



TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE APIZACO

ESCANER 3D USANDO RASPBERRY PI 4

TITULACIÓN INTEGRAL INFORME DE RESIDENCIA PROFESIONAL

PARA OBTENER EL TÍTULO PROFESIONAL DE: INGENIERO EN MECATRONICA

**PRESENTA:
JESÚS HERNÁNDEZ LOZADA**

**NOMBRE DEL ASESOR:
DR. FEDERICO CASCO VÁSQUEZ**



Apizaco, Tlaxcala, **18/agosto/2023**

Departamento: Ingeniería Eléctrica y Electrónica
No. OFICIO IEE/443/2023

Asunto: Liberación de Proyecto
para la Titulación Integral

MARTIN ROJAS RAMÍREZ
JEFE DE LA DIVISIÓN DE ESTUDIOS PROFESIONALES
P R E S E N T E.

Por este medio me permito informar a usted que ha sido liberado el siguiente proyecto para la Titulación Integral:

Nombre del egresado:	JESÚS HERNÁNDEZ LOZADA
Carrera:	INGENIERÍA MECATRÓNICA
No. de Control:	16370843
Nombre del proyecto:	"ESCANER 3D USANDO RASPBERRY PI 4".
Producto:	TITULACIÓN INTEGRAL INFORME DE RESIDENCIA PROFESIONAL

Agradeciendo de antemano su valioso apoyo en esta importante actividad para la formación profesional de nuestros egresados, le envié un cordial saludo.

ATENTAMENTE

*Excelencia en Educación Tecnológica
Pensar para Servir, Servir para Triunfar.*

HÉCTOR ESCOBAR JARDINES
JEFE DEL DEPTO. DE INGENIERÍA
ELÉCTRICA Y ELECTRÓNICA




JOSÉ FEDERICO CASCO VÁSQUEZ
Nombre Y Firma y Asesor

C.c.p Expediente





Apizaco, Tlax., **12/septiembre/2023**
No. OFICIO: DEP-CT/645/2023

ASUNTO: AUTORIZACIÓN DE IMPRESIÓN

JESUS HERNANDEZ LOZADA
EGRESADO DE LA CARRERA DE INGENIERÍA MECATRÓNICA
CON NÚMERO DE CONTROL: 16370843
P R E S E N T E

Por este medio, me permito informar a usted que por aprobación de la comisión revisora asignada para valorar el trabajo que mediante la opción **TITULACIÓN INTEGRAL INFORME DE RESIDENCIA PROFESIONAL** cuyo tema es: **"ESCANER 3D USANDO RASPBERRY PI 4"**, conforme a lo establecido en el procedimiento para la obtención del Título Profesional en el Instituto Tecnológico, la División de Estudios Profesionales a mi cargo le otorga:

AUTORIZACIÓN DE IMPRESION

Sin otro particular por el momento, le envió un cordial saludo.

ATENTAMENTE

Excelencia en Educación Tecnológica
Pensar para Servir, Servir para Triunfar



SECRETARÍA DE EDUCACIÓN PÚBLICA
TECNOLÓGICO NACIONAL
DE MÉXICO


MARTÍN ROJAS RAMÍREZ INSTITUTO TECNOLÓGICO DE APIZACO
JEFE DE LA DIVISIÓN DE ESTUDIOS PROFESIONALES
ESTUDIOS PROFESIONALES

c.p. archivo



Agradecimientos.

En primer lugar, deseo expresar mi agradecimiento a los docentes que nos brindaron ayuda en la asesoría de esta tesis de residencias, Dr. Federico Casco Vásquez y Mtro. Mario Eduardo Leal López, por la dedicación y apoyo que ha brindado a este trabajo, dándome la oportunidad de recurrir a su capacidad y conocimiento, por el respeto a mis sugerencias e ideas y por la dirección y el rigor que ha facilitado a las mismas, así como también a agradezco a la Universidad “Instituto Tecnológico de Apizaco” por haberme aceptado, formado parte de ella y abierto las puertas de su seno educativo para poder estudiar mi carrera, al igual a los diferentes docentes que brindaron sus conocimientos y apoyo para seguir adelante día a día.

Y para finalizar agradezco también a mis amigos y familiares por todo el apoyo moral brindado en este tiempo y siempre.

Resumen.

La tecnología avanza y se renueva mientras avanza el tiempo cada día las nuevas generaciones requieren de aprender nuevos conocimientos que fortalezcan su educación y estos sean aplicados en beneficio a la sociedad. De este modo este proyecto tiene una gran posibilidad de aplicaciones y áreas de estudio para investigación en las cuales predomina la programación y el uso de softwares que ayudan a la reconstrucción 3D además de herramientas de diseño CAD.

Es por eso por lo que el objetivo principal de este proyecto es realizar un escáner 3D, donde se buscó optimizar costos en el uso de este tipo de escáneres de la tal manera que se pueda hacer una representación morfológica digital de objetos reales usando sistemas más económicos, donde el trabajo se realizó a través de softwares como SolidWorks, que ayuda para el diseño del prototipado, así como también se utilizó el apoyo en los softwares de Matlab, Python, para la programación de la nube de puntos frames y Meshlab y Cura para el modelado y depuración de la figura para poder así imprimirla escala 3D.

Índice

Agradecimientos.....	1
Resumen.....	2
Índice.....	3
Introducción.....	5
Descripción de la empresa u organización y del puesto o área del trabajo del estudiante.....	6
Problemas para Resolver, priorizándolos.....	6
Problemas y falta de información en el escaño	6
Archivo faltante para impresión 3D	6
Objetivos (General y Específicos).....	8
General	8
Específicos	8
Justificación.....	9
Estado de la Técnica.....	10
.....	10
Marco teórico (fundamentos teóricos).....	18
Procesamiento Digital de Imágenes	18
Cuantificación	23
Formatos de imágenes digitales	24
Modo de color	25
Desarrollo.....	44
Resultados	72
Conclusiones	79

Competencias desarrolladas.....	80
Fuentes de información.....	83
Anexos	85
Código de Cámara Raspberry:	85

Introducción.

Este proyecto fue creado a partir de la necesidad de poder digitalizar piezas mecánicas utilizando los recursos disponibles a nuestra disposición contando con una tarjeta raspberry pi 4 la cual utilizamos como centro de control gracias a su amplia aplicación y sus dimensiones reducidas.

Entre las ventajas con las que cuenta una microcomputadora que tiene un sistema operativo como raspbian, con el cual se logró programar en lenguaje de python, y gracias a sus pines de entrada y salida denominados GPIO nos permiten una amplia variedad de configuraciones gracias a librerías precargadas en el sistema dichos pines se utilizaron para el control y manejo de motores a pasos el cual nos servía para poder automatizar el movimiento de la pieza y poder girarla 360° para obtener la mayor información posible además de poder controlar la activación y control de iluminación el cual nos servía para la obtención del color de la pieza o por lo menos un aproximado al color real y así mismo sucedía con el láser utilizado para delimitar el contorno de las figuras y aquí cuando se activaba el láser la luz debería de estar apagada de esta manera gracias a los ángulos generados por el láser y la cámara podíamos obtener la información sobre el contorno de la pieza.

Para poder obtener la morfología de alguna pieza se requirió de 2 aspectos indispensables para la obtención de datos, uno de ellos el ángulo del láser con respecto al ángulo de la cámara frente a la pieza y el otro la altura y de formación causada por el lente de la cámara también llamada aberración esférica que no son más que errores refractivos en el espacio de fourier y con una buena calibración se puede ajustar y corregir ese tipo de error y obtener lo más real a la pieza original.

Por otro lado utilizamos el programa de matlab para poder realizar el procesamiento de imágenes y poder realizar tratamientos a las imágenes obtenidas por la tarjeta raspberry pi 4 comenzando por la extracción de los fotogramas ya que con la tarjeta raspberry pi 4 obtenemos la digitalización mediante vídeos obtenidos por 2 cámaras simultáneamente y una vez dentro de matlab extraemos la mayor cantidad posible de fotogramas tratando siempre de que el número de fotogramas nos diera un fotograma por cada grado en este caso poder obtener 360 fotogramas y posteriormente se realizó.

Descripción de la empresa u organización y del puesto o área del trabajo del estudiante.

En el instituto tecnológico de Apizaco hay una gran variedad de áreas en que trabajar en específico nos concentramos en una en especial en la cual se está trabajando y es el área de optoelectrónica donde integramos el dominio/campo de la automatización de sistemas, desde programación en general hasta el control del sistema, el procesamiento y digitalización de imágenes, como también el diseño 3D mediante imágenes y datos digitales, los cuales se manipulan de manera manual para procesar mediante algoritmos que procesan las imágenes digitalmente por filtros los cuales son tratamientos basados en matrices.

Problemas para Resolver, priorizándolos.

Algunas de las problemáticas al iniciar este proyecto y en su desarrollo que surgieron fueron las siguientes a continuación mencionadas:

Problemas y falta de información en el escaneo

- generación de nubes de puntos (1 o 2 nubes)
- uso de filtros (según sea el caso de la cama)
- unión de nubes de puntos
- adquisición de datos con color real
- generación de objetos 3 d con niveles de gris
- reconstrucción 3D

Archivo faltante para impresión 3D

- el mallado y conversión ha solido
- conversión de extensión para la lectura de archivo
- generación de archivo para impresión
- problemas en el sistema de control (hardware)
- entre los cuales se encuentran
- entrada del objeto al sistema de escaneo
- centrado del objeto
- centrado del láser
- su entrada del cámara(s)
- control actualizado de encendido (botones ocultos)
- rediseño o diseño de fuente(s) (Motores del sistema iluminación y laser)
- calibración del sistema semi automatizado
- adaptación de una pantalla touch
- problemas en el diseño y armado en la estructura del sistema

Objetivos (General y Específicos).

General

- Analizar, desarrollar y mejorar los elementos de software como de hardware que se requieren para el funcionamiento de un escáner 3D generando modelos digitales que sirvan como herramienta digital de diseño que funcionen con un sistema mas económico.

Específicos

- Reconstruir físicamente a partir de la pieza digitalizada en el escáner 3D .
- Diseñar o rediseño del sistema de escaneo 3D
- Mejorar en la técnica de adquisición de datos.
- Implementar un sistema integrado automatizado.
- Realizar pruebas de escaneo.
- Analizar softwares de tratamiento de nube de puntos para enmallado.
- Realizar pruebas de impresión 3D.
- Presentar resultados

Justificación.

En el banco de proyectos del Departamento de Ingeniería Eléctrica – Electrónica del Instituto Tecnológico de Apizaco, con intención de seguir generando más conocimiento hacia las áreas de electrónica se cuenta con un proyecto de investigación enfocado en la reconstrucción 3D a partir de un escáner 3D de sistema pasivo usando una tarjeta Raspberry Pi 4. La presente problemática que nos llevó a desarrollar un prototipo de esta tecnología nos aporta los conocimientos suficientes para posteriores investigaciones de las áreas que se han aplicado, así como la metodología ideal para la resolución del problema.

El presente trabajo pretende aportar las bases necesarias para la construcción de un escáner 3D tomando en cuenta la metodología, limitaciones y diseño para poder implementarlo a partir de dos procesos, el primer proceso consta de una tarjeta Raspberry Pi 4 como sistema de control en donde más adelante se realiza un análisis de los componentes de hardware y software que conforman esta parte elemental del escaneo además del segundo proceso que lo conforma una cámara de video y un láser los cuales se mantendrán en una cierta posición que permiten el escaneo 3D sin contacto siendo este denominado sistema pasivo de escaneo 3D, la adquisición de la información de escaneo se procesa en un software llamado Matlab en cual se realiza el filtrado necesario de la fotometría para generar una nube de puntos que contiene la información necesaria para la reconstrucción de la geometría de la superficie de la pieza escaneada, también se usa el software Meshlab que es de código abierto el cual se especializa en enmallado de nubes de puntos.

Por último, se usó del software UltiMaker CURA de código abierto el cual se usa para impresión 3D en el cual se le dan las dimensiones al modelo digital escaneado para convertirlo a un modelo físico.

El acoplamiento de estos procesos genera una herramienta para una gran posibilidad de aplicaciones además de que su construcción es más económica a comparación de los que ya existen en el mercado. Por lo que en este trabajo se desarrollará la metodología para la implementación y desarrollo de un escáner 3D.

Estado de la Técnica

Tabla 1. Estado de la Técnica / Estado del Arte.

No	Artículo, reporte técnico o trabajo profesional	Resumen o Abstract	Autores o referencias	Año de publicación	Qué aporta el artículo al trabajo de residencia
1	Identificación humana mediante superposición de imágenes. Una propuesta metodológica	Se presenta un nuevo método para la identificación humana por medio de análisis de imágenes y superposición fotográfica. Se discuten las ventajas que aporta la incorporación de imágenes tridimensionales del cráneo, ya que, por un lado, facilitan la superposición y por otro, eliminan errores de escalado al estar el modelo 3D a tamaño real. Este trabajo se ha desarrollado por un equipo interdisciplinar y su objetivo	1. Alemán, 2. MC. Botella, 3. F. Navarro, 4. Ó. Cordón, 5. S. Damas 6. J. Santamaría	Octubre, 2008	Este artículo nos describe como se construye una figura o imagen 3D, aplicado en el área de la salud neuro-craneal, el cual nos sirve por el conocimiento en la superposición de las imágenes para un construcción de una figura 3D además de la detección de bordes colocando puntos en un espacio y generando modelos más exactos y corrigiendo errores de escalado.

		fundamental es proporcionar una herramienta semiautomática de identificación humana, basada en el reconocimiento craneofacial.			
2	Impresión 3D de maquetas y prototipos en arquitectura y construcción	La fabricación de maquetas y prototipos en el ámbito de la arquitectura y la construcción ha tenido, y tiene todavía, una componente	1. Domínguez, I.A. 2. Romero, L. 3. Espinosa, M.M. 4. Domínguez, M.	Noviembre, 2013	Este artículo nos describe cómo es la obtención de maquetas de construcción o piezas en la industria lo cual nos sirve por el

		artesanal muy importante. El desarrollo de los sistemas de diseño asistido, con su consiguiente evolución hacia los sistemas de modelado sólido y los actuales sistemas de modelado de información en edificación, ha permitido obtener maquetas electrónicas, imágenes fotorrealísticas y animaciones virtuales de la construcción con una calidad muy aceptable. Pero todavía no se puede decir lo mismo respecto de las maquetas físicas, obtenidas a partir de ese modelo electrónico del edificio a través de			conocimiento de obtención de la pieza mediante máquinas de impresión en 3 dimensiones y bases en manipulación de ficheros electrónicos, donde nos beneficia para la construcción sólida de la pieza.
--	--	---	--	--	---

		máquinas de fabricación rápida de maquetas y prototipos. En el presente trabajo se aborda el estado actual del arte en cuanto a la obtención de maquetas de construcción mediante máquinas de impresión en tres dimensiones (máquinas de fabricación rápida prototipos), analizando las dificultades que todavía no han sido subsanadas, y se plantea una línea de trabajo en cuanto a máquinas, procedimientos de trabajo y manipulación de los ficheros electrónicos que puede ayudar tanto al usuario final (para aprovechar al			
--	--	--	--	--	--

		máximo la tecnología actual), como al desarrollador de soluciones para la construcción, pues se marcan las pautas de lo que se persigue en este sector.			
3	Modelado 3D automático de segmentos hepáticos, incluidas las ramas segmentarias de la tríada portal y la vena hepática en función de las imágenes seccionadas.	Los modelos dimensionales (3D) del hígado, que constan de ocho segmentos que incluyen la tríada portal (vena porta, arteria hepática y conducto biliar), son necesarios porque es difícil diseccionar un hígado y sus estructuras internas. Pero es difícil producir modelos 3D a partir de imágenes seccionadas en color y de alta resolución. Este estudio presenta	J. Morphol.	Junio, 2018	el siguiente artículo describe cómo se construye un modelado 3 d aplicado en el área médica el cual nos sirve por la digitalización de modelados automáticos a partir de imágenes tomadas y seleccionadas de un grupo y que mejora el color y nos proporciona una alta resolución.

		métodos automáticos y precisos para producir modelos 3D de hígado a partir de imágenes seccionadas. Sobre la base de las imágenes seccionadas y las imágenes llenas de color del hígado, se realizó un modelo 3D que incluía tanto la tríada portal como la vena hepática.			
4	Multirresolución adaptativa de mallas triangulares basado en criterios de textura.	Los modelos 3D están generalmente compuestos por miles de polígonos. En ocasiones estas representaciones pueden obtenerse con la misma calidad visual pero con un menor número de polígonos. En este artículo se propone un método para reducir el tamaño de imágenes 3D texturadas basadas en mallas triangulares,	Barranquilla	Julio – Diciembre, 2007	El siguiente artículo aquí mencionado describe el uso de mallas poligonales en texturas de modelados 3D lo cual me brinda hoy el conocimiento de un nuevo método del mallado poligonal hoy para obtener la misma calidad con un menor número de

		conservando la calidad visual del modelo. Se introduce un criterio de textura que controla el proceso de decimación triangular. Para eliminar puntos sin necesidad de realizar una nueva triangulación, sobre la nube de puntos se usa un algoritmo poligonal de decimación. Para definir cuáles puntos deben ser removidos, se usa un filtro de Sobel 2D sobre la textura correspondiente. Se muestra que se puede usar el algoritmo para reducir los tiempos de carga, de renderización, de transferencia y de almacenamiento de una imagen 3D texturada.			polígonos en las nubes de puntos
--	--	--	--	--	---

5	Un algoritmo de corte de nubes de puntos para el procesamiento y copia de objetos 3D	En este artículo se propone un método que, tomando como base una nube de puntos de un objeto previamente escaneado, calcula una serie de cortes a lo largo del eje principal del objeto, el cual es estimado usando Análisis de Componentes Principales (PCA). Estos cortes, pueden ser impresos en diferentes tipos de materiales (poliestireno, MDF, espuma, cartón, madera, etc.) para luego obtener una reproducción a mayor escala del objeto escaneado. Por tratarse de puntos, se facilita el trabajo para ampliar el tamaño del objeto usando interpolación u otro método equivalente. El método propuesto no	1. Esmeide A. Leal Narváez 2. Nallig E. Leal Narváez, 3. Mario Mercado Coronado 4. Pablo Maestre Pérez	Enero-Junio, 2018	El siguiente artículo nos propone un método de procesamiento y copia de objetos 3D El cual nos sirve en la recreación física de los modelados 3D calculando una serie de cortes a lo largo del eje principal del objeto con lo cual nos permite ser impreso en una amplia variedad de polímeros.
---	--	--	---	---------------------------------	---

		necesita de representaciones intermedias como las mallas triangulares, ya que opera directamente sobre la nube de puntos. Se hicieron comparaciones del método propuesto con herramientas similares que se encuentran en el mercado, mostrando su eficiencia computacional en carga de datos y el despliegue y visualización.			
--	--	--	--	--	--

Marco teórico (fundamentos teóricos).

Procesamiento Digital de Imágenes

El procesamiento de imágenes digitales implica la utilización de algoritmos informáticos para crear, procesar, comunicar y visualizar imágenes digitales [1].

Los algoritmos de procesamiento de imágenes digitales se pueden emplear para:

- Convertir señales de un sensor de imágenes en imágenes digitales
- Mejorar la claridad y eliminar el ruido y otros artefactos
- Extraer el tamaño, escala y número de objetos de una escena

- Preparar imágenes para su visualización o impresión
- Comprimir imágenes para su comunicación a través de una red

¿Qué es una imagen?

Se entiende por imagen la representación visual de algo o alguien, es decir, una figura o ilustración que se asemeja a algo, que refleja la apariencia de algo, ya sea real o imaginario, concreto o abstracto. Generalmente llamamos imágenes hoy en día a las fotografías, los cuadros, las ilustraciones o las obras de diseño gráfico, así como las estatuas y, también al reflejo del espejo. La palabra imagen proviene del latín *imago*, traducible como “copia” o “imitación” [2].



Ilustración 1 Representaciones de una imagen

Entonces podemos entender que una imagen también es una representación plana de un objeto de tres dimensiones.

¿Qué es una imagen digital?

Una imagen es una función de dos dimensiones $f(x, y)$ de una intensidad de luz, donde x e y se refieren a las coordenadas del plano cartesiano y el valor de f en cualquier punto (x, y) es proporcional al brillo (o nivel de grises) de una imagen en cualquier punto, aspectos que combinados dan la sensación de proveer tres dimensiones, generando por lo tanto imágenes espaciales [3].

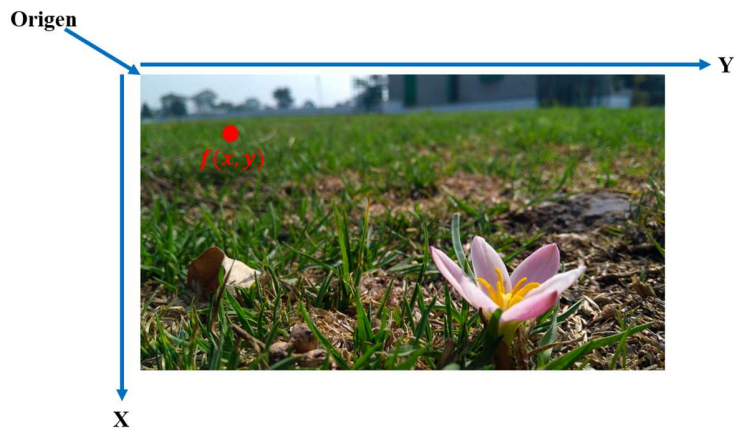


Ilustración 2 Representación de los ejes de una imagen digital

Podemos considerar una imagen digital como una matriz en la que la fila M y la columna N representan un punto en la imagen y el valor del elemento de la matriz corresponde al nivel de gris de ese punto. Los elementos de la matriz se llaman puntos o píxeles [4].

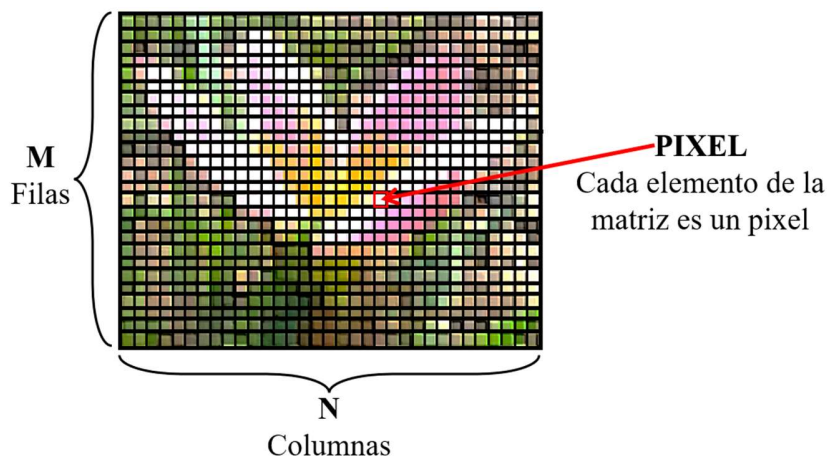


Ilustración 3. Representación en forma de matriz de una imagen digital

¿Qué es un píxel?

Para poder realizar un procesamiento digital de imágenes debemos entender cómo se conforma una imagen empezando por la parte más pequeña que la compone.

Como se mencionó anteriormente un píxel o punto es un elemento que hace parte de la matriz $M \times N$ (imagen digital) el cual toma un valor de profundidad o nivel de tono de gris y adquiere un valor digital. Normalmente se toman valores de N y para el número de niveles de gris en potencias de 2 [4].

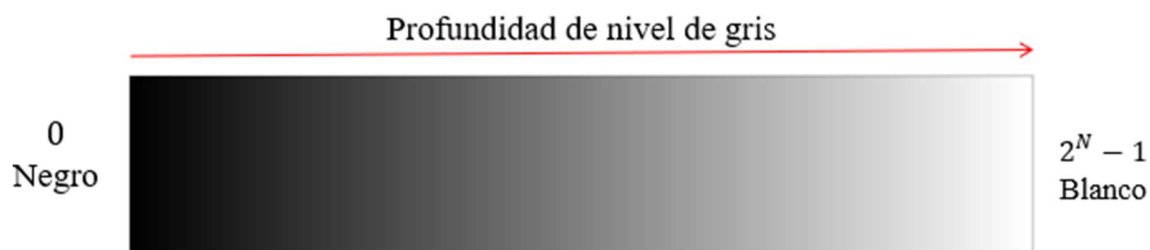


Ilustración 4. Representación de la profundidad de tonos de grises de una imagen digital

Dependiendo de los niveles de tonos de gris se le asigna un valor digital a cada píxel de la imagen digital por ejemplo en la ilustración 5 tenemos un cuadrado negro con fondo blanco. Cada punto con la representación digital corresponde al área del espacio del objeto, y un valor digital es asignado en cada punto de la imagen digital que se relaciona a la intensidad del área del espacio del objeto. El cuadrado negro se representa mediante ceros, y la intensidad del fondo blanco se representa por 1's en la imagen digital [5].

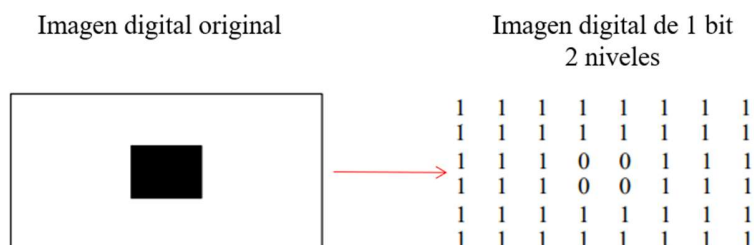


Ilustración 5. Ejemplo de una imagen digital de 2 niveles (negro y blanco)

Los bits se utilizan para representar intensidad de cada posición por ejemplo con 8 bits de profundidad puede haber 256 combinaciones posibles de esos 8 dígitos. Si disponemos de 8 bit para describir el color de cada píxel, podremos elegir entre 256 niveles. Esta es una profundidad de color suficiente para las imágenes construidas en el modo denominado escala de grises, porque con 8 bits cada pixel puede adoptar un tono entre 256 valores posibles de gris, entre el negro absoluto (00000000) y el blanco absoluto (11111111). Y así, cuanto mayor sea la profundidad se utilizará una cantidad mayor de colores para describir la imagen [6].

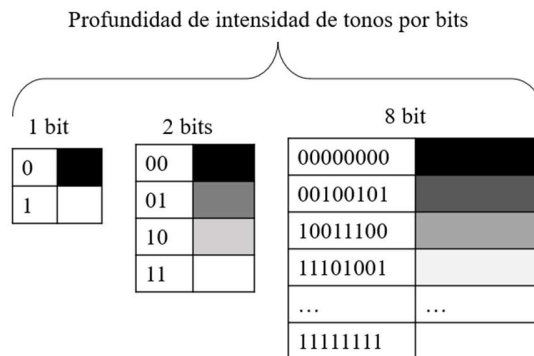


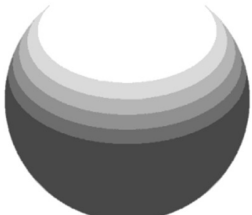
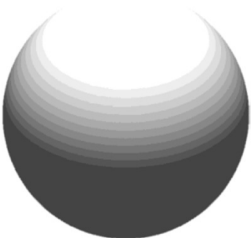
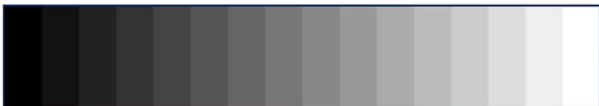
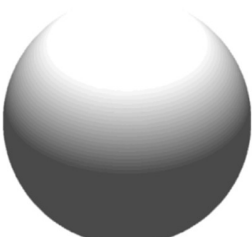
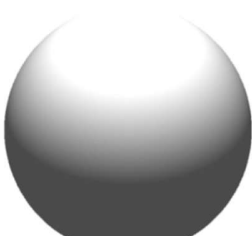
Ilustración 6. Representación de la profundidad de píxeles o profundidad de bits

Un píxel es una unidad de información, pero no una unidad de medida, ya que no se corresponde con un tamaño concreto. Un píxel puede ser muy pequeño (0.1 milímetros) o muy grande (1 metro). Cuando se modifica su tamaño, se modifican grupos de píxeles, no los objetos o figuras que contiene, por lo que estos suelen deformarse o perder alguno de los píxeles que los definen [5].

Para entender mejor se muestra la siguiente tabla en donde se puede apreciar la relación de los niveles de gris con los bits en este caso solo nos enfocamos en un solo canal.

Tabla 2. Relación de tonos de gris con los bits de profundidad de una imagen digital.

Imagen digital	Profundidad de tono
	1 bit 0 Negro $(2^1 - 1) = 1$ $N = 1$ 2 niveles Blanco
	2 bits 00 Negro $(2^2 - 1) = 3$ $N = 2$ 4 niveles Blanco

	
	000 Negro 3 bits  $(2^3 - 1) = 7$ $N = 3$ 8 niveles Blanco
	0000 Negro 4 bits  $(2^4 - 1) = 15$ $N = 4$ 16 niveles Blanco
	00000 Negro 5 bits  $(2^5 - 1) = 31$ $N = 5$ 32 niveles Blanco
	00000000 Negro 8 bits  $(2^8 - 1) = 255$ $N = 8$ 256 niveles Blanco

Cuantificación

Resolución: Es la cantidad de píxeles que definen la imagen [7]. Sus unidades de medida de los píxeles son por pulgada (ppp o ppi, pixels per inch, en inglés) o los píxeles por centímetro. Cuanto mayor sea esta resolución, más contenedores de información (píxeles) tiene el archivo digital, más calidad tendrá la imagen y más peso en Kb tendrá el archivo [5].

El “peso” de una imagen digital la determina la cantidad de píxeles que contiene como sabemos 8 bits es equivalente a 1 byte, 1024 bytes a 1Kb, 1024 kB a 1 MB y así sucesivamente, dependiendo su resolución podemos saber cuánto pesa aproximadamente una imagen.

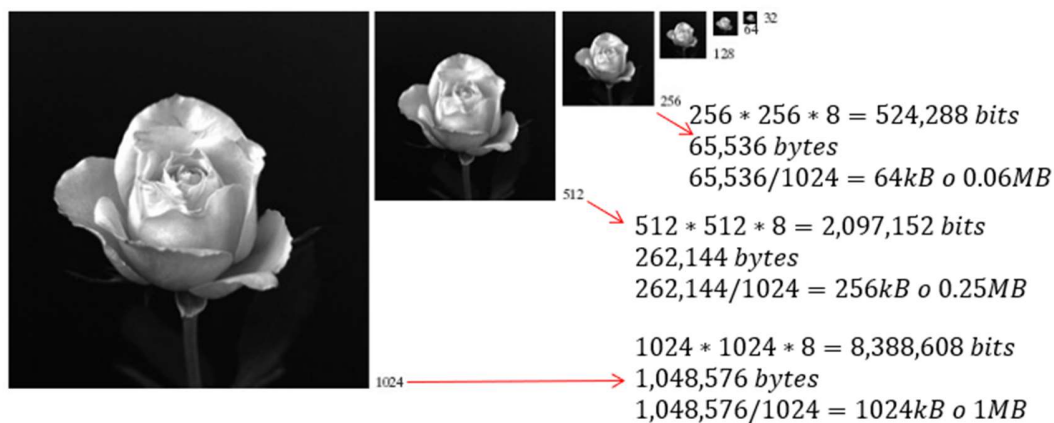


Ilustración 7 Ejemplo de la resolución de una imagen y sus respectivos pesos en bytes, kB y MB

La resolución de una imagen está relacionada con su tamaño, de tal forma que cuando se le asigna una resolución se estará asignando un tamaño a los píxeles que la forman, con lo que se sabe qué tamaño tiene la imagen. Por ejemplo, si una imagen tiene 100 píxeles por pulgada, querrá decir que cada 2,54 cm. habrá 100 píxeles, con lo que cada píxel equivaldrá a 2,54 mm. Si esa imagen tiene una resolución de 1 píxel por pulgada, lo que se concluye es que ahora cada píxel tendrá un tamaño de 2,54 cm. Otra consecuencia de la relación resolución-tamaño es que para mantener la calidad de reproducción, al variar el tamaño de una imagen, se tiene que variar también su resolución [5].

Formatos de imágenes digitales

Las imágenes digitales pueden presentarse en distintos formatos. El formato es algo así como el contenedor en el que se guarda la información. La elección del formato viene determinada por cómo interese preservar

la imagen: con mayor o menor pérdida de calidad, de mayor o menor peso, con capacidad para soportar ajustes, etc. [8].

Actualmente, los formatos principales son:

- BMP (bitmap, mapa de bits): formato propiedad de Microsoft y sólo se utiliza en el sistema operativo Windows, para guardar sus imágenes.
- JPG: el más empleado para mandar fotografías y archivos gráficos por su bajo peso y su calidad decente.
- PNG: formato de mejor calidad (y mayor peso) que JPG, empleado en programas de edición y que soporta el uso de transparencias.
- TIFF: el formato de calidad óptima, empleado por ejemplo en imágenes con fines de maquetación
- GIF: el formato de imágenes digitales animadas.

Modo de color

Dicho lo más fácilmente posible, el modo de color de una imagen digital definirá la cantidad de colores que pueden poseer los píxeles que la componen. Los más conocidos son el modo RGB y el CMYK, pero hay más [9]. Algunos de ellos son:

- RGB: Red (rojo), Green (verde) y Blue (azul) conforman todo el resto de los colores. El modo RGB permite una gran variedad de colores en los píxeles de la imagen digital mediante la suma de estos tres colores básicos. Es el que se usa en pantallas y es el más usado actualmente.
- Modo CMYK: A diferencia del anterior, el modo C(Cian) M(Magenta) Y(Yellow-amarillo) y K(Black- negro) se usa para imprenta. Estos colores anteriormente mencionados son las tintas más usadas para imprimir, por lo que la mezcla de ellos será visible en la imagen, el resto de los colores, no.
- Monocromático: La imagen solo se forma de píxeles blancos o negros puros. No hay colores ni grises en ella.

- Escala de grises: Este modo de color de imagen digital permite hasta 256 tonos de grises, blanco y negro para configurar la imagen.

Profundidad de color

Otra de las características que configuran una imagen digital es la profundidad del color. En un mapa de bits o en una imagen digitalizada, los píxeles están coloreados uno a uno. Así, el archivo que contiene estos píxeles almacena los datos y la información de cada color que aparece. La cantidad de bits que se necesitan para guardar estos datos es lo que se conoce como profundidad del color [9].

Una imagen digital en color se genera con sus componentes R, G y B. Así pues, la imagen en color se compone de 3 "imágenes": la que contiene la parte roja, la verde y la azul. Cada una de ellas es una imagen en escala de grises, pero como va asociada a cada uno de los colores primarios, al visionarla en el ordenador la colorea adecuadamente. La suma de las 3, por síntesis aditiva, componen la imagen final. Cada píxel va definido por 3 Bytes: el rojo, el verde y el azul. Si cada uno de ellos tiene una gama de 256 tonos, en la imagen final habrá una gama de 16,777.216 colores posibles ($256 \times 256 \times 256$), que se suele abreviar como "16 millones de colores". Es lo que se llama una imagen con una profundidad de color de 24 bits (8 por cada color) [10].

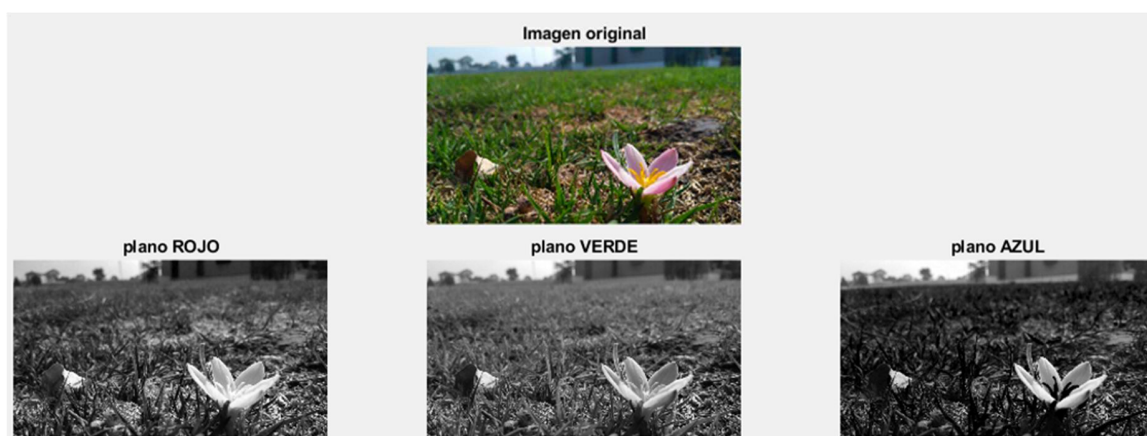


Ilustración 8. Ejemplo de una imagen digital con sus respectivos planos RGB

Histogramas de una imagen digital

¿Qué es un histograma?

Un histograma es similar en apariencia a un diagrama de barras, pero en lugar de comparar categorías o buscar tendencias a lo largo del tiempo, cada barra representa cómo se distribuyen los datos en una única categoría. Cada barra representa un rango continuo de datos o el número de frecuencias de un punto de datos específico [11].

¿Qué es un histograma de una imagen digital?

Los histogramas de imagen digital resumen visualmente la distribución de una variable numérica continua midiendo la frecuencia con la que determinados valores aparecen en la imagen. Los histogramas tienen dos ejes. El eje X, el horizontal, es una línea numérica que visualiza el rango de valores de píxeles de la imagen, muestra los diferentes tonos de la escala de grises: desde el negro más oscuro (izquierda) hasta el blanco más puro (derecha). Por otro lado, el eje Y, el vertical, representa la cantidad de píxeles que contiene cada tono de la imagen mostrado en el eje horizontal [12] [13].

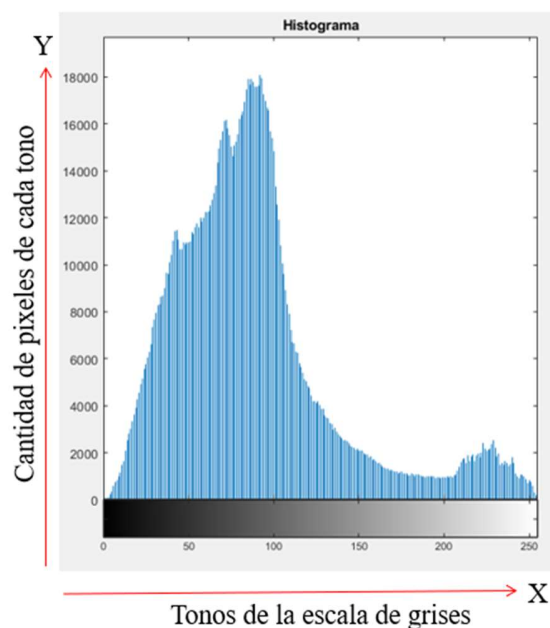


Ilustración 9. Histograma de una imagen

Estos gráficos están presentes tanto en las cámaras digitales como en los programas de edición de imagen modernos pues permiten corregir y mejorar sus propiedades de las imágenes digitales. El histograma resulta muy útil para determinar si el contraste o la exposición de una imagen digital son los adecuados.

Consideraciones importantes de los histogramas:

Cada imagen tiene su propio histograma. Pero como regla general se considera que una imagen tiene un buen contraste si su histograma se extiende ocupando casi todo el rango de tonos [14].

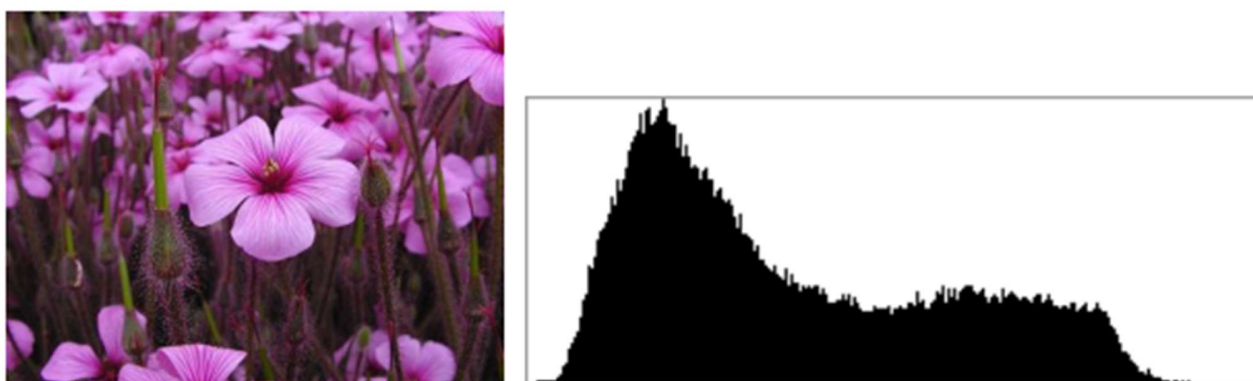


Ilustración 10. Histograma de una imagen con una buena distribución de tonos

Una imagen oscura por ejemplo una imagen del cielo de noche, su histograma está fuertemente desplazado al lado izquierdo (zona de tonos oscuros) y es evidencia de que no hay apenas ninguna zona de la imagen muy clara.

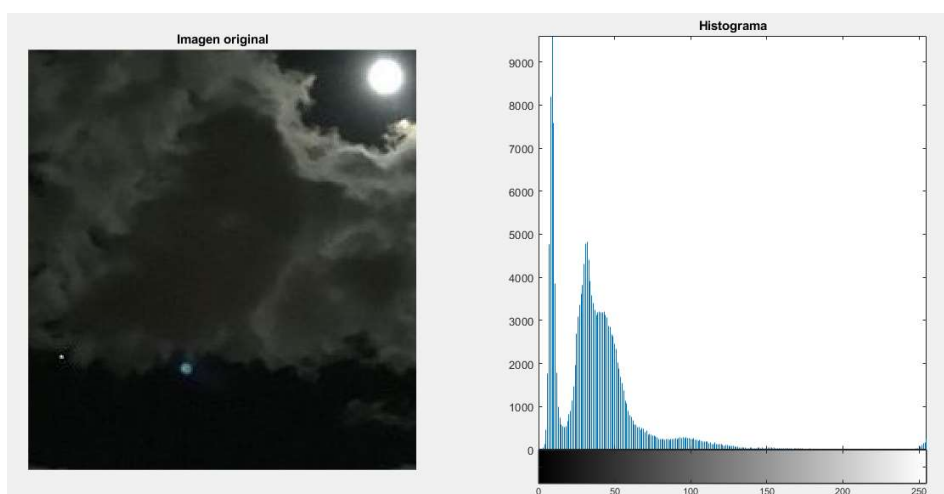


Ilustración 11. Histograma de una imagen oscura

En una imagen clara se muestra su histograma desplazado hacia la derecha ya que predominan los colores claros esto significa que la imagen tiene mucho brillo o que es una imagen demasiado sobre expuesta.

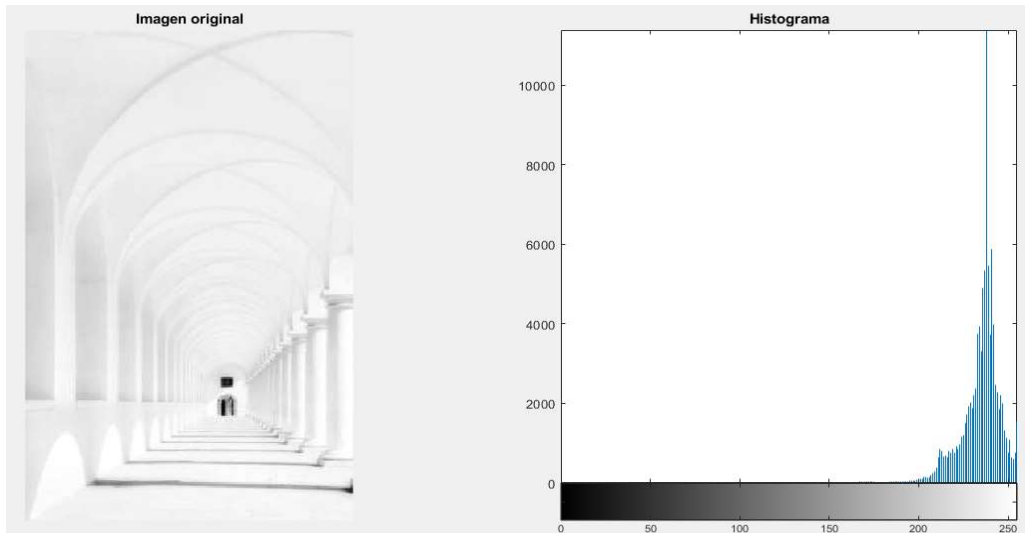


Ilustración 12 Histograma de una imagen clara

Si se entiende por contraste la diferencia de brillo entre las zonas más claras y oscuras en una imagen, el siguiente ejemplo es una imagen de bajo contraste. No hay nada completamente oscuro ni tampoco muy claro. Esto se traduce en un histograma estrecho

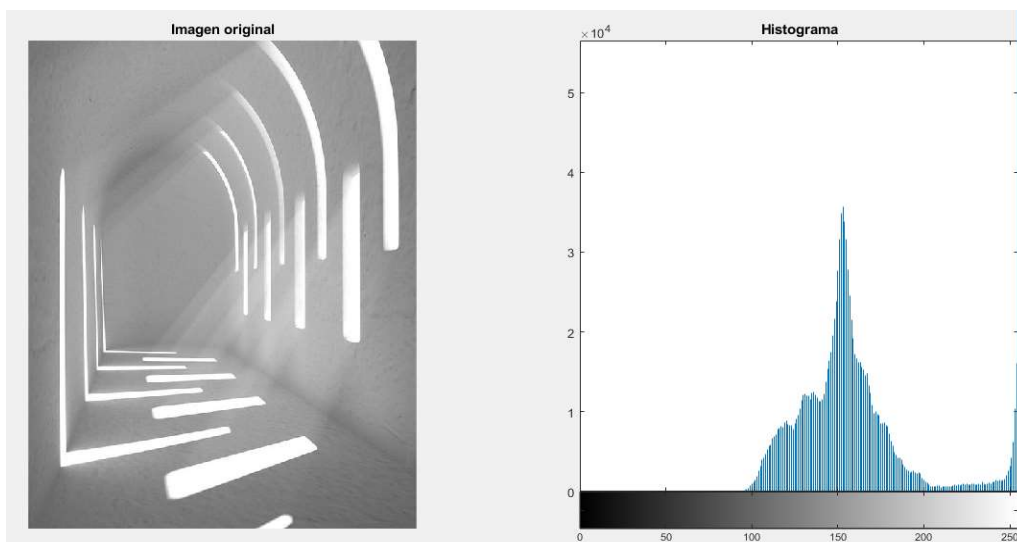


Ilustración 13 Histograma de una imagen de bajo contraste

Con el histograma podemos darnos cuenta de cuál es la distribución de luz en la imagen o si hay ausencia de ella, es una herramienta que nos ayuda en el procesamiento digital de imágenes ya que nos da la información suficiente para poder editar alguna imagen de interés y resaltar lo que importa analizar en este proyecto fue necesario aprender sobre los histogramas ya que toda operación aplicada a la imagen se veía reflejada en el histograma , el cual nos iba indicando si el procesamiento era el indicado.

Técnicas básicas de procesamiento digital de imágenes con Matlab

Como se ha mencionado el procesado de imágenes digitales implica la utilización de algoritmos y herramientas que proporcionan un entorno exhaustivo para el análisis de datos, la visualización y el desarrollo de algoritmos.

En este proyecto fue utilizado el *software Matlab* como herramienta de procesado de imágenes de tal modo que nos ayudó a analizar las partes de la imagen que eran de interés para ello Matlab tiene muchas funciones para este fin desde leer una imagen hasta el procesamiento para una nube de puntos.

Para esto es necesario leer una imagen y esta debe encontrarse en la carpeta de trabajo de Matlab. Los formatos de imagen soportados por Matlab son: TIFF, JPG, JPEG, GIF, BMP, PNG, XWD.

El formato elegido para el procesado depende de las necesidades del programador para editar las imágenes correctamente para eso se debe adentrar a sus características de cada formato y elegir cual es la que más conviene.

```
a_RGB=imread("FLORITAPIZACO.JPG");
```

Ilustración 14. Comando para leer una imagen

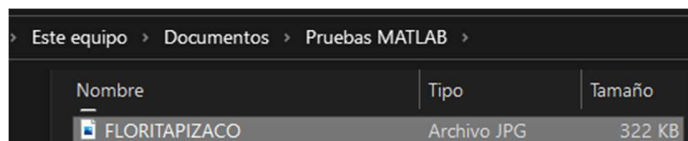


Ilustración 15. Dirección de la carpeta de trabajo de Matlab

Una de las partes importantes dentro del software de *Matlab* es la ventana *Workspace* en donde se puede observar las variables y sus características en el algoritmo, así como los cambios que se le van haciendo a dichas variables, es una herramienta primordial cuando estamos haciendo el procesado de la imagen. Por ejemplo, en la siguiente ilustración se puede observar una variable con el nombre *a_RGB* de tres planos (x3) incluso se puede observar las dimensiones de tamaño de la imagen.

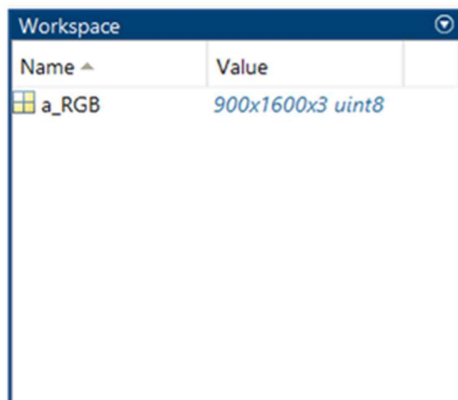


Ilustración 16 Ventana Workspace de Matlab

Anteriormente se mencionó que una imagen digital se compone de tres planos, rojo verde y azul mejor conocidos como planos RGB, como sabemos son tres matrices bidimensionales cada una, en *Matlab* podemos desplegar cada plano con un comando en específico, a cada plano se le pueden hacer los ajustes necesarios para resaltar lo que más nos interese de cada uno de ellos y obtener un mejor procesado de la imagen.

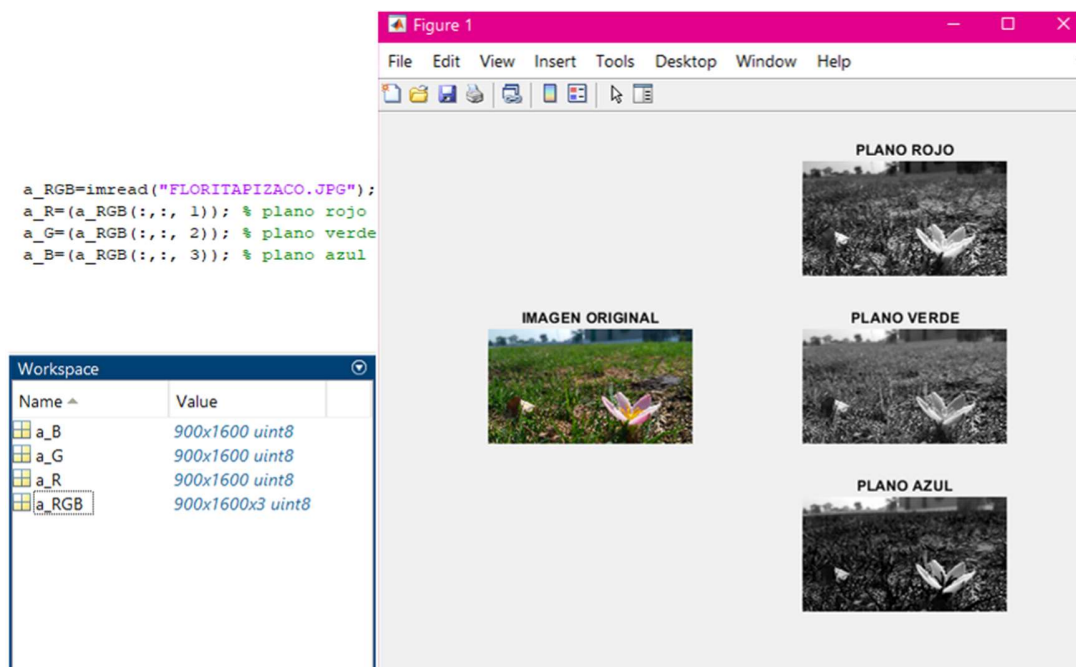
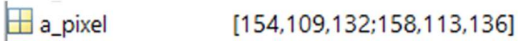
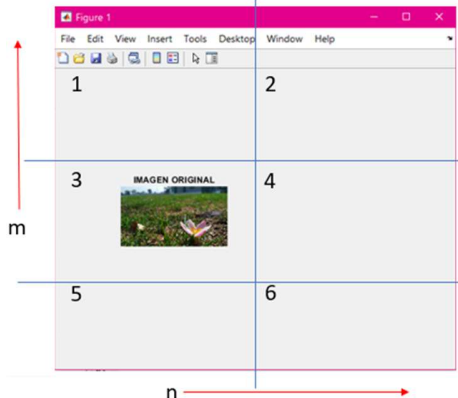


Ilustración 17. Ejemplo del despliegue de los planos de una imagen en Matlab

En la siguiente tabla se muestran algunos comandos básicos de Matlab importantes para el procesamiento de una imagen.

Tabla 3. Comandos básicos para el Procesamiento digital de imágenes.

Comando	Descripción
<code>imshow (a_RGB)</code>	Este comando permite visualizar una imagen en Matlab abriéndose en automático una ventana con un nombre por default como “ <i>Figure 1</i> ”. Solo muestra imágenes del tipo uint8.
<code>a_size= size(a_RGB);</code>	Este comando nos proporciona la información de las dimensiones que tiene la imagen, en la ventana <i>Workspace</i> se aprecia esta información: 
<code>imwrite(a_RGB, 'C:\Users\Imagenes\FLO2.jpg');</code>	Este comando sirve para guardar una imagen con un nuevo nombre en una nueva dirección y con un tipo de formato igual o diferente al original.
<code>a_pixel= impixel(a_RGB);</code>	Este comando abre una ventana donde se visualiza la imagen, pero al pasar el cursor sobre ella se observa una cruz que permite seleccionar un píxel, se puede obtener con esto la información de un solo píxel.  

<pre>>> whos a_RGB</pre>	Nos proporciona la información más detallada de una variable <table><thead><tr><th>Name</th><th>Size</th><th>Bytes</th><th>Class</th><th>Attributes</th></tr></thead><tbody><tr><td>a_RGB</td><td>900x1600x3</td><td>4320000</td><td>uint8</td><td></td></tr></tbody></table>	Name	Size	Bytes	Class	Attributes	a_RGB	900x1600x3	4320000	uint8	
Name	Size	Bytes	Class	Attributes							
a_RGB	900x1600x3	4320000	uint8								
<pre>imcrop (a_RGB);</pre>	Permite recortar una imagen: 										
<pre>subplot (3,2,3); imshow(a_RGB); title ('IMAGEN ORIGINAL');</pre> subplot (m,n,p); m = filas n = columnas p =posición de la imagen	Generalmente se usa para mostrar varias imágenes en una sola ventana: 										

Tipo de datos

Las matemáticas de MATLAB admiten cuatro clases numéricas diferentes para la visualización de imágenes [15]:

- punto flotante de doble precisión (double, 8 bytes por elemento)
- número entero sin signo de 16 bits (uint16, 2 bytes por elemento)
- número entero sin signo de 8 bits (uint8, 1 byte por elemento)
- logical los valores son 0 ó 1 (1 bit por elemento)

Cuando realizamos transformaciones matemáticas de imágenes, a menudo necesitamos que la imagen sea de tipo double. Si la imagen es de tipo double, los datos que la constituyen son del tipo float y se encuentran en el intervalo [0,1]. Pero cuando la leemos y almacenamos ahorramos espacio usando codificación entera.

Operaciones sobre matrices

Desde este punto tenemos en cuenta que una imagen es equivalente a una matriz y como cualquier matriz se puede someter a cálculos matemáticos donde cada valor en su interior cambiará dependiendo de la función matemática que se le aplique.

MATLAB puede operar con matrices por medio de operadores y por medio de funciones. Los operadores matriciales de MATLAB son los siguientes:

- + adición o suma
- - sustracción o resta
- * multiplicación
- ' traspuesta
- ^ potenciación
- \ división-izquierda
- / división-derecha
- .* producto elemento a elemento
- ./ división elemento a elemento
- .^ elevar a una potencia elemento a elemento

Filtrado

Existen una gran variedad de procedimientos que permiten, a partir de una imagen, obtener otra modificada (técnicas de filtrado). Se trata de métodos con los que se puede resaltar o suprimir, de forma selectiva, información contenida en una imagen, para destacar algunos elementos de ella, o también para ocultar valores anómalos.

El filtrado de imágenes consiste en la modificación de las matrices correspondientes a la imagen digitalizada mediante el procedimiento que se halla diseñado. Uno de los procedimientos más utilizado es la llamada convolución de matrices [16].

La operación matemática en que consiste la convolución es simplemente una suma “pesada” de píxeles vecinos del píxel fuente. Los pesos son determinados por una matriz llamada máscara de convolución, que determina los coeficientes a aplicar sobre los puntos de una determinada área.

Las dimensiones de la matriz son normalmente impares e iguales, de forma que se pueda determinar un centro de una matriz cuadrada. La posición del valor central se corresponde con la posición del píxel de salida. La suma de los pesos en la máscara de convolución afecta a la intensidad general de la imagen resultante [17].

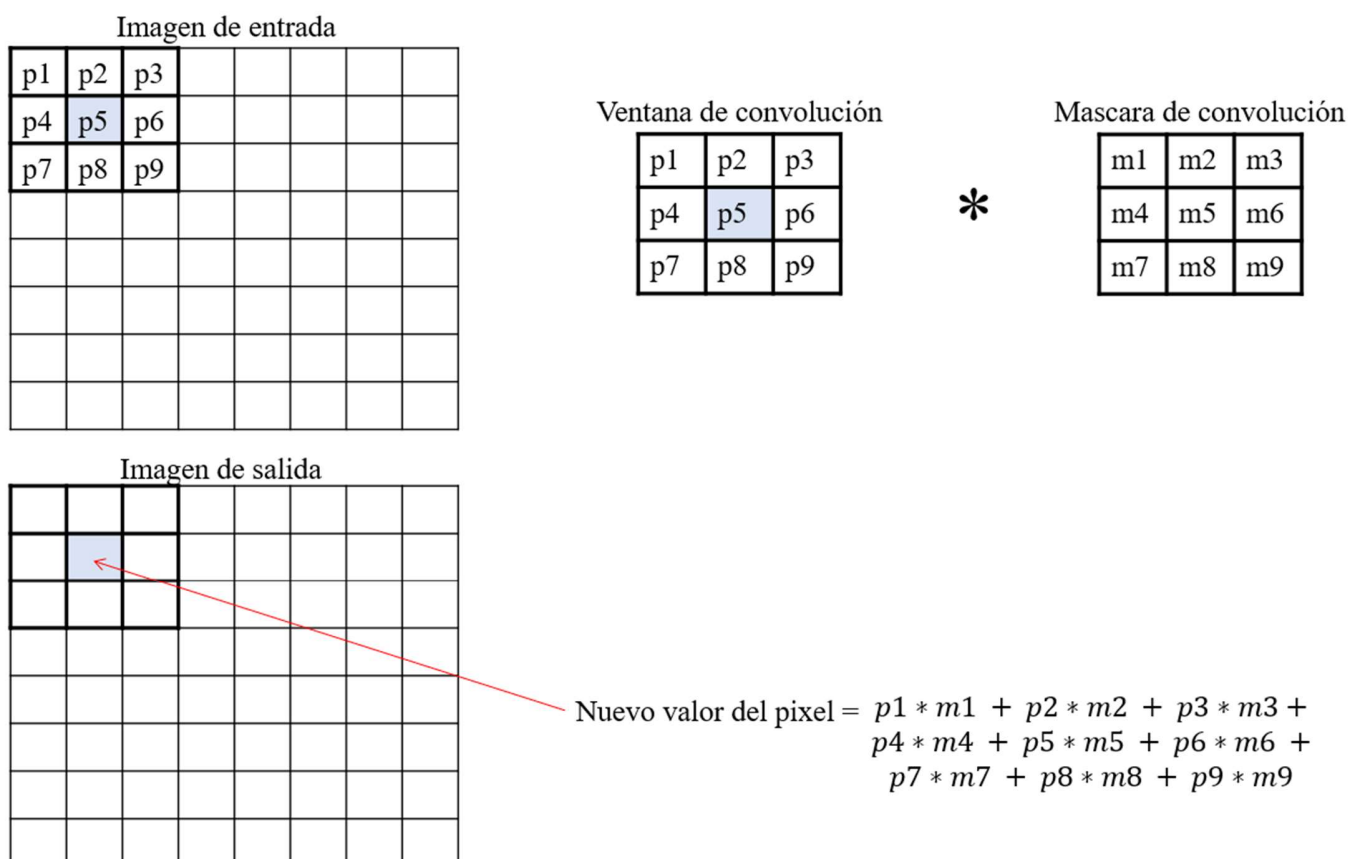


Ilustración 18. Proceso de convolución

La siguiente ilustración muestra algunas de las matrices máscaras más usadas:

Enfoque	Desenfoque	Realce de bordes	Repujado
$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} -2 & -1 & 0 \\ -1 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix}$
Detección de bordes	Filtro de tipo Sobel	Filtro de tipo Sharpen	
$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & -2 & 1 \\ -2 & 5 & -2 \\ 1 & -2 & 1 \end{bmatrix}$	
Filtro Norte	Filtro Este	Filtro de tipo Gauss	
$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & -1 & -1 \end{bmatrix}$	$\begin{bmatrix} -1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 3 & 1 & 1 \\ 2 & 7 & 11 & 7 & 2 \\ 3 & 11 & 17 & 11 & 3 \\ 2 & 7 & 11 & 7 & 2 \\ 1 & 2 & 3 & 2 & 1 \end{bmatrix}$	

Ilustración 19 Máscaras más usadas.

Filtros espaciales no lineales

Los filtros espaciales no lineales también operan sobre entornos. Ejemplos de filtros no lineales habituales son los filtros mínimos, máximo y de mediana que son conocidos como filtros de rango. El filtro de mediana tiene un efecto de difuminado de la imagen, y permite realizar una eliminación de ruido, el filtro de máximo se emplea para buscar los puntos más brillantes de una imagen produciendo un efecto de erosión, y el filtro de mínimo se emplea con el objetivo contrario, buscar los puntos más oscuros de una imagen produciendo un efecto de dilatación.

Filtros espaciales

- Filtros de suavizado (Smoothing) para difuminar la imagen. El suavizado de imágenes se utiliza normalmente bajo dos supuestos: dar a una imagen un difuminado o efecto especial y para la eliminación de ruido. El suavizado o filtrado espacial paso bajo borra los detalles más finos de una imagen, atenúa las altas frecuencias. El ruido en general es una función impulsiva sumada a la imagen,

estos filtros tienden a eliminar el ruido. Sin embargo los bordes que también son abruptos tienden a ponerse borrosos con estos filtros.

- Filtros de realce (Sharpening) para eliminar zonas borrosas. El realce de una imagen aumentará su contraste atenuando las componentes de bajas de frecuencias y la imagen aumentara su nitidez.
- Filtros diferenciales que se componen de varios tipos de máscaras (Laplaciano, Prewitt, Sobel, etc.), y se utilizan para la detección de bordes. El proceso de detección de bordes se basa en realizar un incremento del contraste en las zonas donde hay una mayor diferencia entre las intensidades, y en una reducción de éste donde no tenemos variación de intensidad.

A una sola imagen se le pueden aplicar los filtros que sean necesarios para obtener la información deseada en este caso se aplicaron los suficientes para obtener una imagen del tipo binaria para formar la nube de puntos.

Nube de puntos

Una nube de puntos es un conjunto de vértices en un sistema de coordenadas tridimensional. Estos vértices se identifican habitualmente como coordenadas X, Y, y Z y son representaciones de la superficie externa de un objeto [18].

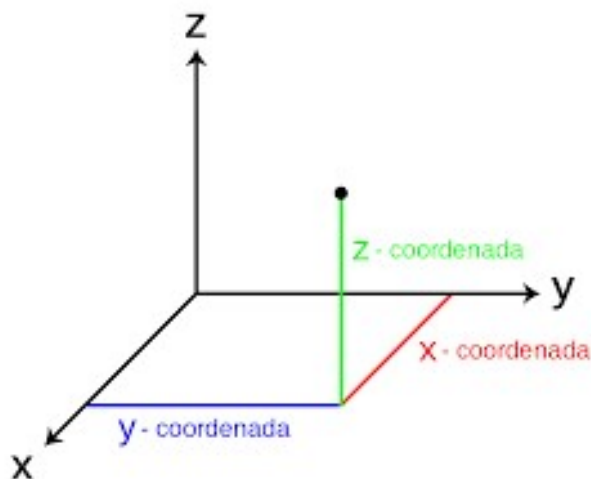


Ilustración 20. Coordenadas tridimensionales

Una nube de puntos 3D es el primer producto resultante del escaneo láser o la fotogrametría digital y el paso previo a la creación de modelos 3D. Se compone por millones de puntos posicionados tridimensionalmente en el espacio, formando con exactitud milimétrica una entidad física y representando su superficie externa. La nube de puntos 3D contiene una amplia información métrica sobre las superficies escaneadas, así como la relativa a su color y reflectividad del material [19].

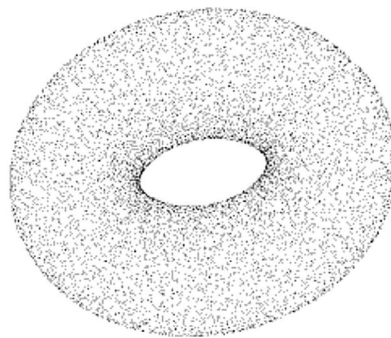


Ilustración 21. Ejemplo de una nube de puntos de un toro

Las aplicaciones de las nubes de puntos son muy variadas, desde la obtención de gemelo digital, documentación as built, levantamientos topográficos, levantamiento de infraestructuras de gran dimensionado, inspección y control de instalaciones industriales y en la detección de interferencias. Además, la nube de puntos nos ayuda a crear y obtener recreaciones en 3D de espacios y volúmenes arquitectónicos [20].

Las principales funciones de una nube de puntos aplicadas a arquitectura son:

- Representación digitalizada de un entorno o edificio
- Documentación para la preservación y restauración de bienes de interés histórico cultural
- Detección de deformaciones en edificios e instalaciones
- Levantamientos topográficos
- Inspección y Control de instalaciones
- Modelado de maquetas virtuales en Autodesk Revit, Autocad Plant 3d, Inventor, Solidworks, Sketchup y muchos otros formatos.

Hay varias técnicas para convertir una nube de puntos en una superficie tridimensional. Algunos procedimientos como la triangulación de Delaunay o las formas alfa construyen una red de triángulos a partir de los vértices de la nube de puntos, mientras que otros convierten la nube de puntos en un volumen de vóxel y reconstruyen la superficie implícita mediante un algoritmo de marching cubes.

Malla

Reconstruir un objeto tridimensional, se refiere a obtener su representación en la memoria de un computador, manteniendo sus características de volumen y forma, partiendo de un conjunto discreto de puntos muestreados a partir de su superficie [21].

Las mallas 3D o mallas poligonales son representaciones tridimensionales de una superficie, que se basan en una colección de vértices unidos por aristas. Un vértice se define como un punto en el espacio tridimensional y que puede almacenar cierta información, una arista une 2 vértices y es un punto de unión entre 2 triángulos, 3 vértices diferentes unidos por aristas forman una faceta/triángulos, que es la unidad básica de representación de los polígonos tridimensionales. Algunos sistemas utilizan polígonos creados por 4 vértices, no obstante internamente se representan como 2 triángulos. Un conjunto de caras o polígonos unidos entre sí, forman una superficie [22].

Simplificación de la Malla

Las mallas pueden ser construidas desde nubes de puntos, datos de rango y de extracción de isosuperficies. La forma objetivo, puede casi siempre ser representada muy bien usando unos polígonos distantes entre sí. Muchos métodos de simplificación de superficies han sido desarrollados para derivar tales aproximaciones simples automáticamente. La mayoría de los algoritmos de simplificación operan aplicando una primitiva de mortandad, tal como la contracción de bordes, o levantamiento de vértices. Estos métodos se enfocan en el adaptamiento de la conectividad de la malla. Un énfasis similar en dirección al remuestreo de la superficie puede ser visto en métodos que construyen una descomposición global de la superficie en parches, los cuales son poligonizados para producir una malla final.



Ilustración 22. Malla aplicada a una esfera

Remallado semi-regular

El objetivo de los trabajos en esta área es reducir la complejidad de la malla, mientras se mantiene la fidelidad geométrica. Los esquemas de remallado semiregular constan de un proceso de dos pasos. Primero ellos descomponen la superficie en un conjunto de triángulos o parches cuadrilaterales. Segundo, ellos remuestrean cada parche por subdivisión recursiva. De forma similar, los sistemas MAPS construyen un dominio de base triangular por simplificación. Ambas son diseñadas para usarlas como la etapa inicial en análisis de subdivisión de superficies.

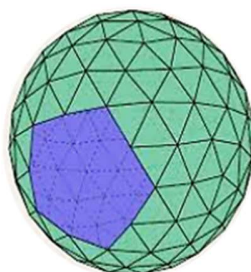


Ilustración 23. Remallado aplicado a una esfera

Las mallas poligonales se pueden clasificar según el método que se utilice para almacenar su información. Aunque se puede construir una malla de manera manual especificando vértices y caras, es mucho más eficiente utilizar el conjunto de herramientas software para gráficos 3d que existen actualmente.

Herramientas software de enmallado

Estos programas hacen uso de la subdivisión y la extrusión. La subdivisión como su propio nombre indica, fracciona los triángulos en otros más pequeños añadiendo vértices y aristas. Por otro lado, la extrusión se aplica a un triángulo o a un conjunto de ellos. Siguiendo el contorno de estas facetas se crea una nueva con

la misma forma, que conecta cada uno de sus vértices con una arista creando un volumen. Así pues la extrusión de un cuadrado sería un prisma [22].

Un software que nos fue de ayuda para aplicar enmallados a las nubes de puntos es *Meshlab*, este software es de código abierto para procesar y editar mallas triangulares 3D. Proporciona un conjunto de herramientas para editar, limpiar, reparar, inspeccionar, renderizar, texturizar y convertir mallas. Ofrece funciones para procesar datos sin procesar producidos por herramientas/dispositivos de digitalización 3D y para preparar modelos para impresión 3D [23].

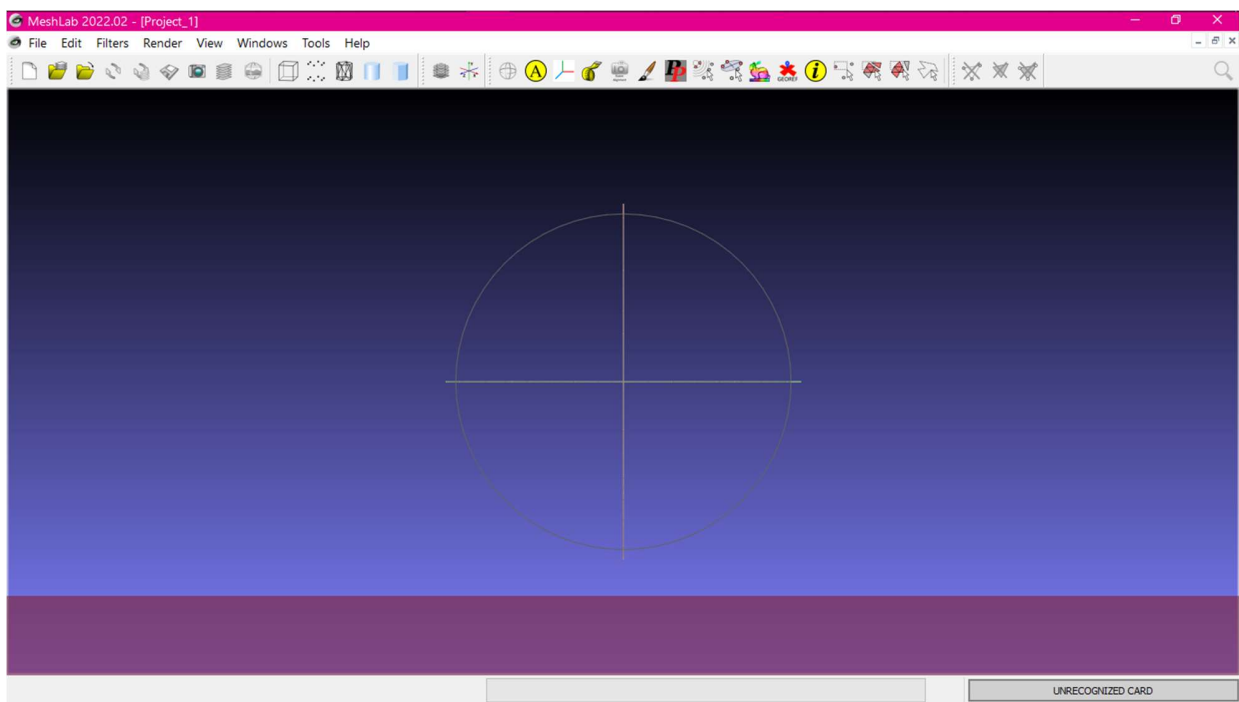


Ilustración 24. Captura de la pantalla principal de Meshlab

Características principales

Tabla 4. Características de Meshlab

Adquisición 3D	Descripción
Alineación	La fase de alineación de datos 3D (también conocida como registro) es un paso fundamental en el proceso de procesamiento de datos escaneados en 3D. MeshLab implementa un paso de alineación uno

	a uno de ICP ajustado, seguido de un paso de distribución de errores de ajuste de paquete global. La alineación se puede realizar en mallas y nubes de puntos provenientes de varias fuentes, incluidos escáneres activos (tanto de corto como de largo alcance) y herramientas 3D a partir de imágenes.
Reconstrucción	El proceso de transformación de adquisiciones independientes, o nubes de puntos, en una malla triangulada de una sola superficie se puede realizar con diferentes enfoques algorítmicos. MeshLab proporciona varias soluciones para reconstruir la forma de un objeto, que van desde volumétricas (Marching Cube) hasta superficies implícitas (Screened Poisson).
Mapeo de color y texturizado	La información del color puede ser tan importante como la geometría, pero varias tecnologías de adquisición no brindan datos de apariencia precisos. MeshLab contiene una canalización para la alineación y proyección de información de color (a partir de un conjunto de imágenes sin calibrar) en un modelo 3D. Se proporcionan varios métodos automáticos y asistidos para obtener una codificación de color de alta calidad, tanto con mapeo por vértice como de textura.
Simplificación, Refinamiento y Remallado	Una necesidad común al procesar un modelo 3D es reducir su complejidad geométrica, creando una geometría con la misma forma pero con menos triángulos (o puntos). MeshLab ofrece diferentes formas de simplificar (diezmar) superficies trianguladas, capaces de preservar los detalles geométricos y el mapeo de texturas, o de reducir selectivamente el número de puntos en una nube de puntos. En otros casos, es posible que el usuario desee aumentar la cantidad de triángulos (o puntos): MeshLab también proporciona diferentes esquemas de subdivisión, filtros de malla y remuestreo

	para aumentar la complejidad geométrica de los modelos 3D o para optimizar la distribución de puntos y la calidad de la triangulación.
Compensación, Vaciado, Cierre	Además de poder exportar a STL (uno de los formatos más comunes para la impresión en 3D), MeshLab se puede usar para preparar modelos en 3D para imprimir mediante la creación de capas internas, el remuestreo/remallado del modelo en 3D para facilitar el corte, el cierre de pequeños orificios para obtener mallas estancas, y aplanando la zona inferior para tener una mejor adherencia a la plataforma.

Herramientas para impresión 3D

Con la impresión 3D es posible ir directamente desde un diseño digital hasta una pieza final. La realización de prototipos 3D nos permiten analizar virtualmente la geometría de la pieza escaneada y de ser necesario cambiar características para la mejora de la impresión.

El STL es un formato de archivo informático de diseño asistido por computadora (CAD) que define la geometría de objetos 3D, sin incluir información como color, texturas o propiedades físicas que sí incluyen otros formatos CAD.

Es el formato estándar para las tecnologías de fabricación aditiva. Utiliza una malla de triángulos cerrada para definir la forma de un objeto. Cuantos más pequeños son estos triángulos, mayor será el número de ellos necesario para cerrar la superficie del objeto, y mayor el tamaño del fichero final. Por tanto, el tamaño de los triángulos está directamente proporcionado con el peso del fichero, por lo que es aconsejable llegar a una solución de compromiso entre la resolución y el peso del fichero. La mayoría de software de modelado 3D permite el guardado del archivo o exportar en esta extensión STL [24].

En el presente trabajo se obtuvo un formato estándar de impresión STL de nuestro prototipo modelado, que será optimizado para su impresión en una impresora 3D marca Ender, para que permita resultados aceptables en la impresión 3D final.

El software que se eligió para darle los parámetros adecuados de impresión al modelado final es el software *Ultimaker Cura*, es una aplicación diseñada para impresoras 3D, en la que se pueden modificar los parámetros

de impresión de código abierto, una de las ventajas de Cura es su facilidad de uso, soporte para diferentes formatos de archivo y compatibilidad con muchas impresoras 3D. Los formatos admitidos son STL, OBJ, X3D y 3MF. Aunque Cura es parte del ecosistema Ultimaker, las impresoras de otros fabricantes también pueden usarlo.

ura tiene una interfaz simple que permite elegir entre dos modos desde el principio: el modo recomendado y el modo personalizado. El modo recomendado es particularmente adecuado para obtener el resultado de la impresión en un mínimo de clics sin tener que hacer cambios manuales en los parámetros. El usuario puede definir parámetros básicos como la altura de las diferentes capas, la adhesión y las estructuras de soporte.

El modo personalizado permite al usuario elegir entre más de 400 opciones de configuración diferentes. Además de los parámetros básicos, como agregar estructuras de soporte o la cantidad de relleno requerida, se pueden agregar fácilmente otras opciones. El usuario también puede tener una idea de cómo Cura cortará el modelo usando el modo de vista previa.

También es posible seleccionar el filamento utilizado antes de imprimir, de modo que los parámetros se modifiquen automáticamente [25].

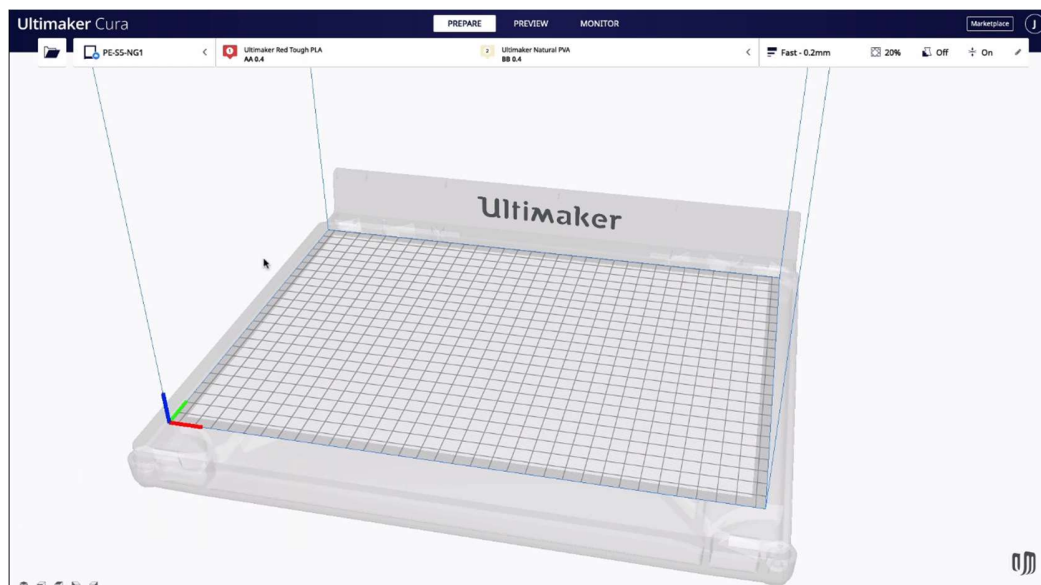


Ilustración 25. Pantalla principal del software Ultimaker Cura.

Desarrollo

Se inició por desarrollar una propuesta de una nueva estructura para el Escáner 3D basándonos en el primer modelo ya establecido

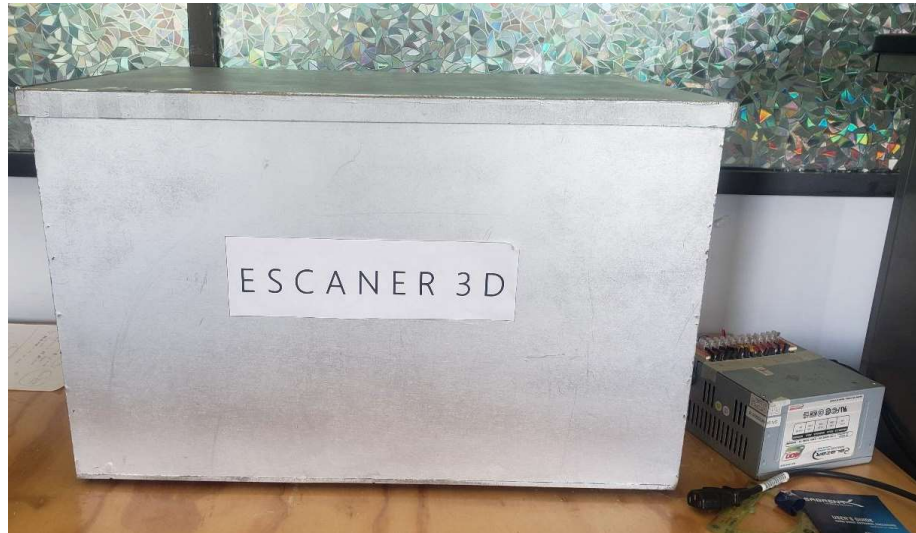


Ilustración 26. Sistema de Escaneo 3

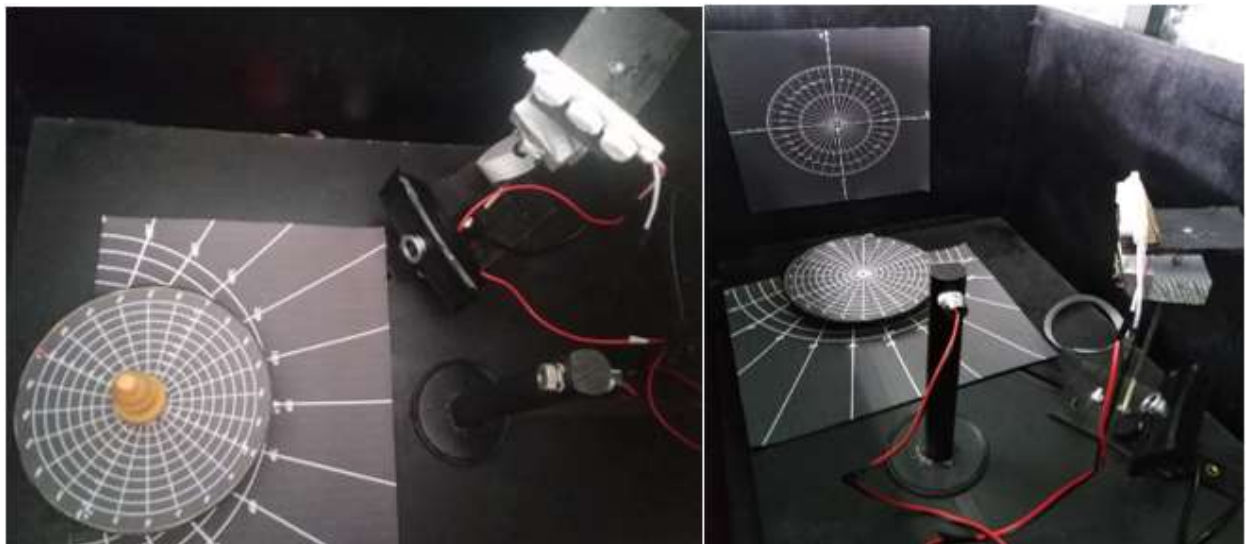


Ilustración 26. Primer diseño de Cámara Oscura

En el primer diseño como se logra apreciar está realizado únicamente con MDF o como comúnmente le llaman madera comprimida

Al estar fabricada de esta manera notamos que la estructura no es muy estable para el diseño interno así que nos dimos a la tarea de crear y rediseñar la estructura a partir de lo que ya tenemos.

Se comenzó creando la estructura exterior y parte de la estructura interior donde se colocarán soportes para un mecanismo, esto pensado con una propuesta para una puerta corrediza automatizada y una base móvil con la cual se podría ingresar y expulsar la pieza a digitalizar.

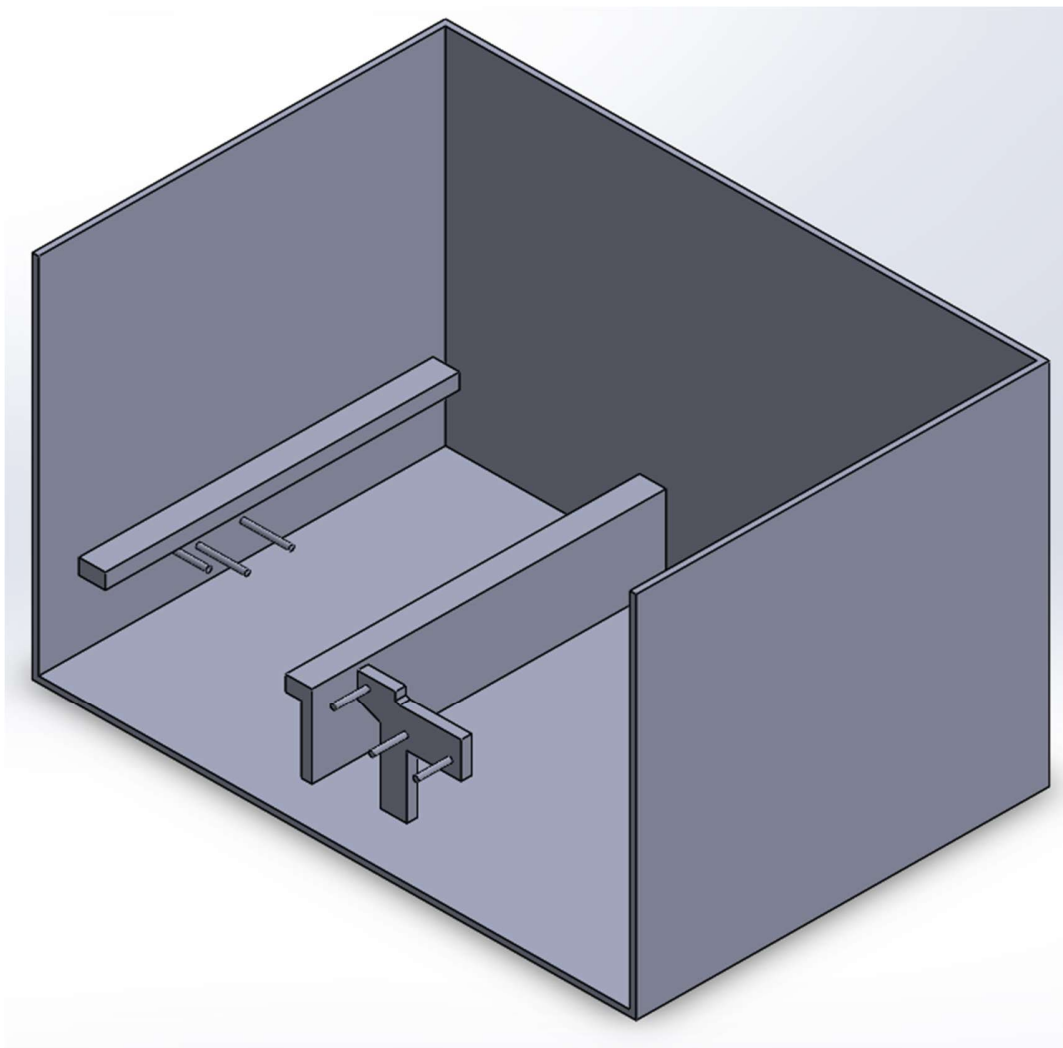


Ilustración 27. Estructura del Sistema Vista isométrica

De igual manera realizando la adaptación para la apertura y expulsión de la pieza en la tapa frontal.

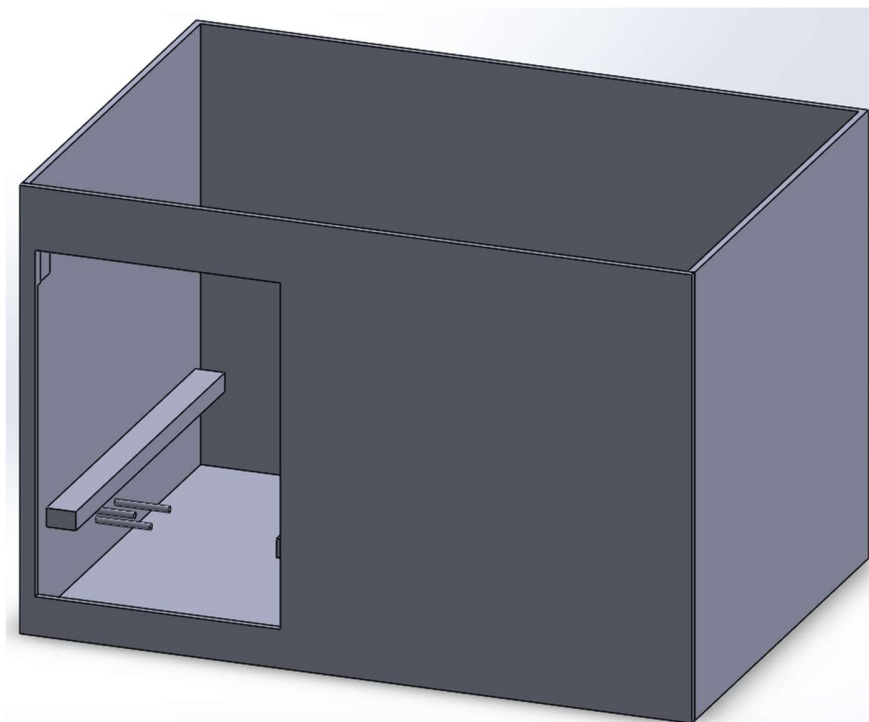


Ilustración 28. Estructura del Sistema Vista Dimétrica

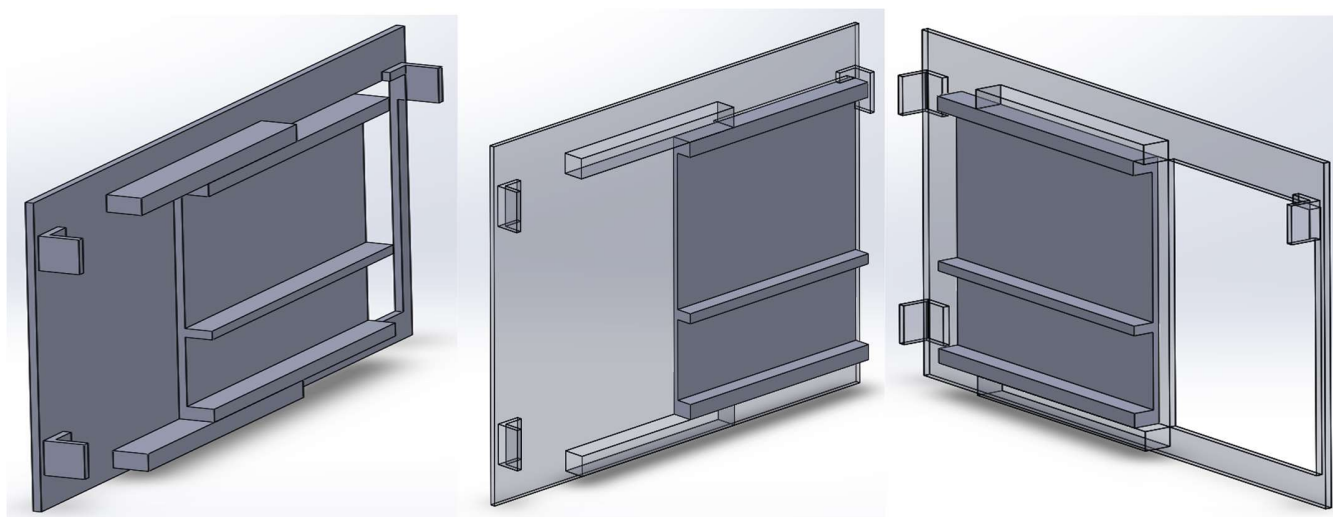


Ilustración 29. Tapa Frontal y Compuerta del Sistema

Posteriormente se realizó el ajuste y cálculo de dimensiones para la base móvil que expulsará e introducirá el objeto quedando de la siguiente manera.

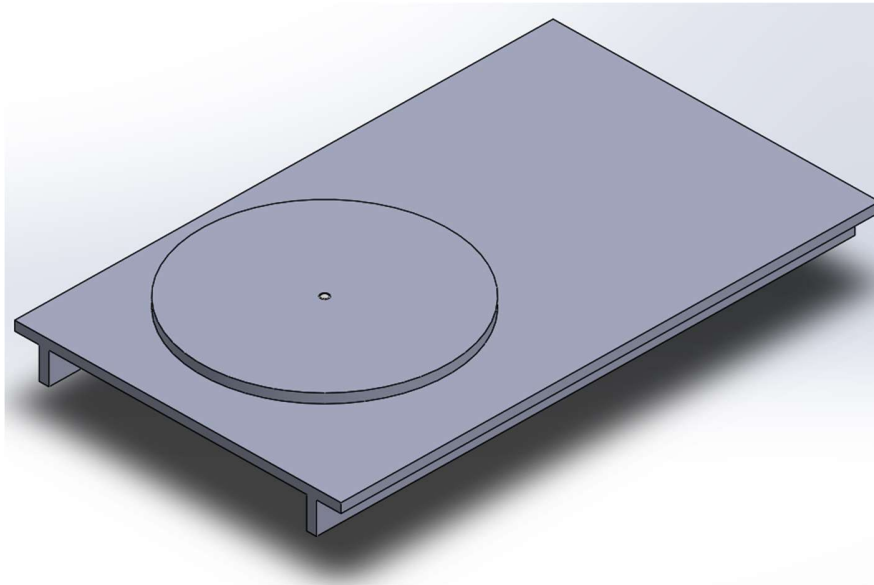


Ilustración 30.26 Charola (Base Giratoria)

Ejemplo:

Medición de piezas:

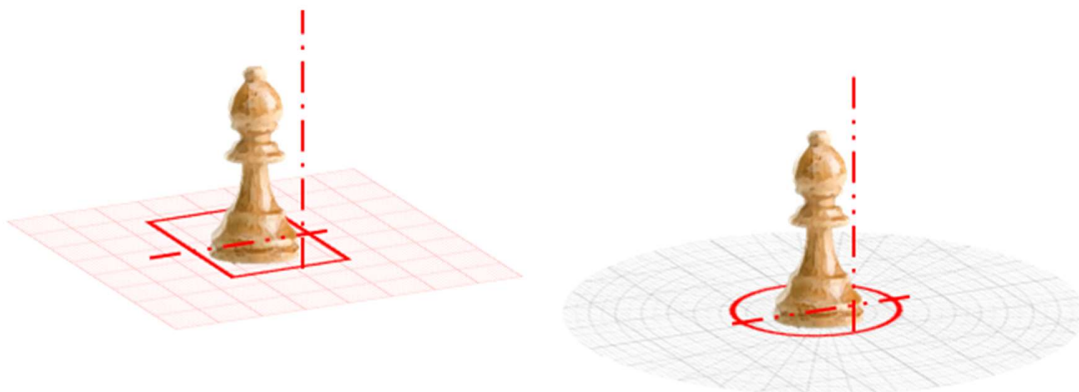


Ilustración. 31. Medición de piezas según su morfología

Colocación de la guía y centrado correcto de la pieza a escanear

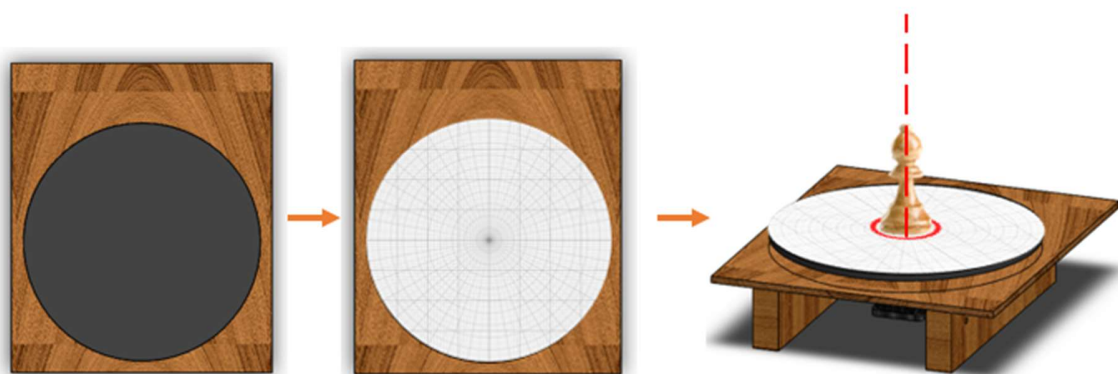


Ilustración 32. Centrado de pieza

También dentro de la misma problemática se observó que es muy poco práctico como se introduce dentro del escáner la figura/pieza, el abrir y cerrar de la tapa superior hace que se mueva el objeto, aunque éste ya se encuentre en su respectivo lugar por lo que una solución es que el plato salga del escáner y se vuelva a meter parecido o igual al sistema de una bandeja de CD.



Ilustración 33. Bandeja de un disco CD

Lo que se dio a la tarea de modificar la base quedando de la siguiente manera como una propuesta para el sistema de apertura/cierre e ingreso de la pieza.

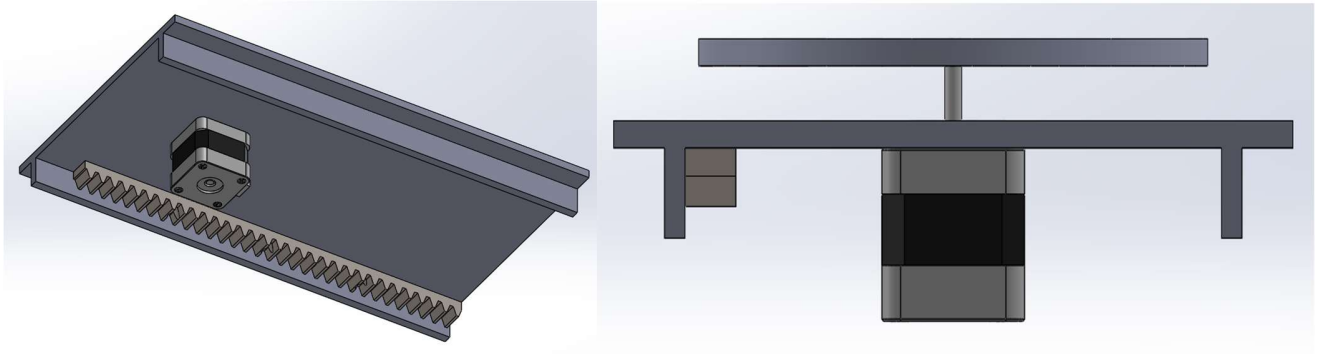


Ilustración 34. Sistema de Introducción y Expulsión

Esta base móvil cuenta con una cremallera dentada pensada para la automatización y movimiento de la misma cuál se conecta mediante un sistema de engranes conectados a un motor nema 16.

Pensando en reutilizar el material con el que ya contábamos se podía hacer la comparativa en algunas piezas como las siguientes

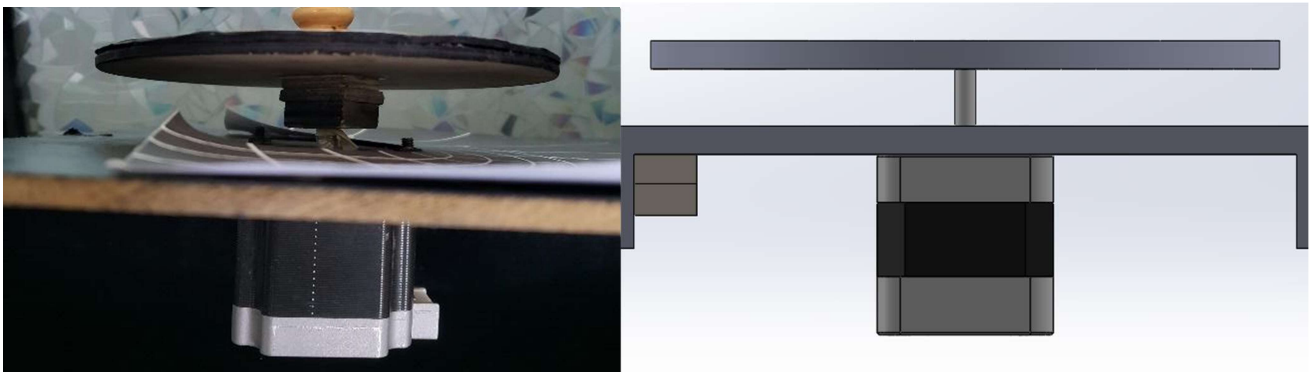


Ilustración 35. Base Giratoria de Escaneo

Para el diseño del sistema se utilizó el mismo sistema para las 2 piezas móviles tanto la puerta de apertura y como la bandeja móvil el para introducir las piezas.

El sistema consta de un engranaje recto con un diámetro de 60 mm y un número total de 18 dientes el cuál está unido a un engranes reductor que cuenta con un radio de 18 mm y un total de 10 dientes el cual nos ayuda y permite aumentar el giro para la apertura de la puerta o expulsión de la charola

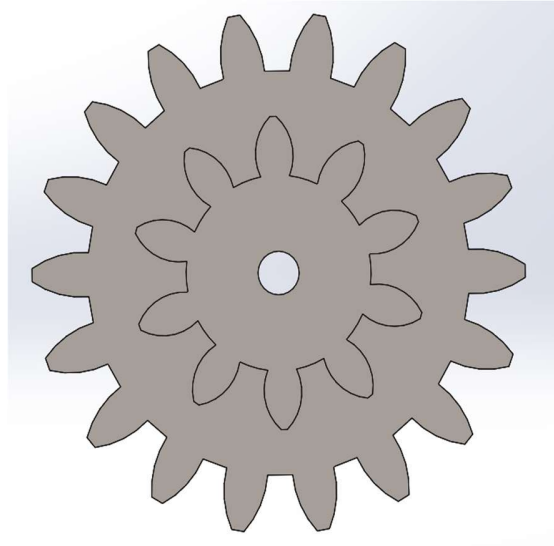


Ilustración 36. Sistema de Engranaje Reductor

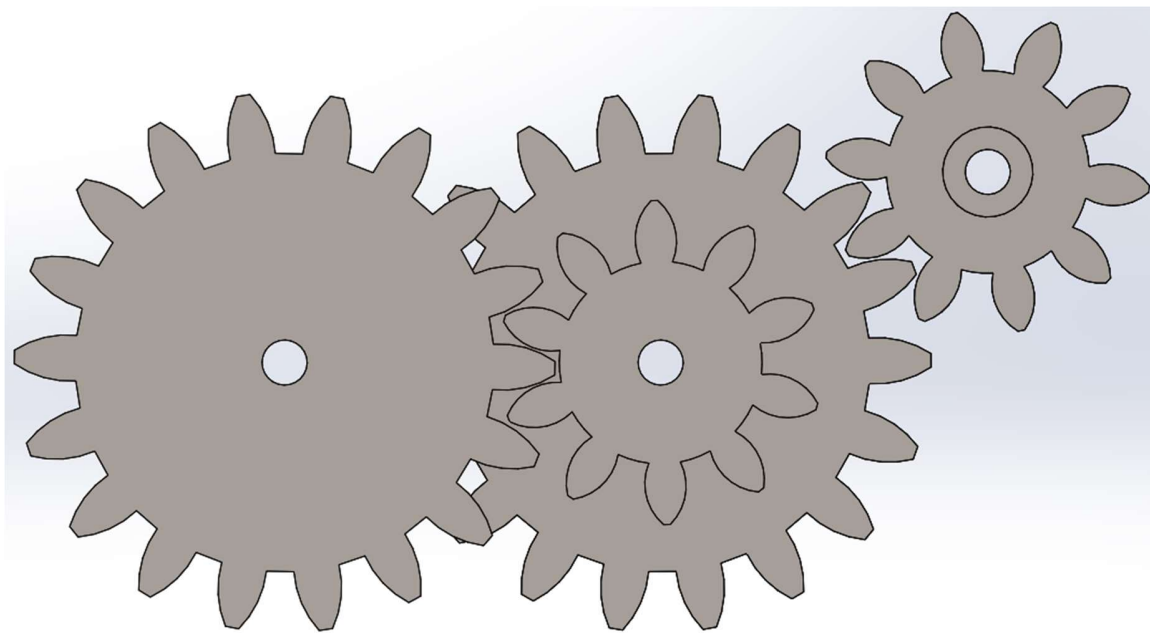


Ilustración 37. Tren de Engranajes

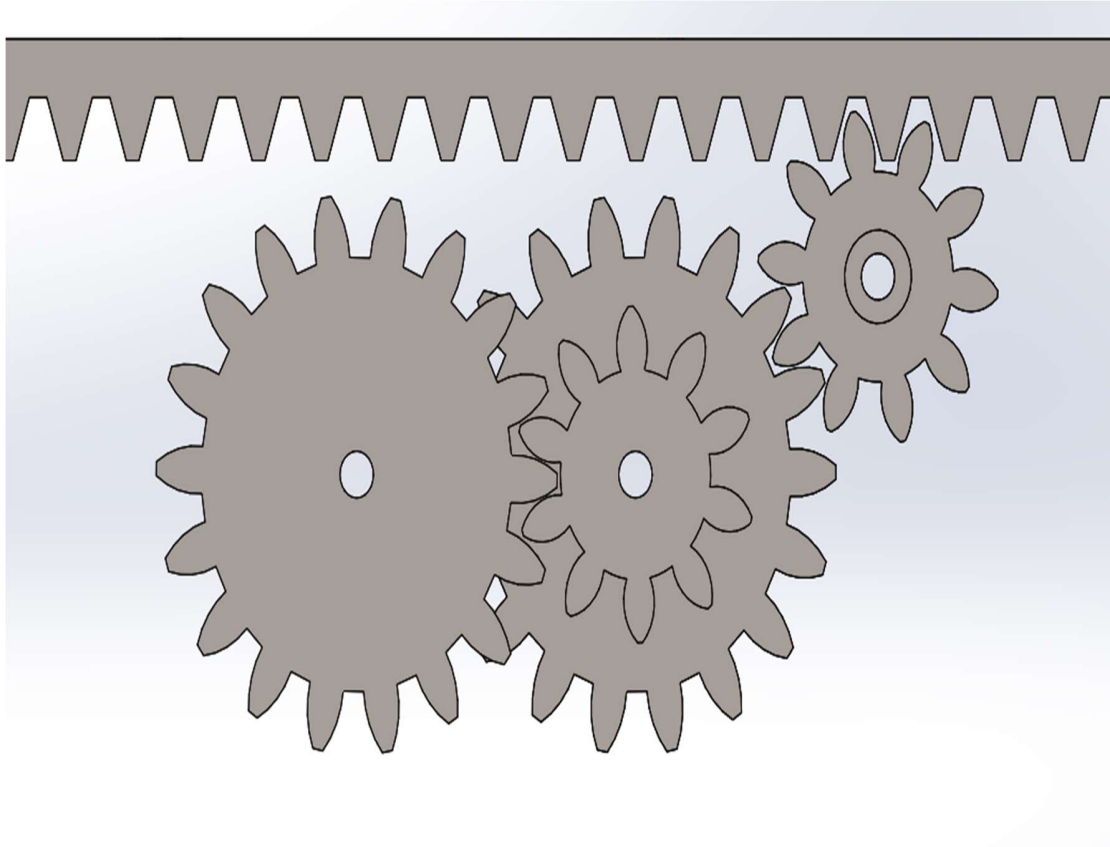


Ilustración 38. Mecanismo de entrada y salida de Base Giratoria.

Posteriormente se realizó la reconstrucción y ensamble de las piezas en el software de diseño solidworks 2023.

el primer ensamble mostrado es la charola la cual consta de las siguientes piezas:

- cremallera dentada
- base con riel
- motor nema
- base circular
- 2 engranes rectos de 18 dientes
- 2 engranes rectos de 10 dientes

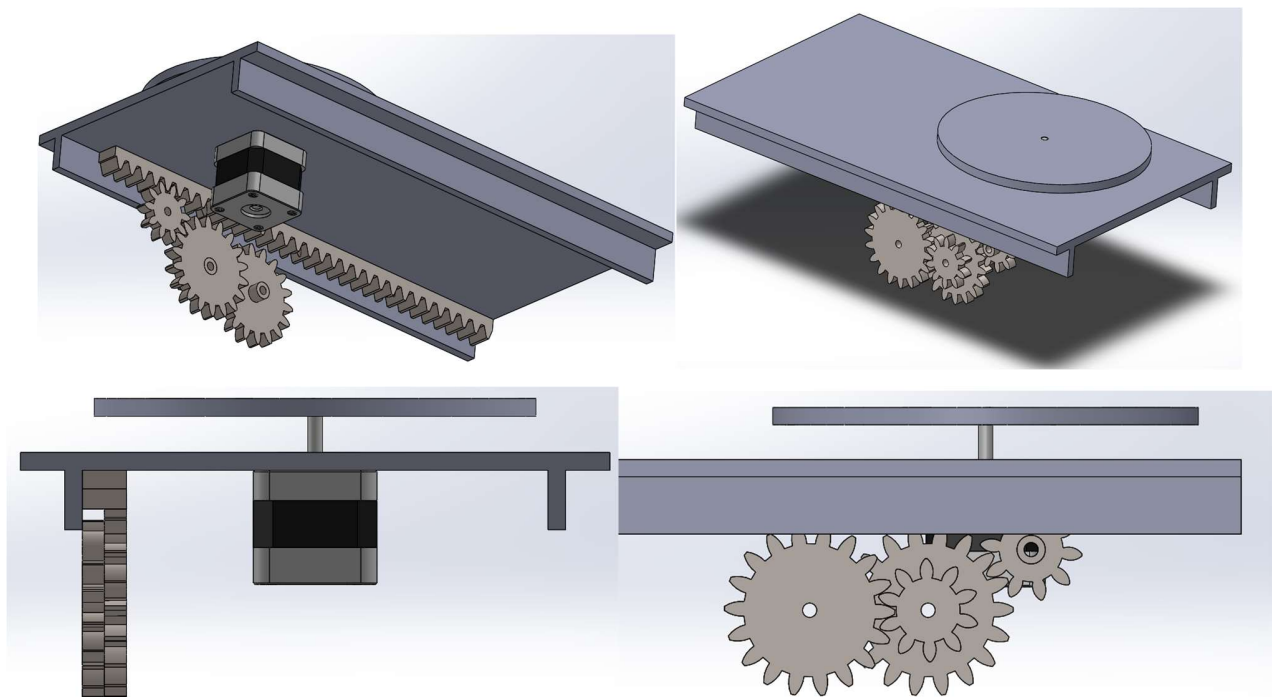


Ilustración 39. Vistas del Mecanismo de Entrada/Salida

En este punto del diseño se nos presentó una problemática que era el centrado del láser lo cual el centrado del láser es sencillo, siempre y cuando el haz láser pase exactamente por la mitad del plato del escáner para ello se debe alinear la base que se diseñó que soportará el láser con el plato antes de fijarlo a la base principal. Se tomó como referencia el siguiente soporte de puntero láser:



Ilustración 40. Base puntero laser

En SolidWorks:

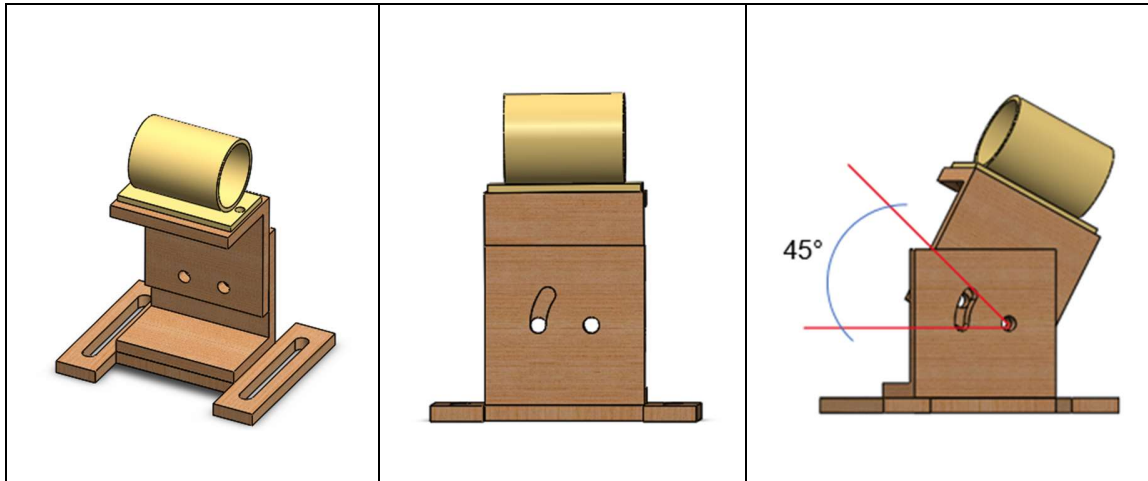


Ilustración 41. Demostración del movimiento del puntero laser diseñado

Y dentro del mismo punto deberíamos verificar el centrado de la cámara respecto al láser

Dentro del mismo punto deberíamos verificar el centrado de la cámara respecto a laser ya que tenemos que tener en cuenta el ángulo por el cual traza el perfil el haz laser para de esta manera obtener la mejor imagen de esta manera Respecto al centrado de la cámara, como sabemos el láser 3D diseñado corresponde a un tipo de escaneo por triangulación espacial que quiere decir que el objeto el láser y la cámara forman un triángulo.

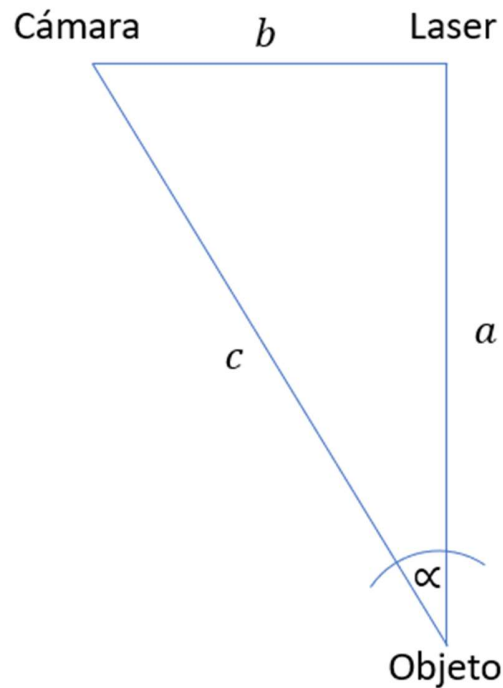


Ilustración 42. Enfoque Geométrico de Triangulación.

El ángulo α se calculó tomando en cuenta que la distancia entre la cámara y el láser es conocida, así como la distancia del láser al centro del plato que por lógica también es el centro del objeto a escanear, aplicando formulas trigonométricas en este caso α dio como resultado 31° , este dato es importante ya que en la programación para el procesamiento digital de imágenes será de ayuda ya que nos indica en que ángulo se tomó la foto para formar la nube de puntos.

También se diseñó una base para la cámara demasiado parecida a la del puntero laser solo que girará horizontal y verticalmente esto ayudará a ajustar la cámara sin estar moviendo toda la base.

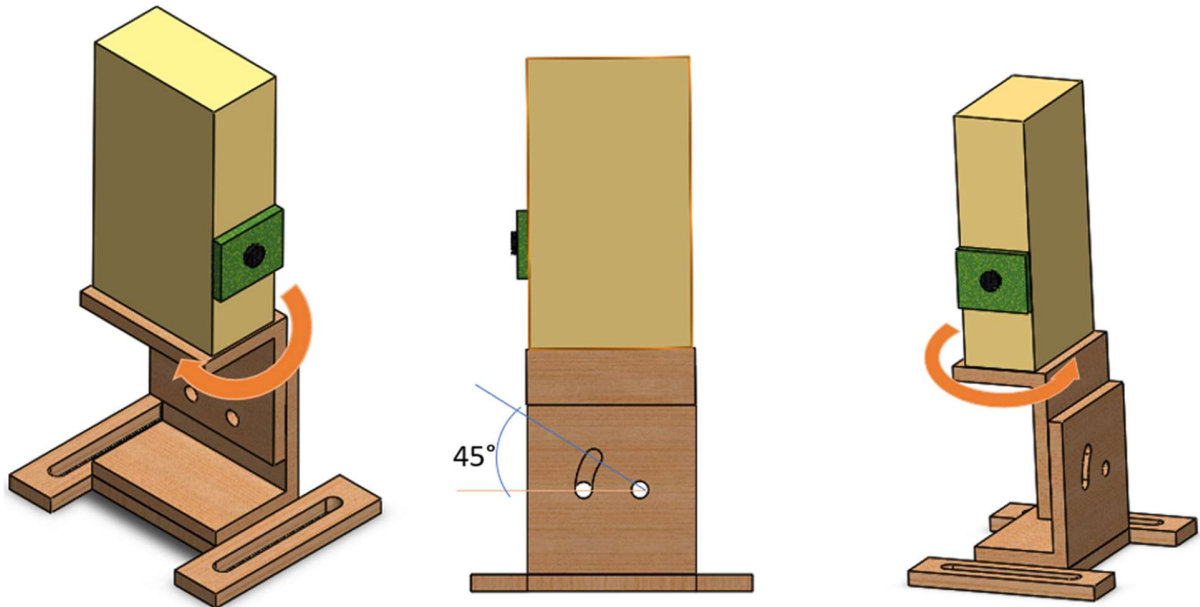


Ilustración 43. Diseño de la base de la Cámara

Es así como se realizó esta propuesta de diseño solo para los tres puntos que al inicio se mencionaron, se da oportunidad de reforzar la parte de control y ayuda al escaneo en general además de que el diseño al interior es más estético.

Continuando con el siguiente ensamble es de la tapa frontal la cual tiene un orificio de apertura para la puerta y consta de las siguientes piezas:

- tapa frontal
- puerta de la tapa frontal con riel
- cremallera dentada
- 2 engranajes rectos de 18 dientes
- 2 engranes rectos de 10 dientes

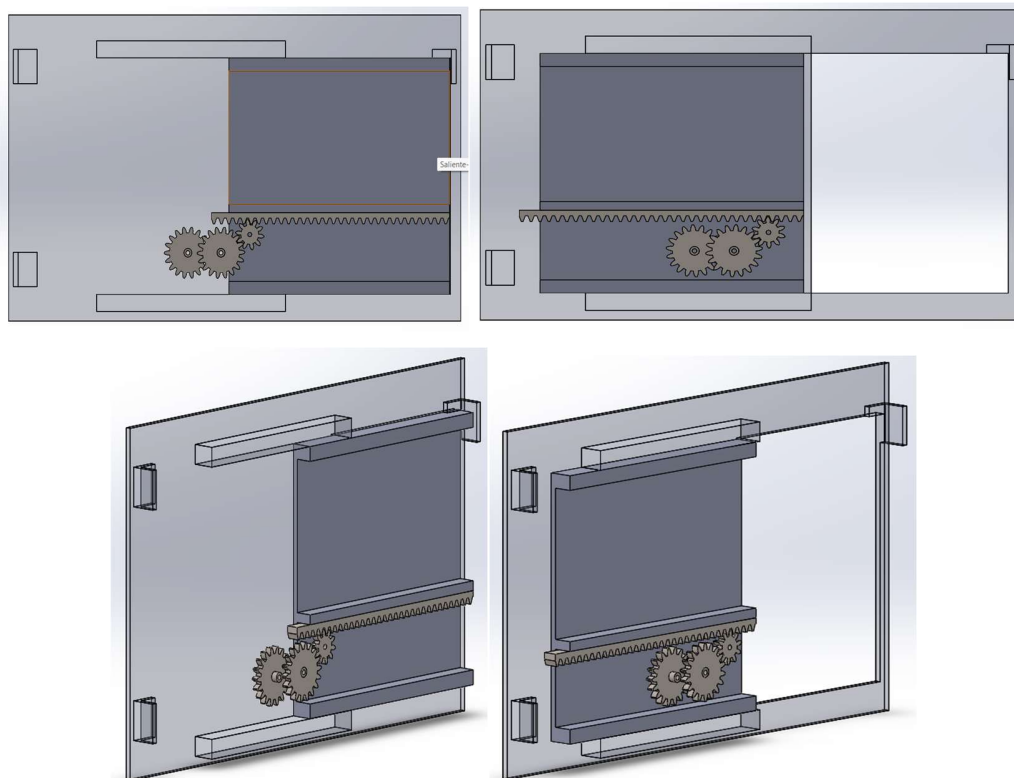


Ilustración 44. Vistas del Mecanismo de Cierre/Apertura del Sistema

Una vez realizado los ensambles se procede a agrupar todo en un solo ensamble el cual consta de todas las piezas siguientes:

- estructura del Escáner 3D
- 4 engranes de 18 dientes
- 4 engranes de 10 dientes
- 2 cremalleras dentadas
- Bandeja móvil
- Base giratoria
- puerta de apertura
- motor nema 16
- Tapa superior de la estructura

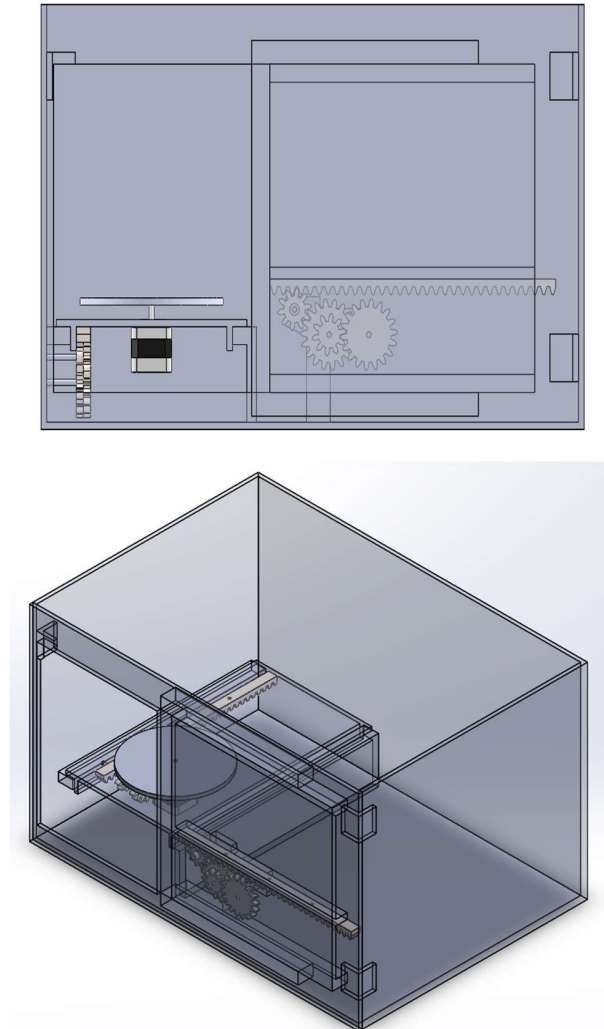


Ilustración 45. Ensamble del sistema en general

Una vez teniendo la propuesta clara procedimos a diseñar el código para el control automatizado de los motores y poder controlar la apertura expulsión introducción y cierre del sistema todo esto mediante una interfaz.

Iniciamos identificando y seleccionando los pins y puertos que utilizaremos.

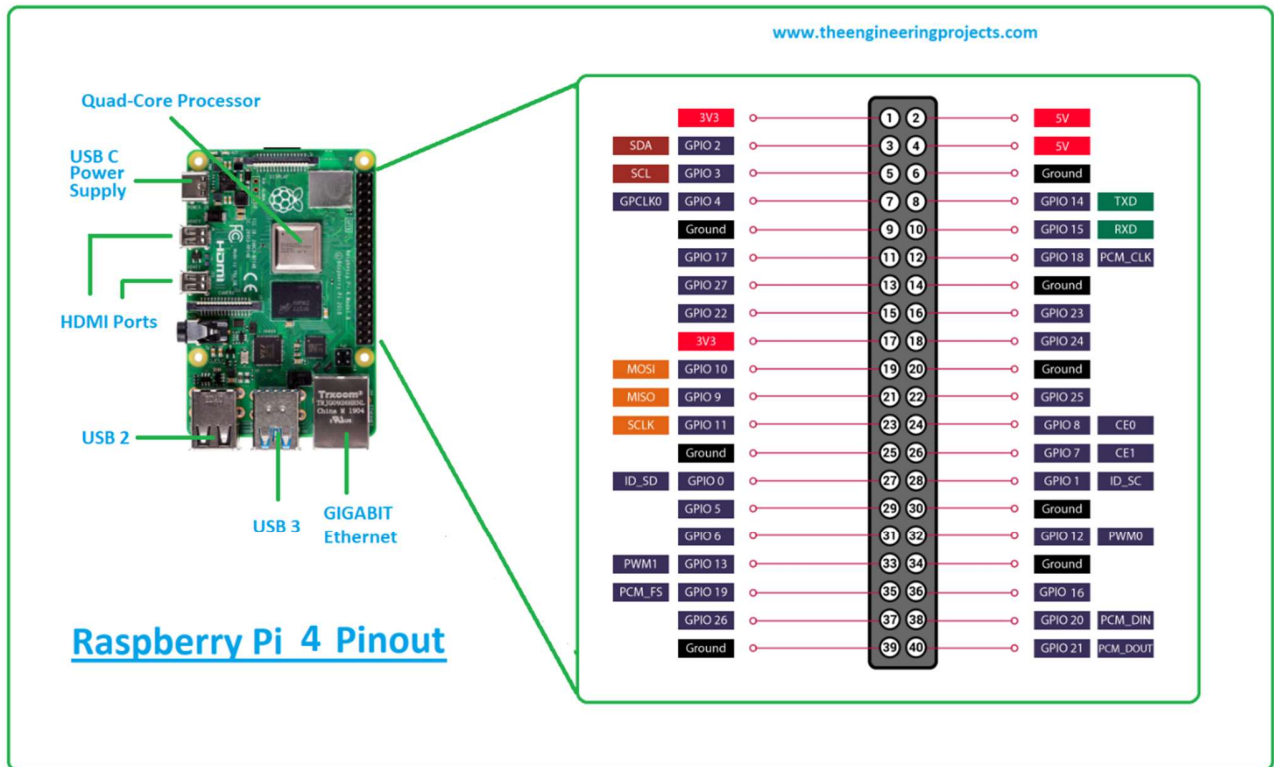


Ilustración 276. Datasheet de Pinout de Raspberry Pi 4.

para el Escáner 3D utilizamos puertos USB con la finalidad de poder conectar una cámara secundaria, Conectar el bus de datos para la pantalla touch y un puerto extra para una memoria flash para poder cargar los códigos, se utilizaron de igual forma el puerto USB power supply como puerto de carga para la tarjeta raspberry pi 4 y el puerto hdmi ports Para mostrar la información en la pantalla.

también se utilizaron los pines denominados GPIO PINES con los cuales podemos controlar pulsos de salida y nos permite reconocer señales de entrada por lo cual procedimos a utilizarlos para el control de motores, iluminación interna y encendido del laser de modo que la propuesta para la automatización del sistema quedó marcado de la siguiente forma:

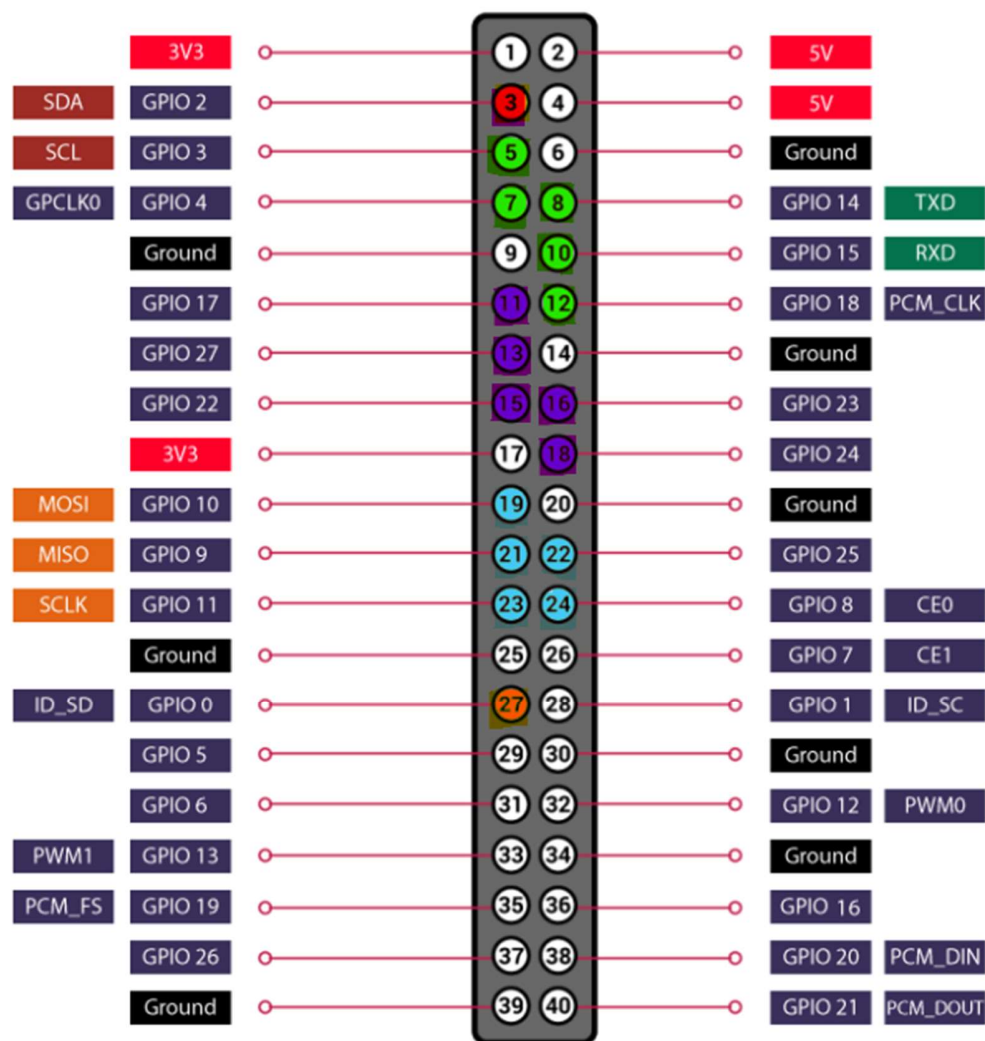


Ilustración 47. Selección de Pinout utilizados en la programación y físicamente.

Utilizamos una configuración de colores personalizada en este caso queda de la siguiente forma:

- ● Para el láser
- ● Pines especificados del motor primario (para movimiento de la pieza)
- ● Pines especificados del motor secundario (para expulsión y entrada de la pieza)
- ● Pince especificados del motor terciario (apertura y cierre de la puerta)

- • Iluminación interna del sistema.

Podemos destacar que para controlar cada motor se requiere de 5 pines esto es porque se utiliza una placa individual la cual controla la entrada de corriente al motor y los pulsos para el encendido y giro del motor al igual que el número de pasos y la velocidad:

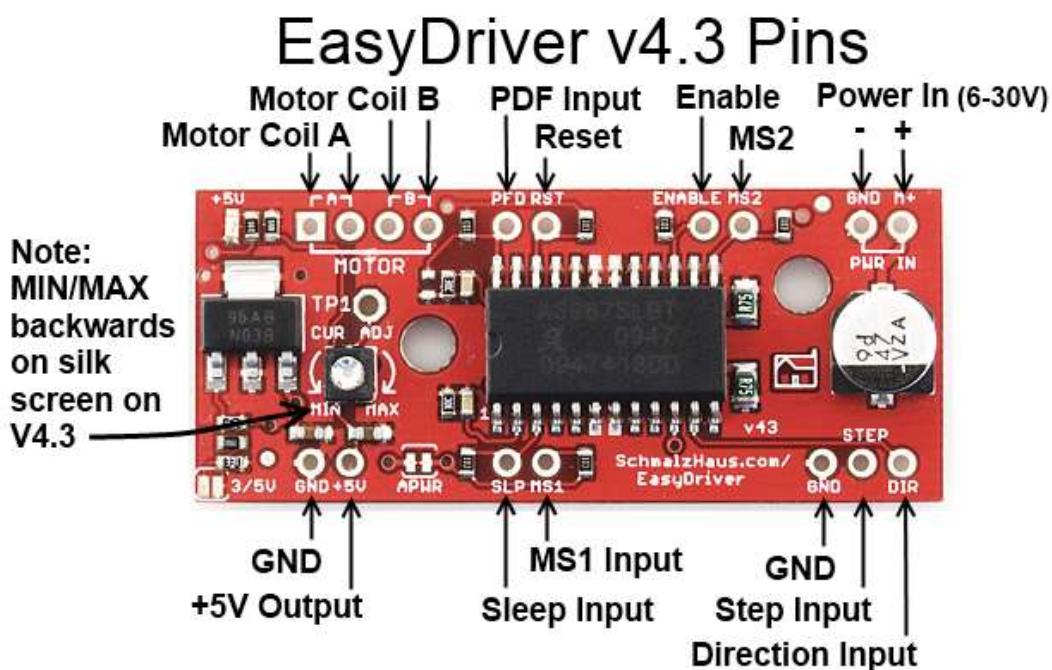


Ilustración 48. Datasheet de tarjeta EasyDriver

Los pines que utilizamos de esta tarjeta controladora para el motor nema fueron los siguientes:

- Direction input
- Step
- Ms1 input
- Ms2
- Enable

Las terminales ya mencionadas son únicamente los pines de salida que se utilizarán de la raspberry y que conectarán con la tarjeta easy-driver para el control del motor y para la alimentación será necesario conectar una fuente externa la cual ira conectada a la tarjeta controladora easy-drive.

Se generó un diagrama eléctrico el cual nos ayuda a identificar los pines conectados a la placa para cada motor el cual se muestra a continuación:

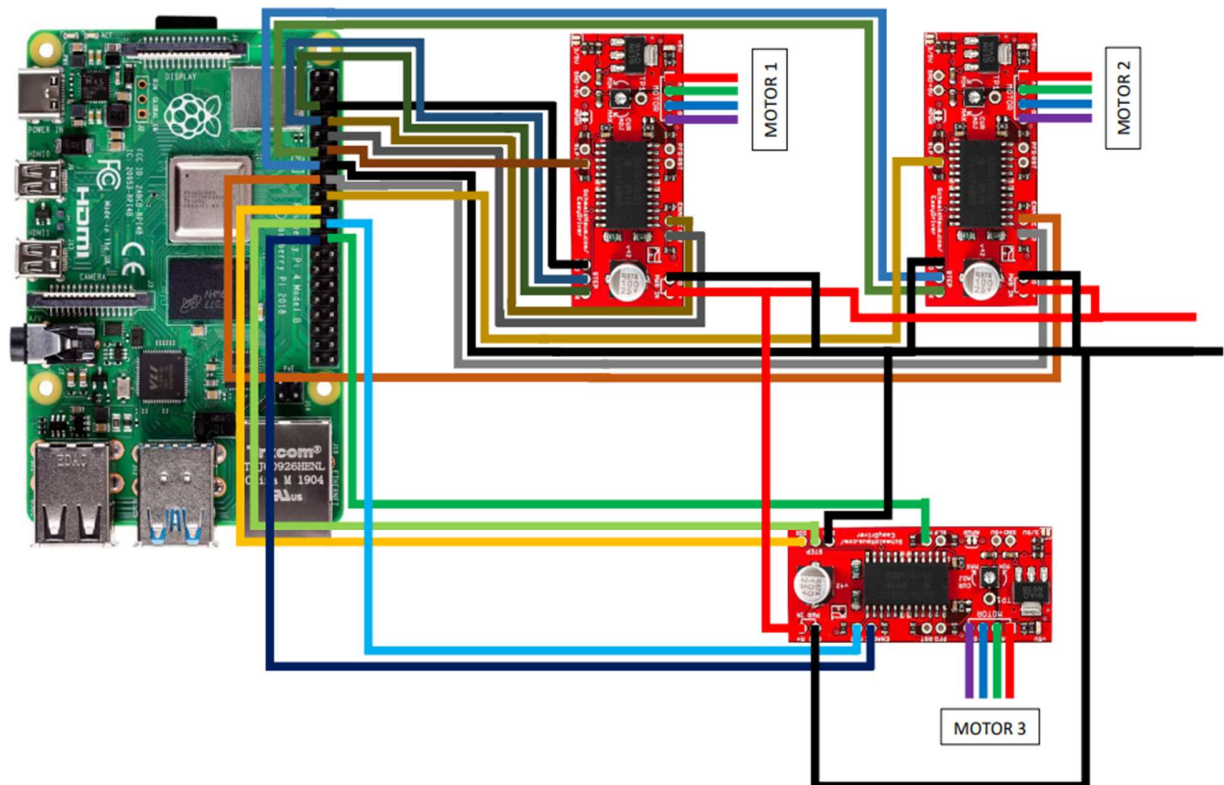


Ilustración 49. Diagrama eléctrico de configuración y conexión para los motores requeridos

A continuación, se muestran las configuraciones realizadas para cada motor:

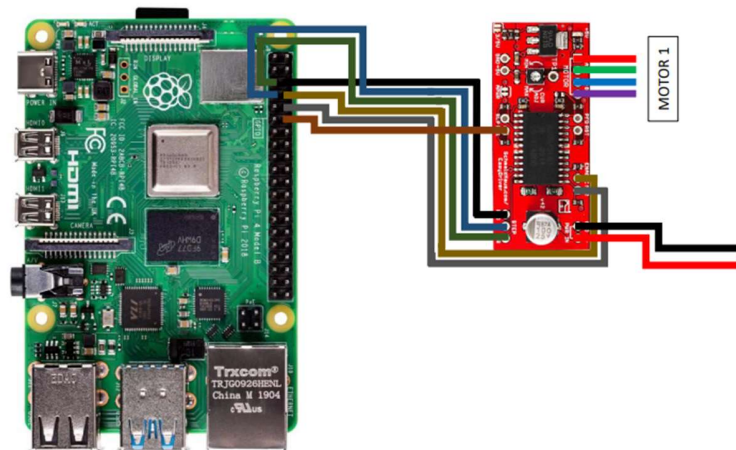


Ilustración 50. Diagrama de conexión de motor primario.

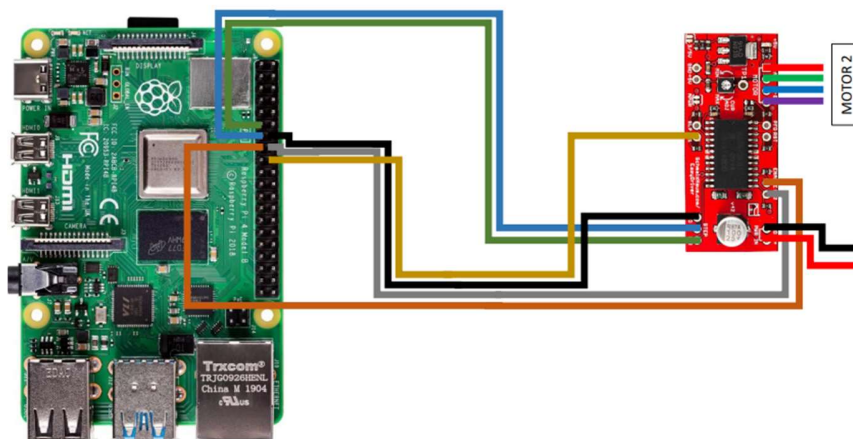


Ilustración 51. Diagrama de conexión de motor secundario.

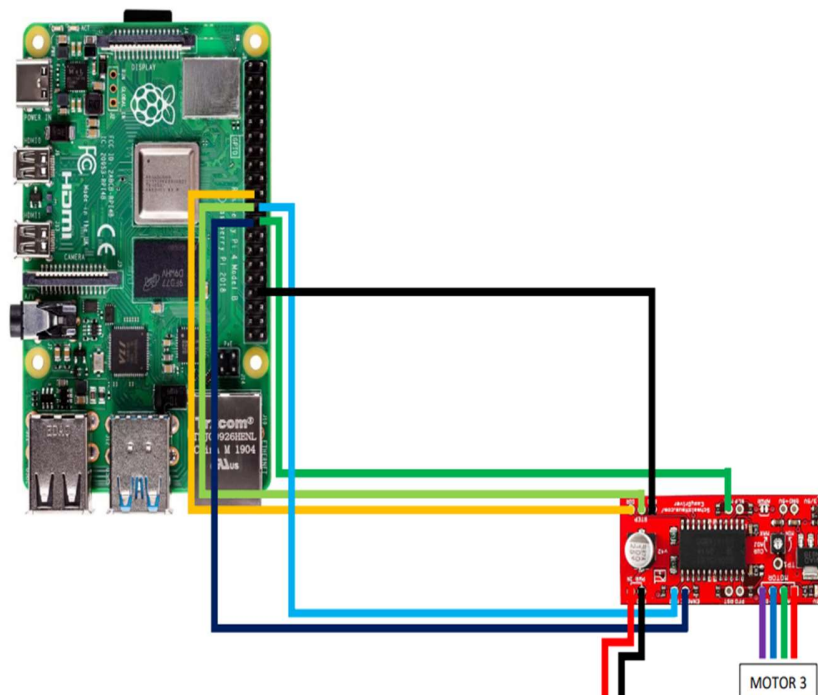


Ilustración 52. Diagrama de conexión de motor terciario

Una vez que determinamos los pines físicos a utilizar Continuamos con la programación de dichos pines digitalmente todo esto gracias al software python que viene precargado en el sistema de la raspberry pi 4. Para poder realizar la interfaz se trabajó con un programa principal el cual llama a diferentes funciones según sea el caso.

Las funciones se metieron en archivos por separado especialmente para lo que se requería en este caso uno fue asignado para determinar pines de entrada y salida para lo antes mencionado quedando asignado de la siguiente forma:

```

6  #--- Configuración de Pines GPIO ---#
7  #el pin de luz se quedara en el pin 27 (CC naranja)
8  pinLuz = 27          #Pin de activacion de luz
9  #el laser se quedara en el pin 3 (CC rojo)
10 pinLaser = 3         #Pin para activar y desactivar el Laser Lineal
11 #Estas variables se ocuparan de igual forma para los demas motores
12 #posterior mente se tendran que ajustar para la velocidad
13 numSteps = 200       #Número de pasos del motor a pasos
14 microPausa = 0.0335 #0.005 #Número de segundos de pausa
15 #----- primer motor (CC verde) -----#
16 pinDir1 = 5          #Pin DIR Controlador Easy Driver (blanco/purpura)
17 pinStep1 = 7         #Pin Step Controlador Easy Driver (rosa/azul marino)
18 pinMS11 = 12         #Pin MS1 para escala del Easy Driver (verde agua/gris)
19 pinMS21 = 10         #Pin MS2 para escala del Easy Driver (azul marino/café)
20 EnableM1 = 8         #Pin Enable para activar el control del motor en el Easy Driver (café/azul marino)
21 #Ground en el pin 6
22 #----- segundo motor (CC morado) -----#
23 pinDir2 = 11         #Pin DIR Controlador Easy Driver (blanco/purpura)
24 pinStep2 = 13        #Pin Step Controlador Easy Driver (rosa/azul marino)
25 pinMS12 = 18         #Pin MS1 para escala del Easy Driver (verde agua/gris)
26 pinMS22 = 16         #Pin MS2 para escala del Easy Driver (azul marino/café)
27 EnableM2 = 15        #Pin Enable para activar el control del motor en el Easy Driver (café/azul marino)
28 #Ground en el pin 14
29 #----- Tercer motor (CC azul) -----#
30 pinDir3 = 19         #Pin DIR Controlador Easy Driver (blanco/purpura)
31 pinStep3 = 21        #Pin Step Controlador Easy Driver (rosa/azul marino)
32 pinMS13 = 24         #Pin MS1 para escala del Easy Driver (verde agua/gris)
33 pinMS23 = 22         #Pin MS2 para escala del Easy Driver (azul marino/café)
34 EnableM3 = 23        #Pin Enable para activar el control del motor en el Easy Driver (café/azul marino)
35 #Ground en el pin 20

```

Ilustración 53. Código con asignación de Pinout

De igual forma se configuraron el encendido y apagado de cada pin utilizado

```

Config_Motores (reajuste de pines).py * <
36 #----- Configuracion de I/O para cada motor -----#
37 #----- PINES UTILIZADOS PARA LUZ LASER LED Y BOTONES -----#
38 GPIO.setup(pinLaser,GPIO.OUT)
39 GPIO.setup(pinLuz,GPIO.OUT)
40 GPIO.output(pinStep1,0)
41 GPIO.output(pinLaser,0)
42 GPIO.output(EnableM1,1)
43 #----- primer motor -----#
44 GPIO.setup(pinDir1,GPIO.OUT)
45 GPIO.setup(pinStep1,GPIO.OUT)
46 GPIO.setup(pinMS11,GPIO.OUT)
47 GPIO.setup(pinMS21,GPIO.OUT)
48 GPIO.setup(EnableM1,GPIO.OUT)
49 #----- segundo motor -----#
50 GPIO.setup(pinDir2,GPIO.OUT)
51 GPIO.setup(pinStep2,GPIO.OUT)
52 GPIO.setup(pinMS12,GPIO.OUT)
53 GPIO.setup(pinMS22,GPIO.OUT)
54 GPIO.setup(EnableM2,GPIO.OUT)
55 GPIO.output(pinStep2,0)
56 GPIO.output(EnableM2,1)
57 #----- tercer motor -----#
58 GPIO.setup(pinDir3,GPIO.OUT)
59 GPIO.setup(pinStep3,GPIO.OUT)
60 GPIO.setup(pinMS13,GPIO.OUT)
61 GPIO.setup(pinMS23,GPIO.OUT)
62 GPIO.setup(EnableM3,GPIO.OUT)
63 GPIO.output(pinStep3,0)
64 GPIO.output(EnableM3,1)

```

Ilustración 54. Configuración y ctrl de Pinout

Una vez terminado el configurado de cada terminal a utilizar se procede a realizar la programación de cada función a realizar en este caso la apertura el cierre la entrada y expulsión de la pieza Esto de tal forma que cuando la interfaz requiera alguna acción simplemente la mande a llamar y cuando termine no genere más consumo de memoria RAM y de esta forma podemos Quiero utilizar una función las veces que necesitemos. las configuraciones para cada motor quedan de la siguiente forma:

```

Config_Motores (reajuste de pines).py * x
101 #----- Proceso del motor a pasos Nema C/Laser -----#
102 def Motor():
103     #global Terminar
104     GPIO.output(pinDir,0)           #Establezco una dirección (0 o 1)
105     GPIO.output(pinMS1,0)          #Calibracion de escala a 1:1
106     GPIO.output(pinMS2,0)          #Calibracion de escala a 1:1
107     #GPIO.output(pinLuz,0)
108     time.sleep(microPausa)          #Esperamos una micropausa
109     GPIO.output(pinLaser, True)     #Activamos el laser
110     for x in range(0,numSteps):
111         GPIO.output(Enable, 0)      #Activamos el control del motor con EasyDriver
112         GPIO.output(pinStep, True)  #Activamos el pulso para motor
113         print("Paso del Motor Numero: " + str(x + 1)) #Registro del N° de pasos en consola
114         time.sleep(microPausa)
115         GPIO.output(pinStep, False) #Desactivamos el pulso para el motor
116         time.sleep(microPausa)      #Esperamos una micropausa
117         #Terminar = x
118         """if Stop == True:
119             break"""
120     GPIO.output(pinLaser, False)     #Desactivamos el laser
121     GPIO.output(Enable,1)            #Bloqueamos el control del motor
122     time.sleep(microPausa)          #Esperamos una micropausa
123 #----- Proceso del motor a pasos Nema S/Laser -----#
124 def Motor_Sin_Laser():
125     #global Terminar
126     GPIO.output(pinDir,0)           #Establezco una dirección (0 o 1)
127     GPIO.output(pinMS1,0)          #Calibracion de escala a 1:1
128     GPIO.output(pinMS2,0)          #Calibracion de escala a 1:1
129     #GPIO.output(pinLuz,0)          #Desactivacion de luz
130     time.sleep(microPausa)          #Esperamos una micropausa
131     GPIO.output(pinLaser, False)    #Desactivamos el laser
132     for x in range(0,numSteps):
133         GPIO.output(Enable, 0)      #Activamos el control del motor con EasyDriver
134         #GPIO.output(pinLuz,1)       #Desactivacion de luz
135         GPIO.output(pinStep, True)  #Activamos el pulso para motor
136         print("Paso del Motor Numero: " + str(x + 1)) #Registro del N° de pasos en consola
137         time.sleep(microPausa)
138         GPIO.output(pinStep, False) #Desactivamos el pulso para el motor
139         time.sleep(microPausa)      #Esperamos una micropausa
140         GPIO.output(Enable,1)        #Bloqueamos el control del motor
141         #GPIO.output(pinLuz,0)       #Desactivacion de luz
142         time.sleep(microPausa)      #Esperamos una micropausa
143         #GPIO.cleanup()             #Para acabar correctamente

```

Ilustración 55. Control para el giro de la pieza y la correcta digitalización.

```

Config_Motores (reajuste de pines).py * x
144 #----- Proceso del motor DE CHAROLA (segundo motor) salida/entrada -----#
145 def Motor_Charola_S():
146     #global Terminar
147     GPIO.output(pinDir,0)          #Establezco una dirección (0 o 1)
148     GPIO.output(pinMS1,0)          #Calibracion de escala a 1:1
149     GPIO.output(pinMS2,0)          #Calibracion de escala a 1:1
150     time.sleep(microPausa)         #Esperamos una micropausa
151     for x in range(0,numSteps):
152         GPIO.output(Enable, 0)      #Activamos el control del motor con EasyDriver
153         GPIO.output(pinStep, True)  #Activamos el pulso para motor
154         print("Paso del Motor Numero: " + str(x + 1)) #Registro del N° de pasos en consola
155         time.sleep(microPausa)
156         GPIO.output(pinStep, False) #Desactivamos el pulso para el motor
157         time.sleep(microPausa)      #Esperamos una micropausa
158         GPIO.output(Enable,1)       #Bloqueamos el control del motor
159         time.sleep(microPausa)      #Esperamos una micropausa
160         #GPIO.cleanup()             #Para acabar correctamente
161 #----- Proceso del motor DE CHAROLA (segundo motor) salida/entrada -----#
162 def Motor_Charola_E():
163     #global Terminar
164     GPIO.output(pinDir,1)          #Establezco una dirección (0 o 1)
165     GPIO.output(pinMS1,0)          #Calibracion de escala a 1:1
166     GPIO.output(pinMS2,0)          #Calibracion de escala a 1:1
167     time.sleep(microPausa)         #Esperamos una micropausa
168     for x in range(0,numSteps):
169         GPIO.output(Enable, 0)      #Activamos el control del motor con EasyDriver
170         GPIO.output(pinStep, True)  #Activamos el pulso para motor
171         print("Paso del Motor Numero: " + str(x + 1)) #Registro del N° de pasos en consola
172         time.sleep(microPausa)
173         GPIO.output(pinStep, False) #Desactivamos el pulso para el motor
174         time.sleep(microPausa)      #Esperamos una micropausa
175         GPIO.output(Enable,1)       #Bloqueamos el control del motor
176         time.sleep(microPausa)      #Esperamos una micropausa
177         #GPIO.cleanup()             #Para acabar correctamente

```

Ilustración 56. Control del acceso y salida del sistema de escaneo 3D.

```

Config_Motores (reajuste de pines).py * x
178 #----- Proceso del motor DE COMPUERTA (tercer motor) apertura y cierre -----#
179 def Motor_Puerta_A():
180     #global Terminar
181     GPIO.output(pinDir,1)          #Establezco una dirección (0 o 1)
182     GPIO.output(pinMS1,0)          #Calibracion de escala a 1:1
183     GPIO.output(pinMS2,0)          #Calibracion de escala a 1:1
184     time.sleep(microPausa)         #Esperamos una micropausa
185     for x in range(0,numSteps):
186         GPIO.output(Enable, 0)      #Activamos el control del motor con EasyDriver
187         GPIO.output(pinStep, True)  #Activamos el pulso para motor
188         print("Paso del Motor Numero: " + str(x + 1)) #Registro del N° de pasos en consola
189         time.sleep(microPausa)
190         GPIO.output(pinStep, False) #Desactivamos el pulso para el motor
191         time.sleep(microPausa)      #Esperamos una micropausa
192         GPIO.output(Enable,1)       #Bloqueamos el control del motor
193         time.sleep(microPausa)      #Esperamos una micropausa
194         #GPIO.cleanup()             #Para acabar correctamente
195 #----- Proceso del motor DE COMPUERTA (tercer motor) apertura y cierre -----#
196 def Motor_Puerta_C():
197     #global Terminar
198     GPIO.output(pinDir,0)          #Establezco una dirección (0 o 1)
199     GPIO.output(pinMS1,0)          #Calibracion de escala a 1:1
200     GPIO.output(pinMS2,0)          #Calibracion de escala a 1:1
201     time.sleep(microPausa)         #Esperamos una micropausa
202     for x in range(0,numSteps):
203         GPIO.output(Enable, 0)      #Activamos el control del motor con EasyDriver
204         GPIO.output(pinStep, True)  #Activamos el pulso para motor
205         print("Paso del Motor Numero: " + str(x + 1)) #Registro del N° de pasos en consola
206         time.sleep(microPausa)
207         GPIO.output(pinStep, False) #Desactivamos el pulso para el motor
208         time.sleep(microPausa)      #Esperamos una micropausa
209         GPIO.output(Enable,1)       #Bloqueamos el control del motor
210         time.sleep(microPausa)      #Esperamos una micropausa
211         #GPIO.cleanup()             #Para acabar correctamente

```

Ilustración 57. Configuración para apertura y cierre del acceso al sistema de escaneo 3D.

De igual forma se generó un archivo el cual tienen como función la activación de las cámaras para ayudarnos a identificar si la pieza está correctamente posicionada el cual quedó de la siguiente forma:

```
Verificar.py * x
1 import numpy as np
2 import cv2
3 import time
4 import RPi.GPIO as GPIO
5
6 def Verificar_P():
7     pinLaser = 16 #Pin para activar y desactivar el Laser Lineal
8     cap = cv2.VideoCapture(0) #definimos la camara 0
9     cap2 = cv2.VideoCapture(1) #definimos la camara 0
10    GPIO.setmode(GPIO.BOARD) #Usar la numeración de pines de la placa
11    GPIO.setup(pinLaser,GPIO.OUT)
12    GPIO.output(pinLaser, GPIO.HIGH) #Activamos el laser
13
14    while True:
15        ret, frame = cap.read() #Devuele un valor booleano si se presenta un cuadro de captura por la camara
16        ret2, frame2 = cap2.read() #Devuele un valor booleano si se presenta un cuadro de captura por la camara
17
18
19        if ret:
20            cv2.imshow('video', frame) #muestra lo captado en la camara
21            cv2.imshow('video2', frame2) #muestra lo captado en la camara
22
23
24        if cv2.waitKey(1) & 0xFF == 27: #En ASCII 27 es la letra ESC por lo que cuando se presione esta tecla sale procede
25            break
26
27    GPIO.output(pinLaser, GPIO.OUT) #Desactivamos el laser
28    cap.release() # libera recursos
29    cap2.release() # libera recursos
30    GPIO.cleanup()
31    cv2.destroyAllWindows()
```

Ilustración 58. Ctrl de camaras y laser para verificación de posición de la pieza.

Y por último el programa principal el cual será la interfaz de usuario que estaremos utilizando para el control y manejo del sistema y el que se encargará internamente de llamar las funciones necesarias.

Interface de control.py * x

```
59 #----- Programa pantalla pricipal -----#
60 syst = tk.Tk()
61 #syst.geometry('900x650')
62 syst.attributes('-fullscreen', True)
63 syst.title("Sistema de escaneo 3D")
64
65 #----- Botones -----#
66 expulsarB = tk.Button(syst, text="Expulsar", command=btn_hide1, font=("arial",20), fg="black", width=10)
67 expulsarB.place(x=900, y=60)
68
69 introducirB = tk.Button(syst, text="Introducir", command=btn_hide2, font=("arial",20), fg="black", width=10)
70 introducirB.place_forget()
71
72 verificarB = tk.Button(syst, text='Verificar', command=Verif_DB, font=("arial",20), fg="black", width=10)
73 verificarB.place(x=120, y=100)
74
75 DigB = tk.Button(syst, text='Digitalizar Pieza', command=Dig_3D, font=("arial",20), fg="black", width=15)
76 DigB.place(x=120, y=200)
77
78 ExitB = tk.Button(syst, text="Exit", command=syst.destroy, font=("arial",20), fg="red", activebackground="red", width=10)
79 ExitB.place(x=920, y=580)
80
```

Ilustración 59. Asignación de función correspondiente a botones de interfaz.

En esta imagen podemos encontrar la configuración de la ventana que está ajustada a pantalla completa y también podemos encontrar las variables y acciones asignadas a cada botón.

En la siguiente imagen podemos encontrar algunas librerías utilizadas que nos permiten hacer uso de ventanas y sus configuraciones y tiempos de espera al igual podemos ver códigos que no sirven para llamar funciones de archivos externos declarando de que archivo proviene y cómo se llama la función a utilizar.

```

Interface de control.py * x
1 import tkinter as tk
2 from tkinter import *
3 import time
4 #import numpy
5 #import threading
6 from Verificar import Verificar_P
7 from P_M import Motor_C_E
8 from P_M import Motor_C_S
9 from P_M import Motor_P_A
10 from P_M import Motor_P_C
11 from Proc_3D import Dig_3D
12
13 def Verif_DB():
14     Verificar_P()
15
16 def Proceso_3D():
17     Dig_3D()
18
19 def btn_hide1():
20     if expulsarB.winfo_ismapped():
21         expulsarB.place_forget()
22         introducirB.place(x=900, y=150)
23         print("Apertura del sistema de escaneo")      #notificacion de proceso
24         time.sleep(0.5)
25         Motor_P_A()
26         time.sleep(1)
27         print("Expulsando bandeja")      #notificacion de proceso
28         time.sleep(0.5)
29         Motor_C_S()
30     else:
31         expulsarB.place(x=900, y=60)
32         #introducirB.configure(text="Ocultar Botón 1")
33
34 def btn_hide2():
35     if introducirB.winfo_ismapped():
36         introducirB.place_forget()
37         expulsarB.place(x=900, y=60)
38         print("Cierre del sistema de escaneo")      #notificacion de proceso
39         time.sleep(0.5)
40         Motor_P_C()
41         time.sleep(1)
42         print("Introduciendo bandeja")      #notificacion de proceso
43         time.sleep(0.5)
44         Motor_C_E()
45         #expulsarB.configure(text="Mostrar Botón 1")
46     else:
47         introducirB.place(x=900, y=150)
48         #expulsarB.configure(text="Ocultar Botón 1")

```

Ilustración 60. Librerías utilizadas y configuración de funciones.

De igual forma en esta imagen podemos identificar que dentro de funciones podemos llamar a otras funciones la ventaja de esto es que nos permite realizar acciones antes de que se haga uso de las funciones a utilizar como lo ocultar o cambiar algún texto de la interfaz de usuario.

A continuación, en la siguiente imagen se puede apreciar cómo es que aparece la interfaz de usuario y las acciones que podemos realizar con ella.

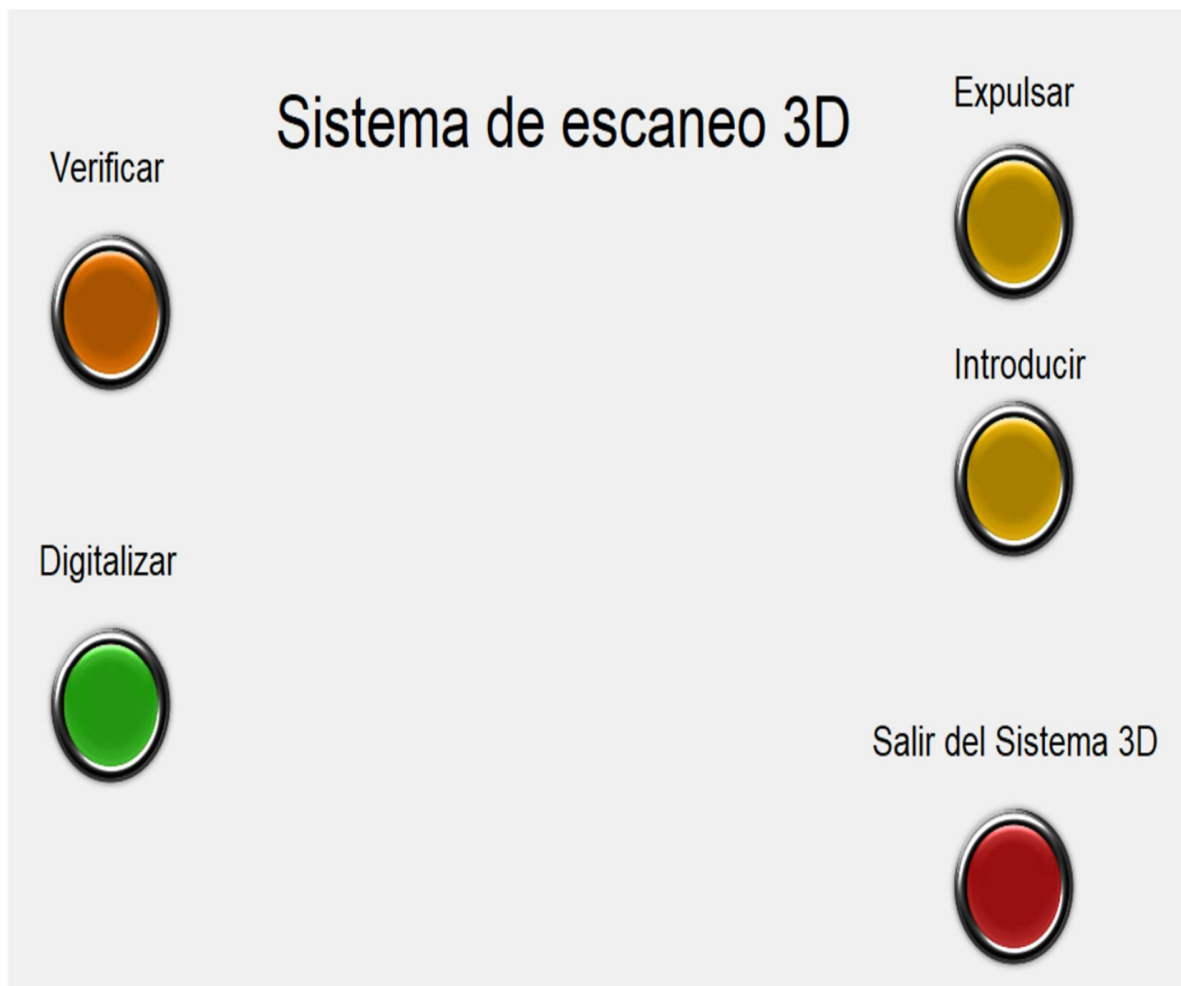


Ilustración 61. Interfaz de usuario para el sistema de escaneo 3D.

Esta es la interfaz de usuario que se diseñó sencilla pero funcional para el trabajo que se necesita. Se puede observar que tenemos inicialmente un código de colores en los botones. Los cuales irán describiendo su función.

Tabla 5. Configuración de botones.

IMAGEN.	TITULO	DESCRIPCIÓN
Expulsar 	expulsar	este botón nos permite abrir la compuerta y expulsar la bandeja donde es colocada la pieza a digitalizar.
Introducir 	introducir	Nos permite realizar la entrada de la bandeja donde estará la pieza y hacer el cierre correcto del sistema de escaneo 3D.
Verificar 	verificar	Nos permite activar las 2 cámaras simultáneamente en conjunto con el láser para permitimos observar y verificar el interior del sistema antes de digitalizar la pieza esto con el fin de verificar que la pieza esté correctamente colocada.
Digitalizar 	digitalizar	Nos permite llamar a la función más compleja que es digitalizar la pieza a su función del motor principal y el accionamiento de las cámaras al igual que el encendido y apagado del láser y la iluminación según sea el caso.
Salir del Sistema 3D 	salir del sistema 3D	Este botón nos permite cerrar la ventana que tenemos y de cierta forma finalizar el uso del programa De forma correcta.

Resultados

La evidencia que presentamos anteriormente demuestra que el ajuste, uso de filtros y el equipo utilizado en la toma de adquisición de imágenes pueden mejorar la definición y el esparcimiento de puntos

Como se pudo notar hay una clara diferencia en la definición de nuestra nube de puntos como se puede apreciar en las imágenes.

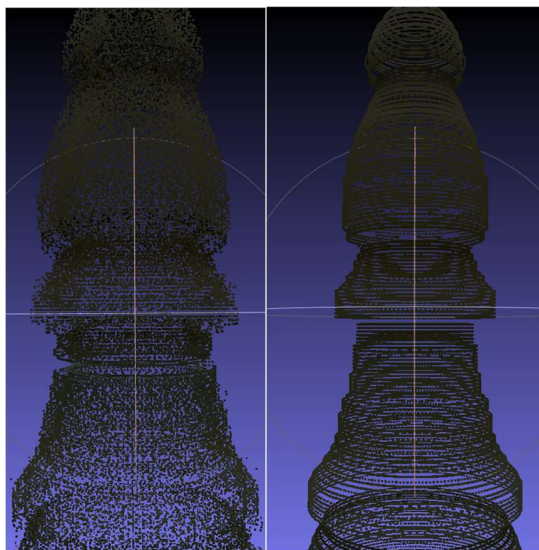


Ilustración 62. Diferencia entre los avances obtenidos.

Esta prueba se realizó mediante nuevos filtros y la utilización de un filtro gaussiano que produce un efecto borroso en la imagen logrando que la línea de moldea el contorno de la pieza sea más uniforme y posteriormente adelgazando la línea conseguimos que mejore la calidad.

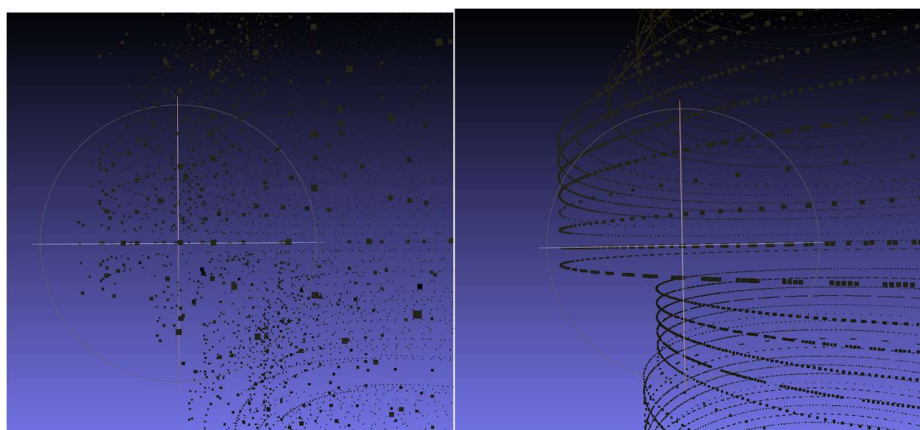


Ilustración 63. Reducción de dispersión de puntos y mejora de definición.

Se logró reducir el ruido que son puntos dispersos en el espacio logrando una mejor definición del objeto en cuestión.

Una vez obtenido este avance se procedió a realizar prueba con los tipos de enmalle o mediante el código de matlab y utilizando software de reconstrucción llamado meshlab esto debido a atributos faltantes presentados en las librerías utilizadas por ejemplo el guardado de enmallado y guardado de archivos en extensiones necesarias para impresión 3D.

Los avances obtenidos mediante este software fueron más eficientes siendo que tuvimos algunos problemas al inicio por ejemplo creación de partes sólidas en áreas donde no existían puntos para enmallar

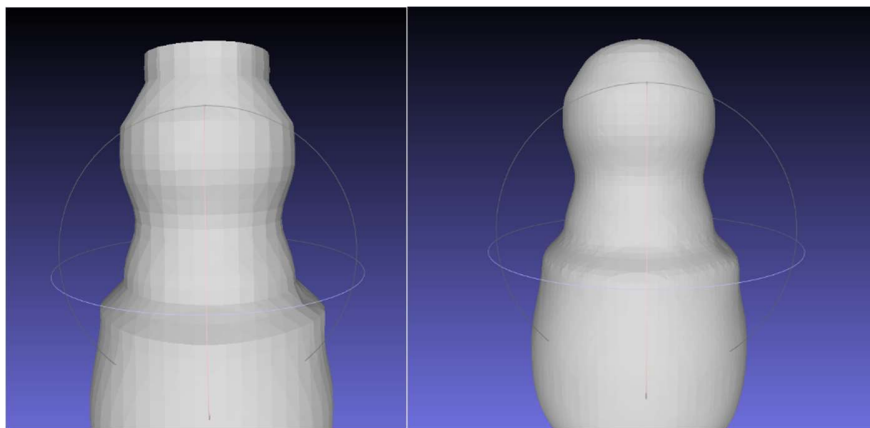


Ilustración 64. Avance de enmallado y sólido.

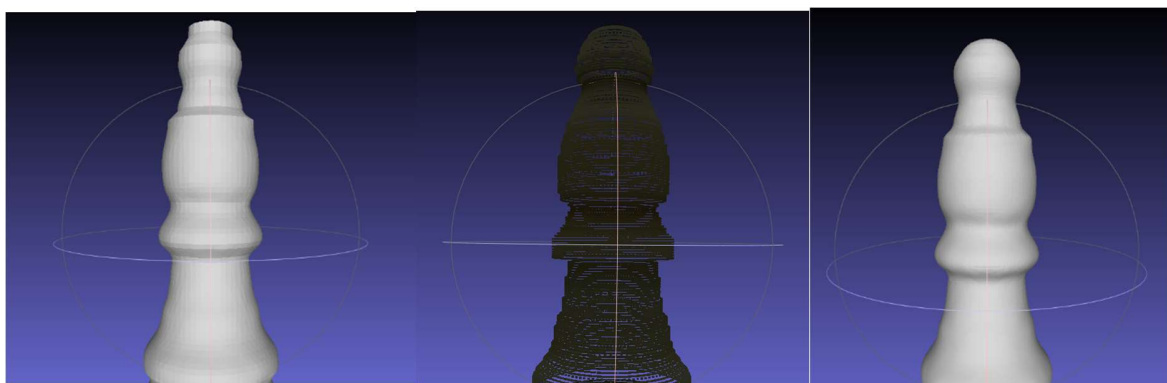


Ilustración 65. Avances obtenidos archivo para impresión.

Una vez mejorando el texturizado y la calidad del enmallado procedimos a convertir el archivo en extensión .STL necesario para el software cura el cual nos ayuda a generar el GCODE para poner en marcha de impresión

Subió en el archivo en cura se procede a hacer ajustes en valores para la impresora requerida de tal forma que la pieza quede lo mejor posible, esto tomando en cuenta valores requeridos para el tipo de material a utilizar en este caso PLA y ABS.

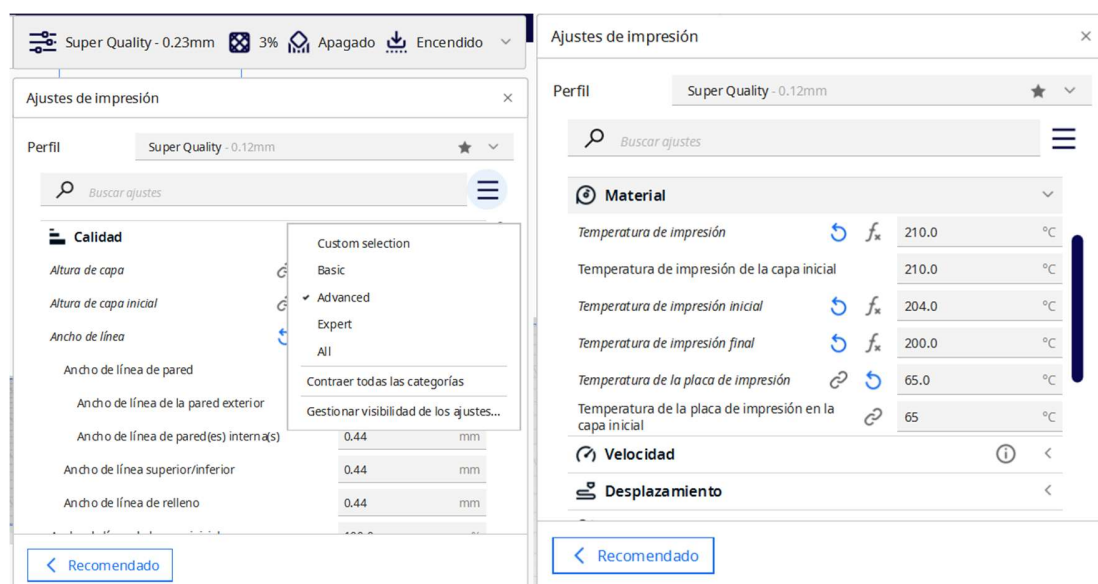


Ilustración 66. Tipo de configuración “avanzado” y ajustes a tomar en cuenta.

Una vez ajustados los parámetros importantes como temperatura inicial temperatura final, temperatura de la cama podemos proceder a dimensionar y ajustar el tamaño de la pieza y observar una vista preliminar de la impresión anexando un promedio aproximado de material a utilizar y tiempo estimado

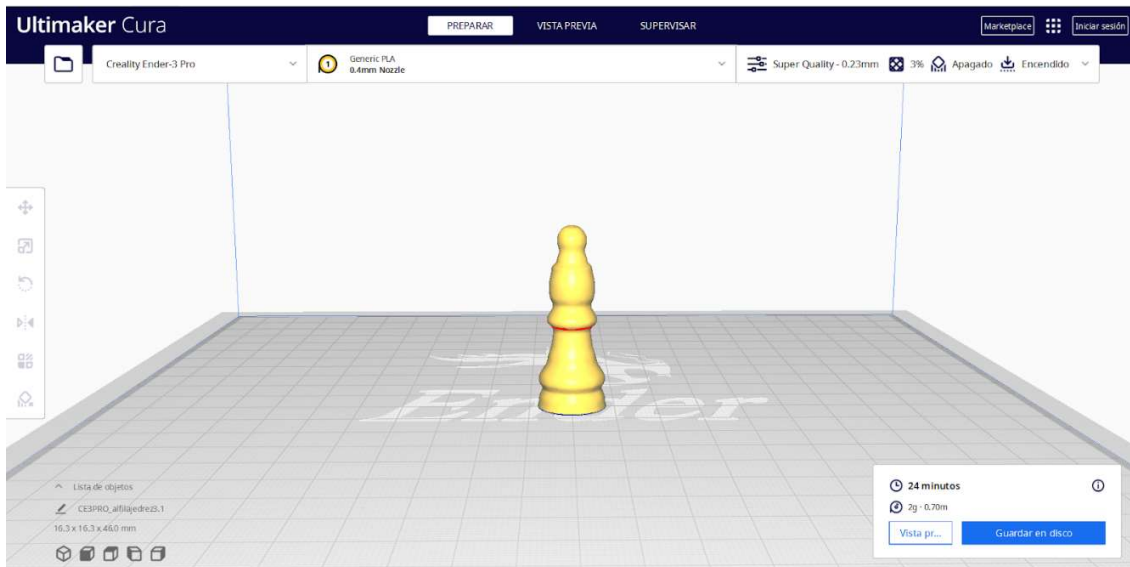


Ilustración 67. Vista preparar proporcionada por el software.

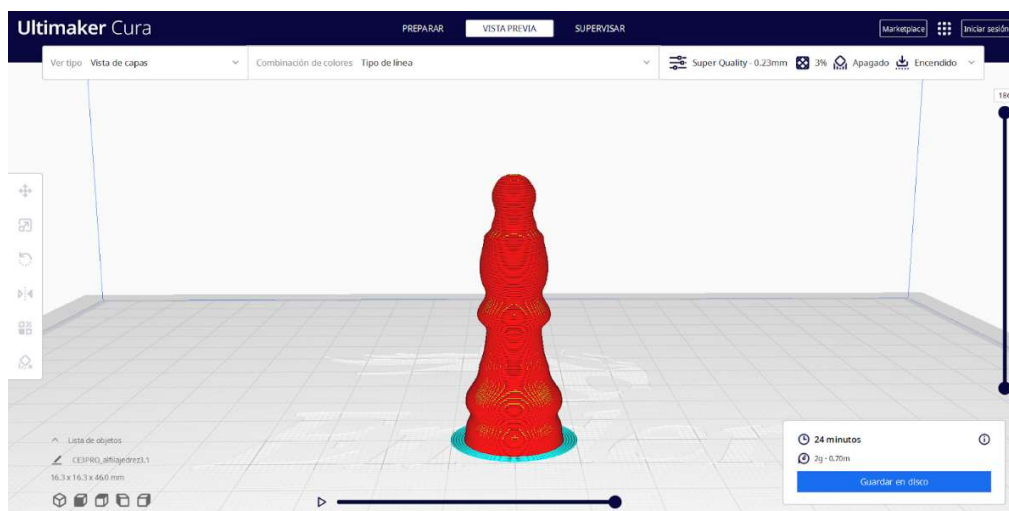


Ilustración 68. Vista previa y simulación de impresión 3D proporcionada por el software.

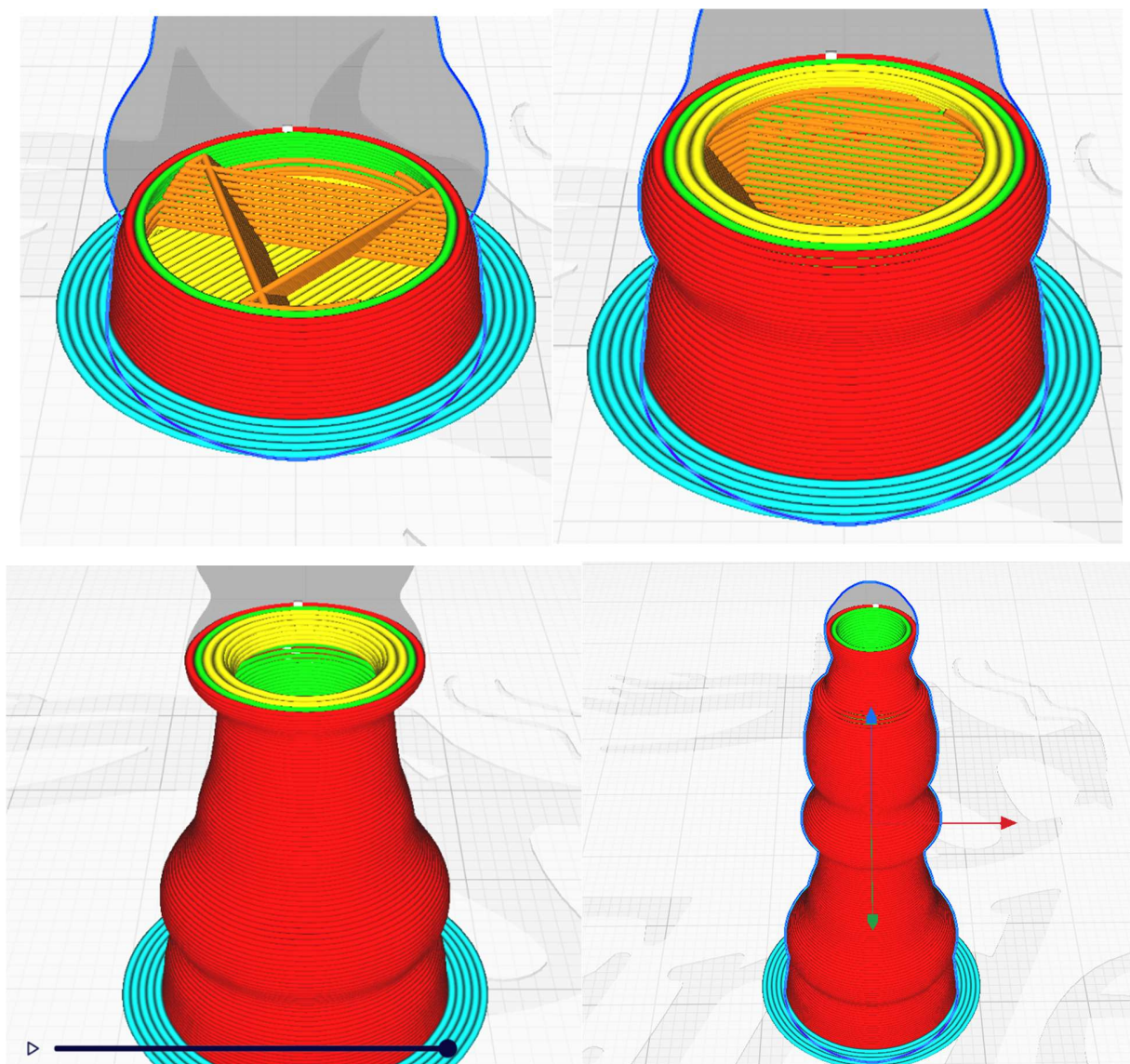


Ilustración 69. Simulación de impresión 3D.

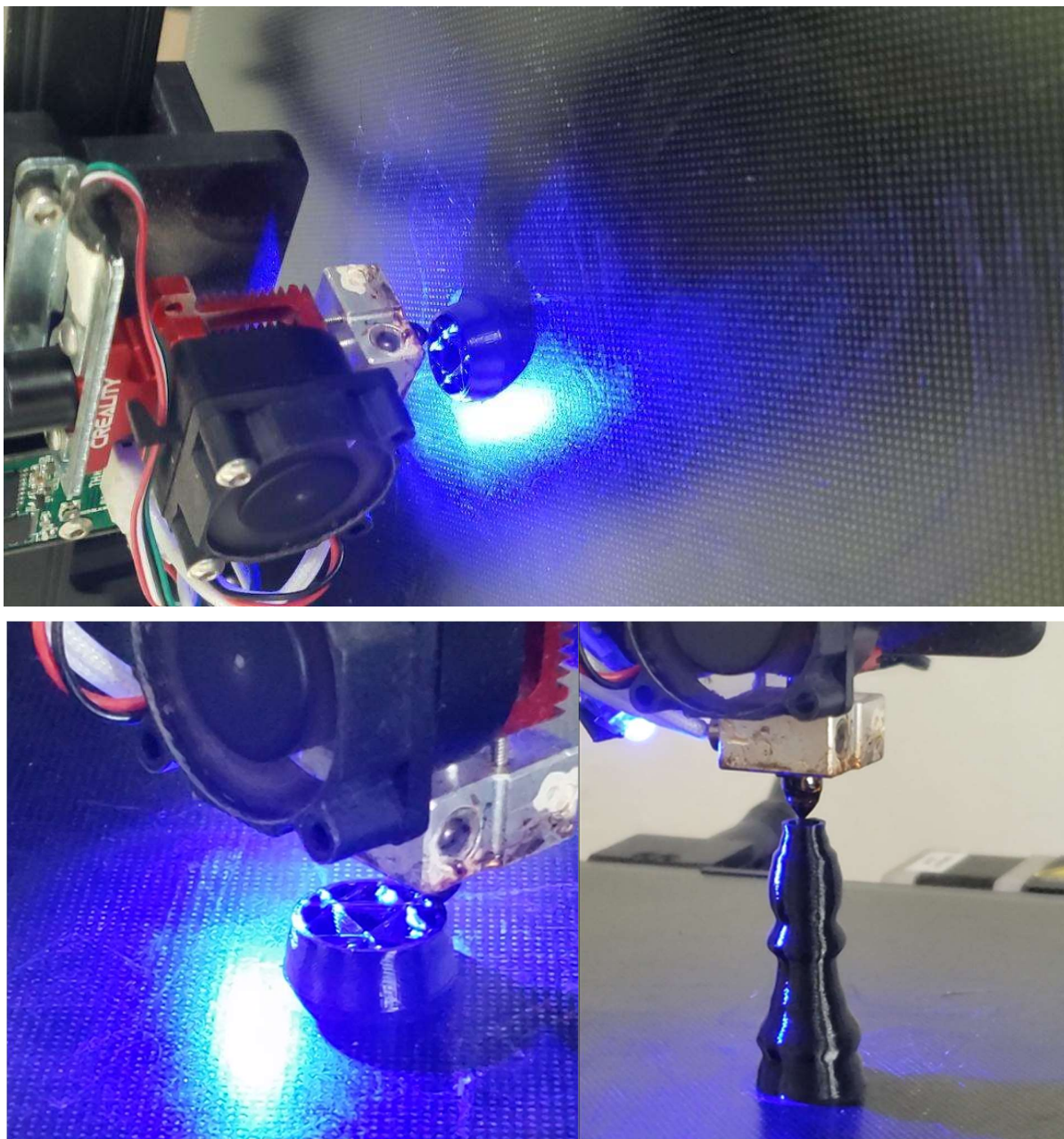


Ilustración 70. Proceso de impresión 3D en material ABS.

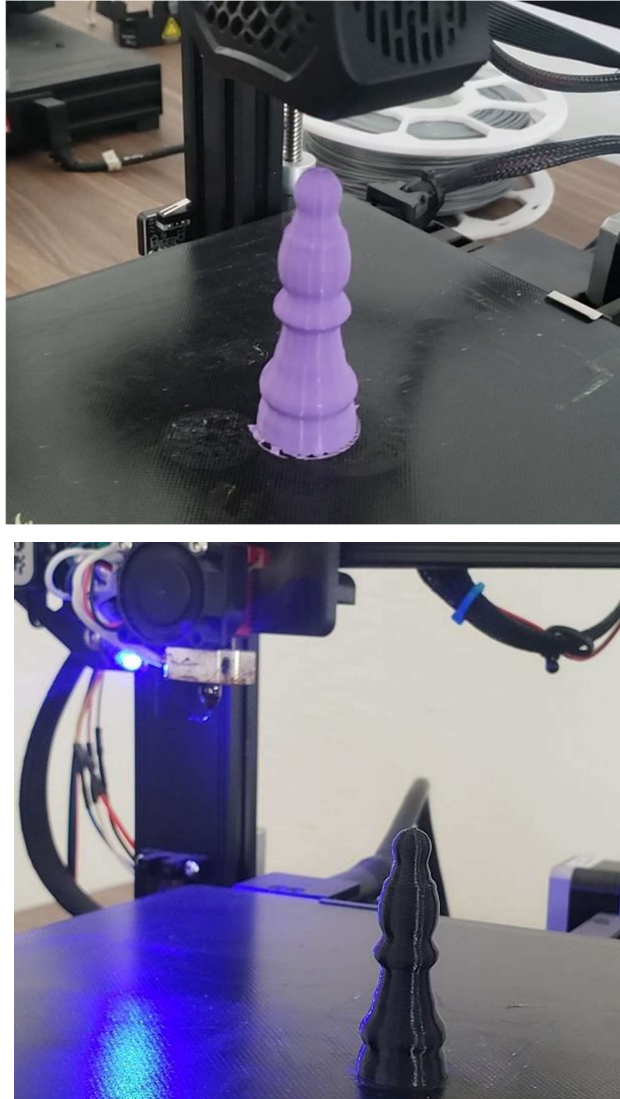


Ilustración 71. Resultados de impresión 3D en material PLA y ABS.

Al finalizar la impresión se obtuvieron resultados favorables en las piezas impresas con ligeros detalles de la impresión, pero eran rigidez Y semejanza en el modelo digital.

Conclusiones

El presente proyecto tiene la posibilidad de aplicarse en áreas de investigación en las que predomina la programación, específicamente el uso de software que permita el diseño de piezas de manera más eficiente y en un corto periodo de tiempo, es decir, De esta manera, durante la realización de este proyecto pudimos visualizar puntos fuertes de mejora e innovación, sobre todo en la forma de procesar datos.

En este contexto, fue posible mejorar el procesamiento de imágenes, definición y filtrado de datos con objetivo de perfeccionar los puntos clave como el espesor de la nube de puntos para obtener un modelado más detallado y apegado a la pieza original, aunque esto implicó un punto positivo también trajo consigo aspectos negativos, tal fue el caso del software, ya que el procesamiento limitado del dispositivo ocasionó que la toma de muestras fuera inexacta, dándonos un resultado demasiado variable, un claro ejemplo fue que , de 360 fotografías requeridas, el programa nos lanzó alrededor de 210 fotos son máximo de 470 fotos, esto debido a varios factores, iniciado por la tarjeta gráfica hasta las condiciones de iluminación, corrigiendo estos factores se logró obtener una variación de 10 fotos, asimismo se obtuvo un mínimo de 355 a un máximo de 365 fotos, aunque mayormente se debía a las condiciones de iluminación, ya que las cámaras utilizadas cuentan con sensores comerciales, no obstante, fue posible obtener resultados favorables en el espesor y definición de las piezas, que nos llevó al siguiente problema: el enmallado, el cual funcionaba bien con nubes de puntos que no estaban tan saturadas de puntos, el problema radica en que al haber una concentración mayor de puntos, la interpolación en el enmallado hace que se encimen las mallas, por lo que optamos por utilizar softwares de enmallado, los cuales nos permitieron editar valores para un enmallado más reducido y preciso, aparte de proporcionarnos un archivo apto para edición e incluso para lectura de softwares en impresiones 3D, obteniendo resultados favorables a la mejora del duplicado de piezas, limitantes en el área del software y el hardware que pueden ser mejoradas.

Competencias desarrolladas

En el desarrollo de este proyecto se realizaron varias investigaciones de las cuales nos dieron la herramientas y conocimientos para poder realizar las actividades necesarias para dar un buen resultado esto conlleva a también el desarrollo de varias habilidades y competencias las cuales fueron el uso de herramientas de software como solidworks desde conocimientos básicos a intermedios los cuales ayudaron para el diseño y ensamble de piezas del sistema propuesto al igual en conocimientos adquiridos en área de programación en softwares como python y MatLab en los cuales se aprendió a utilizar librerías precargadas en el sistema como librerías externas y de igual forma a instalar dichas librerías el manejo de dispositivos de entrada como fue la cámara tipos de codex y formatos para grabación que nos permitieron tener un mejor resultado al igual que identificar el tipo de cámara a utilizar y los tipos de error que podríamos obtener con cada cámara ya que cuentan con lentes diferentes y nos podían producir errores denominados errores esféricos que van relacionados con la curvatura del lente utilizado en la cámara al igual que la apertura de la cámara también llamada apertura de foco el cual una cámara permitía mayor paso de luz que la principal utilizada y esto podría ser afectado para la adquisición de datos otra de las variables que obtuvieron conocimientos previos fue el tipo de iluminación y láser a utilizar se hicieron pruebas con un láser rojo y un láser verde y se determinó el mejor láser teniendo en cuenta información como fue la longitud de onda para cada láser y el tipo de sensor de las cámaras y poder obtener un mejor resultado teniendo en cuenta que el tipo de sensor que tienen las cámaras pueden captar mejor o alterar algunos colores otra competencia desarrollada fue el manejo de tarjetas y controladores físicos para motores así como diseño de circuitos creación de diseños impresos para impresión en tarjetas y fabricación de las mismas, se aprendieron métodos de transferencia de datos a través de conexiones inalámbricas dependiendo de los datos a transmitir y el tipo de conexión inalámbrica a utilizar

Se desarrollaron habilidades y conocimientos para el manejo del software cura el cual está diseñado para impresión 3D y nos permite hacer ajustes las impresoras de materiales plásticos controlar variables como son temperatura velocidad de instrucción del material y velocidad de impresión al igual que nos permite configurar el relleno de la pieza el tipo de relleno y estructura que lleva por dentro qué es un dato importante el cual nos permite volver una pieza más rígida o incluso hueca de igual forma nos permite controlar el

número de capas que queremos que tenga la pieza y nos permite obtener un valor aproximado del tiempo de impresión dependiendo los ajustes del material esto claro es importante para el uso de materiales plásticos denominados PLA los cuales cuentan con diferentes características como pueden ser térmicas plásticas o de dureza que nos permiten obtener mejores acabados según sean necesarios para piezas finales como son flexibilidad dureza rigidez algo a tener en cuenta para poder obtener estas características depende del material a utilizar ya que contamos con una amplia variedad el más común y el que utilizamos fue el tipo PLA al igual que filamento ABS el filamento PLA es un material económico muy fácil de trabajar ya que no requiere altas temperaturas y se dieron fácilmente pero una desventaja es que cuenta con menor resistencia frente a golpes rayones o esfuerzos cortantes sin embargo el filamento ABS es un tipo de filamento con mucha resistencia tanto impactos como altas temperaturas que permite realizar trabajos sobre una pieza una vez impreso y un aspecto importante a resaltar es que este material requiere una temperatura mayor de entre 210 y 250°C para que el material pueda derretirse y adherirse perfectamente las capas.

Fuentes de información

- [1] Mathworks, «Procesado de imágenes digitales,» [En línea]. Available: <https://la.mathworks.com/discovery/digital-image-processing.html>. [Último acceso: 25 marzo 2023].
- [2] Equipo editorial, Etecé., «Imagen,» 2021 agosto 5. [En línea]. Available: <https://concepto.de/imagen/>. [Último acceso: 25 marzo 2023].
- [3] LULU, «Capítulo 1: Principios Básicos de Imágenes y navegación,» 2002. [En línea]. Available: http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/mendez_a_ky/capitulo1.pdf. [Último acceso: 25 marzo 2023].
- [4] R. C. Gonzalez y R. E. Wood, Digital Image Processing, Upper Saddle River, New Jersey: Pearson Education, Inc, 2008.
- [5] S. A. Sanchez Tirado, «Integración de técnicas de manejo de la imagen digital para la web y aplicaciones multimedia. Facultad de Ingeniería en Sistemas e Informática.,» ESPE. Sede Sangolquí., abril 2006. [En línea]. Available: <http://repositorio.espe.edu.ec/handle/21000/906>. [Último acceso: 2023 marzo 20].
- [6] Y. Segura Domingo, «CURSO BÁSICO DE EDICIÓN DE IMÁGENES,» Universidad Politécnica de Valencia, [En línea]. Available: https://yosedo.webs.upv.es/curso_edicion_de_imagenes_nivel_basico/1_7.html. [Último acceso: 20 marzo 2023].
- [7] R. Wainschenker, J. M. Massa y T. Paula, «Procesamiento Digital de Imágenes,» 1 junio 2011. [En línea]. Available: <https://users.exa.unicen.edu.ar/catedras/pdi/FILES/TE/CP1.pdf>.
- [8] C. López Jurado, «Imagen digital: qué es, formatos y cómo se calcula su peso,» CCM Benchmark, 21 febrero 2023. [En línea]. Available: <https://es.ccm.net/aplicaciones-e-internet/museo-de-internet/enciclopedia/10540-que-es-una-imagen-digital-y-como-se-calcula-su-peso/>.
- [9] ELBS, «5 conceptos sobre imagen digital que debes conocer,» Escuela de Liderazgo, 15 octubre 2020. [En línea]. Available: <https://escuelaelbs.com/conceptos-imagen-digital/>.

- [10] Avila, Andres, «Tratamiento de imágenes,3.-Características de la imagen digital,» Google Sites, [En línea]. Available: <https://sites.google.com/site/tratamientodeimagenesandres/3--caracteristicas-de-la-imagen-digital>. [Último acceso: 20 marzo 2023].
- [11] IBM, «Gráficos de histograma,» IBM Corporation , 7 diciembre 2021. [En línea]. Available: <https://www.ibm.com/docs/es/spss-statistics/beta?topic=types-histogram-charts>.
- [12] Adobe, « Aprende cómo leer un histograma en fotografía,» [En línea]. Available: <https://www.adobe.com/es/creativecloud/photography/discover/how-to-read-a-histogram.html#:~:text=Los%20histogramas%20son%20una%20representaci%C3%B3n,negros%2C%20la%20foto%20est%C3%A1%20subexpuesta..> [Último acceso: 28 marzo 2023].
- [13] ArcGIS Pro 3.0, «Histograma de imágenes,» esri, [En línea]. Available: <https://pro.arcgis.com/es/pro-app/latest/help/analysis/geoprocessing/charts/image-histogram.htm>. [Último acceso: 28 marzo 2023].
- [14] V. Atienza Vanacloig, «El histograma de una imagen digital,» Universidad Politecnica de Valencia, [En línea]. Available: https://riunet.upv.es/bitstream/handle/10251/12711/El_histograma_una_imagen_digital.pdf?sequence=1.
- [15] Mathworks, «Trabajar con imágenes en gráficas de MATLAB,» [En línea]. Available: https://la.mathworks.com/help/matlab/creating_plots/working-with-images-in-matlab-graphics.html. [Último acceso: 29 marzo 2023].
- [16] F. Gimenez Palomares, J. Monsoriu y E. Alemany Martinez, «Aplicación de la convolución de matrices al filtrado de imágenes,» Universidad Politécnica de Valencia, 2016. [En línea]. Available: <https://riunet.upv.es/bitstream/handle/10251/69639/4524-15767-1-PB.pdf>.
- [17] Universidad Nacional de La Plata, «Procesamiento de Imágenes,» 2011. [En línea]. Available: http://www2.fisica.unlp.edu.ar/materias/procesamiento_de_imagenes/Clase2_Imagenes_2011.pdf.
- [18] Wikipedia, «Nube de puntos,» 3 julio 2022. [En línea]. Available: https://es.wikipedia.org/wiki/Nube_de_puntos.

- [19] Dronica, «NUBE DE PUNTOS: Qué es y para qué sirve,» 7 octubre 2018. [En línea]. Available: <https://www.dronica.es/blog-de-escaneo-3d/nube-de-puntos-que-es-y-para-que-sirve/>.
- [20] BIMSCAN, «¿Qué es una nube de puntos?,» [En línea]. Available: <https://www.bimscan.es/que-es-una-nube-de-puntos/>.
- [21] S. P. Mateus, «Método de Regularización de Mallas Cuadrilaterales en Reconstrucción de Objetos 3D,» 2008. [En línea]. Available: <https://docplayer.es/11141401-De-mallas-cuadrilaterales-en-reconstruccion-de-objetos-3d.html>.
- [22] B. J. Herráez Concejo, «Segmentación y clasificación de mallas 3D,» [En línea]. Available: <https://riunet.upv.es/bitstream/handle/10251/60708/HERR%C3%81EZ%20-%20Segmentaci%C3%B3n%20y%20clasificaci%C3%B3n%20de%20mallas%203D.pdf?sequence=2>. [Último acceso: 30 marzo 2023].
- [23] Meshlab, «Meshlab Noticias,» [En línea]. Available: <https://www.meshlab.net/>. [Último acceso: 1 abril 2023].
- [24] F. Rico Delgado, D. Antón García y J. E. Nieto, «UN NUEVO ENFOQUE EN LA GESTIÓN DE NUBES DE PUNTOS OBTENIDOS POR ESCÁNER Y FOTOGRAMETRÍA, PARA LA OPTIMIZACIÓN DE FORMATOS IMPRIMIBLES 3D,» 22 mayo 2018. [En línea]. Available: <https://idus.us.es/bitstream/handle/11441/98915/Un%20nuevo%20enfoque%20en%20la%20gesti%C3%B3n%20de%20nubes%20de%20puntos%20obtenidos%20por%20esc%C3%A1ner%20y%20fotogrametr%C3%ADa%20para%20la%20optimizaci%C3%B3n%20de%20formatos%20imprimibles%203D%20PONE>.
- [25] M. Alicia, «Las características de Cura, el slicer más extendido en el mercado de la impresión 3D,» 3Dnatives, 23 septiembre 2022. [En línea]. Available: <https://www.3dnatives.com/es/cura-laminador-para-impresion-3d-250320202/>.

Anexos

Código de Cámara Raspberry:

```
function varargout = RaspiScanner3D(varargin)

% RASPISCANNER3D MATLAB code for RaspiScanner3D.fig

%   RASPISCANNER3D, by itself, creates a new RASPISCANNER3D or raises the existing
%   singleton*.

%
%   H = RASPISCANNER3D returns the handle to a new RASPISCANNER3D or the handle to
%   the existing singleton*.

%
%   RASPISCANNER3D('CALLBACK',hObject,eventData,handles,...) calls the local
%   function named CALLBACK in RASPISCANNER3D.M with the given input arguments.

%
%   RASPISCANNER3D('Property','Value',...) creates a new RASPISCANNER3D or raises the
%   existing singleton*. Starting from the left, property value pairs are
%   applied to the GUI before RaspiScanner3D_OpeningFcn gets called. An
%   unrecognized property name or invalid value makes property application
%   stop. All inputs are passed to RaspiScanner3D_OpeningFcn via varargin.

%
%   *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%   instance to run (singleton)".

% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help RaspiScanner3D

% Last Modified by GUIDE v2.5 20-Jun-2022 12:13:10
```

```

% Begin initialization code - DO NOT EDIT

gui_Singleton = 1;

gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @RaspiScanner3D_OpeningFcn, ...
                  'gui_OutputFcn',  @RaspiScanner3D_OutputFcn, ...
                  'gui_LayoutFcn',  [] , ...
                  'gui_Callback',    []);

if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargin
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

% End initialization code - DO NOT EDIT


% --- Executes just before RaspiScanner3D is made visible.
function RaspiScanner3D_OpeningFcn(hObject, eventdata, handles, varargin)

% This function has no output args, see OutputFcn.

% hObject    handle to figure

% eventdata  reserved - to be defined in a future version of MATLAB

% handles     structure with handles and user data (see GUIDATA)

% varargin    command line arguments to RaspiScanner3D (see VARARGIN)

```

```

% Choose default command line output for RaspiScanner3D
handles.output = hObject;

handles.points = [];
handles.colors = [];

handles.filename = 'output';

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes RaspiScanner3D wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = RaspiScanner3D_OutputFcn(hObject, eventdata, handles)

% varargout cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

function filename_Callback(hObject, eventdata, handles)

```

```

handles.filename = get(hObject,'String');

guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.

function filename_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in extraction.

function extraction_Callback(hObject, eventdata, handles)

%---Descargar Archivos por comunicación FTP---%

%PONER EL CODIGO!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

%---Extracción de fotogramas de videos---%

disp('Iniciando...');

pause(1);

v = VideoReader('C:\Users\musik\Desktop\Pruebas nube de puntos\nuevo avance\Videos con Laser\Prueba1.avi');%lee video

numfr = v.NumFrames; %da el dato de cuantos fotogramas son en el video

%frrate = v.FrameRate;%FOTOGRAMAS POR SEGUNDO FPS

disp('Se realizará el proceso de extracción de imagenes con laser de la cam1, por favor espere...');

pause(1);

disp('Total de fotogramas');

disp(numfr);

pause(1);

for i=1:20    %ciclo para descargar fotogramas en la carpeta

    folder = 'C:\Users\musik\Desktop\Pruebas nube de puntos\nuevo avance\Fotogramas CAM1CL';

    frame = read(v,i);

```

```

    ImgName = strcat(int2str(i),'.png');
    imwrite(frame,fullfile(folder,ImgName))
end

disp('¡Se ha completado la extracción de imagenes con laser de la cam1!');
pause (1);

%---EXTRACCION DE FOTOGRAMAS SIN LASER(COLOR)-----
disp('Extraccion de fotogramas sin laser (COLOR)...');
v = VideoReader('C:\Users\musik\Desktop\Pruebas nube de puntos\nuevo avance\Videos sin Laser\Prueba2.avi');%lee video
numfr = v.NumFrames; %da el dato de cuantos fotogramas son en el video
frate = v.FrameRate;%FOTOGRAMAS POR SEGUNDO FPS
disp('Se realizará el proceso de extracción de imagenes sin laser de la cam1, por favor espere...');
disp('Total de fotogramas');
disp(numfr);
pause(1);
for i=1:20
    folder = 'C:\Users\musik\Desktop\Pruebas nube de puntos\nuevo avance\Fotogramas CAM1SL';
    frame = read(v,i);
    ImgName = strcat(int2str(i),'.png');
    imwrite(frame,fullfile(folder,ImgName))
end
disp('Ha terminado el proceso de extracción de imagenes')
pause (1);

function extraction_CreateFcn(hObject, eventdata, handles)
% hObject    handle to extraction (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

```

% --- Executes on button press in start.

function start_Callback(hObject, eventdata, handles)

v = VideoReader('C:\Users\musik\Desktop\Pruebas nube de puntos\nuevo avance\Videos sin Laser\Prueba2.avi');

numfr = v.NumFrames;

angulo = numfr/360;

disp('Angulo =');

disp(angulo);

disp('Recorte el fotograma');

foto = imread('C:\Users\musik\Desktop\Pruebas nube de puntos\nuevo avance\Fotogramas CAM1CL\1.png');

[J,rect] = imcrop(foto);

disp('Fotograma recortado');

imshow (J);

for i= 1:numfr

ruta = 'C:\Users\musik\Desktop\Pruebas nube de puntos\nuevo avance\Fotogramas CAM1CL'; %fotogramas con laser

archivo = [num2str(i) '.png'];

af = imread(fullfile(ruta,archivo)); % leer fotogramas con laser

crop = imcrop (af,[rect]);

folder = 'C:\Users\musik\Desktop\Pruebas nube de puntos\nuevo avance\FotogramasCL';

ImgName = strcat(int2str(i),'.png');

imwrite(crop,fullfile(folder,ImgName))

end

disp('Recorte con laser terminado');

pause (2)

```

disp('Ajustando el recorte a color');

pause (2)

for i= 1:numfr

ruta= 'C:\Users\musik\Desktop\Pruebas nube de puntos\nuevo avance\Fotogramas CAMISL';%fotogramas SIN laser

archivof= [num2str(i) '.png'];

af= imread(fullfile(ruta,archivof)); % leer fotogramas con laser

crop = imcrop (af,[rect]);

folder = 'C:\Users\musik\Desktop\Pruebas nube de puntos\nuevo avance\FotogramasSL';

ImgName = strcat(int2str(i),'.png');

imwrite(crop,fullfile(folder,ImgName))

end


disp("LISTO");

disp("Fotogramas recortados");

disp("Espere...");

pause (2)


disp('¡Se iniciará el proceso de nube de puntos!');

disp('Aplicando filtros...');

i= 1;

for currentAngle = 0:angulo:90

% i= 1;

%for i=1:4

ruta = 'C:\Users\musik\Desktop\Pruebas nube de puntos\nuevo avance\FotogramasCL';%fotogramas con laser recortados

ruta2 = 'C:\Users\musik\Desktop\Pruebas nube de puntos\nuevo avance\FotogramasSL';%fotogramas sin laser recortados

archivo = [num2str(i) '.png']; % para los fotogramas con laser

archivo2 = [num2str(i) '.png']; %para los fotogramas sin laser

a= imread(fullfile(ruta,archivo)); % leer fotogramas con laser

```

```
imgSL = imread(fullfile(ruta2,archivo2)); %leer fotografamas sin laser
```

```
a = fliplr(a); %Cambia la perspectiva de vista de la imagen
```

```
imgSL = fliplr(imgSL);
```

```
an=double(a);
```

```
ar=double(an(:, :, 1));
```

```
ag=double(an(:, :, 2));
```

```
ab=double(an(:, :, 3));
```

```
s= size(a);
```

```
ared= ar*0;
```

```
agreen= ag*0;
```

```
ablue= ab*0;
```

```
kernelgaussiano = [1 2 1;2 4 2;1 2 1]/16 ;
```

```
for k = 2:s(1)-1
```

```
    for j = 2:s(2)-1
```

```
        ventanar= ar(k-1:k+1, j-1:j+1);
```

```
        ventanag= ag(k-1:k+1, j-1:j+1);
```

```
        ventanab= ab(k-1:k+1, j-1:j+1);
```

```
        multir= ventanar.* kernelgaussiano;
```

```
        multig= ventanag.* kernelgaussiano;
```

```
        multib= ventanab.* kernelgaussiano;
```

```
        pix1= sum(sum(multir));
```

```
        pix2= sum(sum(multig));
```

```
        pix3= sum(sum(multib));
```

```
        ared(k,j)= pix1;
```

```
        agreen(k,j)= pix2;
```

```
        ablue(k,j)= pix3;
```

```

end

end

%-----FILTROS PARA PROCESAMIENTO DIGITAL DE IMAGENES

op1 = agreeen + ablue;%suma de freq. bajas plano verde y azul con la suma de estos dos plano se ve definida la figura
op2 = an - ared;%freq.altas en plano rojo se quitan las bajas rojas
%op3 = an - agreeen;%freq.altas en plano verde se quitan las bajas verdes
%op4 = an - ablue;%freq.altas en plano azul se quitan las bajas azules
%op3 y op4 generan mucho ruido alrededor del contorno de la linea
%laser al hacer operaciones con estas frecuencias altas, lo mejor es descartar las
%frecuencias altas de estos dos planos y solo trabajar con las altas
%rojas y las bajas del plano verde(agreeen) y azul(ablue).

op6 = op1 *3 ;

op66= op6 -20;% Se reduce el brillo de la imagen haciendo los pixeles claros mas oscuros para que no hayan pixeles
%claro al rededor de la linea laser generando ruido y distorcion

op5 = op2 *4;% Se aumenta el brillo y claridad de la imagen de las frecuencias altas rojas

op8 = op5 +20 ;

op7 = op2 + op66 +op8;% se suman las operciones anteriores para lograr una mejor definicion de la figur

gr= .299*op7(:,1) + .587*op7(:,2) + .114*op7(:,3);%convierte la imagen a escala de grises

level = graythresh (gr);

BN = imbinarize (gr, level);

imshow (BN);

[points,colors] = generatePoints(BN,imgSL,currentAngle);

handles.points = [handles.points; points];

handles.colors = [handles.colors; colors];

i=i+1 ;

end

%scatter3(handles.axes1,handles.points(:,1),handles.points(:,2),handles.points(:,3),5,'filled');

pcshow(handles.points,handles.colors,'Parent',handles.axes1,'MarkerSize',10);

```

```

handles.output = pointCloud(handles.points,'Color',handles.colors);
assignin('base','output',handles.output);
guidata(hObject, handles);

function start_CreateFcn(hObject, ~, handles)
% hObject    handle to start (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% --- Executes on button press in save.
function save_Callback(hObject, eventdata, handles)

pcwrite(handles.output,handles.filename,'PLYFormat','binary')
guidata(hObject, handles);

function save_CreateFcn(hObject, eventdata, handles)
% hObject    handle to save (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% --- Executes on button press in mesh.
function mesh_Callback(hObject, eventdata, handles)

ptCloud = pcread(strcat(handles.filename,'.ply'));%pcread('output.ply');
x = ptCloud.Location(:,1);
y = ptCloud.Location(:,2);
z = ptCloud.Location(:,3);
k = boundary(x,y,z);
trisurf(k,x,y,z,0.9,'Facecolor','w','EdgeColor','k','FaceAlpha',0.9);
axis equal

```

```
guidata(hObject, handles);
```

```
function mesh_CreateFcn(hObject, eventdata, handles)
```

```
% hObject handle to mesh (see GCBO)
```

```
% eventdata reserved - to be defined in a future version of MATLAB
```

```
% handles empty - handles not created until after all CreateFcns called
```

```
% --- Executes on button press in fill.
```

```
function fill_Callback(hObject, eventdata, handles)
```

```
ptCloud = pcread(strcat(handles.filename, '.ply')); %pcread('output.ply');
```

```
x = ptCloud.Location(:,1);
```

```
y = ptCloud.Location(:,2);
```

```
z = ptCloud.Location(:,3);
```

```
colors = ptCloud.Color;
```

```
k = boundary(x,y,z);
```

```
s = trisurf(k,x,y,z);
```

```
s.FaceVertexCData = colors;
```

```
s.EdgeColor = 'interp';
```

```
s.FaceColor = 'interp';
```

```
s.FaceAlpha = 0.9;
```

```
axis equal
```

```
guidata(hObject, handles);
```

```
function fill_CreateFcn(hObject, eventdata, handles)
```

```
% hObject handle to fill (see GCBO)
```

```
% eventdata reserved - to be defined in a future version of MATLAB
```

```
% handles empty - handles not created until after all CreateFcns called
```

% --- Executes during object creation, after setting all properties.

function axes1_CreateFcn(hObject, eventdata, handles)

hObject.XTick = [];

hObject.YTick = [];

hObject.ZTick = [];

hObject.XTickLabel = [];

hObject.YTickLabel = [];

hObject.ZTickLabel = [];

hObject.XColor = 'none';

hObject.YColor = 'none';

hObject.ZColor = 'none';

hObject.Color = 'white';

rotate3d(hObject,'on');