura. One of Textura's case studies. Consultado el 12 de mayo de 2024. [Online]. Available: <https://textura.ai/one-of-case-study/>
- [15] Kiddle, “Curva de bézier,” 2023, consultado el 7 de junio de 2024. [Online]. Available: https://ninos.kiddle.co/Curva_de_B%C3%A9zier

- [16] Google Arts & Culture. Fabricando moda: una historia de la moda y la industria textil. Consultado el 5 de mayo de 2024. [Online]. Available: https://artsandculture.google.com/story/_QKS0J-OeT7HIA?hl=es
- [17] J. D. Guillot, “El impacto de la producción textil y de los residuos en el medio ambiente,” *Parlamento Europeo*, diciembre 2020. [Online]. Available: <https://www.europarl.europa.eu/topics/es/article/20201208STO93327/el-impacto-de-la-produccion-textil-y-de-los-residuos-en-el-medio-ambiente>
- [18] Autor, “Título del artículo,” *El País*, junio 2023. [Online]. Available: <https://elpais.com/sociedad/moda-futuro-y-accion/2023-06-08/pacto-textil-2030-las-lineas-rojas-de-la-union-europea-para-una-moda-mas-sostenible.html>
- [19] Academias ISA, “Patronaje industrial computerizado,” <https://academiasisa.com/patronaje-industrial-computerizado/>, 2024.
- [20] M. S. Alemany Mut, “Desarrollo de modelos estadísticos de predicción del ajuste y talla de prendas de ropa a partir de la percepción y características antropométricas del usuario,” Ph.D. dissertation, Universitat Politècnica de València, 2024.
- [21] A. México. (2024) 5 prendas inteligentes para llenar tu armario de tecnología. [Online]. Available: <https://www.anahuac.mx/mexico/noticias/5-prendas-inteligentes-para-llevar-tu-armario-de-tecnologia>
- [22] D. y UNIVERSIDAD REY JUAN CARLOS, “Programa de colaboración entre empresas,” Documento PDF, 2022, documento PDF proporcionado por el tutor.
- [23] URJC. (2018) La spin-off desilico entra en el mercado internacional. Consultado el 1 de mayo de 2024. [Online]. Available: <https://www.urjc.es/todas-las-noticias-de-actualidad/3035-la-spin-off-desilico-entra-en-el-mercado-internacional>
- [24] Seddi. (s/f) Seddi - grants. Consultado el 1 de mayo de 2024. [Online]. Available: <https://seddi.com/grants/>
- [25] Atlassian. (2024) Kanban vs scrum. [Online]. Available: <https://www.atlassian.com/es/agile/kanban/kanban-vs-scrum>
- [26] Google LLC. (2024) Google meet. [Online]. Available: <https://meet.google.com/>
- [27] Microsoft Corporation. (2024) Microsoft teams. [Online]. Available: <https://teams.microsoft.com/>
- [28] Rocket.Chat. (2024) Rocket.chat. [Online]. Available: <https://es.rocket.chat/>
- [29] Trello. (2024) Trello. [Online]. Available: <https://trello.com/>
- [30] GitLab Inc. (2024) Gitlab. [Online]. Available: <https://about.gitlab.com/>
- [31] The Git Project. (2024) Git. [Online]. Available: <https://git-scm.com/>
- [32] R. Brouet, A. Sheffer, L. Boissieux, and M.-P. Cani, “Design preserving garment transfer,” *ACM Transactions on Graphics*, vol. 31, no. 4, p. Article No. 36, 2012, <https://hal.archives-ouvertes.fr/hal-00695903v2>.
- [33] “Adobe Illustrator.” [Online]. Available: <https://www.adobe.com/products/illustrator.html>
- [34] “Lectra.” [Online]. Available: <https://www.lectra.com/es>
- [35] “Gerber AccuMark.” [Online]. Available: <https://www.gerbertechnology.com/es-es/landing-pages/elija-su-suscripc%C3%B3n/>
- [36] CLO Virtual Fashion. (2024) Clo3d. [Online]. Available: <https://www.clo3d.com/es/>

BIBLIOGRAFÍA

- [37] “Marvelous designer,” Recuperado de <https://marvelousdesigner.com/>.
- [38] Delogue. (2024) Clo3d vs browzwear vs optitex vs style3d: 3d fashion software. [Online]. Available: <https://www.delogue.com/en/blog/clo3d-vs-browzwear-vs-optitex-vs-style3d-3d-fashion-software>
- [39] C. 3D. Parametric pattern creation (ver. 7.0). Consultado el 15 de mayo de 2024. [Online]. Available: <https://support.clo3d.com/hc/en-us/articles/6089589594265-Parametric-Pattern-Creation-ver-7-0>
- [40] Browzwear. Parametric avatars overview. Consultado el 15 de mayo de 2024. [Online]. Available: <https://help.browzwear.com/hc/en-us/articles/5095631742873-Parametric-Avatars-Overview>
- [41] Delogue. How to spec a garment with points of measure. Consultado el 15 de mayo de 2024. [Online]. Available: <https://www.delogue.com/en/blog/how-to-spec-a-garment-with-points-of-measure>
- [42] “Seddi,” <https://www.linkedin.com/company/Seddi/>, consultado el 12 de mayo de 2024.
- [43] “Seddi,” <https://seddi.com/>, consultado el 12 de mayo de 2024.
- [44] “Textura,” <https://textura.ai/>, consultado el 12 de mayo de 2024.
- [45] “Seddi Author,” <https://www.seddiauthor.com/>, consultado el 12 de mayo de 2024.
- [46] React. React official website. Consultado el 17 de mayo de 2024. [Online]. Available: <https://es.react.dev/>
- [47] TypeScript. Typescript official website. Consultado el 17 de mayo de 2024. [Online]. Available: <https://www.typescriptlang.org/>
- [48] Redux. Redux official website. Consultado el 17 de mayo de 2024. [Online]. Available: <https://redux.js.org/>
- [49] ——. Redux official website (español). Consultado el 17 de mayo de 2024. [Online]. Available: <https://es.redux.js.org/>
- [50] M. W. Docs. Canvas api - web api — mdn. Consultado el 17 de mayo de 2024. [Online]. Available: https://developer.mozilla.org/es/docs/Web/API/Canvas_API
- [51] ——. Webgl api - web api — mdn. Consultado el 17 de mayo de 2024. [Online]. Available: https://developer.mozilla.org/es/docs/Web/API/WebGL_API