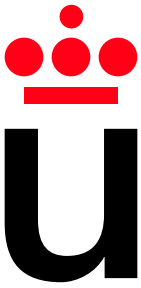


Universidad
Rey Juan Carlos

Escuela Técnica Superior
de Ingeniería Informática

Grado en Diseño y Desarrollo de Videojuegos

Curso 2022-2023

Trabajo Fin de Grado

SISTEMA DE CAPTURA 3D PARA HUMANOS BASADO EN 3DPISCAN

Autor: Álvaro Olavarría Bezanilla
Tutor: Jorge López Moreno

Agradecimientos

En primer lugar, me gustaría agradecer a mi tutor, Jorge López Moreno, por su inestimable ayuda y dedicación, durante estos meses de trabajo. También quiero agradecer a la empresa SEDDI, por los recursos técnicos y humanos que me ha facilitado para la construcción del escáner.

Índice de contenidos

Índice de figuras

1 Introducción

La captura y reconstrucción tridimensional del cuerpo humano ha ganado un interés significativo en diversos campos, como el entretenimiento, la medicina y la moda. Estas tecnologías permiten obtener modelos digitales precisos del cuerpo humano, lo cual brinda numerosas aplicaciones, desde la visualización y análisis del cuerpo hasta su implementación en servicios de prueba virtual de ropa. Sin embargo, muchos de los métodos existentes para la captura y reconstrucción del cuerpo humano requieren equipos costosos y un procesamiento complejo de datos.

En este contexto, el presente trabajo se centra en el desarrollo de un escáner basado en el proyecto Pi3DScan [1], el cual utiliza múltiples Raspberry Pi simultáneamente sincronizadas para controlar la iluminación y capturar el avatar del humano, añadiendo la posibilidad de obtener múltiples iluminaciones en un intervalo ínfimo de tiempo. Este sistema multi-vista emplea fotografías tomadas por las Raspberry Pi y las transfiere a un servidor, para iniciar de forma automática el proceso de reconstrucción de una nube de puntos "sparse" y "dense", almacenando en el proceso para cada punto de la nube de puntos "sparse" sus coordenadas (x,y) de las imágenes de las que proviene. La nube de puntos densa es procesada para extraer únicamente la forma del humano escaneado, eliminando aquellos elementos indeseados en la reconstrucción mediante el uso de un modelo preentrenado para la detección de humanos. Finalmente, se realiza una alineación precisa de la nube de puntos con el eje de coordenadas mediante un algoritmo propio y se optimiza a partir de esta un modelo SMPL [2, 3] preparado para ser texturizado a partir de la proyección de los conjuntos de imágenes con diferentes iluminaciones obtenidos en la captura.

Asimismo, se ha desarrollado una aplicación con un visualizador 3D y una in-

terfaz que permite a cualquier persona, incluso sin conocimientos especializados en el campo, utilizar el escáner. La aplicación ofrece funciones para el mantenimiento y actualización del software del escáner, control del pipeline de reconstrucción y diferentes formas de visualizar los resultados generados en el proceso.

En comparación con los enfoques existentes en el campo de la reconstrucción tridimensional del cuerpo humano, este proyecto presenta una solución completamente automatizada de principio a fin evitando la necesidad de ajustes manuales en alguna parte del proceso.

En resumen, este trabajo presenta el desarrollo de un escáner basado en Pi3DScan que automatiza el proceso de captura y reconstrucción tridimensional del cuerpo humano, capturado con diferentes iluminaciones en un breve periodo de tiempo. La aplicación diseñada permite que cualquier persona pueda utilizar el escáner sin conocimientos especializados y obtener un modelo SMPL preparado para su futuro texturizado a partir de las diferentes iluminaciones obtenidas. Los puntos trabajados en este proyecto son los siguientes:

- Preparación de la estructura, dispositivos y el cableado del escáner.
- Preparación de las Raspberry Pi para recibir ordenes del servidor.
- Configuración del arranque automático de las Raspberry Pi.
- Optimización de la velocidad de captura de imágenes.
- Programación de las tiras LED y proyectores.
- Automatización de la reconstrucción de la nube de puntos a partir de imágenes.
- Automatización en la limpieza y alineamiento de la nube de puntos resultante en la reconstrucción.
- Optimización de la nube de puntos refinada a un modelo SMPL.
- Creación una interfaz sencilla para utilizar las funcionalidades del escáner y la reconstrucción independiente de los conocimientos en el campo.

2

Trabajos previos

2.1. Escáner para captura de avatares

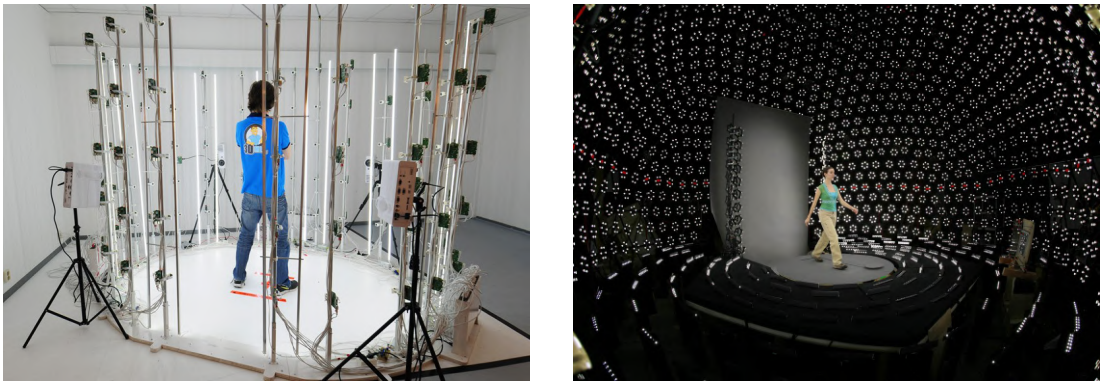
La fotogrametría es una técnica utilizada para obtener información tridimensional de objetos y escenas del mundo real a través de imágenes fotográficas. Consiste en el proceso de capturar y analizar fotografías desde diferentes ángulos con el objetivo de reconstruir la geometría y las características visuales de los objetos o entornos escaneados. Mediante esta técnica, se puede obtener información precisa sobre la forma, tamaño, posición relativa y texturas de los objetos capturados. Esto se logra al analizar la geometría y la disposición de los puntos clave presentes en las imágenes, así como los patrones de luz y sombra que se observan en ellas. "Multiple View Geometry in Computer Vision"[4] es un libro de referencia ampliamente utilizado en el campo de la visión por computador, que aborda los fundamentos matemáticos y algoritmos necesarios para comprender y aplicar la geometría de múltiples vistas en la visión por ordenador.

En el contexto de este trabajo, a diferencia de la captura del avatar mediante el uso de una cámara individual, el uso de un escáneres multi-vistas permite obtener en un intervalo corto de tiempo, múltiples conjuntos de imágenes con una iluminación personalizada sobre el sujeto humano escaneado evitando en gran medida el movimiento inconsciente del humano en ese lapso de tiempo, dando una precisión mayor en los resultados futuros. En la búsqueda de soluciones para la creación de escáneres 3D, existen algunas alternativas como lo son los proyectos de Pi3DScan [1] y Lightstage [5, 6].

Por un lado, el proyecto de Pi3DScan se basa en la construcción de una estructura utilizando dispositivos Raspberry Pi conectadas en LAN a un servidor para realizar la captura fotográfica simultanea. El sistema utiliza múltiples cámaras sincronizadas y controladas por los dispositivos Raspberry Pi para capturar fotografías desde diferentes ángulos obteniendo una cobertura total alrededor del sujeto humano con una iluminación personalizada.

Lightstage utiliza técnicas de fotogrametría, pero se enfoca en captura de geometría tridimensional, la apariencia y los datos de iluminación del objeto escaneado. Utiliza múltiples fuentes de luz controladas que aportan mayor control sobre la iluminación en el humano a escanear y cámaras de alta resolución en una configuración esférica o semiesférica para capturar imágenes desde diferentes ángulos y condiciones de iluminación. Este sistema es mas sofisticado y se utiliza principalmente en aplicaciones de alta gama, como la creación de personajes digitales en películas o videojuegos.

En la búsqueda de soluciones accesibles y de bajo costo para la creación de escáneres 3D, el proyecto de Pi3DScan se adapta mejor a las capacidades actuales del proyecto en una primera versión de pruebas, aunque no se descarta en un futuro utilizar una estructura mas parecida a la presentada por Ligthstage.



(Pi3DScan) (Lightstage)

Figura 2.1: Escáner Pi3DScan y Lightstage

2.2. Reconstrucción de avatares humanos

La reconstrucción de modelos humanos paramétricos es un tema fundamental en la animación y la visión computacional. Consiste en generar modelos tridimensionales realistas de cuerpos humanos a partir de información visual o datos de captura de movimiento. Estos modelos paramétricos permiten representar de manera eficiente y compacta la forma y la pose del cuerpo humano, lo que resulta crucial en numerosas aplicaciones, como la animación de personajes, la realidad virtual, el análisis de movimiento y la síntesis de imágenes. SMPL-X [7] es una

solución avanzada del modelo SMPL [2, 3] para la reconstrucción de humanos paramétricos.

En este proyecto, se emplea el framework SMPL-X. Esta es una herramienta altamente efectiva y ampliamente utilizada en la comunidad de la investigación, que se basa en una combinación de técnicas de ajuste y aprendizaje automático.

El problema de modelar el cuerpo en 3D ha sido abordado anteriormente dividiendo el cuerpo en partes y modelando estas partes por separado. Enfoques anteriores se centraron en métodos que aprenden modelos estadísticos de forma a partir de escaneos en 3D. Blanz y Vetter [8] fueron pioneros en esta dirección con su modelo facial 3D moldeable a partir de datos de escaneos. Una característica clave de estos modelos es su capacidad para representar diferentes formas faciales y una amplia gama de expresiones. La mayoría de los enfoques se centran únicamente en la región facial y no en toda la cabeza. Sin embargo, FLAME [9], en contraste, modela toda la cabeza, captura las rotaciones en 3D de la cabeza y también modela la región del cuello.

Los modelos más similares a SMPL-X son Frank [10] y SMPL+H [11]. Frank combina tres modelos diferentes: SMPL (sin mezclas de formas de pose) para el cuerpo, un esqueleto creado por un artista para las manos y el modelo FaceWarehouse [12] para la cara. El modelo resultante no es completamente realista. Por otro lado, SMPL+H combina el cuerpo de SMPL con un modelo de mano en 3D que se aprende a partir de escaneos en 3D. La variación de forma de la mano se obtiene a partir de escaneos de cuerpo completo, mientras que las deformaciones dependientes de la pose se aprenden a partir de un conjunto de datos de escaneos de manos. Sin embargo, SMPL+H no incluye una cara deformable.

SMPL-X parte de SMPL+H y le añade el modelo de cabeza FLAME. Este toma el modelo completo y lo ajusta a 5586 escaneos en 3D, aprendiendo las mezclas de formas dependientes de la forma y la pose. Esto resulta en un modelo de aspecto natural con una parametrización coherente. Al estar basado en SMPL, es diferenciable y fácil de integrar en aplicaciones que ya utilizan SMPL.

3 Escáner

3.1. Estructura y mantenimiento del escáner

Como se menciona en la introducción, la estructura general del escáner se basa en el proyecto Pi3DScan. Todos los dispositivos tienen asignadas una dirección IP estática dentro del rango 192.168.1.0/24. En este caso, el inventario empleado consta de los siguientes componentes:

Servidor. Se utiliza un ordenador como centro de control del escáner, configurado con la dirección IP 192.168.1.2.

Switch NetGear de 48 puertos [13]. Debido a la gran cantidad dispositivos a conectar, se han utilizado 2 switch. El servidor y todas las Raspberry Pi, se conectan al los switch mediante cables Ethernet facilitando la comunicación y el intercambio de datos entre ellos. Se han configurado de manera óptima para garantizar una transferencia de información rápida y eficiente entre los dispositivos del escáner y el servidor. Los switch se configuran con direcciones IP 192.168.1.3 y 192.168.1.4.

Raspberries Pi 3B+ [14]. Se utilizan 78 Raspberries que se dividen en diferentes roles dentro del escáner: 74 para el control de cámaras, 3 para el control de proyectores y 1 como controlador de las luces LED para la iluminación. Para proporcionar la alimentación necesaria a las Raspberries, se emplean 9 hubs, asegurando un suministro estable y adecuado de energía a todos los dispositivos.

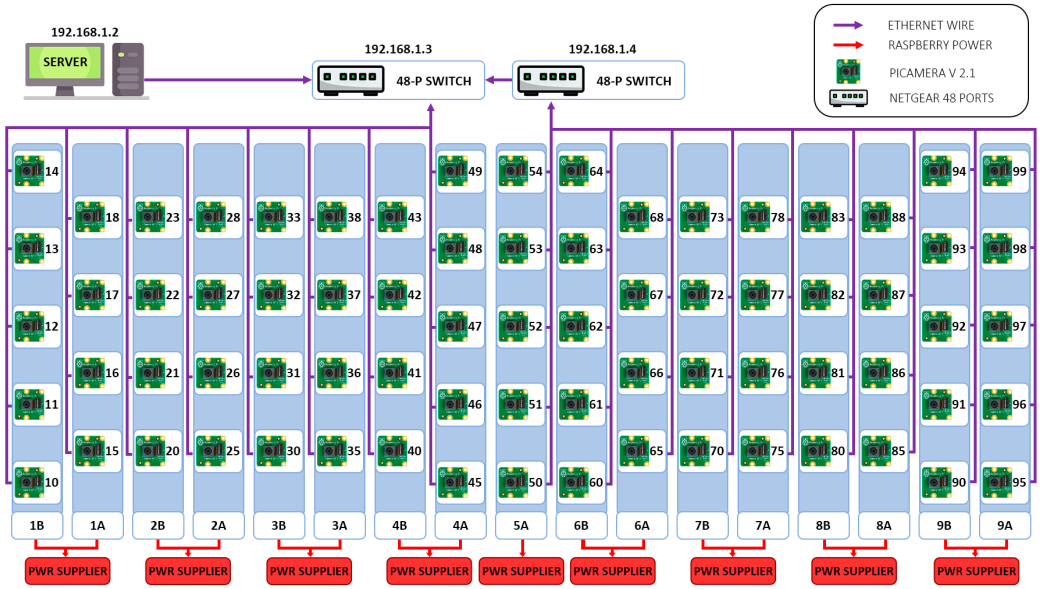


Figura 3.1: Esquema sobre la conexión y distribución de las cámaras por columna.

Tira WS2812B LED [15]. Se utilizan aproximadamente 44 metros de tiras LED (4 metros por columna) con la distribución del esquema 3.2 permitiendo obtener una iluminación personalizada desde cualquier dirección. Cada columna cuenta con una fuente de alimentación asignada. El controlador de las luces LED se configura con la dirección IP 192.168.1.5.

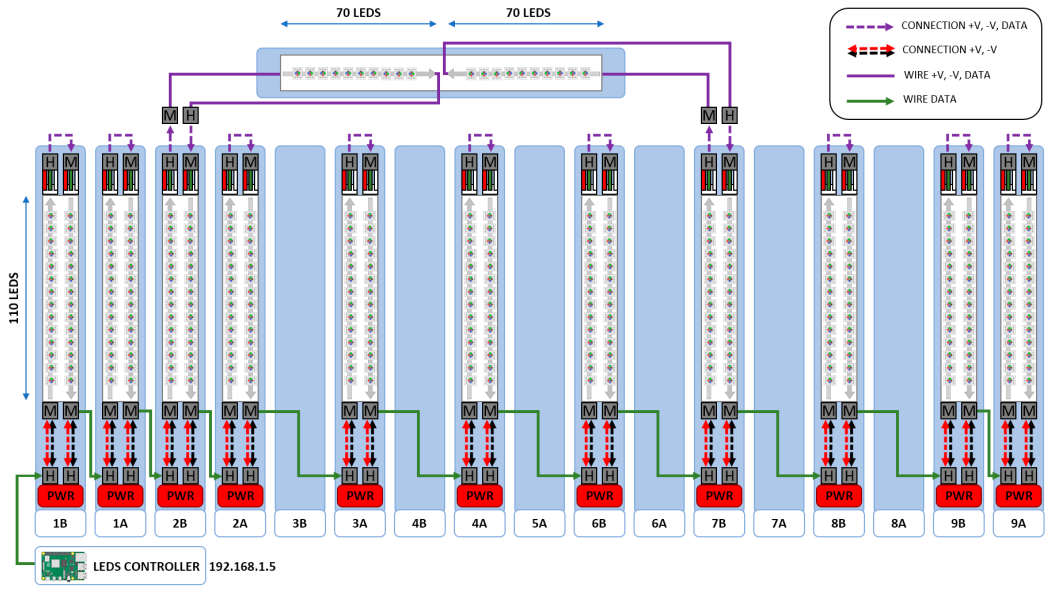


Figura 3.2: Esquema sobre la conexión y distribución eléctrica de las tiras LED.

Proyectores. Se utilizan 3 proyectores para emitir patrones sobre el humano a escanear, permitiendo una cobertura completa. Esto permite al software de reconstrucción detectar similitudes de una forma mucho mas robusta y eficiente en las imágenes resultantes, obteniendo una nube de puntos con mas información [16]. Los proyectores se configuran con direcciones IP en el rango de 192.168.1.6 a 192.168.1.8.

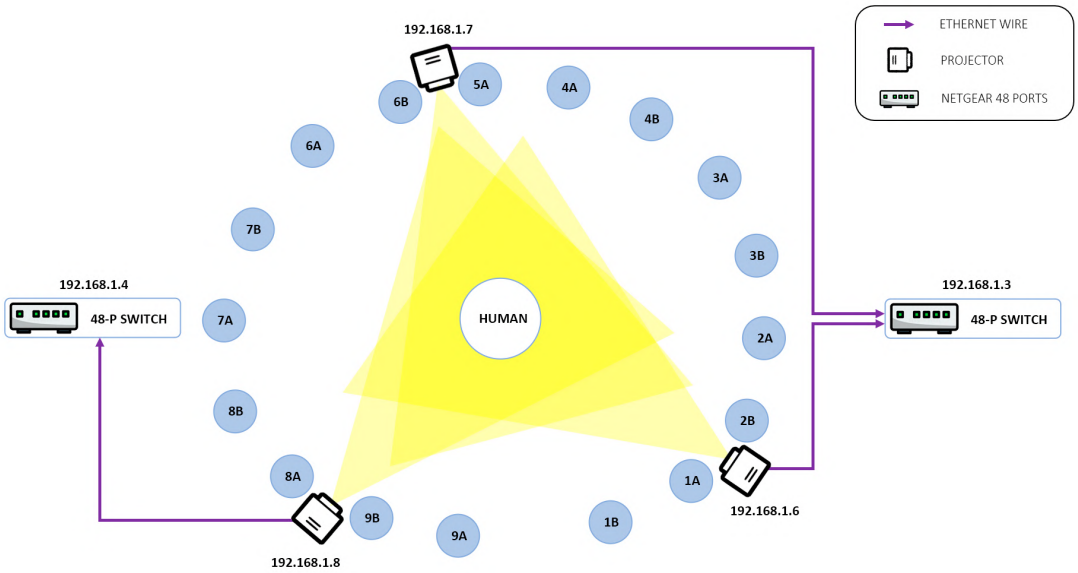


Figura 3.3: Esquema cenital del escáner y los proyectores.

Todas las Raspberries utilizan la misma imagen en sus tarjetas SD. Dicha imagen tiene la distribución de Linux basada en Debian oficial de Raspberry, Raspbian. Para garantizar un funcionamiento fluido y confiable del sistema, cada Raspberry Pi inicia en el arranque automáticamente un servicio que ejecuta el script principal correspondiente a su función (cámara, proyector o controlador de luces LED). En caso de que ocurra un error en el arranque del sistema y el script no se ejecute correctamente, se implementa un mecanismo que reinicia el script automáticamente después de 5 segundos, evitando así la necesidad de una intervención manual para solucionar problemas.

Estos scripts están diseñados para escuchar a direcciones Multicast específicas y realizar las tareas asignadas. Se ha decidido utilizar mensajes Multicast frente a Broadcast debido a que en entornos controlados, es mas eficiente su uso. Estos scripts, comparten aquellas funciones designadas al mantenimiento y actualización:

- Apagar y reiniciar las Raspberries.

- Limpiar la carpeta en la que se almacenan las imágenes con el objetivo de liberar memoria.
- Actualizar el sistema si es necesario. Una vez terminado devuelve al servidor un mensaje indicando si el proceso fue exitoso o si hubo algún error. Para esta función la Raspberry necesita conexión a internet.
- Actualizar Python si es necesario. Una vez terminado devuelve al servidor un mensaje indicando si el proceso fue exitoso o si hubo algún error. Para esta función la Raspberry necesita conexión a internet.
- Sincronizar las Raspberries con el servidor mediante NTP. Devuelve al servidor un mensaje con el nuevo valor del reloj del sistema.
- Comprobar que esta escuchando. Esta función espera a recibir la orden del servidor y le devuelve un mensaje vació. Si el servidor recibe dicha conexión, significa que la orden ha sido enviada y recibida por la Raspberry de forma correcta y que por tanto el script se esta ejecutando correctamente.

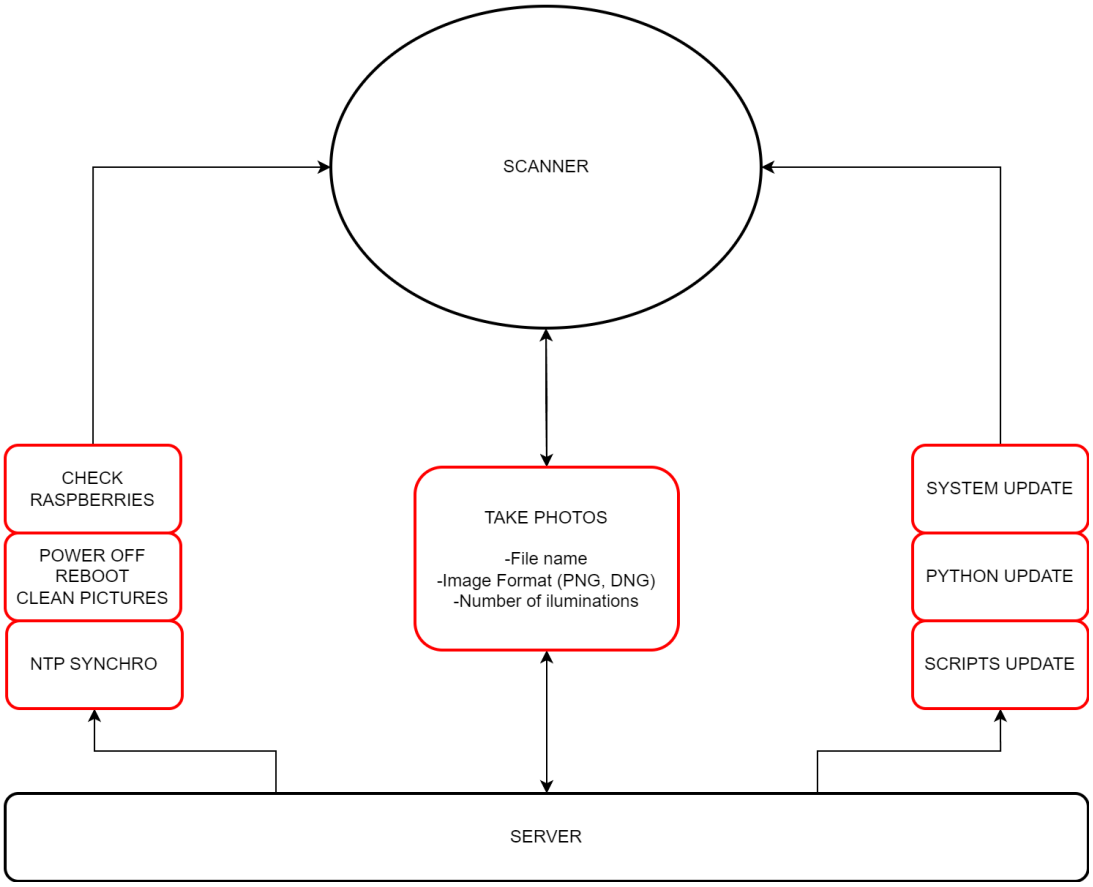


Figura 3.4: Relación del escáner con el servidor.

3.2. Configuración de Raspberries

Es importante configurar todas las Raspberries del escáner para que estén preparadas para realizar las diferentes acciones mencionadas en el apartado 3.1. Partimos del modelo de Raspberry Pi 3B+ con una tarjeta micro SD de 16 GB. Para hacer mas ameno el proceso, se ha configurado una sola tarjeta de la que se ha hecho una imagen completa para escribirla en el resto de las tarjetas necesarias para las Raspberries del escáner.

Con una tarjeta SD completamente vacía, lo primero que se debe hacer es instalar el sistema operativo de Raspberry, Raspbian. Esto se puede hacer desde el instalador oficial ofrecido en la misma pagina de Raspberry. Una vez escrito, se debe introducir la tarjeta en la Raspberry y encenderla.

3.2.1. Configuraciones generales

Pese a no ser necesario, es bueno habilitar la conexión SSH para poder conectarse individualmente a la consola de la Raspberry deseada desde el servidor. Para ello con el comando `raspi-config`, seleccionar la opción `Interface Options` y habilitar SSH.

En esta misma opción se puede habilitar la cámara. Dependiendo de la librería de Python elegida para utilizar la cámara, sera necesario activarla o no. En la sección 3.4 se trata sobre las librerías Picamera y Picamera2 escogiendo finalmente por la segunda opción. En caso de utilizar Picamera es necesario habilitar dicha opción, para que la librería pueda hacer uso de la cámara. Por el contrario, para el correcto funcionamiento de Picamera2, no habrá que activar dicha opción. En caso de hacerlo, la ejecución del script que utilice esta librería dará errores y no podrá hacer uso de la cámara.

Además, para poder realizar capturas con cualquiera de las 2 librerías en máxima resolución `3280x2464` se ha tenido que aumentar la RAM asignada a la cámara de 128MB por defecto a 256MB. Esto se puede hacer modificando el parámetro `gpu_mem=128` del archivo `/boot/config.txt`.

Como configuración opcional se puede modificar la zona horaria y el teclado a gusto del usuario.

3.2.2. Establecer IP estática

El establecimiento de una dirección IP estática en una Raspberry Pi es una tarea fundamental para aquellos casos en los que se requiere una configuración de red consistente y predecible. A través de la modificación del archivo

`/etc/dhcpcd.conf` y la adecuada configuración de los parámetros correspondientes, es posible asignar manualmente una dirección IP estática y establecer la puerta de enlace predeterminada. Los pasos para lograr esta configuración son los siguientes:

1. Acceso al archivo de configuración. Iniciar sesión en la Raspberry Pi utilizando una terminal o una conexión SSH como se ha mencionado anteriormente. Abrir el archivo `/etc/dhcpcd.conf` para desactivar el cliente DHCP (Dynamic Host Configuration Protocol) y configurar las de interfaces de red, la dirección IP, y otros parámetros de red en caso de ser necesarios.

2. Identificación de la interfaz de red. Para establecer una dirección IP estática en una interfaz específica, es necesario identificar el nombre de la interfaz de red correspondiente. En el caso de una conexión Ethernet, la interfaz predeterminada se denomina `eth0`. Sin embargo, si su configuración implica una interfaz inalámbrica, el nombre de la interfaz podría ser diferente, como `wlan0`.

3. Asignación de la dirección IP estática. En el archivo `/etc/dhcpcd.conf`, se debe agregar las siguientes líneas al final, reemplazando los valores según corresponda:

```
interface eth0:
static ip_address = <dirección IP deseada>
static routers = <puerta de enlace>
```

Después de completar este procedimiento, al reiniciar la Raspberry Pi, la dirección IP estática se habrá configurado correctamente.

Es importante tener en cuenta que, al configurar una dirección IP estática, es posible que existan conflictos de direcciones IP si otras máquinas en la red están utilizando la misma dirección. Por lo tanto, es recomendable asignar una dirección IP que sea única y no esté siendo utilizada por otros dispositivos en la red.

Para establecer la IP de forma mas sencilla, se ha creado un script de Python que al ejecutarlo, pide como parámetros de entrada la nueva puerta de enlace y dirección IP que se desea establecer. Este script busca en el sistema el archivo `/etc/dhcpcd.conf`, lo modifica con esta nueva información y finalmente reinicia la Raspberry.

3.2.3. Carpeta compartida en LAN

Habilitar una carpeta compartida en una Raspberry Pi, permite el acceso y la modificación de archivos desde otros dispositivos en una red de área local (LAN). Para esto, se ha decidido que la carpeta compartida sea accesible a través del protocolo Samba, que es ampliamente utilizado para compartir archivos en entornos Windows y Linux.

Una vez instalado Samba mediante el instalador de paquetes de Linux, se debe configurar la carpeta compartida. Para ello, se debe modificar el archivo de configuración de Samba ubicado en `/etc/samba/smb.conf`. Dentro del archivo de configuración, se deben agregar las siguientes líneas al final:

```
[scanner]
    path = /home/pi/Desktop/3DScanner
    browsable = yes
    guest ok = yes
    read only = no
    writable = yes
    create mask = 0777
    directory mask = 0777
```

Estas líneas definen una nueva sección llamada `[scanner]`, que representa la carpeta compartida. El parámetro `path` indica la ruta de la carpeta a compartir en el sistema de archivos de la Raspberry Pi. Los demás parámetros permiten el acceso de invitados, lectura y escritura de archivos, y establecen los permisos de creación de archivos y directorios. Para que los cambios surtan efecto es necesario reiniciar el servicio de Samba. Es importante habilitar los permisos deseados en la carpeta para que otros usuarios, como es en este caso el servidor, tengan los permisos adecuados para realizar modificaciones en la carpeta compartida.

Una vez completada la configuración, se puede acceder a la carpeta compartida desde otros dispositivos conectados a la misma red LAN. Para ello, se debe abrir el administrador de archivos y buscar la ubicación `//192.168.1.X/scanner` en servidores Windows, donde `X` es el distintivo de la dirección IP de la Raspberry Pi en la red LAN y `scanner` es el nombre de la carpeta compartida definida en la configuración. Al acceder a esta ubicación, se podrá ver y modificar los archivos contenidos en la carpeta compartida desde el servidor, tanto vía script como de forma manual.

Como consideraciones de seguridad, es importante tener en cuenta que, al habilitar una carpeta compartida con permisos de escritura y acceso de invitados, se pueden presentar riesgos de seguridad, por ello en este sistema se mantiene a las Raspberries sin conexión a internet para que solo desde el servidor conectado se pueda acceder a dichas carpetas.

3.2.4. Inicio automático

En entornos de sistemas embebidos como la Raspberry Pi, es común tener la necesidad de ejecutar automáticamente scripts al iniciar el dispositivo. Esto se logra mediante el uso de servicios que gestionan la ejecución y supervisión de los scripts en el sistema operativo.

En este trabajo, se ha utilizado un servicio personalizado para habilitar en el arranque de forma automática un script de Python correspondiente al rol que tome la Raspberry en el escáner. El objetivo principal de este servicio es iniciar y mantener en funcionamiento dicho script, encargado de escuchar al servidor y procesar los eventos. El servicio en cuestión es el siguiente:

```
[Unit]
Description=Client Listen Service
After=launcher.service

[Service]
User=pi
Type=simple
ExecStart=/usr/bin/python3 /absolute/path/listener/script.py
Restart=always
RestartSec=5s

[Install]
WantedBy=multi-user.target
```

Para lograrlo, se debe guardar el servicio en `/etc/systemd/system` y otorgar los permisos necesarios para ejecutar scripts de Python. La sección `[Unit]`, describe el servicio como "Client Listen Service" y establece que debe iniciarse después de otro servicio llamado "launcher.service". Este servicio actualiza el repositorio de Git si la Raspberry está conectada a internet, pero finalmente se decidió no utilizarlo por motivos de seguridad.

En la sección `[Service]`, especifica los detalles del servicio, como el usuario que lo ejecutará "User=pi" y el tipo de servicio "Type=simple". La opción "ExecStart" indica el comando para iniciar el script de Python utilizando el intérprete de Python 3 y la ruta completa del script. Las opciones "Restart=always" y "RestartSec=5s" permiten reiniciar automáticamente el servicio en caso de fallos, cada 5 segundos.

Por ultimo, en la sección `[Install]`, la opción "WantedBy=multi-user.target" activa el servicio durante el inicio del sistema en el nivel de ejecución multiusuario.

3.3. Sincronización de los dispositivos

Una de las características fundamentales del escáner desarrollado es la sincronización precisa de los relojes en todas las Raspberry Pi utilizadas. Esta sincronización garantiza que las imágenes se capturen simultáneamente, teniendo en cuenta posibles desfases temporales mínimos entre ellas. Es importante destacar que un menor desfase permite realizar un mayor número de capturas de un objeto con iluminaciones diferentes en un intervalo de tiempo específico.

Para abordar este desafío de sincronización, se ha implementado el Protocolo de Tiempo de Red (Network Time Protocol, NTP) [17]. El uso de NTP permite establecer una sincronización precisa entre las Raspberry Pi y un servidor de tiempo externo. En este proceso, cada Raspberry Pi envía una petición al servidor para ajustar su reloj y asegurar que todas estén en sincronía. El desfase temporal logrado mediante esta configuración oscila aproximadamente entre 10 y 40 milisegundos.

Se han explorado otras alternativas para la sincronización [18], como el Protocolo de Tiempo de Precisión (Precision Time Protocol, PTP) [19]. Sin embargo, debido a limitaciones de compatibilidad a nivel de hardware en el modelo de Raspberry Pi utilizado (PI 3B+), no se ha podido implementar ni realizar pruebas de eficiencia con este método. Según estudios [19], el PTP alcanza una precisión de hasta 100 nanosegundo, lo que lo convierte en uno de los métodos más precisos y efectivos para sincronizar dispositivos en la actualidad.

En resumen, la sincronización de las Raspberries en el escáner construido es un aspecto crítico para garantizar la captura simultánea de imágenes. Se ha utilizado el Protocolo de Tiempo de Red (NTP) para lograr una sincronización precisa con un desfase temporal aproximado de entre 10 y 40 milisegundos.

3.4. Captura de imágenes

La captura de imágenes de alta calidad es un factor fundamental en la fotogrametría, ya que estas imágenes se utilizarán posteriormente en el proceso de reconstrucción tridimensional. A continuación, se presentan algunos aspectos clave al capturar imágenes para asegurar su idoneidad en fotogrametría, según Glev Alexandrov [20].

Resolución. La resolución de las imágenes juega un papel crucial en la captura de detalles precisos y en la obtención de texturas finales con una mayor calidad. Cuanto mayor sea la resolución de las imágenes, mejor se podrán capturar los detalles necesarios para una reconstrucción precisa.

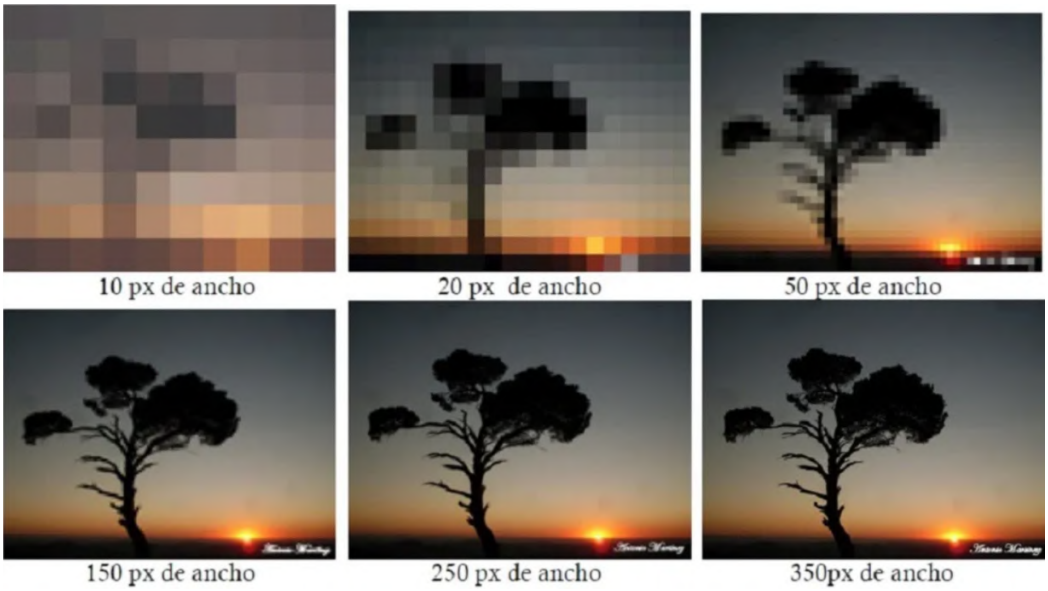
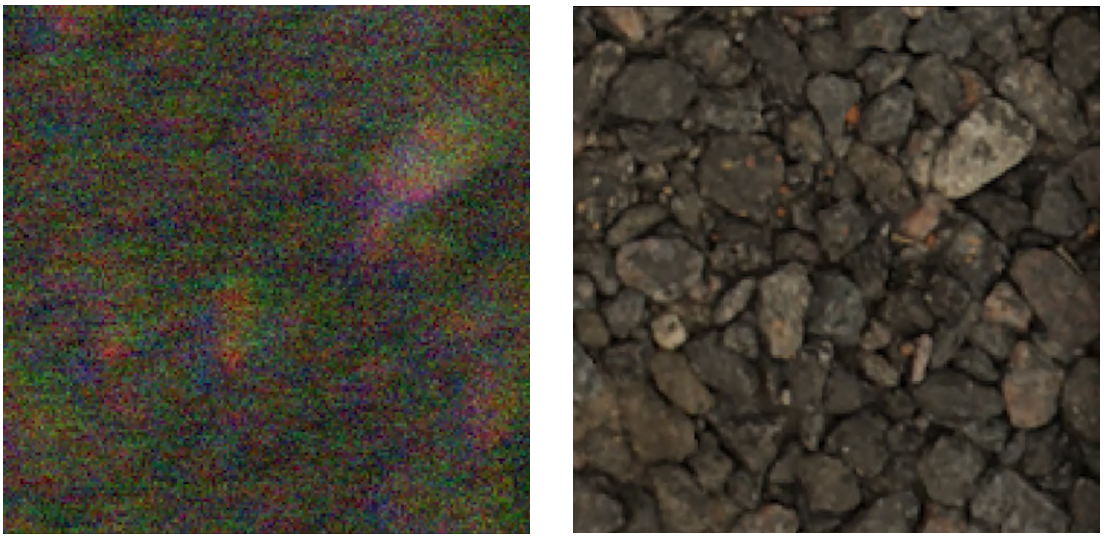


Figura 3.5: Diferencia entre diferentes resoluciones.

Fidelidad espacial: La fidelidad espacial se refiere a la capacidad de las imágenes para representar con precisión las características y detalles de la escena. En caso de tener que decidir, es preferible tener una menor resolución para obtener una mayor fidelidad espacial, ya que al ampliar las imágenes se preservará la definición y los detalles en lugar de simplemente obtener una mayor cantidad de colores RGB. Para mejorar la fidelidad espacial, se deben tener en cuenta aspectos como el tamaño del sensor, el ruido ISO, la distorsión, la nitidez de las esquinas, el efecto viñeta y la aberración cromática.



(alta aberración cromática) (baja aberración cromática)

Figura 3.6: Diferencia entre baja y alta fidelidad espacial

Obtención de datos RAW: La capacidad de obtener datos en formato RAW directamente del sensor de la cámara, ofrece una serie de ventajas significativas en cuanto a la calidad de las imágenes capturadas. A diferencia de los formatos de imagen comprimidos o procesados, el formato RAW conserva una mayor cantidad de información y detalles originales de la escena. Una de las principales ventajas de capturar imágenes en este formato, es que proporciona un mayor rango dinámico. Esto significa que se capturan detalles tanto en las áreas más brillantes como en las más oscuras de la escena, lo que permite una mayor flexibilidad durante la fase de procesamiento posterior. Además, al tener acceso a una mayor cantidad de datos sin procesar del sensor, se pueden aplicar algoritmos y técnicas de corrección más avanzadas y personalizadas.

En el caso del escáner construido, se ha utilizado el modelo de cámara Picamera v2.1 [21], la cual cuenta con el sensor Sony IMX219. Mediante el uso de las bibliotecas de Python, Picamera [22] y su versión más moderna Picamera2 [23], se llevaron a cabo diversas pruebas para determinar la forma más eficiente de capturar el mayor número de fotografías en el menor tiempo posible, lo que permitiría capturar múltiples configuraciones de iluminación en un intervalo de tiempo reducido. Debido a que Picamera2 ofrece mayor control en el flujo de la obtención de la imagen, se decidió finalmente utilizar frente a su versión más antigua, pese que aun se encuentre en su versión 0.4.2.

En la figura 3.7 se muestra el flujo de los datos desde su captura en el sensor hasta su almacenamiento. Al poder evitar el procesamiento del procesador de señal de la imagen (Image Signal Processor, ISP) obteniendo los datos directamente en RAW, se puede conseguir una mayor velocidad en la captura secuencial de imágenes.

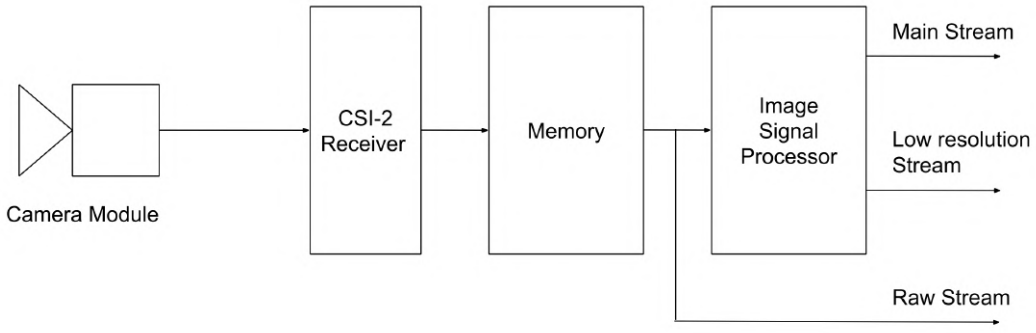


Figura 3.7: Secuencia de eventos de la cámara desde el sensor hasta su visualización.

Además, la librería esta optimizada para trabajar a mayor velocidad los datos del sensor si se utilizan ciertas configuraciones por defecto (sensor modes), a diferencia de utilizar una configuración personalizada.

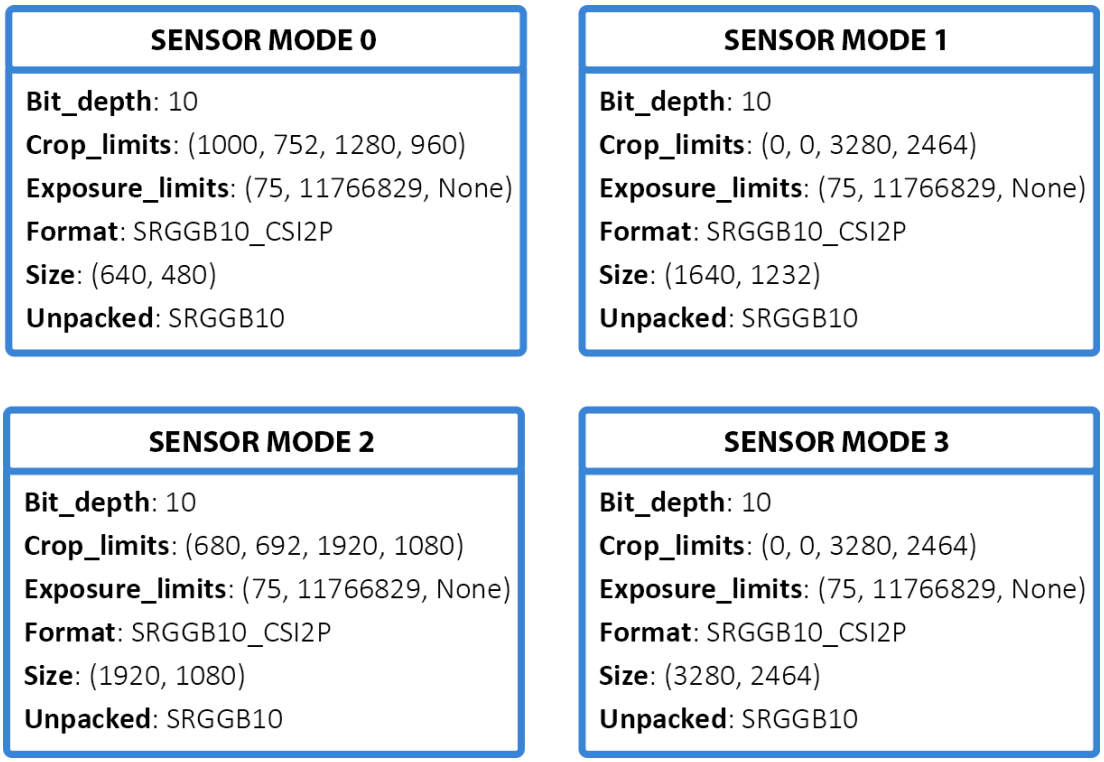


Figura 3.8: Algunas de las configuraciones predeterminadas del sensor.

Conociendo esta información, cuando se recibe una orden para capturar un número de imágenes, en lugar de realizar una captura convencional y procesar cada imagen en un formato legible, se almacena cada buffer de información de la imagen proveniente del sensor de la cámara en una colección de bytes, junto con los meta datos correspondientes. Esto evita el tiempo perdido en el procesamiento del ISP para obtener una imagen en el formato seleccionado. Posteriormente, se utiliza una funcionalidad de la biblioteca para convertir estos buffers y meta datos en imágenes con formato DNG, el cual es aceptado directamente por Meshroom [24]. Es importante tener en cuenta que las imágenes capturadas de esta forma pueden presentar un efecto viñeta de color verde hacia las esquinas como el de la figura 3.9, el cual debe ser corregido mediante el uso de técnicas de corrección y sus meta datos correspondientes, de tal forma que todas las imágenes de los distintos dispositivos tengan valores similares.

Tras diferentes pruebas, utilizando el sensor con unas resoluciones de 1920x1080 y 3280x2464, se han llegado a conseguir 38,7 y 16,6 imágenes por segundo respectivamente. Esto, añadido al pequeño desfase temporal en la sincronización de los relojes de los dispositivos permite obtener a esas resoluciones un número cercano de conjuntos de iluminaciones en el intervalo de un segundo.

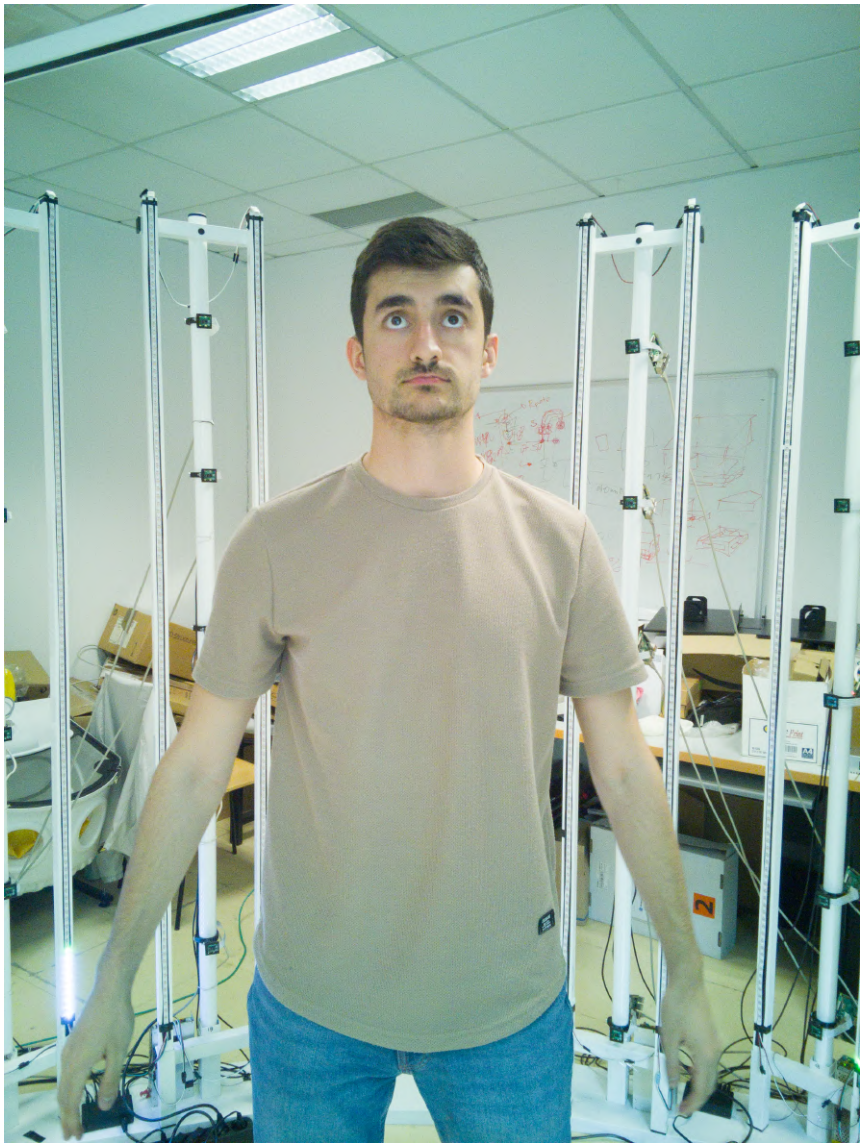


Figura 3.9: Imagen DNG obtenida sin tratar.

En el caso de que el servidor envíe una orden para capturar una sola imagen, se ejecuta el proceso convencional de la biblioteca principal (Main Stream) almacenándose en PNG.

Una vez que las imágenes están listas, se envían al servidor a través de un puerto específico al cual este último está escuchando. El servidor recibe las conexiones de aquellas Raspberry Pi con cámara y abre un nuevo hilo para cada conexión, lo que reduce drásticamente el tiempo empleado en recibir las imágenes. Además, el servidor almacena las imágenes en una carpeta específica para cada configuración de iluminación, y añade al nombre del archivo un distintivo basado en la dirección IP de origen siguiendo el formato "nombre_nºIluminación_IPOrigen.formato". Por ejemplo: "Prueba_1_93.dng".

3.5. Proyección de patrones

El objetivo principal de la proyección de patrones en este proyecto es proporcionar puntos de referencia visuales que faciliten la captura y el seguimiento preciso de la forma y los movimientos del cuerpo humano. Se plantea el uso de proyectores estratégicamente colocados para lograr una cobertura total sobre el humano, asegurando así una captura completa de su superficie.

El sistema propuesto aprovecha la primera captura del conjunto de diferentes iluminaciones para proyectar el patrón seleccionado. Una vez comienza el proceso de captura de imágenes, las Raspberry encargadas reciben la señal y activan la función del script encargada de proyectar durante el breve periodo de tiempo que dura la primera captura, proyectando un patrón de rejilla QR. La elección de este patrón se basa en la capacidad de introducir un "ruido" o variabilidad regular en el humano, especialmente en las superficies lisas sin textura. Esta variabilidad facilita la detección y el seguimiento de puntos por parte del software de reconstrucción utilizado, lo que resulta en nubes de puntos más densas y uniformes.

Es importante destacar que solo el primer conjunto de iluminación se le aplica la proyección de patrones, mientras que el resto de conjuntos de imágenes no reciben ninguna proyección. Esta decisión se toma para maximizar el número de conjuntos de iluminaciones disponibles y extraer posteriormente múltiples texturas limpias. Al evitar proyectar cualquier patrón en los conjuntos posteriores, se evita la contaminación de la textura resultante, lo que garantiza una reconstrucción más precisa y detallada de como afecta la iluminación sobre el modelo generado.

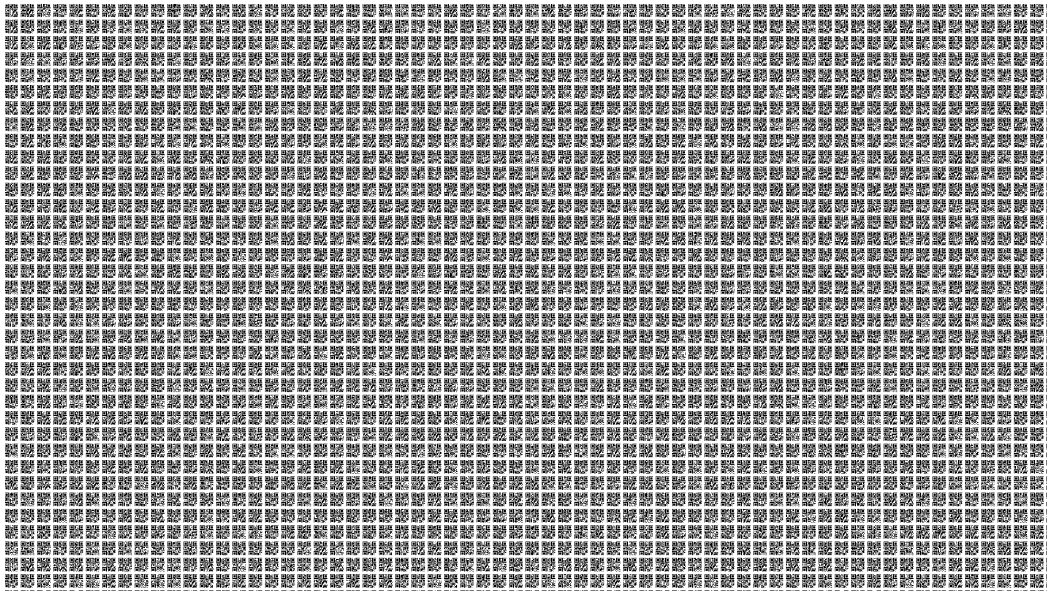


Figura 3.10: Patrón de rejilla QR.

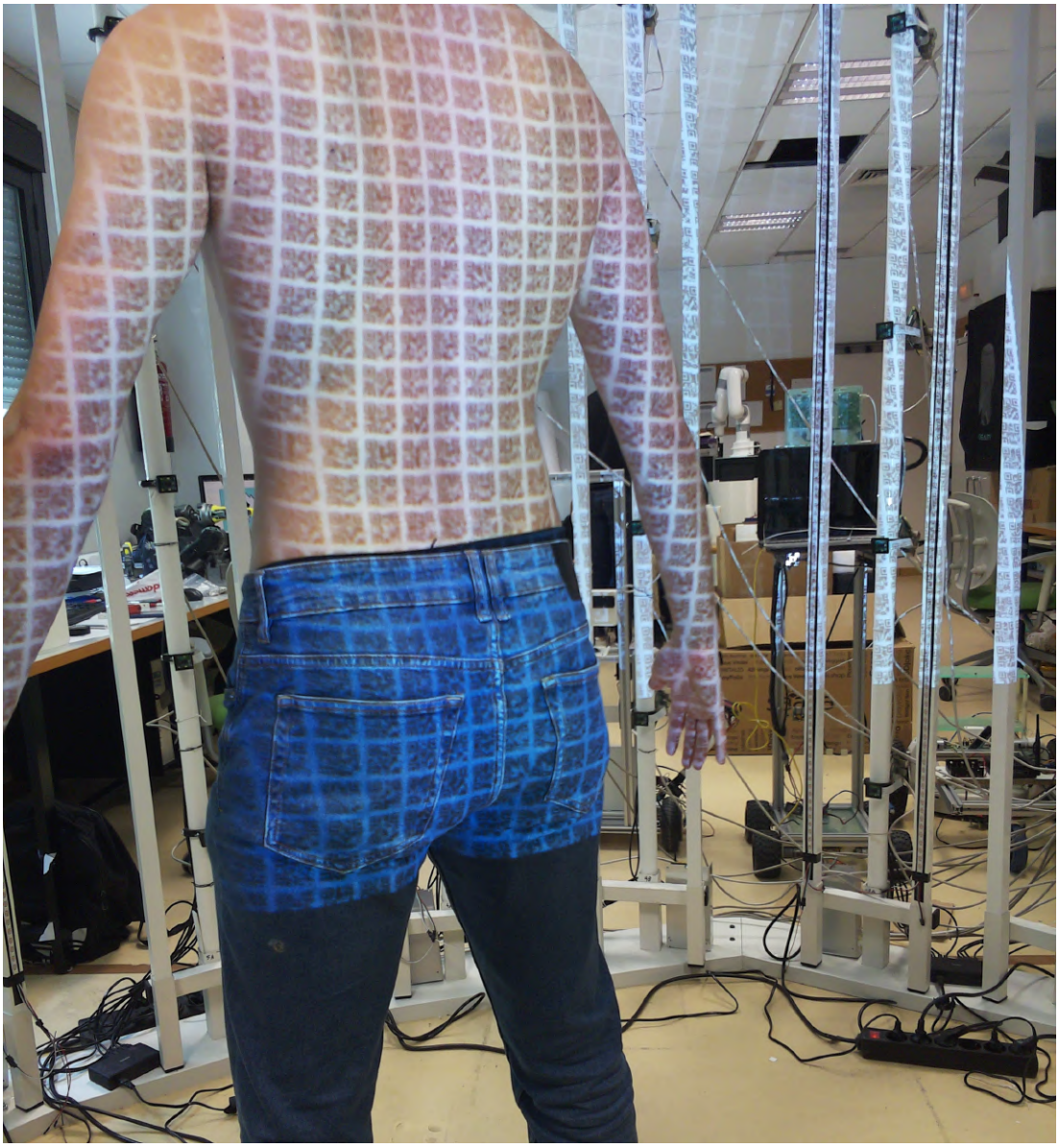


Figura 3.11: Ejemplo de la proyección del patrón en la piel.

4

Reconstrucción

4.1. Semántica en humanos

Una vez que el servidor recibe las imágenes capturadas por las Raspberries del escáner, cada una de ellas contiene información adicional sobre el fondo y el propio escáner, que no es relevante para la reconstrucción del humano.

Se ha buscado una solución automatizada para obtener máscaras precisas del humano. Esta técnica no sólo acelera significativamente el procesamiento de las imágenes en las etapas posteriores de reconstrucción al tener que tratar una menor cantidad de píxeles, sino que también permite delimitar las áreas que deben ser reconstruidas, obteniendo así una nube de puntos del humano lo mas limpia posible sin interferencias del resto de la escena.

Para lograr esto, se ha utilizado la biblioteca de Python llamada Detectron2 [25], una plataforma de desarrollo de software de código abierto que se especializa en la detección de objetos y la segmentación semántica de imágenes. Esta biblioteca cuenta con modelos de aprendizaje profundo altamente efectivos, como Fast R-CNN [26] y Fully Convolutional Network [27], que se utilizan para la detección de objetos y la segmentación semántica, respectivamente.

Mediante el uso del modelo preentrenado "cascade_mask_rcnn_X_152_32x8d_FPN_IN5k_gn_dconv" ofrecido por la biblioteca Detectron2, se ha logrado obtener una máscara precisa del humano en cada imagen. Además, si el servidor utilizado es compatible con CUDA, se aprovecha esta capacidad para acelerar el proceso de obtención de las máscaras de forma significativa.

(imagen de la cámara) (mascara de la imagen)

Figura 4.1: Máscara generada automáticamente respecto a su imagen original

Se ha empleado el mismo modelo preentrenado para reconocer las coordenadas (x, y) de las articulaciones deseadas en la imagen del sujeto humano. Estas coordenadas detectadas se utilizan posteriormente para proyectar las ubicaciones de las articulaciones al espacio 3D, considerando la posición y orientación de las cámaras correspondientes. La proyección de estos puntos en el espacio 3D se realiza teniendo en consideración las características ópticas y geométricas de las cámaras y los sensores utilizados, garantizando así una correspondencia precisa entre las coordenadas de las articulaciones detectadas en las imágenes y su ubicación en el espacio 3D.

Al conocer todas las proyecciones en el espacio 3D, se busca para cada articulación el punto medio en el que se encuentran todas las proyecciones, obteniendo un punto 3D aproximado del lugar en el que se encuentra dicha articulación. En este momento, es posible aplicar transformaciones de posición y rotaciones para alinear la nube de puntos resultante con un modelo de referencia de tal forma que queden solapadas las articulaciones detectadas con la posición de estas en el modelo de referencia.

Cabe destacar que esta librería actualmente solo se encuentra disponible para sistemas operativos basados en Linux, siendo la única parte de todo el proceso que necesita esta condición para funcionar.

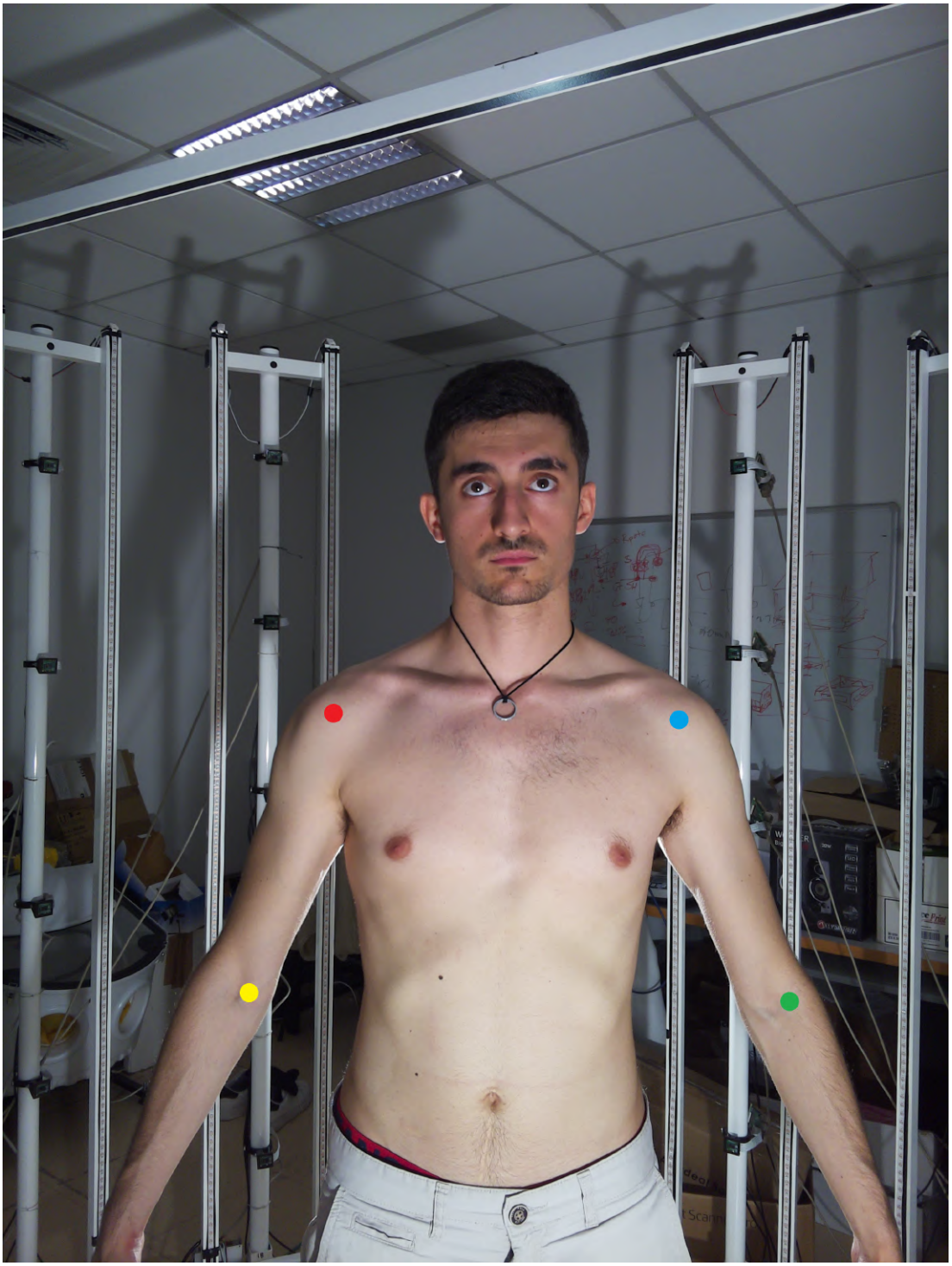


Figura 4.2: Articulaciones detectadas: Hombro izquierdo - Rojo. Hombro derecho - Azul. Codo izquierdo - Amarillo. Codo derecho - Verde.

4.2. Pipeline de reconstrucción

4.2.1. Obtención de la nube de puntos

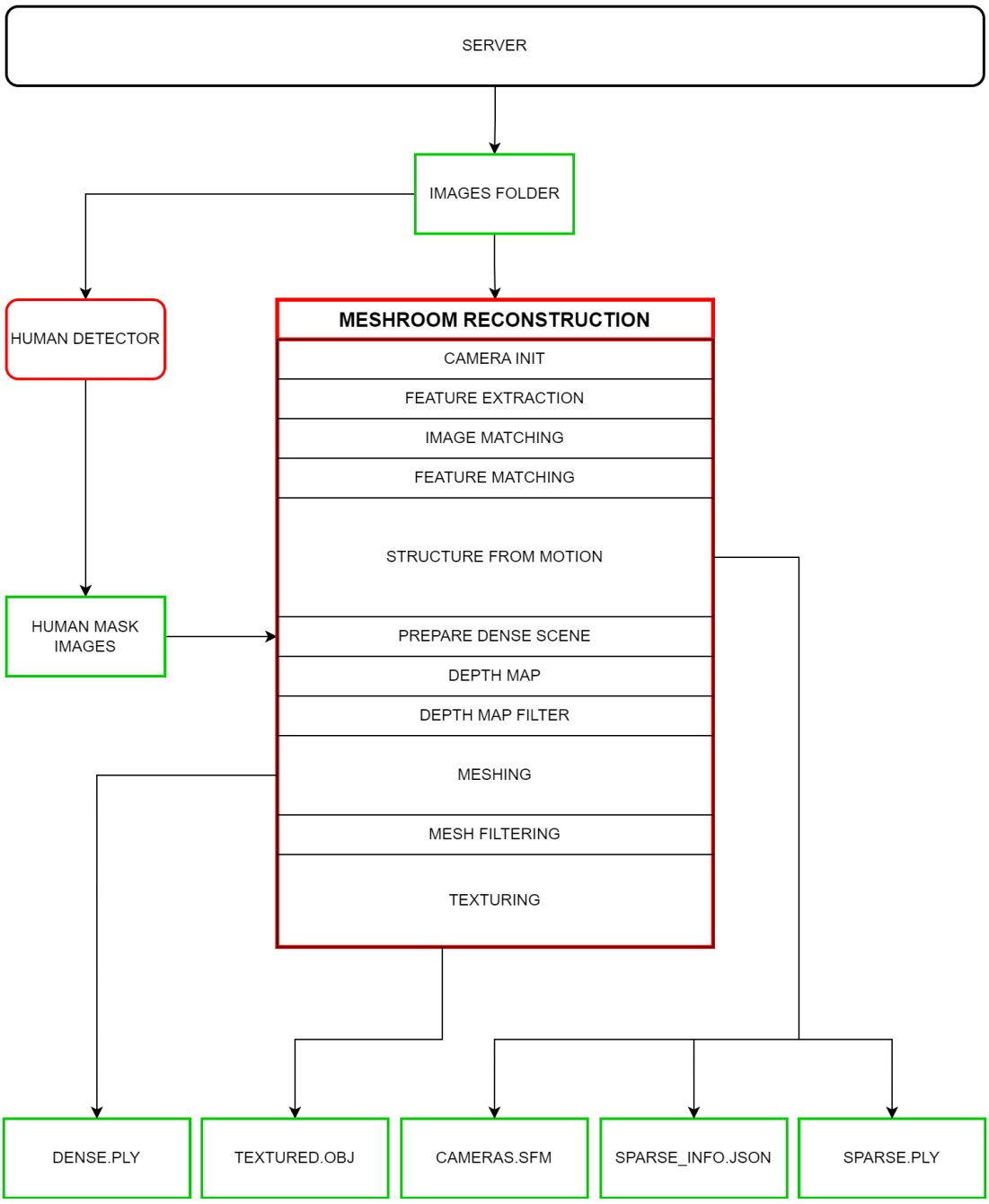


Figura 4.3: Flujo del proceso automático de reconstrucción de la nube de puntos en Meshroom.

La reconstrucción de la nube de puntos se llevó a cabo utilizando el software Meshroom [24], como se mencionó anteriormente. Este software implementa un pipeline de procesamiento que automatiza el proceso de fotogrametría al extraer características de las imágenes, encontrar correspondencias, reconstruir una nube de puntos a partir de ellas, generar una malla 3D, simplificar el modelo y aplicar texturas. Estos procesos pueden ajustarse según las necesidades para obtener resultados más precisos o eficientes.

Dado que Meshroom no proporciona una librería nativa para Python, se desarrolló una pseudo librería en un script que simula la ejecución de los mismos nodos utilizados por Meshroom a través de la línea de comandos desde Python, mediante la ejecución de los binarios y otros archivos que vienen con la instalación del software. Los resultados de cada nodo se almacenan en una nueva carpeta con un indicador de la fase del proceso y su nombre correspondiente. La ejecución de cada nodo depende de los archivos generados en la fase anterior, lo que requiere seguir un pipeline secuencial sin la posibilidad de aplicar paralelismo.

Además, se incluyó un nodo adicional para convertir los archivos generados por Meshroom en el formato .ABC a .PLY. Este último formato es ampliamente compatible con otros software utilizados en el campo de la reconstrucción 3D, no limitándose exclusivamente a Meshroom.

Los nodos principales en el proceso de reconstrucción son los siguientes:

1. CameraInit: En este nodo se cargan los meta datos de las imágenes y la información del sensor. A partir de las imágenes de entrada, se calculan las posiciones y orientaciones iniciales de las cámaras.

2. FeatureExtraction: Este nodo se encarga de extraer características (puntos de interés) de las imágenes de entrada. Estas características son utilizadas para calcular la posición y orientación relativas de las cámaras. Es importante destacar que en este paso se selecciona el descriptor utilizado para extraer las características de las imágenes. Tras realizar múltiples pruebas, se ha observado que los algoritmos “Dfsift” [28] y “Akaze” [29] ofrecen resultados finales altamente satisfactorios, especialmente en superficies complejas como la piel.

3. ImageMatching: Con el objetivo de identificar imágenes similares y descartar aquellas que no aporten información relevante a la reconstrucción, se lleva a cabo un proceso de emparejamiento adicional entre las imágenes.

4. FeatureMatching: En este nodo, se realiza el emparejamiento de las características extraídas en la etapa anterior entre todas las imágenes. Esto permite calcular las posiciones y orientaciones relativas de todas las cámaras de manera global.

5. StructureFromMotion: Este nodo se utiliza para reconstruir una escena tridimensional a partir de las imágenes y características identificadas en etapas anteriores. Como resultado, se obtiene una nube de puntos "sparse" junto con información sobre la relación de cada punto con las coordenadas en píxeles (x,y) de las imágenes correspondientes. Además, se genera un archivo .sfm que contiene información sobre la posición y matriz de rotación en el espacio para cada cámara.

6. PrepareDenseScene: En este nodo, se realizan preparativos en las imágenes para eliminar su distorsión y utilizarlas en las fases siguientes. Es en este punto, se seleccionan las máscaras del humano mencionadas en la sección 4.1 para procesar únicamente esa parte de la imagen. Además, aqui se tiene la opción de incluir imágenes con patrones de luz proyectados para mejorar la calidad de la nube de puntos resultante.

7. DepthMap: Utilizando la técnica de estereovisión, este nodo calcula mapas de profundidad para cada imagen de entrada. Estos mapas proporcionan información sobre la distancia de cada punto en la imagen con respecto al objeto. Cabe destacar que este nodo requiere el uso de CUDA para su funcionamiento.

8. DepthMapFilter: Con el propósito de corregir y filtrar los mapas de profundidad calculados en la etapa anterior, este nodo se encarga de aislar las áreas que podrían estar ocultas por otros mapas, mejorando la calidad y reduciendo el ruido en los mapas de profundidad.

9. Meshing: Este nodo genera la nube de puntos "dense" y, a partir de ella, crea una malla 3D. La malla resultante puede ajustarse para obtener diferentes niveles de detalle según las necesidades del proceso. A partir de este punto, el resto del proceso en Meshroom es opcional, pero se ha implementado para aquellos casos en los que se requiera un modelo de alto poligonaje, con texturas y altamente realista.

10. MeshFiltering: En esta etapa, se lleva a cabo un proceso de limpieza y filtrado de los puntos pertenecientes a la nube de puntos generada en la fase de Meshing. El objetivo principal es mejorar la calidad general de la nube de puntos al reducir el ruido, además, de reducir el número de polígonos de la malla resultante.

11. Texturing: Esta etapa tiene como propósito generar texturas realistas a partir de las imágenes de entrada y aplicarlas de manera adecuada a la malla 3D generada en la etapa de Meshing. En el caso de que la malla carezca de coordenadas de textura (UVs), Meshroom emplea algoritmos sofisticados para generar estas coordenadas automáticamente [30]. Los resultados brindan una apariencia visualmente coherente y detallada del humano escaneado.

4.2.2. Refinamiento de la nube de puntos

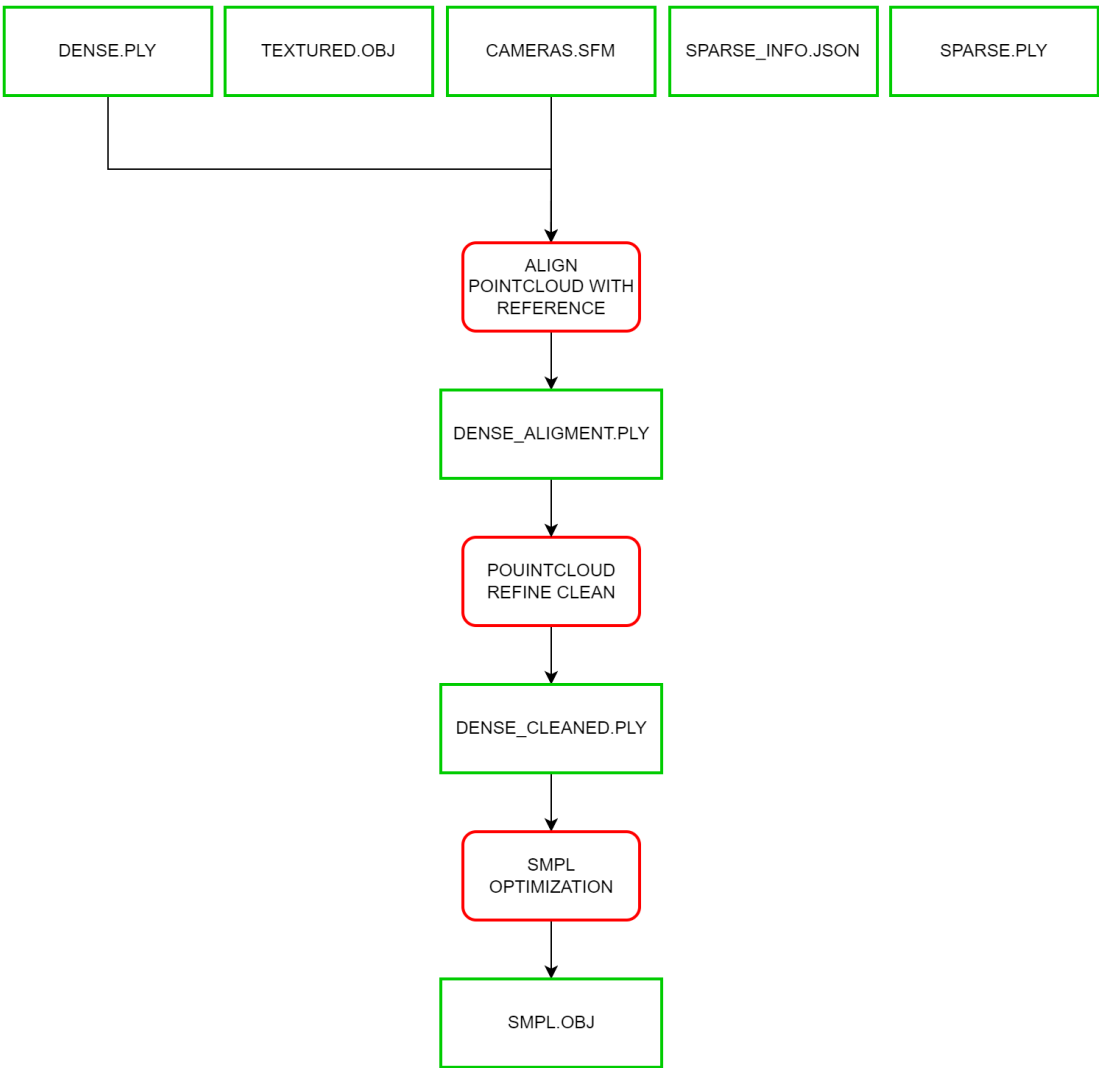


Figura 4.4: Flujo del proceso automático del refinamiento de la nube de puntos.

Para poder optimizar la nube de puntos a un modelo SMPL de la forma mas precisa posible, es necesario que la nube utilizada este completamente exenta de puntos que no pertenecen al humano y, además, este alineada con el eje de coordenadas.

Gracias al enfoque mencionado en la sección 4.1, sobre semántica de humanos, el proceso de reconstrucción produce una nube de puntos que exhibe un humano casi perfecto, con la excepción de algunos puntos dispersos que se originan debido al margen conservado en la máscara utilizada para la detección de humanos. Este margen se establece para evitar problemas y pérdida de información relacionada con la figura humana. Es preferible conservar puntos pertenecientes al resto de la

escena en lugar de perder información valiosa asociada a los puntos que conforman al humano. No obstante, estos puntos no deseados son escasos en número y se distribuyen de forma dispersa en la nube de puntos. Mediante la aplicación de un proceso automático de limpieza que elimina los puntos en áreas de baja densidad, se logra obtener una malla completamente depurada y libre de irregularidades.

Una vez limpia la nube de puntos, se ha trabajado 2 formas diferentes de alinearla.

Por un lado, dado que se dispone de información sobre la posición de las cámaras en relación al escaneo humano en el archivo `cameras.sfm` y se conoce la distribución de las cámaras en columnas, es posible utilizarlas para realizar el alineamiento.

En primer lugar, se aplica una traslación a la nube de puntos de forma que el centroide formado a partir de la posición en la que se encuentran todas las cámaras se sitúe en el eje de coordenadas. De esta forma la nube de puntos se sitúa en el mismo lugar que la referencia pero con una orientación errónea. Para corregir su orientación, se identifica qué cámaras pertenecen a cada columna del escáner, donde, como cada columna del escáner se encuentra perpendicular al suelo, se puede deducir que el vector de la recta que pasa por el centro de todas las cámaras de una columna es proporcional al vector normal del suelo.

Dado que las posiciones de las cámaras en el entorno 3D son relativas y pueden presentar un pequeño error respecto a su posición real en el escáner, se aplica una regresión lineal a todas las posiciones de las cámaras de una columna. Esto permite obtener la recta más probable que contiene a las cámaras y obtener un vector normal del suelo con mayor precisión en comparación a utilizar el vector obtenido a partir de la diferencia de posiciones de dos cámaras de una columna, ya que este último puede presentar una dirección errónea si alguna de las cámaras tiene un error significativo en su posición original.

Una vez se obtiene este vector se aplica una rotación a todos los puntos de la nube de puntos de forma que el vector normal del suelo coincida y sea proporcional al vector [0,1,0]. Como resultado, se obtiene una nube de puntos que se superpone a la referencia utilizada, pero que todavía presenta una rotación alrededor del eje Y [0,1,0] que debe corregirse.

Para ello, considerando que el humano escaneado siempre mira hacia la columna central del escáner (columna 5A), se puede aplicar una rotación que alinee el vector formado por el centro de la escena [0,0,0] y la posición de una de las cámaras de dicha columna con el vector del eje Z [0,0,1] (vector Lookat). Al aplicar dicha rotación, se logra un alineamiento casi perfecto de la nube de puntos respecto a la referencia tanto en posición como en orientación, lo que permite tener una nube de puntos preparada para su optimización SMPL.

La segunda opción de alineación se basa en la detección de articulaciones. Una

vez se obtienen las coordenadas de las articulaciones en las imágenes originales, se proyectan estas coordenadas al espacio 3D desde la posición y dirección de la cámara correspondiente. Esto implica considerar características de la lente, como la longitud focal y los parámetros de distorsión producidos por la lente. Al proyectar las coordenadas de las articulaciones desde múltiples imágenes originales, se observa que las proyecciones convergen en un área específica de la nube de puntos, lo cual indica aproximadamente la ubicación de la articulación en cuestión. Utilizando el algoritmo RANSAC [31], es posible estimar el punto medio preciso donde es más probable que se encuentre dicha articulación. Dado que se conoce la posición de todas las articulaciones en el modelo de referencia SMPL, se aplica una traslación y rotación a la nube de puntos para superponer la ubicación de la articulación detectada con la correspondiente en el modelo de referencia.

Esta opción de alineamiento depende de la calidad de la detección de las articulaciones. Sin embargo, en la práctica, brinda resultados casi perfectos al igual que el otro método de alineación, lo que permite posicionar la nube de puntos en la ubicación adecuada para la posterior optimización mediante SMPL.

4.2.3. Optimización SMPL

El uso del modelo de cuerpo humano SMPL (Skinned Multi-Person Linear Model) en el proceso de reconstrucción tridimensional presenta múltiples beneficios y contribuye significativamente a la obtención de resultados precisos y realistas. SMPL es un modelo paramétrico altamente versátil capaz de representar la anatomía humana de forma compacta mediante una combinación lineal de formas y una aproximación de deformación cutánea suave. El modelo SMPL se caracteriza por una topología consistente y una estructura jerárquica de huesos con sus respectivos pesos sobre los vértices de la malla, lo que facilita su aplicación en tareas de reconstrucción y animación.

La optimización de SMPL implica ajustar un conjunto limitado de parámetros que posan y dan forma al modelo con el fin de conseguir una malla lo más parecida a la nube de puntos del usuario resultante en las secciones anteriores.

Gracias a la función de un proyecto de SEDDI se ha podido obtener un modelo SMPL muy próximo a la nube de puntos refinada. Esta función realiza una búsqueda de puntos vecinos, calculando las distancias mínimas al cuadrado de cada vértice de la malla al punto de la nube de puntos más cercano, adoptando un enfoque de optimización numérica con el fin de obtener un modelo cuya malla se superponga lo máximo posible con los puntos de la nube de puntos. En particular, el algoritmo de optimización L-BFGS-B [32] implementado con la librería de Python, Scipy [33], es el método utilizado para acercar el modelo paramétrico a la nube de puntos.

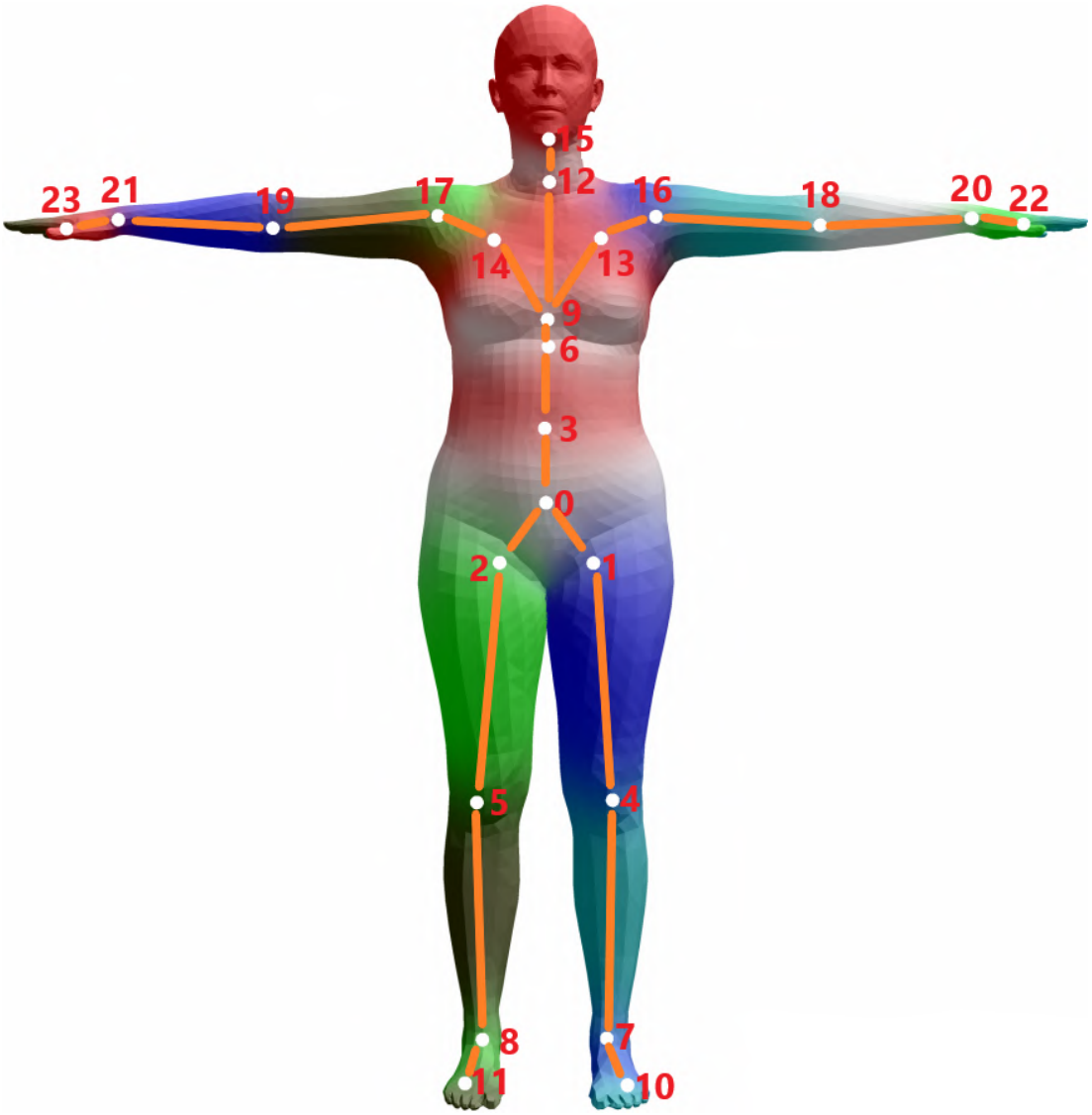


Figura 4.5: Esqueleto modelo SMPL.

4.3. Visualizador 3D

Como se menciono en la introducción, para agrupar todas las funcionalidades mencionadas a lo largo de este trabajo, se ha creado una aplicación con un visualizador 3D y una interfaz acompañada de botones para facilitar el control del escáner y el proceso de reconstrucción. Para ello, se ha utilizado la librería de Python, Open3D [34], la cual proporciona una amplia gama de funciones y algoritmos para manipular y analizar datos en formato tridimensional, así como una interfaz gráfica para visualizar modelos 3D de manera interactiva en tiempo real. Esta capacidad de visualización en tiempo real es muy útil para la inspec-

ción y la depuración de los resultados obtenidos a lo largo de este proceso de reconstrucción.

Se ha utilizado programación concurrente multihilo para mejorar el rendimiento. La programación multihilo es una técnica de programación en la que se utilizan múltiples hilos de ejecución simultáneos dentro de un programa. Cada hilo representa una secuencia independiente de instrucciones que pueden ejecutarse de forma paralela o concurrente. Esto permite que varias partes del programa se ejecuten al mismo tiempo, lo que puede mejorar la eficiencia y la capacidad de respuesta de una aplicación. En caso de no utilizarlo, al ejecutar un botón, la aplicación quedaría congelada hasta que el proceso en cuestión terminase, impidiendo utilizar otras funciones del escáner o el propio visualizador, cosa inviable en este tipo de aplicaciones cuyos procesos tienden a durar horas.

Cabe destacar que la aplicación ha sido diseñada pensando en la posibilidad de seguir ampliando su funcionalidad en un futuro. En este momento se encuentra en una versión muy primaria del desarrollo, aunque tiene implementadas todas las funcionalidades explicadas en este trabajo. Se ha implementado un control de errores, evitando que la aplicación se cierre al intentar ejecutar un proceso sin ser el momento. Cuando esto ocurre, aparece un mensaje en medio de la pantalla indicando que esta acción no se puede realizar todavía o que este archivo todavía no esta disponible para su visualización.

4.3.1. Barra de menú

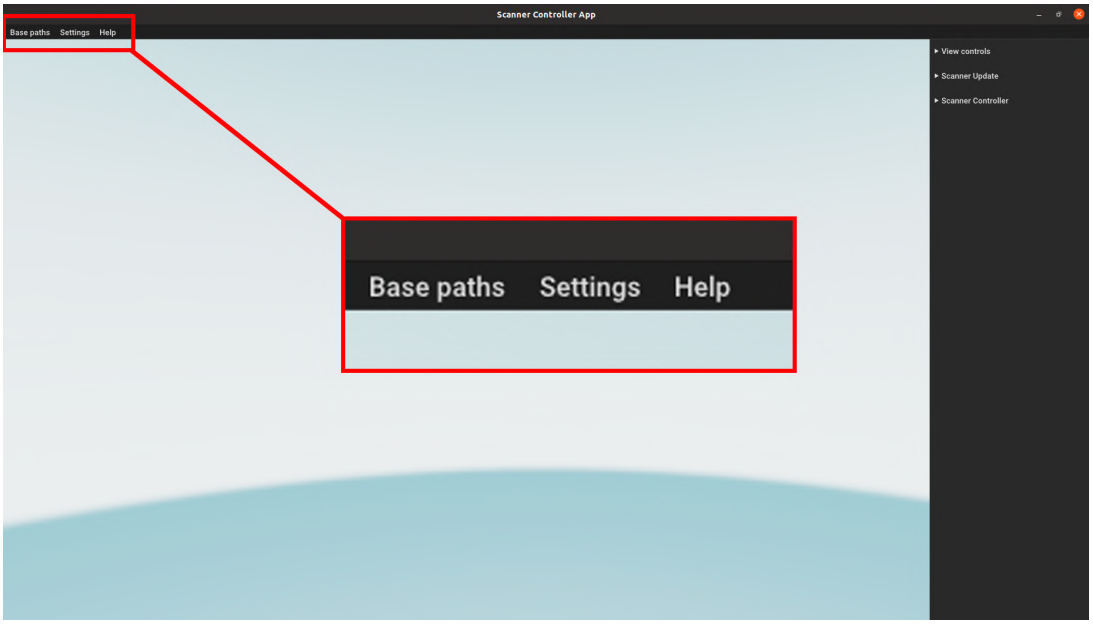


Figura 4.6: Barra de menú.

Como se puede aprecia en la figura 4.6, la aplicación consta de una barra de menú en la parte superior del visualizador donde se puede encontrar los siguientes botones:

Base paths: Al pulsar sobre este botón, se despliega una ventana emergente en medio del visualizador como el de la figura 4.7. En este se muestran 3 cuadros de textos en los que hay que introducir:

Figura 4.7: Rutas base para el visualizador.

- En el primer cuadro de texto se debe introducir la dirección en la que se quiere almacenar los proyectos que se van a ir generando según se hagan fotos y reconstrucciones.

- El segundo cuadro de texto se debe poner la ruta de la carpeta de los binarios de Meshroom. Dicha ruta se encuentra en una carpeta llamada "/Bin" en el lugar de la instalación del software de Meshroom. Es necesario indicar donde se encuentra ya que se utilizan los binarios de Meshroom para ejecutar los nodos directamente desde la linea de comandos.
- Por ultimo, en el ultimo cuadro de texto, se debe indicar la ruta de la base de datos de sensores que ofrece Meshroom en su instalación. Esto permite a Meshroom reconocer el tipo de cámara que se ha utilizado al introducir las imágenes del proyecto a realizar. Meshroom se aprovecha de la información de los meta datos de las imágenes para conocer el tipo de cámara con el que se hicieron y aplicar unos u otros valores por defecto en el proceso de reconstrucción de la nube.

Una vez definidos las rutas, el usuario podrá cerrar simplemente la pestaña sin almacenar los cambios pulsando "Close" o guardarlo con el botón "Save". En caso de guardarlos, se creara en la misma ruta del ejecutable de la aplicación un nuevo archivo .JSON, o en caso de que exista se sobrescribirá. De esta forma se asegura persistencia en la aplicación y no es necesario tener que definir dichas rutas cada vez que se abre. Además, permite definir en cada ordenador las rutas necesarias sin tener que almacenarlas directamente en código, encapsulando el código de tal forma que no sea necesario modificarlo en diferentes dispositivos.

El botón "Settings" muestra un desplegable con la opción "Control panel". Al pulsar sobre dicha opción se activara o desactivara el panel de control del escáner situado a la derecha del visualizador de la aplicación.

Por ultimo, en el botón de "Help" abre una pestaña emergente con información del proyecto y nociones básicas sobre el uso de la aplicación, siguiendo la linea de otros software de la industrial.

4.3.2. Controles de visualización

En el panel derecho se pueden encontrar los diferentes controles tanto del visualizador como del escáner. Actualmente existen 3 despegables: "View Controls" para los diferentes controles relativos al visualizador, "Scanner Update" con funciones para el mantenimiento y actualización del software del escáner, y por ultimo "Scanner Controller", para el control y ejecución del pipeline de reconstrucción.

Al desplegar la sección "View Controlls" se muestran las opciones de la figura 4.8. El primer apartado se pueden ver diferentes opciones para el control del movimiento de cámara y el entorno del visualizador.

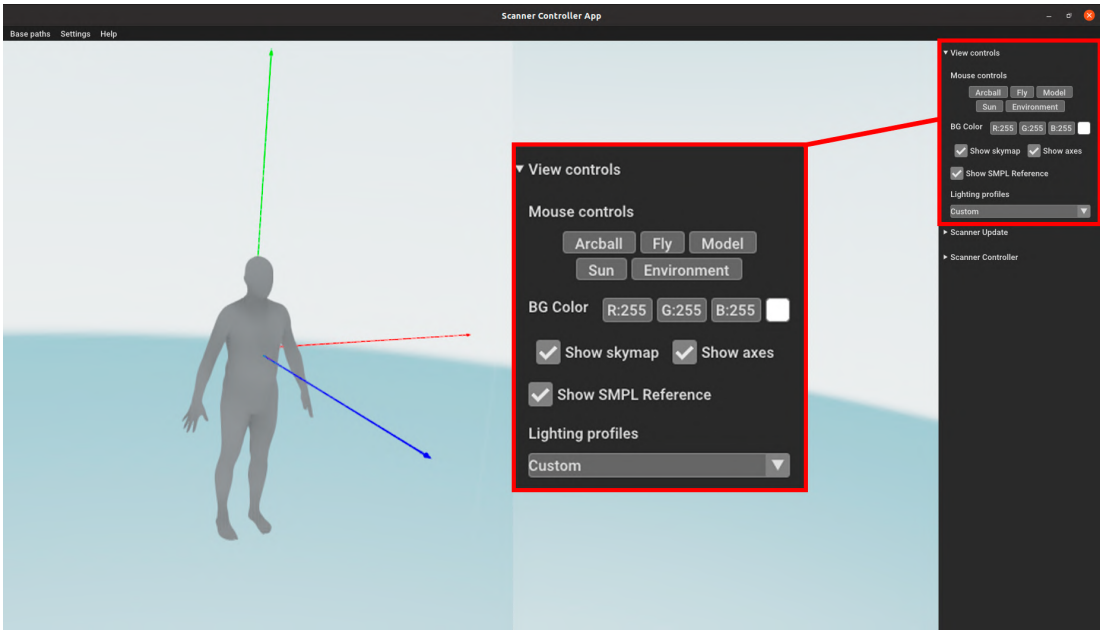


Figura 4.8: Controles de visualización.

1. "Arcball": Esta opción viene por defecto y permite controlar la rotación del entorno teniendo como punto centrar el eje de coordenadas.

2. "Fly": Permite al usuario moverse por el entorno 3D con las teclas W (adelante), S (retroceder), A (izquierda), D (derecha). Además se puede rotar la cámara en tonel con las teclas Q (izquierda) y E (derecha).

3. "Model": Al seleccionar esta opción, en vez de rotar la cámara y el entorno, se rotara únicamente la nube de puntos o la malla que este cargada en ese momento.

4. "Sun": Esta opción permite controlar la luz direccional del sol del visualizador. Pulsando sobre la rueda central del ratón, se puede controlar la luz direccional del sol sin necesidad de meterse en esta opción.

5. "Enviroment": En vez del modelo o la cámara, se rota el propio entorno de la escena. en un futuro se podrán cargar mapas HDRI, lo que permitirá poder orientarse en la dirección deseada.

En la primera fila de la segunda sección se muestran los valores de color del fondo. A la derecha de estos valores aparece un recuadro que al pulsar sobre él, se abre una pipeta de colores para poder elegir de forma visual, 4.9. Para poder ver el fondo con el color seleccionado se debe desactivar la opción "Show skymap" de la segunda fila que viene seleccionada de forma predeterminada. La opción "Show axes" de su derecha muestra al activarse el eje de coordenadas en el espacio

3D. Debajo se encuentra la casilla “Show SMPL Reference” la cual muestra un modelo SMPL de referencia situado sobre origen de la escena. Dicho modelo es la referencia utilizada para alinear la nube de puntos. Por ultimo, en el apartado “Lighting Profiles” aparecen una serie de configuraciones predeterminadas para colocar el sol (luz direccional) en diferentes posiciones respecto a los ejes de coordenadas. En caso de modificar el sol de forma personalizada, el desplegable se marcara como “Custom”.

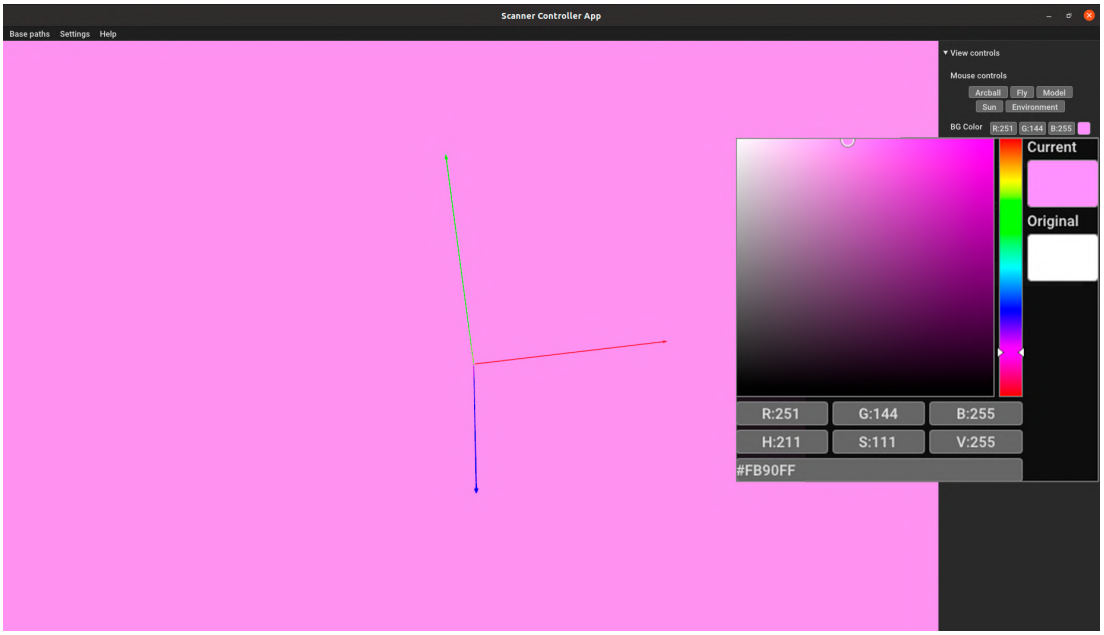


Figura 4.9: Selector de color de fondo.

4.3.3. Controles de mantenimiento y actualización

El segundo desplegable “Scanner Update” 4.10, consta de 2 botones en la primera fila,“System Update” y “Python Update”. El primero de ellos envía una señal a todas las Raspberries detectadas del escáner para que ejecuten la función encargada de actualizar el sistema de paquetes de Raspbian mientras que con segundo ejecutan aquella función encargada de actualizar Python a su ultima versión. Para que dichas funciones se ejecuten sera necesario que las Raspberries del escáner tengan acceso a internet. Los dispositivos del escáner para que funcionen no tienen la necesidad de estar conectadas a la red de internet, además de que es recomendable no hacerlo ya que se crearían vulnerabilidades de seguridad. Asimismo, se han creado estas funciones por si en algún momento se desea actualizar los paquetes de sistema o la versión de Python para todas las Raspberries de la estructura.

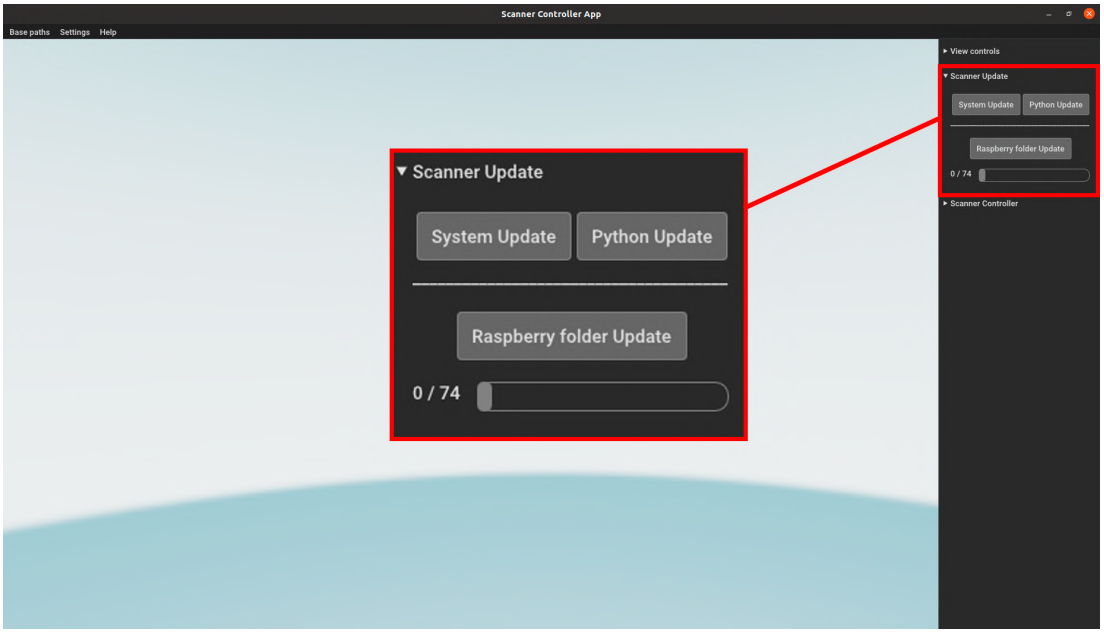


Figura 4.10: Controles de mantenimiento y actualización.

El siguiente y ultimo botón del desplegable "Raspberry Folder Update" se utiliza para actualizar o agregar scripts de Python del escáner que han sido modificado o añadidos en el servidor para introducírselo a los dispositivos. El servidor envía a la correspondiente carpeta compartida en LAN de Samba por las Raspberries, todos los scripts de la carpeta seleccionada. De esta forma, las Raspberries siempre tienen la ultima versión de los scripts creados sin necesidad de un sistema de control de versiones con necesidad de acceder a la nube. Debajo de este botón se ha creado una barra de progreso para saber cuando se han actualizado todas las Raspberries de actualizar los scripts

4.3.4. Controles de la captura y reconstrucción

Los primeros 2 botones de esta sección "Power Off", "Reboot" sirven para apagar todas las Raspberries y reiniciarlas respectivamente. Es importante tener en cuenta que pese a apagar las Raspberries mediante este botón, están seguirán recibiendo alimentación, por lo que para apagar completamente el sistema sera necesario cortar la corriente de las regletas correspondientes.

A su lado se encuentra un tercer botón "Clean" que se encarga de limpiar la carpeta en la que se almacenan las imágenes en cada propia Raspberry con el objetivo de liberar memoria en caso de haberse realizado un gran número de capturas.

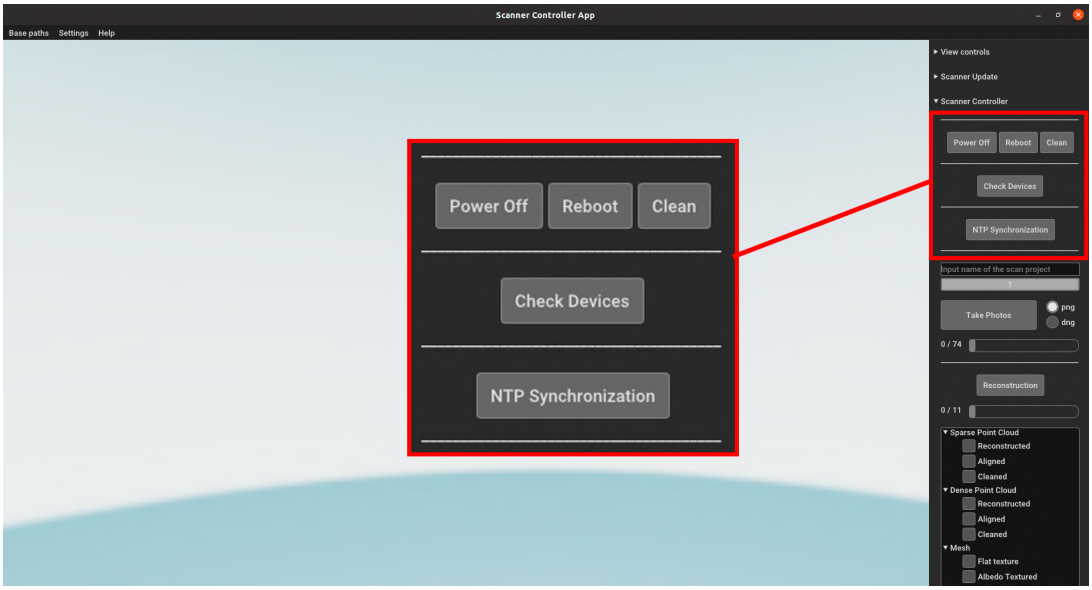


Figura 4.11: Controles del escáner.

La siguiente sección tiene un único botón utilizado para detectar las Raspberries que se encuentran escuchando al escáner correctamente, lo que implica que están preparadas para hacer una captura de imagen. Se abre una pestaña emergente en medio del visualizador con el formato de la figura 4.12 donde aparecen todas las columnas del escáner con sus nombre, y las IP de los dispositivos que no ha detectado por cada columna. Dicha estructura sigue el esquema mencionado en la figura 3.1. Al final del cuadro de mensaje, se indica el número total de dispositivos detectados. Además, se actualiza la interfaz de tal forma que las barras de progreso indiquen el nuevo número de cámaras máximas detectadas.

Figura 4.12: Pestañas emergentes mostrando 0, 62 y 68 Raspberries detectadas.

El botón “NTP Synchronization” envía un mensaje a los dispositivos del escáner para que ejecuten la función encargada de sincronizar los relojes mediante NTP con el servidor, es decir, el ordenador que esta ejecutando el visualizador. Como resultado, se abrirá una pestaña emergente con la diferencia de tiempo de cada dispositivo respecto al servidor, y la mayor diferencia de tiempos entre las Raspberries del escáner.

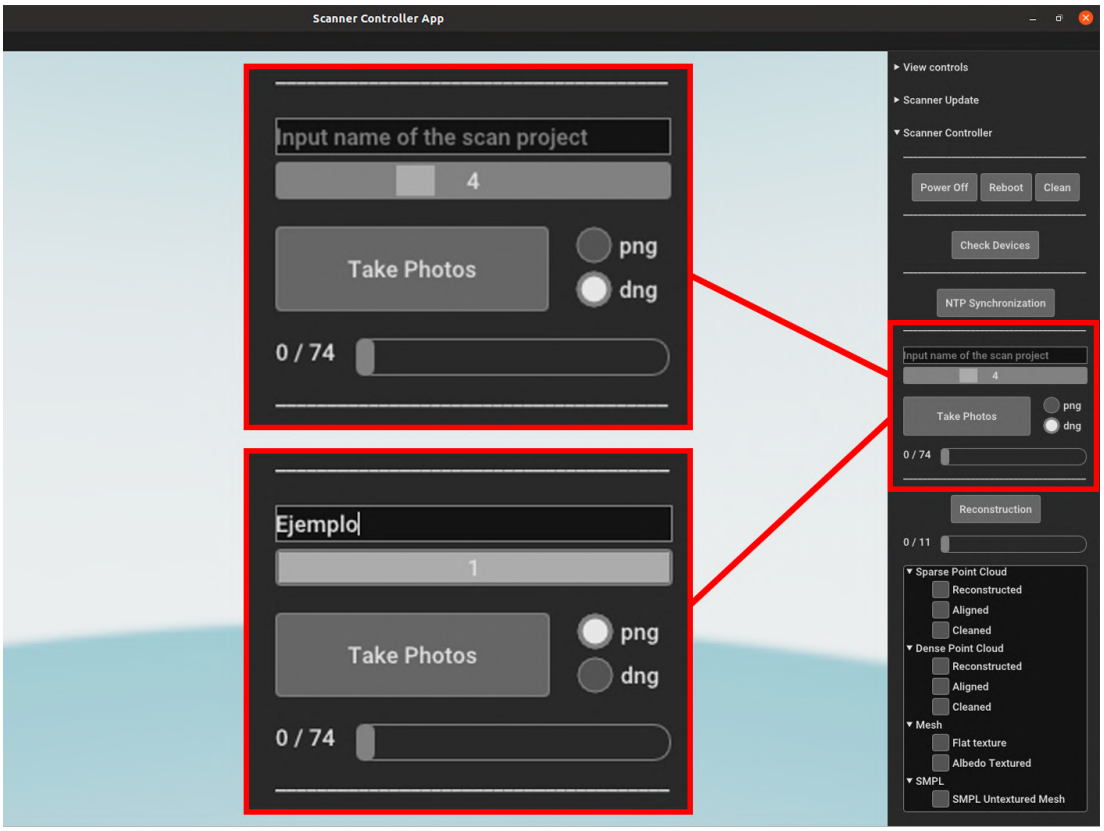


Figura 4.13: “Take Photo”: 4 conjuntos en DNG o 1 en PNG.

El siguiente apartado es el encargado de enviar la orden de realizar la captura de imágenes. Como se ve en la figura 4.13, aparece un recuadro de texto donde el usuario debe introducir el nombre del proyecto que le quiere dar a las capturas y posterior reconstrucción. Esto definirá el nombre de la carpeta en la que se almacenaran las imágenes y su carpeta de la reconstrucción, cuya ruta se ha definido en la sección de ”Base paths”. Debajo de este cuadro de texto se encuentra un deslizador, utilizado para definir el número de capturas (conjunto de iluminaciones) que se quiere realizar. De forma predeterminada, solo se podrá capturar una iluminación, ya que esta seleccionada la opción de PNG, la cual indica que las imágenes se tomaran en dicho formato. Para poder realizar múltiples capturas, se debe seleccionar la opción DNG, ampliándose el número máximo de 1 hasta 10 (número definido para hacer ejemplos). Una vez se ha seleccionado las opciones

deseadas, al pulsar el botón "Take photos", se enviara una señal a las cámaras para comenzar la secuencia de proyección de patrones, captura y diferentes iluminaciones. La barra de progreso de debajo indica el número de cámaras que han finalizado correctamente el envió de las imágenes que han realizado.

Una vez recibidas todos las imágenes se podrá pulsar sobre el botón "Reconstruction", iniciando así el proceso completo y automático de reconstrucción. Dado que este proceso es muy largo, se puede comprobar en que fase del proceso esta siguiendo la barra de progreso de debajo, indicando el número de la fase en la que se encuentra el proceso de Meshroom en cada momento.

Una vez finalizan los diferentes procesos del escáner y las barras de progreso de llenan al 100 %, dicho botón se pondrá en verde indicando que ha finalizado correctamente. En caso de escribir en el cuadro de texto el nombre de un proyecto que ya esta creado, los botones volverán de color verde para avisar que dicha reconstrucción ya esta disponible y se puede ver directamente los resultados.

Finalmente, para visualizar los diferentes archivos generados en la reconstrucción 4.14, se han creado 4 desplegables con casillas de activación para los diferentes pasos del proceso de reconstrucción realizado:

- "Sparse Point cloud": Se visualiza la nube de puntos dispersa junto a los ficheros con las correcciones de alineación y limpieza.
- "Dense Point cloud": Se muestra la nube de puntos densa junto a los ficheros con las correcciones de alineación y limpieza.
- "Mesh": Se visualiza la malla reconstruida a partir de la nube de puntos densa por Meshroom, y su versión texturizada.
- "SMPL": Se muestra el resultado final del proyecto. La malla del modelo SMPL generado a partir de la nube de puntos densa.

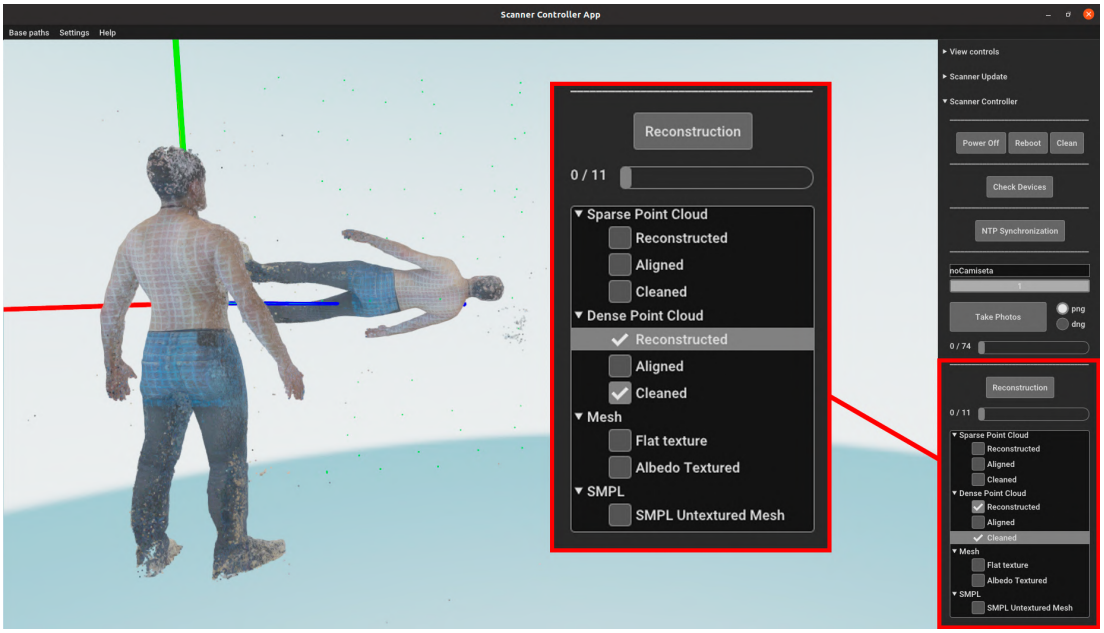


Figura 4.14: Visualización de la nube de puntos densa y refinada simultáneamente.

5 Resultados

Esta sección presenta los resultados obtenidos en el proceso de adquisición de imágenes y reconstrucción automática de un proyecto, junto con los archivos generados durante el proceso, así como comparaciones de tiempos al modificar algunos parámetros de calidad en la reconstrucción.

Sin embargo, es importante destacar que existen varios problemas que aún no se han resuelto por completo, lo que limita la exhibición completa de las funcionalidades del escáner. Algunos de los problemas actuales son los siguientes:

- Problemas eléctricos en la iluminación: Las tiras LED no funcionan actualmente y la causa exacta de este problema se ha descubierto en los últimos momentos. Para controlar la iluminación, la Raspberry correspondiente envía la señales a través del pin GPIO18. Debido a interferencias, la tira LED no estaba recibiendo correctamente las señales, ya que al utilizar dicho pin de la Raspberry, es necesario desconectar el audio `dtparam=audio=off` del archivo `/boot/config.txt`. Como resultado de una solución tan tardía, no ha sido posible realizar una exploración completa con múltiples configuraciones de iluminación para mostrarlo en esta sección.
- Corrupción de tarjetas SD: Por alguna razón, algunas tarjetas SD se corrompen y no se pueden reutilizar. Esto ha impedido el uso simultáneo de todas las Raspberries del escáner. No obstante, este problema no es tan grave como el mencionado anteriormente, ya que solo han fallado entre 6 y 10 cámaras. A pesar de esto, aún había suficientes cámaras disponibles para lograr una cobertura total del humano a escanear.

- Problemas de arranque de las Raspberries: Similar al problema anterior, algunas Raspberries no se inician correctamente en el primer intento y es necesario desconectar y volver a conectar el cable varias veces hasta que el sistema operativo Raspbian funcione correctamente en esa tarjeta. En este punto, las tarjetas son propensas a dejar de funcionar por completo, tal como se explicó en el caso anterior.

Además, se incluyen una serie de recomendaciones para mejorar los resultados:

- Reconstrucción en postura "A" en lugar de "T": Se observa que la reconstrucción del modelo humano en la postura "A" permite que el optimizador SMPL utilizado se ajuste de manera más efectiva a la forma real. En comparación, la reconstrucción en postura "T" ofrece menos precisión y una menor cantidad de puntos en la zona de los brazos, ya que se requiere un mayor número de cámaras para capturar todo el brazo en su totalidad.
- Escaneo de humanos con ropa: Se recomienda que la ropa utilizada durante el escaneo presente variaciones en su tonalidad, ya sea a través de estampados o arrugas. Evitar el uso de camisetas de un solo color plano ayudará al software de reconstrucción a obtener más información y mejorar los resultados.

Una vez conocidas las limitaciones y recomendaciones actuales del escáner, se muestra el proceso completo de varios proyectos realizados.

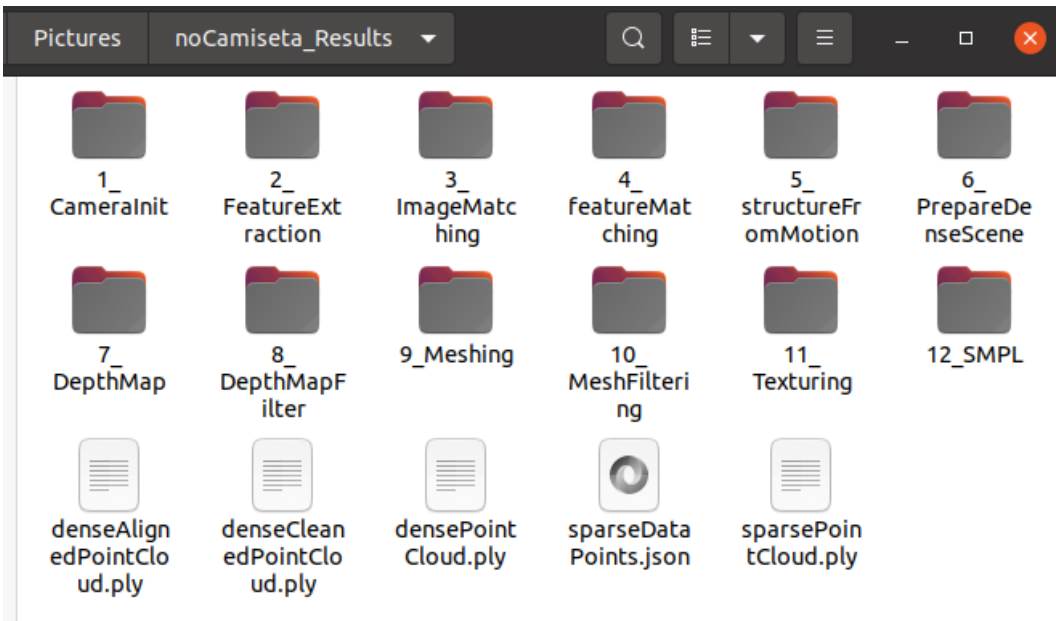
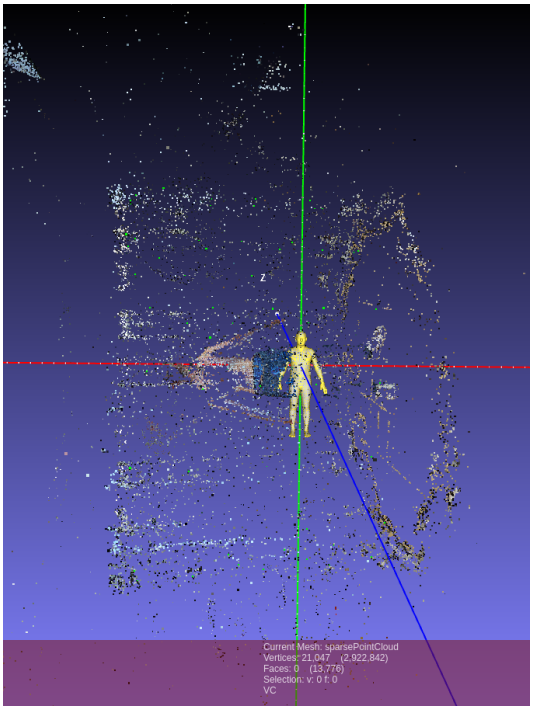
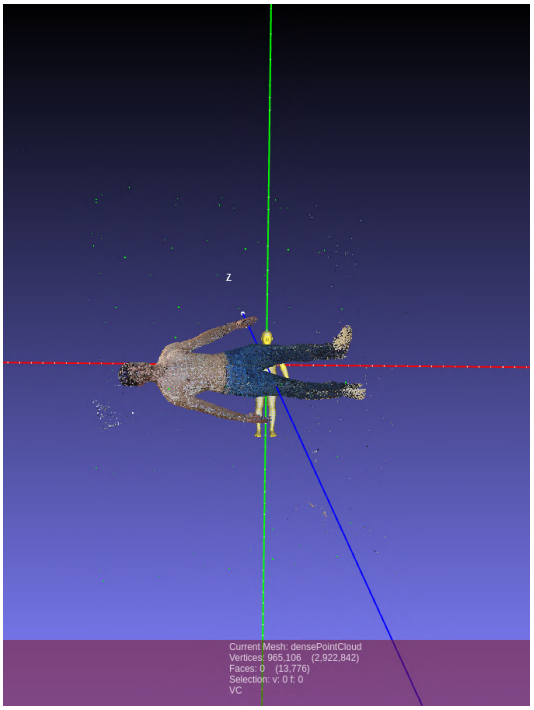


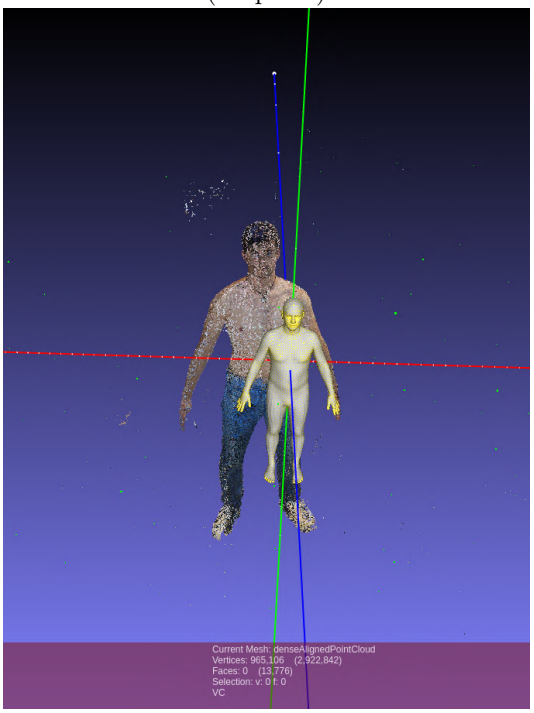
Figura 5.1: Carpetas y archivos resultantes del proyecto "noCamiseta".



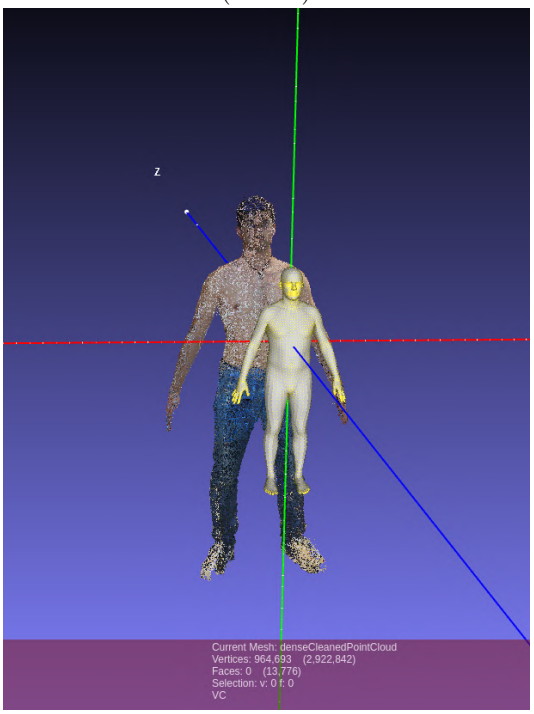
(Dispersa)



(Densa)



(Densa alineada)



(Dense limpiada)

Figura 5.2: Resultados del proyecto "noCamiseta". Proceso de obtención de nube de puntos dispersa y densa, su correspondiente alineamiento con el modelo SMPL de referencia y limpieza de los puntos no deseados.

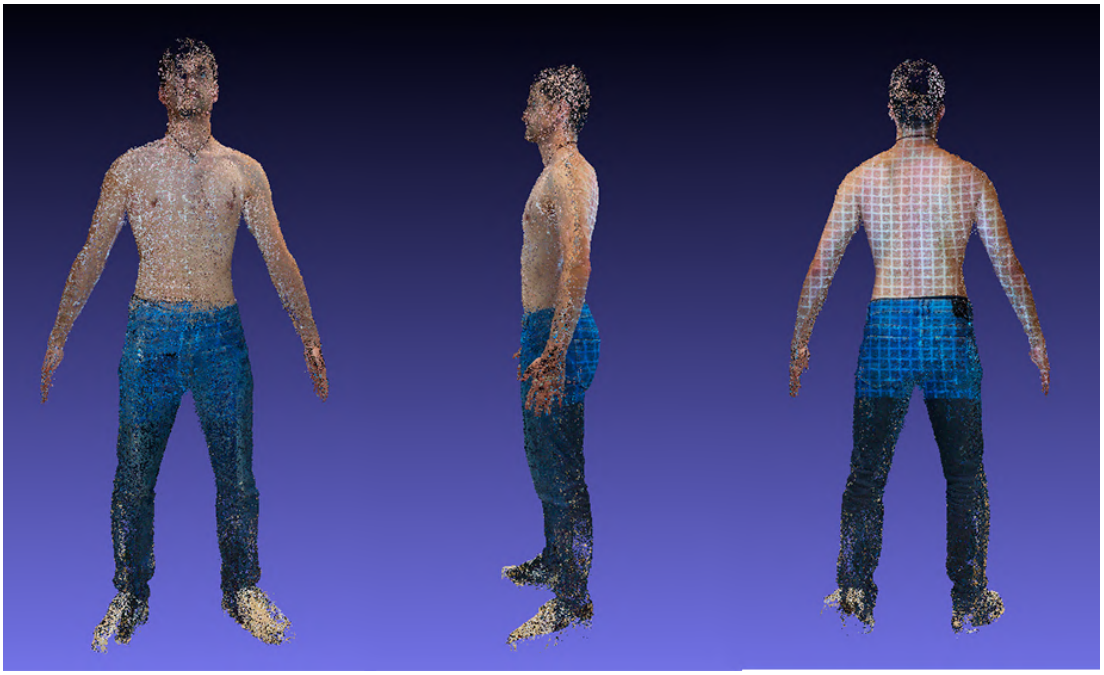


Figura 5.3: Alzado perfil y espalda de la nube de puntos densa.

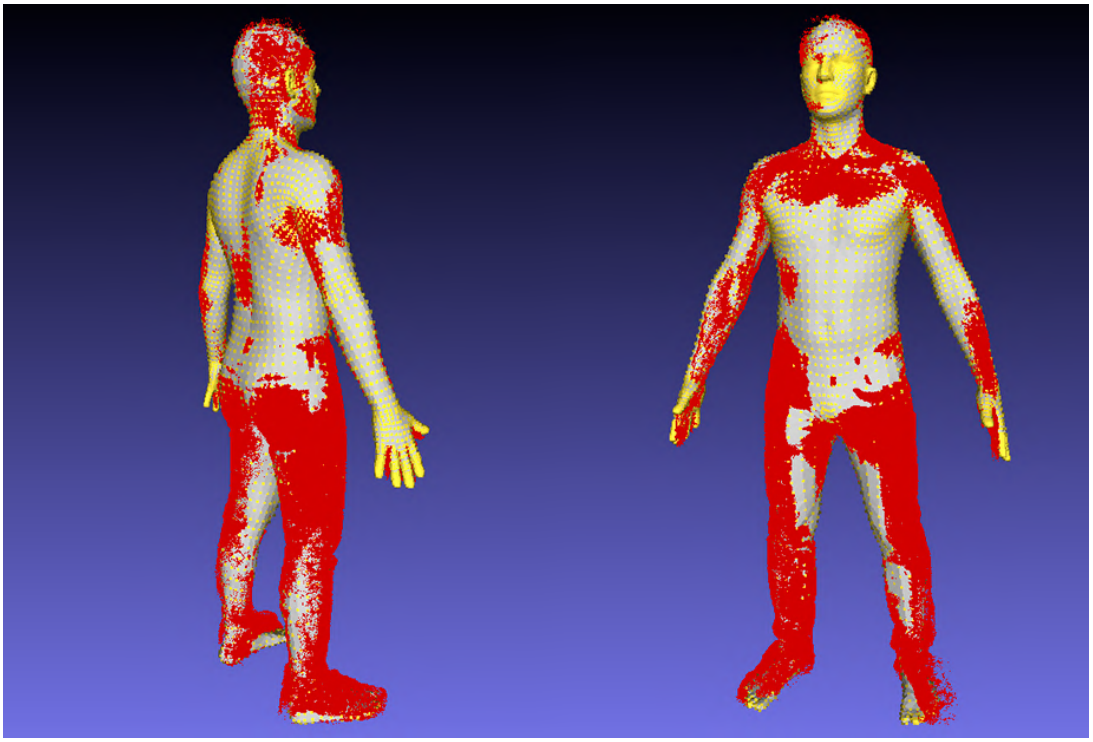


Figura 5.4: Comparación entre la nube de puntos refinada (rojo) superpuesta con la malla SMPL final (amarillo) del proyecto “noCamiseta”

Las figuras 5.2, 5.3 y 5.4 muestran el orden en el que van ocurriendo los hechos hasta llegar al resultado de la malla SMPL final en el proyecto de nombre "noCamiseta". Es importante recordar que, aunque en la figura 5.2 la nube de puntos y la referencia no se encuentran en la misma escala, la función para optimizar la nube a SMPL explicada en el apartado 4.2.3 se encarga de ello. La nube de puntos roja en la figura 5.4 muestra la conversión de escala ya aplicada. Se puede apreciar en la figura 5.3 la mejora en la densidad de puntos en la espalda del humano escaneado al utilizar el patrón proyectado sobre ella respecto al resto de la nube. Este proyecto utiliza en Meshroom ajustes de alta calidad para la extracción de similitudes entre imágenes con el nodo `FeatureExtraction` y para generar los mapas de profundidad en el nodo `DepthMap` sin aplicarles la bajada de resolución por defecto. Estos ajustes generan resultados de mayor calidad y cantidad de información, pero suponen un aumento considerable en el tiempo de procesado.

Las figuras 5.5 y 5.6 pertenecen a dos proyectos diferentes que utilizan los mismo parámetros. El proyecto "noCamiseta" hace uso de mascaras a diferencia de "conIluminado". Se puede apreciar la cantidad de puntos eliminados gracias al uso de estas a la hora de generar la nube de puntos densa. Además, tener que procesar una menor cantidad de puntos, se produce un drástico decremento en el tiempo requerido en las etapas `DepthMap, DepthMapFilter y Meshing`, encargadas de generar la nube de puntos densa.

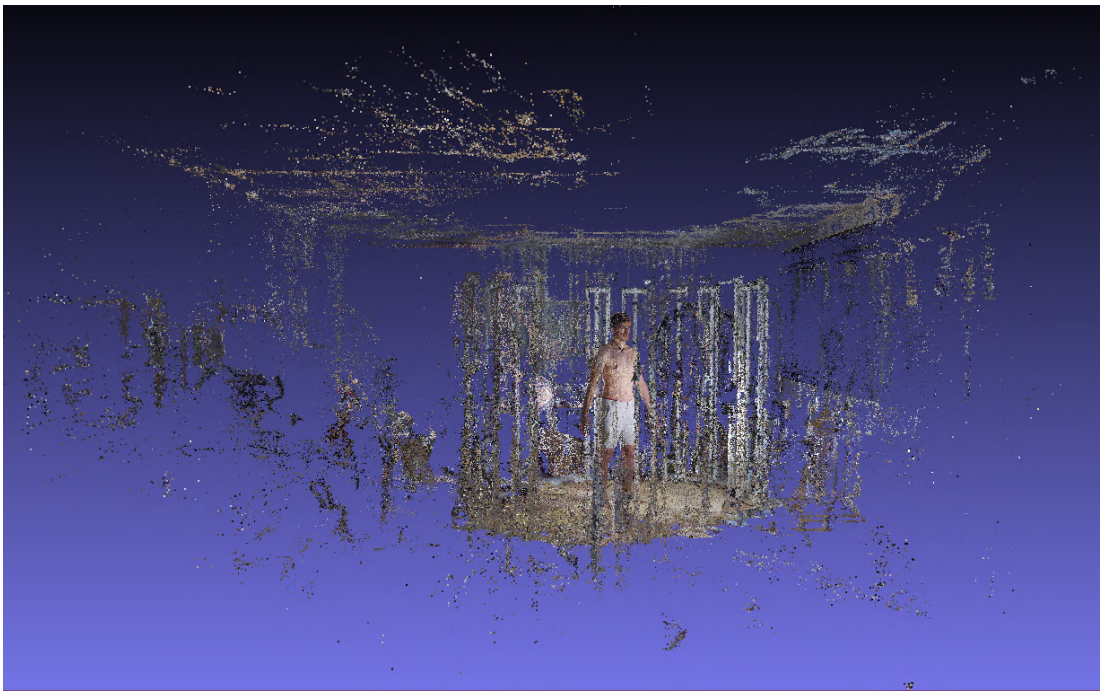


Figura 5.5: Nube de puntos sin hacer uso de mascaras. Proyecto "conIluminado". Número de puntos: 1946840

Figura 5.6: Nube de puntos utilizando mascaras. Proyecto “noCamiseta”. Número de puntos: 646490

Se han realizado diferentes ajustes de calidad para el proyecto ‘noCamiseta” mostrado a lo largo de esta sección y se han registrado los tiempos que ha tardado Meshroom en realizar cada nodo. Cabe destacar que pese a realizar una reconstrucción con los mismos ajustes e imágenes de entrada, no necesariamente tarda lo mismo en ejecutar cada nodo. Esto se debe a que Meshroom utiliza algoritmos como RANSAC, que selecciona un subconjunto aleatorio de puntos de características y ajustan la nube 3D a estos puntos.

La media de los tiempos de los 3 proyectos según su calidad se puede ver en la tabla 5.1. Los diferentes ajustes son los siguientes:

- En el de baja calidad se han utilizado los valores por defecto dados por Meshroom para hacer una reconstrucción. Además, en este ejemplo no se están utilizando mascaras.
- El de alta calidad, se ha asignado al nodo `FeatureExtraction` los valores `Describer density = High` y `Describer quality = High`. Para el nodo

`DepthMap` se ha modificado `Downscale = 1`. Se han realizado las mediciones de tiempo tanto con mascaras como sin ellas.

Calidad del Proyecto	Baja	Alta	Alta con mascaras
CameraInit	0.21s	0.19s	0.27s
FeatureExtraction	4m 48s	10m 22s	12m 50s
ImageMatching	0.12s	0.11s	0.16s
FeatureMatching	12m 42s	21m 12s	24m 17s
StructureFromMotion	1m 22s	1m 38s	3m 26s
PrepareDenseScene	1m 14s	1m 16s	2m 58s
DepthMap	14m 47s	1h 20m 7s	52m 53s
DepthMapFilter	4m 53s	19m 56s	12m 34s
Meshing	7m 29s	1h 54m 21s	27m 17s
MeshFiltering	24.44s	40.31s	49.42s
Texturing	4m 53s	7m 31s	4m 48s
Tiempo total	0h 52m 33s	4h 17m 4s	2h 21m 52s

Tabla 5.1: Tiempo por nodo de un proyecto con diferentes ajustes.

Se puede apreciar como en los nodos `DepthMap, DepthMapFilter y Meshing` el tiempo requerido sin el uso de mascaras, además de generar mucha mas suciedad en la nube de puntos, tarda casi el doble respecto al proyecto que las usa.

Para finalizar, en la figura 5.7 se muestra los resultados de otro proyecto de la nube de puntos refinada y su correspondiente malla SMPL generada.

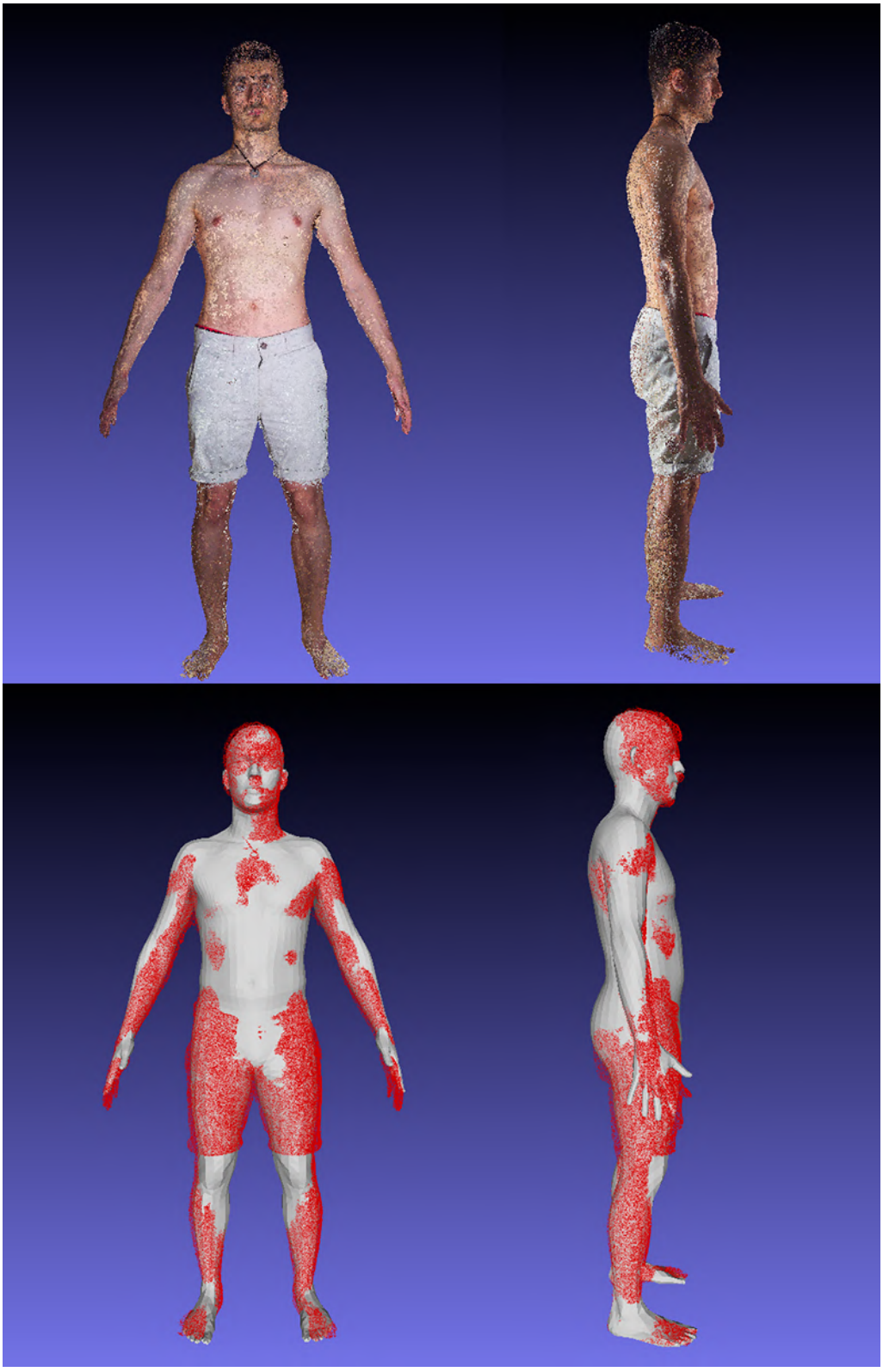


Figura 5.7: Proyecto “semiCuerpo”.

6

Conclusiones y trabajos futuros

En este proyecto se han abarcado múltiples competencias necesarias para la creación del escáner y la posterior automatización de la reconstrucción. Además de los conocimientos obtenidos en el grado, ha sido necesario aprender sobre los siguientes ámbitos para llegar al resultado obtenido:

- Conocimientos sobre fotografía para sacar el máximo provecho a las capturas realizadas, así como conocimientos sobre el funcionamiento de una cámara y la forma en que trabajan los datos obtenidos del sensor.
- Conocimientos sobre el uso de múltiples imágenes en fotogrametría para la obtención de la nube de puntos en la reconstrucción 3D, así como su posterior tratamiento y conversión a la malla final.
- Conocimientos sobre electrónica para las conexiones de todo el cableado del escáner, principalmente las tiras LED en la iluminación.
- Conocimientos sobre arquitecturas cliente-servidor y transferencia de datos entre múltiples dispositivos en LAN además del funcionamiento de un switch gestionable.
- Conocimientos sobre sistemas operativos basados en Linux como Ubuntu para el servidor y Raspbian para las Raspberry Pi y su posterior configuración para el arranque automático y diferentes funcionalidades mencionadas a lo largo del documento.

Este proyecto no deja de ser la parte inicial de un proyecto mucho mas ambicioso que busca obtener a partir del escaneado de un humano, un modelo SMPL completamente texturizado del que se es capaz de conocer como incide la luz desde diferentes puntos iluminados (que se obtendría de los diferentes conjuntos de iluminación obtenidos en este proyecto), con el objetivo de utilizarse en entornos virtuales como videojuegos o probadores de ropa en linea.

Para mejorar el proyecto en vistas al futuro, se puede tener en cuenta los siguientes puntos:

- Funcionamiento correcto en la iluminación del escáner.
- Eliminar los problemas de alimentación y daño en las tarjetas SD.
- Mejora en el sistema de detección de articulaciones. Actualmente el sistema no es lo suficientemente robusto como para ser utilizado, por lo que se utiliza el otro algoritmo mencionado.
- Depurar algunos detalles tanto visuales como técnicos de la aplicación creada.
- Añadir un nuevo paso para normalizar la densidad de puntos de la nube refinada puede ayudar a la función de optimización SMPL a generar mejores resultado.

Actualmente el proyecto finaliza con la obtención de un modelo SMPL del humano escaneado. En proyectos futuros, se buscara la forma de texturizar dicho modelo. Para ello, dado que en el archivo `sparseDataPoints.json` se almacena las coordenadas de las imágenes que han ayudado a generar cada punto de la nube de puntos, se puede generar un sistema de ecuaciones con el color de dichos puntos en todas las imágenes de la misma cámara de los diferentes conjuntos de iluminaciones. Este sistema resolverá la incógnita de cual es el color del punto en una iluminación concreta, así como poder deducir cual seria el color para el punto con una iluminación próxima. De esta forma se puede tener un modelo SMPL texturizado del que se puede conocer como interactúa la iluminación global en tiempo real. Existen algunos como el de Julian Chibane y Gerard Pons-Moll [35] que pretenden abordar problemas parecidos, aunque la resolución de este trabajo futuro sera un desafió en el estudio de la mejor solución para ello.

Bibliografía

[1] "Pi3dscan." [Online]. Available: https://www.pi3dscan.com/

[2] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, "SMPL: A skinned multi-person linear model," *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, vol. 34, no. 6, pp. 248:1–248:16, Oct. 2015.

[3] ——, "Smpl: A skinned multi-person linear model," *ACM Trans. Graph.*, vol. 34, no. 6, nov 2015. [Online]. Available: https://doi.org/10.1145/2816795.2818013

[4] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. New York, NY, USA: Cambridge University Press, 2003.

[5] C. T. H.-P. D. W. S. M. S. Paul Debevec, Tim Hawkinsy, "The light stages and their applications to photoreal digital actors," *SIGGRAPH 2000 Conference Proceedings*, 2000.

[6] P. Debevec, "Acquiring the reflectance field of a human face," *University of Southern California Institute for Creative Technologies*, 2012.

[7] 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," in *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 10 975–10 985.

[8] V. Blanz and T. Vetter, "A morphable model for the synthesis of 3d faces," in *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '99. USA: ACM Press/Addison-Wesley Publishing Co., 1999, p. 187–194. [Online]. Available: https://doi.org/10.1145/311535.311556

[9] T. Li, T. Bolkart, M. J. Black, H. Li, and J. Romero, "Learning a model of facial shape and expression from 4D scans," *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, vol. 36, no. 6, pp. 194:1–194:17, 2017. [Online]. Available: https://doi.org/10.1145/3130800.3130813

[10] H. Joo, T. Simon, and Y. Sheikh, "Total capture: A 3d deformation model for tracking faces, hands, and bodies," *CoRR*, vol. abs/1801.01615, 2018. [Online]. Available: http://arxiv.org/abs/1801.01615

[11] J. Romero, D. Tzionas, and M. J. Black, "Embodied hands," *ACM Transactions on Graphics*, vol. 36, no. 6, pp. 1–17, nov 2017. [Online]. Available: https://doi.org/10.1145%2F3130800.3130883

[12] C. Cao, Y. Weng, S. Zhou, Y. Tong, and K. Zhou, "Facewarehouse: A 3d facial expression database for visual computing," *IEEE transactions on visualization and computer graphics*, vol. 20, pp. 413–25, 03 2014.

[13] *48-Port Gigabit PoE+ Smart Managed Pro Switch With 4 SFP Ports*. [Online]. Available: https://www.downloads.netgear.com/files/GDC/GS752TPv2/GS752TPv2_GS752TPP_IG_EN.pdf

[14] Raspberry, "Raspberry pi 3 model b+." [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/

[15] Adafruit, *WS2812B Intelligent Control LED.* [Online]. Available: https://cdn-shop.adafruit.com/datasheets/WS2812B.pdf

[16] T. P. T. P. Kristijan Bartol, David Bojanic, "A review of body measurement using 3d scanning," *University of Zagreb, Faculty of Electrical Engineering and Computing, Department of Electronic Systems and Information Processing.*

[17] D. Mills, "Network time protocol (ntp)," *RFC EditorUnited States*, 1985.

[18] M. V. Markus Ullmann, "Delay attacks — implication on ntp and ptp time synchronization," *2009 International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, 2009.

[19] "Precision time protocol (ptp)." [Online]. Available: https://endruntechnologies.com/pdf/PTP-1588.pdf

[20] A. B. Glev Alexandrov, "Photogrammetry course: Photoreal 3d with blender and reality capture." [Online]. Available: https://creativeshrimp.gumroad.com/l/photogrammetry-course

[21] Raspberry, "Raspberry pi camera module 2." [Online]. Available: https://www.raspberrypi.com/products/camera-module-v2/

[22] D. Jones, "Picamera library." [Online]. Available: https://picamera.readthedocs.io/en/release-1.13/index.html

[23] Raspberry, "Picamera2 library." [Online]. Available: https://github.com/raspberrypi/picamera2

[24] C. Griwodz, S. Gasparini, L. Calvet, P. Gurdjos, F. Castan, B. Maujean, G. D. Lillo, and Y. Lanthony, "Alicevision Meshroom: An open-source 3D reconstruction pipeline," in *Proceedings of the 12th ACM Multimedia Systems Conference - MMSys '21.* ACM Press, 2021.

[25] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2," https://github.com/facebookresearch/detectron2, 2019.

[26] R. Girshick, "Fast r-cnn," 2015.

[27] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 640–651, 2017.

[28] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004. [Online]. Available: http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94

[29] P. Fernández Alcantarilla, "Fast explicit diffusion for accelerated features in nonlinear scale spaces," 09 2013.

[30] C. Allene, J.-P. Pons, and R. Keriven, "Seamless image-based texture atlases using multi-band blending," in *2008 19th International Conference on Pattern Recognition*, 2008, pp. 1–4.

[31] M. Fischler and R. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981. [Online]. Available: /brokenurl#http://publication.wilsonwong.me/load.php?id=233282275

[32] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, "Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization." *ACM Trans. Math. Softw.*, vol. 23, no. 4, pp. 550–560, 1997. [Online]. Available: http://dblp.uni-trier.de/db/journals/toms/toms23.html#ZhuBLN97

[33] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[34] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018.

[35] J. Chibane and G. Pons-Moll, "Implicit feature networks for texture completion from partial 3d data," 2020.