

UNIVERSIDAD TECNICA FEDERICO SANTA MARIA

Memoria de Título

“ESTIMACIÓN AUTOMATIZADA DE FORMA HUMANA Y MEDICIONES CORPORALES DE PERSONAS VESTIDAS, UTILIZANDO VISIÓN POR COMPUTADOR”

Memoria de Titulación para optar al título de
Ingeniero Civil Electrónico

Alumno
Luciano Gino Muñoz Mendoza

Profesor Supervisor
Marcos Zuñiga Barraza

Profesor Co-Referente
Gonzalo Carvajal Barrera

30 de septiembre, 2021, Valparaíso, Chile

AGRADECIMIENTOS

Primero me gustaría agradecer por todas las experiencias vividas durante estos largos años de universidad, ha sido un bello camino de constante aprendizaje y nuevos desafíos.

Agradezco a mis padres, Tere y Gino, por la educación que me entregaron, por su paciencia y comprensión, sobretodo agradecerles que jamás dejaron de creer en mí. A mi hermanas, a Cote, que siempre ha sido un pilar en mi vida, y a Ale, que me inspira a seguir adelante con su fortaleza. A mis sobrinos, que me motivan a mejorar para ser un ejemplo en sus vidas. A Cami, por su compañía durante estos años y por ayudarme a crecer.

A Marcos Zúñiga mi profesor guía, por el tiempo invertido y su constante preocupación y dedicación en este trabajo. Además de ser un excelente profesional, una excelente persona.

A aquellos con los que compartí a lo largo de estos años universitarios, a mis amigos Susana, Ignacio, Diego, Philip, Benito, Seba y Sandra, gracias por los buenos momentos. A mis amigos de la iniciativa USMZero, a Jorge, Jano y Pablo, por enseñarme a soñar en grande. Por último, a todos aquellos con los que compartí durante esta etapa universitaria.

RESUMEN

ACTUALMENTE la generación de modelos digitales de seres humanos es un tópico que ha cobrado bastante relevancia en áreas como: animación digital, videojuegos y vestuario. En la industria del vestuario, el generar un modelo digital que represente de manera clara la forma corporal de una persona, permite automatizar el diseño de prendas a medida y la elección de talla de vestir.

En este trabajo se enfrenta el problema de estimar un modelo digital del cuerpo de un usuario con el objetivo de obtener sus mediciones antropométricas y talla de vestir. Se desarrolla una aplicación ejecutable desde el motor de videojuegos Unity 3D, la que automatiza una captura RGB-D, y luego, se integra con una aplicación en el entorno de Python, la que estima el modelo corporal y la talla del usuario. Se logra integrar métodos de inferencia de modelos generativos como SMPL en el entorno de creación de videojuegos. También, se comprueba que los resultados de estimación basados en métodos del estado del arte como BCNet [1] son suficientes para aplicaciones en la industria del vestuario.

Palabras Claves

Modelo humano generativo, Modelo SMPL, Unity 3D, Python, Cámara RGB-D.

ABSTRACT

CURRENTLY , the generation of digital models of human beings is a topic that has become quite relevant in areas such as: digital animation, video games and clothing industry. In the apparel industry, generating a digital model that clearly represents a person's body shape allows automating the design of custom garments and the choice of dress size.

This work addresses the problem of estimating a digital model of a user's body in order to obtain their anthropometric measurements and dress size. An executable application is developed in the Unity 3D videogame engine, which automates an RGB-D capture, and then connects with an application of the Python environment, which estimates the body model and the size of the user. The integration of generative model inference methods such as SMPL in the videogame creation environment is achieved. Also, it is found that the estimation results based on state of the art methods such as BCNet cite bcnet are sufficient for applications in the clothing industry.

Keywords

Generative human model, SMPL model, Unity 3D, Python, RGB-D camera.

ÍNDICE

AGRADECIMIENTOS	I
RESUMEN	II
ABSTRACT	III
ÍNDICE DE FIGURAS	VI
ÍNDICE DE TABLAS	1
1. INTRODUCCIÓN	2
1.1. Contexto	2
1.2. Planteamiento del problema	2
1.3. Hipótesis	3
1.4. Acercamiento a la solución	3
1.5. Objetivos	4
1.6. Estructura del documento	4
2. ESTADO DEL ARTE	5
2.1. Marco Teórico	5
2.1.1. Geometría Proyectiva	5
2.1.2. Captura de nube de puntos	8
2.1.3. Modelos digitales generativos	9
2.1.4. Motor de videojuegos, Unity 3D	12
2.1.5. Antropometría	12
2.1.6. Tallas de Vestir	13
2.2. Alternativas de Solución	14
2.2.1. Estimación de forma humana	15
2.2.2. Estimación de mediciones corporales	21
3. SOLUCIÓN PROPUESTA	23
3.1. Pruebas preliminares	24
3.2. Herramientas de integración	31
3.3. Implementación	33
3.3.1. Implementación en el entorno de Python	34
3.3.2. Implementación en entorno de Unity	39

4. PRUEBAS Y RESULTADOS	43
4.1. Estructura de pruebas	43
4.2. Resultados	44
4.2.1. Modelo corporal	45
4.2.2. Medidas y Talla	46
4.2.3. Medidas Ajustadas	49
4.2.4. Tiempo de ejecución	49
4.3. Validación	51
5. CONCLUSIONES Y TRABAJOS FUTUROS	52
5.1. Conclusiones	52
5.2. Trabajo Futuro	53
BIBLIOGRAFÍA	55

Índice de figuras

2.1. Relación proyectiva entre píxeles de la imagen y coordenadas reales.	6
2.2. Corrección de distorsión en tablero de ajedrez.	7
2.3. Imagen de tablero de ajedrez.	8
2.4. Sensor Azure Kinect.	8
2.5. Efecto de blendshapes dependientes de la pose. Imagen obtenida del artículo original de SMPL [2].	10
2.6. Componentes principales de forma obtenidos. Imagen obtenida del artículo original de SMPL [2].	10
2.7. Clases principales de Unity.	13
2.8. Ejemplos de mediciones antropométricas.	14
2.9. Tabla de tallas de marca Maui & Sons.	14
2.10. Fases principales del proyecto.	15
2.11. Reconstrucción de nube de puntos frontal utilizando IF-Net. Imagen obtenida del artículo original de IF-Nets [3].	16
2.12. Reconstrucción de imagen RGB-D utilizando NormalGAN. imagen obtenida del artículo original de SMPL [2].	17
2.13. Diagrama funcional de IP-Net. Imagen obtenida del artículo original de IP-Net [4].	17
2.14. Esquema general IP-Net.	18
2.15. Estimación de modelo corporal utilizando el método de C. Zhang. Imagen obtenida del artículo original de C. Zhang et al. [5].	18
2.16. Mapeo a correspondencias de modelo generativo. Imagen obtenida del artículo original de LoopReg [5].	19
2.17. Diagrama funcional de I2L-Mesh. Imagen obtenida del artículo original de I2L-MeshNet [6].	20
2.18. Estimación de modelo vestido utilizando PIFuHD. Imagen obtenida del artículo original de PIFuHD [7].	21
2.19. Caminos circulares entre vértices de interés. Imagen obtenida del artículo original de S. Yan et al. [8].	22
3.1. Diagrama funcional simplificado de la solución.	23
3.2. Reconstrucción utilizando IF-Net.	25
3.3. Reconstrucción utilizando NormalGAN.	25
3.4. Reconstrucción utilizando IP-Net, nube de puntos frontal.	26
3.5. Resultados de estimación utilizando IP-Net.	27

3.6. Resultados de estimación utilizando LoopReg.	27
3.7. Comparativa de estimación de modelo corporal SMPL.	28
3.8. Reconstrucción utilizando PifuHD.	29
3.9. Propuesta de integración secuencial.	30
3.10. Propuesta de integración con ajuste.	31
3.11. Arquitectura de la integración entre las implementaciones en los entornos de Unity y Python.	34
3.12. Módulos funcionales considerando un modelo SMPL a la salida.	35
3.13. Ejemplo de etapa de pre-procesamiento.	36
3.14. Ajuste geométrico de un camino de la zona del pecho.	37
3.15. Módulos funcionales considerando un modelo SMPL-X a la salida.	38
3.16. Diagrama de estados de la aplicación.	40
3.17. Diagrama de secuencia del estado de ejecución.	41
3.18. Modificaciones en la escena al finalizar la estimación.	42
4.1. Distribución de los usuarios partícipes de las pruebas.	44
4.2. Ejemplo de estimación de modelo corporal.	45

Índice de tablas

2.1. Comparativa modelos generativos del cuerpo humano.	11
2.2. Definiciones de medidas antropométricas.	15
3.1. Comparativa métodos estudiados.	30
4.1. Métricas de error en estimación de estatura.	46
4.2. Métricas de error en medidas generadas por la aplicación completa.	47
4.3. Métricas de error en medidas generadas según la estatura real.	48
4.4. Métricas de error en medidas basadas en estatura real, ajustadas porcentualmente.	49
4.5. Tiempos de ejecución promedio para estimación según SMPL y SMPL-X	50

INTRODUCCIÓN

1.1. Contexto

DURANTE la historia, el diseño y confección de vestuario ha sido delegado al oficio de la sastrería. Un sastre, se encarga de confeccionar ropa a la medida y al gusto del usuario, comprendiendo la antropometría de su cuerpo a través de la medición manual de contornos y largos corporales, utilizando una cinta métrica. Con la llegada de la industrialización la ropa se empezó a producir en masa, lo que se logra con el uso de tallas predefinidas. El método general para definir la talla de una persona está basado en la utilización de una tabla de tallas. Una tabla de tallas es una herramienta que permite relacionar rangos de medidas corporales con tallas de ropa.

Por otro lado, considerando los avances en tecnologías de realidad virtual (VR) y realidad aumentada (AR), el desarrollo de modelos humanos virtuales cobra mayor relevancia. En ese contexto, la generación de un modelo virtual o forma humana de un usuario real, es una temática con aplicaciones en distintas áreas, entre las que destacan: videojuegos, telepresencia, diagnóstico médico y vestuario.

1.2. Planteamiento del problema

En Chile, las tiendas de retail ofrecen variedad de vestuario basado en tallas predefinidas. Para comprobar la correcta elección de la talla de una prenda a adquirir, las tiendas ofrecen un probador en las instalaciones de la tienda. En la actualidad, durante la pandemia de covid-19, debido a las condiciones sanitarias, no es posible realizar una prueba del vestuario para escoger la talla.

Recientemente, gracias a las mejoras en los dispositivos de captura de imágenes y en las técnicas de aprendizaje automático, los métodos para estimar un modelo digital de un usuario permiten obtener representaciones cada vez más realistas y precisas de seres humanos virtuales. Considerando los métodos existentes, se plantea utilizar un modelo digital para estimar las mediciones corporales de un usuario y así obtener su talla de vestir de manera automatizada.

El presente trabajo de titulación nace como un proyecto de la empresa Nexar, entidad

dedicada al marketing digital interactivo y soluciones inteligentes aplicadas a la industria. Se busca que el método implementado se pueda integrar en las soluciones de Nexar, por lo cual, la empresa plantea como requerimiento la ejecución de la solución en el motor de videojuegos Unity 3D, la cual, es una aplicación que permite desarrollar y crear videojuegos.

Por lo tanto, la problemática a tratar durante el presente trabajo de título, es la de automatizar el proceso de selección de talla de vestir en compras al retail, basándose métodos de aprendizaje automático y visión por computador, según los requerimientos planteados por la empresa Nexar.

1.3. Hipótesis

Se plantea como hipótesis a corroborar durante la presente investigación:

“Es posible estimar con precisión un modelo virtual del cuerpo de un usuario adulto y su talla de vestir, utilizando métodos de aprendizaje automático y visión por computador”.

Según la hipótesis mencionada anteriormente, nacen las siguientes preguntas de trabajo:

- ¿Es necesario utilizar una cámara monocular RGB-D¹ como dispositivo de entrada? ¿Es posible utilizar solo una cámara monocular RGB para estimar un modelo del usuario?
- ¿Es posible estimar un modelo digital tridimensional del cuerpo del usuario, que represente claramente su morfología corporal?
- ¿Es posible estimar con precisión las mediciones antropométricas² relacionadas con la talla de vestir del usuario, considerando un error de estimación menor al 15 % relativo a su valor real?
- ¿Es posible implementar la solución, considerando un tiempo de ejecución total menor a 15[seg]?

1.4. Acercamiento a la solución

Para abordar la solución se plantea el desarrollo de un software que permita realizar una captura del cuerpo de usuario vestido, procesar la imagen, y posteriormente, mostrar en pantalla los resultados de estimación de su modelo corporal y de su talla.

En cuanto al procesamiento, este consta de distintas etapas: pre-procesamiento, estimación de modelo corporal y estimación de mediciones corporales. En el pre-procesamiento, se adapta la imagen del usuario para que ingrese en el formato correcto a la siguiente etapa. La estimación del modelo corporal del usuario, se ejecuta a través de la inferencia de resultados utilizando una red neuronal artificial entrenada, la cual retorna un modelo ajustado a la imagen de entrada. La estimación de talla, se realiza a través de la medición de contornos de interés en el modelo generado y su conversión a escala real.

¹Cámara que captura el espectro visible, el color RGB, y además, la profundidad o distancia entre objetos y la cámara D, *depth*.

²Mediciones corporales definidas según normas de antropometría. Ej: contorno de cintura, cadera, pecho.

1.5. Objetivos

- Objetivo general
 - Desarrollar y validar un software que permita estimar un modelo corporal realista de un usuario vestido y obtener su talla de vestir basándose en mediciones obtenidas desde el modelo.
- Objetivos específicos
 - Escoger métodos tecnológicos que permitan obtener un modelo digital y/o mediciones corporales de un usuario.
 - Plantear una propuesta de solución basada en los métodos seleccionados.
 - Desarrollar la solución e integrarla a un motor de videojuegos según requerimiento de la empresa.
 - Validar y cuantificar resultados a través de pruebas en diferentes usuarios.

1.6. Estructura del documento

La memoria de título realizada presenta cuatro capítulos, a saber:

- a) **Estado del arte:** Se plantean conceptos y técnicas relevantes para la implementación de la solución, además se presentan métodos actuales presentes en la literatura, que permiten resolver la problemática utilizando técnicas de visión por computador y aprendizaje automático.
- b) **Solución Propuesta:** Se ejecutan pruebas preliminares con el objetivo de definir el método a integrar en la solución. Luego, se presentan herramientas de integración del método escogido al motor de videojuegos. Por último, se detalla la implementación de la solución en la plataforma requerida, considerando la interacción del usuario, la inferencia de resultados y la renderización de estos.
- c) **Pruebas y Resultados:** Se evalúa el desempeño de la solución desarrollada a través de diferentes pruebas. Se consideran pruebas para diferentes usuarios en variados contextos de captura, midiendo resultados cualitativos y el error de estimación de talla.
- d) **Conclusiones y trabajos futuros:** Se menciona a que nivel se logra comprobar la hipótesis, se responde las preguntas de trabajo y se sintetizan los resultados más relevantes a lo largo de la investigación. En el contexto del problema, se plantean las actuales limitaciones, además de mejoras y trabajos futuros.

ESTADO DEL ARTE

EN este capítulo, primero se revisa el marco teórico que sustenta este trabajo, tanto para su entendimiento general, como para el diseño, construcción y evaluación de la solución. Se definen conceptos básicos de visión por computador, animación digital y diseño de vestuario.

Luego, se detallan alternativas de solución a la problemática basadas en artículos recientes, que presentan métodos tecnológicos enfocados en generar un modelo 3D del cuerpo de un usuario y obtener sus mediciones corporales, utilizando imágenes del usuario vestido obtenidas desde una cámara.

2.1. Marco Teórico

2.1.1. Geometría Proyectiva

La geometría proyectiva es una rama de la matemática que describe las propiedades e invariantes de las figuras geométricas bajo una proyección. A diferencia de la geometría cartesiana o euclidiana, la geometría proyectiva permite definir puntos al infinito y la intersección de dos rectas paralelas. Para representar una coordenada en geometría proyectiva (X_w, Y_w, Z_w, w) es necesaria una dimensión extra w , y es posible obtener su equivalente cartesiano (X, Y, Z) según:

$$\begin{bmatrix} X_w \\ Y_w \\ Z_w \\ w \end{bmatrix} = \begin{bmatrix} X_w/w \\ Y_w/w \\ Z_w/w \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad (2.1)$$

En el caso más simple, se escoge $w = 1$, y así las coordenadas homogéneas son equivalentes a las cartesianas, pero también es posible utilizar $w = 0$ y así representar un punto en el infinito.

La geometría proyectiva permite modelar la geometría detrás de la formación de una imagen, es decir, permite describir las relaciones entre las coordenadas de un punto en el espacio y su correspondiente píxel en la imagen. Para modelar esto, es necesario comprender los distintos sistemas de coordenadas presentes en la formación de la imagen: coordenadas

del mundo real, coordenadas de la cámara y coordenadas del plano de la imagen. En la Figura 2.1, es posible observar la relación geométrica entre los distintos ejes coordenados.

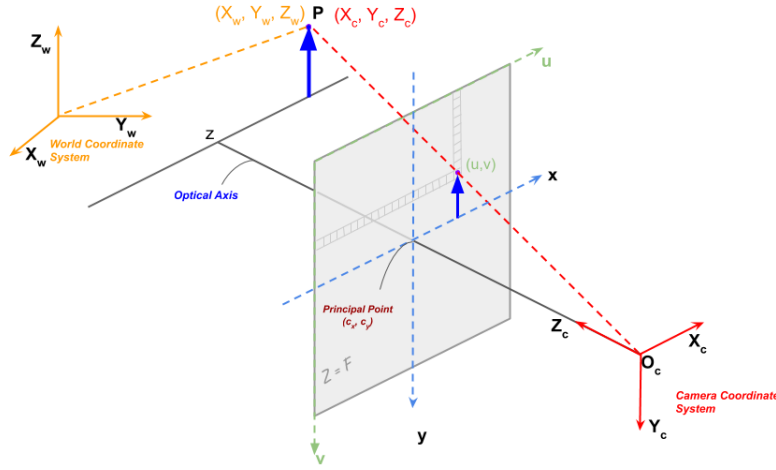


Figura 2.1: Relación proyectiva entre píxeles de la imagen y coordenadas reales.

El sistema coordenado del mundo real (Z_r, X_r, Y_r) , corresponde a un sistema definido según un origen en una referencia arbitraria en el mundo real, mientras que, el sistema de la cámara (Z_c, X_c, Y_c) corresponde a coordenadas con su origen posicionado en el lente de la cámara y alineado con su eje óptico. La relación entre puntos desde un sistema coordenado y otro, está definida por parámetros de traslación y rotación. Para los parámetros, en general se utiliza una matriz de rotación de 3×3 y un vector de traslación de 3×1 , los cuales componen la Matriz Extrínseca $[R|t]$, nótese la ecuación 2.2.

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = [R|t] \begin{bmatrix} X_r \\ Y_r \\ Z_r \\ 1 \end{bmatrix} \quad (2.2)$$

Por otro lado, también es posible describir la relación entre puntos en el sistema coordenado de la cámara y píxeles en la imagen. Para esto se modela la cámara según el modelo “pinhole”, el cual plantea un punto infinitesimal que captura la luz y la proyecta en un plano a una distancia focal f del centro óptico la cámara. Utilizando geometría básica es posible relacionar los puntos en el sistema coordenado de la cámara (X_c, Y_c, Z_c) con los píxeles de la imagen (x, y) , según las siguientes ecuaciones:

$$x = f_x \frac{X_c}{Z_c} \quad y = f_y \frac{Y_c}{Z_c} \quad (2.3)$$

Además, es posible indicar el desplazamiento entre el centro de la imagen y el eje óptico de la cámara (u_0, v_0) , y escalar los parámetros de distancia focal según su largo en píxeles, obteniendo así la Matriz Intrínseca del sistema:

$$\begin{bmatrix} x_w \\ y_w \\ w \end{bmatrix} = \begin{bmatrix} f_x & \gamma & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \quad (2.4)$$

Luego, es posible definir la matriz de proyección perspectiva, la cual se compone de la multiplicación de la matriz intrínseca y extrínseca, permitiendo así mapear una coordenada espacial del mundo real a su equivalente en píxeles en la imagen:

$$\begin{bmatrix} x_w \\ y_w \\ w \end{bmatrix} = \begin{bmatrix} f_x & \gamma & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} [R|t] \begin{bmatrix} X_r \\ Y_r \\ Z_r \end{bmatrix} \quad (2.5)$$

La matriz de proyección perspectiva definida en la ecuación 2.5, corresponde a una transformación utilizando un modelo de cámara pinhole, el cual corresponde a un modelo simplificado de una cámara real. Mientras que, una cámara real utiliza lentes para capturar más luz y así obtener una imagen más clara y definida, pero su utilización genera efectos de distorsión en la imagen.

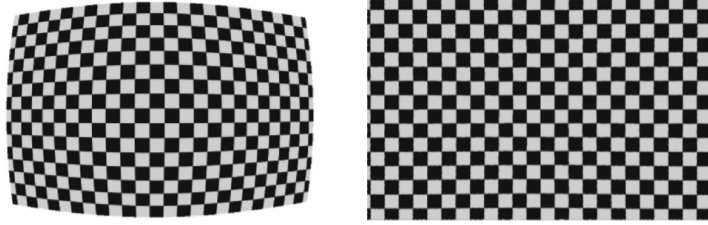


Figura 2.2: Corrección de distorsión en tablero de ajedrez.

Existen tres tipos de distorsión: radial, tangencial y de prisma delgado; siendo la distorsión radial la más común. Nótese la Figura 2.2, donde a la izquierda se observa un tablero de ajedrez con distorsión radial. Es posible modelar el efecto de la distorsión y añadirlo a la matriz de proyección perspectiva. Para esto se realiza una transformación de los puntos en el sistema coordenado de la cámara (X_c, Y_c, Z_c) . Se define $x' = \frac{X_c}{Z_c}$ y $y' = \frac{Y_c}{Z_c}$. Luego, se modela la distorsión según tres tipos de coeficientes de distorsión: radiales k_1, k_2, k_3, k_4, k_5 y k_6 , tangenciales p_1 y p_2 , y de prisma delgado s_1, s_2, s_3 y s_4 , los cuales, con $r^2 = (x')^2 + (y')^2$, definen la siguiente transformación:

$$x'' = x' \frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6}{1 + k_4 r^2 + k_5 r^4 + k_6 r^6} + 2p_1 x' y' + p_2 (r^2 + 2x'^2) + s_1 r^2 + s_2 r^4 \quad (2.6)$$

$$y'' = y' \frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6}{1 + k_4 r^2 + k_5 r^4 + k_6 r^6} + 2p_2 x' y' + p_1 (r^2 + 2x'^2) + s_3 r^2 + s_4 r^4 \quad (2.7)$$

Luego, para proyectar estos puntos a los píxeles de la imagen, es posible utilizar la matriz intrínseca mencionada en la ecuación 2.4.

$$\begin{bmatrix} x_w \\ y_w \\ w \end{bmatrix} = \begin{bmatrix} f_x & \gamma & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x'' \\ y'' \\ 1 \end{bmatrix} \quad (2.8)$$

El proceso de obtener la matriz intrínseca, la matriz extrínseca y los coeficientes de distorsión de un sistema, se conoce como calibración y es un problema con solución conocida. Para obtener cada uno de estos parámetros, es necesario conocer puntos en la imagen

y su correspondencia en coordenadas reales. La biblioteca OpenCV dispone herramientas para hacer el proceso de calibración de manera semi-automatizada. Para esto, recomienda realizar múltiples capturas a un tablero de ajedrez, y tomar las esquinas del tablero como coordenadas conocidas para realizar el cálculo de los parámetros de proyección. Nótese la Figura 2.3, donde se puede observar una imagen de un tablero de ajedrez y la detección de sus esquinas.

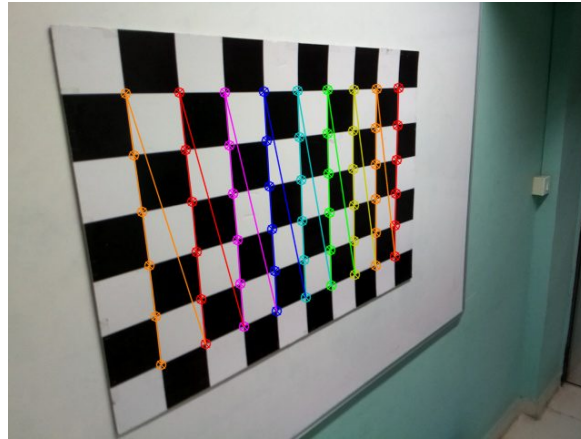


Figura 2.3: Imagen de tablero de ajedrez.

2.1.2. Captura de nube de puntos

Una nube de puntos es un grupo de puntos posicionados en el espacio cartesiano (X,Y,Z) , los cuales pueden representar la forma tridimensional de un objeto. Son generadas principalmente por escáneres 3D, cámaras RGB-D o algún software de fotogrametría.

Una cámara RGBD es una cámara que además de entregar datos de color en cada píxel, también entrega información de la profundidad asociada al píxel. Si la posición de la cámara y su calibración es conocida, esta información puede interpretarse como puntos de color posicionados en el espacio tridimensional o una nube de puntos. Entre los sensores RGBD más utilizados, destacan Azure Kinect e Intel Real Sense, note la Figura 2.4.

Un escáner tridimensional, permite obtener una nube de puntos con una perspectiva en 360°



Figura 2.4: Sensor Azure Kinect.

de un objeto. Para obtener una nube de puntos del cuerpo de un usuario, existen escáneres especializados para esta labor, los cuales son costosos y de gran tamaño.

Para prescindir del uso de un costoso escáner 3D y obtener una perspectiva completa del usuario basándose en una cámara RGBD, es posible realizar un proceso similar a la fotogrametría. La idea es fusionar distintas imágenes de una secuencia de vídeo obtenida al mover una cámara alrededor del usuario, y así generar una nube de puntos con la perspectiva general del cuerpo de este. Existen distintas variaciones y desarrollos basados en ese concepto, como por ejemplo: KinectFusion [9], DynamicFusion [10] y DoubleFusion [11]. Cabe señalar que, estas técnicas son costosas en hardware y tiempo de procesamiento, pero permiten obtener una nube de puntos detallada del usuario.

2.1.3. Modelos digitales generativos

Un modelo de animación digital está basado en dos capas; el esqueleto o articulaciones (*joints*) y la capa más externa o la piel (*mesh*). Para representar deformaciones de la capa externa del modelo según el movimiento de los joints, existen técnicas como Linear Blend Skinning(LBS) o Dual Quaternion Blend Skinning(DQBS). Por otro lado, se define un blendshape como un set de offsets de vértice en una pose de descanso, los cuales son utilizados para modificar la forma de un modelo 3D.

Un modelo generativo, es un modelo que es capaz de generar nuevos datos basándose en el aprendizaje desde una gran muestra de datos similares. Para la creación de modelos digitales del cuerpo humano, se utilizan modelos generativos que permiten crear diferentes cuerpos, modificando su morfología y su pose, basándose en ciertos parámetros de entrada.

Con respecto a modelos del cuerpo humano, se detallan modelos realistas, ampliamente utilizados y disponibles públicamente. Para este proyecto es relevante mencionar:

- **SMPL: Skinned Multi-Person Linear Model** [2]

Actualmente es uno de los modelos paramétricos de *Minimally Clothed Shape*(MCS) más utilizados en animación digital y visión por computador. Esto debido al alto realismo de los modelos generados y su compatibilidad con distintos motores de renderización como Unity o Maya.

SMPL es aprendido desde datos y permite representar cualquier forma humana en cualquier pose. La morfología de un modelo SMPL está basada en la modificación de un modelo base, según dos tipos de blendshapes aprendidos, blendshapes dependientes de la pose y dependientes de la forma humana.

Los blendshapes de pose permiten dar realismo a las variaciones en la geometría del cuerpo generadas por el movimiento de las articulaciones, es decir, permiten contraer o ampliar zonas del cuerpo cercanas a articulaciones según distintas poses. Estos son aprendidos de un set de datos de 1.800 muestras de 20 hombres y 24 mujeres en distintas poses. En la Figura 2.5(b) se puede observar el efecto correctivo de los blendshapes de pose en las articulaciones comparando con la Figura 2.5(a).

Por otro lado, los blendshapes de forma son aquellos que generan la identidad del modelo, es decir, permiten representar distintos tipos de cuerpos. El aprendizaje de estos blendshapes está basado en el set de datos CAESAR, del cual se utilizan alrededor de 2000 muestras en una misma pose para cada género. Para el aprendizaje de estos parámetros, se utiliza Principal Component Analysis(PCA) para obtener las principales

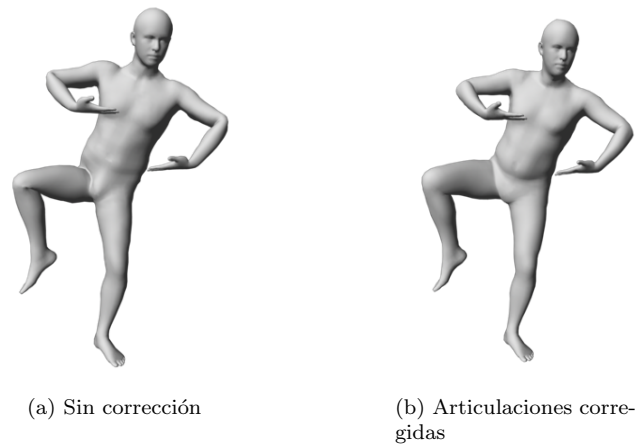


Figura 2.5: Efecto de blendshapes dependientes de la pose. Imagen obtenida del artículo original de SMPL [2].

variaciones de forma observadas en la población. La cantidad de blendshapes de forma a utilizar depende de la versión de SMPL y el nivel de detalle requerido de cada aplicación. En general, según el autor del artículo, con 10 blendshapes es suficiente, pero SMPL tiene versiones con hasta 300 blendshapes. En la Figura 2.6, se pueden observar los primeros tres componentes principales de forma obtenidos.

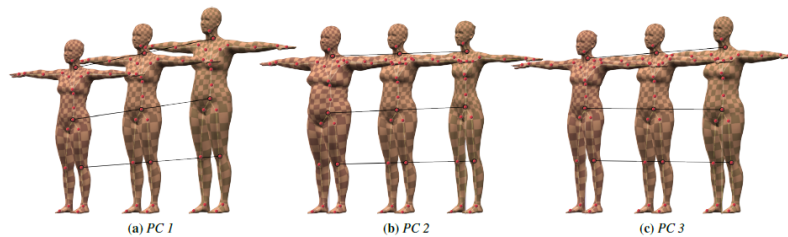


Figura 2.6: Componentes principales de forma obtenidos. Imagen obtenida del artículo original de SMPL [2].

Además, al finalizar la fase de entrenamiento, es posible generar un modelo SMPL simplemente definiendo los parámetros de pose (θ), *skeleton joints* y los parámetros de forma (β), nótese la Figura 2.6.

El modelo SMPL está disponible en distintas versiones para su descarga con fines académicos, las versiones ofrecidas ya se encuentran entrenadas, por lo cual su utilización se remite al correcto uso de los parámetros de forma y pose humana.

Por último, en cuanto a performance, la generación de un modelo toma alrededor de 4[ms] por frame utilizando un Macbook Pro 2014, según el artículo del autor.

■ STAR: Sparse Trained Articulated Human Body Regressor [12]

Se presenta como la versión mejorada de SMPL. Está basado en técnicas de aprendizaje

similares a las de SMPL. En este caso se presentan mejoras al aprendizaje de los blendshapes dependientes de la pose, los cuales se enfocan en correcciones cercanas a cada articulación. Además utiliza solo 93 blendshapes correctivos de pose, versus los 207 blendshapes de SMPL.

Los parámetros de forma de STAR también están entrenados según CAESAR, pero además utilizan el dataset SizeUSA, el cual al igual que CAESAR dispone de capturas tridimensionales de distintos usuarios, en este caso, de 10.000 ciudadanos estadounidenses. Obteniendo así, una forma humana más completa y actualizada.

En cuanto a tiempos de procesamiento, no se mencionan resultados en el artículo del autor, pero se asume que debería ser mas rápido que SMPL debido a su mayor simplicidad.

El modelo se encuentra disponible públicamente para ser utilizado para fines no comerciales.

A modo de resumen, se detallan métricas importantes en el Cuadro 2.1

Nombre	Fecha	Vértices	Muestras Forma ¹	BS de Forma ²	BS de Pose ³
SMPL	Oct. 2015	6890	4.000	10 a 300	207
STAR	Ago. 2020	6890	14.000	10 a 300	93

¹ = Muestras utilizadas para aprender los blendshapes (BS) de Forma del modelo. ² = Blendshapes correctivos dependientes de la forma. ³ = Blendshapes correctivos dependientes de la pose.

Tabla 2.1: Comparativa modelos generativos del cuerpo humano.

■ CAPE: Clothed Autoperson Encoding [13]

CAPE es un modelo generativo de humanos vestidos en 3D. El enfoque de este modelo humano es obtener una versión vestida de SMPL. Para esto se crea una capa superior sobre SMPL que representa la ropa del modelo. Esta capa no está basada en blend shapes, más bien representa un desplazamiento relativo de todos los vértices desde la capa SMPL. Para aprender y parametrizar esta capa, se utiliza un dataset que contiene frames de video de personas utilizando distintas ropas en diferentes poses (CAPE Dataset).

El modelo se encuentra disponible públicamente para ser utilizado para fines no comerciales.

■ SMPL-X: SMPL eXpressive [14]

SMPL-X es un modelo generativo de cuerpos humanos que además de considerar el movimiento y la forma corporal, añade el movimiento de las manos y expresiones faciales. La construcción del modelo está basada en 10.475 vértices y 54 joints; en cuanto a los joints, están compuestos de 24 correspondientes al cuerpo, mandíbula y órbitas de los ojos, mientras que, 30 corresponden a las articulaciones de las manos. Al igual que en SMPL, se conservan 10 blendshapes principales para la forma corporal, pero además se añaden 10 blendshapes correspondientes a la expresión facial. Por otro lado, en este caso se contabilizan 486 blendshapes correctivos de pose, ya que al igual que en SMPL, estos son proporcionales a los joints del modelo.

La utilización del modelo SMPL-X se remite al uso de 119 parámetros, clasificados en tres tipos de vectores principales: parámetros de pose (θ), parámetros de forma (β) y parámetros de expresión facial (ψ). Los parámetros de pose se puede dividir en

dos partes. Primero, los de pose del cuerpo, los cuales conforman de un arreglo de 75 valores, expresando la rotación relativa entre los joints en notación *axis-angle*. Segundo, los parámetros de pose de las manos, los cuales se representan en una dimensionalidad reducida en un arreglo de 24 valores. Por otro lado, los parámetros de forma corporal y expresión facial, conforman un arreglo de 10 valores correspondientes a los blendshapes en ambos casos.

2.1.4. Motor de videojuegos, Unity 3D

Unity es un motor de videojuegos multiplataforma desarrollado por *Unity Technologies*, el cual entrega todas las herramientas necesarias para desarrollar videojuegos, simuladores o experiencias interactivas, de manera accesible para distintos tipos de desarrolladores y diseñadores.

Utilizando Unity, es posible desarrollar videojuegos con gráficas en 2D o en 3D, compatibles con distintas plataformas: smartphones, computadores de escritorio, consolas de videojuegos y otros.

Un juego de Unity posee diferentes entidades o clases, las cuales definen la jerarquía y la interacción durante el desarrollo. Cabe mencionar: *Scene*, *Game Object* y *Component*, las cuales se detallan en la Figura 2.7.

Una escena o *Scene*, representa el lugar o espacio donde se posicionan los objetos, es decir, donde existen los personajes, obstáculos, decoración, cámaras, iluminación, etc. En videojuegos elaborados, es común utilizar una escena distinta para cada nivel.

Un Objeto de juego o *Game Object*, es la clase principal de Unity, la cual contiene a todos los objetos presentes en el juego, ya sean personajes, cámaras o efectos especiales. El comportamiento de cada objeto está definido según los componentes añadidos a él.

Un componente o *Component* es capaz de implementar diferentes funcionalidades para un objeto. Existen componentes predefinidos entregados por Unity, por ejemplo: *Transform*, *Physics*, *Audio*, etc. También es posible crear componentes personalizados, a través de la *Scripting API* ofrecida por Unity, herramienta que permite realizar Scripts escritos en C# para interactuar con los objetos de Unity.

El flujo de trabajo se basa principalmente en dos pasos iterativos. Primero, modificar una escena, ya sea creando objetos y/o implementando funcionalidades. Segundo, ejecutar la simulación y observar el resultado renderizado por el motor de videojuegos.

2.1.5. Antropometría

Al estudio de las medidas corporales de una persona, se le llama antropometría. Actualmente, la antropometría tiene relevancia en áreas como: ergonomía, medicina y diseño de vestuario. Además, la toma de mediciones antropométricas se encuentra estandarizada según la norma ISO 8559:1989¹. En la Figura 2.8, se ejemplifican mediciones antropométricas. Las mediciones son obtenidas basándose en puntos o áreas del cuerpo humano. Además, en la Figura 2.8 se destacan dos puntos claves en distinto color. Primero, destacado en azul, en la parte posterior del cuello, se encuentra *la séptima vértebra*. Luego, destacado en rojo, en la zona superior del hombro, se encuentra *el acromión*, que corresponde a la parte superior de la escápula a la altura del hombro. Según K. Simmons et al. [15], existen 16 mediciones antropométricas relevantes en el área del diseño de vestuario a medida. Se detalla la definición de cada una de ellas en la Tabla 2.2.

¹<https://www.iso.org/obp/ui/#iso:std:iso:8559>

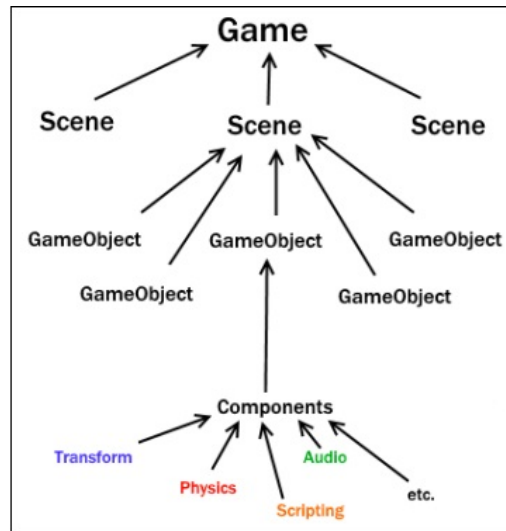


Figura 2.7: Clases principales de Unity.

2.1.6. Tallas de Vestir

Antes de la industrialización, toda la ropa era hecha a medida. Esta era fabricada por algún sastre o símil, basándose en medidas corporales del usuario. Actualmente, la sastrería se utiliza solo en industrias específicas como las de ropa deportiva y vestimenta formal. Para la fabricación de una prenda de vestir a medida, es necesario saber las medidas corporales de la persona que utilizará la prenda, y además, definir una cierta holgura, lo que genera la medida real de la prenda.

Con la llegada de la industrialización la ropa se empezó a producir en masa, lo que se logró con el uso de tallas predefinidas. Una talla de ropa es una convención que busca ajustarse a un segmento de usuarios con características corporales similares. En general una talla de ropa está basada en mediciones corporales.

El método general para definir la talla de una persona está basado en la utilización de una *tabla de tallas*. Una tabla de tallas es una herramienta que permite relacionar rangos de medidas corporales con tallas de ropa. Nótese la Figura 2.18, donde se muestra una tabla de tallas de la marca Maui & Sons.

Existen distintos estándares para tablas de tallas, pero mencionarlos no es necesario, ya que actualmente cada fabricante o marca utiliza sus propias tablas. Esto se debe a que cada marca busca utilizar una tabla de tallas que represente mejor al segmento de mercado al que apuntan.

Con respecto a la ropa ofrecida por el retail, para obtener la talla de ropa de un usuario, en general basta saber las medidas de: contorno de pecho, contorno de cintura, contorno de cadera, largo de hombro a brazo y largo de costura. Mientras que para confeccionar ropa a medida, son relevantes las 16 mediciones mencionadas en la Tabla 2.2.

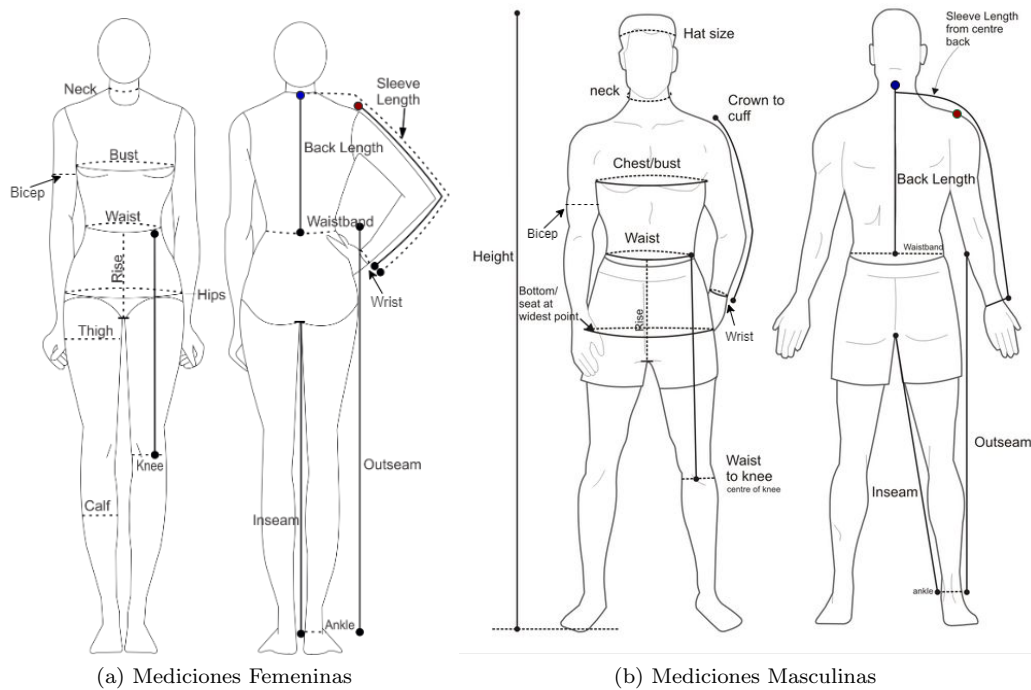


Figura 2.8: Ejemplos de mediciones antropométricas.

TALLA	Pecho (cm)	Cintura (cm)	Cadera (cm)
XS	80	76	84
S	84	80	88
M	88	84	92
L	92	88	96



Figura 2.9: Tabla de tallas de marca Maui & Sons.

2.2. Alternativas de Solución

En esta sección, se detallan métodos basados en publicaciones recientes, que permiten obtener la forma humana o representación digital del cuerpo de un usuario, además de métodos que permitan obtener sus mediciones antropométricas o corporales.

Según la investigación realizada, el proceso puede dividirse en tres fases principales. Primero reconstruir un modelo vestido del usuario utilizando una captura RGB-D o una imagen RGB. Luego, utilizar el modelo vestido e inferir su forma humana bajo la ropa, generando un modelo corporal o mínimamente vestido del usuario. Por último, obtener ciertas mediciones corporales del usuario, utilizando el modelo corporal generado. En la Figura 2.10, se muestra el esquema general de la solución en cuestión.

Nº	Medición	Traducción	Definición
1	Neck	Cuello	Circunferencia horizontal del cuello a la altura de la séptima vértebra.
2	Chest/Bust	Pecho o busto	Máxima circunferencia horizontal a la altura del pecho.
3	Waist	Cintura	Minima circunferencia horizontal a la altura de la costilla mas inferior.
4	Hip	Cadera	Máxima Circunferencia horizontal a la altura de los trocánteres.
5	Arm length	Brazo	Distancia entre el acromión y la muñeca, pasando por la articulación del codo.
6	Sleeve length	Manga	Distancia entre la base del cuello y la muñeca, pasando por el acromión y la articulación del codo.
7	Inseam	Entrepierna	Distancia entre la entrepierna y la planta del pie.
8	Outseam	Costura	Distancia entre la parte lateral de la cintura y la planta del pie.
9	Shoulder length	Hombro	Distancia entre la parte lateral de la base del cuello y el acromión.
10	Across chest	Medio pecho	Mitad delantera del contorno pecho, entre puntos laterales a la altura de la articulación del brazo.
11	Neck to waist	Talle trasero	Distancia vertical entre la séptima vértebra y la cintura.
12	Rise	Tiro	Distancia entre la cintura y la entrepierna.
13	Crotch length	Entrepierna	Distancia entre la parte trasera y delantera de la cintura pasando por la entrepierna.
14	Thigh	Muslo	Máxima circunferencia del muslo cercana a la entrepierna.
15	Bicep	Bicep	Máxima circunferencia paralela al piso del bicep en posición de descanso.
16	Wrist	Muñeca	Circunferencia de muñeca sobre el hueso de la muñeca.

Tabla 2.2: Definiciones de medidas antropométricas.

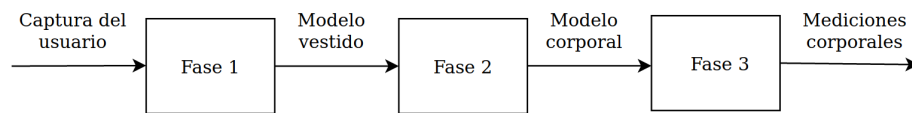


Figura 2.10: Fases principales del proyecto.

2.2.1. Estimación de forma humana

En este punto se analizan los métodos presentes para la Fase 1 y/o 2 de la solución, según el esquema expresado en la Figura 2.10. Para los métodos de estimación de forma humana o modelo digital del usuario, en primera instancia se detalla el paso a paso de diferentes alternativas basadas en una nube de puntos incompleta del usuario, como la obtenida por

la captura frontal de una cámara RGBD. Luego, se analizan alternativas que utilizan como entrada una malla completa del usuario vestido, similar a lo generado por un escáner corporal 3D.

Métodos con cámara monocular RGBD o modelo vestido

Se estudiaron cuatro métodos para este caso, los cuales se detallan a continuación:

- J. Chibane et al. [3], presentan “IF-Nets: Implicit Feature Networks”, artículo enfocado en la reconstrucción detallada de diferentes entradas 3D basándose en el aprendizaje de funciones implícitas. En el contexto de este proyecto, se destaca la reconstrucción utilizando nubes de puntos incompletas como entrada, lo cual permite obtener un modelo similar a uno generado por un escáner 3D, es decir, permite implementar la Fase 1 según la Figura 1.

La reconstrucción basada en el aprendizaje de funciones implícitas, en general consiste en definir si un punto de prueba está dentro o fuera de la superficie a reconstruir. Para lograr esto, se alimenta una red con la nube de puntos a reconstruir y luego, se realiza una prueba de ocupación con cada uno de los puntos de una malla tridimensional equidistante. Utilizando los puntos con una ocupación en la superficie, se completa una malla con la superficie reconstruida utilizando el algoritmo *marching cubes*.

Para el caso de IF-Net, se realiza un procedimiento similar, definiendo un índice de ocupación para los puntos que se encuentran en la superficie. Pero en este caso se realiza un proceso de codificación aprendida o, *encoder*, de la nube de puntos de entrada. Se codifica según convoluciones aprendidas de manera recursiva, lo que permite generar diferentes mallas de características o *deep features grids*, las cuales extraen características desde lo general a lo particular. Luego, considerando estas características codificadas, se realiza la prueba de ocupación de la superficie a través de una decodificación aprendida o *decoder*, obteniendo así los puntos que pertenecen a la superficie a reconstruir.

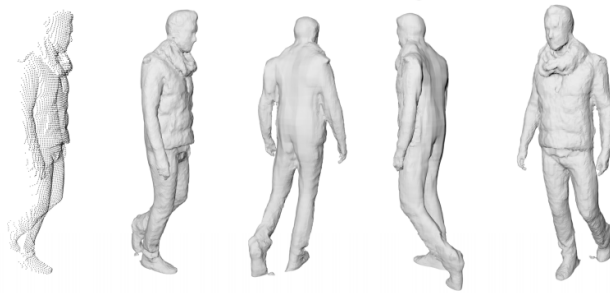


Figura 2.11: Reconstrucción de nube de puntos frontal utilizando IF-Net. Imagen obtenida del artículo original de IF-Nets [3].

- L. Wang et al. [16] presentan NormalGAN: Learning Detailed 3D Human from a Single RGB-D Image. Artículo enfocado en utilizar una captura frontal RGB-D para reconstruir el modelo vestido de un usuario, es decir, permite ejecutar la Fase 1, según el esquema de la Figura 2.10. El método está basado en redes GAN, *Generative Adversarial Networks*, utilizadas para rectificar la vista frontal e inferir la vista posterior del usuario.

El método utiliza una imagen de color y de profundidad como entrada, primero realiza una proyección de las imágenes a un espacio ortogonal, luego rectifica y completa las imágenes utilizando una GAN condicionada según un mapeo normal o *Normal Map*, obteniendo así la vista frontal del modelo. Un mapeo normal es un mapeo de textura utilizado en modelos compuestos de pocos polígonos, el cual permite renderizar el modelo con mayor detalle sin añadir mas vértices. Luego, se realiza la inferencia de la vista posterior utilizando una red similar a la utilizada para la vista frontal. Por último, se unen ambas vistas para obtener un modelo vestido del usuario, el cual contiene detalles asociados al condicionamiento normal.

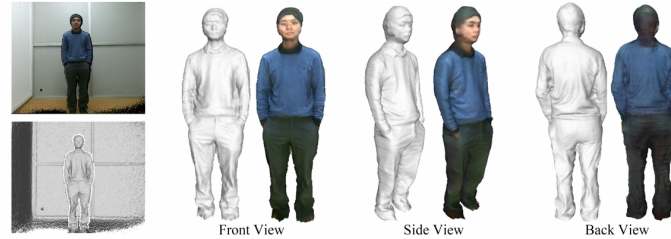


Figura 2.12: Reconstrucción de imagen RGB-D utilizando NormalGAN. imagen obtenida del artículo original de SMPL [2].

- B. Bhatnagar et al. [4], presentan IP-Net: Implicit Part Network, artículo basado en IF-Nets, enfocado también en la reconstrucción utilizando nubes de puntos incompletas. En este caso, además de inferir una superficie externa del usuario vestido, se infiere una superficie interna, correspondiente a la forma humana bajo la ropa, y además se entregan correspondencias a las partes de un modelo SMPL, lo cual permite ajustar de mejor manera un modelo SMPL a la superficie interna obtenida. Por lo tanto, permite realizar las Fases 1 y 2 señaladas en la Figura 2.10.

El método se basa en tres etapas principales. Primero se realiza un preprocesamiento,

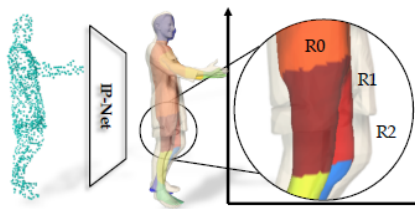


Figura 2.13: Diagrama funcional de IP-Net. Imagen obtenida del artículo original de IP-Net [4].

donde se ajusta la escala de la nube de puntos de entrada, además de voxelizar la nube de puntos. La siguiente etapa consiste en reconstruir el cuerpo del usuario y además, segmentar la superficie corporal de la superficie vestida de este. Para esto, al igual que en IF-Nets, se obtiene un índice de ocupación utilizando puntos de prueba. Pero en este caso, se utilizan tres opciones de ocupación: ocupar la superficie interna,

ocupar la superficie externa, o estar fuera de alguna superficie. Utilizando los puntos con ocupación en la superficie interna y externa, se construyen mallas de la superficie corporal y la superficie vestida del usuario, respectivamente. Además del índice de ocupación, IP-Net retorna correspondencias entre las 14 partes del modelo SMPL y los puntos de la superficie interna generada. La última etapa, consiste en realizar una optimización de la superficie interna a un modelo SMPL, mejorada según las correspondencias antes mencionadas. Nótese la Figura 2.14 donde se grafica el método.

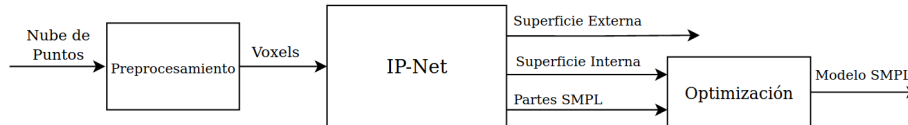


Figura 2.14: Esquema general IP-Net.

- C. Zhang et al. [5] presentan un método para obtener la forma humana bajo la ropa basándose en una secuencia de escaneos 3D. El método se basa en el hecho de que la ropa se vuelve mas ajustada en ciertos sectores cuando la persona cambia de pose. El primer paso consiste en segmentar la ropa y la piel según el color. Luego, se fusiona la información de los distintos escaneos para así capturar el efecto del movimiento de la ropa. Por último, se utiliza esta nube de puntos fusionada para realizar el ajuste a un modelo SMPL según una función de minimización de energía que considera el efecto de la ropa y de la piel como elementos aditivos. Cabe señalar que, en este artículo se presenta el set de datos BUFF (Bodies under Clothing), el cual contiene secuencias de 3 hombres y 3 mujeres utilizando dos tipos de ropa y su correspondiente MCS. Este set de datos es utilizado para evaluar resultados de estimación de la forma humana bajo la ropa.



Figura 2.15: Estimación de modelo corporal utilizando el método de C. Zhang. Imagen obtenida del artículo original de C. Zhang et al. [5].

- B. Bhatnagar et al. [17], presentan LoopReg: Self-supervised Learning of Implicit

Surface Correspondences, Pose and Shape for 3D Human Mesh Registration. Artículo enfocado en el ajuste de escaneos 3D a un modelo paramétrico SMPL, es decir, permite ejecutar la Fase 2 según la Figura 2.10. El método está basado en dos etapas, primero la obtención de correspondencias al modelo generativo, esta etapa busca mapear los puntos de la nube de entrada, a su correspondencia a un modelo SMPL en pose y forma neutral, note la Figura 2.16, donde se ejemplifica la correspondencia de tres puntos. Para el caso de LoopReg, esta etapa es realizada según una red aprendida, la cual mapea los puntos de entrada a una difusión del modelo en el espacio tridimensional, lo que permite mapear correspondencias fuera de la superficie del modelo SMPL.

La siguiente etapa consiste en modificar iterativamente la pose y la forma del modelo SMPL, buscando minimizar la distancia entre los puntos correspondientes. Esto se realiza basándose en aprendizaje, ejecutando un ajuste recursivo de manera auto-supervisada.

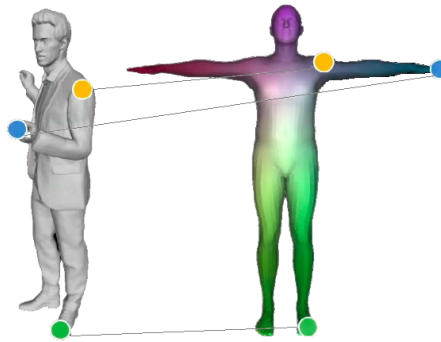


Figura 2.16: Mapeo a correspondencias de modelo generativo. Imagen obtenida del artículo original de LoopReg [5].

Durante el proceso de investigación, destacan diferentes métodos que permiten obtener la forma del usuario utilizando cámaras RGB, estos presentan resultados similares a los obtenidos con cámaras RGBD. Por lo tanto, se abre la posibilidad de utilizar una solución con un menor requerimiento de hardware, es decir, sólo con una cámara RGB.

Métodos con cámara monocular RGB

A continuación se detallan el análisis y el paso a paso de tres métodos actuales que utilizan una imagen a color para reconstruir un modelo digital del usuario:

- G. Moon et al. [6] presentan I2L-MeshNet: Image-to-Lixel Prediction Network for Accurate 3D Human Pose and Mesh Estimation from a Single RGB Image. Artículo enfocado en la generación de un modelo corporal de un usuario, utilizando una imagen a color como entrada, es decir, permite ejecutar la Fase 1 y 2, según la Figura 2.10.

El método se compone de dos partes, primero la estimación de la pose 3D del usuario, para esto se aprende una red, *PoseNet*, que retorna tres mapas de calor unidimensionales basados en lixel o *1-D lixel-based heatmaps*, los cuales corresponden a la posición de los joints estimados. El término líxel nace de la unión de las palabras *line+pixel*. La utilización de mapas de calor basados en lixel, permite preservar las

relaciones espaciales obtenidas de los píxeles de la imagen.

La siguiente parte corresponde a *MeshNet*, red que utiliza la pose estimada y la imagen de entrada, para estimar los vértices del modelo corporal del usuario. Esto se logra con una arquitectura similar a la etapa anterior, retornando también mapas de calor basados en píxel para los vértices del modelo. Cabe destacar que si bien el modelo obtenido está basado en un modelo SMPL, la estimación no está basada en obtener los parámetros de entrada del modelo SMPL, más bien retorna directamente los vértices de este. Por lo cual, no es posible obtener los parámetros de forma del modelo desde I2L-Mesh.

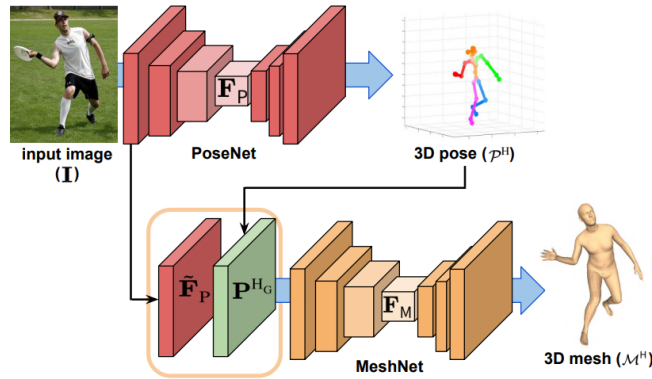


Figura 2.17: Diagrama funcional de I2L-Mesh. Imagen obtenida del artículo original de I2L-MeshNet [6].

- B. Jiang et al. [1], presentan BCNet: Learning Body and Cloth Shape from A Single Image.

Artículo enfocado en la generación de un modelo corporal y un modelo de las vestimentas de un usuario, utilizando una imagen de color, de un usuario idealmente en pose “A”, como entrada, es decir, permite ejecutar las Fases 1 y 2 según la Figura 2.10. Además, implementa un modelo generativo para la ropa del usuario, el cual permite generar distintas vestimentas superiores e inferiores, basándose en parámetros de entrada y *skinning weights*.

El método se basa en tres etapas principales, primero una etapa de codificación, donde se extraen características utilizando *ResNet-18*. Luego, una etapa de clasificación, basada en capas *fully-connected*, que permiten obtener los parámetros de un modelo generativo SMPL, además de una característica latente relacionada con la vestimenta. Para los parámetros de pose del modelo SMPL, se infieren las matrices de rotación que representan las posiciones de los joints del modelo. La tercera etapa, se basa en utilizar la característica latente de la vestimenta, para inferir los parámetros de la vestimenta superior e inferior, además de inferir los *skinning weights*, del modelo de la ropa a generar.

- S. Saito et al. [7] presentan PIFuHD: Multi-Level Pixel-Aligned Implicit Function for High-Resolution 3D Human Digitization, Artículo basado en 'PIFu:Pixel-Aligned

Implicit Function for High-Resolution Clothed Human Digitization' [18], enfocado en la reconstrucción de un modelo vestido de un usuario, utilizando una imagen RGB como entrada. El método se basa en el mismo principio mencionado antes en IF-Nets, realiza la reconstrucción clasificando la ocupación de puntos de prueba en la superficie a generar.

En la implementación de PIFu, se realiza la reconstrucción según índices de ocupación en dos niveles, una etapa general y otra más particular, capturando así la geometría completa del usuario. Respecto a esta implementación, se presentan varias mejoras: primero, la inferencia se realiza desde una imagen con mayor resolución, además, se estiman mapas normales para la superficie frontal y posterior del usuario, esto se utiliza como una entrada extra a la red, lo cual genera una reconstrucción más detallada del cuerpo vestido del usuario.



Figura 2.18: Estimación de modelo vestido utilizando PIFuHD. Imagen obtenida del artículo original de PIFuHD [7].

2.2.2. Estimación de mediciones corporales

En este punto se analizan métodos propuestos para la Fase 3 de la solución, según el esquema expresado en la Figura 2.10, es decir, aquellos métodos que permiten obtener mediciones antropométricas basándose en un modelo generativo. Se detallan dos soluciones a continuación:

- A. Weiss et al. [19], realiza las mediciones antropométricas generando una función lineal que permite hacer una regresión de parámetros de forma, a mediciones corporales manuales. Esta función es aprendida desde el set de datos CAESAR, del cual se utilizan las medidas corporales y escaneos de cada muestra. Luego, se utiliza esta función para obtener las mediciones corporales en base a los parámetros de forma del modelo 3D. El error medio en las mediciones corporales es de 22[mm]. Cabe señalar que el error aumenta según la imprecisión del modelo 3D generado por la Fase 1.
- S. Yan et al. [8], genera un modelo SMPL según la captura de un escáner 3D. Para realizar las mediciones corporales según este modelo, se definen caminos circulares, *paths*, entre vértices cercanos a la zona de medición corporal del modelo SMPL. Nótese la Figura 2.19, donde se definen tres caminos para: contorno de cintura, cadera y muslo.

Para cada zona, se calculan perímetros sumando las distancias entre los vértices de cada camino.

Luego, se ejecuta una regresión para estimar el valor de la medición corporal,

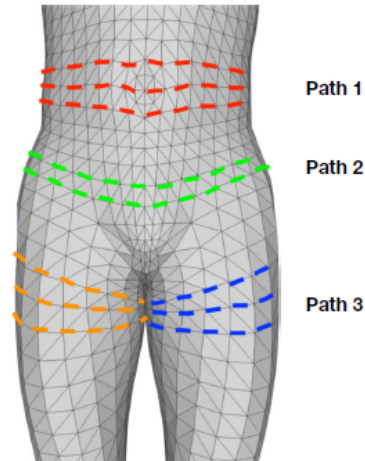


Figura 2.19: Caminos circulares entre vértices de interés. Imagen obtenida del artículo original de S. Yan et al. [8].

utilizando como entrada los perímetros mencionados anteriormente. La estimación puede ser realizada a través de una regresión lineal, pero se reportan mejores resultados utilizando métodos no lineales como *Support Vector Regression (SVR)*. Para realizar la regresión, es necesario tener un set de datos que contenga pares de modelos SMPL y sus correspondientes mediciones, para esto se utiliza el set de datos presentado en el artículo: 'Nomo3D'.

SOLUCIÓN PROPUESTA

ESTE capítulo está dedicado a plantear los detalles del desarrollo e implementación de la solución propuesta. Para esto, se ejecutan pruebas preliminares con un usuario real, las cuales permiten decidir empíricamente el o los método(s) más adecuado(s) para integrarse en el núcleo la solución.

El núcleo de la solución está basado en la inferencia de un modelo corporal a través de redes neuronales artificiales, las cuales están construidas en un entorno de desarrollo basado en el lenguaje Python y la biblioteca Pytorch, mientras que los componentes de Unity están basados en el lenguaje C#. Es por esto que se presentan herramientas de integración al motor de videojuegos Unity 3D, las cuales permiten conectar el núcleo de la solución con los componentes del motor.

Por último, se implementa el método escogido en el motor, y se detallan consideraciones técnicas respecto a la captura automatizada de la imagen, la interfaz del usuario, el modelo corporal generativo, y la estimación de mediciones corporales.

En la Figura 3.1 es posible observar un diagrama de la solución a implementar, detallando como se integran los componentes del motor de videojuegos para realizar la captura automatizada y la posterior renderización, y también, la posición del núcleo de la solución en la implementación general.

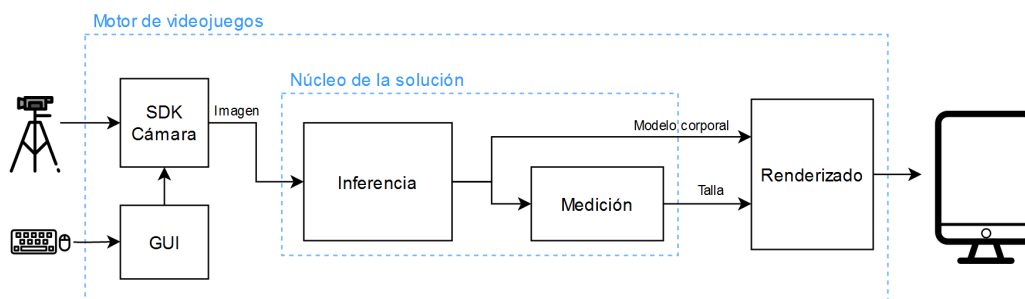


Figura 3.1: Diagrama funcional simplificado de la solución.

3.1. Pruebas preliminares

En la presente sección, se realiza una evaluación preliminar de las alternativas de solución mencionadas en el capítulo anterior, a través de pruebas de los distintos métodos en un ambiente local. Para esto, se utilizan implementaciones disponibles públicamente en un repositorio del autor del método, las que incluyen el código fuente y los pesos de la red neuronal ya entrenada.

Se realizan pruebas utilizando como entrada la captura del cuerpo de un usuario vestido, y como salida un modelo digital tridimensional estimado. Se modifican parámetros de entrada si es que el método lo permite, y se evalúan resultados cualitativos y tiempo de ejecución en cada entorno. Luego, considerando las métricas obtenidas, se definen los métodos que constituyen el núcleo de la solución.

Las primeras pruebas se realizan en un computador portátil, modelo HP 15-3011cs, procesador i5 10th gen, tarjeta de video Nvidia GTX 1050, el cual se menciona como ‘GTX1050’, para efectos de comparación. Para aquellos métodos que se consideren más relevantes, se comparan tiempos en otro equipo, un computador portátil modelo HP 15-dh0005, procesador i7 9th gen, tarjeta de video Nvidia RTX 2070, el cual se menciona como ‘RTX2070’.

Al igual que en la sección anterior, se clasifica según el esquema de la Figura 2.10 y las pruebas se realizan para métodos que cubren las Fases 1 y/o 2 de la solución (Fase 1: Reconstrucción de modelo vestido, Fase 2: Estimación de modelo corporal.). Primero, se realizan pruebas para métodos que procesan a la entrada una nube de puntos del usuario, ya sea: una nube incompleta obtenida por una cámara RGBD, o también, un modelo vestido o nube completa desde un escáner tridimensional. Luego, se realizan pruebas con aquellos métodos que utilizan una imagen RGB.

Imagen RGB-D o modelo vestido como entrada

En este caso se realizan pruebas para cuatro métodos distintos:

- IFNet [3]: Como se menciona en el capítulo 2, el método permite reconstruir un modelo del usuario vestido basándose en una nube de puntos incompleta como entrada. Considerando la implementación del artículo disponible en GitHub¹, se realizan pruebas preliminares para evaluar los resultados del método utilizando una nube de puntos frontal generada por una Kinect V2 (Figura 3.2 (a)). Se realiza una prueba local de reconstrucción utilizando el equipo ‘gtx1050’. Se obtiene un tiempo de ejecución de 5 [min] y 20[seg] y los resultados cualitativos se pueden observar en la Figura 3.2. Si bien el resultado de reconstrucción es consistente, la morfología reconstruida no se corresponde con el usuario.
- NormalGAN [16]: Al igual que el caso anterior, el método permite reconstruir una nube de puntos incompleta, en este caso, en formato RGB-D. Según la implementación disponible por el autor en GitHub², se realizó una prueba utilizando una Kinect V2. Para obtener las imágenes de color y profundidad se utilizó la biblioteca ‘pylibfreenect2’³, la cual permite proyectar los píxeles de color al espacio de profundidad y así obtener dos imágenes de una resolución de 512x424 píxeles. Note ambas imágenes en la Figura 3.3 (a) y (b).

En cuanto a los resultados cualitativos, estos se presentan en la Figura 3.3 (b) y (c),

¹<https://github.com/jchibane/if-net>

²<https://github.com/LizhenWangT/NormalGAN/>

³<http://r9y9.github.io/pylibfreenect2/stable/>

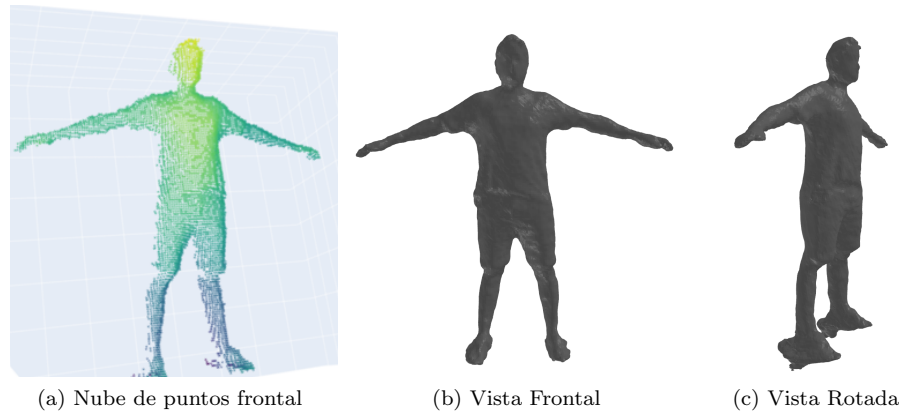


Figura 3.2: Reconstrucción utilizando IF-Net.

donde se puede observar que la reconstrucción no entrega un resultado consistente, ya que el modelo reconstruido no representa un posible cuerpo humano. Según lo consultado al autor, esto puede deberse al posicionamiento de la cámara, ya que la captura es óptima a una distancia de 1.5[m], aproximadamente. La prueba se realiza utilizando el equipo "GTX1050z tarda 3.8[seg] en ejecutarse.

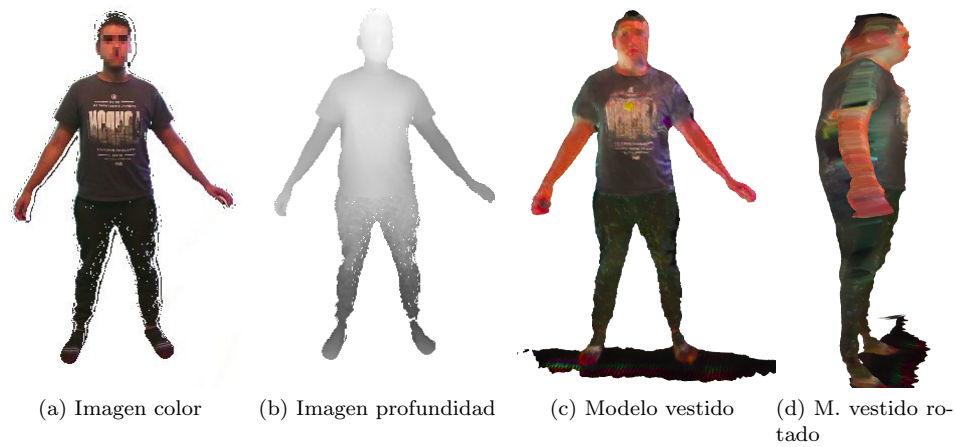


Figura 3.3: Reconstrucción utilizando NormalGAN.

- IPNet: Como se menciona en el capítulo anterior, el método permite reconstruir una nube de puntos del usuario, y además inferir un modelo corporal de éste. Utilizando la implementación del autor disponible en GitHub⁴, se realizan diferentes pruebas. En primera instancia se evalúa utilizando una nube de puntos frontal obtenida desde una Kinect V2 (Figura 3.4(a)). Cabe señalar que la nube de puntos debe ser normalizada para mejorar los resultados en la etapa de optimización. En cuanto

⁴<https://github.com/bharat-b7/IPNet>

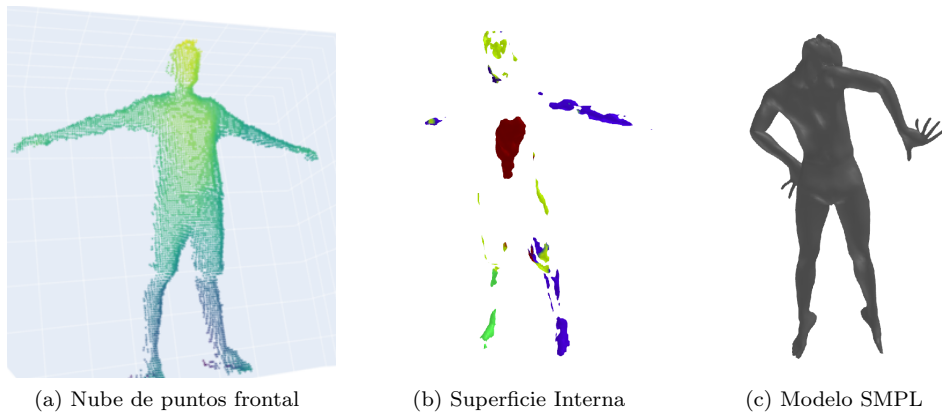


Figura 3.4: Reconstrucción utilizando IP-Net, nube de puntos frontal.

a los resultados, estos se pueden observar en la Figura 3.4(b) y (c), cabe señalar que la superficie interna se encuentra bastante incompleta, mientras que el modelo SMPL se encuentra deformado, ya que intenta ajustarse a una superficie que entrega muy poca información. Luego de consultar en el repositorio del autor, se comprende que la implementación disponible está entrenada para trabajar con nubes de puntos completas, mientras que para el uso de entradas incompletas es necesario hacer un re-entrenamiento de la red en cuestión.

Teniendo esto en consideración, el siguiente paso es realizar pruebas utilizando una nube de puntos completa. Por lo tanto, se considera a priori que la implementación disponible de IP-Net sólo permite ejecutar la Fase 2 (Estimación de modelo corporal desde modelo vestido), según la Figura 2.10.

Para las pruebas con nubes de puntos completas, se descargan dos modelos gratuitos de personas vestidas en formato *.obj, formato compuesto por la posición espacial de los vértices y el ordenamiento de las caras de un modelo. Luego, se normaliza la posición de los vértices con valores máximos y mínimos entre $[-1,1]$ y centrados en $(0,0,0)$. En cuanto a los modelos y los resultados cualitativos, estos se pueden observar en la Figura 3.5, donde se observa que el modelo SMPL estima de forma satisfactoria la forma del cuerpo bajo la ropa del usuario. Cada prueba se realiza utilizando el equipo "GTX1050z tarda 21[min] y 55[seg] en ejecutarse, mientras que utilizando el equipo "RTX2070 cada prueba tarda 4[min] y 26[seg].

- **LoopReg:** Como se menciona en el capítulo anterior, el método permite obtener un modelo corporal de un usuario, basándose en una nube de puntos completa de este. Utilizando la implementación disponible por el autor en GitHub⁵, se realizan pruebas utilizando el equipo "GTX1050z cada ejecución tarda 2[min] y 14[seg]. Los resultados cualitativos pueden observarse en la Figura 3.6, donde se puede observar que se estima de manera consistente la forma humana bajo la ropa, pero el modelo generado es más ancho de lo que debería, si comparamos con el contorno del usuario en la imagen original.

⁵<https://github.com/bharat-b7/LoopReg>

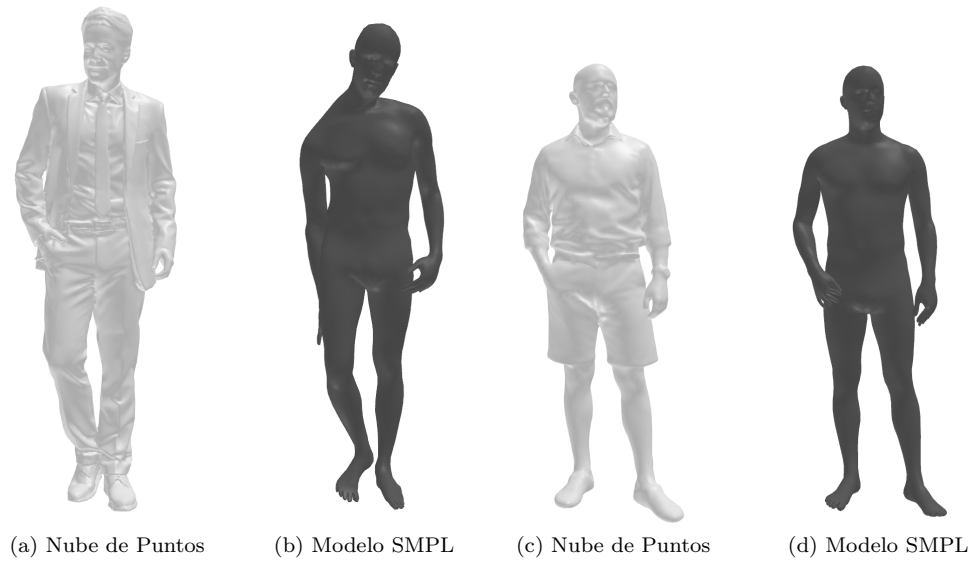


Figura 3.5: Resultados de estimación utilizando IP-Net.

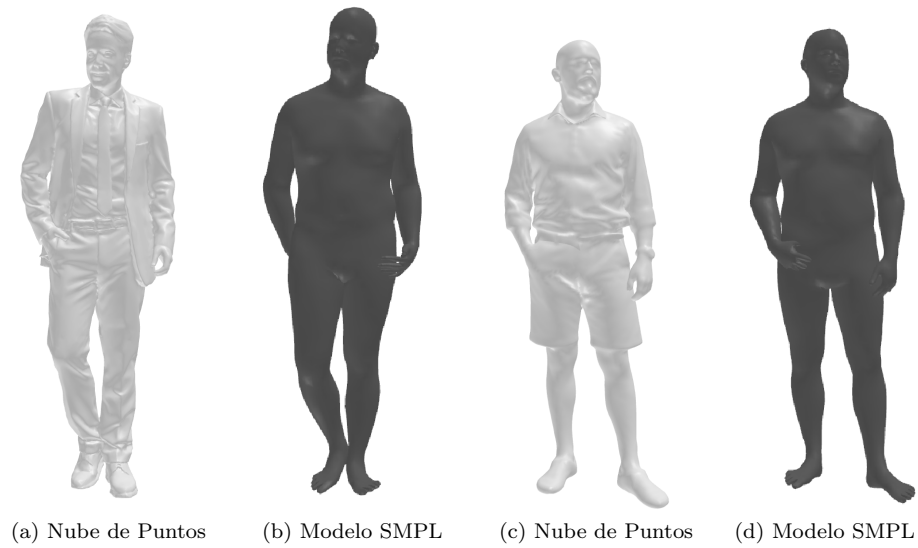


Figura 3.6: Resultados de estimación utilizando LoopReg.

Imagen RGB como entrada

En este caso se realizan pruebas para tres métodos distintos:

- I2LMesh-Net [6]: Como se menciona en el capítulo anterior, el método permite obtener un modelo corporal de un usuario desde una imagen plana. Utilizando

la implementación del artículo disponible por el autor en Github⁶, se realiza una prueba preliminar utilizando una imagen frontal. Los resultados cualitativos se pueden observar en la Figura 3.7(b), donde se observa un resultado con vértices de un modelo SMPL con características femeninas. Ya que el método retorna directamente los vértices del modelo, las características de género del modelo dependen del entrenamiento de la red, por lo cual no es posible modificar el género del modelo SMPL retornado. Se realizan pruebas utilizando el equipo “GTX1050z cada ejecución tarda 5[seg] y 9[ms].

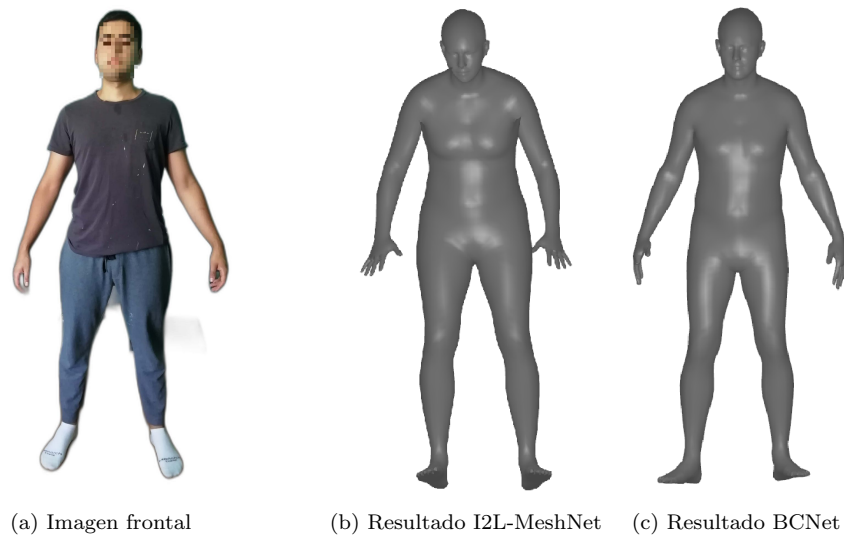


Figura 3.7: Comparativa de estimación de modelo corporal SMPL.

- BCNet [1]: Al igual que el caso anterior, el método permite obtener un modelo corporal de un usuario. Utilizando la implementación del artículo disponible por el autor en Github⁷, se realiza una prueba preliminar utilizando una imagen frontal de un usuario en pose 'A'. Los resultados cualitativos se pueden observar en la Figura 3.7(c), donde se observa un resultado satisfactorio. Además, el método retorna los parámetros de entrada del modelo SMPL, permitiendo así corregir la forma corporal en una etapa posterior. Se realizan pruebas utilizando el equipo 'gtx1050' y cada ejecución tarda 7[seg] y 6[ms], mientras que utilizando el equipo 'rtx 2070' tarda 5[seg] y 5[ms].
- PifuHD [7]: Como se menciona en el capítulo anterior, el método permite reconstruir un modelo vestido del usuario. Utilizando la implementación del artículo disponible por el autor en Github⁸, se realiza una prueba preliminar utilizando una imagen frontal. Los resultados cualitativos se pueden observar en la Figura 3.8, donde se observa una reconstrucción satisfactoria. Se realizan pruebas utilizando el equipo “RTX2070”, y se

⁶https://github.com/mks0601/I2L-MeshNet_RELEASE

⁷<https://github.com/jby1993/BCNet>

⁸<https://github.com/facebookresearch/pifuhd>

obtienen distintos tiempos de ejecución según la resolución de voxels. Para resoluciones de 512, 256 y 128, se obtienen tiempos de 25.3[s], 12.4[s] y 7.3[s] respectivamente.

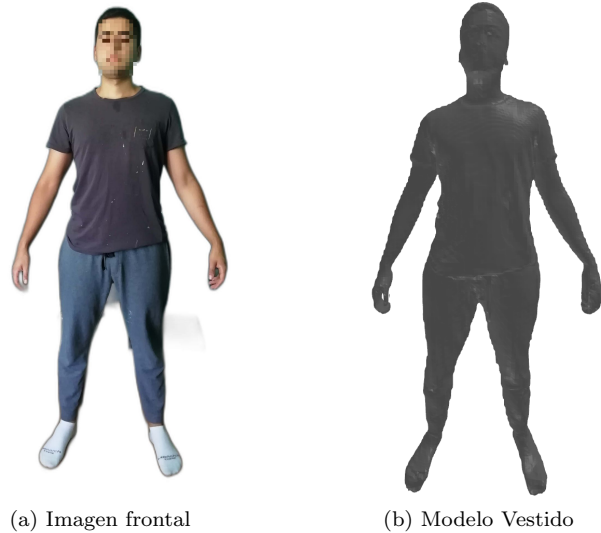


Figura 3.8: Reconstrucción utilizando PifuHD.

Considerando los métodos mencionados anteriormente, se realiza una tabla a modo de resumen de las métricas más importantes. Note la Tabla 3.1, donde se puede observar que los métodos IP-Net, BCNet y PIFuHD presentan las mejores métricas cualitativas, pero en el caso de IP-Net, el tiempo de ejecución es alto.

De acuerdo al estudio de los métodos mencionados y la comparativa desarrollada en el Cuadro 3.1, se confirma la posibilidad de utilizar una cámara RGB como dispositivo de entrada, ya que es posible obtener resultados satisfactorios en este caso. Esto se logra debido a una mayor disponibilidad de set de datos para imágenes RGB, lo que permite realizar mejores estimaciones basadas en aprendizaje. Además, se confirma el método BCNet, como el más adecuado para incluirse en la solución, considerando los resultados cualitativos y de tiempo de ejecución.

Núcleo de la solución propuesto

Con el fin de obtener un mejor resultado basado en las implementaciones disponibles, se plantean dos posibilidades de integración de métodos para implementar el núcleo de la solución, las que se detallan a continuación:

- La primera propuesta de integración, consiste en utilizar los métodos de PIFuHD y IP-Net secuencialmente. Para esto, se ejecuta primero PIFuHD para reconstrucción del modelo vestido en la Fase 1, y posteriormente se ejecuta IP-Net para estimación del modelo corporal bajo la ropa del usuario en la Fase 2.

Esto se puede realizar sin mayores cambios, ya que simplemente se utilizaría la salida de PIFuHD como entrada a IP-Net. La problemática presente es el tiempo de ejecución

Método	Entrada	Fase(s)	Evaluación Cualitativa [1-10]	Tiempo ejecución	
				RTX 1050	GTX 2070
IF-Net	Imagen RGB-D	1	5	5[min] 20[s]	-
NormalGAN	Imagen RGB-D frontal	1	2	3[s] 8[ms]	-
IP-Net	Modelo Vestido	2	8	21[min] 53[s]	4[min] 26[s]
LoopReg	Modelo Vestido	2	7	2[min] 14[s]	-
I2LMesh	Imagen RGB frontal	1 y 2	7	5.9[s]	-
BCNet	Imagen RGB frontal	1 y 2	8	7.6[s]	5.6[s]
PifuHD	Imagen RGB	1	9	-	25.3[s]* 12.4[s]** 8.4[s]***

*= Resolución de 512 voxels, **=Resolución de 256 voxels, ***=Resolución de 128 voxels

Tabla 3.1: Comparativa métodos estudiados.

de ambos métodos, el cual se estima superior a los 20[min]. En el mismo esquema de integración secuencial, otra posibilidad es utilizar PIFuHD y LoopReg secuencialmente. Esto se realiza análogamente al punto anterior. Cabe señalar que LoopReg retorna un modelo SMPL más ancho de lo esperado, lo cual genera un error en las mediciones corporales. Por otro lado, el tiempo se estima menor que el punto anterior, cercano a 2[min]. En la Figura 3.9 se grafica la propuesta de integración para este caso y el anterior.

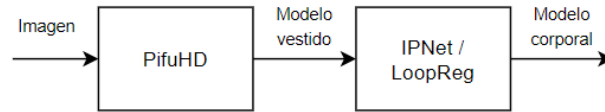


Figura 3.9: Propuesta de integración secuencial.

- En el contexto de la industria textil, es relevante obtener de manera precisa las mediciones de contorno de pecho, cintura y cadera. Los resultados de BCNet a priori no detallan estos contornos, mientras que PIFuHD si logra expresar de manera clara estas geometrías relevantes de un usuario real.

La integración consiste en utilizar los métodos BCNet y PIFuHD en conjunto. Se propone lograr esto, utilizando BCNet para obtener un modelo SMPL preliminar, el cual es mejorado según las predicciones de PIFuHD.

Inicialmente se utiliza la misma imagen de entrada para ejecutar la estimación corporal de BCNet y la reconstrucción vestida de PIFuHD. Luego, se realiza un ajuste lineal del modelo SMPL en ciertos contornos relevantes, según el modelo vestido de PIFuHD. En la Figura 3.10 se grafica la propuesta de integración.

En este caso se estima que es posible mejorar los resultados de las mediciones antropométricas relevantes, manteniendo el tiempo de ejecución menor a 1[min].

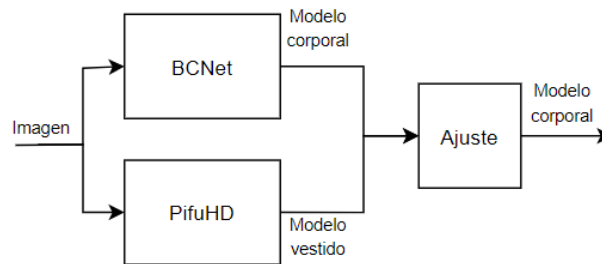


Figura 3.10: Propuesta de integración con ajuste.

Considerando ambas posibilidades, la segunda opción de integración ofrece un mejor resultado de estimación, manteniendo las restricciones de tiempo de ejecución menor a 1 [min]. Sin embargo, el hecho de implementar e integrar diferentes métodos de estimación a la solución, aumenta la complejidad del desarrollo y extiende el tiempo de ejecución del proyecto. Es por esto que se decide posponer la integración de diferentes métodos de estimación para una etapa posterior al trabajo de esta memoria.

En el contexto actual del proyecto, se decide implementar el método BCNet para la inferencia de forma humana en el núcleo de la solución, poniendo énfasis en el correcto pre-procesamiento de la imagen de entrada, y en la escalabilidad de la implementación. De esta forma, se apunta a mejorar el resultado de estimación del método y además considerar la compatibilidad para mejoras futuras en la solución.

3.2. Herramientas de integración

En esta sección, se plantea la arquitectura de integración entre el núcleo de la solución y el motor de videojuegos, detallando la herramienta escogida para esta labor y las funcionalidades implementadas. Para esto, se plantean propuestas de herramientas para la integración, luego, se escoge la herramienta más adecuada para el proyecto en cuestión, y por último, se detalla la arquitectura de la integración a implementar con la herramienta escogida.

Como se menciona al inicio del capítulo, el motor de videojuegos Unity, está diseñado para codificar componentes utilizando el lenguaje C# y el framework .NET, mientras que el núcleo de la solución está pensado para ser implementado utilizando el lenguaje Python y la biblioteca Pytorch. Para integrar estos dos entornos, una posibilidad es disponer de un entorno de Python dentro del entorno de Unity, a través de un intérprete embebido en el motor, mientras que, otra posibilidad es disponer de un entorno de Python externo al del motor, y establecer un protocolo de comunicación entre ambos entornos.

Python desde el entorno de Unity

El utilizar un intérprete de Python embebido en el motor, facilita manejo de errores y excepciones desde el motor, y además, permite generar un archivo ejecutable que encapsule también el entorno de Python. Además, hace que la ejecución del intérprete sea más lenta, ya que requiere virtualizar el entorno de Python dentro del motor. En este contexto, existen implementaciones de Python que se integran con el framework .NET: PythonNet¹ y

¹<https://github.com/pythonnet/pythonnet>

IronPython². Estas herramientas están pensadas para acceder a las funcionalidades de .NET desde un entorno de Python y además acceder a las funcionalidades de Python desde un entorno C# .NET. Para utilizar Python desde C# según lo provisto por estas herramientas, es necesario reescribir el código de Python en un formato específico. Además, en ambos casos, la disponibilidad de bibliotecas es limitada. Por lo cual, se descarta la utilización de alguna de estas herramientas en la integración.

Otra alternativa para utilizar Python y sus dependencias en el motor de videojuegos, es utilizar ciertos paquetes provistos por Unity: Python for Unity y Unity Barracuda, los cuales se detallan a continuación.

Python for Unity, es un paquete que permite la ejecución de Python en el motor de videojuegos a través de su Python API. La API permite ejecutar scripts de Python como un proceso desde C#. La API dispone de dos variaciones, *High Level API* y *Low Level API*. Para este proyecto es relevante la primera variación. La *High Level API* o API de alto nivel, permite ejecutar como un proceso líneas de código o archivos escritos en Python, utilizando métodos provistos para C#. La integración se basa en importar clases del motor de Unity desde Python, lo que permite interactuar con componentes y objetos que sean parte del entorno gráfico. El paquete dispone de una versión estable compatible con Python 2.7 y también una versión experimental compatible con Python 3.6.

Se realizan pruebas para instalar el paquete experimental para Python 3.6 en el motor de videojuegos, y luego de intentos en distintas versiones de Unity, no se logra instalar el paquete correctamente, ya que este no logra linkear el intérprete de Python en su ejecución inicial.

Unity Barracuda, es un paquete que permite la inferencia de redes neuronales en formato ONNX ó *Open Neural Network Exchange*. Este es un formato de código abierto para definir modelos de aprendizaje de máquinas, que permite portar modelos entrenados de redes neuronales de distintas plataformas, como los generados por Pytorch, en un archivo de extensión *.onnx, el cual puede ser importado y utilizado para hacer inferencia. Barracuda facilita la ejecución de archivos *.onnx desde Unity, utilizando pocas líneas de código C#. Se realizan pruebas del paquete, y el paquete funciona correctamente en el motor, logra cargar modelos y posteriormente hacer inferencia con ellos. Luego, se realiza una prueba con un modelo *.onnx que encapsula el método BCNet mencionado en la sección anterior, y en este caso, el paquete no logra cargar correctamente el modelo, ya que el código BCNet no se encuentra escrito en el formato correcto y utiliza operadores no soportados por Barracuda.

Considerando incompatibilidad de IronPython y Pythonnet, y además, los errores obtenidos al realizar pruebas iniciales con los paquetes ofrecidos por Unity, se descarta la opción de utilizar un entorno de Python embebido en el motor de videojuegos y se decide optar por un entorno externo comunicado a través de un protocolo de red local.

Python externo al entorno de Unity

Utilizar un intérprete de Python externo al entorno de Unity, permite utilizar una versión nativa de Python en el sistema operativo, y así portar todas las bibliotecas y dependencias necesarias para el funcionamiento de la implementación. En este caso la integración consiste en establecer un protocolo de comunicación entre el entorno de Python y el de Unity. Considerando esto, es posible establecer comunicación a través de una red local en la que la implementación en Python cumpla el rol de servidor, y la implementación Unity el de cliente.

²<https://ironpython.net/>

Para implementar el protocolo de comunicación, es posible utilizar la biblioteca ZeroMQ³, biblioteca de código abierto, diseñada para establecer comunicación asincrónica de manera rápida y minimalista entre múltiples plataformas. ZeroMQ soporta distintos patrones de mensajería: *requester/responder*, *publish/subscribe*, entre otros.

Se plantea utilizar la biblioteca ZeroMQ para Python y la biblioteca NetMQ para ser utilizada desde C#. Esta última, es una implementación nativa en C# de la biblioteca ZeroMQ. Para validar la conexión basada en ZeroMQ, se realiza una prueba de comunicación entre una aplicación básica de cada entorno, utilizando el patrón de requerimiento/respuesta. A grandes rasgos, la prueba consiste en un objeto de Unity que envía un requerimiento con el texto 'Hello' y luego recibe una respuesta desde Python con el texto 'World'. El resultado es satisfactorio, la comunicación se ejecuta de manera rápida y confiable, por lo tanto, se confirma utilizar un intérprete de Python externo al entorno de Unity como herramienta de integración

Arquitectura de integración

Teniendo esto en consideración, se define la arquitectura de la integración con un entorno de Python independiente de Unity. Se decide dividir el núcleo de la solución en distintos módulos funcionales y utilizar un patrón de requerimiento/respuesta para cada módulo de Python. Luego, cada uno de estos módulos, se inicia según un requerimiento desde el motor, y se envía la respuesta cuando el módulo finaliza su ejecución. Así es posible obtener resultados intermedios y un mayor control desde el motor de videojuegos.

En la Figura 3.11, se puede observar un diagrama que representa la arquitectura de la integración entre ambos entornos. En el entorno de Python se observan tres etapas funcionales a ser iniciadas por el servidor según requerimientos desde el cliente: Pre-procesamiento, Inferencia y Medición. El entorno de Unity se encarga de interactuar con el usuario a través del hardware, e iniciar los requerimientos al servidor de Python. Luego, el servidor de Python inicia los módulos requeridos, para luego enviar como respuesta el resultado del procesamiento. Por último, el entorno de Unity renderiza los resultados del procesamiento.

3.3. Implementación

En esta sección, se plantean los detalles de la implementación de la solución. En términos generales, la solución se compone de un programa ejecutable en un equipo con el sistema operativo Windows 10, que permite: realizar la captura automatizada del cuerpo de un usuario vestido, ejecutar la estimación de su modelo corporal y medidas, y por último, renderizar en pantalla los resultados de estimación. La captura automatizada y la renderización son realizadas en un entorno del motor de juegos Unity, mientras que la estimación del modelo corporal y sus mediciones se realiza en un entorno de Python.

El proceso de estimación de modelo corporal y mediciones se compone de múltiples módulos funcionales, los cuales permiten ejecutar las etapas de pre-procesamiento, inferencia y medición. La utilización de módulos funcionales, permite que la solución sea escalable, ya que éstos permiten añadir, mejorar o remover una funcionalidad específica según sea necesario, sin alterar la arquitectura general de la solución.

Cada módulo funcional está pensado para ser requerido en una solicitud por la implementación en Unity o cliente, es decir, el cliente, es el encargado de entablar la comunicación con la implementación en Python, o servidor, y requerir la ejecución de

³<https://zeromq.org/>

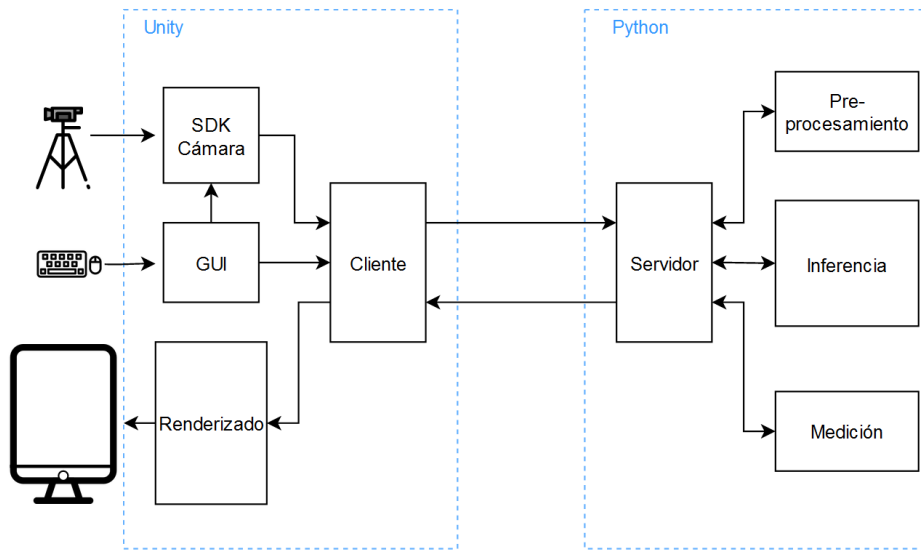


Figura 3.11: Arquitectura de la integración entre las implementaciones en los entornos de Unity y Python.

cada módulo por separado. En otras palabras, se establece una comunicación del tipo *requester/responder* con cada módulo, donde Unity envía los datos a procesar como requerimiento y Python responde con el resultado de la ejecución del módulo.

Durante la sección, primero se describen cada uno de los módulos funcionales administrados por el servidor en la implementación del núcleo de la solución en sus distintas variaciones. Posteriormente, se detalla la implementación en el entorno de Unity, explicando de qué manera los módulos son requeridos por el cliente según la interacción de los objetos gráficos en la aplicación.

3.3.1. Implementación en el entorno de Python

La implementación en Python se compone de diferentes scripts o archivos *.py. El script principal es aquel que implementa el servidor. Éste se encarga de iniciar una conexión local basada en ZeroMQ y mantenerse en estado de espera por algún requerimiento desde el cliente. Al recibir un requerimiento, el servidor identifica el tipo de requerimiento e inicia la ejecución del módulo asociado. Posterior a la ejecución, responde al cliente con los resultados obtenidos y vuelve al estado de espera.

Cada módulo es implementado en un archivo distinto. En cada archivo se declara la funcionalidad del módulo encapsulada en una función. Esto permite importar la función asociada a cada módulo en el servidor y llamar a su ejecución cuando sea necesario.

Para facilitar el manejo de imágenes y mallas tridimensionales, se utilizan las bibliotecas *Pillow*¹ y *Trimesh*². *Pillow*, también llamada PIL(*Python Image Library*), permite cargar, procesar y guardar imágenes a través de métodos de la clase 'Image', mientras que *Trimesh* permite realizar un análogo para mallas tridimensionales o archivos *.obj a través de méto-

¹<https://pillow.readthedocs.io/en/stable/>

²<https://github.com/mikedh/trimesh>

dos de la clase 'trimesh'.

La solución implementada está sujeta a ciertos requerimientos de la empresa Nexar. Uno de éstos, es la utilización de un modelo corporal que incluya la pose de las manos. Los métodos de estimación de forma humana mencionados en la Sección 2.2.1, y también aquellos evaluados en la Sección 3.1, consideran retornar la forma humana basada en un modelo SMPL, el cual no permite modificar la pose de las manos. Para obtener una forma humana con las características requeridas, es necesario obtener un modelo SMPL-X a la salida, ya que éste permite la animación de las manos y la expresión facial. Teniendo esto en consideración, se extiende la solución pensada para SMPL, para que también permita retornar un modelo SMPL-X. A continuación, primero se describen los módulos funcionales de la aplicación considerando obtener un modelo SMPL como resultado, y luego, las modificaciones y nuevos módulos añadidos a la aplicación, que permiten obtener un modelo SMPL-X a la salida.

Modelo SMPL

Como se menciona anteriormente, el núcleo de la solución se clasifica en tres etapas principales: pre-procesamiento, inferencia y medición. En el pre-procesamiento se modifica la imagen de entrada para que se encuentre en el formato más adecuado para la etapa de inferencia. En la etapa de inferencia se ejecuta la estimación basada en redes neuronales artificiales que mapean la imagen de entrada a los parámetros de entrada de un modelo SMPL. En la etapa de medición, se utilizan los vértices que componen la malla del modelo corporal, y un parámetro de escala, para obtener las medidas de contornos relevantes en el cuerpo. En la Figura 3.12, se describe la ejecución secuencial de los módulos que componen la aplicación considerando un modelo SMPL a la salida, y su correspondencia con las etapas mencionadas.

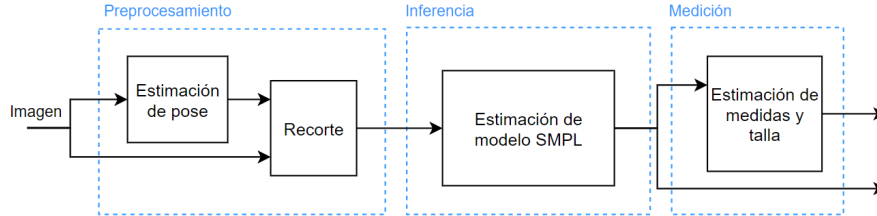


Figura 3.12: Módulos funcionales considerando un modelo SMPL a la salida.

A continuación, se explica la funcionalidad implementada para cada módulo:

- **Estimación de pose:** El presente módulo, permite estimar la pose 2D del usuario, es decir, obtener la posición de puntos clave o keypoints de su cuerpo en la imagen. Se considera parte del pre-procesamiento, ya que la pose es utilizada como entrada para el siguiente módulo, que es donde se retorna la imagen pre-procesada. Aún así, se define como un módulo independiente, ya que disponer de la pose del usuario es útil para desarrollos futuros.

Para implementar la estimación de pose, se escoge el método OpenPose [20], método basado en una representación no paramétrica, *Part Affinity Fields*, para aprender las partes del cuerpo de personas en una imagen. El método es ampliamente utilizado en visión por computador y permite estimar la pose del cuerpo, manos, pies y rostro de

manera precisa y en tiempo real. Para implementar el módulo asociado, por simplicidad se escoge utilizar la versión ejecutable para Windows³, seleccionando la opción para retornar la pose de la imagen en un archivo de formato *.json. En cuanto al módulo implementado para Python, éste se compone de una función que recibe una imagen del tipo PIL(Pillow) Image, luego, inicia el ejecutable de OpenPose como un subproceso de Python, y por último, retorna una variable con el resultado de estimación de pose en formato de texto json. En la Figura 3.13(a), se muestra un ejemplo de los keypoints estimados en un usuario.

- **Recorte:** Este módulo se encarga de recortar o extender la imagen según la pose bidimensional del usuario, es decir, utilizando la posición de las articulaciones del usuario, se modifica la imagen para dejar el cuerpo del usuario en el centro de ésta. El módulo además modifica la resolución de la imagen, retornando una imagen cuadrada de 540x540 píxeles. La implementación del presente módulo se basa en el procesamiento inicial del método PifuHD⁴, mencionado en la Sección 3.1. En la Figura 3.13(b), se ejemplifica el funcionamiento del módulo.

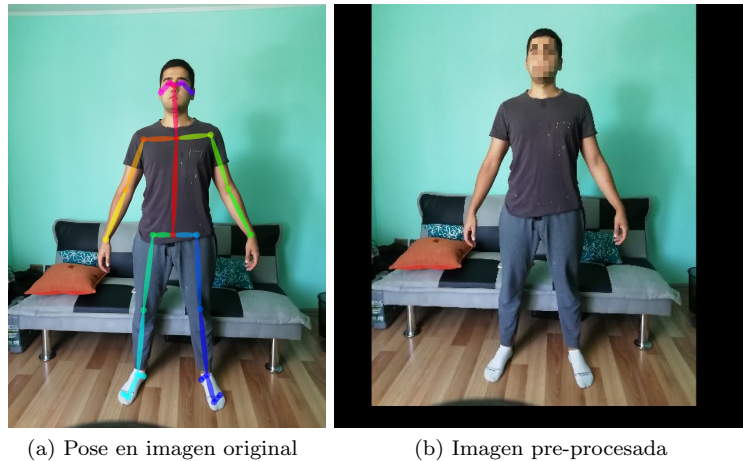


Figura 3.13: Ejemplo de etapa de pre-procesamiento.

- **Estimación de modelo SMPL:** Como se menciona en la Sección 3.1, se escoge el método propuesto por B. Jiang et al. [15], BCNet, para implementar la estimación del modelo corporal SMPL. El método permite obtener a la salida los parámetros de un modelo SMPL de género neutral, además de los parámetros de modelos generativos de la ropa del usuario posicionados sobre el modelo SMPL, considerando dos modelos, uno para la ropa superior y otra para la inferior. En el contexto de este proyecto, sólo se utilizarán los resultados del modelo SMPL.

Al igual que en las pruebas preliminares, se utiliza la implementación disponible en Github, la cual contiene la red neuronal y los pesos de un modelo pre-entrenado. La implementación está pensada para ser ejecutada desde la terminal, obtener imágenes desde archivos y guardar los resultados de estimación en distintos archivos *.obj, por lo

³https://github.com/CMU-Perceptual-Computing-Lab/openpose/blob/master/doc/01_demo.md

⁴<https://github.com/facebookresearch/pifuhd/blob/main/lib/data/EvalWPoseDataset.py>

cual se hace necesario modificar el proyecto para adaptarse a la arquitectura modular. Entre las modificaciones al proyecto, se retira etapa de recorte implementada en el método, ya que en la etapa de pre-procesamiento ya se adapta la resolución de la imagen según se requiere. Además, se encapsula la implementación en una función que recibe una imagen del tipo *PIL Image*, y retorna un diccionario con ciertos parámetros del modelo SMPL estimado: posición de vértices, caras, blendshapes de forma y matrices de rotación de pose.

- **Estimación de medidas y talla:** Considerando el método propuesto por S. Yan et al. [8], el cual establece mediciones basadas en caminos en zonas de medición, mencionado en la Sección 2.2.2, es posible implementar el módulo como una variación de éste. El método utiliza como zonas de medición los contornos de: Cuello, Base del Cuello, Pecho, Bíceps, Cintura, Cintura del pantalón, Cadera, Muñeca, Muslo, Pantorrilla y Tobillo. Además, considera mediciones de largo de: Hombro a Hombro y Hombro a Muñeca.

Se definen entre 4 y 9 caminos por zona de medición, donde cada camino corresponde a un arreglo de vértices del modelo SMPL. Se propone realizar la medición directamente desde un ajuste geométrico de los caminos del modelo SMPL, utilizando las zonas de medición mencionadas y múltiples caminos por zona. Primero, se obtienen puntos co-planares para cada camino. Luego, se ajusta una curva cerrada que pase por estos puntos, utilizada para realizar la medición del perímetro. Nótese la Figura 3.14, donde se pueden observar los puntos de un camino circular(verde), los puntos co-planares al camino(azul) y la curva cerrada (rojo). Además, se realiza un promedio entre los perímetros de cada camino obtenidos en la zona. Finalmente, es necesario escalar el valor del perímetro promedio para que se corresponda con el valor real de la persona. Para esto se puede utilizar como referencia la altura del usuario. De esta manera, es posible obtener un valor de medición para cada zona.

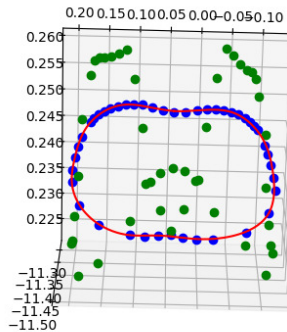


Figura 3.14: Ajuste geométrico de un camino de la zona del pecho.

Por otro lado, el módulo también considera estimar la talla del usuario según las mediciones obtenidas. Para esto es necesario disponer de una tabla de tallas, herramienta que permite relacionar una medición corporal con una talla según un rango definido. Ya que las tablas de tallas varían según las marcas de ropa, se definen los rangos de la tabla de talla en un archivo en formato *.csv, para luego importar

sus valores a Python. Se implementa la estimación de talla para tres mediciones de contorno: Pecho, Cintura y Cadera. Para esto, se realiza una comparación entre el valor de la medición y el rango definido para cada talla según la tabla. Obteniendo así la talla del usuario para cada contorno, por ejemplo, Pecho: S, Cintura: M, Cadera: M.

En cuanto a la implementación del script correspondiente al módulo, éste considera una función que recibe los vértices del modelo SMPL y la estatura del usuario, y retorna un diccionario que asocia los nombres de los contornos con sus mediciones y talla.

Modelo SMPL-X

Al igual que en el caso anterior, en la Figura 3.15, se describe la ejecución secuencial de los módulos que componen la aplicación considerando un modelo SMPL-X como resultado, y su correspondencia con las etapas mencionadas. La etapa de pre-procesamiento se conserva igual que el caso anterior, mientras que las etapas de inferencia y medición requieren modificaciones.

Se añade el módulo de estimación de género como una mejora a la solución, ya que la conversión de SMPL a SMPL-X permite seleccionar el género del modelo resultante. Así, es posible obtener un modelo con una morfología más adecuada al género del usuario.

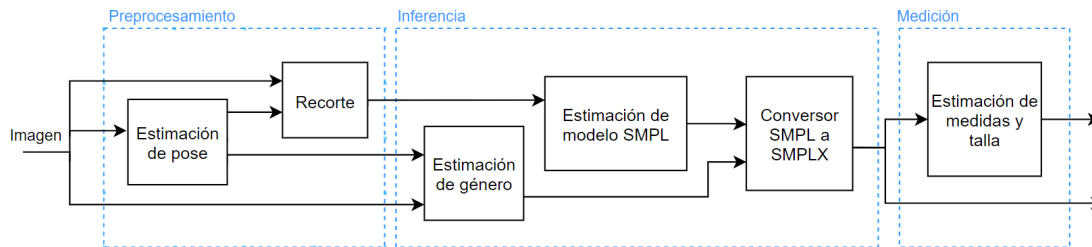


Figura 3.15: Módulos funcionales considerando un modelo SMPL-X a la salida.

A continuación, se explican funcionalidades implementadas en los nuevos módulos y también modificaciones en los módulos anteriores:

- **Estimación de género:** El presente módulo permite estimar el género del usuario (femenino o masculino), basándose en la inferencia desde una imagen de cuerpo completo y los keypoints generados por OpenPose. El método se basa en la implementación del proyecto Homogenus⁵, proyecto desarrollado como parte del método Smplify-x [14].

Para implementar el módulo, se define un umbral de probabilidad sobre el cual se asume que el género es correcto. Luego, en caso de no satisfacer el umbral, se asume un género neutral. En otras palabras, para probabilidades mayores al umbral, el género es femenino o masculino según corresponda, mientras que para valores menores al umbral, el género es neutral. Para el módulo implementado se utiliza un umbral de 85 %.

⁵<https://github.com/nghorbani/homogenus>

Al igual que los casos anteriores, se define una función para encapsular el módulo, la cual recibe como entrada una imagen del tipo *PIL Image* y la pose en un formato de texto json, y retorna como salida el género estimado y la probabilidad asociada de estimación.

- **Conversor SMPL a SMPL-X:** Si bien SMPL-X se considera una extensión de SMPL que incluye la pose de las manos y la expresión facial, los parámetros de forma y pose son distintos para cada modelo, ya que ambos están contruidos según procesos de entrenamiento diferentes. Por lo cual, si se busca obtener un modelo SMPL-X con las mismas características de un modelo SMPL, es necesario calcular sus parámetros.

Para obtener los parámetros en este caso, los desarrolladores de SMPL-X presentan un proyecto llamado *transfer model*⁶ que permite realizar la conversión entre los parámetros de distintos modelos generativos (SMPL, SMPL-H y SMPL-X). El proyecto recibe como entrada la posición de los vértices del modelo de referencia y retorna los parámetros de entrada del modelo esperado, permitiendo además escoger el género del modelo esperado. La conversión se ejecuta a través de la optimización de los parámetros del modelo esperado, buscando minimizar la distancia entre los vértices de ambos modelos. El proceso de optimización de basa en múltiples iteraciones utilizando el método *trust-ncg* (*Newton conjugate gradient trust-region*).

Al igual que los casos anteriores, se define una función para encapsular la implementación del proyecto en un módulo, la cual recibe como entrada los vértices del modelo SMPL de referencia y el género del modelo esperado, y retorna como salida los parámetros del modelo SMPL-X.

- **Estimación de medidas y talla:** La implementación del presente módulo es similar a la del caso anterior. En este caso, se añade una modificación para que el módulo pueda también estimar las mediciones y talla de un modelo SMPL-X.

La distribución de los vértices de SMPL-X es diferente a la de SMPL, por lo cual, es necesario obtener la posición de los vértices correspondientes a los caminos en las zonas de medición, para un modelo SMPL-X. Para definir la correspondencia entre los vértices de ambos modelos, se mapea cada vértice de SMPL a su par más cercano en SMPL-X, es decir, se busca para cada vértice de SMPL, el vértice más cercano en SMPL-X. Luego, se define un arreglo de mapeo, en el cual la posición corresponde al vértice en SMPL-X y el valor corresponde al vértice en SMPL. De esta manera, es posible utilizar vértices de SMPL-X, para hacer el cálculo de mediciones según SMPL. Por lo tanto, la modificación al módulo, consiste en importar el arreglo de mapeo cuando se realicen mediciones para SMPL-X. De esta manera, es posible pasar los vértices de SMPL-X a SMPL y ejecutar la medición al igual que el caso anterior.

3.3.2. Implementación en entorno de Unity

Al finalizar la Sección 3.1, se logra decantar por un método de estimación de modelo corporal basado en una imagen RGB como entrada, esto debería permitir prescindir de una cámara RGBD para la implementación de la solución. Pero, se decide utilizar como dispositivo de captura la cámara RGBD Azure Kinect, debido a múltiples razones asociadas al contexto del proyecto.

⁶https://github.com/vchoutas/smplx/tree/master/transfer_model

La aplicación se basa en una escena con diferentes objetos gráficos o *Game Objects*. Respecto al renderizado o lo observable visualmente, es posible mencionar tres objetos principales: el lienzo de texto, el maniquí y el vídeo de fondo. El lienzo de texto o *canvas*, se posiciona delante del maniquí y se puede asociar a una película transparente, donde se posiciona el texto con los resultados de estimación. El maniquí, se posiciona en el centro de la escena, y representa el modelo corporal del tipo SMPL-X. El vídeo de fondo, es una estructura rectangular plana, posicionada detrás del maniquí, que muestra la captura de vídeo generada por la cámara.

En cuanto al maniquí, para disponer del maniquí como un objeto dentro de Unity, es necesario utilizar un modelo en formato *.fbx que contenga el modelo animable. El formato fbx, es un formato de archivos propietario de Autodesk, que permite la interoperabilidad entre aplicaciones de creación de contenido digital. En Unity, permite cargar diferentes modelos de animación digital, modificar su forma a través de blendshapes, y su pose según sus articulaciones. Para la presente implementación, se utilizan simultáneamente las tres versiones disponibles en formato fbx del modelo SMPL-X: femenino, masculino y neutral.

A un nivel de experiencia del usuario, la aplicación en primera instancia muestra el texto inicial, incluyendo sólo los títulos de los valores a obtener, e inicia la captura de vídeo a través del sensor Kinect. En la Figura 3.18(a), es posible observar el estado inicial de la escena en la aplicación. Luego, la ejecución de la estimación se inicia cuando el usuario se posiciona frente a la cámara en pose 'A' y es presionada la tecla espacio. Al finalizar la ejecución de la estimación de modelo corporal y talla, es posible observar en pantalla un modelo SMPL-X con una forma corporal adaptada al usuario y texto indicando los resultados de su género, medidas y talla. En la Figura 3.18(b), es posible observar un ejemplo del estado de la aplicación posterior a la estimación.

Según lo antes mencionado, es posible definir tres estados funcionales en la aplicación: Espera, Ejecución y Finalizado. En la Figura 3.16, se observa la relación entre los estados y los eventos que gatillan su modificación.

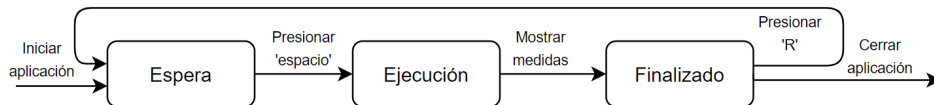


Figura 3.16: Diagrama de estados de la aplicación.

A continuación se caracteriza cada estado, detallando las funcionalidades implementadas y eventos asociados para cada caso:

- **Espera:** Se establece al iniciar la aplicación o al presionar la tecla 'R' en el estado Finalizado. En este estado, se inicia la captura de vídeo de la cámara y se inicia el servidor de Python a la espera de un requerimiento. Además, se muestra en pantalla la captura RGB de la cámara y un texto con los títulos de los resultados a obtener durante la ejecución.

En paralelo se ejecuta la estimación de estatura, donde se utilizan los puntos superiores e inferiores de la nube de puntos del cuerpo para obtener el valor de estatura. El método va mejorando la estimación de estatura a medida que recibe distintas capturas del cuerpo del usuario, y converge a un valor fijo luego de unos segundos. Además, se muestra en pantalla un texto con el valor de altura estimado.

- **Ejecución:** Es el estado donde se inicia la ejecución del núcleo de la solución y se establece al presionar la tecla espacio desde el estado de Espera. En la Figura 3.17, se puede observar un diagrama de secuencia que detalla los eventos que suceden de manera interna durante el estado.

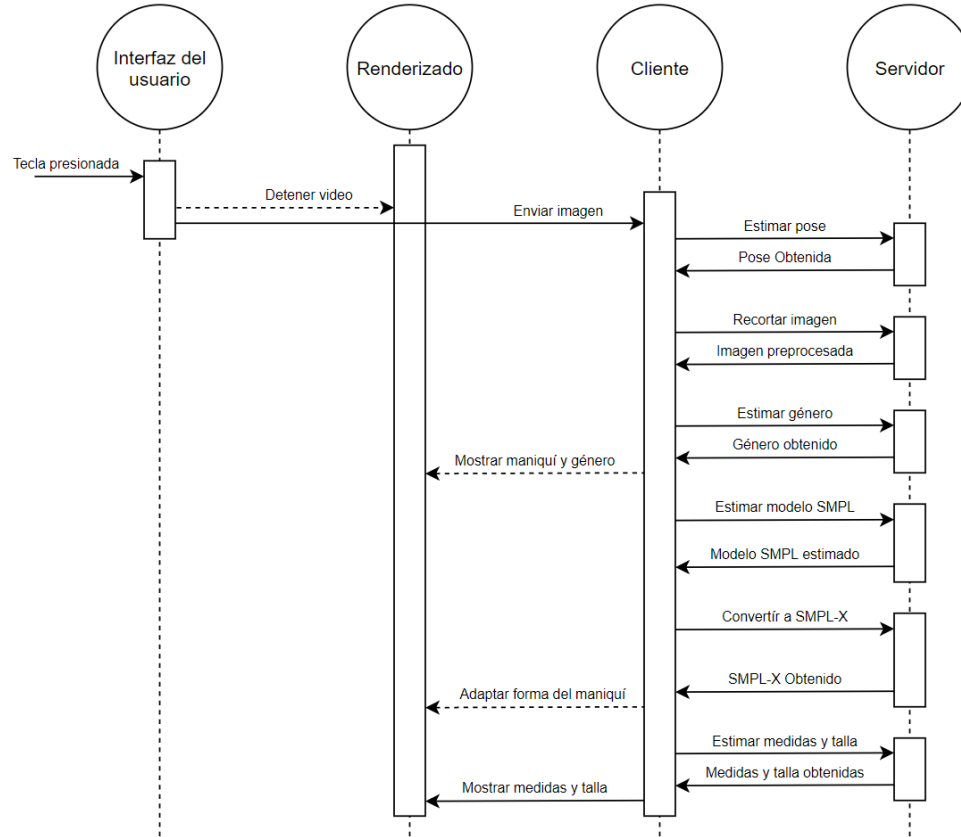
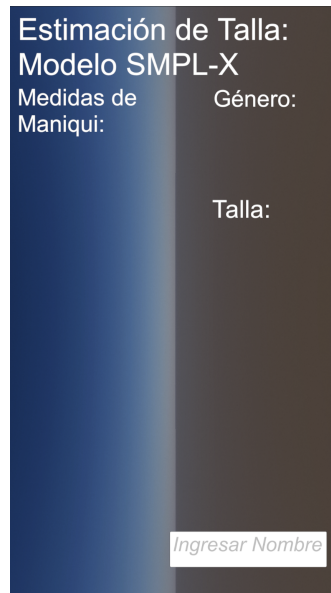


Figura 3.17: Diagrama de secuencia del estado de ejecución.

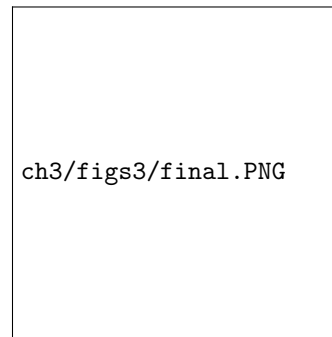
El primer evento se gatilla al presionar la tecla espacio. En ese momento se pasa a memoria la imagen y estatura actuales, se detiene la captura de vídeo, y se da inicio los requerimientos del cliente, quien envía la imagen del usuario como requerimiento para iniciar la etapa de pre-procesamiento. El orden de los requerimientos obedece al diagrama de módulos funcionales de la Figura 3.15. Teniendo esto en consideración, se ejecutan secuencialmente los requerimientos asociados a los módulos de estimación de pose, recorte y estimación de género. Luego, al recibir la respuesta del género estimado, se muestra en pantalla el maniquí asociado al género escogido y el texto con el género estimado. Posteriormente, se inician los requerimientos asociados a la estimación del modelo corporal y conversión a modelo SMPL-X. A continuación, al recibir la respuesta con los parámetros del modelo SMPL-X, se modifican los blendshapes del maniquí, adaptando así su forma corporal. Por último, se inicia el requerimiento asociado a la estimación de medidas y tallas del modelo obtenido. Luego, al recibir la respuesta con las medidas estimadas y su talla, se muestra en pantalla el texto con cada uno de los

valores estimados.

- **Finalizado:** Se establece al finalizar la ejecución del último módulo del núcleo de la solución, es decir, cuando se reciben los valores estimados de medidas corporales y talla. Se caracteriza por mostrar en pantalla los resultados del modelo corporal estimado y sus mediciones. En este estado, es posible volver al estado de espera presionando la tecla 'R' o bien, cerrar la aplicación.



(a) Estado de espera en la escena



(b) Estado finalizado en la escena

Figura 3.18: Modificaciones en la escena al finalizar la estimación.

PRUEBAS Y RESULTADOS

EL presente capítulo está dedicado a detallar las pruebas realizadas en usuarios con el objetivo de validar el desempeño de la implementación de la solución, según métricas que permitan responder a la hipótesis y las preguntas de trabajo planteadas en la Sección 1.3. Para esto, se realizan pruebas de estimación sobre una muestra heterogénea de usuarios, y luego, se cuantifican métricas asociadas a la calidad de la estimación del modelo corporal, sus mediciones y su talla.

Durante el capítulo, primero se plantea la estructura prevista para la realización de las pruebas. Se detallan las características de la aplicación utilizada para ejecutar las pruebas, la que se compone de la implementación de la solución, modificada con el fin de facilitar el almacenamiento de resultados en cada prueba. Además, se caracteriza según género y rango etario la muestra escogida de usuarios partícipes de las pruebas.

Luego, se ejecutan las pruebas descritas para cada usuario, y se presentan los resultados obtenidos para cada caso. Considerando estos resultados, se evalúan cualitativamente los modelos corporales estimados, y también, se validan cuantitativamente los valores de tiempo de ejecución y mediciones corporales obtenidos. Para esto, se calculan métricas de error que evalúan el desempeño de las mediciones corporales, comparadas con una referencia real. Por último, se analizan las métricas y resultados obtenidos en cada caso, entregando una validación de las preguntas de trabajo presentadas en la Sección 1.3.

4.1. Estructura de pruebas

Con el objetivo de facilitar la captura de resultados, se modifica la implementación de la solución añadiendo funcionalidades que permitan la correcta realización de las pruebas en sus distintas variaciones.

La solución está pensada para estimar un modelo corporal y su talla según un modelo SMPL-X, ejecutando los módulos funcionales de la Figura 3.15. Para efectos de las pruebas, también es relevante considerar la arquitectura anterior, es decir, estimar según un modelo SMPL, ejecutando los módulos de la Figura 3.12. Considerando esto, se añade la opción de escoger en qué modelo corporal basar la estimación, ya sea SMPL o SMPL-X.

La siguiente funcionalidad añadida, consiste en la opción de guardar como un archivo el resultado de cada módulo ejecutado por el núcleo de la solución. También, se añade un

texto de entrada en pantalla, que permite ingresar nombre del usuario. Cada uno de los archivos se nombra según la hora de inicio de la estimación, el nombre ingresado y el tipo de resultado. De esta manera, en cada estimación ejecutada, es posible guardar los siguientes elementos según corresponda: imagen de entrada(.png), pose del usuario(.json), imagen recortada(.png), género(.txt), modelo corporal(.obj), y valores de medidas y talla(.csv).

La solución está pensada para estimar según la imagen y la estatura obtenida en tiempo real desde la cámara Azure Kinect. Considerando esto, se añade la opción de realizar la estimación utilizando una imagen obtenida desde un archivo e ingresar la estatura del usuario y la dirección de la imagen de manera manual. De esta manera, es posible prescindir del dispositivo de captura y del usuario para la realización de pruebas.

Por otro lado, se plantea realizar pruebas en un grupo de personas heterogéneas en edad y sexo. Se definen cinco rangos etarios: 16 a 20 años, 21 a 30 años, 31 a 40 años, 41 a 60 años, y mayores de 60 años. Para cada rango, se buscan cuatro personas de diferente sexo.

Considerando las funcionalidades añadidas, se plantea la estructura de las pruebas a realizarse para cada usuario. Primero, al usuario se le toman sus mediciones corporales de manera manual, utilizando cinta métrica. Después, se inicia la aplicación escogiendo guardar los resultados. Luego, se posiciona al usuario en pose 'A' frente a la cámara Azure Kinect y se ejecuta la estimación para un modelo SMPL. Finalmente, se realiza la estimación análoga para un modelo SMPL-X.

4.2. Resultados

Se realizan las pruebas según la estructura prevista, y participan quince usuarios en la instancia, los cuales se distribuyen según edad y sexo, como se muestra en la Figura 4.1. Para cada usuario, se obtienen los resultados estimación según un modelo SMPL, y también, según SMPL-X.

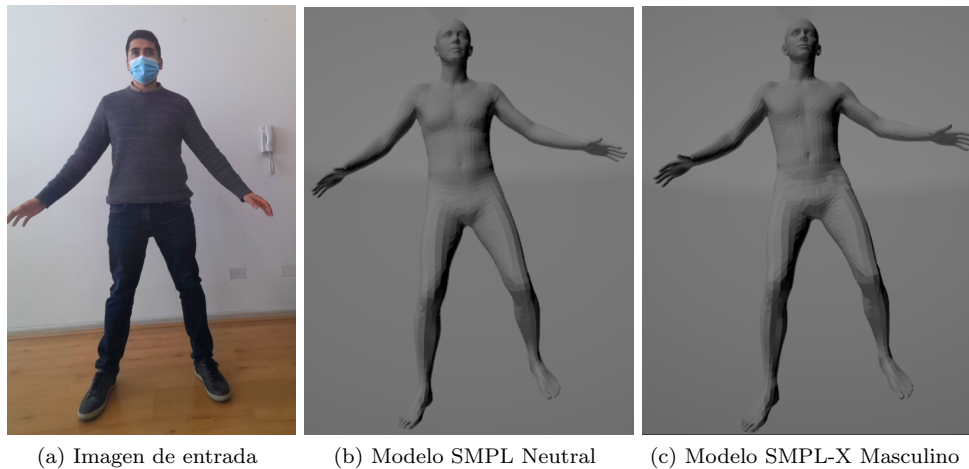
Rango etario	Sexo	
	Masculino	Femenino
16 a 20 años		
21 a 30 años		
31 a 40 años		
41 a 60 años		
más de 60 años		

Figura 4.1: Distribución de los usuarios partícipes de las pruebas.

A continuación se describen y analizan los resultados obtenidos. Inicialmente, se analizan cualitativamente los resultados de estimación de modelo corporal. Después, se cuantifica el error de estimación de las medidas y la talla generadas. Por último, se revisan los tiempos de ejecución en cada caso.

4.2.1. Modelo corporal

Al finalizar las pruebas, se obtienen los modelos corporales basados en SMPL y SMPL-X, estimados para cada usuario. Cabe señalar que, el modelo SMPL generado es del tipo neutral, mientras que el modelo SMPL-X varía según el género estimado. En general los modelos son similares entre sí, pero el usar un modelo específico por género, permite realzar las características corporales más típicas del género. Luego de analizar cualitativamente los resultados, se observa que los modelos corporales estimados caracterizan de manera correcta la morfología general y la pose de los usuarios, comparando con el contorno del cuerpo de la imagen original. Sin embargo, en general el cuerpo estimado es más delgado que el real. También, de manera más específica, se observa que el modelo estimado no acierta en la forma del cuerpo en las zonas de interés para la obtención de la talla. En otras palabras, el modelo estimado representa vagamente los contornos de pecho, cintura y cadera. En la Figura 4.2, se muestra a modo de ejemplo, modelos corporales obtenidos para un usuario, donde se pueden comprobar las observaciones cualitativas mencionadas.



(a) Imagen de entrada

(b) Modelo SMPL Neutral

(c) Modelo SMPL-X Masculino

Figura 4.2: Ejemplo de estimación de modelo corporal.

4.2.2. Medidas y Talla

Como se menciona anteriormente, luego de ejecutadas las pruebas, se dispone de los resultados de las medidas reales y las medidas estimadas para cada usuario. Las medidas de contorno disponibles, almacenadas en formato .csv, son: muñeca, cintura del pantalón, muslo, base del cuello, cuello, cintura, rodilla, cadera, codo, pecho, pantorrilla, bíceps y tobillo. Además, se dispone de las medidas de largo: estatura, hombro a hombro y hombro a muñeca.

Las pruebas se realizan ejecutando la estimación de estatura de manera automática, y luego, se emplea el valor de estatura estimado para pasar las medidas del modelo corporal a escala real. Por lo anterior, un aumento en el error de la estimación de estatura, aumentaría de manera proporcional el error en las mediciones. Es por esto que se cuantifica primero el error en la estimación de estatura y medidas corporales generadas por la aplicación, y luego, se calcula el error de estimación de medidas considerando escalar las medidas según la estatura real del usuario. Esto último, permite obtener un error de medición independiente del error añadido por la estimación de estatura.

Para el cálculo de las métricas de error en la estimación de cada medición, se considera cuantificar la diferencia entre el valor medido manualmente y el valor estimado por la aplicación. Se cuantifica a través de las métricas de Error cuadrático medio (*RMSE*, *Root Mean Square Error*) y Error relativo promedio (*MRE*, *Mean Relative Error*), donde x_i corresponde a cada muestra real, $\hat{x}_i$ a las muestras estimadas y N a la cantidad de muestras, en ambos casos.

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \hat{x}_i)^2} \quad (4.1)$$

$$MRE = \frac{1}{N} \sum_{i=1}^N \frac{(x_i - \hat{x}_i)}{x_i} \quad (4.2)$$

Considerando las definiciones anteriores, se calculan las métricas de error en la estimación de estatura para la ejecución basada en SMPL y SMPL-X, y ya que el proceso de estimación es el mismo en ambos casos, se promedian los resultados y se calculan métricas totales. En la Tabla 4.1, es posible observar las métricas asociadas. Si bien las métricas obtenidas son positivas, el nivel de error no es despreciable, considerando que el error asociado a la estatura afecta directamente a las mediciones a estimar.

Resultados estatura	SMPL	SMPL-X	TOTAL
MRE	2.34 %	3.55 %	2.965 %
RMSE [cm]	6.98	7.03	7.00

Tabla 4.1: Métricas de error en estimación de estatura.

Las métricas de error para las mediciones corporales se obtienen de manera análoga al caso anterior, es decir, se calcula RMSE y MRE para cada medición. En la Tabla 4.2, se observan las métricas de error resultantes para las pruebas con SMPL y SMPL-X. Además, se calcula el Error en mediciones de interés, que se basa en el promedio de los errores de CinturaPantalon, Cadera y Pecho. También, se calcula el Error total, basado en el promedio

de todos los errores de medición.

	SMPL		SMPL-X	
Medición	RMSE [cm]	MRE	RMSE [cm]	MRE
Muñeca	1.64	6.29 %	1.72	7.43 %
CinturaPantalon	15.35	12.37 %	15.87	13.20 %
Muslo	7.30	11.85 %	8.50	14.14 %
Hombro-Muñeca	4.29	5.91 %	4.95	6.41 %
Hombro-Hombro	7.14	12.56 %	7.55	11.96 %
BaseCuello	4.86	9.48 %	3.46	6.96 %
Cuello	2.53	5.59 %	3.08	6.84 %
Cintura	11.03	8.78 %	11.76	10.04 %
Rodilla	5.20	8.61 %	5.85	9.83 %
Cadera	11.63	8.18 %	12.14	8.81 %
Codo	2.19	5.61 %	3.20	9.27 %
Pecho	12.01	8.62 %	8.87	6.78 %
Pantorrilla	5.11	8.74 %	5.73	10.69 %
Bicep	3.46	9.08 %	6.74	20.80 %
Tobillo	3.33	12.24 %	4.25	15.96 %
Error m. Interés	13.00	9.72 %	12.29	9.60 %
Error Total	6.47	8.93 %	6.91	10.61 %

Tabla 4.2: Métricas de error en medidas generadas por la aplicación completa.

Después de analizar los resultados en la Tabla 4.2, en general se observan métricas positivas. Cabe señalar que el error en las métricas de interés es mayor al error total en ambos casos. Esto se condice con los resultados cualitativos de los modelos corporales, donde no se muestra de manera realista la forma corporal en las zonas de pecho, cintura y cadera. Zonas donde se espera que el error sea mínimo, ya que esas medidas caracterizan la talla de vestir del usuario.

Como se menciona anteriormente, se calculan métricas de error para las mediciones corporales, independientes del error añadido por la estatura estimada. Para esto, se modifica la escala de las mediciones, para que estas se basen en el valor real de estatura del usuario para su estimación. De esta manera, el error de las mediciones sólo considera las diferencias basadas en la forma del modelo corporal. Las métricas de error se obtienen de manera análoga al caso anterior, es decir, se calcula MRE y RMSE para cada medición. En la Tabla 4.3, se observan las métricas de error resultantes para SMPL y SMPL-X. Al revisar los

	SMPL		SMPL-X	
Medición	RMSE [cm]	MRE	RMSE [cm]	MRE
Muñeca	1.58	6.24 %	1.30	5.94 %
CinturaPantalon	15.55	12.40 %	14.21	12.45 %
Muslo	7.15	12.13 %	7.17	11.30 %
Hombro-Muñeca	3.35	4.98 %	4.66	5.81 %
Hombro-Hombro	6.91	12.60 %	7.34	12.32 %
BaseCuello	4.29	8.18 %	3.76	7.73 %
Cuello	2.02	4.52 %	3.33	6.32 %
Cintura	10.71	8.73 %	10.57	9.11 %
Rodilla	5.42	8.82 %	4.86	8.61 %
Cadera	12.20	8.33 %	9.72	7.66 %
Codo	2.18	5.78 %	2.44	7.27 %
Pecho	11.87	8.21 %	6.95	4.60 %
Pantorrilla	5.00	8.78 %	4.73	9.15 %
Bicep	3.01	8.34 %	6.77	21.30 %
Tobillo	3.71	13.78 %	4.01	15.41 %
Error m. Interés	13.20	9.64 %	10.29	8.24 %
Error Total	3.21	8.79 %	6.12	9.66 %

Tabla 4.3: Métricas de error en medidas generadas según la estatura real.

resultados de la Tabla 4.3, se observan resultados similares al caso anterior. Sin embargo, se observa que las métricas de error son levemente mejores en el caso de SMPL-X, y es más clara la diferencia en el caso de las medidas de interés. Esto se debe netamente al realce en las características propias de cada género en SMPL-X.

Al comparar los resultados de las Tablas 4.2 y 4.3, se observa que las métricas de error se comportan según lo esperado, ya que en general disminuyen su valor al eliminar el efecto de la estimación de estatura.

Finalmente, se cuantifica el error de estimación de talla de vestir de la aplicación. Para esto, se define una métrica de error, con el objetivo de comparar los valores estimados de talla con los valores reales. Se define, un 0 % de error cuando la talla estimada corresponde a la real, un error de 50 % cuando existe una talla de diferencia entre estimada y real, y un 100 % de error cuando existen dos o más tallas de diferencia. Por ejemplo, si la talla estimada es M y la real es L, se considera un 50 % de error.

Considerando la métrica de error para la estimación de talla, se realiza un promedio entre el error de talla para todos los usuarios. Se obtiene un error de talla del 54 % para SMPL y del 49 % para SMPL-X.

4.2.3. Medidas Ajustadas

Ya que el modelo corporal generado es más delgado que el cuerpo real de los usuarios, se plantea ajustar las medidas obtenidas considerando la estatura real, multiplicando cada valor obtenido por un factor y así minimizar el error de estimación.

Se realizan pruebas con diferentes factores porcentuales, y se observa que al aumentar cada medida estimada en un 6 %, las métricas de error disminuyen considerablemente. En promedio, las métricas de error en medidas de interés se reducen en un 24,05 % y las métricas de error totales se reducen en un 1,46 %. En la Tabla 4.4 es posible observar los resultados de error, considerando un aumento del 6 % para cada medida.

Para hacer un ejercicio comparativo, se promedian las métricas totales y de interés de MRE obtenidas para cada variación presentada. Primero se calcula el promedio para la aplicación completa, donde se obtiene un MRE general de 9,75 %. Luego, para las medidas considerando la estatura real el MRE general es de 9,08 %. Por último para las medidas ajustadas porcentualmente se obtiene un MRE general de 8,19 %.

	SMPL		SMPL-X	
Medición	RMSE [cm]	MRE	RMSE [cm]	MRE
Muñeca	1.61	7.93 %	1.58	8.74 %
CinturaPantalon	11.45	8.68 %	9.68	8.27 %
Muslo	9.78	17.76 %	9.85	16.99 %
Hombro-Muñeca	3.45	4.97 %	4.12	5.55 %
Hombro-Hombro	5.36	9.21 %	6.24	11.30 %
BaseCuello	6.39	13.92 %	5.90	13.23 %
Cuello	2.60	5.63 %	3.88	6.66 %
Cintura	8.02	7.06 %	6.99	5.56 %
Rodilla	4.64	8.64 %	3.83	7.19 %
Cadera	8.82	5.46 %	6.36	4.71 %
Codo	1.70	5.27 %	2.41	7.77 %
Pecho	9.31	6.74 %	7.93	6.51 %
Pantorrilla	3.84	6.82 %	3.63	7.13 %
Bicep	2.36	6.67 %	8.77	28.45 %
Tobillo	4.75	17.41 %	5.29	19.76 %
Error m. Interés	9.86	6.96 %	7.99	6.50 %
Error Total	2.85	8.81 %	5.76	10.52 %

Tabla 4.4: Métricas de error en medidas basadas en estatura real, ajustadas porcentualmente.

4.2.4. Tiempo de ejecución

Como se menciona anteriormente, se disponen de los resultados de tiempo de ejecución para cada módulo generados durante la realización de las pruebas. Para cuantificar los resultados, simplemente se realiza un promedio de cada tiempo obtenido. Los valores promedio de tiempo de ejecución se pueden observar en la Tabla 4.5, donde se pueden observar tiempos razonables menores a 15[seg] para la ejecución de la aplicación según SMPL, mientras que para SMPL-X, la duración de la etapa de conversión es excesiva. Esto se debe a que es una etapa basada en optimización, y el tiempo de convergencia del algoritmo

es variable. Por lo tanto, la estimación según SMPL, cumple con las condiciones técnicas de la solución, mientras que la estimación según SMPL-X requiere mejoras para satisfacer los requerimientos.

	Tiempo de ejec. promedio[seg]	
Módulo	SMPL	SMPLX
Estimación de pose	2.45	2.72
Estimación de género	1.49	1.66
Recorte	0.05	0.06
Estimación de modelo SMPL	8.27	9.12
Conversor SMPL a SMPL-X	-	138.22
Estimación de medidas	1.01	1.14
Total	13.28	143.64

Tabla 4.5: Tiempos de ejecución promedio para estimación según SMPL y SMPL-X

4.3. Validación

Al analizar el desempeño de la aplicación según los resultados de las pruebas y el cálculo de métricas pertinentes, es posible validar la solución implementada dando respuesta a las preguntas de trabajo mencionadas en la Sección 1.3, las cuales se listan a continuación:

- *¿Es necesario utilizar una cámara monocular RGB-D como dispositivo de entrada? ¿Es posible utilizar solo una cámara monocular RGB para estimar un modelo del usuario?*

Durante el análisis de pruebas preliminares en el Capítulo 3.1, se plantea que es posible obtener un modelo corporal realista utilizando sólo una captura RGB como entrada, por lo cual, es posible inferir la forma corporal, utilizando sólo una imagen a color, lo que se implementa en la solución a través del método BCNet.

Por otro lado, también es necesario pasar las medidas estimadas a escala real. Esto se logra utilizando la estatura del usuario como valor de referencia. Para obtener la estatura del usuario, es posible utilizar las técnicas de geometría proyectiva mencionadas en la Sección 2.1.1, y pasar ciertos píxeles de interés a coordenadas reales.

Utilizando las técnicas mencionadas, es posible utilizar sólo una cámara RGB para implementar la solución. Sin embargo, el proyecto actual basa la estimación de estatura en una cámara RGB-D. Por lo tanto, considerando el nivel de desarrollo actual, aún no es posible prescindir del sensor RGB-D para implementar la solución.

- *¿Es posible estimar un modelo digital tridimensional del cuerpo del usuario, que represente claramente su morfología y rasgos cualitativos?*

Si, es posible generar un modelo corporal. En la Sección 4.2.1 del presente capítulo se muestra la forma de modelos generados. Los modelos se adaptan al cuerpo del usuario, pero aún son mejorables en cuanto a la forma corporal en las zonas de pecho, cintura y cadera.

- *¿Es posible estimar con precisión las mediciones antropométricas relacionadas con la talla de vestir del usuario, considerando un error de estimación menor al 15 % relativo a su valor real*

Sí, es posible estimar las medidas. En la Sección 4.2.2 se muestran las métricas de error obtenidas al estimar las medidas, las cuales se encuentran menores al 10 % relativo en todos los casos. Además, en la Sección 4.2.3 se realiza un ajuste lineal a las medidas, lo que entrega métricas aún mejores.

- *¿Es posible implementar la solución en tiempo real, considerando un tiempo de ejecución total menor a 15[seg]?*

En la Sección 4.2.2 del presente capítulo, se presentan los tiempos de ejecución para cada caso. En el caso de la estimación para SMPL, es posible cumplir con los requerimientos de tiempo. Sin embargo, para el caso de SMPL-X, los tiempos se disparan, ya que el conversor entre modelos, demora en ejecutarse. Por lo tanto, es posible cumplir con el requisito de tiempo real estimando un modelo SMPL, mientras que, al estimar un modelo SMPL-X e incluir la pose de las manos, no se cumple el requisito.

CONCLUSIONES Y TRABAJOS FUTUROS

5.1. Conclusiones

EN esta memoria se presentó una aplicación ejecutable desde el motor de videojuegos Unity 3D, que permite estimar un modelo corporal de un usuario vestido y sus medidas antropométricas, basándose en la captura de una cámara RGB-D.

Primero se hizo una revisión del marco teórico en el que se sustenta la problemática a resolver, se estudian conceptos de visión computador, animación digital y diseño de vestuario. Además, se revisaron métodos del estado del arte, que permiten dar respuesta a la problemática y se compararon entre sí a nivel teórico. De esta manera, se logró establecer el conjunto de herramientas que permiten implementar la solución al problema.

En el siguiente capítulo, se realizó una validación preliminar de los métodos del estado del arte. Luego, se escogió el método BCNet para implementar la estimación corporal, ya que entrega los mejores resultados cualitativos y de tiempo de estimación. También, se planteó una mejora en la estimación corporal, considerando integrar el método BCNet con el método PifuHD, pero se descartó su implementación en el proyecto actual.

Además, se hizo un análisis de las herramientas que permiten implementar la integración entre el entorno del motor de videojuegos Unity 3D y el entorno del lenguaje Python. Para esto, se ejecutaron pruebas con diferentes herramientas, y se decidió utilizar un intérprete de Python externo al motor e implementar la integración a través de la biblioteca ZeroMQ.

Posteriormente, se implementó la solución basada en las herramientas y métodos escogidos, donde se desarrolló la interfaz de usuario y los resultados visuales en Unity 3D. Los algoritmos de procesamiento e inferencia se ejecutaron en un entorno de Python. Se utilizó una cámara RGB-D Azure Kinect como dispositivo de entrada, lo que permitió automatizar la captura RGB y obtener una referencia del valor de la estatura del usuario. Se implementó el método BCNet para estimar el modelo corporal y, se incluyó una etapa de conversión desde un modelo SMPL a un modelo SMPL-X, lo que permite retornar un modelo que incluya la animación de las manos. En el mismo contexto, se añadió una etapa final de estimación de mediciones corporales, basada en el modelo corporal generado.

Finalmente, se evaluó el desempeño de la aplicación a través de pruebas realizadas a quince usuarios. Para esto, se evaluó la forma del modelo estimado de forma cualitativa, el cual se adapta al cuerpo de cada usuario, pero no logra definir correctamente la morfología del usuario en las zonas de pecho, cintura y cadera. Se analizaron métricas de error para las mediciones corporales estimadas, donde en general se observan métricas positivas, en general se obtuvo un 9,7 % de error relativo promedio. Además, se realizó un ajuste lineal al valor de las métricas, mejorando aún más los resultados, obteniendo en general un 8,19 % de error relativo promedio, comprobando así que las mediciones estimadas son suficientes para ser utilizadas en aplicaciones de la industria del vestuario. Por otro lado, se revisaron los resultados de tiempo de ejecución de las pruebas, los cuales son positivos para el caso de SMPL, obteniendo 13,28[seg] en promedio, en cambio para SMPL-X, se obtienen 143,6[seg], esto debido a que el conversor de SMPL a SMPL-X tarda bastante en optimizar.

5.2. Trabajo Futuro

En el corto y mediano plazo se plantean una serie de mejoras a la aplicación desarrollada, con el objetivo de cumplir con los requerimientos futuros planteados por la empresa, y también, pensando en mejorar los resultados obtenidos. Como mejora inicial, es necesario disminuir el tiempo de procesamiento de la etapa de conversión del modelo SMPL a SMPL-X, para así cumplir con el requisito de tiempo real (tiempo de ejecución menor a 15[seg]). La implementación actual basa la conversión en la optimización de todos los parámetros de SMPL-X, incluyendo pose corporal, pose de las manos, forma corporal y expresión facial. En cambio, el modelo SMPL sólo considera parámetros de pose y forma corporal. Por lo tanto, se plantea como mejora optimizar sólo los parámetros de pose y forma corporal del modelo a generar. En el mismo contexto, sabiendo que el usuario se posiciona en una posición cercana a la pose 'A', inicializar el modelo SMPL-X a optimizar en pose 'A', haría que el modelo a generar converja más rápido.

Considerando la mejora anterior, debería ser posible disminuir los tiempos de optimización considerablemente. En caso de no ser suficiente la disminución, la alternativa es utilizar otro método de estimación de modelo corporal, alguno que retorne un modelo SMPL-X basado en una imagen RGB.

Es también necesario mejorar la forma corporal del modelo estimado, para que se asemeje más al usuario en las zonas de pecho, cintura y cadera. En la Figura 3.10, se muestra una estructura para la estimación del modelo corporal, basada en la integración de los métodos BCNet y PifuHD. De esta manera, se plantea una mejora del modelo corporal generado por BCNet, implementada a través de un ajuste específico en las zonas de pecho, cintura y cadera, según los resultados de reconstrucción de PifuHD.

Otra posibilidad de mejora, es prescindir del uso de una cámara RGB-D como Azure Kinect. Para lograr esto, es necesario modificar el método de estimación de estatura implementado, ya que éste utiliza la captura de profundidad como referencia para obtener la estatura del usuario. La mejora planteada es estimar la estatura basándose en una transformación de la posición de los joints de pose del usuario a coordenadas reales. Para implementar la transformación, es necesario tener una referencia real y aplicar las técnicas de geometría proyectiva mencionadas en la Sección 2.1.1.

Como última mejora planteada, es posible modificar ejecución secuencial de módulos en la solución. En la implementación actual, se ejecuta un módulo a la vez, es decir, es necesario que un módulo finalice su ejecución para que inicie el siguiente. La mejora consiste en permitir la ejecución de más de un módulo en paralelo, siempre y cuando se tengan los

datos de entrada necesarios. Para implementar esto, se requiere que el cliente genere un hilo de ejecución por cada módulo iniciado, y que el servidor implemente lo mismo.

BIBLIOGRAFÍA

- [1] B. Jiang, J. Zhang, Y. Hong, J. Luo, L. Liu, and H. Bao, “Bcnet: Learning body and cloth shape from a single image,” 2020.
- [2] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, “Smpl: A skinned multi-person linear model,” *ACM Trans. Graph.*, vol. 34, no. 6, Oct. 2015. [Online]. Available: <https://doi.org/10.1145/2816795.2818013>
- [3] J. Chibane, T. Alldieck, and G. Pons-Moll, “Implicit functions in feature space for 3d shape reconstruction and completion,” 2020.
- [4] B. L. Bhatnagar, C. Sminchisescu, C. Theobalt, and G. Pons-Moll, “Combining implicit function learning and parametric models for 3d human reconstruction,” 2020.
- [5] C. Zhang, S. Pujades, M. Black, and G. Pons-Moll, “Detailed, accurate, human shape estimation from clothed 3d scan sequences,” 2017.
- [6] G. Moon and K. M. Lee, “I2l-meshnet: Image-to-lixel prediction network for accurate 3d human pose and mesh estimation from a single rgb image,” 2020.
- [7] S. Saito, T. Simon, J. Saragih, and H. Joo, “Pifuhd: Multi-level pixel-aligned implicit function for high-resolution 3d human digitization,” 2020.
- [8] S. Yan, J. Wirta, and J.-K. Kämäräinen, “Anthropometric clothing measurements from 3d body scans,” 2019.
- [9] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, “Kinectfusion: Real-time dense surface mapping and tracking,” in *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, 2011, pp. 127–136.
- [10] R. A. Newcombe, D. Fox, and S. M. Seitz, “Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 343–352.
- [11] T. Yu, Z. Zheng, K. Guo, J. Zhao, Q. Dai, H. Li, G. Pons-Moll, and Y. Liu, “Doublefusion: Real-time capture of human performances with inner body shapes from a single depth sensor,” 2018.
- [12] A. Osman, T. Bolkart, and M. Black, “Star: Sparse trained articulated human body regressor,” 08 2020.

- [13] Q. Ma, J. Yang, A. Ranjan, S. Pujades, G. Pons-Moll, S. Tang, and M. J. Black, “Learning to dress 3d people in generative clothing,” 2020.
- [14] G. Pavlakos, V. Choutas, N. Ghorbani, T. Bolkart, A. A. A. Osman, D. Tzionas, and M. J. Black, “Expressive body capture: 3d hands, face, and body from a single image,” 2019.
- [15] K. Simmons, “Body measurement techniques: a comparison of three-dimensional body scanning and physical anthropometric methods,” 01 2001.
- [16] L. Wang, X. Zhao, T. Yu, S. Wang, and Y. Liu, “Normalgan: Learning detailed 3d human from a single rgb-d image,” 2020.
- [17] B. L. Bhatnagar, C. Sminchisescu, C. Theobalt, and G. Pons-Moll, “Loopreg: Self-supervised learning of implicit surface correspondences, pose and shape for 3d human mesh registration,” 2020.
- [18] S. Saito, Z. Huang, R. Natsume, S. Morishima, A. Kanazawa, and H. Li, “Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization,” 2019.
- [19] A. Weiss, D. Hirshberg, and M. J. Black, “Home 3d body scans from noisy image and range data,” in *2011 International Conference on Computer Vision*, 2011, pp. 1951–1958.
- [20] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, “Openpose: Realtime multi-person 2d pose estimation using part affinity fields,” 2019.

