



Escuela  
Politécnica  
Superior

# Reconstrucción 3D del cuerpo humano a partir de imágenes mediante aprendizaje profundo



Grado en Ingeniería Informática

## Trabajo Fin de Grado

Autora:

Camelia Beltrá García

Tutores:

Jorge Azorín López

Andrés Fuster Guilló



Universitat d'Alacant  
Universidad de Alicante



# Reconstrucción 3D del cuerpo humano a partir de imágenes mediante aprendizaje profundo

---

Subtítulo del proyecto

**Autora**

Camelia Beltrá García

**Tutores**

Jorge Azorín López

Andrés Fuster Guilló

*TECNOLOGIA INFORMATICA Y COMPUTACION*



Grado en Ingeniería Informática



Escuela  
Politécnica  
Superior



Universitat d'Alacant  
Universidad de Alicante

ALICANTE, Julio 2022



# Resumen

Actualmente hay muchos estudios y proyectos llevados a cabo para la reconstrucción 3D del cuerpo humano, en diferentes ámbitos de aplicación, desde el textil, deportivo o los videojuegos hasta las aplicaciones médicas como es el caso dietético-nutricional que enmarca este proyecto.

Este trabajo tiene como objetivo la obtención del modelo 3D del cuerpo humano a partir de imágenes obtenidas por cámaras de consumo, como las disponibles en los dispositivos móviles.

Dado que estamos en un ámbito sanitario, se necesitan comparaciones sobre las medidas obtenidas a partir de los modelos 3D del cuerpo humano ya que estas deben ser precisas, para ello se va a utilizar el proyecto llamado Tech4diet [Tech4Diet (2019)], que es capaz de representar la forma y textura de forma precisa dado que realiza esta adquisición con una red de cámaras RGBD.

Para abordar el objetivo se propone el uso de métodos basados en aprendizaje profundo capaces de proporcionar la reconstrucción 3D del cuerpo a partir de una sola vista 2D de la persona.



# Agradecimientos

Este trabajo no habría sido posible sin el apoyo de mis tutores Andrés y Jorge, como de Nahuel, su paciencia, disposición y comprensión tanto con mis dudas como conmigo misma, han sido clave durante este proyecto.

También quiero agradecer a mis amigos, que por muy lejos que estén han estado cerca de mí apoyándome y animándome para terminar mis estudios en uno de los momentos más complejos de mi vida.

Agradecer a mi familia, que siempre han tratado de darme soluciones a mis problemas apoyarme y animarme para terminar este ciclo.

No puedo terminar estos agradecimientos sin mencionar a mis compañeros de la Universidad de Alicante; Luis, Sandra, Edgar, Dani y el resto de personas que han estado a mi lado, gracias por los años que he estado a vuestro lado y por toda la ayuda y apoyo que nos hemos brindado, estoy muy agradecida de haberos tenido a mi lado durante mis años de carrera.

Es a ellos a quien dedico este trabajo.





*A mi abuela Carmen, que no ha podido verme terminar esta etapa.*



# Índice general

<b>1</b>	<b>Introducción</b>	<b>1</b>
1.1	Motivación y contexto . . . . .	1
1.2	Estado del arte . . . . .	4
1.3	Objetivos . . . . .	10
<b>2</b>	<b>Antecedentes Metodológicos y Tecnológicos</b>	<b>11</b>
2.1	Marching Cubes . . . . .	11
2.2	PIFu: Pixel-aligned Implicit function . . . . .	12
2.2.1	Reconstrucción de la superficie . . . . .	13
2.2.2	Obtención de la Textura . . . . .	15
2.3	Cálculo de distancias . . . . .	16
2.3.1	Distancia Chamfer . . . . .	16
2.3.2	Distancia Hausdorff . . . . .	17
2.4	Herramientas tecnológicas . . . . .	17
2.4.1	GIMP . . . . .	17
2.4.2	MeshLab . . . . .	17
2.4.3	Blender . . . . .	18
<b>3</b>	<b>Reconstrucción del modelo 3D y textura a partir de una imagen</b>	<b>19</b>
3.1	Estudio de los parámetros de test . . . . .	19
3.2	Enmascaramiento de imágenes . . . . .	20
3.2.1	Enmascaramiento de imágenes con el fondo de un sólo color . . . . .	20

3.2.2	Enmascaramiento de fotos realizadas con teléfonos móviles. . . .	21
3.2.2.1	Preprocesado de las fotos . . . . .	22
3.3	Obtención del modelo 3D . . . . .	24
<b>4</b>	<b>Experimentación</b>	<b>25</b>
4.1	Preprocesado de los modelos de Tech4diet . . . . .	25
4.2	Alineamiento de modelos . . . . .	26
4.3	Creación de las nubes de punto correspondientes a cada modelo . . . .	27
4.4	Cálculo distancias . . . . .	28
4.4.1	Distancia Hausdorff . . . . .	28
4.4.1.1	Quality Mapper . . . . .	29
4.4.2	Distancia Chamfer . . . . .	30
4.5	Análisis y comparativas de los modelos . . . . .	31
<b>5</b>	<b>Conclusiones</b>	<b>33</b>
5.1	Conclusión . . . . .	33
5.2	Líneas Futuras . . . . .	34
	<b>Bibliografía</b>	<b>35</b>

---

# Índice de figuras

1.1	Captación del modelo del cuerpo mediante el sistema de cámaras (a). Nubes de puntos sin textura generada (b). Mediciones del cuerpo (c). Visualización de los resultados en la aplicación y con gafas de realidad virtual [García-D’Urso y cols. (2021)] . . . . .	2
1.2	Set-up de red de 13 cámaras RGB-D. "nv" indica que la cámara no está visible en la imagen [García-D’Urso y cols. (2021)]. . . . .	5
1.3	Ejemplos. forma y pose 3D estimada mediante el método desarrollado por Bogo[Bogo y cols. (2016)] usando las fotos de Leeds Sports Pose Dataset[Johnson y Everingham (2010)]. Se ve la imagen original a la izquierda, luego el modelo comparado en la foto en el centro y por último a la derecha el modelo 3D desde otro punto de vista. . . . .	6
1.4	Descripción general de la propuesta PyMAF. PyMAF aprovecha una Feature Pyramid. Dada una predicción del modelo, los alineamientos de la malla son extraídos de características de una resolución más fina y retroalimentan el regresor para así actualizar la malla. [H. Zhang y cols. (2021)] . . . . .	7

---

1.5	Arquitectura general de la propuesta BodyNet[Varol y cols. (2018)]. Donde comienza a partir de una imagen RGB que se le pasa a dos subredes donde estiman tanto la pose como la segmentación 2D, con estas predicciones y la imagen se utilizan como entrada para la siguiente red, que estima en este caso la pose 3D, con todo lo anterior se utiliza como entrada en la siguiente red donde obtiene ya la forma del cuerpo, donde ya por último se ajusta con el modelo SMPL[Loper y cols. (2015)]	8
1.6	Ciclo de obtención de textura[Kanazawa y cols. (2018)] . . . . .	8
2.1	Pixel-aligned Implicit function (PIFu) [Saito y cols. (2019)]. . . . .	12
2.2	Pipeline usada en (PIFu) [Saito y cols. (2019)]. Dada una imagen, PIFu predice la probabilidad continua interior/exterior de un cuerpo humano vestido. Tex-PIFu infiere un valor RGB dados los puntos 3D de la su- perficie con con topología arbitraria . . . . .	14
3.1	Pipeline usada en para la obtención del modelo. De izquierda a derecha, dada una imagen $I$ , se escala, recorta y se ajustan los colores generando la imagen $i$ y se genera la máscara de la imagen $m_i$ a partir de $i$ . La red de PIFu utilizará las imágenes $i$ y $m_i$ para generar el modelo que se ve a la derecha. . . . .	24
4.1	Proceso limpieza modelo Tech4diet . . . . .	25
4.2	Proceso alineamiento modelos . . . . .	26
4.3	Modelos alineados . . . . .	27
4.4	Diferencia de altura de los modelos, el modelo de la derecha de cada pareja es el generado por la red . . . . .	31
4.5	Diferencia de altura de los modelos, ya escalados . . . . .	31

---

# Índice de tablas

3.1	Parámetros de las cámaras elegidas. . . . .	22
-----	---	----





# Índice de Códigos

3.1	Código obtención máscara 1 . . . . .	21
3.2	Código obtención máscara 2 . . . . .	23
4.1	Código obtención distancia chamfer . . . . .	30



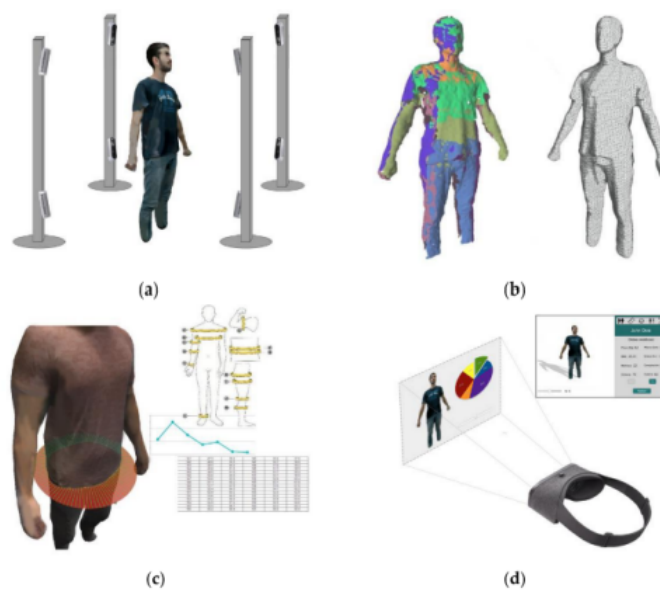
# 1 Introducción

## 1.1 Motivación y contexto

Mi interés por la visión artificial se desarrolló durante el tercer año de carrera, donde me presenté junto a la Universidad de Alicante a un proyecto llamado Vodafone Campus Lab[Vodafone (2020)], donde te proponen unos problemas a resolver y nosotros decidimos plantear una solución con tecnologías como la visión artificial, además esta propuesta requería de procesos como la adquisición del cuerpo 3D y reconocimiento de este, entre otros. Aquí fue cuando busqué propuestas de trabajo final de grado (TFG) similares o sobre este tema, porque me quedé con la curiosidad de llevarlo a cabo y porque quería saber, aprender e investigar más sobre la adquisición del objetos 3D a partir de imágenes, cuando vi las propuestas fue cuando conocí el proyecto de investigación [Tech4Diet (2019)].

El proyecto de investigación Tech4Diet cuenta con el apoyo de la Agencia Estatal de Investigación (AEI) y del Fondo Europeo de Desarrollo Regional (FEDER) con referencia "TIN2017-89069-R" perteneciente al programa Retos 2017 en el que su investigador jefe es Jorge Azorín. En este proyecto se busca facilitar el estudio de la evolución morfológica ocasionada por tratamientos de obesidad. Hoy en día, estos tratamientos son muy costosos pero a su vez muy necesarios, ya que los problemas de obesidad o sobrepeso pueden ocasionar enfermedades crónicas como la hipertensión, diabetes tipo II, cáncer. También pueden ocasionar enfermedades patológicas neurodegenerativas como el Alzheimer o demencias [Fuster-Guilló y cols. (2020)]

El sistema utilizado para esta finalidad, dispone de una red de cámaras RGB-D que obtienen un modelo 3D del cuerpo del paciente. El proceso de obtención de un modelo 3D se realiza en diferentes sesiones médicas, lo que permite una visualización real de la evolución del cuerpo del paciente. Para que el paciente pueda visualizar su progreso, no solo dispondrá de una aplicación de ordenador, sino que también podrá visualizar los diferentes modelos de su cuerpo mediante unas gafas de realidad virtual. La realidad virtual tiene como finalidad incrementar la adherencia del usuario al tratamiento. Además, podemos encontrar desarrollos tecnológicos como "Google Cardboard" que nos permiten convertir cualquier teléfono móvil en unas gafas de realidad virtual sin necesidad de realizar un gasto de dinero elevado. A parte de la visualización, sobre este modelo 3D obtenido con las cámaras se pueden realizar mediciones de diferentes partes del cuerpo a niveles de 1D, 2D y 3D.



**Figura 1.1:** Captación del modelo del cuerpo mediante el sistema de cámaras (a). Nubes de puntos sin textura generada (b). Mediciones del cuerpo (c). Visualización de los resultados en la aplicación y con gafas de realidad virtual [García-D'Urso y cols. (2021)]

En este contexto, este trabajo final de grado tiene como objetivo principal la re-

---

---

construcción de modelos 3D del cuerpo humano con textura, a partir de imágenes 2D adquiridas con dispositivos de consumo como las disponibles en teléfonos móviles. Esto permite mejorar la portabilidad y coste de sistemas basados en redes de cámaras RGBD, como es el caso de Tech4diet[Tech4Diet (2019)]. En este TFG se analizará de forma comparativa la precisión de las medidas de los modelos 3D obtenidos en relación a las medidas obtenidas mediante los sistemas más costosos como el de Tech4diet.

---

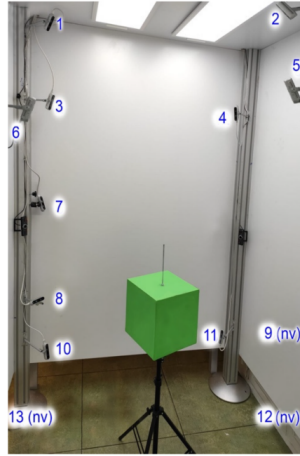
## 1.2 Estado del arte

La reconstrucción del cuerpo humano 3D es un tema amplio abordado de diferentes maneras, por una parte tenemos la visión por computador que nos permite entender la información visual del entorno capturada a través de las cámaras [Z. Zhang (2016)], por otro lado nos encontramos avances recientes en la estimación del cuerpo humano 3D basado en imágenes que han sido impulsados por la mejora significativa en el poder de representación que ofrecen las redes neuronales profundas. [Saito y cols. (2020)]

Se ha llevado a cabo una revisión del estado del arte que abordan las diferentes formas de obtener resultados respecto a la reconstrucción del cuerpo humano.

Por un lado, recientes trabajos abordados mediante el uso de la visión por computador utilizan cámaras RGB-D calibradas [García-D'Urso y cols. (2021)]. El trabajo abordado por García D'Urso propone el uso de dispositivos RGB-D (como Microsoft Kinect o Intel RealSense), debido a que integran sensores de color y profundidad, y utilizan tecnologías de profundidad como luz estructurada, ToF (Time of Flight, sensor que mide distancias utilizando el tiempo que usan los fotones en viajar entre dos puntos) o activa estereoscópica. En este proyecto aparte de la obtención del modelo, también han tratado otros puntos como la visualización 3D del cuerpo utilizando la realidad virtual, además de que pueden obtener las medidas de volúmenes seleccionados del cuerpo humano. Con esta propuesta lo que llevan a cabo es una red de cámaras RGB-D de 13 cámaras ubicadas en una cabina donde la persona se ha de colocar durante 4 segundos como se puede ver en la figura 1.2, esta red de cámaras se ha de calibrar de manera que cada una de ellas quede calibrada intrínsecamente y extrínsecamente para cada cámara y multi-cámara para así estimar la posición relativa de cada sensor en la red, una vez la red de cámaras se encuentren calibradas sigue el proceso de la obtención del modelo 3D con textura que consta de 5 fases principales, que se ven en la figura 1.1: adquisición, pre-procesado, unión de vistas, generación de la malla y proyección de la textura.

---



**Figura 1.2:** Set-up de red de 13 cámaras RGB-D. "nv" indica que la cámara no está visible en la imagen [García-D'Urso y cols. (2021)].

Por otro lado nos encontramos recientes trabajos abordados mediante el uso de visión artificial y deep learning, en estos a diferencia del anterior no se necesita una cabina formada por una red de cámaras. Todos estos trabajos tienen el mismo objetivo de producir resultados naturales y bien alineados [H. Zhang y cols. (2021)], en concreto han habido varios paradigmas investigados, por un lado tenemos los métodos basados en la optimización que ajustan explícitamente los modelos a las imágenes 2D [Bogo y cols. (2016)], dentro de este paradigma nos encontramos con el trabajo de investigación por parte de Bogo que desarrolla un método para automáticamente estimar la pose y forma 3D del cuerpo humano a partir de una imagen, para ello siguen dos pasos, primero estiman los 2D joints (uniones, en este caso sería por ejemplo la rodilla que une la parte inferior y superior de la pierna), esto lo consiguen usando una red neuronal convolucional (CNN) llamado DeepCut [Pishchulin y cols. (2015)], esta red es buena estimando la pose 2D pero no bueno con la pose 3D por lo que el siguiente paso es el que estima la pose y la forma 3D partiendo de los 2D joints utilizando el modelo SMPL [Loper y cols. (2015)] encajando dentro de un paradigma clásico bottom up de estimación (CNN) seguida de una verificación descendente (modelo generativo) como se puede ver en la figura 1.3



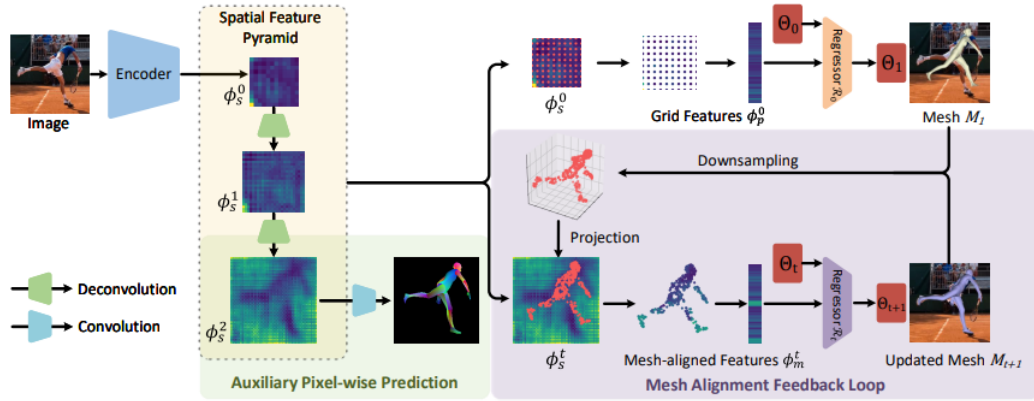
**Figura 1.3:** Ejemplos. forma y pose 3D estimada mediante el método desarrollado por Bogo[Bogo y cols. (2016)] usando las fotos de Leeds Sports Pose Dataset[Johnson y Everingham (2010)]. Se ve la imagen original a la izquierda, luego el modelo comparado en la foto en el centro y por último a la derecha el modelo 3D desde otro punto de vista.

Los métodos basados en la optimización como el de Bogo ajustan explícitamente los modelos a las imágenes, lo que produce resultados con buena precisión de los alineamientos entre malla-imagen, pero tienden a ser lentos y sensibles al inicio[H. Zhang y cols. (2021)].

Y, por otro lado nos encontramos con métodos basados en modelos de regresión que sugieren directamente predecir el modelo a partir de los parámetros de las imágenes [H. Zhang y cols. (2021)], en este apartado nos encontramos propuestas similares a la de H. Zhang, estos modelos se basan en los parámetros existentes en las imágenes y están demostrando resultados prometedores pero aún existen errores principalmente de alineamientos entre malla e imagen, para resolver este problema H. Zhang propone y diseña PyMAF (Pyramidal Mesh Alignment Feedback).

La idea central de este enfoque es corregir las desviaciones paramétricas de manera explícita y progresiva basado en el estado del alineamiento, en PyMAF el alineamiento se realiza extrayendo las características espaciales respecto a la proyección 2D de la malla estimada y luego retroalimenta los regresores para que actualicen los parámetros.

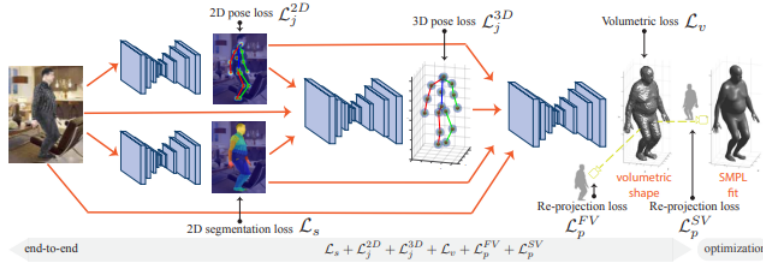




**Figura 1.4:** Descripción general de la propuesta PyMAF. PyMAF aprovecha una Feature Pyramid. Dada una predicción del modelo, los alineamientos de la malla son extraídos de características de una resolución más fina y retroalimentan el regresor para así actualizar la malla. [H. Zhang y cols. (2021)]

Existen otro tipo de paradigma investigado que esta basado en la representación por vóxeles [Wikipedia (2022)] (un vóxel representa un valor en una malla de tres dimensiones) en este nos encontramos con el proyecto conocido como BodyNet [Varol y cols. (2018)], Varol propone una red neuronal entrenable de extremo a extremo enfocada en la inferencia directa de la forma volumétrica del cuerpo a partir de una sola imagen. Esta red genera probabilidades en el campo de ocupación 3D de la malla de una persona, dado que se usa una sola vista hay existen pérdidas de información que tratan de solventar aproximando el espacio de vóxeles, aumentando la importancia de los vóxeles que se encuentran en la frontera del cuerpo de la persona y el resto de imagen.

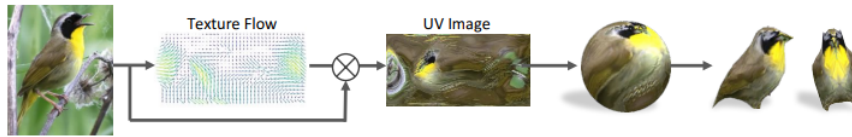
También hay proyectos como el que se explicará detalladamente en los siguientes puntos como PIFu[Saito y cols. (2019)] que consta de una función que alinea localmente píxeles de imágenes 2D con el contexto global de su objeto 3D correspondiente, para ello se propuso el uso de una red neuronal profundo de extremo a extremo la cual es capaz de representar a humanos de manera que tanto el pelo como la ropa se ve detallada, que es uno de los puntos débiles de los proyectos mencionados con anterioridad, dado que estos [Bogo y cols. (2016), H. Zhang y cols. (2021)] realmente no representan la ropa ni el pelo, tan solo la pose y el cuerpo es una representación del cuerpo humano



**Figura 1.5:** Arquitectura general de la propuesta BodyNet[Varol y cols. (2018)]. Donde comienza a partir de una imagen RGB que se le pasa a dos subredes donde estiman tanto la pose como la segmentación 2D, con estas predicciones y la imagen se utilizan como entrada para la siguiente red, que estima en este caso la pose 3D, con todo lo anterior se utiliza como entrada en la siguiente red donde obtiene ya la forma del cuerpo, donde ya por último se ajusta con el modelo SMPL[Loper y cols. (2015)]

desnudo, a diferencia de el proyecto llevado a cabo por Fuster-Guilló [Fuster-Guilló y cols. (2020), García-D'Urso y cols. (2021)], este proyecto si que es capaz de representar la ropa.

Aparte de la reconstrucción del cuerpo 3D, existe otro punto esencial que es la obtención de la textura del modelo 3D, por este lado podemos observar que mientras realiza PIFu[Saito y cols. (2019)] una estimación directa de los colores RGB mediante una función implícita, también existen más maneras de representar la textura como la parametrización de la superficie como por ejemplo [Kanazawa y cols. (2018), Güler y cols. (2018)] al igual que en el PIFu[Saito y cols. (2019)] que aparte de la representación de la textura en un objeto 3D, representan dicho objeto 3D, Kanazawa propone



**Figura 1.6:** Ciclo de obtención de textura[Kanazawa y cols. (2018)]

Como podemos ver en la figura 1.6, se predice el flujo de textura  $F$  que es usado como entrada a la vez que la propia imagen inicial  $I$  para generar la imagen de la

textura  $I^{uv}$  esta imagen es a su vez utilizada para texturizar la malla obtenida en la reconstrucción del modelo 3D.

Güler en su propuesta reconstruye una malla del cuerpo humano, una vez la tiene segmenta esta malla en diferentes partes, de esta malla obtienen las coordenada que luego son utilizadas para renderizar un píxel de manera que se compara directamente la posición verdadera y luego mapean directamente una textura RGB que han utilizado de [Varol y cols. (2017)] a las coordenadas corporales UV estimadas.

Con esta investigación del estado del arte podemos observar que hay diferentes maneras de realizar la reconstrucción del cuerpo humano, dentro de cada manera hay varios trabajos parecidos dado que unos se retroalimentan de otros, también hemos visto de que manera obtienen la textura este tipo de métodos donde a partir de una imagen reconstruyen un modelo 3D.

Los trabajos realizados mediante visión artificial y redes neuronales están más enfocados en general a las poses de las personas, dado que comenzó inicialmente en ello, pero lo que nosotros queremos y buscamos es que sean capaces de reconstruir de forma fiel el cuerpo dado que se busca usar esto en un ámbito dietético-nutricional de la obesidad, por lo que es muy importante que las medidas sean correctas.

## 1.3 Objetivos

- **Objetivo 1: Analizar métodos del estado del arte para la reconstrucción del modelo 3D del cuerpo humano a partir de imágenes 2D.**

Para ello se realizarán tareas como:

- **Objetivo 2: Uso de PIFu[Saito y cols. (2019)] para la obtención del modelo 3D y textura a partir de una vista 2D de la persona**

Para este objetivo se tendrán que realizar las siguientes tareas específicas:

- Estudio de los diferentes parámetros para el entrenamiento y prueba de la red propuesta en PIFu.
  - Obtención de imagen del cuerpo y generación de máscara de la imagen.
  - Obtención y estudio de los resultados.
- **Objetivo 3: Análisis comparativo de la precisión de las medidas obtenidas a partir de modelos 3D del cuerpo humano**
- Estudio sobre las diferentes formas de realizar el cálculo de las diferencias obtenidas.
  - Alineamiento de objetos 3D.
  - Cálculo de distancias usando Hausdorff Distance
  - Cálculo de distancias usando Chamfer distance
-

## 2 Antecedentes Metodológicos y Tecnológicos

En el siguiente capítulo, se presentan los métodos que son necesarios conocer para el planteamiento apropiado del método usado para la reconstrucción del cuerpo humano 3D, que constituye el núcleo del presente trabajo. Primero, se detallará el algoritmo Marching Cubes, que es utilizado más tarde en PIFu que a su vez es el siguiente punto y por último veremos como se van a calcular las comparaciones entre dos modelos 3D diferentes.

### 2.1 Marching Cubes

Marching Cubes [Lorensen y Cline (1987)] es un algoritmo utilizado en la reconstrucción del cuerpo humano 3D que es usado en PIFu, este algoritmo genera mallas triangulares de densidad constante desde una función implícita.

Este algoritmo fue presentado por Lorensen y Cline, calcula los vértices usando un enfoque de divide y vencerás para aproximar la localización en un cubo creado por 8 píxeles, 4 por cada parte del cubo. El algoritmo determina como la superficie intersecta en un cubo y luego se mueve al siguiente cubo (on marches). Se utiliza una tabla de casos que define la tipología del triángulo.

Este algoritmo tiene 5 fases:

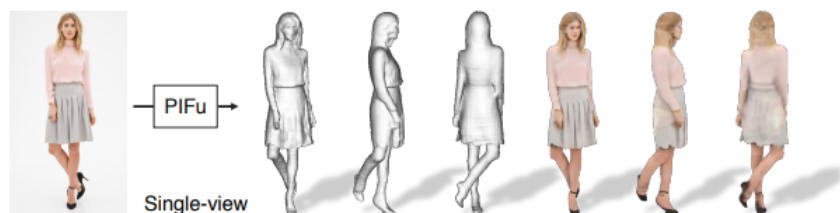
1. Determinación del índice caso de cada celda.

2. Determinación de los bordes intersecados.
3. Cálculo de intersecciones mediante interpolación lineal.
4. Triangulación de las intersecciones.
5. Cálculo de las superficies normales que apuntan hacia fuera.

Marching Cubes procesa cada celda de volumen de forma independiente.[ScienceDirect (s.f.)]

## 2.2 PIFu: Pixel-aligned Implicit function

Como se ha dicho antes en el capítulo 1.3 PIFu [Saito y cols. (2019)] alinea de manera local las características individuales a nivel de píxel con el contexto global de todo el objeto, esto lo hace de una manera totalmente convolucional y ayuda a no requerir un alto uso de memoria como en otro tipo de representaciones como las que se hacen por vóxel[Varol y cols. (2018)], esto es importante dado que PIFu es capaz de reconstruir aparte de la forma y pose del cuerpo humano, la ropa, el cabello y los accesorios; por lo tanto la reconstrucción de estos puede ser muy deformable, detallada y con una topología complicada.



**Figura 2.1:** Pixel-aligned Implicit function (PIFu) [Saito y cols. (2019)].

---

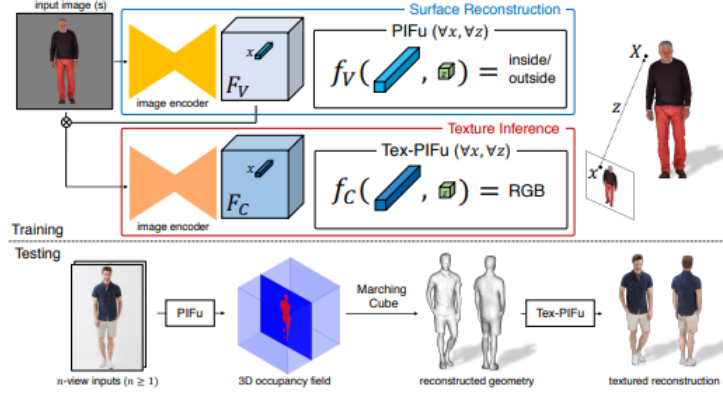
Inicialmente se entrena un encoder (codificador) que aprenda sobre vectores de características para cada píxel que existe en la imagen teniendo en cuenta el contexto global relativo a su posición, con este vector y una profundidad en el eje  $z$  especificada a lo largo del rayo de cámara saliente del píxel, este aprende a partir de una función implícita que puede clasificar si un punto 3D correspondiente a esta profundidad  $z$  está dentro o fuera de la superficie, que en nuestro caso es el cuerpo de una persona.

### 2.2.1 Reconstrucción de la superficie

Para realizar una buena reconstrucción de manera eficiente en PIFu utiliza una función implícita para definir así la superficie, esta función consiste en un codificador  $g$  de imágenes totalmente convolucional y una función  $f$  continua e implícita representada por una red MLP (multi-layer perceptrons), donde la superficie esta definida como un conjunto de nivel de:

$$f(F(x), z(X)) = s : s \in \mathbb{R} \quad (2.1)$$

Donde para un punto 3D  $X$ ,  $x = \pi(X)$  es su proyección 2D,  $z(X)$  es el valor de profundidad en el espacio de coordenadas de la cámara,  $F(x) = g(I(x))$  es la característica de la imagen en  $x$ . Se obtiene la función de alineamiento  $F(X)$  usando un muestreo bilineal, porque la proyección 2D de  $X$  se define en un espacio continuo en lugar de uno discreto (es decir, píxel). El punto principal de esta función implícita es que aprende a partir del espacio 3D con las características de los píxeles alineados en vez de aprender según las características globales, además esta función se puede presentar como un marco general que puede ser extendido a otras cosas, como por ejemplo predecir los colores RGB.



**Figura 2.2:** Pipeline usada en (PIFu) [Saito y cols. (2019)]. Dada una imagen, PIFu predice la probabilidad continua interior/exterior de un cuerpo humano vestido. Tex-PIFu infiere un valor RGB dados los puntos 3D de la superficie con con topología arbitraria

Para la reconstrucción de la superficie, se representa como un conjunto de nivel de 0,5 dentro de un campo continuo 3D:

$$f_v^* = \begin{cases} 1, & X \text{ esta dentro de la superficie de la malla} \\ 0, & \text{resto de casos} \end{cases} \quad (2.2)$$

Se entrena la función implícita que alinea píxeles  $f_v$  minimizando la media del mean squared error (MSE):

$$\mathcal{L}_V = \frac{1}{n} \sum_{i=1}^n |f_v(F_V(x_i)z(X_i)) - f_v^*(X_i)|^2 \quad (2.3)$$

Donde  $X_i \in \mathbb{R}^3$ ,  $F_V(x) = g(I(x))$  es la característica de la imagen del codificador de imágenes  $g$  en  $x = \pi(X)$  y  $n$  es el número de puntos muestrados. Dadas una imagen y la malla 3D correspondiente a la imagen que está espacialmente alineada con la imagen, los parámetros del codificador  $g$  y de PIFu  $f_v$  se actualizan unidas minimizando con la ecuación 2.3 como [Bansal y cols. (2017)] demuestra, el entrenamiento de un codificador de imágenes con un subconjunto de píxeles no le hace daño a la convergencia



en comparación a entrenarlo con todos los píxeles[Saito y cols. (2019)].

Durante la obtención de la superficie, se muestrea el campo de probabilidad sobre el espacio 3D y se extrae la iso-superficie el campo de probabilidad con un umbral de 0.5 utilizando Marching Cube.

Este método permite el muestreo directo de puntos 3D sobre la marcha desde la malla en la resolución original utilizando un algoritmo de trazado de rayos eficiente.

### 2.2.2 Obtención de la Textura

Para la obtención de la textura de la imagen para el modelo 3D, PIFu nos permite predecir de manera directa los colores RGB en la superficie  $s$  en la ecuación 2.4 como un vector con la información de los 3 valores RGB. Sin embargo, extender PIFu a la predicción de colores no es una tarea trivial, ya que los colores están solo definidos en la superficie a diferencia en la reconstrucción tenemos en cuenta todo el espacio 3D.

Dado unos puntos 3D muestreados en la superficie  $X \in \omega$ , la función objetivo para la inferencia de textura es el promedio de la función L1 error de los colores como se puede ver en la siguiente ecuación:

$$\mathcal{L}_C = \frac{1}{n} \sum_{i=1}^n |f_c(F_C(x_i)z(X_i)) - C(X_i)| \quad (2.4)$$

Donde  $C(X_i)$  es la verdad básica de los valores de los colores RGB en el punto de superficie ( $X_i \in \omega$  y  $n$  es el numero de puntos muestreados). El problema es que se observó que  $f_c$  con la función de pérdidas sufría overfitting, esto es provocado porque  $f_c$  no solo aprende los colores en la superficie sino que también las fronteras de las superficies 3D del objeto por eso  $f_c$  puede obtener la textura de las superficies que no se ven con una pose y forma diferente durante la obtención, lo que supone un gran desafío. Para solucionar este problema se realizaron varios cambios, el primero es que la condiciona el codificador de imágenes para la obtención de texturas con las características de la imagen aprendidas en la reconstrucción de la superficie  $F_V$ , con esto

el codificador puede enfocarse en la obtención del color dada una geometría incluso los objetos invisibles tienen diferentes formas, poses o topología. Por otro lado se introduce un offset  $\epsilon \sim \mathcal{N}(0, d)$  para los puntos de la superficie sobre la superficie normal  $N$  así el color puede ser definido no solo por la superficie exacta si no que por el alrededor de este. Con esas modificaciones la función queda así:

$$\mathcal{L}_C = \frac{1}{n} \sum_{i=1}^n |f_c(F_C(x'_i, FV), X'_{i,z}) - C(X_i)| \quad (2.5)$$

Donde  $X'_i = X_i + \epsilon$ . Se usa un  $d = 1.0$  cm para todos los experimentos.

## 2.3 Cálculo de distancias

Dado que se necesita saber cuán capaz es de representar la realidad este método se necesita realizar una serie de comparaciones, para ello vamos a utilizar dos métodos diferentes.

### 2.3.1 Distancia Chamfer

Por un lado tenemos la distancia de Chamfer es una métrica de evaluación, en este caso la aplicaremos sobre dos nubes de puntos (la obtenida por nuestro método y la del proyecto de Tech4diet[Tech4Diet (2019)]). Tiene en cuenta la distancia de cada punto. Para cada punto en cada nube, esta distancia calcula el punto más cercano en el otro conjunto de puntos y suma el cuadrado de la distancia. [LINYI (2022)]

$$CD(S_1, S_2) = \frac{1}{S_1} \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2^2 + \frac{1}{S_2} \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|_2^2 \quad (2.6)$$

Donde  $S_1$  y  $S_2$  son las dos nubes de puntos y  $x$  e  $y$  son sus correspondientes puntos.

---

### 2.3.2 Distancia Hausdorff

Por otro lado tenemos la distancia de Hausdorff que al igual que la de Chamfer es una métrica de evaluación, esta es la distancia más larga de un conjunto de puntos vecino en el otro conjunto [Grégoire y Bouillot (2022)].

$$h(S_1, S_2) = \max_{a \in S_1} [\min_{b \in S_2} [d(a, b)]] \quad (2.7)$$

Donde  $a$  y  $b$  son los puntos de los conjuntos  $S_1$  y  $S_2$  respectivamente y  $d(a, b)$  es cualquier métrica entre esos dos puntos, esta puede ser la distancia euclídea o cualquier otra.

## 2.4 Herramientas tecnológicas

En este punto se detallan las aplicaciones utilizadas para realizar este trabajo.

### 2.4.1 GIMP

GIMP (GNU Image Manipulation Program)[The GIMP Development Team (s.f.)] es una aplicación de imágenes digitales en forma de mapa de bits. Esta aplicación se trata de un software de código abierto y gratuito, englobado dentro del proyecto GNU y disponible bajo licencia pública general de GNU y GNU Lesser General Public License

Este programa ha sido utilizado para crear archivos con unas medidas concretas, escalar imágenes y recortarlas, para el proceso de generación de máscara comentado en el punto 3.

### 2.4.2 MeshLab

MeshLab [Cignoni y cols. (2008)] es un sistema de software de procesamiento de mallas 3D, este software al igual que GIMP es libre (de código abierto) y gratuito y disponible bajo la licencia pública general de GNU.

---

Con MeshLab hemos podido ver los modelos obtenidos por PIFu con la textura inferida y hemos utilizado sus herramientas para poder calcular la distancia de Hausdorff y para exportar los modelos como nubes de puntos, utilizadas mas tarde para calcular la distancia de Chamfer.

### **2.4.3 Blender**

Blender [Community (2018)] es un programa informático multiplataforma, dedicado especialmente al modelado, iluminación, renderizado, animación y creación de gráficos 3D. Este programa fue inicialmente distribuido de manera gratuita, pero no constaba como software libre, aunque tiempo después pasó a ser software libre.

Se ha utilizado Blender para el preprocesado de los modelos 3D.

---

## 3 Reconstrucción del modelo 3D y textura a partir de una imagen

En este capítulo se va a detallar como se consigue la reconstrucción 3D del cuerpo humano a partir de una imagen utilizando el método PIFu. El proceso se puede dividir en varios puntos:

- Estudio de los parámetros de test
- Generación de la máscara de una imagen
- Obtención del modelo 3D

### 3.1 Estudio de los parámetros de test

Lo primero que se tiene que realizar es la instalación del proyecto PIFu[Saito y cols. (2019)] con sus respectivos requerimientos, una vez inicializado, se analizó el proyecto para poder comenzar con la experimentación y obtener modelos. Para ello se estudió tanto la fase test como la fase de entrenamiento, donde en este caso solo se ha utilizado la fase de test ya que es la fase que genera los modelos 3D. Con este estudio se puede observar las necesidades de la red para su correcto comportamiento, en este caso se observó que la red necesita en realidad de 2 imágenes, una de ellas la imagen de la que se busca hacer la representación 3D y la otra es la máscara de esta imagen. Por otro lado, nos dimos cuenta de que las imágenes han de tener el mismo tamaño y que los

tamaños donde mejor trabaja la red es sobre 512x512 píxeles y 1024x1024 píxeles. Una vez realizado el estudio se puede comenzar a abarcar los siguientes puntos.

## 3.2 Enmascaramiento de imágenes

Este proceso nos sirve para quitar información de la imagen inicial, PIFu por dentro realiza la eliminación del fondo, para ello utiliza la máscara de la imagen. Donde en la imagen dada, los píxeles que tengan el valor de  $[255, 255, 255]$  mantendrá esa información ya que es útil para la red porque es la persona, y los píxeles donde el color sea  $[0, 0, 0]$  será información descartada por la red y se eliminará de la imagen, entendiendo que es el fondo de esta.

En este punto se va a explicar como se ha llegado a obtener la máscara de una imagen.

### 3.2.1 Enmascaramiento de imágenes con el fondo de un sólo color

Dado que se necesita aparte de la imagen de la persona, la máscara de esta se tuvo que realizar un método que fuese capaz de este proceso, utilizando como base la imagen que se le pasa a la red.

Al inicio se comenzó con imágenes de prueba con fondos blancos o sin fondos, por lo que la extracción es sencilla, todo lo blanco se cambia a negro y el resto de colores diferentes al blanco se cambian al blanco, esto es trivial en python dado que con dos asignaciones obtenemos los resultados que se buscan.

---

Código 3.1: Código obtención máscara 1

```
1  import numpy as np
2  import cv2
3
4  red = [0, 0, 255]
5  white = [255, 255, 255]
6  black = [0, 0, 0]
7  maskpath = 'img_mask.png'
8  img = cv2.imread('img.png', cv2.COLOR_RGB2BGR)
9  img[np.where((img==red).all(axis=2))] = black
10 img[np.where((img!=red).all(axis=2))] = white
11 cv2.imwrite(maskpath, img)
```

El código 3.1 es capaz de cambiar todo lo rojo a negro y el resto de los colores a blanco. Con esto se pudo comprobar el funcionamiento de la red utilizando imágenes de prueba.

### 3.2.2 Enmascaramiento de fotos realizadas con teléfonos móviles.

El punto anterior nos sirvió para comprobar el correcto funcionamiento de la red, pero esto no sirve dado que se busca realizar comparaciones y analizar las métricas de los modelos utilizando el proyecto Tech4diet[Tech4Diet (2019)], para ello se hicieron una serie de fotos, donde se usaron dos cámaras diferentes de dos teléfonos móviles diferentes y el uso de una tela verde para facilitar el proceso de enmascaramiento.

Modelo	SM-A528B	Mi 9 SE
Número Cámaras	4	3
Cámara Principal	64 megapíxeles	48 megapíxeles
Cámara ultra gran angular	12 megapíxeles	13 megapíxeles
Cámara telefoto	-	8 megapíxeles
Cámara macro	5 megapíxeles	-
Cámara profundidad	5 megapíxeles	-
Resolución	10120x6328 píxeles	8000x6000 píxeles
Apertura focal	f/1.8	f/1.8
Tamaño del sensor	1/1.72" pulgadas	1/2" pulgadas

**Tabla 3.1:** Parámetros de las cámaras elegidas.

Como se puede observar son cámaras con mucha resolución ambas por lo tanto a la hora de hacer las fotos no se encontró ningún problema.

### 3.2.2.1 Preprocesado de las fotos

Una vez hechas las fotos, estas tuvieron que pasar por un preprocesado antes del enmascaramiento y sucesivamente de usarlas en la red. Para este preprocesado se utilizó la herramienta GIMP. El proceso que se hizo es el siguiente:

1. Recorte con la información necesaria.
2. Escalar imagen.
3. Creación de un nuevo archivo 1024x1024 con fondo verde y pegar la imagen obtenida después de escalarla.
4. Retocar colores.

Donde el primer punto se hizo porque como se puede ver en la figura 3.1 en la imagen inicial de la que se parte, la tela verde no ocupa toda la imagen y se busca mantener ese fondo verde. El segundo punto, es el escalado, pues las imágenes parten de un tamaño de 3468x4624 píxeles, aún habiendo sido recortadas, el tamaño era superior a 1024, por lo que se escala de manera que se cumpla con dicho tamaño. Una vez hecho esto, se

---



crea un archivo con las medidas necesarias para la red, con el fondo verde y se coloca la imagen después de su escalado. En ocasiones nos hemos encontrado con errores a la hora del proceso de enmascarado, esto es porque la tela deja pasar la luz y se refleja en el cuerpo, para ello utilizamos un pincel con poca opacidad del color suplementario, en este caso es un color morado, se pinta por encima y con eso se contrarresta la luz verde.

Ya teniendo la imagen como se necesita, se realiza el proceso de generación de máscara, para ello se modificó el código 3.1, donde ya no se quiere quitar un color en concreto, si no que en este caso es un rango de color, en este caso se quiere reconocer todos los tipos de verdes posibles.

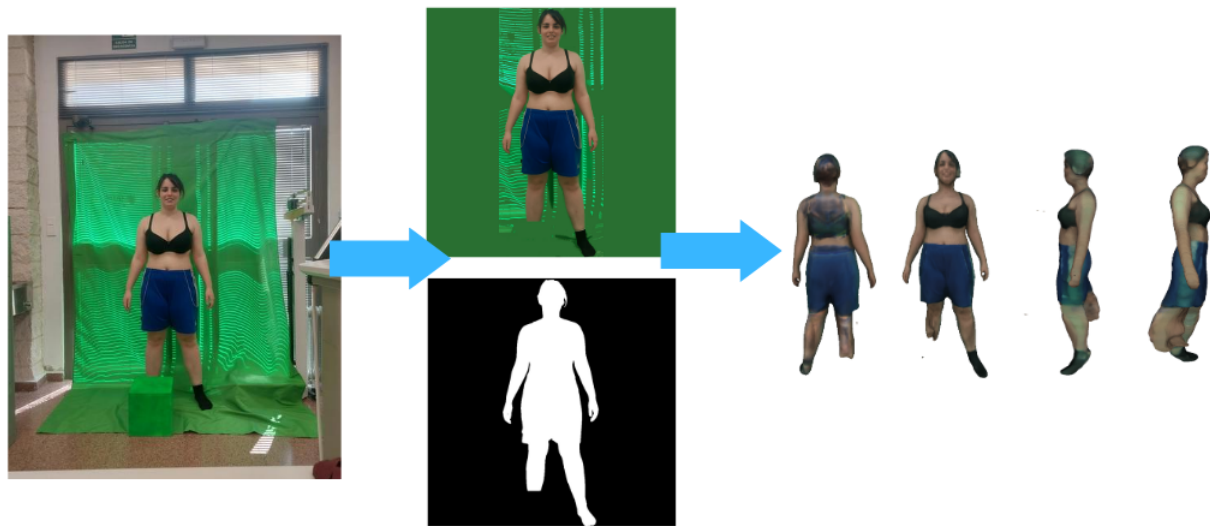
Código 3.2: Código obtención máscara 2

```
1  import numpy as np
2  import cv2
3
4  maskpath = 'img_mask.png'
5  img = cv2.imread('img.png')
6  hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
7  lower_green = np.array([36, 25, 25])
8  upper_green = np.array([70, 255, 255])
9  mask = cv2.inRange(hsv, lower_green, upper_green)
10 result = cv2.bitwise_and(frame, frame, mask=mask)
11 b, g, r = cv2.split(result)
12 filter = g.copy()
13 ret,mask = cv2.threshold(filter,10,255, 1)
14 cv2.imwrite(maskpath, maskpath)
```

En ambos códigos 3.1 y 3.2 se inicializa la imagen y también los colores que se quieren reconocer, y después se realiza el proceso donde se va a generar la máscara, en el código 3.2 se utiliza el modelo HSV (Hue, Saturation, Brightness - Matiz, Saturación, Brillo) para el rango de color verde.

### 3.3 Obtención del modelo 3D

La obtención del modelo 3D una vez tenemos la máscara de la imagen es trivial, pues tan solo hay que utilizar la red de PIFu con la máscara y su imagen respectiva.



**Figura 3.1:** Pipeline usada en para la obtención del modelo. De izquierda a derecha, dada una imagen  $I$ , se escala, recorta y se ajustan los colores generando la imagen  $i$  y se genera la máscara de la imagen  $m_i$  a partir de  $i$ . La red de PIFu utilizará las imágenes  $i$  y  $m_i$  para generar el modelo que se ve a la derecha.

La figura 3.1 detalla el proceso utilizado, en este caso quisimos comprobar como reacciona la red añadiendo ruido a la imagen, en este caso el ruido es el cubo verde, de manera que falta una pierna.

En el apartado 4 se explica el estudio de los modelos, donde se pueden ver más ejemplos de modelos 3D obtenido por la red.

---

## 4 Experimentación

En este punto del trabajo se va a mostrar toda la experimentación realizada en este proyecto con la red, para ello se va a comenzar comentando los arreglos que se hicieron necesarios para poder realizar la comparación de los modelos.

### 4.1 Preprocesado de los modelos de Tech4diet

Los modelos generados por Tech4diet, tienen suciedad en la parte de los pies, pues representan la plataforma que pisas para hacerte la reconstrucción del cuerpo.

Este proceso fue necesario para tratar de tener una mejor comparativa de modelos obtenidos por PIFu, dado que hasta la alineación era compleja por este motivo. El proceso se realizó con la herramienta de Blender, se eliminaron las partes sobrantes como se puede ver en la siguiente imagen.



**Figura 4.1:** Proceso limpieza modelo Tech4diet

Como se puede ver en la figura 4.1 se ha limpiado la plataforma, que no forma parte del cuerpo humano.

## 4.2 Alineamiento de modelos

Para poder calcular la distancia de Hausdorff se necesita que los modelos estén alineados, dado que se va a calcular en MeshLab, este proceso también se va a realizar en este software.

Este proceso se hace mediante la herramienta *Align*, y con esta herramienta aparece una ventana donde nos aparecen diferentes opciones, lo primero que haremos será seleccionar el modelo obtenido por Tech4diet, dado que es el modelo que se usa para comprobar los resultados obtenidos y se elige la opción *GlueMeshHere*, esto lo que hace es que la malla se va a quedar fija en el lugar y la otra malla es la que se va a alinear con esta. Para poder alinearla se selecciona la opción *PointBasedGlueing*.

Una vez elegida la opción para alinear, aparece otra ventana donde salen los dos modelos como se puede ver en la figura 4.2, aquí se eligen punto a punto en ambas mallas y a la vez seleccionamos la opción de escalado para que ambos modelos tengan el mismo tamaño.

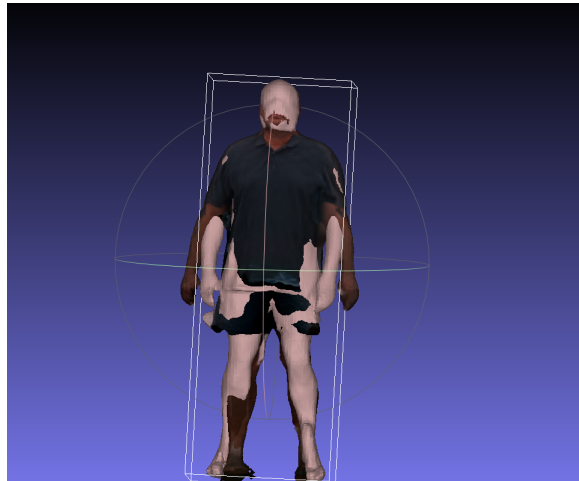


**Figura 4.2:** Proceso alineamiento modelos

En la figura 4.2 podemos observar dos modelos, el de la izquierda es el obtenido

mediante la red, el de la derecha es el obtenido por Tech4diet

Una vez seleccionados los puntos correctamente y seleccionado *AllowScaling* (esta opción se elige porque la red siempre devuelve un modelo del mismo tamaño, ya que está normalizado), se realiza el alineamiento.



**Figura 4.3:** Modelos alineados

En la figura 4.3 se observa el resultado obtenido después de la alineación.

Con la alineación realizada ya se puede usar la herramienta que nos calcula la distancia de Hausdorff.

### 4.3 Creación de las nubes de punto correspondientes a cada modelo

Para poder realizar el cálculo de la distancia de Chamfer es necesario tener las nubes de punto, porque las nubes de punto son *arrays* con las coordenadas (también tienen la información del color y las normales, pero en este caso se va a eliminar esta información). Para ello se puede usar las aplicaciones MeshLab o CloudComparer.

He seleccionado CloudComparer, porque podías eliminar la información de los colores y de las normales desde la misma herramienta.

## 4.4 Cálculo distancias

Para poder comparar los modelos se han utilizado las distancias de Hausdorff y Chamfer como se ha mencionado con anterioridad, en este punto se va a explicar como se han obtenido las medidas.

### 4.4.1 Distancia Hausdorff

Como se ha comentado en el punto 4.1, la distancia de Hausdorff se ha calculado mediante la herramienta de MeshLab que te permite calcularla, esta te da un resultado parecido al siguiente:

```
1  "Hausdorff Distance computed
2  Sampled 58888 pts (rng: 0) on result_camelia0.obj searched closest on Camelia↔
                                     ↪ .obj
3  min : 0.000000 max 0.339742 mean : 0.020914 RMS : 0.029951
4  Values w.r.t. BBox Diag (1.819615)
5  min : 0.000000 max 0.186711 mean : 0.011493 RMS : 0.016460"
```

Donde *min* significa la mínima distancia que hay entre las mallas, *max* la máxima distancia existente de un punto a otro de las mallas, *mean* es la media y *RMS* significa Root Mean Square, que es la raíz cuadrada de la media aritmética de los cuadrados de los valores.

Tenemos dos líneas de valores en la primera esta sobre la unidad de medida que en este caso es en metros, y la segunda línea son los valores anteriores obtenidos pero divididos por la longitud de la diagonal de la bounding box (cuadro delimitador de las mallas) de la malla que se usa como referencia, en este caso usamos la malla obtenida por Tech4diet.

---

#### 4.4.1.1 Quality Mapper

También se ha usado la herramienta *QualityMapper* para observar donde se encuentran las mayores diferencias entre las dos mallas.

Esta herramienta colorea la malla en un gradiente, donde el color rojo significa que es muy distante de la malla y contra más azul oscuro más se acerca una malla a la otra.

Esta herramienta nos ha permitido poder analizar los modelos desde otro punto de vista, pero para poder utilizarlo se ha realizado antes el cálculo de la distancia de Hausdorff.

AÑADIR ANGULO IZQUIERDA DELANTE ANDRES COMPARATIVA Y AÑADIR ANGULO ESPALDA ANDRES COMPARATIVA

---

### 4.4.2 Distancia Chamfer

Por otro tenemos la distancia Chamfer, para ello se ha programado utilizando la siguiente función en python:

Código 4.1: Código obtención distancia chamfer

```
1 def chamfer_distance(x, y, metric='l2'):  
2     x_nn =NearestNeighbors(n_neighbors=1, leaf_size=1, algorithm='kd_tree', ↵  
                           ↵ metric=metric).fit(x)  
3     min_y_to_x =x_nn.kneighbors(y)[0]  
4     y_nn =NearestNeighbors(n_neighbors=1, leaf_size=1, algorithm='kd_tree', ↵  
                           ↵ metric=metric).fit(y)  
5     min_x_to_y =y_nn.kneighbors(x)[0]  
6     chamfer_dist =np.mean(min_y_to_x) +np.mean(min_x_to_y)  
7     return chamfer_dist
```

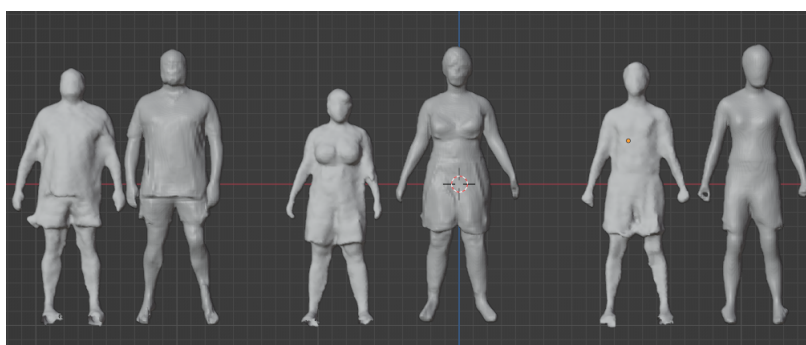
En el código 4.1 se calcula la distancia media mínima para cada punto que hay de la malla  $x$  a la  $y$  y viceversa.

---

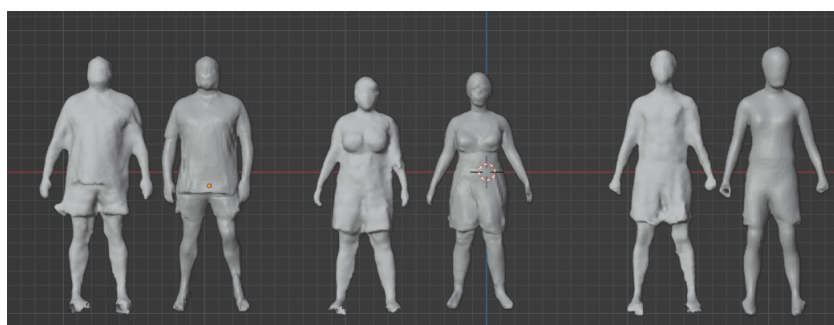


## 4.5 Análisis y comparativas de los modelos

Por un lado comentar que la red genera modelos del mismo tamaño sin importar la altura de la persona, a la hora del cálculo de la distancia de Hausdorff se soluciona porque permite el propio escalado la aplicación, para la distancia de Chamfer se ha escalado debidamente cada modelo obtenido por PIFu para que tengan los mismos tamaños.



**Figura 4.4:** Diferencia de altura de los modelos, el modelo de la derecha de cada pareja es el generado por la red



**Figura 4.5:** Diferencia de altura de los modelos, ya escalados

Con los modelos escalados, y con todos los procesos explicados anteriormente pasamos a realizar todos los cálculos con diferentes modelos 3D.

Una vez los cálculos hechos



## 5 Conclusiones

En este capítulo se resaltan las principales conclusiones obtenidas como consecuencia del trabajo desarrollado y a su vez líneas futuras derivadas de este.

### 5.1 Conclusión

En el presente trabajo se han planteado una serie de objetivos relacionados con la necesidad de poder ser capaces de generar un modelo 3D del cuerpo humano con el uso de una cámara de teléfono móvil y con una sola vista.

Se ha seleccionado el uso de una cámara de teléfono móvil porque actualmente toda persona adulta tiene uno, y por lo tanto es un sistema muy flexible y portable.

En relación al estudio de la red profunda que se ha usado, hemos podido comprobar que ha sido alimentada y entrenada por modelos que tienen un físico más normativo, dado que cada vez que se ha probado con un cuerpo menos normativo más errores nos hemos encontrado en el proceso, como por ejemplo una pérdida de peso considerable en comparación cuando observabas los modelos en ángulos másladeados.

En relación al estudio de los resultados obtenidos se concluye con que hay ciertos ángulos que favorecen a la red, hemos podido comprobar que el ángulo donde se ve la espalda nos ha proporcionado modelos con menos problemas. Se ha repetido en la mayoría de las ocasiones que la vista del modelo 3D se ve perfecta desde el ángulo de la imagen pero cuando empiezas a mirar el resto de ángulos se ven muchos desperfectos y deformidades.

## 5.2 Líneas Futuras

Como consecuencia del trabajo desarrollado se han derivado una serie de líneas futuras entre las que podemos destacar:

Ampliación del uso de la red a multivista, ya que como se ha comentado en las conclusiones, se puede observar que el ángulo utilizado en la imagen 2D, se representa bien en el modelo 3D, el problema ocurre una vez se observa desde más ángulos, esto se puede solucionar dándole a la red más información con diferentes ángulos. Además PIFu es exportable a multivista, por lo que facilita este trabajo.

Entrenamiento con diferentes modelos sobre diferentes tipos de cuerpos humanos para obtener mejores resultados, y prepararlo para multivista, para ello se necesitará un amplio DataSet.

Por último realizar comparaciones de los resultados obtenidos con multivista y una vista.

---

# Bibliografía

- Bansal, A., Chen, X., Russell, B., Gupta, A., y Ramanan, D. (2017). *Pixelnet: Representation of the pixels, by the pixels, and for the pixels*. arXiv. Descargado de <https://arxiv.org/abs/1702.06506> doi: 10.48550/ARXIV.1702.06506
- Bogo, F., Kanazawa, A., Lassner, C., Gehler, P. V., Romero, J., y Black, M. J. (2016). Keep it SMPL: automatic estimation of 3d human pose and shape from a single image. *CoRR*, *abs/1607.08128*. Descargado de <http://arxiv.org/abs/1607.08128>
- Cignoni, P., Callieri, M., Corsini, M., Dellepiane, M., Ganovelli, F., y Ranzuglia, G. (2008). MeshLab: an Open-Source Mesh Processing Tool. En V. Scarano, R. D. Chianara, y U. Erra (Eds.), *Eurographics italian chapter conference*. The Eurographics Association. doi: 10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136
- Community, B. O. (2018). Blender - a 3d modelling and rendering package [Manual de software informático]. Stichting Blender Foundation, Amsterdam. Descargado de <http://www.blender.org>
- Fuster-Guilló, A., Jorge Azorín-López, M. S.-C., Castillo-Zaragoza, J. M., García-D'Urso, N., y Fisher, R. B. (2020). RGB-D based framework to Acquire, Visualize and Measure the Human Body for Dietetic Treatments..
- García-D'Urso, N., Galán-Cuenca, A., Manchón-Pernis, C., Fuster-Guilló, A., y Azorín-López, J. (2021). Arquitectura de visión 3D para medición y visualización del cuerpo humano..

- Grégoire, N., y Bouillot, M. (2022). *Hausdorff distance*. Descargado de <http://cgm.cs.mcgill.ca/~godfried/teaching/cg-projects/98/normand/main.html>
- Güler, R. A., Neverova, N., y Kokkinos, I. (2018). *Densepose: Dense human pose estimation in the wild*. arXiv. Descargado de <https://arxiv.org/abs/1802.00434> doi: 10.48550/ARXIV.1802.00434
- Johnson, S., y Everingham, M. (2010). Clustered pose and nonlinear appearance models for human pose estimation. En *Proceedings of the british machine vision conference* (pp. 12.1–12.11). BMVA Press. (doi:10.5244/C.24.12)
- Kanazawa, A., Black, M. J., Jacobs, D. W., y Malik, J. (2017). End-to-end recovery of human shape and pose. *CoRR*, *abs/1712.06584*. Descargado de <http://arxiv.org/abs/1712.06584>
- Kanazawa, A., Tulsiani, S., Efros, A. A., y Malik, J. (2018). *Learning category-specific mesh reconstruction from image collections*. arXiv. Descargado de <https://arxiv.org/abs/1803.07549> doi: 10.48550/ARXIV.1803.07549
- LINYI, J. (2022). *Github - um-arm-lab/chamfer-distance-api: A python class that calculates chamfer distance between point clouds using tensorflow*. Descargado de <https://github.com/UM-ARM-Lab/Chamfer-Distance-API>
- Loper, M., Mahmood, N., Romero, J., Pons-Moll, G., y Black, M. J. (2015). Smpl: a skinned multi-person linear model. *ACM Trans. Graph.*, *34*, 248:1-248:16.
- Lorensen, W. E., y Cline, H. E. (1987). Marching cubes: A high resolution 3d surface construction algorithm. En *Proceedings of the 14th annual conference on computer graphics and interactive techniques* (p. 163–169). New York, NY, USA: Association for Computing Machinery. Descargado de <https://doi.org/10.1145/37401.37422> doi: 10.1145/37401.37422
-

- 
- Natsume, R., Saito, S., Huang, Z., Chen, W., Ma, C., Li, H., y Morishima, S. (2019). *Siclope: Silhouette-based clothed people*. arXiv. Descargado de <https://arxiv.org/abs/1901.00049> doi: 10.48550/ARXIV.1901.00049
- Pishchulin, L., Insaftudinov, E., Tang, S., Andres, B., Andriluka, M., Gehler, P. V., y Schiele, B. (2015). Deepcut: Joint subset partition and labeling for multi person pose estimation. *CoRR*, *abs/1511.06645*. Descargado de <http://arxiv.org/abs/1511.06645>
- Saito, S., Huang, Z., Natsume, R., Morishima, S., Kanazawa, A., y Li, H. (2019). Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. *CoRR*, *abs/1905.05172*. Descargado de <http://arxiv.org/abs/1905.05172>
- Saito, S., Simon, T., Saragih, J., y Joo, H. (2020). PIFuHD: Multi-Level Pixel-Aligned Implicit Function for High-Resolution 3D Human Digitization..
- ScienceDirect. (s.f.). *Marching cube algorithm*, *sciencedirect*.
- Tech4Diet. (2019). Project tin2017-89069-r spanish state research agency (aei). 4d modelling and visualization of the human body to improve adherence to dietetic-nutritional intervention of obesity.. (<http://tech4d.dtic.ua.es/tech4d/> (accessed Jun. 2022))
- The GIMP Development Team. (s.f.). *Gimp*. Descargado de <https://www.gimp.org>
- Varol, G., Ceylan, D., Russell, B., Yang, J., Yumer, E., Laptev, I., y Schmid, C. (2018). *Bodynet: Volumetric inference of 3d human body shapes*. arXiv. Descargado de <https://arxiv.org/abs/1804.04875> doi: 10.48550/ARXIV.1804.04875
- Varol, G., Romero, J., Martin, X., Mahmood, N., Black, M. J., Laptev, I., y Schmid, C. (2017, jul). Learning from synthetic humans. En *2017 IEEE conference on computer vision and pattern recognition (CVPR)*. IEEE. Descargado de <https://doi.org/10.1109/CVPR.2017.492> doi: 10.1109/cvpr.2017.492
-

- Vodafone. (2020). Vodafone campus lab, el programa que te enseña a desarrollar soluciones a grandes problemas sociales en 4 meses, con la ayuda de los mejores expertos de vodafone.. (  
<https://vodafonecampuslab.es> (accessed Jun. 2022))
- Wikipedia. (2022). *Voxel — Wikipedia, the free encyclopedia*. <http://en.wikipedia.org/w/index.php?title=Voxel&oldid=1095329935>. ([Online; accessed 28-June-2022])
- Zhang, H., Tian, Y., Zhou, X., Ouyang, W., Liu, Y., Wang, L., y Sun, Z. (2021). 3d human pose and shape regression with pyramidal mesh alignment feedback loop. *CoRR*, *abs/2103.16507*. Descargado de <https://arxiv.org/abs/2103.16507>
- Zhang, Z. (2016). Camera calibration: a personal retrospective. *Machine Vision and Applications*.. (  
<https://doi.org/10.1007/s00138-016-0809-z>)
-