Departamento de Electrónica, Instrumentación y Control
F.I.E.T

RECONSTRUCCIÓN 3D DE UNA LACERACIÓN EN TEJIDO EPIDÉRMICO PARA CIRUGÍA ASISTIDA POR COMPUTADOR

Tesis:

Para Optar Por El Título De Ingeniero en Automática Industrial

FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES

DEPARTAMENTO DE ELECTRÓNICA, INSTRUMENTACIÓN Y CONTROL

TESIS:

RECONSTRUCCIÓN 3D DE UNA LACERACIÓN EN TEJIDO EPIDÉRMICO PARA CIRUGÍA ASISTIDA POR COMPUTADOR

Por:
Juan Diego Gutiérrez Torres
Christian David Vallejo Constain

DIRECTOR:
PhD. Maximiliano Bueno López
COODIRECTOR:
Mg. Hermes Fabián Vargas Rosero

RECONSTRUCCIÓN 3D DE UNA LACERACIÓN EN TEJIDO EPIDÉRMICO PARA CIRUGÍA ASISTIDA POR COMPUTADOR

Tesis:

Para Optar Por El Título De Ingeniero en
Automática Industrial
Marzo, 2022

Por
Juan Diego Gutiérrez Torres
Christian David Vallejo Constain

Foto de portada: Área de prensa Unicauca, 2022
Publicado por: Unicauca, F.I.E.T, Tulcan, Colombia

Aprobación

Juan Diego Gutiérrez Torres
Christian David Vallejo Constain-

.....

Jurado 1

.....

Jurado 2

Introducción

En este proyecto se busca llevar a cabo la reconstrucción 3D en un entorno virtualizado (Realidad Aumentada) de laceraciones en tejido epidérmico, las cuales se plasmaron en una almohadilla de pruebas construida de forma artesanal en materiales de fácil acceso, tales como Foamy. Para lograr el objetivo general de este proyecto se implementó un escenario que simula un quirófano como se puede apreciar en el tercer capítulo de este documento y así dar por alcanzado el primer objetivo específico de este trabajo. Para poder llevar a cabo la reconstrucción de las heridas en un entorno virtual, se usó una cámara de bajo costo, teniendo en cuenta la relación costobeneficio sus prestaciones son algo limitadas ya que su principal uso es para visión de maquina en procesos industriales (pick and place, clasificación, etc), se tuvo en cuenta un software para el procesamiento y tratamiento de la imagen, ayudándonos de algoritmos con el time of Flight y SMF, con esto se obtiene el segundo objetivo del proyecto, logrando extraer características físicas de tamaño de la herida como largo, ancho y profundidad. A través del uso de la cámara y la implementación de un script de captura de imagen, se logra guardar el modelo de la herida en escala de grises; que para darle textura y color al modelo se realiza una tarea de procesamiento de imagen para posteriormente tener una reconstrucción exitosa en el entorno de realidad aumentada, permitiendo visualizar el modelo 3D sobre un marcador, pudiendo ser apreciado para este caso en cualquier dispositivo Android que soporte la reproducción de aplicativos de Realidad Aumentada, cumpliendo así el tercer objetivo específico del trabajo y dando por finalizado el cumplimiento del objetivo general del proyecto.

Abstract

In this project, the 3D reconstruction of several wounds was carried out, which were reflected in a test pad built by hand in easily accessible materials, such as Foamy, silicone, scalpel. To achieve the general objective of this project, a scenario that simulates an operating room was implemented, as can be seen in the third chapter of this document and in this way to achieve the first specific objective of this work. In order to carry out the reconstruction of the wounds in a virtual environment, a low cost camera was used, which is the IntelRealsense D435i, given this its benefits are somewhat limited since its main use is for machine vision in industrial processes (pick and place, classification), a software was taken into account for the processing and treatment of the image, helping us with algorithms with the time of Flight and Slam, with this the second objective of the project is obtained, managing to extract characteristics of the wound such as length , width and depth. Through the use of the IntelRealsense D435i camera and an image capture script, it is possible to save the grayscale wound model in a format. ply. To give texture and color to the model, it was processed in the Blender software to later have a successful reconstruction in the augmented reality environment that works in the Vuforia development software, allowing us to visualize the 3d model in a marker, being able to be appreciated for this case in any Android device with a camera, thus fulfilling the third specific objective of our work.

Agradecimientos

Juan Diego Gutiérrez Torres
Christian David Vallejo Constain

Juan Diego Gutierrez Torres, Ing. en Automática Industrial, Universidad del Cauca

Agradezco a nuestros conocidos porque de manera indirecta han contribuido en nuestra formación ya sea como profesionales o han aportado a nuestro crecimiento personal.

En especial a mi madre, a mi familia y amigos, que me han brindado un soporte constante a lo largo de esta vida y siempre han estado presentes en los malos y buenos momentos y quiero decirles que han sido un motor fundamental, el cual me llenó de inspiración para poder llevar a cabo la culminación de una nueva etapa y dar un paso más adelante.

A mi alma mater por permitir adquirir y poner en práctica los conocimientos y habilidades aprendidas dentro de sus aulas y aportarnos a un excelente cuerpo docente capacitado y dispuesto a trasmitir sus saberes.

Christian David Vallejo Constain, Ing. en Automática Industrial, Universidad del Cauca

Primero que todo dar gracias a Dios por todas las obras que ha realizado en mí, A mis padres por ser ese apoyo incondicional a pesar de las adversidades, a mi prometida Luisa por ser el amor y fuerza para salir adelante.

También agradecer a mis familiares, amigos y conocidos por siempre brindar esa voz de aliento para lograr este gran paso en mi vida.

A nuestros tutores **Maximiliano Bueno**, Phd, Director de tesis, **Hermes Fabian Vargas**, candidato Phd, Co-Director de tesis y cuerpo de profesionales que forman parte del estamento de profesores por brindarnos los conocimientos necesarios y aportar su granito de arena para la finalización de este proyecto.

Lista de Contenidos

Lista de Tablas

Lista De Figuras

Lista de Códigos

1 Introducción

La caracterización y medición de heridas y laceraciones por medio de recursos y habilidades desarrolladas en el campo de la ingeniería, proveen al personal médico de una serie de herramientas que les permiten influir directamente en la realización de un adecuado diagnóstico, tratamiento y apoyo en las investigaciones médicas [1] que impulsan las tecnologías de tratamiento, debido a que los cambios en estos parámetros físicos de la herida significan indicaciones en el proceso de cicatrización y recuperación. Sin embargo, los métodos convencionales de medición se basan en enfoques simples de mediciones de área apoyados en reglas o en sistemas no invasivos de fotografías [2]. Por lo tanto, como un método no invasivo y confiable de medición, la reconstrucción de heridas como modelo 3D facilita el análisis complejo de características relevantes y objetivas, con ventajas tanto para los pacientes como para los médicos, evitando procedimientos incómodos y dolorosos para los primeros y siendo más convenientes para los segundos [2]. Esta reconstrucción de laceraciones y heridas simuladas en un entorno virtual de 3 dimensiones abre la posibilidad de realizar mediciones volumétricas de altura, profundidad y diferencias en las superficies de las cicatrices o lesiones escaneadas, lo cual representa una ventaja para el diagnóstico de datos y la evaluación de las acciones a tomar por un posterior sistema quirúrgico, siendo aplicado de esta forma en campos como la colaboración médica y la telemedicina.

1.1 Motivación

La extracción y reconstrucción de estructuras y mapas en 3D son aplicaciones dentro del campo de la visión por computadora que obtienen cada vez más relevancia debido a que sus resultados enriquecen diversas áreas de la ingeniería y la ciencia en sectores como el industrial para manufactura, el aeroespacial, el automotriz y el biomédico entre otros. Sin embargo, gran porcentaje de los aportes realizados en el campo han sido realizados por grandes industrias de la tecnología de visión o en laboratorios de alta tecnología de universidades de prestigio, los cuales tienen en común la disponibilidad de altos presupuestos de inversión para investigación y desarrollo. El actual trabajo pretende minimizar la brecha entre los tipos de investigaciones mencionados anteriormente y las investigaciones realizadas en universidades, de igualmente alto nivel, públicas, al demostrar resultados satisfactorios en la obtención de modelos 3D de heridas y laceraciones en un entorno virtualizado y su aplicación en ámbitos de la medicina, con recursos alcanzables, ya que con los avances tecnológicos en sistemas de computación y con la disminución progresiva de los precios de hardware y software que implican convertir una imagen bidimensional en una forma tridimensional, es posible alcanzar los objetivos y, con esto, realizar un aporte importante en el área de estudio al programa de ingeniería en automática industrial y a nuestra alma máter en general.

1.2 Objetivos

Proponer un sistema de reconstrucción 3D de una laceración en tejido epidérmico basado en extracción de características. Parar lograr este objetivo, se presentan a continuación los objetivos específicos:

1.2.1 Objetivos específicos

- Proponer un mecanismo de captura y extracción de características de la laceración.
- Seleccionar un algoritmo para la reconstrucción 3D de una laceración en la piel.
- Realizar una aplicación de realidad aumentada que permita registrar el detalle de la reconstrucción 3D.

1.3 Estructura del documento

El presente documento está dividido en cinco capítulos, incluido el actual, los cuales siguen un orden de desarrollo determinado, empezando por la introducción de la base conceptual hasta llegar a los elementos que constituyen la implementación de un sistema de reconstrucción 3D de una laceración en tejido epidérmico basado en extracción de características físicas de tamaño, terminando con conclusiones y trabajos futuros. En el Capítulo II, se expone la base teórica de los conceptos necesarios para lograr la implementación del sistema de reconstrucción. En este capítulo están documentados los conceptos, su principio de funcionamiento y la explicación secuencial del proceso de reconstrucción, desde el procesamiento de imagen (obtención de mapa de profundidad), pasando por los descriptores de movimiento (matriz de rotación R y vector de traslación T), hasta la creación del modelo 3D de la herida (determinación de la nube de puntos y generación de la malla). También se presentará un conjunto de posibles técnicas existentes hasta la actualidad que concluyen en la obtención de dicho modelo.

En el capítulo 3 se explica el hardware y software necesario para la implementación del sistema de pruebas, esto con el fin de poder realizar la reconstrucción de la laceración en un entorno virtualizado, teniendo en cuenta que va a ser un sistema de low cost.

En el capítulo 4 se muestran los resultados obtenidos a partir de la implementación del sistema de captura y extracción de características físicas de la herida y se puede visualizar el resultado de la reconstrucción 3D de la laceración y por último, en el capítulo 5, se abordan las conclusiones obtenidas después de la investigación y realización del proyecto.

2 Marco Teórico

La necesidad de estimar, medir y registrar los aspectos generales de los pacientes y aspectos locales de las heridas, ¡surgen de la utilidad que estas representan en el proceso de valoración, diagnóstico y posterior tratamiento de los pacientes; consiguiendo predecir las probabilidades de cicatrización a la vez que hace posible valorar, de manera más confiable, la eficacia de los tratamientos aplicados. Un modelo obtenido a partir de métodos no invasivos, representa una herramienta ventajosa en el abordaje del seguimiento y estudio de los pacientes que han sufrido heridas y laceraciones en alguna parte de su cuerpo. Entre las diversas investigaciones de visión por medio de sistemas y algoritmos de visión por computadora, se encuentra el área de visión tridimensional de objetos basada en la reconstrucción de objetos. Estas investigaciones han sido desarrolladas desde hace décadas en el Centro de Investigaciones en Óptica, A. C. [3]. Es posible clasificar estos sistemas en dos conjuntos o ramificaciones principales, a saber: aquellos que utilizan solamente una cámara y aquellos que incluyen dos o más, denominados sistemas multicámara o de estéreo imágenes, para los cuales, se entrará en detalle más adelante en esta sección. Sin embargo, cabe resaltar que, para la extracción de estructuras en escenas tridimensionales, el trabajo inicial fue desarrollado principalmente por la comunidad Image Understanding (IU) y financiado por la Agencia de Proyectos de Investigación Avanzada (ARPA) a finales de la década de los 70´s e inicios de los 80's [4].

En este capítulo se presenta la base conceptual necesaria para el entendimiento del proceso de reconstrucción de estructuras en 3D a partir de imágenes y los diferentes elementos que componen estos sistemas, así como la formulación matricial de las operaciones realizadas con las imágenes procesadas.

2.1 Proceso de reconstrucción de volúmenes en 3D a partir de imágenes

El denominado proceso de reconstrucción 3D hace referencia a la reproducción física de objetos reales en la memoria de un sistema de computación [5].Por lo tanto, esta reproducción, se encuentra en la capacidad de mantener características físicas del objeto, tales como morfología, tamaño (ancho, largo y profundidad) y textura entre otras.

En la literatura actual, existe una cantidad considerable de propuestas para la reconstrucción de objetos 3D, las cuales se diferencian por el número de etapas descritas o por variantes en algunas de estas. Por ejemplo, el sistema de análisis de escenas estereoscópicas presentado por Koch en [6], la propuesta de Remondino en [7] o el proceso de reconstrucción 3D expuesto en [8] por Pollefeys. Uno de los estudios que recopila una mayor cantidad de información acerca de la reconstrucción 3D, de forma sintética y categórica, se puede encontrar en [9]. En él, López ordena por etapas el proceso de reconstrucción de una superficie 3D

en: una primera etapa de registro de vistas parciales, o adquisición (donde se calcula una referencia común), una etapa de integración de las vistas, integración y registro (donde se obtiene una representación matemática única de la superficie) y una etapa posterior de procesamiento [10].

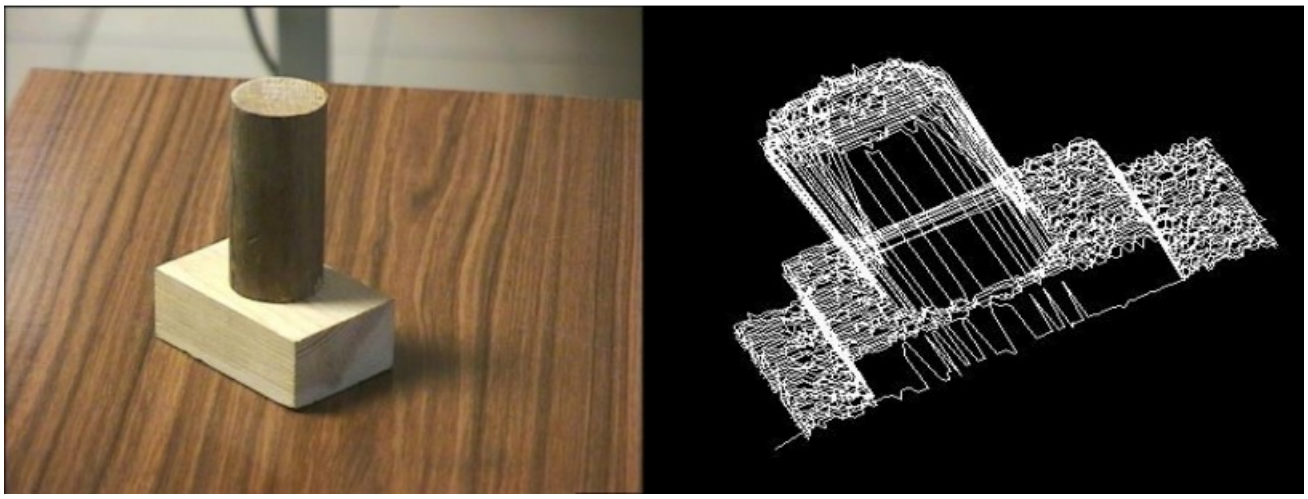


Figura 2.1: Sólido generado desde una imagen tomado de [9]

2.2 Etapas del proceso de reconstrucción

Para realizar la reconstrucción del objeto es necesario seguir una serie de etapas que serán presentadas a continuación.

2.2.1 Adquisición o registro de imágenes

En [3], Camacho esclarece la etapa de adquisición, o etapa primera, al presentar como antecedente la división de dos ramas principales en la obtención de imágenes; a saber, los sistemas en los que se utiliza solo una cámara y los que involucran un arreglo de dos o más denominado sistema de visión multicámara y, resalta el hecho de que los sistemas multicámara son más eficaces que los otros, aunque representan un aumento en los costos de montaje experimental (como es posible inferirlo). Sobre este punto, investigaciones como la de Ceballos en [11], aseguran que la correspondencia estéreo entre fotogramas tienen como resultado mayor robustez y aplicación exitosa en cuanto a extracción de geometrías a partir de imágenes, ya que una sola imagen no es suficiente, mientras el solapamiento y alineación de diferentes vistas permite calcular el sistema de referencia común antes mencionado. De esta etapa dependerán en un futuro la calidad de los resultados en los procesos de segmentación y reconocimiento de objetos [9], [12].

1. **Visión estereoscópica:**
   La visión estereoscópica es la que pretende reconstruir un escenario en 3 dimensiones a partir de la adquisición de dos o más imágenes, cabe añadir que un número mayor de imágenes significa una mayor precisión del modelo. El proceso es mejor conocido como ‘triangulación’ y basa su idea en el funcionamiento del ojo humano, el cual obtiene una percepción de las características espaciales a través de este procedimiento [13].

Se pueden describir los objetos tridimensionales a reconstruir mediante coordenadas tradicionales cartesianas (X, Y, Z) para denotar puntos que, para esta área de visión por computadora, son equivalentes a píxeles, con los cuales se realizan los cálculos y operaciones denominadas transformaciones de rotación, traslación y escalado necesarias para reconstruir los modelos 3D requeridos [13]. Por lo tanto, para una mejor comprensión de la visión estereoscópica, a continuación, se presenta la teoría correspondiente. Morato en [13], se basa en la exposición de la segunda edición de "Digital Image Processing" [14] para explicar las trasformaciones en visión por computadora. Por lo tanto, en esta sección se expondrán estas bajo esta bibliografía.

2. **Transformación de traslación:**
   Al suponer que se pretende trasladar un punto de coordenadas (X, Y, Z) a otra nueva localización mediante un desplazamiento (X_0, Y_0, Z_0). Según [14], dicha operación puede conseguirse usando las ecuaciones:

$$x' = x_0 + x; \;\; y' = y_0 + y; \;\; z' = z_0 + z \tag{2.1}$$

donde (X', Y', Z') son las coordenadas del nuevo punto. La ecuación anterior puede expresarse de manera matricial mediante esta otra nueva expresión:

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 1 & z_0 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2.2}$$

A veces es útil concatenar diversas transformaciones para producir un resultado compuesto, como traslaciones, seguidas de rotaciones y luego de escalado. Si usamos matrices cuadradas se simplificará notablemente la representación del proceso. Así reescribiendo 2.2 se obtiene:

$$\begin{bmatrix} X' \\ Y' \\ Z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 1 & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.3}$$

De aquí en adelante se usará la siguiente representación para denotar el proceso de traslación:

$$v' = Tv \tag{2.4}$$

donde v es el vector columna original conteniendo las coordenadas originales, v' es el vector columna cuyas componentes son las coordenadas transformadas y manteniendo esta notación, la matriz utilizada para la traslación es T:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = v'; \quad \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = v; \quad \begin{bmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 1 & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = T \tag{2.5}$$

3. **Transformación de escalado:**
   Las operaciones de escalado pueden representarse de manera sencilla mediante la siguiente matriz [14], donde $S_x, S_y y S_z$ son los factores de escalado a aplicar sobre los ejes X, Y, Z respectivamente.

$$S = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & 0 & S_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

4. **Transformación de rotación:**
   Las transformaciones usadas para realizar rotaciones en 3D son más complejas que las explicadas hasta el momento. Si se desea rotar un punto sobre otro arbitrario en el espacio se procede a 3 transformaciones: la primera traslada el punto arbitrario al origen, la segunda implementa la rotación y la tercera traslada el punto a su posición inicial [14]. Las correspondientes matrices de rotación para cada eje de coordenadas X, Y, Z son respectivamente:

$$R_\alpha = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

$$R_\beta = \begin{bmatrix} \cos\beta & 0 & -\sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.8}$$

$$R_\theta = \begin{bmatrix} \cos\Theta & \sin\Theta & 0 & 0 \\ -\cos\Theta & \cos\Theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.9}$$

   donde θ, α y β se miden en el sentido contrario de las agujas del reloj como puede apreciarse en la figura 2.2

   Para realizar diversas transformaciones, éstas pueden representarse mediante una matriz $4x4$. Por ejemplo, al trasladar, escalar y rotar un punto v sobre el eje Z se realiza la siguiente operación matricial $v' = A_v$, donde $A = R_\Theta ST$. Hay que tener muy en cuenta que se está trabajando con matrices, luego el orden en que se realicen las transformaciones es importante.

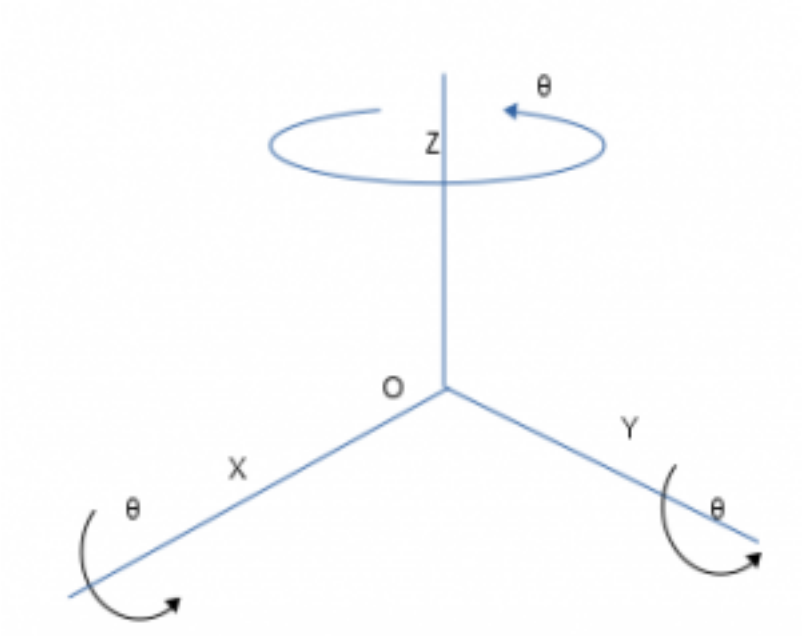


Figura 2.2: Rotación sobre los ejes coordenados

Todo lo explicado anteriormente puede aplicarse también para el caso de m puntos [14]. Sencillamente se han de cambiar los vectores v y v' por matrices $4xm$, de manera que en cada columna de estas matrices se obtienen los m puntos sobre los que se desea realizar las transformaciones.

Algunas de las transformaciones explicadas hasta el momento tienen matrices inversas, las cuales realizan la operación contraria. Como ejemplo de estas matrices se muestran las matrices inversas de traslación T^{-1} y rotación R_{Θ}^{-1}. Las matrices inversas de transformaciones más complejas se obtienen normalmente mediante técnicas numéricas.

5. **Integración de las vistas:**
   La etapa de integración permite obtener una representación continua de toda la superficie del objeto, por lo tanto, en esta etapa se consigue la disminución de información redundante causada por tomas parecidas con poca variación [10] y se logra a su vez rellenar las regiones con ausencia de datos [9]. El resultado de la etapa de integración se conoce como nube de puntos, la cual es un conjunto de coordenadas volumétricas (xyz) las cuales representan la superficie externa del objeto. No obstante, estas nubes no proveen información geométrica ni topológica de la superficie, razón por la cual deben ser procesadas y modeladas en representaciones poligonales denominadas mallas. La conversión de una nube de puntos en un modelo poligonal o de malla de 3 dimensiones es al que se le denomina como proceso de reconstrucción [9].

2.2.2 Procesamiento

Por último, en la etapa de procesado se realizan las operaciones que suavizan, detectan y eliminan los elementos erróneos y rellenan los huecos de las reconstrucciones [9]. Este procesamiento de la malla poligonal es posible realizarlo de

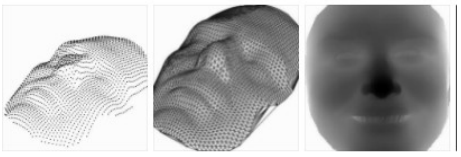


Figura 2.3: integración de las vistas, tomado de [15]

forma manual o mediante algoritmos de detección y corrección automatizados, tal y como lo expone Ju en [16]. El resultado de la etapa final es objeto de especial interés por su aplicación en diferentes tipos de sistemas metrológicos y de diseño. Para este caso en particular, se pretende utilizar la reconstrucción de una laceración como modelo para, posteriormente, determinar los puntos de entrada salida adecuados de una sutura ejecutada por un robot quirúrgico.

Con respecto a los algoritmos necesarios, en la revisión de la literatura se han encontrado una cantidad considerable de estudios sobre la reconstrucción de volúmenes en tres dimensiones a partir de imágenes, sin embargo, como mencionan los autores de [17] entre todas las teorías y métodos es posible resaltar los tres que han tenido una mayor relevancia en las investigaciones, estas son: localización y mapeo simultáneos (SLAM), estructura a partir del movimiento (SfM) y algoritmos estéreo de múltiples vistas (MVS) como los desarrollados en [18] y [19]. Los mismos autores también mencionan la resolución de las imágenes y la cantidad de imágenes utilizadas en el procesamiento como un factor no despreciable que influye en los tiempos de cálculo y en el coste computacional, lo que al final representa una limitación de estos algoritmos para ser utilizados en aplicaciones de reconstrucción de alta velocidad. No obstante, a pesar de ser estos algoritmos los más destacados, existen investigaciones con resultados sobresalientes. Como ejemplo, en [19] se desarrolla un esquema asistido por computador capaz de reconstruir un cuerpo volumétrico a partir de un arreglo de puntos implementando la técnica conocida como Sheet Of Light (SOL) u hoja de luz en español, de la cual llama particularmente la atención que el procesamiento de las imágenes es desarrollado y ejecutado en el software de ingeniería MatLab a diferencia de muchas otras investigaciones que se encuentran desarrolladas en Python. Incluso estudios más recientes, como el de NieBner and Zolh en [20], han contribuido con sistemas en línea para la reconstrucción volumétrica basada en estructuras de datos de memoria y velocidad que agilizan el cálculo que involucra la fusión gradual en un modelo 3D de los mapas de profundidad obtenidos por medio de cámaras, abriendo paso a aplicaciones como realidad aumentada, guías autónomas para reconstrucción de robots o la generación de comentarios inmediatos para el usurario durante el proceso de escaneo 3D [20].

1. **Simultaneous Localization and Mapping SLAM**

   La ubicación y la simultanea obtención del mapa de desplazamiento, en el contexto de la robótica para sistemas en entornos desconocidos, ha creado la necesidad de ubicarse y mapear simultáneamente las posiciones y recorridos. De aquí el nombre del algoritmo SLAM (Simultaneous Localization and Mapping), estas investigaciones ya tienen un volumen considerable en

el estado del arte, iniciando en 1988 con estudios que buscaban la aplicación de este algoritmo para minimizar el ruido de las marcas de carretera en el mapa y la posición de los vehículos. Por consiguiente, los algoritmos SLAM direccionaron las investigaciones y el desarrollo de sensores visuales, hasta llegar a los más conocidos en la actualidad como Kinect y Xtion Pro Live, los cuales son utilizados en una variedad de aplicaciones de mapeo y reconstrucción de entornos en 3D [21] y derivando en algoritmos, tales como el algoritmo lineal de 8 puntos o el algoritmo de 5 puntos [22].

Este algoritmo ha permitido el desarrollo de aplicaciones médicas de reconstrucción 3D como la de [23], donde los autores utilizan un endoscopio monocular que, a través del desplazamiento y la captura de cuadros en diferentes posiciones de la cámara a lo largo del intestino, permite extraer información de profundidad para la reconstrucción de la superficie en 3D a partir de la geometría que escena. Cabe resaltar que, en este trabajo de investigación, realizan la implementación de la reconstrucción 3D en Ubuntu 14.04 utilizando programación C/C++. Todos los experimentos se realizan en una estación de trabajo equipada con CPU Intel Xeon (R) de cuatro núcleos a 2,8 GHz, memoria de 32 GB y una tarjeta gráfica NVIDIA GeForce GTX 970. El tamaño de las secuencias de imágenes de simulación es de 1024x768 píxeles y el tamaño del vídeo del endoscopio en vivo es de 840×640 píxeles, de lo cual es posible realizar una estimación de la potencia de cómputo requerida en la implementación de este tipo de aplicaciones, tomando en cuenta que, para este caso en particular, estas características están condicionadas por el requerimiento de reconstrucción 3D en tiempo real.

Finalmente, como expone Liu en [1], es importante tener en cuenta que el algoritmo SLAM es más adecuado para aplicaciones que impliquen objetos con características abundantes de textura geométrica, ya que es fácil perder fotogramas en la operación de rotación y, por otro lado, la nube de puntos en el mapa obtenido es escasa.

2. **Structure from Motion SfM**

   Este algoritmo se basa en los mismos principios de la fotogrametría estereoscópica, razón por la cual, la reconstrucción en 3D se obtiene a partir de la superposición de imágenes obtenidas desde diferentes ángulos de captura. Esta, también denominada técnica, permite obtener un conjunto de coordenadas para conformar la nube de puntos de la estructura en 3D junto con información adicional de canales RGB, entre otros espacios de colores. Por lo tanto, de esta forma es posible obtener características tanto ópticas como geométricas para la extracción y procesamiento de parámetros relevantes en aplicaciones médicas para reconstrucción de heridas en 3D. Este tipo de algoritmo, al ser estereoscópico, no requiere información interna de las cámaras ni datos de orientación o ubicación de las mismas para lograr la reconstrucción de la escena en 3D [24].

   La obtención de un conjunto de características ópticas ha permitido el desarrollo de aplicaciones y herramientas de medición de heridas basadas en

el procesamiento de imágenes. Debido a que la detección de la herida, como primera etapa del sistema, es una función que determina el desempeño total de la aplicación, los algoritmos SfM permiten realizar estos cálculos en conjuntos de bloques de histogramas de color obtenidos a través de espacios de color como el HSV y el enfoque del 'vecino más cercano', como fue desarrollado en [25]. En la técnica implementada para la investigación recién mencionada, posteriormente, utilizan también un algoritmo ICP (Iterative Closest Point) para determinar la transformación del cuerpo rígido, TSDF (Truncated Signed Distance Function) para mejorar el color en la fusión de la escena y el algoritmo de marcha de cubos para la creación de la malla como superficie del objeto reconstruido. Al igual que un número considerable de proyectos enfocados en la reconstrucción 3D de heridas, este fue desarrollado en C++. Sus pruebas fueron realizadas en una computadora portátil con Intel Core i7 47 10HQ CPU, tarjeta gráfica Nvidia 860M GPU y, una de las características más relevantes del sistema, un sensor Kinect v2.

Es importante agregar que, para aislar solo la herida como un modelo 3D separado de toda la escena 3D generada, se utiliza en la anterior investigación el Visualization Toolkit (VTK), ya que el filtro de selección de la biblioteca VTK utiliza el límite interpolado de spline para generar datos escalares.

3. **Time of Fligth**

   Es un método empleado para capturar información de una superficie u objeto 3D, muy usado por la robótica médica. Es un método que consiste en medir el tiempo de vuelo de un haz de luz infrarroja, de ahí su nombre literal en inglés [26].

   Los cámaras ToF (time of fligth), tiene un emisor de la luz infrarroja y un receptor, por lo que, para la adquisición de datos se tiene en cuenta el desfase de la frecuencia del rebote de la luz, de lo que se puede concluir que se tienen datos innecesarios y se hace obligatorio hacer filtros para poder tener una reconstrucción más precisa [27], por otro lado, los datos tienen un margen de error "pequeño" (depende del uso) del orden de los milímetros.

   En la figura 2.4 se puede observar el principio de funcionamiento de las cámaras ToF, el cual se puede describir en un modelo matemático simple a través de una correlación entre las dos señales, la de salida del emisor y la de llegada al receptor.

   Se define la función de la señal de salida del emisor como:

$$g(t) = \cos wt \tag{2.10}$$

   y para la señal de regreso se produce un desfase, se tiene:

$$S(t) = b + a * \cos(wt + \Phi) \tag{2.11}$$

   Al hacer una convolución entre las dos funciones 2.10 y 2.11 se obtiene:

$$c(t) = S \otimes g = \int_{-\infty}^{\infty} S(t) * g(t + \tau) \cdot dt \tag{2.12}$$

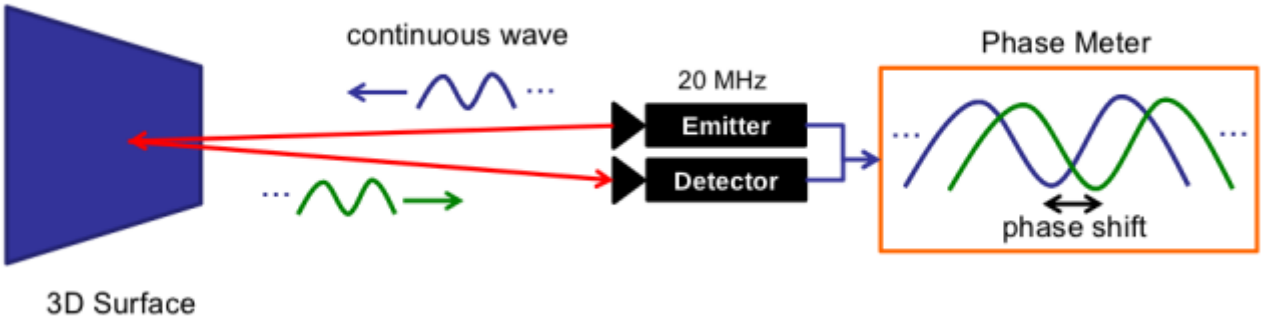


Figura 2.4: Principio de funcionamiento del ToF

La ecuación 2.12 se puede simplificar de la siguiente manera:

$$c(t) = b + \frac{a}{2} * \cos(wt + \Phi) \tag{2.13}$$

Sí se tienen 4 muestreos de A_0 hasta A_3, las fórmulas para amplitud, desfase y distancia, quedan resumidas en las ecuaciones 2.14, 2.15 y 2.16 respectivamente.

$$a = \frac{1}{2} * \sqrt{(A_3 - A_1)^2 + (A_0 - A_2)^2} \tag{2.14}$$

$$d = \frac{C}{4\Pi\Omega} \tag{2.15}$$

$$\Phi = \frac{(A_3 - A_1)}{(A_2 - A_0)}; \;\; A_i = \left(C, \frac{\Pi}{2}\right) \tag{2.16}$$

Las ecuaciones 2.14, 2.15 y 2.16 sugieren que deben realizarse varias tomas desde puntos específicos para poder obtener una reconstrucción precisa y puede llegar a ser un trabajo tedioso y poco eficiente en distintos usos de la robótica médica sino se cuenta con un robot asistencial para el desplazamiento calculado y una orientación especificada para poder asociar la disparidad de los píxeles [27]

En los últimos años se viene trabajando con cámaras Kinect en vez de ToF por su menor coste en el mercado.

2.3 Sistemas de reconstrucción 3D

Debido al alto costo de los sistemas comerciales integrados para reconstrucción de objetos en 3 dimensiones, es menester optar por la implementación de un sistema de adquisición de bajo o mediano costo que se encuentre sustentado en investigaciones anteriores. Por ejemplo, en [28] el autor realiza una recopilación detallada de aplicaciones para reconstrucción 3D con puntos de referencia que fueron realizadas solamente con tarjetas Kinect $v1$ y $v2$ y arreglos de estas, resaltando que estos dispositivos proveen buenas características de costo, calidad

de imagen, facilidad de uso y calibración, al igual que la cámara de profundidad Asus Xtion [20].

Sin embargo, la inclusión del cálculo de profundidad acelerado por hardware a través de conexión USB en los sensores RGB portátiles, como en la mencionada Kinect de Microsoft, generó gran popularidad en el uso de estos sensores en la visión por computadora y la robótica [29], debido a esto en 2015 Intel anunció una familia de cámaras con sensor de profundidad RGBD denominada Intel RealSense, las cuales son altamente portátiles, estereoscópicas, con características como precisión de disparidad para subpíxeles, iluminación asistida y robustez para su uso en ambientes no controlados [29]. En este punto, es importante tener en cuenta investigaciones como la de Zoul et al. en [30], la cuales nos permite conocer resultados comparativos de las características y el desempeño de los dispositivos RGBD comunes mencionados, a saber: Microsoft Kinect, Intel RealSense y Asus Xtion PRO Live en relación al desempeño y los resultados para escaneo y reconstrucción 3D.

En la literatura actual existe un número significativo de investigaciones en diversos campos en los que se utiliza, en mayor proporción, la cámara Intel RealSense debido a que provee un método factible, eficiente, rápido y de bajo costo para la reconstrucción 3D, como por ejemplo, en la reconstrucción tridimensional del pie para diagnóstico de deformidades en [31] o para la obtención de modelos 3D de la planta y maquinaria de fábricas reales en el contexto de la industria 4.0 como en [20].

Por otro lado, existen metodologías de reconstrucción por medio de visión de máquina, como la que presenta Camacho en [3], en la cual plantea un sistema de adquisición apoyado en técnicas de metrología óptica, específicamente con el uso de una fuente de luz estructurada y coherente como lo es la luz láser para formar un mapa de franjas o puntos que contenga la información necesaria para extraer las características tridimensionales del objeto.

También cabe añadir que es alentador encontrar que, debido al constante desarrollo de cámaras digitales de alta calidad a bajo costo, ha surgido una comunidad muy activa de investigación para el área de visión por computadora interesada en acercarse también a la reconstrucción tridimensional de objetos a partir de fotografías, resultado de esta tendencia, entidades como la agencia cartográfica francesa IGN (Institut Géographique National) han desarrollado herramientas de código abierto y gratuito para la tarea de calibración fotogramétrica de conjuntos de imágenes como APERO y MICMAC, un programa para calcular mapas de profundidad a partir de imágenes orientadas [32].

También se cuenta con muchas técnicas de reconstrucción de un objeto en 3D, como lo son: el uso de una cámara simple para la captura de la imagen y para la profundidad se emplea ya sea un láser o una misma captura de imagen pero con RGB para detallar la profundidad de los píxeles por color como lo hacen [33], pero estas técnicas son empleadas para realizar reconstrucciones 3D de mapas por su alta precisión en el cálculo de la profundidad de los píxeles aunque es algo costoso su implementación por los equipos que se usan, en la robótica asistida por computador, aun no se han probado, ya que hacer el escaneo con

láser puede tomar un buen tiempo y no es beneficioso para una cirugía demorarse en obtener los datos, el RGB podría llegar a presentar problemas con la coloración y densidad de los fluidos corporales (sangre) a la hora de asignar los colores [34].

Para el tratamiento y procesamiento de imágenes se tienen muchos software, pero hay uno en particular que permite hacer la captura y procesamiento casi que de inmediato (depende del equipo pc en el que se esté trabajando), y es Matlab, con las últimas actualizaciones que ha tenido el software, hoy se encuentra dotado de comandos para hacer lectura de una cámara comunicada al pc y hacer capturas, y gracias a la librería LIDAR 3D se puede realizar el tratamiento de la captura para llevar a cabo una reconstrucción 3D.

Las ventajas de usar Matlab es que se cuenta con licencia y soporte por parte del equipo de MathWorks y en su página oficial hay tutoriales con muy buena explicación para comprender y entender el funcionamiento de cada uno de los comandos.

Al hablar de visión asistida por computadora se debe hacer un acercamiento a las señales de profundidad que logramos captar los humanos y llevarlas a la máquina, en [1] hace el análisis de señales de profundidad captadas desde una imagen monoscópica y una estereoscópica, al hacer el estudio se muestra que de una visión monoscópica se pueden extraer o deben estudiarse 9 señales de profundidad, las cuales son:

- Tamaño de la imagen referente a la retina: lo que nos indica a los humanos que tenemos un objeto cerca o lejos.
- Perspectiva de los objetos con respecto a un observador, muestra la interacción de los objetos desde el punto de vista desde un punto fijado en el entorno (observador).
- La interposición o sobre posicionamiento de los objetos, si hay 2 o más objetos entre el observador y el punto hacia donde está mirando el observador, debe notarse la interrupción de los objetos y saber cuál está más cerca al sujeto.
- La perspectiva aérea, lo que nos hace notar que estamos en un punto y podemos tener distintas percepciones del entorno, podemos notar objetos más lejanos o cercanos, más claros o con una mayor dificultad de visibilidad.
- Luz y sombra, es una de las señales más importantes en la visón monoscópica, ya que orienta hacia donde hay fuentes de luces y hace que se identifique una profundidad notoria en un objeto.
- El gradiente de la textura, podemos tener varios objetos de la misma forma y al alejarnos podemos percibir que son iguales, pero al acercarse se puede notar una diferencia en sus cuerpos o en su superficie.
- Movimiento en paralelo de los objetos, fijarlos a un punto y ver que conservan su forma, pero al entrar en movimiento se pueden distorsionar para el observador.

- Y por último la visión de campo, que se refiere a la distancia que debe estar un objeto para que se desenfoque, esta varía dependiendo del tamaño retinal del objeto.

Mientras que, en la visión estereoscópica, hay 3 señales de profundidad a estudiar y tener en cuenta, una es la convergencia, donde cada ojo genera una imagen y para que haya una convergencia los 2 ojos deben fijar un punto común y el cerebro forma una única imagen a partir de las 2 generadas por ambos ojos (es necesario puntos en común).

Por otra parte, aparece la disparidad, la cual se da cuando los objetos cercanos a un punto de convergencia se muestran, esto es a causa de las 2 imágenes captadas por los ojos y el procesamiento del cerebro para acomodarla en una sola.

También se plantea una zona de confort visual, lo que se puede relacionar con la cámara para se pueda enfocar bien la imagen y obtener la mayor cantidad de detalle de la escena. A consecuencia de esto se muestra un modelo matemático para poder obtener información y un modelo matemático del sistema estereoscópico, que se muestra a continuación:

$$K_l = \begin{bmatrix} f & 0 & pl \\ 0 & f & q \\ 0 & 0 & 1 \end{bmatrix} ; \; R_l = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} ; \; t_l = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.17}$$

y la matriz de la cámara izquierda queda: $P_l = K_l(R_l \backslash t_l)$

$$P_l = \begin{bmatrix} f & 0 & pl & 0 \\ 0 & f & q & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.18}$$

De manera análoga se obtiene la matriz P_r de la cámara derecha.

Teniendo en cuenta que este modelo planteado es para dos cámaras que capturaran vídeo, de manera similar podría funcionar para una cámara estereoscópica de imágenes, teniendo en cuenta que el nombre de la técnica es comúnmente conocido como triangularización, y es una de las más conocidas en el ámbito de grabación de entornos.

Para hacer la reconstrucción de la escena en 3D usan las matrices inversas del sistema y se obtiene un modelo matemático como sigue:

$$K^{-1} = \begin{bmatrix} K & 0 \\ 0^T & 1 \end{bmatrix} ; \; E = \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} ; \tag{2.19}$$

Donde se puede hacer una reconstrucción precisa del entorno teniendo en cuenta la disparidad de los píxeles y las señales de profundidad en conflicto.

2.4 Sistema de virtualización

La importancia de obtener una reconstrucción tridimensional de las heridas y laceraciones obtenidas a partir de imágenes transversales en 2D, radica en proveer al personal médico, o quirúrgico, de una herramienta que les permite estudiar de una mejor forma la composición anatómica de cada caso específico en un ambiente virtualizado, conservando las proporciones y la información relevante para el tratamiento y el diagnóstico. Por ejemplo, y específicamente en el campo médico, Wake et al. [33] mostraron cómo el abordaje quirúrgico cambió de un 30 al 50 % por parte de los médicos después de una visualización de los modelos en 3D de los riñones.

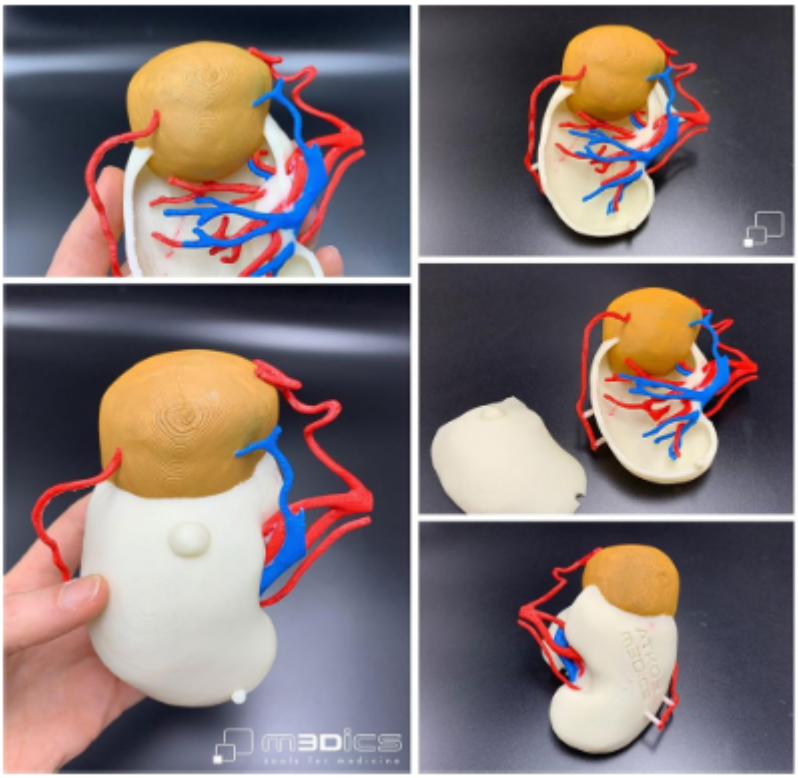


Figura 2.5: Modelo 3D impreso a partir de reconstrucción en Unity

Tomado de [15]

Por lo tanto, al demostrar la importancia y, a su vez, el extenso campo de aplicación de la virtualización 3D como herramienta de apoyo de ejecución y planeación quirúrgica, se han desarrollado investigaciones como la de Wake en [34], para la reproducción de estos modelos en ambientes software como Unity 3D e implementados de forma secundaria en el sistema Microsoft HoloLens, obteniendo modelos de riñón en realidad aumentada.

En este sentido, otras investigaciones como la de Porpiglia en [35], han conseguido desarrollar y obtener un modelo virtual en 3D de la próstata, con la finalidad de examinar el órgano del paciente, controlar sus valores de traslación, rotación y transformación de escala. Este modelo virtual fue desarrollado de igual forma en la plataforma Unity y C#, cargado a una consola quirúrgica remota Da Vinci por medio de la aplicación pViewer, como se muestra en la figura 2.6.

Estas y otras investigaciones nos permiten concluir, parcialmente, la versatilidad y eficacia que desempeña un software libre como Unity 3D como base para la reconstrucción de los modelos 3D a obtener en el marco de la actual investigación. Razón por la cual se plantea el desarrollo de las pruebas y simulaciones de las heridas reconstruidas en esta herramienta software. Además, una vez obtenidos los modelos virtuales 3D, se pueden aplicar en entornos diferentes, tales como: procedimientos cognitivos y procedimientos de realidad aumentada. Este último, es un campo de aplicación en el cual es menester sentar las bases para investigaciones futuras, debido a que las tendencias en la ingeniería apuntan a las intervenciones quirúrgicas asistidas por robots.

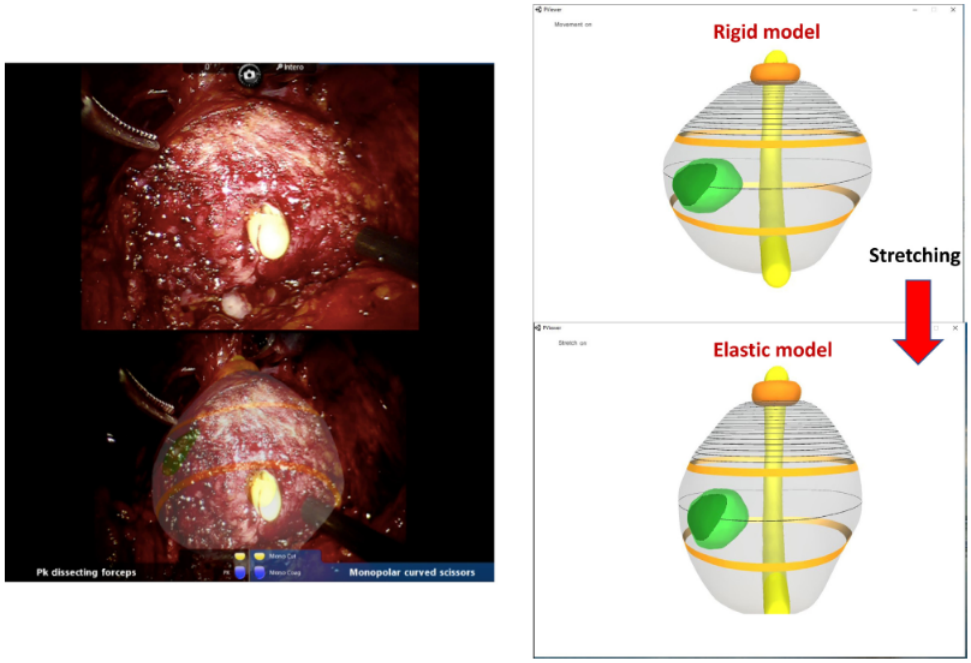


Figura 2.6: Reconstrucción y superposición del modelo 3D con características de rigidez y elasticidad de la próstata.

Tomado de [15]

3 Metodología

Continuando con el proyecto se dispuso una cámara de profundidad, la cual facilita la reconstrucción que se desea para la herida, esta cámara es del fabricante Intel, proveniente de la linea de RealSense y la referencia es D435i. A continuación, se mostrará un breve resumen de la misma:

- **Intel Realsense Depth Camera D435i:**
  La cámara 3D de Intel utiliza visión estéreo activa para calcular la profundidad. Cuenta con un proyector de infrarrojos, módulo de visión estereoscópico, procesador de visión y sensor RGB. El cálculo de la profundidad se realiza con el procesador de visión D4 que se encuentra dentro de la cámara.

  El control de las características de adquisición de la cámara, modos de operación y datos extrínsecos e intrínsecos asociados a esta pueden ser realizados por medio del API desarrollado por Intel y se puede integrar con diferentes plataformas. Las principales características en la cámara se muestran en la tabla:

Características Intel Realsense Depth Camera D435i	
Cámara RGB	1920 x 1080
Cámara de profundidad	1280 x 720
Distancia máxima	10 m (puede ser más dependiendo de la iluminación)
Distancia mínima	0.2 cm

Tabla 3.1: Características de la cámara

La alimentación del sensor es vía USB y sus dimensiones son: $90mmx25mmx25mm$ (largo x ancho x alto). La figura 3.1 muestra la cámara de profundidad D435i.

Figura 3.1: Cámara Intel RealSense D435i

Para la implementación del sistema de reconstrucción 3D de la laceración se realizó un montaje experimental de la base, almohadilla e iluminación de la siguiente forma.

- **Estructura de soporte para la cámara:**
  Se utilizó un soporte metálico con un trípode para agregarle balance y evitar la caída, maneja un sistema de empotre para la cámara a medida y además el soporte es ajustable tanto en eje Y (altura) y X (desplazamiento).

Figura 3.2: Estructura de soporte para la cámara y sistema de iluminación

Fuente: Autores

- **Almohadillas de pruebas:**
  Inicialmente se empezó trabajando con almohadillas en Foamy de forma artesanal simulando diferentes incisiones para poner a prueba el sistema de reconstrucción, como se puede observar en las figuras 3.3a, 3.3b y 3.3c.

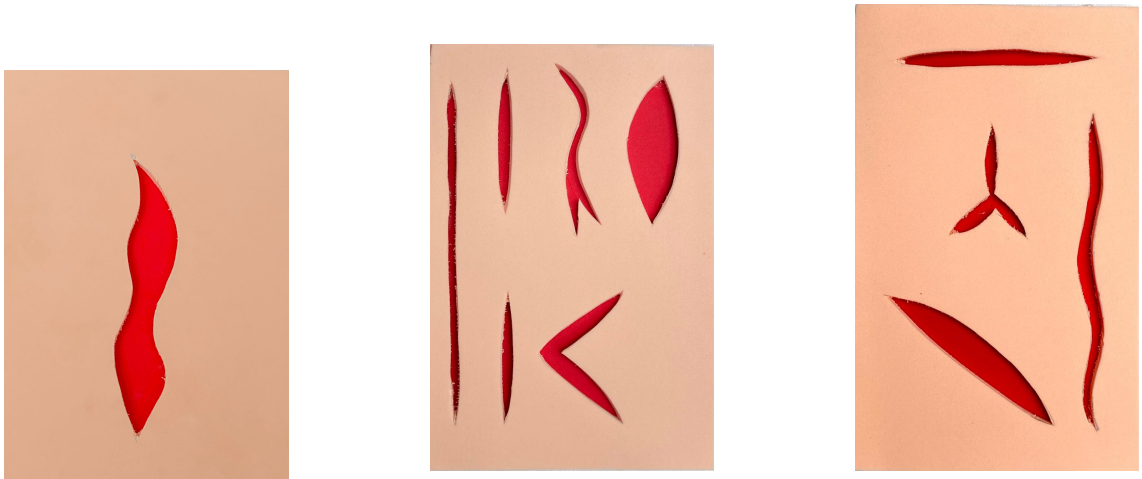
  Cabe resaltar que se realizan distintos tipos de cortes para detectar falencias en el sistema de reconstrucción y evidenciar sus causas y tratar de dar solución sin aumentar el costo del sistema.

- **Sistema de iluminación:**
  Para lograr una mejor vista de la escena se requiere un sistema de iluminación, en este caso se utiliza un aro de luz de 12 cm de radio con una potencia aproximada de 10 watts y entre 2400 y 3000 lummens, se aprecia en la figura 3.4.

  Por otra parte la cámara se empotra en la mitad del aro para optimizar las ganancias de luz.

3.0.1 Configuración del escenario de pruebas

Para obtener un óptimo resultado en las pruebas se llegó a la conclusión de que la cámara debe estar situada a una altura de entre 17cm a 50cm, siendo la que mejor resultados arrojó en cuanto a resolución y niveles de pixelación fue una altura de 24,5cm, la cámara se colocó en el centro del aro de luz donde mejoraba



(a) Incisión grande

(b) incisiones más delgadas y distintas formas

(c) Incisiones con grosor medio y varias formas

Figura 3.3: Almohadillas artesanales de pruebas

Fuente: Autores

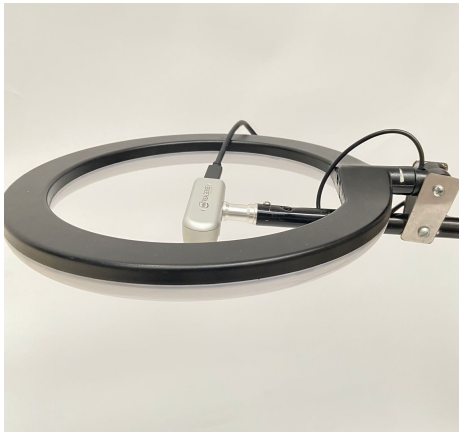


Figura 3.4: Sistema de iluminación

Fuente: Autores

la visualización del entorno por los niveles de luz. Resumiendo esta información en la tabla 3.2 para que pueda ser replicada y el escenario se muestra en la figura 3.5.

3.1 Softwares utilizados en el desarrollo del aplicativo del proyecto

A continuación, se mostrará la integración de los diferentes elementos para el desarrollo del proyecto.

Primero y más importante a utilizar es Unity 3D, ya que en él se realiza la captura de la escena y reconstrucción de la herida a través de la cámara Intel Realsense D435i, en este programa también se utilizan los diferentes filtros para la mejora de la reconstrucción. Se utilizan los filtros que se encuentran a disposición en el programa tales como el Bilinear y el Trilinear, que permiten manipular algunos

Figura 3.5: Escenario de pruebas

Item	**Valor**
Distancia de la cámara a la almohadilla	24,5 cm
Altura de soporte de la cámara	70 cm
Área de procesamiento de imagen	22 cm x 16 cm
Radio de lámpara led	12 cm
Potencia lámpara led	10.2 Watts
Tasa de frames de la cámara	30 FPS
Resolución de la cámara	2 Mp (1920x1080)
Campo de visión	69° x 42°

Tabla 3.2: Tabla de configuración de parámetros de la escena

parámetros de las capas de superficies de la nube de puntos, para evitar específicamente el ruido en los bordes del phanton y mejorar la calidad de reconstrucción del sólido en la escena, aplicándolos a los canales de color y profundidad previamente configurados con formatos válidos y aceptados por la cámara.

Por otra parte, se configuran los canales de profundidad y color con formatos válidos para el programa y soportados por la cámara.

El programa Unity se utiliza para la reconstrucción y para la extracción de características físicas de tamaño de la herida tales como longitud (largo y ancho) y la profundidad, se usa el IDE PyCharm para trabajar directamente con Python en conjunto con la librería de OpenCV.

Dentro de Python para poder realizar la extracción de características se utilizan

las siguientes librerías:

- **Numpy:**
  Es la encargada de manejar operaciones de aritmética y valores numéricos.
- **Pyrealsense2:**
  Encargada de la comunicación y el manejo entre la cámara y el software.
- **CV2:**
  Librería de Open Cv que permite la captura y manipulación de archivos de imagen y vídeo, como también ayuda en la reconstrucción de la herida, adicionalmente extrae las características de la misma.

3.2 Implementación del script de reconstrucción

Se inicia importando las librerías necesarias (en caso de que no se tengan instaladas deben instalarse antes, para evitar errores de compilación), como se muestra a continuación:

```
1  import numpy as np
2  import pyrealsense2 as rs
3  import CV2
```

Código 3.1: Importación de librerías

Por otra parte se hace la configuración de la profundidad y color del canal de la cámara:

```
4  #configuracion de prodfundidad y color
5  pipeline = rs.pipeline()
6  config = rs.config()
```

Código 3.2: Configuración de los canales de profundidad y color

Luego se captura la información de la cámara como el módelo y línea del producto.

```
7  # Obtener el dispositivo y la linea del producto en una resolución
       soportada
8  pipeline_wrapper = rs.pipeline_wrapper(pipeline)
9  pipeline_profile = config.resolve(pipeline_wrapper)
10 device = pipeline_profile.get_device()
11 device_product_line = str(device.get_info(rs.camera_info.
       product_line))
```

Código 3.3: Reconocimiento del dispositivo

En las siguientes lineas de código se habilitan los canales de profundidad y color, previamente configurados y se les da los formatos validos, en este caso se usa un formato de profundidad Z16 para medir la media de texturas del objeto y un formato de color RGB, en específico RGB8, y estableciendo todo a una tasa de 30 FPS, formatos soportados por la cámara. En la linea 29 se ejecuta el inicio de la transmisión de datos al computador y se genera la visualización de la almohadilla para la posterior extracción de sus características.

```
    found_rgb = False
    for s in device.sensors:
        if s.get_info(rs.camera_info.name) == 'RGB Camera':
            found_rgb = True
            break
     if not found_rgb:
        print("The demo requires Depth camera with Color sensor")
    exit(0)

    config.enable_stream(rs.stream.depth, 640, 480, rs.format.z16, 30)

    if device_product_line == 'L500':
        config.enable_stream(rs.stream.color, 960, 540, rs.format.bgr8
            , 30)
    else:
        config.enable_stream(rs.stream.color, 640, 480, rs.format.bgr8
            , 30)

    # Inicio de transmición de la camara al computador
    pipeline.start(config)
```

Código 3.4: Inicio de la transmisión

Posteriormente entra en el bucle while, que espera la información y actualiza los frames (fps) de color y profundidad, al recibir la información en forma de matrices de cada píxel y con una función de la librería recopila toda esa información y se arregla como imagen a través de la librería Cv2. Se aplican las escalas dependiendo el brillo y realiza el mapa de colores y de profundidad, esto se encapsula en un bloque try - finally para evitar fallos inesperados que interrumpan la secuencia del ciclo, terminando con la finalización de la captura de datos.

```
    try:
        while True:

        # Esperan un par de frames: profundidad y color
            frames = pipeline.wait_for_frames()
            depth_frame = frames.get_depth_frame()
            color_frame = frames.get_color_frame()
            if not depth_frame or not color_frame:
                continue

        # convierte imagenes en vectores
            depth_image = np.asanyarray(depth_frame.get_data())
            color_image = np.asanyarray(color_frame.get_data())

        # aplica un mapeo de color en la profundidad de la imagen (la
            imagen tuvo que ser convertida a 8-bit por pixel)
            depth_colormap = cv2.applyColorMap(cv2.convertScaleAbs(
                depth_image, alpha=0.03), cv2.COLORMAP_JET)

            depth_colormap_dim = depth_colormap.shape
            color_colormap_dim = color_image.shape

        # If depth and color resolutions are different, resize color
            image to match depth image for display
```

```
            if depth_colormap_dim != color_colormap_dim:
                resized_color_image = cv2.resize(color_image, dsize=(
                    depth_colormap_dim[1], depth_colormap_dim[0]),
                    interpolation=cv2.INTER_AREA)
                images = np.hstack((resized_color_image,
                    depth_colormap))
            else:
                images = np.hstack((color_image, depth_colormap))


            def get_point(event, x, y, flags, param):
                if event == cv2.EVENT_LBUTTONDBLCLK:
                    distance = depth_image[y, x]
                    print(str(distance) + " " +  str(x) +" "+ str(y))

                # Tell pointcloud object to map to this color frame
                #pc.map_to(color)

                # Generate the pointcloud and texture mappings
                #points = pc.calculate(depth)

                #print("Saving to 1.ply...")
                #points.export_to_ply("1.ply", color)
                # print("Done")

            # Show images
            point = (400, 300)
            cv2.namedWindow('RealSense', cv2.WINDOW_AUTOSIZE)
            cv2.setMouseCallback('RealSense', get_point)
            cv2.imshow('RealSense', images)
            cv2.waitKey(1)

    finally:

        # Stop streaming
        pipeline.stop()
```

Código 3.5: Tratamiento de los datos capturados por la cámara

Se tiene en cuenta las dimensiones de la imagen y el tamaño del arreglo que llega desde la cámara para en caso de no coincidir se le realice un escalado o ajuste dependiendo del canal de color y de profundidad y una vez realizado se define la función “get point” que se encarga de poner un evento que permite medir el click del ratón (doble click), mide la distancia con la función predefinida de la cámara en su librería, con el comando print, se despliega separando primero (distancia en mm, coordenada en x en píxeles, coordenada en y en píxeles), con la función de procesamiento de imagen se crea las ventanas en las que se van a mostrar las imágenes en este caso la de color como de profundidad.

Por último se obtienen las características deseadas como se muestran en la figura 3.6

Por otro lado, se debe extraer un modelo 3D para poder tener una base de trabajo en la aplicación de la realidad aumentada (AR) la cual se trabaja en Vuforia, como se explica más adelante, para poder extraer este modelo se usó el siguiente script

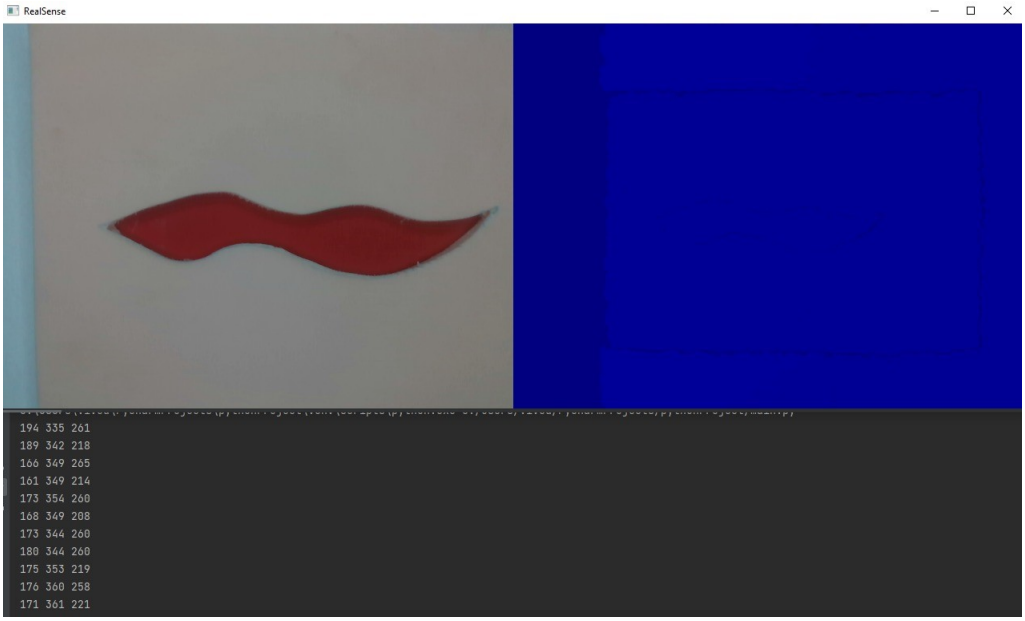


Figura 3.6: Reconstrucción en vivo y extracción de características de la herida

que se explica con mayor detalle:

```
import math
import ctypes
import pyglet
import pyglet.gl as gl
import numpy as np
import pyrealsense2 as rs
```

Código 3.6: Importación de librerías

Lo primero que se hace es importar las librerías, para poder usar los comandos especiales de cada una de ellas, Math permite el uso de funciones entre números reales (no soporta operaciones entre complejos), ctypes permite el uso del kernel y se trabaja en python nativo con compatibilidad de trabajar en C, y hacer uso de los archivos .dll, pyglet y pyglet.gl permiten implementar funciones específicas a teclas de un game path o un teclado normal mientras el script esté en ejecución numpy y la de realsense2 ya se han explicado anteriormente.

```
    axis = np.asarray(axis)
    axis = axis / math.sqrt(np.dot(axis, axis))
    a = math.cos(theta / 2.0)
    b, c, d = -axis * math.sin(theta / 2.0)
    aa, bb, cc, dd = a * a, b * b, c * c, d * d
    bc, ad, ac, ab, bd, cd = b * c, a * d, a * c, a * b, b * d, c * d
    return np.array([[aa + bb - cc - dd, 2 * (bc + ad), 2 * (bd - ac)
        ],
                     [2 * (bc - ad), aa + cc - bb - dd, 2 * (cd + ab)
                         ],
                     [2 * (bd + ac), 2 * (cd - ab), aa + dd - bb - cc
                         ]])
```

Código 3.7: Matrices de rotación

Se crean las matrices de rotación para obtener la posición respecto a los ejes coordenados y posteriormente definir un estado de posesión mediante una clase de python para generar variables de tipo objeto.

```
    class AppState:

        def __init__(self, *args, **kwargs):
            self.pitch, self.yaw = math.radians(-10), math.radians
                (-15)
            self.translation = np.array([0, 0, 1], np.float32)
            self.distance = 2
            self.mouse_btns = [False, False, False]
            self.paused = False
            self.decimate = 0
            self.scale = True
            self.attenuation = False
            self.color = True
            self.lighting = False
            self.postprocessing = False

        def reset(self):
            self.pitch, self.yaw, self.distance = 0, 0, 2
            self.translation[:] = 0, 0, 1

        @property
        def rotation(self):
            Rx = rotation_matrix((1, 0, 0), math.radians(-self.pitch))
            Ry = rotation_matrix((0, 1, 0), math.radians(-self.yaw))
            return np.dot(Ry, Rx).astype(np.float32)

    state = AppState()
```

Código 3.8: Creación de la clase para medir el estado

Se crea una clase en python, se inicializan los atributos de la misma y se definen 2 métodos uno de reset y el de rotación que permite la manipulación de los datos de la rotación en y R_y y la rotación en x R_x, finalizando con la creación de un objeto de esta clase.

```
    pipeline = rs.pipeline()
    config = rs.config()

    pipeline_wrapper = rs.pipeline_wrapper(pipeline)
    pipeline_profile = config.resolve(pipeline_wrapper)
    device = pipeline_profile.get_device()

    found_rgb = False
    for s in device.sensors:
        if s.get_info(rs.camera_info.name) == 'RGB Camera':
            found_rgb = True
            break
    if not found_rgb:
        print("The demo requires Depth camera with Color sensor")
        exit(0)

    config.enable_stream(rs.stream.depth, rs.format.z16, 30)
    other_stream, other_format = rs.stream.color, rs.format.rgb8
```

```
19      config.enable_stream(other_stream, other_format, 30)
```

En esta parte se configura la transmisión, configurando los canales de color, profundidad y los FPS, un setup idéntico al anterior script, formato de profundidad Z16, formato de color RGB8, y 30 FPS, posteriormente si inicia la transmisión como se muestra a continuación.

```
1       pipeline.start(config)
2       profile = pipeline.get_active_profile()
3
4       depth_profile = rs.video_stream_profile(profile.get_stream(rs.
            stream.depth))
5       depth_intrinsics = depth_profile.get_intrinsics()
6       w, h = depth_intrinsics.width, depth_intrinsics.height
```

Código 3.9: Inicio de la transmisión

Seguido se hace el procesamiento de la captura de la transmisión y se definen las teclas de control durante la ejecución del código, en este caso, se usa un teclado de computador.

Como resultado este script entrega un modelo 3D en escala de grises, en un formato .PLY para su posterior tratamiento en Blender, para poder darle textura y color y hacerle un filtrado del ruido.

3.2.1 Configuración de Unity 3D

Lo primero que se hace es crear una nueva escena, se conecta la cámara a un puerto com, se espera a que la cámara sea leída y se configura como cámara principal, una vez se realiza esta parte se procede a configurar los canales de color y profundidad con los parámetros y formatos que se mostraron en las figuras 3.7, 3.8 , luego se le agregan los filtros correspondientes para mejorar la visualización del phanton como se visualiza en las figuras 3.9a y 3.9b donde se aprecia que se agregó el filtro bilinear.

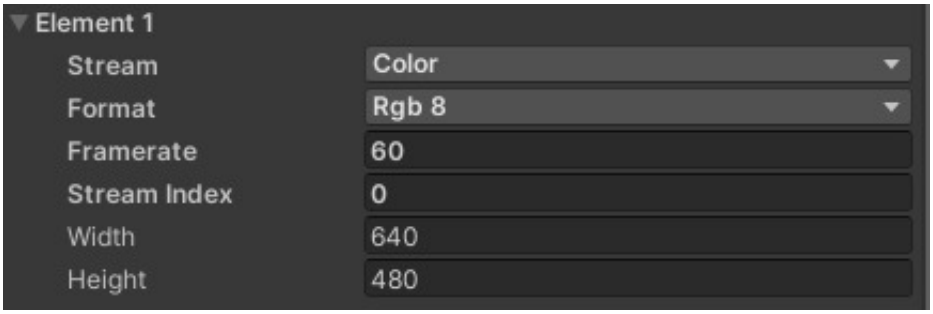


Figura 3.7: Configuración del canal de color en Unity

El algoritmo de reconstrucción utilizado por la cámara es uno mixto que involucra el de time of fligth y el slam, ya que ella envía una señal infrarroja y espera a que rebote para volver a leerla y hace reconstrucción capa a capa texturizando.

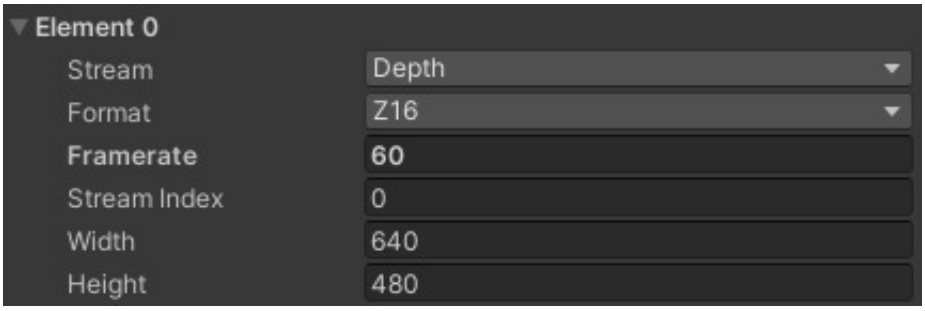
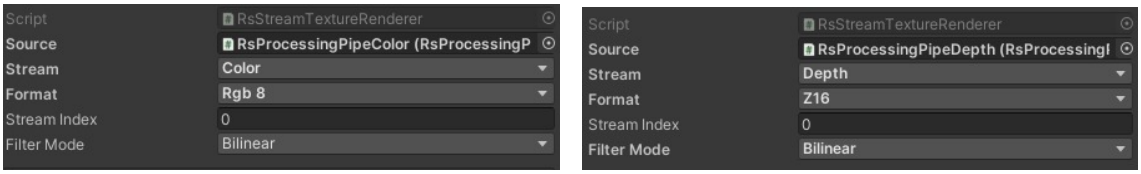


Figura 3.8: Configuración del canal de profundidad en Unity



(a) Filtro para el canal de color

(b) Filtro Bilinear para el canal de profundidad

Figura 3.9: Filtros

Lo anterior se utilizó para la reconstrucción de la herida, para poder tener una base del archivo en 3D y poder utilizarlo en el aplicativo en AR, el aplicativo de AR fue desarrollado en el software Vuforia, con una integración en Unity.

3.2.2 Blender

En este software se hace un tratamiento de imagen, para poder texturizar y darle color al modelo capturado desde la cámara Realsense D435i en 3D, puesto que este modelo sale en escala de grises en el formato (.PLY) y arrastra consigo una cantidad de ruido que dificulta la reconstrucción del phantom en el aplicativo de AR como se puede apreciar en la figura 4.5.

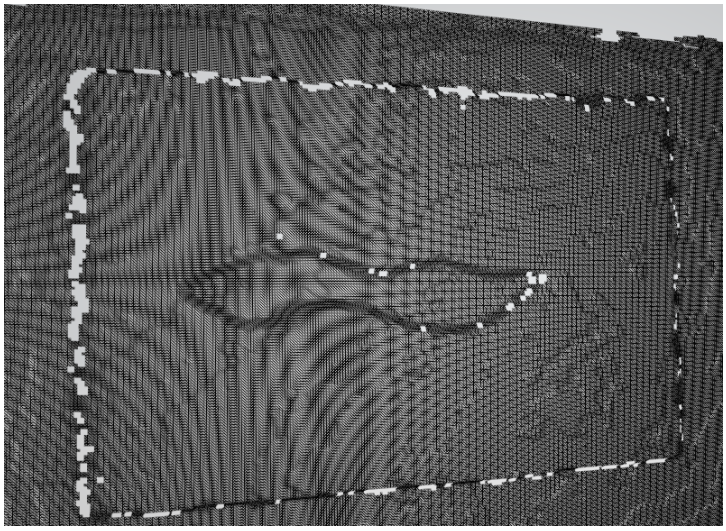


Figura 3.10: Imagen del phantom en formato .PLY

Posterior a este procedimiento, se exporta el nuevo modelo en 3D para poder ser

visualizado en la aplicación de AR ver figura 3.11.

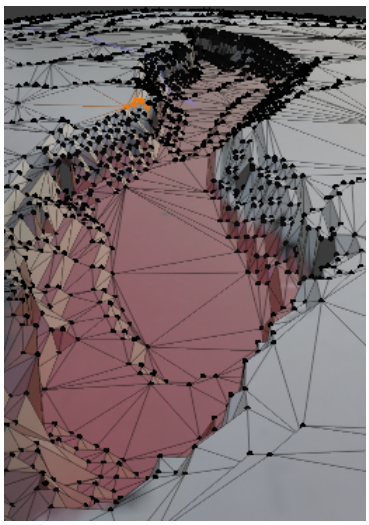


Figura 3.11: Phantom texturizado y a color después del tratamiento de imagen en Blender

3.2.3 Vuforia

Es un software para el desarrollo multiplataforma de aplicativos de VR y AR, donde se trabaja la aplicación de AR de reconstrucción de la herida para poder ser manipulada desde un dispositivo con la capacidad de reproducir este tipo de tecnología.

Como en la mayoría de programas destinados al diseño y tratamiento de imagen, es necesario crear un fichero o biblioteca con los elementos necesarios para poder llevar a cabo la creación de un nuevo material y un nuevo objeto 3D, la cual se encuentra en linea en una base de datos, se define un marcador para la reproducción del objeto reconstruido en el dispositivo Android, el marcador puede ser cualquier imagen o grafo, en este caso usamos el que se observa en la figura 3.12.

Figura 3.12: Marcador para reproducir la herida en AR

4 Resultados:

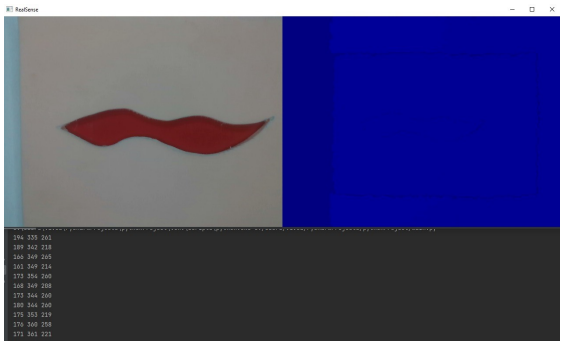
Para el cumplimiento de los objetivos se realizó un montaje simulando el entorno de un quirófano en donde se usaron los elementos mencionados en el capítulo anterior tales como:

- 3 almohadillas artesanales que simulan las laceraciones epidérmicas.
- 1 sistema de iluminación con soporte para la cámara IntelRealsense D435i.
- soporte para las almohadillas simulando una mesa de quirófano.
- 1 computador que cuenta con los softwares necesarios para el uso de la cámara (Python, Unity, Blender).

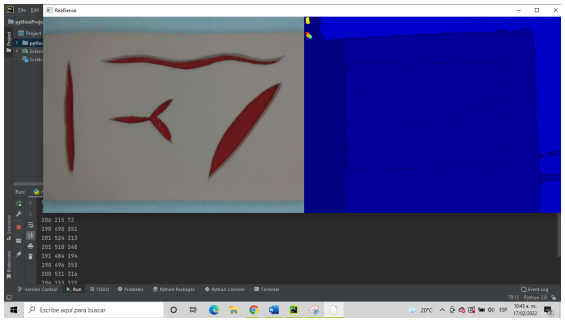
Con la integración de estos elementos se realiza el montaje con parámetros definidos para lograr un correcto uso, funcionamiento y tener una escena apropiada para la reconstrucción.

4.1 Extracción de características de la laceración

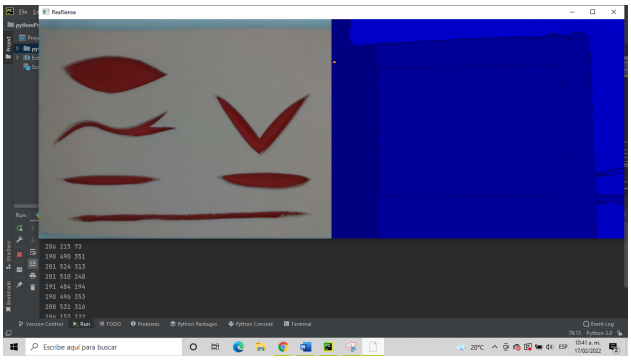
Las imágenes mostradas a continuación (4.1a, 4.1b y 4.1c) surgen al ejecutar el script de extracción de características en Python, dando como resultado una ventana en la que podemos apreciar la herida en formatos RGB, específicamente RGB8 y un formato de profundidad Z16, ayudándose de la librería de OpenCv.



(a) Extracción de características de la almohadilla 1



(b) Extracción de características de la almohadilla 2



(c) Extracción de características de la almohadilla 3

Figura 4.1: Extracción de características físicas de la herida

Tomando como referencia la almohadilla 1 y la parte baja de la figura 4.1a, la cual se muestra de manera más amplia en la figura 4.2, arroja en consola los siguientes parámetros para ser interpretados:

```
199 273 225
193 257 136
193 268 170
192 265 283
189 113 254
201 584 246
```

Figura 4.2: Coordenadas de los píxeles (Z, X, Y)

Se maneja de forma horizontal, primero columna Z, segunda columna X, tercera columna Y. Con estos valores se procede a calcular las características físicas como: la profundidad, largo y ancho de la herida de la siguiente forma:

- Los datos de las filas 1 y 2 corresponde a la profundidad (coordenadas del

eje Z) de la herida y se calcula con la siguiente fórmula:

$$\begin{aligned} Profundidad &= Z_1 - Z_2 \\ Profundidad &= 199 - 193 \\ Profundidad &= 6mm \end{aligned} \tag{4.1}$$

El resultado de la fórmula da los valores equivalentes en milímetros.

- Los datos de las filas 3 y 4 corresponde al ancho de la herida y se calcula con la siguiente fórmula:

$$\begin{aligned} Ancho &= \frac{Y_1 - Y_2}{FE} \\ Ancho &= \frac{283 - 170}{25} \\ Ancho &= 4{,}5cm \end{aligned} \tag{4.2}$$

Tanto para el ancho y el largo de la herida los valores resultantes de las fórmulas son dados en cm.

- Los datos de las filas 5 y 6 corresponde al largo de la herida y ese calcula con la siguiente formula:

$$\begin{aligned} Largo &= \frac{X_1 - X_2}{FE} \\ Largo &= \frac{584 - 113}{25} \\ Largo &= 18{,}84cm \end{aligned} \tag{4.3}$$

Se debe tener como consideración que tanto para largo y ancho el Factor de Escala (FE) para pasar de píxeles a centímetros es aproximadamente 25, también se deben ajustar las valores para tener un valor en positivo dependiendo de donde se quiera realizar la medida.

Para los resultados se halló un error de $+/-2m.m.$ Con esto se puede concluir el primer objetivo específico de captura y extracción de características de la herida.

4.2 Selección de algoritmo para la reconstrucción de la laceración en 3D

Debido a que la cámara tiene un sistema cerrado, y los fabricantes no compartieron un algoritmo en específico con nosotros, hicimos uno a partir del proceso realizado y la información recolectada en el marco teórico, el algoritmo en mención se puede apreciar en la figura 4.3

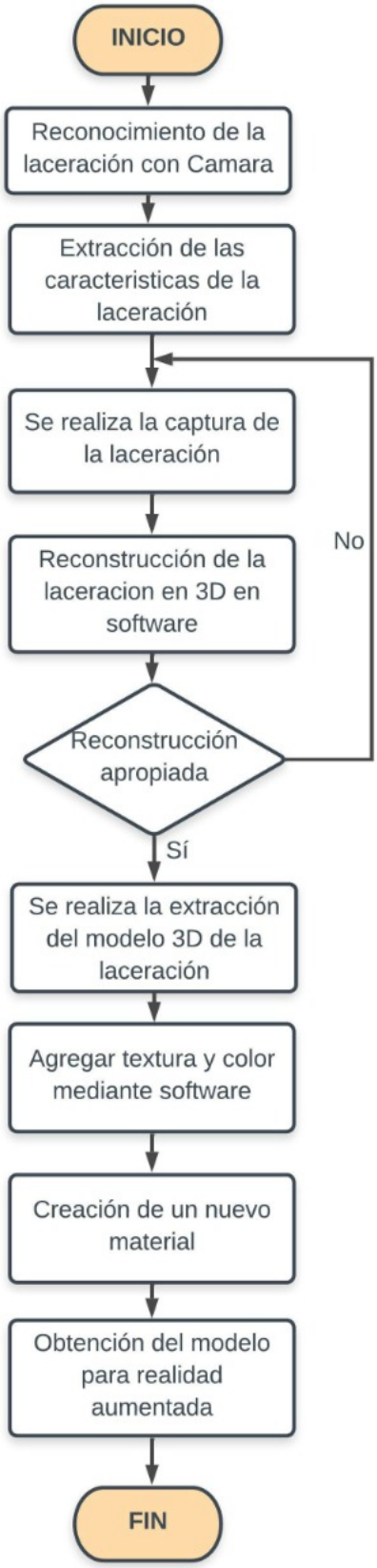
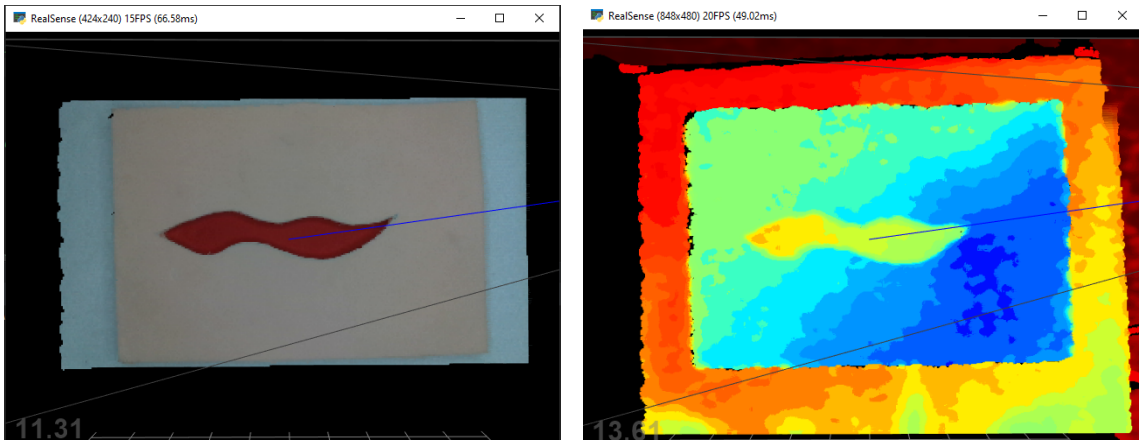


Figura 4.3: Algoritmo del proceso de extracción de características y reconstrucción 3D de la herida

4.3 Reconstrucción 3D de la laceración

Continuando con los resultados obtenidos en el proyecto, se dispone a realizar la reconstrucción en 3D de la herida, con el script correspondiente arrojando las siguientes imágenes (4.4a, 4.4b) que pueden ser vistas tanto en Python como en Unity.



(a) Reconstrucción realizada con python con canales de profundidad y color Z16 y RGB8 respectivamente

(b) Reconstrucción realizada con python con canales de profundidad y color Z16 y RGB respectivamente

Figura 4.4: Diferencia entre configuración de canales

Como se puede apreciar en la figura 4.4a, se obtiene una reconstrucción de la almohadilla muy similar a la real, mientras que en la figura 4.4b, se tiene una capa en nube de puntos con silueta en escala RGB.

Por lo tanto, se trabaja con el phantom creado a partir de la reconstrucción lograda en la figura 4.4a.

4.4 Agregando textura y color al sólido 3D:

Con este modelo se procede a extraer el volumen en 3D, generando un archivo en formato .PLY, el cual se entrega en escala de grises, esto se puede apreciar en la figura 4.5, donde se puede visualizar que las únicas características físicas de la herida que se conservan son las de morfología (forma de la herida) y tamaño (largo, ancho y profundidad), dejando de lado, la textura y el color correspondiente a cada píxel de la reconstrucción.

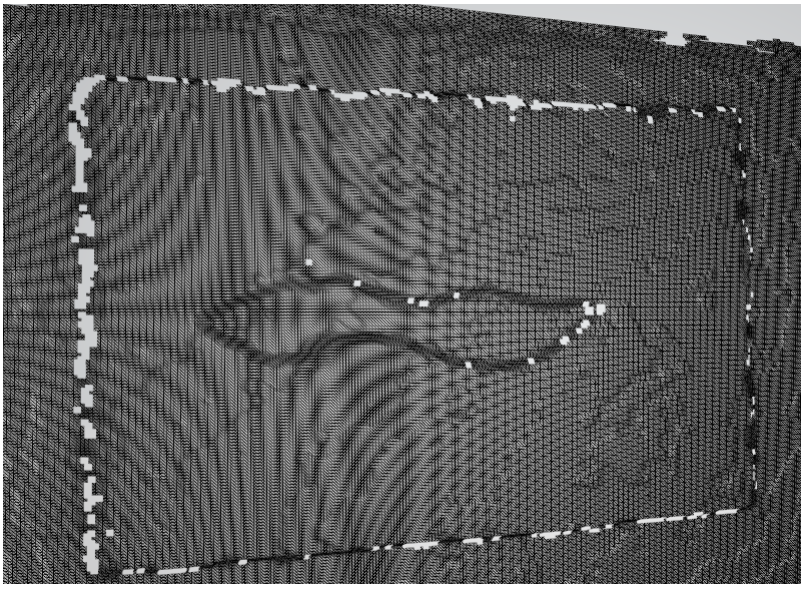
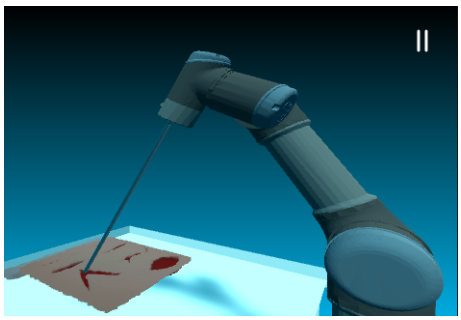


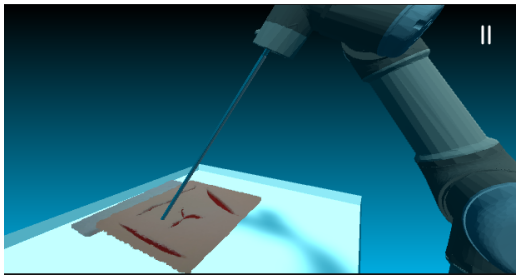
Figura 4.5: Solido 3D generado en Python para tratamiento en Blender

El modelo generado, alcanza a captar ruido, por el amplio rango de visión de la cámara, por lo que es necesario hacer el filtrado y limpieza al sólido, este proceso se realiza en el software libre Blender, también se realiza una reducción a la cantidad de puntos que están inmersos en la nube de puntos con el objetivo de hacer un archivo mas liviano y poder mantener información de vértices, aristas y mapeados geométricos de la estructura del sólido para posteriores trabajos que lo requieran.

Por otra parte en Unity 3D se puede obtener una reconstrucción en tiempo real sobre una escena prediseñada (donde se muestra un escenario reconstruido de un quirófano y un robot a escala real) como se ve en las siguientes imágenes (4.6a y 4.6b), debido a que tiene una conexión directa (cableada) con la fuente de captura principal, facilitando la transmisión de la información sin perdidas de detalle, donde las deformaciones (mínimas, en los bordes del phantom) son causadas por la resolución de la cámara.



(a) Almohadilla 2 reconstruida en Unity 3D

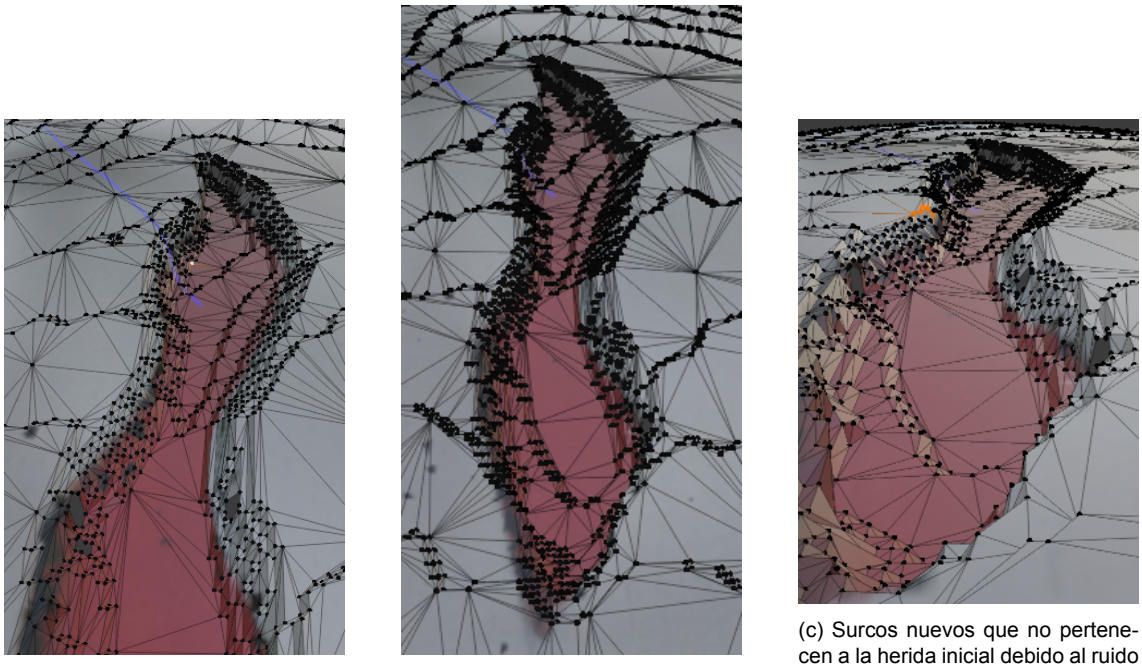


(b) Alohadilla 3 reconstruida en Unity 3D

Figura 4.6: Reconstrucción en tiempo real en Unity3D

Como resultado del procesamiento y tratamiento del modelo en 3D dado en Blen-

der, arroja un modelo adecuado de la herida reconstruida, mostrando un contraste con la figura de la almohadilla real que se ve en la figura 4.8, en lo que se logra apreciar una generación de relieves en la parte profunda de la herida, después del tratamiento en Blender, esto se ocasiona debido al ruido del modelo 3D usado en escala de grises, a la hora de texturizar y agregar color esas imperfecciones son tratadas por lo tanto se generan estas nuevas capas, pero generan una aproximación muy cercana a la morfología de la herida real y pueden llegar a ser despreciadas ya que no interfieren en el proceso quirúrgico y son de tamaño mínimo.



(a) Reconstrucción en Blender

(b) Se crean relieves despreciables

(c) Surcos nuevos que no pertenecen a la herida inicial debido al ruido captado por la cámara

Figura 4.7: Herida después del tratamiento en Blender

Figura 4.8: Almohadilla real de prueba

Con esto se procede a la creación del aplicativo AR en Vuforia.

4.5 Aplicación en AR

A pesar de no ser perfecta la reconstrucción, ya que debido al ruido, genera imperfecciones en la capa profunda de la herida, imperfecciones que pueden llegar a ser despreciadas, en este modelo se tiene información importante como son: la nube de puntos, vértices, caras, superficies y texturas. Lo que lo convierte en un archivo con demasiados datos para hacer una trasmisión en tiempo real y la banda de comunicación no soporta esta capacidad de transmisión, por lo tanto, es necesario hacer un tratamiento para realizar una reducción en el número de caras, dejándolo por debajo de las 40.000 caras, adicionalmente el modelo es guardado en escala real.

Para la realización de la aplicación se utilizará Vuforia Engine para Unity, ya que tiene integrado un toolkit de reconocimiento de imágenes, se utilizará una imagen de referencia a forma de marcador, que servirá para que la aplicación detecte que debe mostrar el modelo y con ello de cargarán los modelos previamente procesados por la Intel Realsense en el PC, es decir, no se hará transmisión en vivo del modelo sino que se guardará el modelo en un archivo en formato .obj y se hará una solicitud vía Wi-fi al PC para cargar este archivo en el dispositivo Android, el modelo podrá ser actualizado por una función del programa en Unity que se encarga de guardar los datos generados por el script CloudPoint para obtener un Frame de la escena que será proyectado a la aplicación de Realidad aumentada.

Este modelo 3D es exportado en formato .fbx y puede ser visto en cualquier programa de edición 3D, se almacena en la base de datos de Vuforia, para posteriormente agregarlo en Unity, donde se coloca una imagen como marcador en la aplicación, arrojando los siguientes resultados, mostrados en las figuras 4.9 y 4.10.

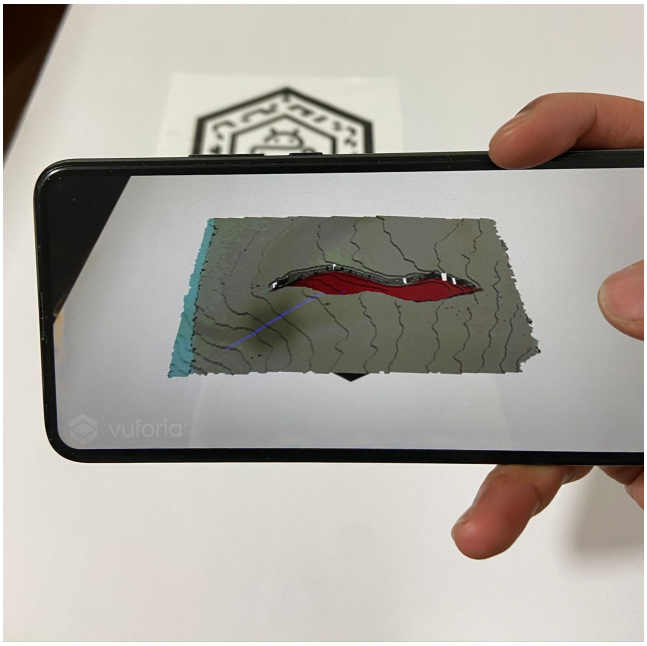


Figura 4.9: Resultados del APK en dispositivo Android (celular en horizontal)

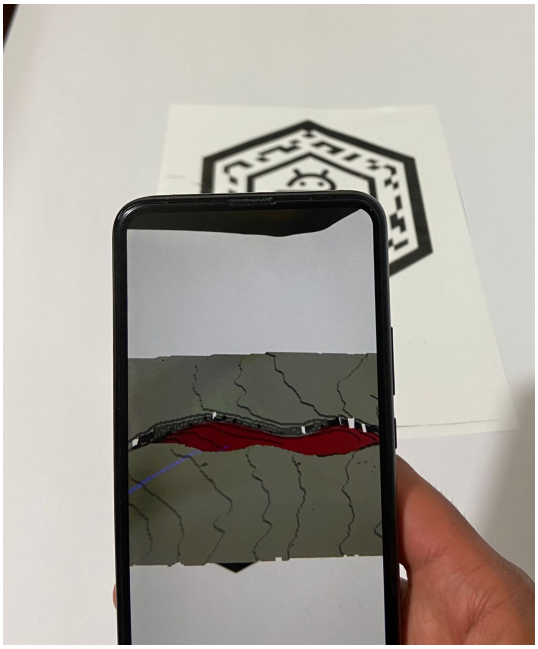


Figura 4.10: Resultados del APK en dispositivo Android (celular en vertical)

Esto se hace para poder que se reproduzca la reconstrucción en cualquier dispositivo que soporte la trasmisión de AR, en este caso se usa un celular Android, dado que los celulares con sistema IOS no tienen una muy buena compatibilidad con aplicativos APK, por temas de seguridad y privacidad de Icloud, lo que limita la investigación y restringe el uso en dispositivos móviles.

El APK permite la instalación de la aplicación de visualización, y es generado desde Unity 3D, como resultados de la visualización del marcador desde el celular

a través de la cámara, usando el APK, se obtiene como resultado la figura 4.11, cabe aclarar que el celular puede estar en cualquier posición sin perder de vista el marcador que la reconstrucción se adapta al ángulo de visión de la cámara del celular.

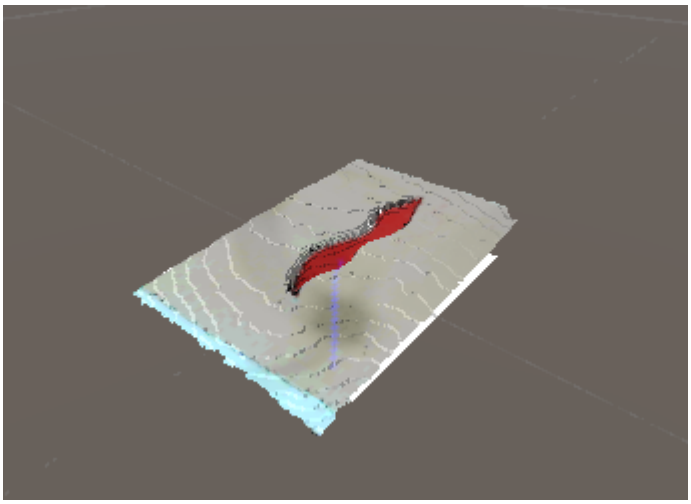


Figura 4.11: Visualización en Unity

El marcador o target utilizado para la reconstrucción en AR debe ser una forma básica y con un mapeado de color uniforme, para que se puede hacer una vectorización de color correcta y se realice la reproducción del sólido de manera satisfactoria, por tanto, no se pudo utilizar una imagen 2D de la herida debido a que se pierde calidad y forma conflicto con algunos píxeles generando deformaciones notorias en las reconstrucciones y perdidas de foco, en algunos casos no llega ni a realizar la reconstrucción, en la documentación recomiendan usar figuras simples y que tengan un contraste de color (blanco y negro es el mejor ejemplo), para así lograr una reconstrucción de manera exitosa en AR.

4.6 Trabajos futuros

Teniendo en cuenta los problemas que se afrontaron a lo largo del desarrollo del proyecto, se recomienda:

- Usar una cámara multicanal para poder hacer una reconstrucción mucho más rápida y evitar problemas de comunicación entre los diferentes medios usados para la reconstrucción.
- Si se desea una mejor calidad de reconstrucción se puede usar una cámara dedicada a la visión médica ya que la utilizada es comúnmente usada en aplicativos de visión de máquina industrial (pick and place, clasificación a gran escala, etc).
- Para hacer un tratamiento de imagen y reconstrucción en tiempo real, se deben tener en cuenta equipos de alto rendimiento y canales de comunicación que soporten altas tasas de envío y recepción de datos, ya que la nube de puntos trabaja con más de 500.000 puntos.

5 Conclusiones

Se logra construir un escenario, simulando el entorno de un quirófano, implementando elementos de bajo costo, dicho escenario es adecuado para la reconstrucción 3D de una laceración en la piel, donde se realiza la captura y extracción de características físicas (morfología, tamaño, texturas y color).

El escenario y el mecanismo de captura son dependientes de la luminosidad del entorno, por lo tanto, se debe tener un buen sistema de iluminación para evitar la pixelación de algunos datos en la nube de puntos; El mecanismo de captura es dependiente de la distancia entre la cámara y el objeto, para este escenario de pruebas se obtuvieron mejores resultados con distancias entre los 10 y 50 centímetros, se recomienda hacer un ajuste manual hasta obtener la captura con las mejores condiciones.

Durante el proceso de reconstrucción del phamton se logra apreciar que usar los filtros Bilinear y Trilinear usados en Unity de forma conjunta no generan gran mejora, solo se logra volver más pesado el archivo del objeto 3D y dificultar el procesamiento de imagen, se concluye que basta con usar solamente el filtro Bilinear que aporta mejoramiento y no crea un archivo tan pesado permitiendo la fácil manipulación de los datos para hacer el filtrado y el tratamiento de imágenes correspondiente.

Se tuvieron resultados satisfactorios en la obtención de modelos 3D de heridas y laceraciones en un entorno virtualizado para su aplicación en robótica médica con recursos de fácil acceso, obteniendo como resultado una reconstrucción de la herida conservando la morfología (forma), el tamaño (largo, ancho, y profundidad), el color, pero las texturas sufren unas pequeñas deformaciones que son despreciables debido a su tamaño, estas son generadas por el ruido captado en el modelo en escala de grises que se usa como base para la reconstrucción en el aplicativo de AR.

Hasta el momento no se cuenta con una librería o un software que permita hacer la reconstrucción 3D en tiempo real en la aplicación de AR (Vuforia), debido al gran tráfico de información entre los canales de comunicación, haciéndose necesario el uso de equipos de alto rendimiento y prestaciones, incrementando así el costo del experimento, esto se puede mejorar con la nueva generación de comunicaciones inalámbricas, como lo es el Wi-Fi 6.0, según la Wi-Fi Alliance .

Bibliografía

[1] C. Liu, X. Fan, Z. Guo, Z. Mo, E. I.-C. Chang **and** Y. Xu, «Wound area measurement with 3D transformation and smartphone images,» *BMC Bioinformatics*, **jourvol** 20, **number** 1, **page** 724, **december** 2019, issn: 1471-2105. doi: 10.1186/s12859-019-3308-1. **url**: https://doi.org/10.1186/s12859-019-3308-1 (**urlseen** 06/04/2021).

[2] D. Filko, R. Cupec **and** E. K. Nyarko, «Detection, Reconstruction and Segmentation of Chronic Wounds Using Kinect v2 Sensor,» en, *Procedia Computer Science*, 20th Conference on Medical Image Understanding and Analysis (MIUA 2016), **jourvol** 90, **pages** 151–156, **january** 2016, issn: 1877-0509. doi: 10.1016/j.procs.2016.07.022. **url**: https://www.sciencedirect.com/science/article/pii/S1877050916312005 (**urlseen** 06/04/2021).

[3] L. Camacho, «Sistema de reconstrucción 3D multicamara,» Tesis doctoral, Centro de investigaciones en óptica, Leon, Guanajuato, 2013.

[4] M. Z. Brown, D. Burschka **and** G. D. Hager, «Advances in computational stereo,» *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **jourvol** 25, **number** 8, **pages** 993–1008, **august** 2003, Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence, issn: 1939-3539. doi: 10.1109/TPAMI.2003.1217603.

[5] N. Grandón-Pastén, D. Aracena-Pizarro **and** C. L. Tozzi, «RECONSTRUCCIÓN DE OBJETO 3D A PARTIR DE IMÁGENES CALIBRADAS,» *Ingeniare. Revista chilena de ingeniería*, **jourvol** 15, **number** 2, **pages** 158–168, **august** 2007, Publisher: Universidad de Tarapacá, issn: 0718-3305. doi: 10.4067/S0718-33052007000200006. **url**: https://scielo.conicyt.cl/scielo.php?script=sci_abstract&pid=S0718-33052007000200006&lng=es&nrm=iso&tlng=es (**urlseen** 06/04/2021).

[6] R. Koch, «3-D surface reconstruction from stereoscopic image sequences,» *Proceedings of IEEE International Conference on Computer Vision*, 1995. doi: 10.1109/ICCV.1995.466799.

[7] F. Remondino, «3-D reconstruction of static human body shape from image sequence,» en, *Computer Vision and Image Understanding*, **jourvol** 93, **number** 1, **pages** 65–85, **january** 2004, issn: 1077-3142. doi: 10.1016/j.cviu.2003.08.006. **url**: https://www.sciencedirect.com/science/article/pii/S1077314203001243 (**urlseen** 06/04/2021).

[8] M. Pollefeys, L. V. Gool, M. Vergauwen, K. Cornelis, F. Verbiest **and** J. Tops, «3D recording for archaeological fieldwork,» *IEEE Computer Graphics and Applications*, **jourvol** 23, **number** 3, **pages** 20–27, **may** 2003, Conference Name: IEEE Computer Graphics and Applications, issn: 1558-1756. doi: 10.1109/MCG.2003.1198259.

[9] D. Esteban López **andothers**, «Reconstrucción de superficies a partir de nubes de puntos en CATIA V5,» Tesis, Universidad de Valladolid, Escuela de Ingenierías Industriales, 2015.

[10] B. Bedoya **and** J. Willian, «Reconstrucción de objetos de forma libre a partir de imágenes de rango empleando una red de parches nurbs,» en, Accepted: 2019-06-25T00:29:33Z, phdthesis, Universidad Nacional de Colombia, 2007. **url**: https://repositorio.unal.edu.co/handle/unal/11688 (**urlseen** 06/04/2021).

[11] A. Ceballos Fernandez, «Diseño de un diseño de realidad aumentada con aplicaciones a la cirugía mínimamente invasiva,» spa, Accepted: 2017-02-01T12:39:22Z, Tesis de maestría, Universidad de Alcalá, 2016. **url**: https://ebuah.uah.es/dspace/handle/10017/28099 (**urlseen** 06/04/2021).

[12] E. Pérez Hernández, «Técnicas de relleno de huecos en superficies adquiridas mediante escáneres 3D,» spa, **october** 2011, Publisher: Universidad Nacional de Educación a Distancia (España). Escuela Técnica Superior de Ingeniería Informática. Departamento de Ingeniería de Software y Sistemas Informáticos. **url**: http://e-spacio.uned.es/fez/view/tesisuned:IngInf-Eperez (**urlseen** 06/04/2021).

[13] R. M. Toro **and** B. A. Piñero, *Estudio de un sistema de reconstrucción 3D en tiempo real*, es, Tesis en ingeniería, Escuela superior de ingenieros, Universidad de Sevilla, España, 2005.

[14] H. Wechsler, «Digital image processing, 2nd ed.,» *Proceedings of the IEEE*, 1981. doi: 10.1109/PROC.1981.12153.

[15] J. Madrigal, I. Ramírez, J. Vargas **and** F. H. Campo, *(Modelación Geométrica de un Escáner 3D Mediante la Técnica del Sheet Of Light Usando Matlab)*, es, Seminario: Programación de Hardware de Bajo Costo con MATLAB, EAFIT.

[16] T. Ju, «Fixing Geometric Errors on Polygonal Models: A Survey,» en, *Journal of Computer Science and Technology*, **jourvol** 24, **number** 1, **pages** 19–29, **january** 2009, issn: 1860-4749. doi: 10.1007/s11390-009-9206-7. **url**: https://doi.org/10.1007/s11390-009-9206-7 (**urlseen** 06/04/2021).

[17] Y. Qu, J. Huang **and** X. Zhang, «Rapid 3D Reconstruction for Image Sequence Acquired from UAV Camera,» en, *Sensors*, **jourvol** 18, **number** 1, **page** 225, **january** 2018, Number: 1 Publisher: Multidisciplinary Digital Publishing Institute. doi: 10.3390/s18010225. **url**: https://www.mdpi.com/1424-8220/18/1/225 (**urlseen** 06/04/2021).

[18] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein **and** R. Szeliski, «A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms,» **in** *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, ISSN: 1063-6919, **volume** 1, **june** 2006, **pages** 519–528. doi: 10.1109/CVPR.2006.19.

[19] C. Strecha, W. v. Hansen, L. V. Gool, P. Fua **and** U. Thoennessen, «On benchmarking camera calibration and multi-view stereo for high resolution imagery,» **in** *2008 IEEE Conference on Computer Vision and Pattern Recognition*, ISSN: 1063-6919, **june** 2008, **pages** 1–8. doi: 10.1109/CVPR.2008.4587706.

[20] M. Nießner, M. Zollhöfer, S. Izadi **and** M. Stamminger, «Real-time 3D reconstruction at scale using voxel hashing,» en, *ACM Transactions on Graphics*, **jourvol** 32, **number** 6, **pages** 1–11, **november** 2013, issn: 0730-0301,

1557-7368. doi: 10.1145/2508363.2508374. **url**: https://dl.acm.org/doi/10.1145/2508363.2508374 (**urlseen** 06/04/2021).

[21] E. N. Johnson **and** J. G. Mooney, «A Comparison of Automatic Nap-of-the-earth Guidance Strategies for Helicopters,» en, *Journal of Field Robotics*, **jourvol** 31, **number** 4, **pages** 637–653, 2014, issn: 1556-4967. doi: https://doi.org/10.1002/rob.21514. **url**: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21514 (**urlseen** 06/04/2021).

[22] D. Cremers, «Direct methods for 3D reconstruction and visual SLAM,» **in** *2017 Fifteenth IAPR International Conference on Machine Vision Applications (MVA)*, **may** 2017, **pages** 34–38. doi: 10.23919/MVA.2017.7986766.

[23] L. Chen, W. Tang, N. W. John, T. R. Wan **and** J. J. Zhang, «SLAM-based dense surface reconstruction in monocular Minimally Invasive Surgery and its application to Augmented Reality,» en, *Computer Methods and Programs in Biomedicine*, **jourvol** 158, **pages** 135–146, **may** 2018, issn: 0169-2607. doi: 10.1016/j.cmpb.2018.02.006. **url**: https://www.sciencedirect.com/science/article/pii/S0169260717301694 (**urlseen** 06/04/2021).

[24] O. O. Arenaza, J. C. B. Arroyo **and** A. S. d. O. Blanco, «Caracterización geomorfológica y análisis de la evolución del deslizamiento rotacional de Andoin, Sierra de Entzia (País Vasco).,» es, *Cuaternario y Geomorfología*, **jourvol** 31, **number** 3-4, **pages** 7–26, **december** 2017, issn: 2695-8589. doi: 10.17735/cyg.v31i3-4.55240. **url**: https://recyt.fecyt.es/index.php/CUGEO/article/view/55240 (**urlseen** 06/04/2021).

[25] D. Filko, E. K. Nyarko **and** R. Cupec, «Wound detection and reconstruction using RGB-D camera,» **in** *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, **may** 2016, **pages** 1217–1222. doi: 10.1109/MIPRO.2016.7522325.

[26] B. Martín Valverde, «Análisis y reconocimiento de objetos mediante tecnología Time of Flight,» spa, **june** 2012, Accepted: 2013-04-29T08:33:37Z. **url**: https://e-archivo.uc3m.es/handle/10016/16847 (**urlseen** 06/04/2021).

[27] A. Loidi Yarza, R. Estop Remacha, E. d. l. Fuente, J. C. Fraile **and** J. Pérez Turiel, «Sistema de visión para el guiado de un asistente robótico en operaciones de cirugía endonasal,» spa, Accepted: 2019-08-09T09:53:39Z Journal Abbreviation: Vision system for the guidance of a robotic assistant in endonasal surgery operations, Universidade da Coruña, Servizo de Publicacións, 2019, **pages** 94–100, isbn: 978-84-9749-716-9. **url**: https://ruc.udc.es/dspace/handle/2183/23668 (**urlseen** 06/04/2021).

[28] N. De, *Reconstrucción 3D del cuerpo humano mediante puntos de referencia Dedicatoria*, Tesis de licenciatura, UNAM, Facultad de Ingeniería, 2018. **url**: https://repositorio.unam.mx/contenidos/261512.

[29] L. Keselman, J. I. Woodfill, A. Grunnet-Jepsen **and** A. Bhowmik, «Intel RealSense Stereoscopic Depth Cameras,» *arXiv:1705.05548 [cs]*, **october** 2017, arXiv: 1705.05548. **url**: http://arxiv.org/abs/1705.05548 (**urlseen** 06/04/2021).

[30] Runyang Zou, Xueshi Ge **and** Geng Wang, «Applications of RGB-D data for 3D reconstruction in the indoor environment,» **in** *2016 IEEE Chinese Guidance, Navigation and Control Conference (CGNCC)*, **august** 2016, **pages** 375–378. doi: 10.1109/CGNCC.2016.7828814.

[31] F. Li, S. Liu, L. Jiang, W. Zhang **and** J. Zhou, «Novel Use of the Intel RealSense SR300 Camera for Foot 3D Reconstruction,» en, *Leather and Footwear Journal*, **jourvol** 20, **number** 2, **pages** 145–152, **june** 2020, issn: 15834433. doi: 10.24264/lfj.20.2.5. **url**: http://revistapielarieincaltaminte.ro/revistapielarieincaltaminteresurse/en/fisiere/full/vol20-nr2/article5_vol20_issue2.pdf (**urlseen** 06/04/2021).

[32] M. Deseilligny **and** I. Clery, «APERO, an open source bundle adjusment software for automatic calibration and orientation of set of images,» *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, **jourvol** XXXVIII-5/W16, **january** 2011. doi: 10.5194/isprsarchives-XXXVIII-5-W16-269-2011.

[33] N. Wake, H. Chandarana, W. C. Huang, S. S. Taneja **and** A. B. Rosenkrantz, «Application of anatomically accurate, patient-specific 3D printed models from MRI data in urological oncology,» eng, *Clinical Radiology*, **jourvol** 71, **number** 6, **pages** 610–614, **june** 2016, issn: 1365-229X. doi: 10.1016/j.crad.2016.02.012.

[34] N. Wake, M. A. Bjurlin, P. Rostami, H. Chandarana **and** W. C. Huang, «Three-dimensional Printing and Augmented Reality: Enhanced Precision for Robotic Assisted Partial Nephrectomy,» eng, *Urology*, **jourvol** 116, **pages** 227–228, **june** 2018, issn: 1527-9995. doi: 10.1016/j.urology.2017.12.038.

[35] F. Porpiglia, E. Checcucci, D. Amparore, M. Manfredi, F. Massa, P. Piazzolla, D. Manfrin, A. Piana, D. Tota, E. Bollito **and** C. Fiori, «Three-dimensional Elastic Augmented-reality Robot-assisted Radical Prostatectomy Using Hyperaccuracy Three-dimensional Reconstruction Technology: A Step Further in the Identification of Capsular Involvement,» eng, *European Urology*, **jourvol** 76, **number** 4, **pages** 505–514, **october** 2019, issn: 1873-7560. doi: 10.1016/j.eururo.2019.03.037.

A Anexos

Para evitar confusion o errores tipográficos en los codigós, se adjuntan en la carpeta en el siguiente link: dar click aquí y para la implementación también adjuntamos un vídeo explicativo en el mismo link.

También se encuentran explicados en el repositorio de tesis de Github.

University of
Cauca
F.I.E.T
Tulcan