UNIVERSITAT DE BARCELONA

GRADO EN INGENIERÍA INFORMÁTICA

Trabajo de final de grado

Reconstrucción de objetos 3D mediante el uso de redes neuronales: del modelo real a su impresión física

---

Autor: Lluís Alcalà Català

Directora:	**Anna Puig Puig**
Realizado en:	**Departamento de Matemáticas e Informática**
Barcelona,	**10 de junio de 2024**

Abstract

This project provides an initial working base for users who need to develop an application capable of generating, from input data captured from reality (such as videos, images or LIDAR), a 3D object printable by most 3D printers.

This process is performed by an application developed with Python, supported by a neural network that reconstructs the scenes. From this reconstruction, a mesh smoothing is performed to be able to print the model on a conventional 3D printer. Throughout the execution, the user will be guided, offering a research environment with a variety of configurations available to refine and compare the results obtained with different configurations.

Resum

Aquest projecte proporciona una base de treball inicial per a usuaris que necessiten desenvolupar una aplicació capaç de generar, a partir de dades d'entrada captades de la realitat (com poden ser vídeos, imatges o LIDAR), un objecte 3D imprimible per la majoria de les impressores 3D.

Aquest procés es realitza mitjançant una aplicació desenvolupada amb Python, sostinguda per una xarxa neuronal que reconstrueix les escenes. A partir d'aquesta reconstrucció es realitza un suavitzat del mallat per poder imprimir el model en una impressora 3D convencional. Durant tota l'execució, es guiarà a l'usuari, oferint-li un entorn de recerca amb una varietat de configuracions disponibles per refinar i comparar resultats obtinguts amb diferents configuracions.

Resumen

Este proyecto proporciona una base de trabajo inicial para usuarios que necesitan desarrollar una aplicación capaz de generar, a partir de datos de entrada captados de la realidad (como pueden ser vídeos, imágenes o LIDAR), un objeto 3D imprimible por la mayoría de las impresoras 3D.

Este proceso se realiza mediante una aplicación desarrollada con Python, respaldada por una red neuronal que reconstruye las escenas. A partir de esta reconstrucción se realiza un suavizado del mallado para poder imprimir el modelo en una impresora 3D convencional. Durante toda la ejecución, se guiará al usuario, ofreciéndole un entorno de investigación con una variedad de configuraciones disponibles para refinar y comparar los resultados obtenidos con diferentes configuraciones.

Agradecimientos

En primer lugar, quiero expresar mi más sincero agradecimiento a mi tutora Anna Puig, quien me ha brindado una gran ayuda durante todo el proceso de desarrollo y documentación de este proyecto. Su dedicación y entusiasmo en la enseñanza no solo facilitan el aprendizaje, sino que también crean un ambiente de trabajo agradable y productivo para todos los estudiantes.

Por último, deseo agradecer a mis padres y a mi pareja por todo el apoyo recibido durante estos años en la universidad. Sin su constante respaldo y cariño, no estaría donde estoy hoy. ¡Muchas gracias de todo corazón!

Índice general

Capítulo 1

Introducción

En este capítulo introductorio, se expondrán de manera concisa el entorno y los fundamentos que rodean al proyecto. Además, se explorarán las motivaciones de emprender este proyecto, así como los objetivos que se persiguen con su realización. Finalmente, se describirá la estructura principal que guiará el desarrollo del ensayo, ofreciendo una vista panorámica de los temas que se abordarán en los siguientes capítulos.

1.1. Contexto y Motivación

Contexto

En los últimos años, la reconstrucción 3D ha experimentado un notable avance gracias a la integración de tecnologías emergentes como las redes neuronales. Esta evolución tecnológica ha permitido superar las limitaciones tradicionales de los métodos de escaneo y modelado manual, proporcionando herramientas más precisas, eficientes y accesibles para la creación de modelos tridimensionales. Desde la preservación del patrimonio cultural hasta la personalización de productos, las aplicaciones de la reconstrucción 3D abarcan una amplia variedad de sectores, incluyendo la ingeniería, la medicina, el entretenimiento y la manufactura, ejemplos de estos pueden encontrarse en el paper [5].

La aparición de redes neuronales ha revolucionado el campo, ofreciendo soluciones innovadoras que permiten capturar detalles finos y complejas geometrías de manera automatizada. Al combinar estas capacidades con la tecnología de impresión 3D, se abre un mundo de posibilidades para materializar modelos digitales en objetos físicos. Sin embargo, esta sofisticación también puede dificultar el acceso inicial a este ámbito, debido al elevado número de parámetros que se han de tener en cuenta en la configuración de la red neuronal y sus efectos en la reconstrucción final.

Motivación

El objetivo principal de este trabajo es explorar y desarrollar métodos para la reconstrucción de objetos 3D obtenidos de la realidad, utilizando redes neuronales, facilitando el acceso a estas tecnologías a una amplia población y agilizando los tiempos de desarrollo para llevar estos modelos digitales a su impresión física. Este enfoque está motivado por varias razones clave:

1. **Estudio y análisis de redes neuronales en el ámbito de la reconstrucción 3D:** Las redes neuronales han demostrado una capacidad excepcional para procesar grandes cantidades de datos y aprender características complejas, lo que las convierte en herramientas ideales para la reconstrucción 3D. Su uso, no obstante, no es una tarea fácil, ya que supone configurar un conjunto de parámetros, incluso en el modelo ya entrenado. En este proyecto se pretende facilitar su uso y exploración para poder probar y analizar el efecto de estos parámetros en la reconstrucción 3D.

2. **Eficiencia y Accesibilidad:** Tradicionalmente, la creación de modelos 3D precisos ha sido un proceso laborioso y costoso. La automatización de este proceso mediante redes neuronales no solo reduce el tiempo y los costos asociados, sino que también hace que la tecnología sea más accesible para una gama más amplia de usuarios y aplicaciones. En este proyecto se pretende agilizar el proceso mediante una interfaz fácil de usar, pero que a la vez incluya las suficientes posibilidades de configuración de las diferentes etapas.

3. **Innovación en Manufactura:** La integración de la reconstrucción 3D con la impresión 3D permite la producción de objetos físicos directamente a partir de modelos digitales, lo que representa una revolución en la manufactura. En este proyecto se pretende hacer una primera aproximación a la personalización masiva, la creación rápida de prototipos y la producción de objetos complejos que serían imposibles de fabricar mediante métodos tradicionales.

4. **Educación y Formación:** Este proyecto también tiene un fuerte componente educativo, proporcionando a los estudiantes, profesores y profesionales en diversos campos, las herramientas y el conocimiento necesario para aplicar estas tecnologías en sus respectivos campos. Así, en este proyecto puede llevar a fomentar la innovación y el aprendizaje continuo en un área de rápido crecimiento y evolución.

En resumen, la reconstrucción de objetos 3D mediante el uso de redes neuronales no solo representa un avance tecnológico significativo, sino que también abre nuevas posibilidades en diversas áreas, contribuyendo al desarrollo de soluciones innovadoras y prácticas que tienen un impacto real en la sociedad. Este proyecto busca explorar estas oportunidades, desarrollando métodos eficientes y accesibles para la creación de modelos 3D precisos y su materialización a través de la impresión física.

1.2. Objetivos Principales

El objetivo principal de este proyecto, es testear la viabilidad y eficacia de las tecnologías basadas en redes neuronales, específicamente para poder reproducir un objeto captado de la realidad mediante una impresión 3D. Para ello, se plantean una serie de objetivos principales.

En primer lugar, se llevará a cabo una revisión exhaustiva del **estado del arte** en tecnologías de generación de modelos 3D, centrándose en aquellas que utilizan redes neuronales o aprendizaje automático. Para finalmente profundizar en su funcionamiento y verificar las capacidades de estas, creando para ello un entorno de investigación.

Para la creación del **entorno de investigación**, se revisarán todas las técnicas y herramientas que puedan influir en la calidad del modelo 3D final, ofreciendo alternativas para diversos tipos de entornos de entrada.

Se llevarán a cabo pruebas utilizando las tecnologías seleccionadas, con el fin de **analizar los resultados** obtenidos en términos de calidad, precisión y aptitud para la impresión en 3D, además de identificando ventajas, limitaciones y áreas de mejora.

Finalmente, se proporcionarán **recomendaciones prácticas** para el uso del aplicativo desarrollado en la generación de mallas 3D imprimibles y se resumirán las conclusiones alcanzadas en relación con el objetivo principal del trabajo. A través de los resultados obtenidos, se busca contribuir al conocimiento en el campo de la generación de modelos tridimensionales mediante el uso de técnicas basadas en redes neuronales, proporcionando información relevante para su aplicación en la impresión 3D.

1.3. Objetivos Específicos

Los objetivos principales anteriores se desglosan en los siguientes objetivos específicos:

- Revisión del estado del arte, el principal objetivo de esta fase es identificar y listar las diferentes tecnologías disponibles para alcanzar el objetivo propuesto, y seleccionar aquellas que mejor se adecuen a los requisitos del proyecto.
- Realizar pruebas con las diferentes tecnologías encontradas y seleccionar la más adecuada a la finalidad del proyecto, según su rendimiento, complejidad y requerimientos.
- Investigar y seleccionar una aplicación para el refinado final de la malla obtenida.

- Diseñar y desarrollar una aplicación específica para las tecnologías seleccionadas, con el propósito de agilizar el proceso de pruebas y análisis de resultados.
- Realizar pruebas exhaustivas del aplicativo desarrollado con el fin de optimizar su rendimiento y determinar la configuración óptima para alcanzar los mejores resultados posibles, además de obtener una serie de recomendaciones para el uso de la red neuronal escogida.

1.3.1. Planificación

El proyecto a grandes rasgos se estructura en 3 etapas principales, distribuidas por los siguientes meses, ver Figura 1.1:

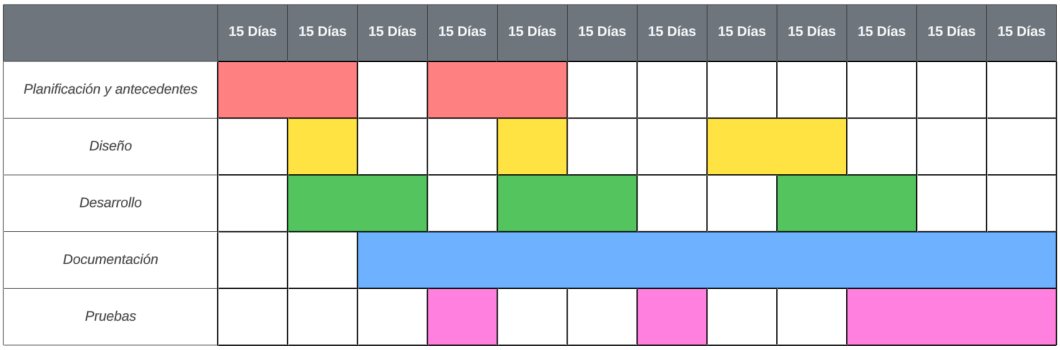


Figura 1.1: Diagrama de Gantt sobre la planificación del proyecto.

Estas etapas no van a realizarse en formato de bloques, sino que se aplicara lo aprendido en la universidad, utilizando la metodología `Scrum`. A grandes rasgos el que se dedicara a cada bloque será el siguiente:

1. **Planificación y antecedentes:** Durante dos meses, se planteará el alcance y los objetivos a los que se quiere llegar en el proyecto. Una vez cerrado el contenido, se se procederá a la investigación de los antecedentes y tecnologías existentes que sean útiles para alcanzar los objetivos preestablecidos. Finalmente, estudiadas todas las vías de actuación, se elegirán las características que debe tener el sistema para poder cumplir los requisitos establecidos.
2. **Diseño y Desarrollo:** Establecidos los objetivos, y las tecnologías adecuadas para llegar a los requerimientos, se procederá al desarrollo del software, este desarrollo tendrá una duración de 3 meses.
3. **Documentación y pruebas:** Una vez finalizado el desarrollo del proyecto, se contará con 1 mes, para su documentación y pruebas de rendimiento del proyecto realizado.

1.3.2. Organización de la memoria

La estructura de la memoria se desglosa en los siguientes capítulos:

- **Introducción:** El capítulo de la introducción tiene como propósito contextualizar el proyecto con breves descripciones de los objetivos principales de este.
- **Análisis del problema:** En el capítulo de Análisis se definen las notaciones básicas (o vocabulario) y los requerimientos del proyecto.
- **Análisis de Antecedentes:** Aquí se estudian posibles soluciones ya elaboradas a los problemas planteados en el capítulo anterior, ya sea desde el punto de vista de los métodos necesarios como de aplicaciones similares.
- **Diseño de la aplicación:** En el capítulo de Diseño se describirá el diseño gráfico y la arquitectura del software, así como la descripción de la solución propuesta en cada una de las etapas del proceso de captación, reconstrucción e impresión.
- **Simulaciones y resultados:** En este capítulo se mostrarán los resultados del proyecto desarrollado.
- **Conclusiones y trabajo futuro:** En este último capítulo se detallan las conclusiones extraídas del proyecto y sus líneas futuras.
- **Apéndice A: Manual Técnico:** En el Manual Técnico se describirán de forma detallada los pasos para la instalación y el buen funcionamiento del aplicativo desarrollado.
- **Apéndice B: Manual de Usuario:** Este apéndice está diseñado para introducir en detalle al usuario en el flujo de trabajo del programa, además de aportar algunas recomendaciones para su ejecución.

Capítulo 2

Análisis del problema

En este capítulo se describen los requerimientos del proyecto, desglosados según las diferentes fases necesarias para poder captar los objetos de la realidad y replicarlos posteriormente mediante una impresora 3D. La secuencia de pasos necesarios, desde la captación de imágenes o escaneo de la escena hasta la impresión del objeto 3D, implica la transformación y el procesamiento de la información. En este capítulo se detalla la secuencia de estas etapas, así como las entradas y salidas de datos de cada una de las fases y las funcionalidades permitidas.

2.1. Requerimientos del proyecto

Esta sección tiene como objetivo definir de manera detallada los requerimientos del proyecto, proporcionando una guía clara y concisa para el desarrollo y la implementación exitosa del mismo. Estos requerimientos establecen las bases fundamentales sobre las cuales se construirá el sistema o producto, delineando las funciones, características y restricciones que deben ser consideradas durante todas las etapas del proceso.

Previamente a los requerimientos, cabe destacar que el presente proyecto está enfocado a aquellas personas con ciertos conocimientos básicos en inteligencia artificial que desean desarrollar un producto basado en el escaneo de objetos 3D. Este usuario busca un software que le facilite el proceso de investigación sobre una tecnología específica, para así de este modo simplificar y optimizar el desarrollo de su producto.

Requerimientos funcionales:

Como se puede observar en la Figura 2.1, todos los requerimientos están fuertemente relacionados.

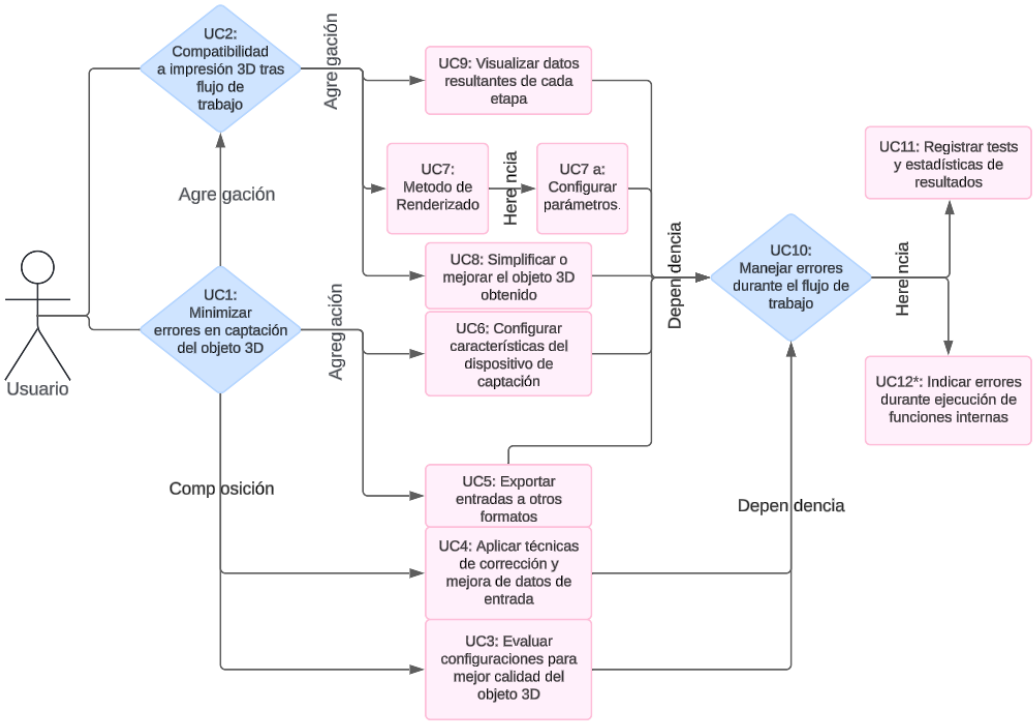


Figura 2.1: Diagrama casos de uso del usuario.

A continuación se detallan los requerimientos funcionales del proyecto mediante la definición de los Casos de Uso:

- **UC1:** El usuario debe disponer de una solución, ya sea física o a través de software, que minimice los errores en la captación del objeto 3D.
- **UC2:** Dado un conjunto de imágenes o un vídeo de entrada, el cual contiene diferentes vistas de un objeto 3D, el usuario debe ser capaz de llegar al final del flujo de trabajo con objeto que sea compatible a la impresión 3D.
- **UC3:** Dadas una entrada de imágenes o vídeo, el usuario debe ser capaz de generar diferentes salidas y poder evaluar que configuración ha generado mejores resultados en la calidad del objeto 3D final.
- **UC4:** Del conjunto de datos de entrada, como son las imágenes o un vídeo, el usuario debe poder aplicar técnicas de corrección y mejora de estos.
- **UC5:** El usuario debe ser capaz de poder exportar las entradas a otros formatos, con la finalidad de investigar el resultado que le reporta más beneficios según sus necesidades, en cuanto a tiempo, memoria y suavizado de la escena.
- **UC6:** Dependiendo del sistema que se elija para la captación del objeto 3D, el usuario debe poder configurar las características específicas del dispositivo.

Ejemplo en el caso de la cámara el sistema de obturación.

- **UC7:** El usuario debe poder seleccionar el método de renderización de la escena según sus necesidades.
- **UC7a:** El usuario debe poder configurar los diferentes parámetros de la herramienta de renderización seleccionada, con la finalidad de obtener los mejores resultados según sus necesidades.
- **UC8:** El usuario debe poder simplificar o mejorar el objeto 3D obtenido al final.
- **UC9:** El usuario debe poder visualizar los datos resultantes de cada etapa.
- **UC10:** En cualquier fase del flujo de trabajo, al producirse un error de cualquier tipo, introducido por el usuario, el programa debe ser capaz de manejar el error y darle al usuario indicaciones del problema.
- **UC11:** El usuario debe poder llevar un registro de los tests que ha realizado y poder recuperar las estadísticas de los resultados.
- **UC12*:** Si durante la ejecución de cualquier función interna del programa, surge un error, el sistema debe indicar al usuario la salida del error.

Requerimientos no funcionales:

En cuanto a los requerimientos no funcionales, destacan especialmente aquellos relacionados con los componentes físicos de la aplicación, como los medios de captura y los dispositivos de impresión 3D. Sin embargo, también es crucial considerar la portabilidad, flexibilidad y robustez del producto final. A continuación, se detallan todos estos aspectos:

- Se debe disponer de un medio de captura fácil, sencillo que garantice una cierta calidad en los datos capturados, estos deben tener una alta resolución y ser consistentes en el espacio.
- El usuario debe poder refinar los datos obtenidos durante la ejecución del pipeline.
- Se debe disponer de un medio de impresión con unos mínimos requerimientos, entre estos tamaños máximo de impresión de 220 x 220 x 250 mm.
- El programa debe ser robusto, a prueba de errores, ya sean intencionados o no intencionados.
- El programa debe aprovechar al máximo los recursos del equipo que lo ejecuta.
- En la medida de lo posible, el programa debe de poderse ejecutar en el máximo posible de equipos (Gama alta, media o baja).

- Los métodos y técnicas requeridas por el proyecto deben ser de una complejidad media o baja de implementar u obtener.
- El proyecto debe minimizar los costos asociados para el usuario.

2.2. Visión general de las etapas a seguir

El flujo de ejecución del programa puede variar en gran medida según las tecnologías seleccionadas. Estas tecnologías pueden requerir una amplia gama de datos, los cuales deben ser manipulados e interconectados de manera adecuada para generar un objeto 3D válido. No obstante, en esta sección se describe una secuencia de etapas básicas (o pipeline), detalladas en la Figura 2.2, por las que fluirán los datos capturas hasta la obtención del objeto impreso final.

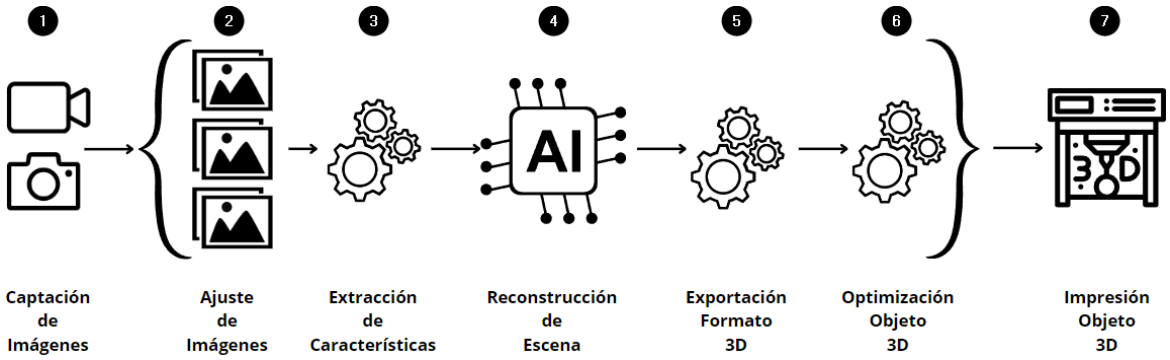


Figura 2.2: Pipeline del Proyecto

Así pues, las etapas identificadas son las siguientes:

1. **Captación de imágenes o escena:** Esta primera fase consistirá en recopilar toda la información que sea posible de la escena, ya sea con el uso de tecnologías sencillas como una simple cámara o utilizando sistemas más complejos como son los escáneres LiDAR, los cuales se describirán en el próximo capítulo.
2. **Ajuste o tratamiento de las imágenes:** Posteriormente a la captación de imágenes, surge esta fase que es crucial para la corrección de los posibles defectos introducidos en la etapa anterior a causa de errores del propio sistema de captación o incluso errores naturales, como las variaciones en la luminosidad del ambiente. Para ello, es esencial poder ajustar de manera adecuada los parámetros de las imágenes obtenidas y descartar aquellas que puedan aportar más ruido del necesario en la reconstrucción final del objeto.
3. **Extracción de características:** Una vez minimizados los errores que pueden obstaculizar la adecuada reconstrucción del objeto, empieza la fase de extracción de características y reconocimiento del entorno. Para ello, mediante el uso

de algoritmos de visión artificial, se extraen los descriptores de la imagen, así como sus puntos clave, para su posterior reconstrucción en el siguiente paso.

4. **Reconstrucción de la escena:** Recogidos y sintetizados todos los datos que describen la escena, se pasa la información a la red neuronal, donde está reconstruye la escena según la calidad de los datos de entrada.

5. **Exportación a un formato de impresión 3D:** Finalizada la etapa de reconstrucción, se procede a la exportación de los datos a un tipo de representación que la impresora 3D sea capaz de interpretar e imprimir, en la actualidad existen diferentes tipos de formatos, entre ellos STL, 3MF, OBJ etc.

6. **Optimización del objeto 3D:** Finalmente, previo a la impresión 3D, se presenta la etapa de optimización del objeto 3D. Dado que la ejecución del pipeline implica el paso por diversas fases, esto puede provocar la acumulación y generación de ruido en la superficie del objeto. Como resultado, el objeto 3D puede no tener la mejor calidad ni los mejores acabados. Por ello, es imprescindible contar con una herramienta capaz de gestionar y corregir los errores de la manera más eficiente posible.

7. **Impresión 3D:** Para concluir, aunque la impresión 3D, no formara parte del propio pipeline del aplicativo que se desarrollara, esta es la última etapa en la que se pasa del modelo virtual renderizado y optimizado, al modelo físico 3D.

A continuación, en las siguientes secciones, se expondrán los requerimientos funcionales de cada una de las etapas, así como de sus entradas, las condiciones de partida, las funcionalidades, los parámetros variables y las salidas esperadas.

2.3. Captación de la escena

En esta fase en la que el usuario debe captar el máximo de detalles sobre la escena y el objeto 3D que va a reproducir. Con el uso de algún sistema o herramienta adecuada, como son las cámaras, escáneres 3D o el uso de tecnología LiDAR.

Para ello, el usuario debe optar a algún medio sencillo de utilizar que le permita minimizar cualquier error y capturar con alta resolución los detalles.

El sistema utilizado para la captación del entorno, determinará el tipo de dato inicial, decisión determinante para las futuras etapas. Aunque a primera vista pueda parecer una etapa sencilla, en esta pueden aparecer numerosos problemas, influenciados por numerosos factores, como son la variación de iluminación en la escena, la posición de la cámara para obtener todos los ángulos del objeto 3D, etc., los cuales puedan entorpecer la obtención de un resultado óptimo.

Algunos de los sistemas de captación se ven afectados por el movimiento que realizas a la hora de escanear la escena, a continuación se describen en más profundidad.

Problemas producidos por el movimiento del sistema de captación:

En el caso de las cámaras, los principales problemas que surgen a la hora de la extracción de características (puntos claves y coincidencias entre las imágenes) son producidos por los movimientos:

- **Movimientos de translación:** En el contexto de la captura de imágenes y la recreación 3D, un movimiento de translación se refiere a cuando la cámara o el objeto en movimiento se mueve en línea recta, ya sea horizontalmente (de izquierda a derecha o viceversa), verticalmente (hacia arriba o hacia abajo), o en profundidad (hacia adelante o hacia atrás), Figura 2.3. Este tipo de movimiento puede alterar la perspectiva de la escena y la posición aparente de los objetos en relación con el observador, lo que influye en la forma en que se capturan y procesan las imágenes para la reconstrucción tridimensional.

Figura 2.3: Ejemplo de movimientos de translación. Fuente imagen Web

- **Movimientos de rotación:** En el contexto de la captura de imágenes y la recreación 3D, un movimiento de rotación se refiere a cuando la cámara o el objeto en movimiento cambia su orientación sin cambiar su posición en términos de desplazamiento espacial, Figura 2.4. Por ejemplo, si una cámara gira sobre su eje vertical, se produce un movimiento de rotación. Este tipo de movimiento puede afectar la perspectiva y la apariencia visual de la escena capturada, lo que influye en la forma en que se procesan las imágenes para la reconstrucción tridimensional.

Figura 2.4: Ejemplo de movimientos de rotación. Fuente imagen Web

- **Movimientos de Zoom:** Cambios en la distancia entre la cámara y los objetos en la escena, Figura 2.5. Los movimientos de Zoom pueden tener un impacto significativo en la representación tridimensional de la escena, ya que afectan la percepción del tamaño y la distancia entre los objetos. En la reconstrucción 3D, es importante tener en cuenta los cambios de escala para evitar una representación inexacta de la escena capturada.

Figura 2.5: Representación del Zoom. Fuente imagen Web

Por este motivo, es necesario que el proyecto cuente con un sistema que ayude al usuario a minimizar el error, evitando la intervención humana, dado que esta desembocara de forma inevitable a la generación de ruido y malformaciones en la representación de la escena 3D.

Otra consideración importante que se debe tener en cuenta, es el tipo de formato inicial, el cual será producido por el sistema de captación, además del tipo de formato de salida, el cual deberá ser seleccionado por el usuario según le convenga. En esta etapa se seleccionara un formato de entrada y uno de salida.

A continuación se detallan los formatos de los archivos más comunes que existen para las diferentes entradas y salidas que se pueden dar según el sistema de captación utilizado. Estos formatos deberán ser soportados por la aplicación final.

Entrada o salida de imágenes

- **JPG/JPEG:** Joint Photographic Experts Group, Es uno de los formatos más comunes para imágenes fotográficas. Utiliza compresión con pérdida, lo que reduce significativamente el tamaño del archivo a costa de una cierta disminución en la calidad de la imagen. Esto puede acelerar la renderización de la escena, pero reduciendo la calidad de esta debido a la compresión.
- **PNG:** Portable Network Graphics, Utiliza compresión sin pérdida, lo que significa que no se pierde calidad al comprimir la imagen. Admite transparencia y permite una alta calidad de imagen. Sin embargo, los archivos PNG suelen ocupar más espacio que los JPEG.
- **TIFF:** Tagged Image File Format, Admite compresión sin pérdida y es altamente flexible, permitiendo almacenar imágenes con múltiples capas y resoluciones. Es ideal para edición y almacenamiento de imágenes de alta calidad.
- **BMP:** Bitmap, Es un formato de imagen de mapa de bits desarrollado por

Microsoft. No utiliza compresión, lo que significa que las imágenes BMP pueden ocupar mucho espacio en disco, pero retienen la calidad completa de la imagen. Este formato es compatible con casi todas las aplicaciones y sistemas operativos.

Entrada o salida de Vídeos

- **MP4:** MPEG-4, es un formato muy usado debido a su capacidad para comprimir archivos sin perder mucha calidad, lo que lo hace ideal para ser usado como base en este proyecto. Es compatible con una amplia gama de herramientas.
- **AVI:** Audio vídeo Interleave, utiliza compresión sin pérdida, lo que significa que no se pierde calidad al comprimir la imagen. Admite transparencia y permite una alta calidad de imagen.
- **MOV:** QuickTime Movie, es un formato de archivo contenedor desarrollado por Apple para almacenar datos multimedia. MOV puede contener múltiples pistas de audio y vídeo, así como datos de subtítulos y metadatos.

Entrada o Salida de formatos complejos

En el ámbito de las tecnologías más avanzadas, como son por ejemplo los sistemas LiDAR, nos podemos encontrar con los siguientes formatos de archivos como entrada o salida:

- **OBJ:** Wavefront OBJ. Es un formato de archivo comúnmente utilizado para representar modelos tridimensionales superficiales basados en mallas de triángulos. Este formato puede ser utilizado para almacenar modelos generados a partir de datos capturados por escáneres 3D y cámaras RGB-D.
- **PLY:** Polygon File Format. Otro formato común para datos de nube de puntos y modelos 3D. PLY es muy utilizado en aplicaciones de gráficos por computadora y visualización tridimensional debido a su capacidad para almacenar información detallada sobre la geometría, el color y otras propiedades de los objetos escaneados. Este cuenta con la ventaja que se puede almacenar de forma binaria, por lo tanto, se puede reducir drásticamente el tamaño de los archivos.
- **LAS:** LIDAR LASer. Este es un formato específico utilizado para datos captados con tecnología LiDAR. Es un formato binario diseñado para almacenar información que incluye coordenadas, XYZ, intensidad del retorno, entre otra información relevante.

Para la elección del formato hay que tener en cuenta algunos factores, entre ellos, la compatibilidad con otros sistemas, el peso de los archivos, la calidad y la dificultad para trabajar con estos.

2.4. Tratamiento de imágenes

En esta fase el usuario debe poder corregir cualquier error producido durante la etapa anterior, esto mediante el uso de varias técnicas que se puedan adaptar a cualquier escena o error cometido.

Dependiendo del formato inicial, tanto de entrada, como de salida, esta fase puede llegar a ser más o menos compleja, dado que si se parte de un tipo más común de archivos, como son los de imágenes y los vídeos, la modificación y alteración de estos requerirá de un hardware y software menos pesado, agilizando de este modo la ejecución del pipeline y facilitando la accesible para más usuarios.

En el caso de seleccionar como salida de la fase anterior un vídeo o imágenes, pueden aplicarse distintas técnicas para minimizar los errores de la etapa anterior, pudiendo así, mejorar los resultados de la reconstrucción de la siguiente etapa:

> **Requerimiento 1**. *En el caso del uso de vídeo como archivo de salida en la etapa anterior, puede ser conveniente primero* ***extraer sus frames****, dado que esto agiliza y facilita la tarea de corrección. Esto esta contemplado dentro del UC3 de la sección 2.1*

A continuación, se detallan los errores producidos por el sistema de captación de imágenes por sí mismo, por efectos de iluminación y por falta de información.

Errores por el sistema de captación de imágenes o vídeo:

Es común que los sistemas incorporen pequeños defectos en los archivos dada su naturaleza electrónica, además de la propia compresión que se aplique de forma automática. Para corregir estos defectos existe la siguiente solución.

Se aplican máscaras mediante el uso de una matriz 2D, con ciertos valores estadísticos, para suavizar un grupo de imágenes o un archivo vídeo. Esto produce una imagen más suave y con el ruido difuminado, lo que puede facilitar la identificación de características y mejorando, así, la precisión de la reconstrucción.

Errores por la iluminación de la escena:

Dado el caso en que la escena presente rayos de luz intensos, o por falta de estos, puede requerirse de una mejora de los aspectos básicos de la imagen o el vídeo. Es por esto que el usuario debe poder acceder a funciones que le permitan ajustar el brillo, el contraste o el color de las imágenes, facilitando de este modo la tarea de identificación de características.

Errores por falta de información:

Puede presentarse la circunstancia en que el entorno del objeto que se está escaneando no aporte suficientes referencias, dificultando de este modo la extracción de características. Es por esto que el usuario de alguna forma sea capaz de aportar

ruido adicional a las imágenes, para ofrecer más puntos de apoyo o referencias al algoritmo de extracción de características, mejorando de esta forma la reconstrucción de la escena.

2.5. Extracción de Características

Si las transformaciones de la etapa anterior se llevan a cabo de forma adecuada esta fase debería ser relativamente sencilla, dado que se trataría de recolectar de las diferentes imágenes de entrada los datos más relevantes de la escena (puntos claves y coincidencias entre imágenes), esto con la finalidad de compactarlos y darlos como entrada a la herramienta de reconstrucción 3D.

El usuario debe poder configurar los diferentes parámetros disponibles en la herramienta de extracción de características, como son `COLMAP`, `Record3D`, etc. De forma adicional, debería tener la posibilidad de ver los resultados de la extracción, dado que de esta forma podría verificar la calidad de los resultados, y de no ser así repetir la extracción con otra configuración más adecuada o incluso volver a la etapa anterior para rectificar aún más los datos iniciales (imágenes/vídeo).

Los datos de salida de esta etapa dependen totalmente de la herramienta que se seleccione para la reconstrucción de la escena de la siguiente etapa. Estos pueden ser desde un archivo de texto estructurado con la información espacial de la escena, como son las nubes de puntos o archivos de texto con información relevante (puntos clave, matrices de transformación) sobre las imágenes de entrada.

Requerimiento 2. *Tal y como se define en el apartado de requerimientos funcionales de la sección 2.1, en el caso de uso UC4, el usuario debe tener herramientas que le permitan mejorar y corregir los posibles errores.*

2.6. Reconstrucción 3D

En esta etapa, a partir de los datos captados y transformados anteriormente, se pasa a la reconstrucción de la escena 3D virtual. Para ello, el usuario debe poder elegir la estrategia inteligente que desee para poder realizar la reconstrucción 3D. En el capítulo 3 se analizarán varias redes neuronales que son capaces de reconstruir modelos 3D. Como veremos, cada una de ellas requieren de una definición de valores de parámetros distintos, además de funcionar bajo una entrada y salida concreta. Así pues, en esta etapa el usuario ha de poder escoger la red neuronal a usar, variar los valores principales que permiten regular la reconstrucción y el formato de salida. En el capítulo 3 también se analizarán los formatos de salida (nubes de puntos, mallado 3D, modelos de superficies implícitas y modelos de volumen) junto con sus ventajas e inconvenientes para ser utilizados posteriormente para la impresión 3D.

Requerimiento 3. *Como se expresa en el capítulo de requerimientos 2.1, concretamente el caso de uso US7, el usuario debe poder seleccionar diferentes opciones de renderización.*

2.7. Exportación a formato compatible para la impresión 3D

Dependiendo de la herramienta seleccionada en la etapa anterior, la salida generada por esta, puede no sea compatible con la impresión 3D, (como son las nubes de puntos, o representaciones basadas en volúmenes como son los Voxels ver en Capítulo 3) de ser este el caso, debe implementarse alguna función que sea capaz de realizar dicha tarea, además de que el usuario pueda configurar la exportación y visualizar los datos de salida para comprobar la calidad de estos.

Los programas encargados de traducir la información de un objeto 3D virtual al lenguaje que entiende la impresora 3D (G-Code), son denominados laminadores, estos típicamente son compatibles y pueden realizar la tarea de traducción de formatos como .obj, .stl o .3mf, por lo tanto, se requiere de un programa que pueda exportar a dichos formatos.

Requerimiento 4. *En la sección de requerimientos 2.1, se expresa claramente en el UC2, la necesidad del usuario de exportar los datos finales a un formato compatible para la impresión 3D.*

2.8. Suavizado de puntos y mallas

Una vez obtenido de la tapa anterior un archivo compatible con la impresión 3D, el usuario debe poder mejorar la calidad del objeto 3D, ya sea simplificándolo mediante operaciones básicas como son eliminar caras duplicadas, vértices no referenciados o realizar operaciones más complejas como son el suavizado de mallas o rellenado de huecos. En esta etapa la entrada y la salida de los datos será del mismo tipo, dado que el último paso es la impresión 3D.

Requerimiento 5. *En referencia al requerimiento UC8, de la sección 2.1, al usuario se le ofrecerá la opción de simplificar o mejorar el objeto 3D obtenido de la renderización.*

Capítulo 3

Análisis de Antecedentes

En este capítulo, se presentarán las diferentes soluciones existentes en la actualidad, para consolidar los distintos requerimientos del proyecto. Analizando de este modo los pros y contras de cada solución, en función de las necesidades del proyecto.

Antes de realizar el análisis de las posibles soluciones en las diferentes etapas definidas anteriormente, se van a describir los distintos modelos para la representación de datos 3D, los cuales permitirán captar la realidad de forma fácil y precisa, para posteriormente ser tratados e impresos en 3D. Así pues, en la primera sección de este capítulo se analizan los modelos 3D, con sus ventajas e inconvenientes.

Una vez escogido el modelo a usar, en las secciones siguientes, se procederá al análisis de las posibles soluciones a los problemas planteados en las distintas etapas del pipeline definido.

3.1. Modelos de representación 3D

A continuación se detallan los principales modelos 3D comúnmente utilizados en las aplicaciones gráficas. Para cada uno se detallan sus ventajas e inconvenientes en relación con el presente proyecto.

1. **Nube de puntos:** Una nube de puntos es una colección de puntos tridimensionales en el espacio, donde cada punto está definido por sus coordenadas x, y, z (ver Figura 3.1). Estos puntos representan una muestra discreta de la superficie de un objeto o de una escena en tres dimensiones.

   Cada punto en la nube de puntos puede contener información adicional además de sus coordenadas espaciales. Por ejemplo, puede incluir información sobre la intensidad del color, la intensidad del retorno del láser (en el caso de escaneo láser), la reflectividad, la normal de la superficie en ese punto, o cualquier otro atributo relevante para la aplicación específica.

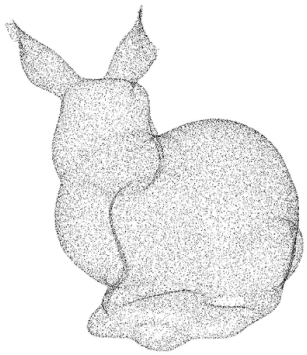


Figura 3.1: Representación de la nube de puntos. Fuente de la imagen: `https://www.researchgate.net/figure/Example-point-clouds-for-Stanford-Bunny_fig3_328350850`

A continuación se exponen las ventajas que conllevaría el uso de esta representación:

a) **Captura precisa de la forma:** Las nubes de puntos pueden capturar con precisión la forma y la geometría de objetos tridimensionales, incluyendo detalles finos y texturas.

b) **Versatilidad en la captura:** Se pueden generar nubes de puntos a partir de una variedad de métodos de captura, como escáneres láser, cámaras 3D o técnicas de fotogrametría, lo que permite adaptarse a diferentes tipos de objetos y entornos.

c) **Representación detallada:** Las nubes de puntos pueden contener una gran cantidad de información, incluyendo coordenadas espaciales, color, intensidad de reflectividad, entre otros, lo que proporciona una representación detallada del objeto.

d) **Flexibilidad en el procesamiento:** Las nubes de puntos son compatibles con una amplia gama de software de modelado y visualización 3D, lo que facilita su procesamiento, análisis y manipulación para diversas aplicaciones.

Todo esto teniendo en cuenta las siguientes desventajas:

a) **Volumen de datos:** Las nubes de puntos pueden generar grandes volúmenes de datos, especialmente para objetos complejos o escenas detalladas, lo que puede requerir recursos computacionales significativos para su procesamiento y almacenamiento.

b) **Requerimientos de hardware:** Para capturar nubes de puntos de alta resolución y precisión, se pueden necesitar equipos especializados, como

escáneres láser de alta gama o cámaras 3D, lo que puede resultar costoso.

c) **Procesamiento y limpieza:** Antes de su uso, las nubes de puntos pueden requerir procesamiento y limpieza para eliminar artefactos de captura, puntos atípicos o ruido, lo que puede ser un proceso laborioso y requerir experiencia técnica.

d) **Requerimientos de conversión de formato:** Antes de la impresión 3D, la nube de puntos debe ser convertida a un formato compatible para ello, como STL, OBJ o AMF. Este proceso de conversión puede requerir software adicional y tiempo de procesamiento adicional.

2. Modelos mallados, en esta clasificación se encuentran las siguientes representaciones de datos:

   - **Mallas triangulares:** Las mallas triangulares son una de las formas más comunes para la representación superficies tridimensionales en el campo de la computación gráfica y la visualización 3D, tal y como se muestra en la Figura 3.2. Consisten en una colección de triángulos que cubren una superficie en el espacio tridimensional de manera aproximada. Cada triángulo está definido por tres vértices y tres aristas, y puede tener propiedades adicionales como coordenadas de textura, color, normales y otros atributos.

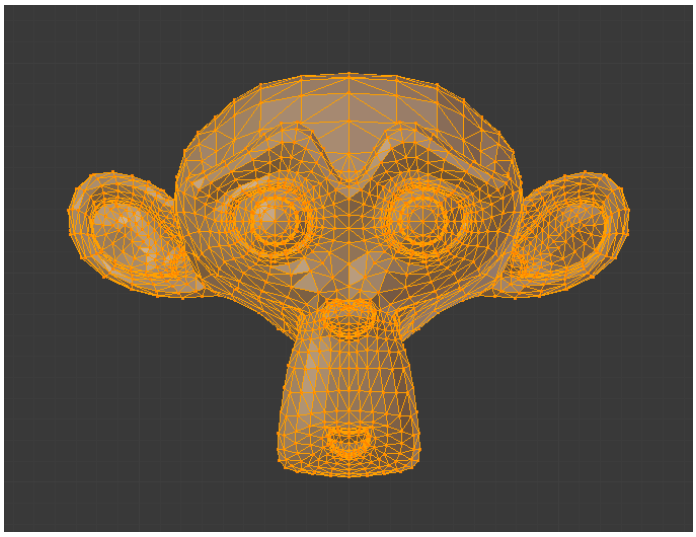


Figura 3.2: Representación de la Malla Triangular. Fuente de la imagen `https://docs.blender.org/manual/es/3.3/modeling/modifiers/generate/triangulate.html`

- **Mallas poligonales (Quads, Ngons):** Estas, a diferencia de las anteriores, pueden ofrecer una mayor flexibilidad en la representación de superficies complejas, ya que pueden contener polígonos con cualquier número de lados (ver Figura 3.3). Esto permite una representación más precisa de la forma del objeto, especialmente en áreas con detalles irregulares o formas no convencionales. Aunque en contraposición a esto, las mallas poligonales pueden requerir más recursos computacionales debido a su mayor complejidad y flexibilidad en la representación de la superficie. Sin embargo, la eficiencia puede variar según el número de polígonos y la complejidad del modelo.

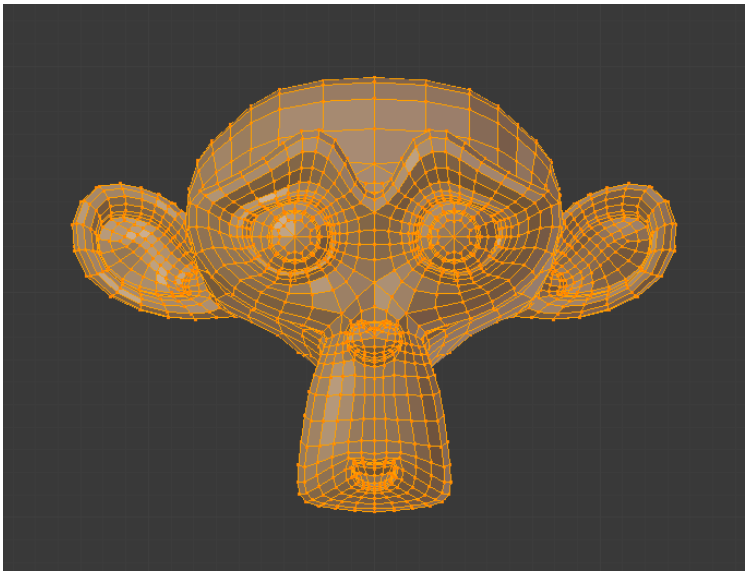


Figura 3.3: Representación de la Malla Poligonal. Fuente de la imagen `https://docs.blender.org/manual/es/3.3/modeling/modifiers/generate/triangulate.html`

A continuación detalladas las ventajas de utilizar esta representación en el proyecto:

a) **Flexibilidad:** Los modelos mallados ofrecen una amplia flexibilidad en la representación de objetos tridimensionales, permitiendo representar de forma adecuada una gran variedad de formas y detalles.

b) **Compatibilidad:** Los modelos mallados son compatibles con una amplia gama de software de diseño, así como con las impresoras 3D estándar.

c) **Precisión:** Los modelos mallados pueden representar con precisión la forma y la geometría de los objetos, lo que permite obtener resultados precisos y detallados en la impresión 3D.

d) **Control de calidad:** Los modelos mallados permiten un control detallado de la calidad de la impresión, lo que facilita la optimización de parámetros como la densidad de la malla, el espesor de las paredes y otros detalles importantes.

e) **Fácil captura de la realidad:** El uso de modelos mallados puede ser relativamente accesible para la captura de objetos 3D, especialmente con técnicas como la fotogrametría y los escáneres 3D.

Aunque también cuentan con sus propias desventajas:

a) **Tamaño del archivo:** Los modelos mallados pueden generar archivos de gran tamaño, especialmente para objetos complejos con una alta resolución de malla, lo que puede afectar la velocidad de procesamiento y el almacenamiento de datos.

b) **Procesamiento complejo:** La creación y manipulación de modelos mallados puede ser compleja y requerir experiencia en software de modelado 3D.

c) **Necesidad de limpieza y reparación:** Los modelos mallados pueden contener errores y defectos, como agujeros, intersecciones no deseadas o geometría no válida, que deben ser detectados y corregidos antes de la impresión 3D.

d) **Resolución limitada:** La resolución de los modelos mallados puede ser limitada por la cantidad de polígonos en la malla, lo que puede afectar la calidad y la precisión de la impresión final.

3. **Modelos de superficies implícitas:** se trata de una técnica avanzada utilizada en el campo de la computación gráfica y la geometría computacional para representar superficies tridimensionales de manera compacta y eficiente. A diferencia de los modelos mallados, que se componen de una colección de polígonos, los modelos de superficies implícitas representan la superficie de un objeto como la solución de una función matemática implícita. En estos modelos se usan las **Función de distancia con signo (SDF)**. Las SDFs se basan en una función matemática que asigna a cada punto en el espacio tridimensional una distancia con signo con respecto a la superficie de un objeto. La distancia se mide desde el punto hacia la superficie del objeto, y puede ser positiva si el punto está fuera del objeto, negativa si está dentro del objeto, y cero si está justo en la superficie del objeto, ver Figura 3.4.

   Como ejemplo sencillo tenemos una ecuación implícita:

$$(x-a)^2 + (y+b)^2 - 1 = 0 \tag{3.1}$$

El conjunto de puntos (x, y) que satisfacen la ecuación definirán la curva de mi objeto, en este caso 2D al estar definida en R^2, en este caso particular los puntos estarán a distancia 1 de (a, b), definiendo así un círculo.

Finalmente, si consideramos (a, b) = (0, 0) el origen de las coordenadas, y la función de distancia firmada definida por todos los puntos de R^2:

$$d(x, y) = \sqrt{(x - a)^2 + (y + b)^2} - 1 \tag{3.2}$$

Analizando d(x, y) en una cuadrícula de puntos definida en el dominio elegido, se obtiene una figura como la siguiente:

- $d(x, y) > 0$ Los puntos externos al círculo
- $d(x, y) < 0$ Los puntos internos al círculo
- $d(x, y) = 0$ Los puntos que están encima del círculo

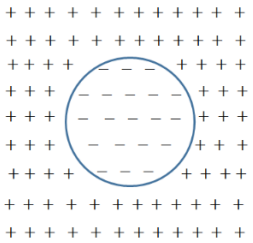


Figura 3.4: Resultado de evaluar la función implícita

La idea de representar la morfología de un objeto 3D mediante una serie de funciones basadas en distancias ofrece varias ventajas, que se detallan a continuación:

a) **Representación compacta:** Las superficies implícitas proporcionan una representación compacta de objetos tridimensionales, ya que solo se necesita almacenar la función implícita que define la superficie. Esto puede resultar en un ahorro significativo de memoria y almacenamiento en comparación con otros métodos de representación.

b) **Suavidad y continuidad:** Las superficies implícitas producen superficies suaves y continuas, lo que las hace ideales para representar formas orgánicas y complejas. La función implícita permite definir la superficie de manera analítica, lo que garantiza una suavidad uniforme en toda la superficie. Lo cual puede producir mejores resultados en impresiones 3D.

c) **Flexibilidad geométrica:** La función implícita puede modificarse fácilmente para controlar la forma y la apariencia de la superficie. Esto permite ajustar rápidamente la geometría del objeto sin tener que reconstruir toda la estructura de la malla.

Aunque pueda parecer una buena opción en un principio, esta representación también cuenta con las siguientes desventajas:

a) **Dificultades de visualización:** A diferencia de las representaciones de malla poligonal, las superficies implícitas pueden ser más difíciles de visualizar y editar directamente. La función implícita puede ser abstracta y difícil de interpretar visualmente, lo que puede dificultar la manipulación directa del objeto.

b) **Dificultades de muestreo:** El proceso de muestreo de una superficie implícita puede ser complejo y computacionalmente costoso. La generación de una malla poligonal a partir de la función implícita puede requerir técnicas avanzadas de discretización y muestreo, lo que puede afectar el rendimiento y la precisión del resultado final.

c) **Compatibilidad de software:** Aunque las superficies implícitas son compatibles con muchos software de modelado y renderizado, pueden no ser completamente inter operables con todas las herramientas y flujos de trabajo. Algunos programas pueden tener dificultades para importar y exportar superficies implícitas debido a diferencias en la implementación y la representación interna. Además de no ser compatible con la impresión 3D directa.

4. **Modelado basado en volúmenes**, se trata de una técnica utilizada para representar objetos tridimensionales mediante la manipulación directa de su información de volumen en lugar de su geometría superficial. El ejemplo más representativo es el modelo de **Voxels**, donde se representan los valores en una cuadrícula regular dentro de un espacio tridimensional, tal y como se muestra en la Figura 3.5. Similar a los píxeles en un mapa de bits 2D, los Voxels generalmente no tienen sus coordenadas explicitadas junto con sus valores. En cambio, los sistemas de renderizado deducen la posición de un vóxel según su relación con otros voxeles en la estructura de datos que conforma una imagen volumétrica única.

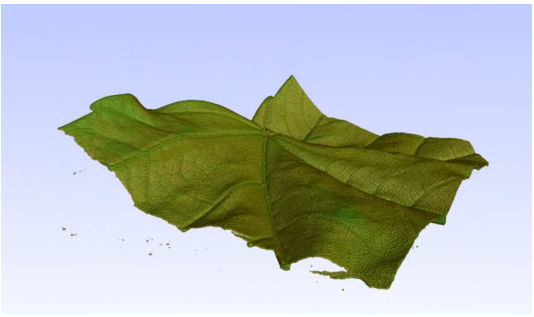


Figura 3.5: Renderizado de un hoja utilizando Voxels

El uso de modelos volumétricos, conlleva con los siguientes beneficios al proyecto:

a) **Representación precisa:** Los voxels permiten una representación precisa de la geometría tridimensional y de los atributos volumétricos, lo que facilita la definición y la manipulación de detalles finos y complejos en los objetos modelados.

b) **Flexibilidad geométrica:** El modelado basado en volúmenes con voxels ofrece flexibilidad para crear y manipular formas tridimensionales de manera intuitiva, lo que permite la generación de objetos con una variedad de formas y estructuras.

Aunque a su vez, las desventajas que conlleva esta representación son numerosas:

a) **Complejidad computacional:** El procesamiento y la manipulación de datos volumétricos representados por voxels pueden ser computacionalmente intensivos, especialmente para conjuntos de datos grandes y de alta resolución, lo que puede requerir hardware y recursos computacionales significativos.

b) **Almacenamiento y transferencia de datos:** Los conjuntos de datos volumétricos pueden ocupar mucho espacio de almacenamiento y pueden ser difíciles de transferir y manipular debido a su tamaño y complejidad, lo que puede dificultar su manejo y su compartición.

c) **Limitaciones en la resolución:** La precisión y el nivel de detalle de los objetos modelados con voxels pueden estar limitados por la resolución de la cuadrícula de estos, lo que puede resultar en una representación menos precisa de los objetos, especialmente en regiones de alta curvatura o detalle.

d) **Interpolación y suavizado:** La representación de objetos basada en voxels puede requerir técnicas de interpolación y suavizado para producir resultados visualmente atractivos y precisos, lo que puede aumentar la complejidad y el costo computacional del proceso.

e) **Incompatibilidad con la impresión 3D:** Este tipo de modelos no son compatibles con la impresión 3D directa, esto significa que deben ser exportados a modelos compatibles como son los mallados, para poder ser impresos en 3D, esto obstaculiza el flujo de trabajo, añadiendo costes computacionales extras.

3.2. Sistemas para la captación de objetos 3D

Esta fase es la base de la pirámide que sostiene todos los procesos posteriores. Por lo tanto, es crucial encontrar un sistema o tecnología que permitan obtener de manera consistente y fiable todos los detalles de la escena posible. La fidelidad y veracidad de la recreación tridimensional se verá directamente afectadas por la calidad y precisión de la información visual obtenida en esta etapa. Por lo tanto, para garantizar resultados óptimos y confiables, es esencial explorar a fondo las diversas opciones y tecnologías disponibles en el campo de la captura de objetos 3D.

Por una parte, se exploran diferentes técnicas dirigidas a capturar objetos 3D de forma consistente, que intentan minimizar errores de captación, como son:

1. **Trípodes y soportes estables:** El uso de trípodes y otros dispositivos de soporte estable para montar la cámara es una forma efectiva de evitar movimientos no deseados durante la captura de imágenes. Esto es especialmente útil en situaciones donde se requiere una precisión extrema.

2. **Sistemas de seguimiento y control de movimiento:** Empleando software o hardware avanzado capaz de rastrear objetos en tiempo real, se logra mejorar significativamente la calidad de los resultados. Esta tecnología permite corregir de manera continua cualquier movimiento que pueda desviar o dificultar la captura del objeto en cuestión, garantizando así una mayor precisión y consistencia en el proceso de captación de imágenes.

3. **Tecnologías de escaneo 3D sin contacto:** Al considerar el empleo de herramientas más sofisticadas y, por ende, costosas, como aquellas que realizan escaneos automáticos de objetos mediante láser u otras tecnologías avanzadas, es posible alcanzar resultados prácticamente perfectos. Este nivel de precisión facilitaría considerablemente todas las tareas posteriores relacionadas con el procesamiento de la información capturada, agilizando así el flujo de trabajo y garantizando la calidad óptima en la recreación de objetos y escenas en entornos tridimensionales.

4. **Técnicas de procesamiento de imágenes y visión por computadora:** Se pueden aplicar técnicas avanzadas de procesamiento de imágenes y visión por computadora para corregir o compensar los efectos de los movimientos no deseados en las imágenes capturadas. Esto puede incluir el uso de algoritmos de estabilización de imagen y técnicas de seguimiento de características para mantener la consistencia entre diferentes fotogramas.

NOTA 1. *Todas estas técnicas mencionadas* ***no son mutuamente excluyentes****, sino que son* ***complementarias****.*

Por lo tanto, basándonos en los requisitos funcionales y no funcionales descritos en el capítulo anterior, la solución final, detallada en el capítulo 4, consistirá en unión de diferentes técnicas.

Por otro lado, existen tecnologías que se enfocan a captar los objetos 3D con calidad, ya que cuanta mayor información y detalles se disponga del objeto 3D o la escena, más facilidad tendrá la red neuronal, recrearlo y posteriormente generar un modelo 3D más detallado. A continuación se enumeran y detallan las tecnologías más actuales (ver tabla resumen 3.1):

1. **Escáneres láser 3D:** Estos aparatos emplean haces de luz láser para mapear la superficie de un objeto y capturar su geometría con gran precisión. Estos dispositivos son altamente valorados por su capacidad para capturar detalles finos y complejas geometrías con una exactitud excepcional. Además, destacan por su fiabilidad, ya que son altamente confiables en multitud de condiciones de iluminación y entorno.

   Sin embargo, es importante considerar que los escáneres láser 3D también tienen sus inconvenientes. Su principal desventaja es el elevado coste asociado, lo que puede añadir un sobrecoste significativo a un proyecto. Además, su complejidad operativa y el requerimiento de conocimientos especializados para su manejo y procesamiento de datos pueden representar otro desafío para su implementación.

2. **Escáneres de luz estructurada:** Los escáneres de luz estructurada utilizan patrones de luz proyectados sobre el objeto para capturar su deformación mediante cámaras especializadas, permitiendo así reconstruir la forma tridimensional de la escena u objeto.

   Sus ventajas incluyen su rapidez para capturar datos, lo que los hace útiles para escanear objetos en movimiento o en entornos dinámicos, así como su buena precisión, adecuada para la reconstrucción de objetos 3D con altos detalles.

   No obstante, a pesar de ser más asequibles que los escáneres láser, aún pueden representar un costo significativo. Además, su rendimiento puede verse

afectado por las condiciones de iluminación y el color del objeto escaneado, limitando su aplicabilidad en ciertos entornos.

3. **Cámaras RGB-D:** Las cámaras RGB-D combinan imágenes RGB con datos de profundidad para capturar tanto la apariencia visual como la información de distancia de los objetos en una escena.

   Una de las ventajas principales de las cámaras RGB-D es su relativo bajo costo en comparación con otros métodos de escaneo 3D.

   Sin embargo, es importante tener en cuenta que estas cámaras también tienen sus desventajas. La resolución de profundidad puede ser limitada en comparación con escáneres láser y de luz estructurada, lo que puede afectar su precisión para capturar detalles finos. Además, pueden ser menos precisas en la captura de estos detalles en comparación con otros métodos más especializados.

4. **Cámaras estereoscópicas:** Las cámaras estereoscópicas utilizan múltiples lentes y sensores de imagen para capturar imágenes desde diferentes ángulos y calcular la profundidad, permitiendo así la reconstrucción tridimensional de la escena.

   Entre sus ventajas, destacan su costo moderado y facilidad de uso, lo que las hace atractivas para aplicaciones que no necesitan la máxima precisión.

   No obstante, requieren algoritmos avanzados para procesar imágenes y calcular la profundidad, lo que puede aumentar la complejidad del software necesario. Además, su precisión de reconstrucción es inferior a la de otros sistemas, como los escáneres láser o de luz estructurada, limitando su aplicabilidad en situaciones que requieren una alta precisión en la captura de datos tridimensionales.

5. **Cámaras de alta resolución:** Las cámaras de alta resolución, con sensores de alta definición, capturan imágenes con gran detalle, siendo ideales para la reconstrucción tridimensional.

   Sus ventajas incluyen la capacidad de capturar imágenes extremadamente detalladas y un costo relativamente asequible según la marca y modelo.

   Sin embargo, no capturan información de profundidad directamente, requiriendo técnicas adicionales para una reconstrucción 3D precisa. Además, el procesamiento de sus imágenes puede ser intensivo en recursos y tiempo, afectando la eficiencia del postprocesamiento.

6. **Tecnología LiDAR (Light Detection and Ranging):** La tecnología LiDAR es una manera precisa de capturar datos tridimensionales de alta calidad. Utilizando pulsos láser para medir distancias a objetos, genera nubes de puntos detallados que representan la superficie de los objetos y el entorno con gran precisión.

Entre sus ventajas, destaca su capacidad para proporcionar datos muy precisos y detallados, lo que la hace ideal para aplicaciones de reconstrucción 3D que requieren alta calidad visual. Además, funciona bien en diversas condiciones de iluminación, incluyendo la oscuridad total.

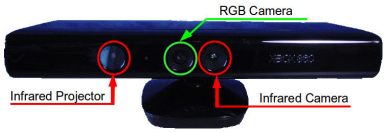
Por otro lado, los sistemas LiDAR pueden resultar costosos, lo que puede limitar su adopción en proyectos con presupuestos reducidos. Además, su operación y procesamiento de datos pueden ser complejos, requiriendo equipos especializados y conocimientos técnicos. También pueden tener dificultades para capturar datos precisos en superficies altamente reflectantes o transparentes.

NOTA 2. *En este caso, las tecnologías sí son* ***excluyentes*** *entre ellas cuando se captura un objeto.*

Por lo tanto, se deberá elegir la más adecuada, según los requerimientos de calidad de imagen establecidos al inicio de este documento, concretamente en la sección de Requerimientos no funcionales.

(a) Ejemplo de Escáner Láser

(b) Ejemplo Escáner Luz Estructurada



(c) Ejemplo cámara RGB-D

(d) Ejemplo Cámara Estereoscópica

(e) Ejemplo Cámara Alta Resolución

(f) Cámara con Tecnología LiDAR

Cuadro 3.1: Representación de los sistemas para la captación de objetos 3D
Fuentes: Escáner Láser Escáner Luz Estructurada Cámara RGB-D
Cámara Estereoscópica Cámara Alta Resolución Cámara con Tecnología LiDAR

3.3. Tratamiento de las imágenes

Esta etapa busca eliminar o reducir al máximo cualquier interferencia o distorsión causada por los métodos de captura de imágenes empleados, con el fin de obtener un resultado final de alta calidad. A continuación se exploran las distintas técnicas utilizadas para: (1) Corregir los errores en los sistemas de captación, (2) Corregir errores producidos por la iluminación de la escena y (3) Técnicas para introducir un cierto ruido en las escenas para facilitar coincidencias entre frames.

Técnicas para la corrección de los errores producidos por los sistemas de captación:

Con los archivos de imágenes y vídeo, existen las siguientes técnicas ordenadas de menor a mayor complejidad de implementación y requerimientos de hardware: los filtros de suavizado, los métodos basados en transformadas y las técnicas basadas en Deep Learning.

Con relación a los **filtros de suavizado**, el Filtro Gaussiano suaviza la imagen reduciendo el ruido, sin perder demasiados detalles. En cambio, el Filtro de Mediana es especialmente eficaz para eliminar el ruido, sal y pimienta. Finalmente, el Filtro Bilateral suaviza la imagen mientras mantiene los bordes.

Los **métodos basados en transformadas**, como la Transformada de Ondículas, se utiliza para descomponer la imagen en diferentes niveles de detalle y eliminar el ruido en niveles específicos. Esta técnica convierte la imagen en una serie de componentes de frecuencia, permitiendo la eliminación de ruido en ciertos niveles. Por otro lado, la Transformada de Fourier filtra el ruido según el dominio de la frecuencia, gracias a convertir la imagen en una representación de sus componentes sinusoidales, facilitando la identificación y eliminación de las frecuencias no deseadas.

Por último, y técnicas de mayor complejidad existen las relacionadas con **Deep Learning**, las Redes Neuronales Convolucionales(CNN) se utilizan para la reducción de ruido en imágenes y vídeos. Estas redes consisten en capas de filtros convolucionales que aprenden a identificar y eliminar el ruido durante el proceso de entrenamiento, mejorando la calidad de la imagen o el vídeo de manera efectiva.

> **NOTA 3:**. *Los métodos de corrección escogidos deben tener un trade-off entre el coste del método, su eficacia de cara a la red neuronal y la complejidad asociada.*

Técnicas para la corrección de color o iluminación de la escena:

En el caso de los archivos de imágenes y vídeo, existen las siguientes técnicas ordenadas (de menor a mayor) por complejidad de implementación y requerimientos de hardware:

En primer lugar, se encuentran los **ajustes de color**. El Balance de Blancos corrige las desviaciones de color debidas a la iluminación, asegurando que los colores se representen de manera precisa. Por otro lado, la Corrección Gamma ajusta los niveles de brillo y contraste, mejorando la visualización de los detalles en la imagen.

Y para concluir con las técnicas de corrección de color e iluminación, se encuentran los **ajustes de histogramas**. La Ecualización del Histograma mejora el contraste de la imagen distribuyendo los valores de intensidad de manera más uniforme. Además, el Adaptive Histogram Equalization (CLAHE) es una versión mejorada que trabaja en pequeñas regiones de la imagen, proporcionando un ajuste más detallado y preciso.

> **NOTA 4:**. *Destacar que la técnica a elegir debe ser lo mas flexible posible a cualquier tipo de entorno posible para ayudar al usuario de forma adecuada.*

Técnicas para añadir ruido a la escena:

En el caso de las técnicas de para aportar información extra a la escena y facilitar de este modo el reconocimiento de coincidencias entre frames, las más utilizadas son el ruido Gaussiano y el ruido denominado Sal y Pimienta, que se detallan a continuación.

(a) Ruido Gaussiano

(b) Ruido Sal y Pimienta

Cuadro 3.2: Comparativa de los resultados de los tipos de ruido analizados. Imágenes extraídas del artículo.

Ruido Gaussiano: El ruido gaussiano, caracterizado por una distribución normal de valores de intensidad, es común en muchas aplicaciones de imagen (ver Figura 3.2(a)). Entre sus ventajas, destaca su realismo, ya que imita las condiciones reales de ruido, y su facilidad de implementación, ya que es sencillo de generar y añadir a las imágenes. Sin embargo, presenta la desventaja de distribuirse uniformemente, lo que puede no reflejar todas las condiciones reales, y requiere un ajuste adecuado de la desviación estándar para simular distintos niveles de ruido de manera precisa.

Ruido Sal y Pimienta: El ruido sal y pimienta se caracteriza por píxeles que son reemplazados aleatoriamente por valores blancos o negros (ver Figura 3.2(b)). Este cuenta con algunas ventajas, entre estas se destaca su capacidad para simular defectos reales, imitando efectivamente ciertos tipos de ruido presentes en sensores defectuosos o en condiciones de baja iluminación. Aunque también cuenta con algunas desventajas, como la inclusión de una alta distorsión, que puede afectar significativamente la calidad de la imagen y la reconstrucción 3D, así como la destrucción de detalles finos, lo que resulta problemático para aplicaciones que requieren precisión como es el caso de este proyecto.

NOTA 5:. *Cada tipo de ruido puede ser útil según distintas situaciones.*

3.4. Reconstruccion 3D

En esta etapa del proyecto es el núcleo más importante y un paso crucial, que sirve como eje central para todos los demás componentes. A continuación se expondrán dos enfoques distintos disponibles en la investigación de renderización de escenas 3D, con el propósito de analizarlas y, basándonos en los requisitos funcionales y no funcionales, seleccionar una de las dos:

3D Gaussian Splatting

3D Gaussian Splatting, es un método para representar y renderizar escenas tridimensionales mediante la proyección de partículas o puntos gaussianos en el espacio 3D. En lugar de utilizar modelos tradicionales de mallas poligonales (comparación entre las dos representaciones en la Figura 3.6), este enfoque se basa en la idea de emplear distribuciones gaussianas para describir la geometría y las propiedades de la superficie de los objetos en una escena. Basado en esta idea, se ha propuesto una técnica de reconstrucción 3D basada en el Descenso del Gradiente Estocástico [7]. Para introducir esta idea, primero se define el concepto de punto gaussiano a continuación.

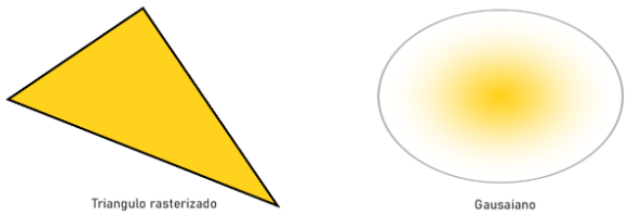


Figura 3.6: Comparación triángulo y Gausiano. Fuente de la imagen web

Los puntos de una superficie en 3D se definen como un conjunto de puntos gaussianos (es decir, cada punto de la superficie tiene asociada una función gaussiana). Por

tanto, es análogo a la rasterización de triángulos en los gráficos por computador, que se utiliza para dibujar muchos triángulos en la pantalla. Sin embargo, en lugar de dibujar triángulos, son distribuciones gaussianas en cada punto. Estas distribuciones se describen mediante los siguientes parámetros:

- **Posición:** dónde está ubicado espacialmente el punto de la superficie (XYZ)
- **Covarianza:** cómo se estira/escala la matriz o base gaussiana asociada al punto (matriz 3×3)
- **Color:** El color base asociado al punto, codificado en RGB
- **Alfa:** El valor de su transparencia (α)

El objetivo del algoritmo de aprendizaje automático es determinar los parámetros de cada gaussiano para cada punto del objeto. La estrategia de aprendizaje 3D Gaussian Splatting parte de una escena en formato de nube de puntos, para transformarla en una escena 3D con millones de partículas (gaussianas), donde cada gaussiano 3D viene con una posición/orientación/escala, así como una opacidad y un color.

Los pasos que sigue esta estrategia son:

1. **Estructura a partir del movimiento:** Se empieza utilizando el método "Structure from Motion"(SfM) para crear una nube de puntos a partir de un conjunto de imágenes de entrada. Esto puede realizarse utilizando la aplicación `COLMAP`.
2. **Convertir a gaussianos:** Cada punto se convierte en gaussiano, lo que hace que ya sea posible para la rasterización. Sin embargo, a partir de los datos de SfM solo se pueden inferir la posición y el color.
3. **Entrenamiento:** Para que una representación produzca resultados de alta calidad, debemos entrenarla. Para ello, el algoritmo corrige los parámetros utilizando el algoritmo Descenso Gradiente Estocástico, que a cada paso:

   a) Visualiza los gaussianos en una imagen usando rasterización gaussiana diferenciable.

      La rasterización gaussiana diferenciable consiste en proyectar en cada píxel, de adelante hacia atrás (según la posición de la cámara), cada punto gaussiano en 2D, combinando las proyecciones de distintos puntos gaussianos que se solapan en el mismo píxel.

   b) Calcula la pérdida en función de la diferencia entre la imagen rasterizada y la imagen real del terreno.

   c) Ajusta los parámetros gaussianos según la pérdida.

   d) Aplica densificación automatizada.

3D Gaussian Splatting presenta varias ventajas, entre ella se destaca su capacidad de proporcionar una representación precisa de volúmenes de datos en imágenes 2D, manteniendo la información espacial tridimensional de manera fiel y detallada. Además, es computacionalmente eficiente, ya que la renderización no requiere un procesamiento pesado, lo cual es beneficioso para aplicaciones que demandan rapidez y eficiencia.

Sin embargo, también tiene ciertas limitaciones. La calidad de la representación visual puede estar restringida por la resolución de la imagen resultante, especialmente en áreas con alta densidad de puntos o detalles finos. Además, la efectividad del Gaussian Splatting depende en gran medida de la configuración adecuada de parámetros, como el ancho del kernel gaussiano y la densidad de los puntos de entrada, lo que puede requerir ajustes y pruebas experimentales.

Otro inconveniente es el posible aumento en los requisitos de almacenamiento y memoria, ya que algunas implementaciones del 3D Gaussian Splatting pueden necesitar almacenamiento adicional para datos intermedios o precalculados. Finalmente, la complejidad de implementación puede ser un obstáculo, dado que requiere conocimientos avanzados en gráficos por computadora y procesamiento de imágenes, lo que puede dificultar su adopción y uso por parte de usuarios menos experimentados.

NOTA 6. *El Guassian Splating es especialmente útil es entornos con bajas prestaciones computacionales y en objetos que no requieran un nivel de detalle muy preciso.*

Instant NGP

"*Neural Radiance Fields*" o NeRF, es una técnica de renderizado 3D que utiliza redes neuronales para modelar escenas tridimensionales a partir de una serie de imágenes 2D. La idea detrás de NeRF es utilizar una red neuronal para aprender la radiancia emitida por un objeto en una escena tridimensional a partir de datos de entrada como imágenes y vídeos. La red neuronal aprende la relación entre la radiancia del objeto y su posición en el espacio, lo que permite generar nuevas vistas de la escena en tiempo real (ver Figura 3.7).

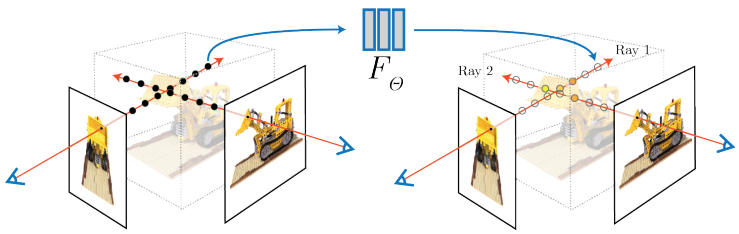


Figura 3.7: Esencia del NeRF. Fuente de la imágenes en el artículo sobre NeRF

El tiempo de entrenamiento puede ser muy caro, necesitando varios minutos o incluso horas para entrenar el modelo con una escena sencilla. Por eso, desde que se publicó el primer artículo, han ido apareciendo nuevos estudios y proyectos que exploran distintas formas de entrenar y agilizar el modelo, aquí es donde aparece el método **Instant NGP**, que utiliza la potencia de las tarjetas gráficas Nvidia para entrenar un modelo en segundos o minutos y, después, permite cargar directamente este modelo entrenado, reduciendo el tiempo de entrenamiento y renderizado de un modelo.

¿Cómo funciona Instant NGP?

Las primitivas gráficas neuronales pueden ser costosas de entrenar y evaluar debido a la alta cantidad de operaciones y accesos a memoria. Con la finalidad de evitar este coste, se modifica la codificación de entrada convencional, la cual reduce el coste del entrenamiento y la evaluación además de permitir la utilización de una red más pequeña sin sacrificar la calidad, reduciendo así de forma significativa el número de operaciones de coma flotante y de acceso a memoria. De forma adicional, esto se complementa con el uso de una tabla hash multirresolución de vectores de características cuyos valores se optimizan mediante el descenso de gradiente estocástico. La estructura multirresolución permite a la red resolver las colisiones de hash, lo que da lugar a una arquitectura sencilla que es trivial de paralelizar en las GPU modernas.

Las ventajas más importantes de utilizar este enfoque son que Instant NGP ofrece una representación precisa al generar imágenes 3D altamente realistas y fotorrealistas de escenas, proporcionando una fiel representación de la geometría y la apariencia de los objetos. Además, permite la exportación mallada, lo que facilita la extracción de los datos renderizados en un formato de malla compatible para la impresión 3D.

No obstante, los requerimientos computacionales son significativos, ya que el proceso de entrenamiento y generación de imágenes con **Instant NGP** puede ser intensivo en términos de recursos, especialmente para escenas complejas y de alta resolución, demandando hardware potente y considerable tiempo de procesamiento. Asimismo, la complejidad de implementación es alta, dado que el ajuste y la puesta en marcha de los modelos de **Instant NGP** pueden requerir conocimientos avanzados en aprendizaje profundo y gráficos por computadora, lo que puede dificultar su adopción y uso por parte de usuarios menos experimentados.

3.5. Suavizado de puntos/mallas

En este apartado se exploran diferentes herramientas capaces de extraer los datos de salida de la reconstrucción de la escena/objeto y en caso de que estos contengan errores e imperfecciones, poder eliminarlas o reducirlas al máximo para garantizar una cierta calidad en la impresión final.

Para ello, se disponen de numerosos productos de modelado 3D que permiten realizar este tipo de suavizado. A continuación, se exploran MeshLab [11] y Blender [4], como aplicaciones más representativas.

MeshLab

MeshLab [3] es una potente herramienta de código abierto diseñada para el procesamiento y la manipulación de modelos tridimensionales. Fue desarrollada por el Instituto Nacional de Investigación en Informática y Automatización (INRIA) en Francia. MeshLab ofrece una amplia gama de métodos para editar, limpiar, visualizar y analizar datos de malla en 3D.

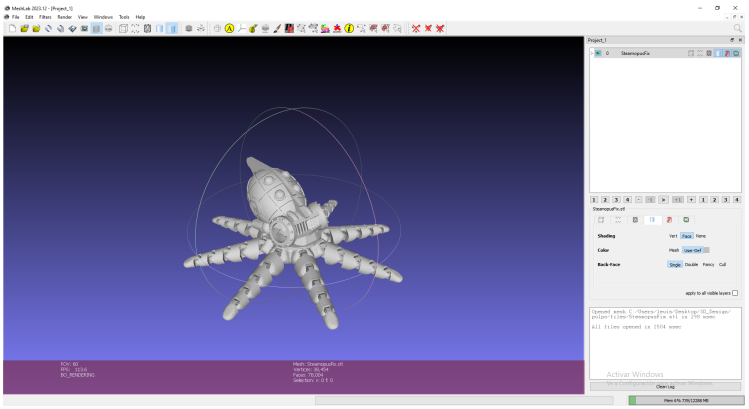


Figura 3.8: Interfaz de MeshLab.

MeshLab se destaca por ofrecer una amplia gama de herramientas de procesamiento, permitiendo a los usuarios realizar diversas operaciones de postprocesamiento en modelos tridimensionales. Además, cuenta con avanzadas herramientas de limpieza y corrección de errores, capaces de corregir errores topológicos, eliminar vértices duplicados y suavizar la geometría de los modelos. Al ser un software de código abierto, MeshLab es altamente personalizable y extensible, lo que permite a los usuarios escribir scripts y automatizar tareas específicas de postprocesamiento según sus necesidades.

Sin embargo, MeshLab presenta una curva de aprendizaje empinada, especialmente para los usuarios que no están familiarizados con el procesamiento de modelos tridimensionales y las herramientas de manipulación de mallas. Asimismo, dependiendo del tamaño y la complejidad del modelo tridimensional, MeshLab puede experimentar limitaciones de rendimiento, lo que puede resultar en tiempos de procesamiento prolongados o falta de capacidad para manejar modelos muy detallados.

Blender

Blender [4] es una suite de creación 3D completa y potente que ofrece herramientas para todas las etapas del proceso de producción, desde el modelado y la animación hasta el renderizado y la composición. Su naturaleza de código abierto y su comunidad activa de usuarios y desarrolladores hacen de Blender una opción atractiva para aquellos que buscan una solución gratuita y de alta calidad para sus proyectos en 3D.

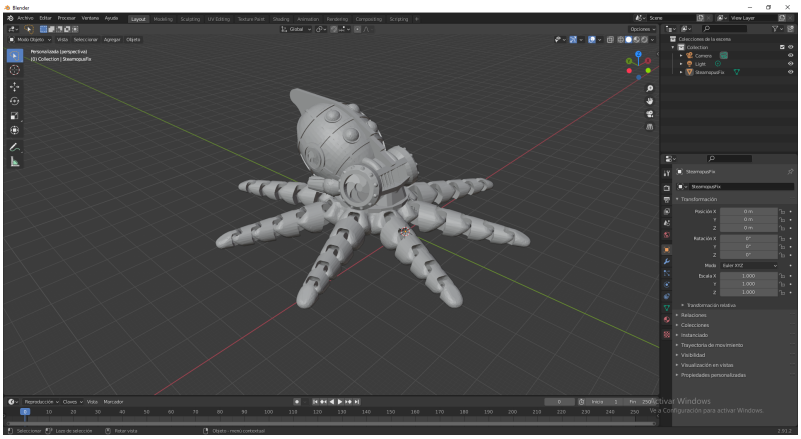


Figura 3.9: Interfaz de Blender

Blender cuenta con una comunidad activa de usuarios y desarrolladores que comparten recursos, tutoriales y complementos, facilitando el aprendizaje y la resolución de problemas. Además, ofrece una amplia gama de herramientas para el procesamiento de modelos tridimensionales, incluyendo modelado, animación, renderizado y composición, permitiendo una variedad de operaciones de postprocesamiento. Al igual que MeshLab, Blender es altamente personalizable y extensible gracias a su API de Python, lo que permite a los usuarios escribir scripts y complementos para automatizar tareas específicas de postprocesamiento según sus necesidades.

Todo esto, teniendo en cuenta las siguientes desventajas. La API de Python para Blender está adaptada para interactuar con la propia aplicación de Blender, lo que dificulta su conexión con programas externos. Además, aunque Blender ofrece una amplia gama de herramientas, su interfaz de usuario es aún más compleja que la de MeshLab, lo que puede resultar abrumador para algunos usuarios y dificultar la navegación y el acceso a las características necesarias para el postprocesamiento.

3.6. Visualización 3D

Al realizar modificaciones sobre la malla procesada anteriormente, el usuario debe disponer de una herramienta que facilite esta tarea, permitiéndole visualizar los cambios aplicados.

Para ello, se pueden plantear dos alternativas:

Si se elige la herramienta `3D Gaussian Splatting`, esta ofrece como salida una nube de puntos. Por lo tanto, una técnica adecuada para visualizar este tipo de representaciones es la de Splatting. En Python, existen diferentes bibliotecas que implementan las funciones necesarias para aplicar la técnica de Splatting, entre ellas `Open3D` y `PyntCloud`.

En cambio, si se opta por la herramienta `Instant NGP`, esta produce una representación mallada. Para visualizar este tipo de representaciones en Python, las opciones disponibles incluyen utilizar la técnica de `Z-Buffer` o `Raytracing`.

NOTA 7. *La elección del método de visualización anterior dependerá directamente de la red neuronal escogida.*

3.7. Impresión 3D

En este apartado se analizan dos laminadores bastante populares en el mundo de la impresión 3D (PrusaSlicer [8] y Ultimaker Cura [2]). Un laminador 3D es una herramienta o software utilizado en el proceso de impresión 3D que convierte un modelo tridimensional digital en una serie de capas horizontales (láminas) que la impresora 3D puede construir secuencialmente. Este proceso, conocido como laminado o slicing, es crucial para la impresión 3D, ya que traduce la geometría del modelo en instrucciones específicas para la impresora, detallando cómo debe depositar o solidificar el material capa por capa para crear el objeto físico.

Las funciones principales que debe ofrecer un laminador 3D son las siguientes:

1. **División del Modelo en Capas:** Descompone el modelo 3D en múltiples capas horizontales, cada una con una altura definida por el usuario o las capacidades de la impresora.
2. **Generación de Trayectorias de Impresión:** Calcula las rutas exactas que debe seguir la boquilla de la impresora para depositar el material en cada capa, optimizando para minimizar tiempos de impresión y maximizar la precisión y calidad.
3. **Configuración de Parámetros de Impresión:** Permite ajustar parámetros críticos como la velocidad de impresión, temperatura del extrusor, altura de capa, relleno, soportes y más.

4. **Creación de Instrucciones G-Code:** Traduce las rutas y configuraciones en un archivo G-Code, que es el lenguaje estándar utilizado por la mayoría de las impresoras 3D para ejecutar las instrucciones de impresión.

PrusaSlicer [8] es un software de laminado desarrollado por Prusa Research, basado en el popular laminador `Slic3r`. Está especialmente optimizado para las impresoras 3D de Prusa, pero también es compatible con una amplia gama de otras impresoras. PrusaSlicer se destaca por su robusta funcionalidad, incluyendo características avanzadas como la generación de soportes personalizados, perfiles de materiales específicos y configuraciones detalladas para usuarios avanzados.

Cura [2] es un software de código abierto para el laminado de modelos 3D, desarrollado por Ultimaker. Es uno de los laminadores más populares y ampliamente utilizados en la comunidad de impresión 3D debido a su versatilidad, facilidad de uso y amplio soporte de impresoras.

Ambos laminadores son potentes herramientas para la impresión 3D, cada uno con sus propias fortalezas. Cura es excelente para aquellos que buscan una amplia compatibilidad y una interfaz amigable, mientras que PrusaSlicer es ideal para usuarios que buscan funcionalidades avanzadas y una optimización específica para impresoras Prusa. La elección entre ambos dependerá de las necesidades específicas del usuario y del tipo de impresora utilizada.

3.8. Conclusiones del capítulo

Para concluir con el capítulo, se enumerarán las tecnologías expuestas anteriormente, añadiendo una conclusión final con la solución escogida para formar parte del proyecto.

1. **Sistemas de captación de imágenes:**

Sistema	**Coste**	**Complejidad**	**Tiempo**
Sistema de Soporte	Bajo	Baja	Bajo
Seguimiento del Objeto	Medio	Alta	Medio
Escaneo sin contacto	Alto	Media	Medio
Técnicas de Visión por Computadora	Bajo	Alta	Alto

Cuadro 3.3: Resumen de sistemas de captación de imágenes consistentes

Sistema	Coste	Complejidad
Escáner Láser	Alto	Alta
Escáner Luz estructurada	Alto	Alta
Cámara RGB-D	Medio	Media
Cámaras Estereoscópicas	Medio	Media
Cámara alta definición	Bajo	Baja
Tecnología LiDAR	Medio	Baja

Cuadro 3.4: Resumen de sistemas de captación de imágenes de calidad

Como se puede observar en las diferentes tablas (3.3 y 3.4) y teniendo como referencia en todo momento los requerimientos funcionales y no funcionales establecidos al principio del proyecto, la decisión más lógica y sencilla es el uso de una cámara convencional de móvil juntamente con la ayuda de un suporte físico. Por lo tanto, será necesario diseñar y construir un soporte para la captación de las imágenes, que garantice mediante algún método, ya sea físico o a través de software, el seguimiento del objeto en el centro de la escena.

En cuanto a los sistemas de captación de imagen, lo más apropiado es el uso de cámaras de alta definición. Aunque, otra alternativa podría ser el uso de los sistemas de cámaras estereoscópicas. Considerando los requerimientos no funcionales definidos en la sección 2.1, y con la finalidad de abaratar costes, se utilizará una cámara de alta definición.

2. **Tratamiento de las imágenes:**

   Este apartado está compuesto por 3 problemas distintos, en primer lugar el de **Eliminación de ruido** (Tabla 3.5), a continuación **Corrección de color e iluminación** (Tabla 3.6) y finalmente para escenarios puntuales la **Inserción de ruido** (Tabla 3.7) en la escena.

Sistema	Coste	Complejidad
Filtros de suavizado	Bajo	Baja
Métodos Basado en Transformadas	Medio	Media
Deep Learning	Alto	Alto

Cuadro 3.5: Resumen Técnicas para la corrección de los errores producidos por los sistemas de captación

Siguiendo los requerimientos establecidos en el Capítulo 2, se elegirá la técnica con menor coste y complejidad asociado. Por lo tanto, en este proyecto se usarán técnicas que implementan filtros de suavizado.

Sistema	**Coste**	**Complejidad**
Ajustes de Color	Bajo	Baja
Ajustes de Histogramas	Medio	Media

Cuadro 3.6: Resumen Técnicas para la corrección de color o iluminación de la escena

Siguiendo los requerimientos establecidos en el Capítulo 2, se elegirá la técnica con menor coste y complejidad asociado. Se implementarán funciones que modifiquen las características básicas de las imágenes como son, luminosidad, contraste y color.

Sistema	**Coste**	**Complejidad**
Ruido Gaussiano	Bajo	Baja
Ruido Sal y Pimienta	Bajo	Baja

Cuadro 3.7: Técnicas para añadir ruido a la escena

Para concluir con la sección, ambas técnicas son de coste y complejidad baja, por lo tanto, pueden implementarse las dos en el proyecto, favoreciendo, de esta forma, las vías de investigación en la configuración del pipeline.

3. **Reconstrucción 3D:**

   A continuación el resumen (Tabla 3.8) sobre las tecnologías sobre la reconstrucción 3D del objeto, comentadas anteriormente.

Sistema	**Requerimientos**	**Complejidad**	**Tiempo**
Gaussian Splatting	Bajos	Alta	Medio
Instant NPG	Medios	Media	Medio

Cuadro 3.8: Resumen de los enfoques para la reconstrucción 3D

El sistema 3D Gaussian Splatting trabaja con puntos gaussianos, al ser una representación basada en volúmenes (recordar sección 3.1), conlleva la transformación de los datos a una representación compatible con la impresión 3D, como son las mallas triangulares. Esta transformación es compleja, además de producir ruido adicional en el proceso.

Por otra parte, tenemos el enfoque de los NeRF, utilizando el software `Instant NPG`. Requiere de un hardware potente para la renderización de escenas debido al cálculo de los rayos, y los valores de los parámetros de la red neuronal. En contraposición, pueden trabajar con sistemas mallados, además de integrar un sistema de exportación a fichero OBJ, esto facilita la exportación de los datos reconstruidos, minimizando así las operaciones posteriores.

Expuestas las ventajas y desventajas, se optará por el uso de la herramienta `Instant NGP` en el proyecto.

4. **Suavizado de puntos/mallas:**

   A continuación la tabla 3.9, donde se resumen las principales características de cada solución en función de los requerimientos de la sección 2.1, para el suavizado de mallas.

Sistema	**Requerimientos**	**Complejidad**	**Tiempo**
MeshLab	Bajos	Media	Bajo
Blender	Medios	Media	Medio

Cuadro 3.9: Resumen de los Softwares para el suavizado de puntos y mallas

   MeshLab, al tener una interfaz sencilla y ligera, requiere de menos recursos para realizar sus funciones, además, cuenta con una librería de Python para realizar todas sus funciones, por lo tanto, se puede acoplar perfectamente a un sistema independiente. En contraposición, Blender, está diseñado para una multitud de tareas adicionales a las propuestas en este punto, provocando así que la interfaz sea más pesada a nivel de hardware. Adicionalmente, cuenta con una API de Python para realizar todo tipo de funciones, pero esta depende del visualizador de la aplicación, provocando que el acople con otros sistemas, sea más complicado.

   Como consecuencia de la comparativa anterior, en este proyecto se opta por el uso e implementación de `MeshLab`.

5. **Visualización 3D:** Al elegirse la herramienta de **Insant NGP**, la implementación para visualizar la malla se realizará utilizando el método de **`Z-Buffer`** debido a su simplicidad y velocidad.

Capítulo 4

Diseño de la aplicación

Analizados los diferentes antecedentes de la actualidad y valorando sus pros y contras en función de los requerimientos establecidos, al principio del proyecto, en este capítulo se expone la arquitectura de la solución propuesta. En primer lugar, se realizará una breve introducción de las decisiones tomadas con relación a las tecnologías en cada etapa del pipeline. A continuación, se describe el flujo de los datos por este pipeline. Para terminar, concluida la introducción, se entrará en detalle de las funciones implementadas en cada etapa, así como, su funcionamiento interno.

4.1. Overview de la solución diseñada

En esta sección se describe la solución final desarrollada. Para ello en la Figura 4.1 aparecen las etapas finales del pipeline, estas son fruto de las tecnologías seleccionadas anteriormente.

Para contextualizar el diagrama de la Figura 4.1, se asume que el usuario realizará las acciones utilizando un ordenador de sobremesa, una impresora 3D y, opcionalmente, la cámara de un móvil con una extensión robótica diseñada en este proyecto. Además, en la figura se pueden observar nuevas herramientas que no han sido explicadas previamente. A continuación, se introducen estas herramientas:

- **`FFMPEG:`** Esta es una potente suite de software libre y de código abierto para manejar archivos multimedia, incluyendo audio, vídeo y otros flujos multimedia. Es conocido por su versatilidad y capacidad para realizar una amplia gama de tareas relacionadas con la manipulación de medios. Esta herramienta se requiere en el proyecto para extraer del vídeo de entrada los correspondientes frames.
- **`Flask:`** Es un microframework de Python utilizado en este proyecto para implementar un microservicio con arquitectura cliente-servidor, empleando una interfaz API RESTful.

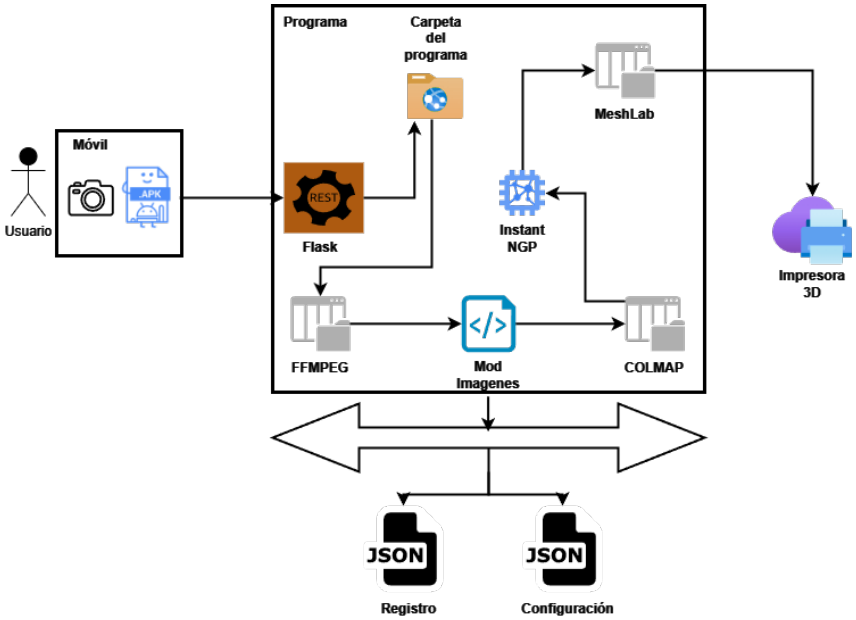


Figura 4.1: Diagrama final de la aplicación desde la captación de la escena real hasta su replicación mediante una impresión 3D.

A continuación, se presenta un resumen general del flujo de trabajo del aplicativo del proyecto:

1. El flujo de todo el proceso empieza capturando la escena real mediante el uso de la cámara de un móvil, aunque este paso es opcional, ya que si se dispone de la grabación o las imágenes del objeto 3D, se puede empezar el proceso desde la aplicación del PC. Cuando se realiza la captura desde un móvil, se usará l'aplicación **`app3d.apk`**, aplicación .apk (para Android) que se encuentra en la carpeta Server en el repositorio del proyecto. Esta aplicación permite mandar de forma sencilla el vídeo realizado desde el móvil al PC a través del microservicio de **`Flask`**, expuesto anteriormente.

2. Una vez el vídeo está en el PC u ordenador de sobremesa, este debe segmentarse utilizando el programa **`FFMPEG`**, integrado en la GUI del aplicativo.

3. Después del segmentado del vídeo, dependiendo de las condiciones del entorno en que se ha tomado este, deberán ajustarse los parámetros de los distintos frames obtenidos. Todo esto se realizará mediante el uso distintas funciones, expuestas posteriormente.

4. Realizadas las modificaciones pertinentes, se pasan las imágenes al programa **`COLMAP`**, el cual extraerá los puntos clave de cada imagen, las coincidencias entre ellas y generará los datos necesarios para el input de la red neuronal de la herramienta **`Instant NGP`**.

5. Una vez generados los datos de input, se pasarán a la herramienta seleccionada para la renderización de la escena **`Instant NGP`**. Virtualizada la información y pasado un tiempo prudente de la renderización, para obtener el mejor resultado, se podrá exportar los datos a formato .obj, mediante el uso interno de la herramienta que `Instant NGP` dispone para ello.

6. Los datos resultantes de la renderización 3D, se pasan a la librería **`MeshLab`**, que se encargará de eliminar ruido y refinar el resultado final lo máximo posible antes de mandarlo a la impresora 3D.

Cada etapa dispone de una ventana que muestra todas las salidas relacionadas con la ejecución de las funciones. Además, en esta ventana se aporta feedback extra al usuario en caso de detectar errores de configuración o parámetros pendientes de establecer.

4.2. Sistema de captación de imágenes

Considerando los argumentos explicados en el capítulo anterior, en el que concluimos que la solución más idónea era la opción de usar una cámara de alta resolución, como por ejemplo la de un móvil.

Para conseguir estos objetivos, se ha planteado la creación y el diseño de un soporte físico para el móvil, que consiste en una pequeña plataforma, en el centro de la cual se situará el objeto a escanear. Esta plataforma dispondrá de un brazo que servirá de soporte del móvil y que irá rotando alrededor del objeto grabando un vídeo de la escena.

El diseño del soporte ha sido elaborado íntegramente con el software **`Autodesk Fusión`** [1]. Se ha optado por el diseño de un soporte de pequeño tamaño, que será suficiente para escanear la mayoría de los objetos. La restricción de tamaño viene dada por la capacidad de la impresora 3D que se dispone y el tiempo que conlleva la impresión de una pieza de mayor tamaño. A continuación se describe el prototipo diseñado y sus diferentes partes:

1. **Brazo:** Utilizado como soporte para el dispositivo de captación de imágenes.

2. **Base:** Sustentación para el brazo y la plataforma.

3. **Plataforma:** Lugar donde se situara el objeto escaneado.

4. **Engranajes:** Estos serán utilizados para girar el brazo.

En la Figura 4.2 se muestra el diseño realizado del **brazo**.

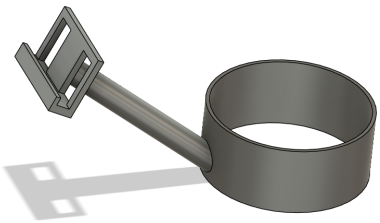


Figura 4.2: Diseño del brazo. Especificaciones Técnicas: Circunferencia central 95 mm de diámetro, Longitud Brazo 110 mm, Soporte cámara 60x40x10 mm

En relación con la **base**, la pieza consta de una hendidura para evitar que la rotación del brazo se desplace del eje central. También consta de una apertura para el cableado eléctrico del motor que girara el brazo (ver Figura 4.3).

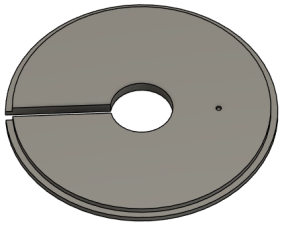


Figura 4.3: Diseño de la base. Especificaciones Técnicas: Circunferencia interior 95 mm de diámetro, Circunferencia exterior 100 mm de diámetro.

En relación con la **plataforma**, se ha diseñado usando una forma circular con dos soportes centrales a modo de columna y un tercer soporte auxiliar para enganchar un motor en caso de ser necesario, como se observa en la Figura 4.4.

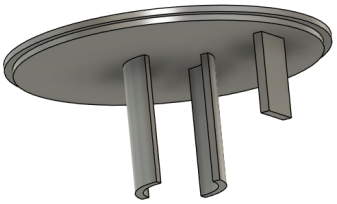


Figura 4.4: Diseño de la plataforma. Especificaciones técnicas: Diámetro columna central 23 mm con offset 5 mm exterior, Circunferencia interior 95 mm de diámetro, Circunferencia exterior 100 mm de diámetro, Soporte motor 13x25x3 mm.

Siguiendo con las especificaciones técnicas de los **engranajes**, consideradas en la Figura 4.5, cabe destacar que tienen unas proporciones de una relación 4 a 1. Esto debido a que si en un futuro se tiene que implementar un algoritmo más complejo, con el que captar las imágenes de la escena, puede calcularse con más facilidad cuantas vueltas tiene que dar el engranaje pequeño para que el grande realice una.

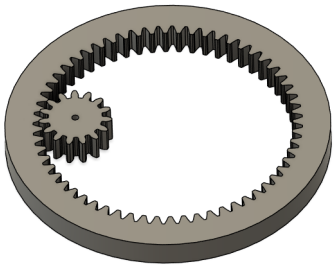


Figura 4.5: Diseño de los engranajes. Especificaciones técnicas: Engranaje menor de 15 dientes y 20 mm de diámetro, Engranaje mayor de 60 dientes y 96 mm de diámetro.

Una vez diseñado el brazo rotatorio, se ha acoplado un motor paso a paso controlado por una Raspberry Pi 3, comunicados mediantes una placa controladora del motor. Así, los componentes para rotar la infraestructura son los siguientes:

- Motor paso a paso: Fasizi 28BYJ-48 5V. Figura 4.1(a)
- Placa controladora del motor: ULN2003. Figura 4.1(a)
- Controlador del sistema de movimiento: Raspberry Pi 3 Model B. Figura 4.1(b)

(a) Motor paso a paso y placa controladora

(b) Raspberry Pi 3 Modelo B

Cuadro 4.1: Componentes de control de movimiento. Imágenes extraídas de Amazon.

Se han configurado las conexiones de la Raspberry según los pins mostrados en la Figura 4.6 para establecer la comunicación con la placa controladora del motor:

1. Fuente de 5 V al polo positivo de la placa controladora del motor.

2. Ground al negativo de la placa controladora del motor.
3. GPIO 17 al pin1 de la placa controladora del motor.
4. GPIO 27 al pin2 de la placa controladora del motor.
5. GPIO 22 al pin3 de la placa controladora del motor.
6. GPIO 23 al pin4 de la placa controladora del motor.

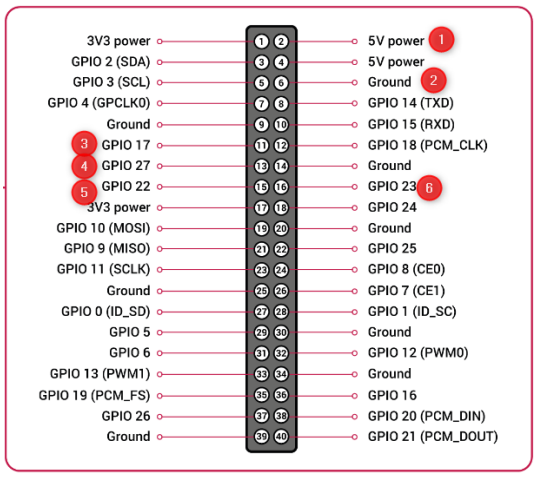


Figura 4.6: Conexiones GPIO en Raspberry Pi 3. Fuente imagen Documentación Raspberry Pi

Resultado final de la pieza rotatoria:

En la Figura 4.7 se puede observar el análisis de un corte de sección del resultado final de los diferentes componentes del diseño 3D ensamblados. Cabe destacar que se ha recreado el objeto de un motor con sus medidas típicas para ubicar de la mejor forma posibles los diferentes componentes.

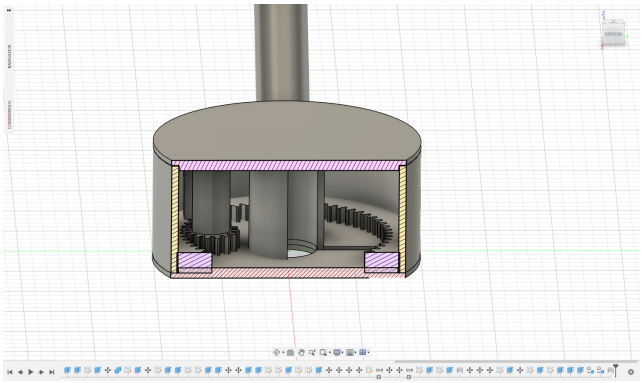


Figura 4.7: Análisis de sección

Para comunicar este sistema de captura de imágenes mediante el móvil situado en el brazo rotatorio al PC o portátil donde residirá el programa principal, se ha desarrollado una aplicación móvil para Android. Esta aplicación cuenta con la opción de cargar un vídeo de la galería y posteriormente, mediante la IP del servidor, mandar una petición POST con el vídeo en el request data al endpoint (`/upload-media`) de la API RESTful de `Flask` (ver la GUI en la Figura 4.8).

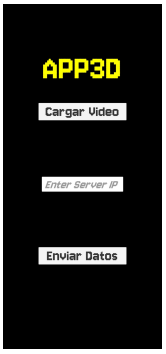


Figura 4.8: GUI del apk android

Antes de enviar~~Previamente a mandar~~ el vídeo, el servidor debe estar en funcionamiento como se observa en la Figura 4.9.

```
(.venv) PS C:\Users\leuis\Desktop\TFG\app>
 *  Historial restaurado

 * Serving Flask app 'Server'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://192.168.1.33:5000
Press CTRL+C to quit
```

Figura 4.9: Servidor Flask en ejecución

4.3. Segmentación de las imágenes

Una vez se ha obtenido el vídeo del objeto de la realidad, se procede a la extracción de sus fotogramas. Para ello, como se ha comentado en el diagrama mostrado en la Figura 4.1, se utiliza la herramienta llamada **`FFMPEG`**.

Se ha seleccionado para el uso en este proyecto debido a las siguientes razones. Por una parte, esta herramienta está altamente optimizada para el rendimiento, utilizando técnicas avanzadas para procesar archivos multimedia rápidamente y con alta calidad. Por otra parte, su versatilidad, ya que soporta una amplia variedad de formatos y codecs, lo que le permite manejar prácticamente cualquier tipo de archivo multimedia.

En el proyecto, esta herramienta se utiliza para la función de segmentación del vídeo en fotogramas, pudiendo exportar estos en diferentes extensiones: PNG, JPG, TIFF y BMP.

La función de segmentación únicamente recibe dos parámetros de entrada:

- **FPS:** Este indica la cantidad de fotogramas que se extraerán del vídeo por segundo.
- **Extensión de las imágenes de salida:** Como su propio nombre índica, este parámetro en que formato se van a exportar las imágenes. Esto puede ser útil dependiendo del espacio que se tenga disponible en el equipo, y a la hora de testear el nivel de rendimiento en función del formato.

4.4. Tratamiento de las imágenes

En esta fase del pipeline, se ha optado por la utilización de funciones propias basadas en librerías disponibles en Python, como son `CV2` y `PIL`.

A continuación se expondrán en más profundidad las funciones disponibles en la aplicación y las técnicas en las que se basan:

- **Detección y eliminación de imágenes borrosas:** Antes de entrar en detalle sobre el funcionamiento interno de la función, debe darse el contexto sobre la siguiente pregunta:

  ¿En qué consiste una imagen borrosa?

  La característica que define que una imagen sea borrosa es la pérdida de detalles y alta frecuencia. Técnicamente, el desenfoque se manifiesta como una reducción en la nitidez de los bordes y una disminución en el contraste de las texturas.

  Funcionamiento interno:

  La función recibe dos argumentos como parámetros, la ruta de la imagen y un umbral de varianza, el cual debe superar la imagen para no ser borrada.

  Una vez recibido los parámetros, se lee la imagen en escala de grises, esto es debido a que como se ha comentado anteriormente con relación a una imagen sea borrosa o no, es la reducción de la nitidez en los bordes lo que provoca una perdida de altas frecuencias. Por lo tanto, no es necesario cargar la imagen en formato RGB, optimizando de este modo la memoria RAM requerida por el programa.

A continuación, se le aplica a la imagen un filtro Laplaciano personalizado:

$$\begin{matrix}[[1 & 1 & 1] \\ [1 & -8 & 1], \\ [1 & 1 & 1]]\end{matrix} \tag{4.1}$$

El cual, aplicado a la imagen, realiza la función:

$$\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{4.2}$$

Esta función realiza una convolución a la imagen para obtener la segunda derivada de la intensidad de la imagen en todas las direcciones. Esto se utiliza para la detección de bordes en las imágenes, dado que el cambio brusco de intensidad se manifiesta como un máximo o un mínimo locales en la segunda derivada.

Posteriormente, se calcula la varianza estadística que indica cuánto se dispersan los valores de una serie de datos respecto a su media.

La función de la varianza $\mathrm{Var}(L)$ se define como:

$$\mathrm{Var}(L) = \frac{1}{N}\sum_{i=1}^{N}(L_i - \mu_L)^2 \tag{4.3}$$

donde:

- N es el número total de píxeles.
- L_i representa el valor i-ésimo de L.
- μ_L es la media de L, definida como $\mu_L = \frac{1}{N}\sum_{i=1}^{N} L_i$.

Como se ha comentado anteriormente, si la varianza supera el umbral establecido a través del parámetro "Threshold", la imagen no será eliminada, dado que esta contiene altas frecuencias, lo que indica un grado suficiente de nitidez.

- **Modificación del tamaño de las imágenes:** Se carga la imagen a través de la ruta indicada por parámetro y se redimensiona hasta el tamaño especificado.
- **Modificación de las características de la imagen:** Las características primarias de las imágenes que se pueden modificar en el programa son las siguientes:
  - **Brillo:** Intensidad promedio de los píxeles.
  - **Color:** Información sobre los diferentes canales de color.

- **Contraste:** Diferencia en intensidades o colores.

Para realizar la modificación de estas características se utiliza la librería `PIL`, concretamente el módulo `ImageEnhance`, esta proporciona los métodos necesarios para dicha tarea.

- **Reducción de ruido:** Esta función está disponible para corregir posibles errores de obturación que puedan ocurrir al capturar una imagen con un dispositivo móvil. Su propósito es eliminar o reducir al mínimo estos errores para mejorar la calidad general de la imagen.

  Funcionamiento interno:

  Mediante el uso de una matriz de convolución (kernel) y un solo parámetro de entrada n (radio). Se realiza la operación de convolución para todo píxel (i, j) en la imagen I con el kernel G se define como:

  Siendo G matriz cuadrada de tamaño $(2 * radio + 1) * (2 * radio + 1)$:

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{4.4}$$

$$I'(i,j) = \sum_{k=-n}^{n} \sum_{l=-n}^{n} I(i+k, j+l) \cdot G(k,l) \tag{4.5}$$

  donde:

  - I' es la imagen resultante.
  - I es la imagen original.
  - G es el kernel Gaussiano.
  - n es el radio del kernel.

- **Añadir ruido de tipo Gaussiano y Sal y Pimienta**: También se ha añadido la funcionalidad de poder añadir de forma artificial ruido a la imagen, esta puede ser interesante de cara a la investigación para verificar su efecto en la representación del objeto 3D.

  Ruido Gaussiano:

  1. Se genera una imagen del mismo tamaño que la original, que será usada como una máscara. Generalmente, la máscara es conocida como kernel o filtro, y consiste en una matriz bidimensional que se utiliza para realizar operaciones locales en una imagen, como el suavizado, el realce, la detección de bordes, entre otros. La máscara se aplica deslizándola sobre la imagen y realizando ciertas operaciones en cada posición de la imagen.

2. Los valores de los píxeles de la máscara siguen una distribución normal con media 0 y desviación estándar igual a la intensidad elegida como parámetro de entrada.

3. Finalmente se unen las dos imágenes sumando el valor de cada píxel.

Ruido Sal y Pimienta:

1. Se genera una imagen del mismo tamaño que la original, que será usada como máscara.

2. Los valores de los píxeles de la máscara siguen valores aleatorios uniformemente distribuidos entre 0 y 1.

3. Finalmente se unen las dos imágenes dependiendo del valor obtenido en la máscara inicial sobre el píxel (x_i, y_n) se aplica el valor 0 o 255 con probabilidad $ruido < \frac{intensidad}{2}$ y $ruido > \frac{1-intensidad}{2}$ respectivamente.

4.5. Extracción de Características

Para la extracción de características de la escena, finalmente se va a utilizar la herramienta `COLMAP` (documentación disponible [9]), esta es una herramienta avanzada y ampliamente utilizada en el campo de la visión por computadora para la reconstrucción 3D y la estimación de estructura a partir de movimiento (SfM, por sus siglas en inglés). También destaca por su precisión y eficiencia en la creación de modelos tridimensionales detallados a partir de colecciones de imágenes.

En este caso particular se utilizará para procesar los datos de las imágenes. En primer lugar, se extraen los puntos clave. A continuación, mapear las coincidencias de puntos clave entre frames y finalmente estimar la ubicación de las diferentes cámaras dentro de la escena 3D.

A modo de introducción la aplicación utiliza la versión de COLMAP 3.9.1 (adaptada para Cuda), está instalada en el repositorio del programa, concretamente en la carpeta `.\Scripts\external`, si este por algún motivo no se ha copiado, la aplicación directamente ejecutará un script para su descarga en el sistema (script implementado solo para Windows).

Funcionamiento interno:

1. **Ejecución del feature extractor de COLMAP:** Este proceso se lleva a cabo a través del algoritmo de visión por computadora SIFT [**?**] (Scale-Invariant Feature Transform), con la finalidad de detectar y describir características locales en imágenes. Este algoritmo es extensamente utilizado por su capacidad para identificar características invariantes a la escala, rotación y cambios de iluminación en menor medida.

Todo esto hace que sea la mejor opción para intentar sobrellevar los diferentes errores que se introduzcan a lo largo de la ejecución del pipeline.

Este algoritmo encuentra y almacena en una base de datos SQLite creada automáticamente por `COLMAP`, los siguientes datos de cada imagen:

2. **Extracción de Keypoints y Descriptores:**

   a) **Construcción de la Pirámide de Escalas:** Se generan múltiples versiones de la imagen con diferentes resoluciones y se crean niveles con distinta escala. Cada nivel de la pirámide se suaviza utilizando un filtro gaussiano, como se observa en la Figura 4.10.

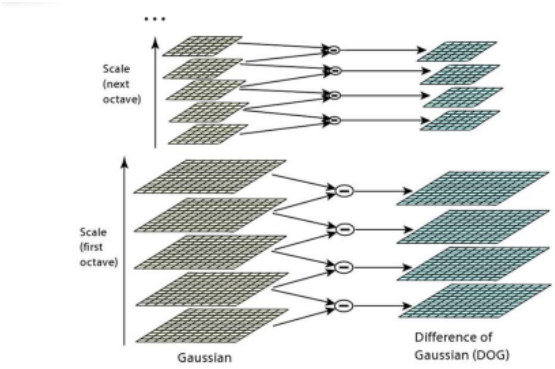


Figura 4.10: Ejemplo Pirámide de Escala. Fuente en documento.

   b) **Detección de Extremos Locales:** Se restan las imágenes suavizadas con el filtro Gaussiano, lo cual genera una aproximación a LoG (Laplacian of Gaussians). Esto indica los puntos clave que se identifican como máximos y mínimos locales, como se puede observar en la Figura 4.11.

Figura 4.11: Ejemplo Keypoints encontrados

   c) **Asignación de Orientación:** Cada punto clave se le asigna una orientación basada en la dirección dominante de los gradientes locales, lo que ayuda a mantener la invariancia a la rotación.

d) **Construcción de Descriptores:** Para cada punto clave, se toma una región de la imagen y se divide en una cuadrícula de 4x4 subregiones. Para cada subregión, se calcula un histograma (representación en la Figura 4.12) de orientaciones de gradiente (8 orientaciones). Esto da como resultado un vector de 128 dimensiones el cual tiene por defecto `COLMAP`.

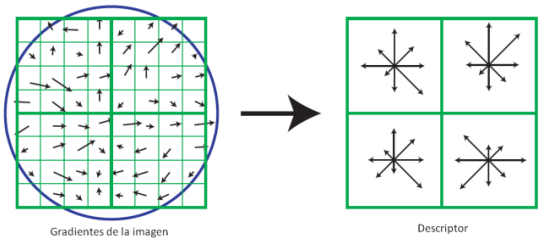


Figura 4.12: Codificación de los descriptores

3. Una vez extraídas las características de las imágenes, se procede a buscar los matches, o correspondencias, entre estas, según los descriptores identificados anteriormente.

4. Finalmente, se reconstruye/mapea el modelo 3D disperso del conjunto de datos mediante el uso de la técnica llamada SfM. Esto con la finalidad de estimar la geometría 3D de la escena y las posiciones de las cámaras a partir de correspondencias entre imágenes.

5. A continuación, se optimiza el resultado anterior, mediante el refinamiento de la estructura 3D y las poses de las cámaras. Este refinamiento consta de distintos pasos:

   a) Se toma la estructura 3D inicial y las poses de las cámaras estimadas por el mapeado inicial.

   b) Se utiliza la técnica de optimización no lineal para minimizar el error inicial de las proyecciones, es decir, la diferencia entre las posiciones proyectadas de los puntos 3D en las imágenes y sus posiciones observadas.

   c) Durante el proceso de ajuste, se optimizan simultáneamente los parámetros intrínsecos y extrínsecos de las cámaras, así como las coordenadas 3D de los puntos reconstruidos.

6. Para concluir con `COLMAP`, los datos calculados anteriormente (posición de las cámaras y datos de las imágenes), los cuales están almacenados en una base de datos SQLite creada automáticamente en el paso 1, son exportados al formato .txt.

7. Por último, mediante una función en Python, todos los archivos creados anteriormente (posición de las cámaras y datos de las imágenes) son condensados

en un fichero llamado **transforms.json**, este servirá como input en la reconstrucción 3D. Los pasos para condensar los datos son los siguientes:

a) **Lectura y procesamiento de las cámaras:**

- Se lee el archivo cameras.txt, se parsea cada línea para extraer información como el modelo de cámara, dimensiones, focales y coeficientes de distorsión y se almacena de forma temporal en un diccionario para ser utilizada más adelante.

b) **Lectura y procesamiento de las imágenes:**

- Se lee el archivo images.txt que contiene las poses de las cámaras (quaterniones y vectores de traslación) y el nombre de las imágenes.
- Se calcula la matriz de transformación de la cámara de cada imagen, usando la función `qvec2rotmat`, la cual convierte un quaternion en una matriz de rotación.
- Para cada imagen, se calcula la nitidez usando el método de la varianza Laplaciana. Esto se utiliza internamente en la red neuronal para ponderar positivamente los valores aportados por aquellas imágenes con mayor nitidez.

c) **Creación del archivo JSON:** Las matrices de transformación resultantes son serializadas en formato JSON y posteriormente guardadas en el archivo transforms.json, adjuntando con ellas, la ruta de su correspondiente imagen y los parámetros de las cámaras.

Adicionalmente, también se han implementado dos funciones para verificar la calidad de los resultados obtenidos en la extracción de características y coincidencias. Para ello, se consulta la base de datos SQLite creada durante el proceso de extracción de características anterior y se reconstruyen las imágenes utilizando los puntos clave y las coincidencias encontradas. Un ejemplo del resultado de esta visualización se puede ver en la Figura 4.13.

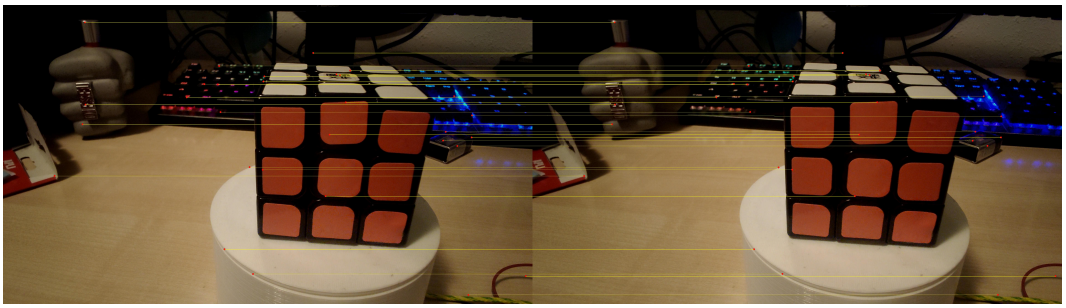


Figura 4.13: Ejemplo de coincidencias entre frames.

4.6. Reconstrucción 3D

Finalmente, para la reconstrucción 3D se ha optado por el uso de la herramienta **`Instant NGP`**. Esta elección se debe a su capacidad para trabajar con objetos mallados, lo que agiliza el desarrollo del software. Además, **`Instant NGP`** ofrece un rendimiento superior en la reconstrucción 3D en comparación con el enfoque de puntos gaussianos de la herramienta 3D Gaussian Splatting. Respecto a otras soluciones presentes en la actualidad, **`Instant NGP`** se caracteriza por su alta velocidad en la renderización de escenas, teniendo un balance calidad/rapidez muy superior, esto es causado por su nuevo enfoque de codificación hash multirresolución.

Llegados a este punto del pipeline, se dispone de la siguiente información condensada en un archivo JSON:

- Características intrínsecas de la cámara empleada para captar las imágenes. (Posición, Distorsiones, Centro de la imagen, Dimensión, etc.)
- AABB scale, parámetro especificado por el usuario, el cual es importante para la red, dado que indica la distancia de renderizado de la escena.
- Lista de los frames/imágenes que componen la escena, con su correspondiente información:
  - Ruta de la imagen
  - Varianza de la imagen (sharpness)
  - Matriz de transformación

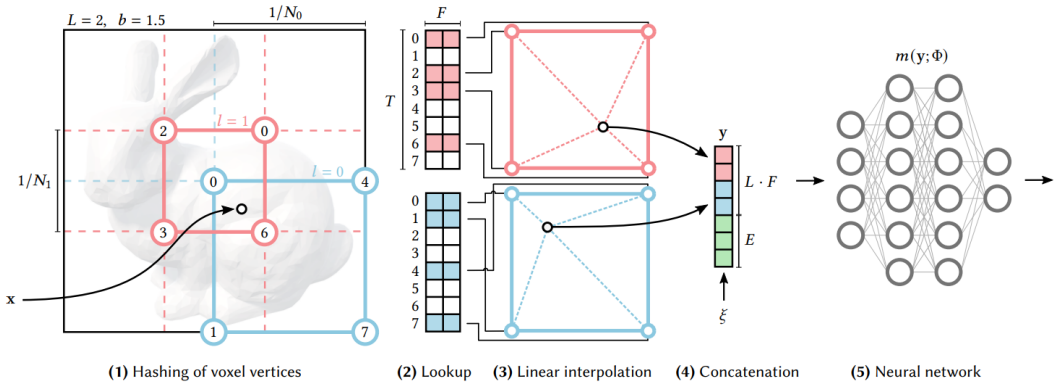


Figura 4.14: Ejemplo de codificación hash en infraestructura Instant NGP
- F = Dimensión del encoding (Por defecto 4)
- L = Número de resoluciones (Por defecto 16)
- E = Características auxiliares como el material
- T = Número máximo de entradas (Por defecto $2^{14} - 2^{24}$)
- N_{MIN} = Resolución más gruesa (Por defecto 16)
- N_{MAX} = Resolución más fina (Por defecto 512 - 524288)

Dados los diferentes parámetros existentes en la infraestructura interna de la aplicación, se presenta el siguiente flujo de ejecución ilustrado en la Figura 4.14. El flujo de ejecución es el siguiente:

1. Dada una coordenada x, del espacio 2D de la imagen, se enumeran los voxels de las diferentes resoluciones que lo rodean (paso (1) en la Figura 4.14).
2. A cada esquina del vóxel se le asigna un índice mediante el uso de una función de hashing.
3. Para cada índice se busca el vector de características correspondiente en la tabla hash, los valores iniciales de estos vectores son creados inicialmente de forma aleatoria (paso (2) en la Figura 4.14).
4. Se interpola linealmente según la posición relativa de x dentro del respectivo vóxel, teniendo en cuenta la distancia del punto respecto a las esquinas.
5. Se concatena cada entrada por nivel además de los inputs auxiliares (estos datos auxiliares dependen del enfoque en el que se use), esto produce el input $y \in \mathbb{R}^{\mathbb{L}\mathbb{F}+\mathbb{E}}$, para la red neuronal.
6. Para entrenar la codificación, los gradientes de pérdida se retropropagan a través de la MLP (ver (5) en la Figura 4.14), la concatenación (paso (4)), la interpolación lineal (paso (3)) y, a continuación, se acumulan en los vectores de características buscados.

Una vez expuesto el funcionamiento y enfoque de la codificación hash multirresolución, a continuación se describen los detalles de lo que se ha incluido en la aplicación final:

- La arquitectura hash multirresolución, presentada anteriormente.
- El resultado de la posición x del espacio 3D, junto con los parámetros de la codificación son pasados a la red MLP, donde produce una salida de densidad con 16 valores, donde estos representan la cantidad de material en el punto de entrada más características adicionales útiles para la siguiente etapa.
- El siguiente paso, es otra red MLP, la cual de entrada recibe la salida anterior, más la "Matriz de transformación". Esta matriz proporciona la dirección de la vista proyectada de los 16 coeficientes de entrada. Esta etapa produce el color (RGB) del píxel de entrada.

4.7. Suavizado de puntos/malla

Para la etapa de suavizado de la malla, se ha seleccionado la opción de utilizar el aplicativo **MeshLab**, dado que este cuenta con un módulo propio de Python, hecho que simplifica la tarea de implementación en el propio código del proyecto. En el código desarrollado, se han incluido las siguientes funciones simples para simplificar la malla:

- Eliminar las caras duplicadas.
- Eliminar las caras plegadas (se refiere a una situación geométrica donde los vértices que definen una cara, usualmente un triángulo, donde están dispuestos de tal manera que la cara no se encuentra en un plano único o se auto-intersecta).
- Eliminar las caras nulas.
- Elimina los vértices duplicados.
- Eliminar vértices no referenciados.

A continuación, también se han incluido funcionalidades más complejas, las cuales requieren de los siguientes parámetros de entrada para ser ejecutadas:

- Aplicar suavizado Taubin ([10]): El suavizado Taubin es una técnica de suavizado de superficies que preserva los detalles geométricos mientras reduce el ruido. Este método se configura con los siguientes parámetros:
  - **Lambda:** Es un factor positivo usado en la etapa de suavizado (o contracción) de la superficie.
  - **Mu:** Este es un factor negativo utilizado en una etapa posterior de descontracción.
  - **Pasos de refinado:** Taubin sugiere que valores de N suficientemente grandes producen una mejor eliminación de ruido, pero a costa de mayores recursos computacionales.
- Eliminar componentes aislados, determinado por el número de caras que contiene el componente.
- Cerrar posibles agujeros en la malla triangular. Este proceso está definido por dos parámetros:
  - **Tamaño máximo del agujero:** Número de vértices que componen el agujero.
  - **Longitud del borde:** La longitud objetivo del borde dentro del agujero.
- Fusionar vértices adyacentes. Esta función únicamente depende de un parámetro, que indica la distancia a la que debe estar los vértices para ser fusionados.

Adicionalmente, el programa desarrollado incluye un visor de mallas 3D integrado, implementado con el módulo `PyOpenGL`. Esta herramienta está diseñada para el desarrollar de aplicaciones gráficas en Python, permitiendo al usuario visualizar el resultado final tras la optimización de la malla.

4.8. Gráficos e historial

Con la finalidad de ayudar al usuario a analizar los datos obtenidos durante toda la ejecución del pipeline, se han incluido en la aplicación desarrollada, una serie de funciones adicionales:

- **Historial:** Los datos obtenidos se pueden guardar en un archivo JSON, que está disponible en cualquier momento para revisar experimentos anteriores.
- **Visualización de gráficas:** Con el respaldo del historial guardado, es posible cargar y visualizar las gráficas correspondientes a ejecuciones anteriores.
- **Configuración:** Toda la configuración realizada durante un pipeline puede ser guardada para aplicarla nuevamente en futuras iteraciones.

4.9. Diseño de la interfaz gráfica

Finalmente, se adjunta el prototipo o sketch inicial de la interfaz de usuario del programa final en la Figura 4.15.

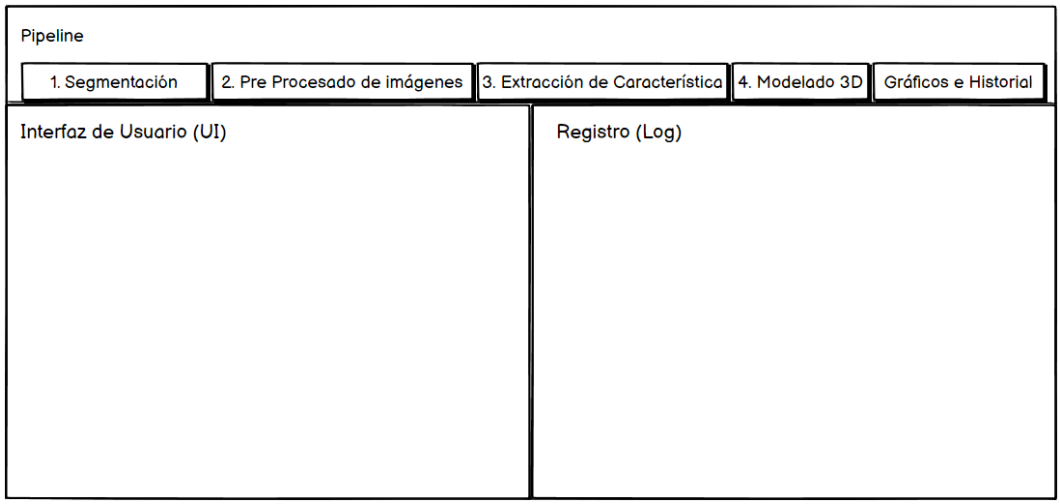


Figura 4.15: Sketch sobre la interfaz de usuario final.

Como se puede observar en la Figura 4.15, la interfaz cuenta con una sección independiente para cada fase del pipeline, donde tendrá sus correspondientes parámetros y configuraciones, además tal y como se ha especificado la sección de requerimientos 2.1, el usuario tendrá en todo momento una ventana a su derecha, indicando los avances, los errores y los parámetros mal configurados que vayan surgiendo a lo largo del proceso.

Capítulo 5

Simulaciones y resultados

En este capítulo se llevarán a cabo los experimentos necesarios para evaluar la viabilidad y el rendimiento del aplicativo desarrollado en el proyecto. Todo esto en función del objetivo principal descrito desde el inicio 1.2.

Además de analizar si el resultado final desarrollado cumple con los objetivos marcados en el principio, hay que verificar también los requerimientos funcionales de la sección 2.1.

5.1. Soporte de captura final

El diseño e impresión del soporte utilizado para la captura de vídeo en este capítulo, ha sido realizado desde cero en el proyecto. El resultado se muestra en la Figura 5.1.

Cuadro 5.1: Foto del prototipo construido para escanear la figura a través del móvil, que se situará en el extremo del brazo.

De forma complementaria a esto, se adjunta la URL del vídeo donde se puede ver mecanismo diseñado escaneando un objeto. Vídeo en Youtube

5.2. Aplicación final

A continuación en la Figura 5.1, el aspecto final que tiene el aplicativo desarrollado en el proyecto.

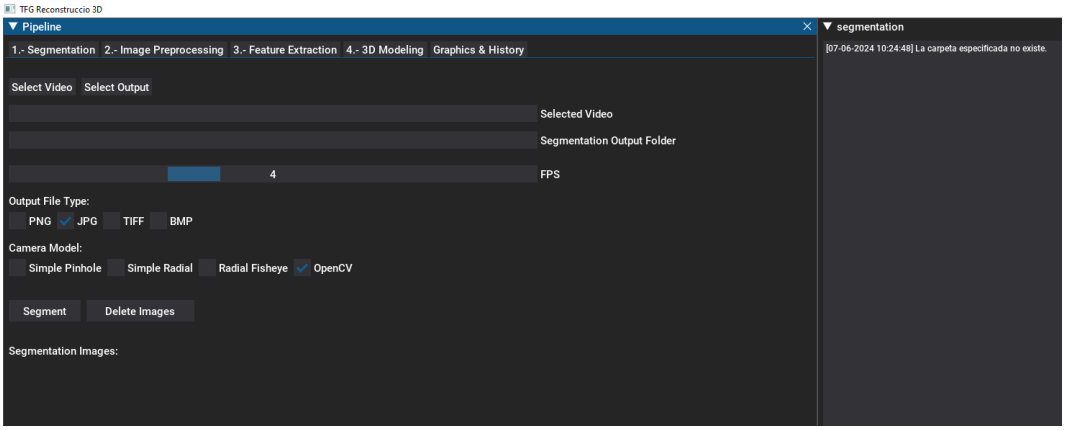


Figura 5.1: Imagen prototipo de escaneo final.

Para el desarrollo de la interfaz gráfica del aplicativo se ha utilizado la biblioteca de interfaces gráficas de `DearPyGUI` [6]. Esta biblioteca permite a los desarrolladores crear aplicaciones con interfaces modernas y funcionales utilizando un enfoque simple basado en Python.

Gracias al uso de esta herramienta se ha podido adaptar a la perfección el sketch inicial del programa 4.15.

5.3. Simulaciones

Debido al extenso nivel de parametrización presente en la aplicación desarrollada, resulta inviable cubrir todas las posibilidades en este capítulo de análisis de simulaciones y resultados. En su lugar, nos centraremos en comparar tres ejecuciones completas del pipeline con perfiles diferentes.

1. **Perfil bajo:** Este utilizará una configuración mínima (en requerimientos de hardware y espacio) y más veloz en tiempo de ejecución del pipeline.
2. **Perfil medio:** Este utilizará una configuración intermedia (en requerimientos de hardware y espacio) y media en tiempo de ejecución del pipeline.
3. **Perfil alto:** Este utilizará una configuración exigente (en requerimientos de hardware y espacio) y lenta en tiempo de ejecución del pipeline.

Hardware empleado para la ejecución de los pipelines:

- **Procesador:** Intel Core I7-7700 3.6GHz.
- **Disco duro:** Samsung 860 EVO Basic SSD 250GB SATA3
- **Memoria RAM:** G.Skill Ripjaws V Red DDR4 2400 PC4-19200 2x8GB CL15
- **Tarjeta Gráfica:** Gigabyte GeForce RTX 3060 GAMING OC 12GB GDDR6

Como base para todos los tests, se ha utilizado un mismo vídeo, este ha sido grabado utilizando la base automatizada explicada en apartados anteriores y mostrada en Figura 4.7.

5.4. Configuración del pipeline con perfil bajo

A continuación el resumen de todas las configuraciones de cada fase del pipeline, más el resultado de estas operaciones (tiempo de ejecución y alteraciones en los datos).

Configuración	**Valor**
FPS	2
Tipo de archivo frames	JPG
Modelo de cámara	OpenCV

Cuadro 5.2: Configuraciones para la etapa de Segmentación

Resultado Fase 1.
- *Duración del proceso: 27 segundos.*
- *Total frames creados: 168.*

Modificación aplicada	**Valor**
Redimensionado de imagen	1920x108

Cuadro 5.3: Modificaciones realizadas en la etapa de preprocesado de imágenes

Resultado Fase 2.
- *Duración de la etapa: 20 segundos.*

Configuración	Valor
Tipo de Reconocimiento	Sequential
Escala AABB	8

Cuadro 5.4: Configuraciones para la etapa de extracción de características

Resultado Fase 3.

- *Tiempo total de extracción: 8:56 min.*
- *Tiempo del renderizado en **Instant NGP**: 2 minutos.*
- *Tamaño del modelo en memoria de la GPU: 4.70 GB,*
- *Triángulos obtenidos en la malla: 2.933.508*

Filtro	Valor
Se aplican todos lo filtros simples	True

Cuadro 5.5: Filtros Aplicados a la malla resultante:

Resultado Fase 4.

- *Se han tardado 6 segundos en aplicar todos los filtros.*
- *Para cargar y visualizar la malla, el sistema tarda 30 segundos.*
- *Figura resultado mostrada en la Figura 5.2*

Resultados de la ejecución del perfil bajo

Resultado Final.

- *Peso total de los archivos generados para la creación del modelo 3D completo en disco 412 MB.*

Figura 5.2: Resultado objeto 3D aplicando perfil de configuración bajo

5.5. Configuración del pipeline con perfil medio

A continuación el resumen de todas las configuraciones de cada fase del pipeline, más el resultado de estas operaciones (tiempo de ejecución y alteraciones en los datos).

Configuración	**Valor**
FPS	4
Tipo de archivo frames	PNG
Modelo de cámara	OpenCV

Cuadro 5.6: Configuraciones para la etapa de Segmentación

Resultado Fase 1.

- *Duración del proceso 1:02 minutos.*
- *Total frames creados 336.*

Modificación aplicada	**Valor**
Eliminar Imágenes Borrosas	15
Redimensionado de imagen	1920x108

Cuadro 5.7: Modificaciones realizadas en la etapa de preprocesado de imágenes

Resultado Fase 2.

- *Duración de la etapa 1:48 minutos.*
- *Han quedado un total de 242 frames*

Configuración	**Valor**
Tipo de Reconocimiento	Sequential
Escala AABB	16

Cuadro 5.8: Configuraciones para la etapa de extracción de características

Resultado Fase 3.

- *Tiempo total de extracción 17:38 minutos.*
- *Tiempo del renderizado en **Instant NGP** 4 min.*
- *Tamaño del modelo en memoria de la GPU 5.15 GB,*
- *Triángulos obtenidos en la malla 3.501.082*

Filtro	Valor
Se aplican todos lo filtros simples	True
Suavizado Taubin	(Lambda=0.5, Mu=-0.53, Pasos=10)
Cerrar Agujeros	(Tamaño máximo=50, Longitud esquinas 4.0)

Cuadro 5.9: Filtros Aplicados a la malla resultante:

Resultado Fase 4.

- *Se han tardado 45 segundos en aplicar todos los filtros.*
- *Para cargar y visualizar la malla, el sistema tarda 40 segundos.*
- *Los resultados obtenidos se muestran en la Figura 5.3*

Resultados de la ejecución del perfil medio

Resultado Final.

- *Peso total de los archivos generados para la creación del modelo 3D completo en disco 985 MB.*

Figura 5.3: Resultado objeto 3D aplicando perfil de configuración medio

5.6. Configuración del pipeline con perfil alto

A continuación el resumen de todas las configuraciones de cada fase del pipeline, más el resultado de estas operaciones (tiempo de ejecución y alteraciones en los datos).

Configuración	**Valor**
FPS	6
Tipo de archivo frames	BMP
Modelo de cámara	OpenCV

Cuadro 5.10: Configuraciones para la etapa de Segmentación

Resultado Fase 1.

- *Duración del proceso 3:40 minutos.*
- *Total frames creados 505.*

Modificación aplicada	**Valor**
Eliminar Imágenes Borrosas	16
Redimensionado de imagen	1920x1080

Cuadro 5.11: Modificaciones realizadas en la etapa de preprocesado de imágenes

Resultado Fase 2.

- *Duración de la etapa 2:40 minutos.*
- *Han quedado un total de 336 frames.*

Configuración	**Valor**
Tipo de Reconocimiento	Exhaustive
Escala AABB	16

Cuadro 5.12: Configuraciones para la etapa de extracción de características

Resultado Fase 3.

- *Tiempo total de extracción 51:10 minutos.*
- *Tiempo del renderizado en **Instant NGP** 9 min.*
- *Tamaño del modelo en memoria de la GPU 5.83 GB,*
- *Triángulos obtenidos en la malla 2.776.649.*

Filtro	Valor
Se aplican todos lo filtros simples	True
Suavizado Taubin	(Lambda=0.5, Mu=-0.53, Pasos=15)
Eliminar componentes aisladas	(Numero Componentes = 50)
Cerrar Agujeros	(Tamaño máximo=50, Longitud esquinas 4.0)

Cuadro 5.13: Filtros Aplicados a la malla resultante:

Resultado Fase 4.

- *Se han tardado 50 segundos en aplicar todos los filtros.*
- *Para cargar y visualizar la malla, el sistema tarda 34 segundos.*
- *Los resultados se muestran en la Figura 5.4.*

Resultados de la ejecución del perfil alto

Resultado Final.

- *Peso total de los archivos generados para la creación del modelo 3D completo en disco 2.50 GB.*

Figura 5.4: Resultado objeto 3D aplicando perfil de configuración alto

5.7. Discusión y recomendaciones finales

A la vista de los resultados (presentes en las Figuras 5.2, 5.3, 5.4 y 5.5). Se describe una configuración óptima con la que se consigue el siguiente resultado.

A modo de ayuda, para iniciarse en el proceso de ejecución y aportar conocimiento sobre los resultados obtenidos, se propone la siguiente ejecución óptima:

Como se puede observar en la siguiente Tabla 5.14, se ha optado por el uso de un formato sin compresión como `PNG`, para evitar perdida de detalle, además el número de `FPS` está elegido para obtener entre 120 - 170 frames. Este es un número ideal para aportar suficiente información a la red de **Instant NGP**, pero sin añadir ruido de más. Por último, la configuración del tipo de cámara es `OpenCV`, ya que se han hecho todas las imágenes con la misma cámara, por lo tanto, se conocen los parámetros de configuración de esta.

Configuración	**Valor**
FPS	2
Tipo de archivo frames	PNG
Modelo de cámara	OpenCV

Cuadro 5.14: Configuraciones para la etapa de Segmentación

Resultado Fase 1.

- *Duración del proceso 41 segundos.*
- *Total frames creados 168.*

Modificación aplicada	**Valor**
Ajustar características imagen	Brillo 1.10, Contraste 1.2 y Color 1.2
Redimensionado de imagen	1920x1080

Cuadro 5.15: Modificaciones realizadas en la etapa de preprocesado de imágenes

Al tener ya bastante luminosidad el vídeo inicial el factor lo configuramos un poco más bajo, los demás se suben el mismo factor (suavemente), para favorecer la nitidez, al existir más contraste y cambio de color. Además, a través de numerosos tests, se ha comprobado que el uso de la resolución 1920x1080, tiene el equilibrio perfecto, entre calidad del modelo y velocidad de ejecución.

Resultado Fase 2.

- *Duración de la etapa: 2 minutos.*

Configuración	Valor
Tipo de Reconocimiento	Sequential
Escala AABB	8

Cuadro 5.16: Configuraciones para la etapa de extracción de características

Al extraer y enumerar secuencialmente las imágenes del vídeo, no es necesario utilizar una configuración exhaustiva, ya que esta requiere mucho más tiempo de ejecución. Además, dado que el objeto escaneado está en primer plano, una escala AABB de 8 es suficiente, lo que reduce la memoria GPU requerida.

Resultado Fase 3.

- *Tiempo total de extracción 9:25 minutos. Tiempo del renderizado en* ***Instant NGP*** *3 min.*
- *Tamaño del modelo en memoria de la GPU 4.68 GB.*
- *Triángulos obtenidos en la malla 3.271.748*

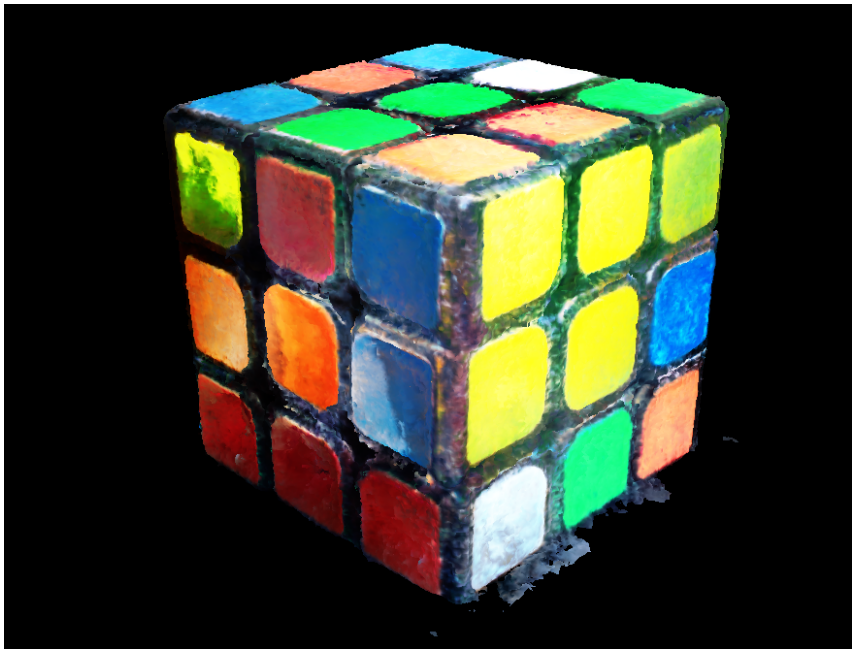


Figura 5.5: Resultado objeto 3D aplicando perfil de configuración óptimo

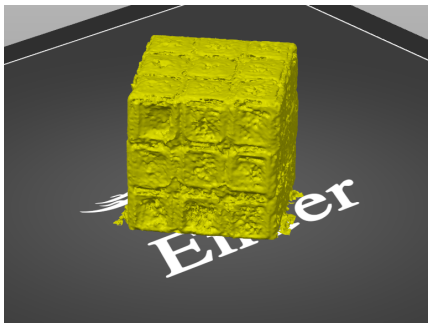
Resultado Fase 4.

- *Se han tardado 50 segundos en aplicar todos los filtros.*
- *Para cargar y visualizar la malla, el sistema tarda 34 segundos.*
- *El resultado obtenido se puede ver en la Figura 5.5.*

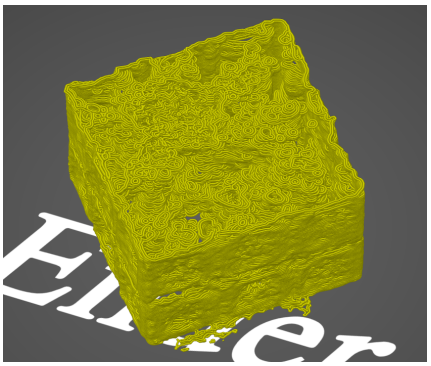
A través del vídeo en funcionamiento del mecanismo Youtube, se puede ver como se han captado las imágenes utilizadas en todas las ejecuciones.

> **Ayuda 1**. *En todas las simulaciones anteriores se ha dejado la configuración interna de* **`Instant NGP`** *por defecto, ya que al realizar tests con diferentes valores y valorando los resultados se ha llegado a la conclusión, que por defecto los valores están balanceados correctamente.*

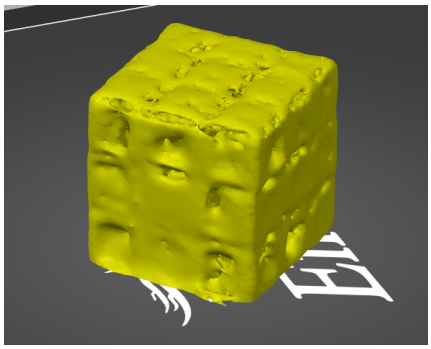
Tras una fase de optimización de la malla de salida de **`Instant NGP`**, se obtienen los siguientes resultados:



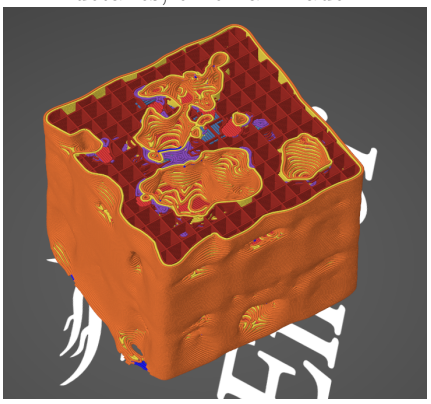
(a) Malla refinada manteniendo detalles



(b) Malla refinada manteniendo detalles, en el laminador



(c) Malla refinada para impresión, sin detalles



(d) Malla refinada para impresión, en el laminador

Cuadro 5.17: Conclusiones finales de la simulación

Como se puede observar en las imágenes anteriores 5.17, dependiendo de la calidad del modelo (a) (c), se puede imposibilitar su impresión 3D (b), ya que si este tiene una alta cantidad de ruido en su interior, el laminador no es capaz de traducir la malla a código G-Code. Esto puede solucionarse con una fase de refinado más intensa, como se puede observar en la imagen (c), aunque esto puede generar una falta de detalle y a su vez magnificar errores presentes en el modelo de salida de la red neuronal.

Capítulo 6

Conclusiones y trabajo futuro

6.1. Conclusiones

Para concluir, en la última sección del proyecto, se revisarán los objetivos establecidos al principio del documento (ver Sección 2.1), con el fin de evaluar si han sido alcanzados o resueltos.

En primer lugar, tenemos la revisión del estado del arte, con la finalidad de conseguir el objetivo de representar un objeto 3D físico, a un modelo virtual y posteriormente realizar su impresión. Durante el proyecto se han analizado distintas opciones, comparándolas entre sí, con sus ventajas e inconvenientes. Este análisis ha servido para determinar la solución planteada en el proyecto. En el punto actual del sistema desarrollado, además, permite comparar distintas aproximaciones de las estudiadas en el análisis y distintas configuraciones para validarlas. Sin embargo, la calidad del modelo 3D final aún requiere mejoras, principalmente porque las herramientas actuales de renderizado todavía necesitan avanzar.

Otro objetivo propuesto era realizar pruebas con diferentes tecnologías para seleccionar las más adecuadas según los requerimientos del proyecto (ver Sección 2.1). Este objetivo también se ha logrado, ya que a lo largo del proyecto se han presentado numerosas técnicas para cada una de las fases del pipeline, destacando las ventajas y desventajas de cada una.

A continuación, se expuso el objetivo de encontrar una herramienta capaz de refinar la malla obtenida. Con el uso de las funciones implementadas mediante el módulo de `Meshlab`, se puede refinar adecuadamente cualquier malla sin duda alguna. Sin embargo, existen limitaciones debido a la falta de una interfaz gráfica interactiva, lo que impide el refinamiento milimétrico de la malla y la selección de caras para su eliminación. No obstante, en términos generales, se puede lograr un buen resultado.

El objetivo de agilizar el proceso de pruebas y análisis de resultados ha sido alcanzado gracias a las funciones implementadas de visualización de resultados mediante gráficas o mapas de calor, además del sistema de guardado de configuración, lo que agiliza aún más la ejecución del pipeline.

Finalmente, el objetivo de realizar pruebas para determinar una configuración óptima y ofrecer recomendaciones generales también se ha alcanzado en la última sección del proyecto (ver Sección 5). En esta sección, además de comparar múltiples ejecuciones con diferentes parámetros, se presenta una solución que equilibra calidad y tiempo de ejecución.

6.2. Trabajo Futuro

Finalmente, para concluir con el proyecto se expondrán las tareas o mejoras que han quedado pendientes de implementación.

En primer lugar, el proyecto ha integrado una única red neuronal, aunque se podrían añadir otras fácilmente en el código desarrollado. Dados los requerimientos del proyecto, se ha optado por utilizar una red neuronal ligera. Sin embargo, actualmente existen alternativas más potentes que demandan hardware más avanzado. Un ejemplo de estas herramientas es `Neuralangelo`. Este proyecto surgió un año después de `Instant NGP` y ofrece una calidad de resultados superior. Sin embargo, como se mencionó anteriormente, requiere más recursos para su ejecución. Aunque modificando ciertas configuraciones avanzadas, se podría adaptar su uso para un público más general.

Finalmente, cabe destacar una funcionalidad que se podría añadir al proyecto actual y que podría suponer un mejor rendimiento del aplicativo, es un sistema temporizado para la captura de imágenes de la escena 3D. Utilizando la funcionalidad Bluetooth de la Raspberry Pi 3 y modificando la APK creada en el proyecto, se podría desarrollar un sistema que capture imágenes de forma secuencial, deteniendo el mecanismo de rotación en el momento adecuado. Esto evitaría la necesidad de realizar el paso de segmentación, resultando en imágenes de mayor calidad y, por ende, en una mejor renderización del objeto 3D.

Apéndice A

Manual técnico

En este apéndice de la memoria se recogen y se describen los diferentes tecnicismos utilizados en el proyecto. Además, se podrán encontrar las versiones de software utilizadas para crear todo el código en su conjunto, así como las dependencias de este. Finalmente, está el apartado dedicado a la explicación de como instalar y ejecutar el programa creado.

A.1. Nomenclatura

- **Convolución**: En el contexto de la visión artificial, la convolución es una operación matemática fundamental que se utiliza en el procesamiento de imágenes para extraer características y aplicar filtros.
- **Histograma**: Es una representación gráfica de la distribución de los valores de una variable en forma de barras.
- **Request**: Es una solicitud de información o acción enviada a un servidor, generalmente a través de un protocolo de comunicación como HTTP.
- **Multi-View Stereo (MVS)**: Es una técnica de visión por computadora que reconstruye modelos 3D a partir de múltiples imágenes tomadas desde diferentes puntos de vista.
- **Ruta de la imagen**: Es la ubicación o dirección en la que se encuentra almacenada una imagen en un sistema de archivos.
- **Rastericación**: Es el proceso de convertir datos vectoriales en imágenes rasterizadas, que consisten en una cuadrícula de píxeles.
- **“Structure from Motion"(SfM)**: Es una técnica de visión por computadora que reconstruye la estructura tridimensional de un objeto o escena a partir de múltiples imágenes bidimensionales.

- **Stochastic Gradient Descent**: Es un algoritmo de optimización utilizado para entrenar modelos de aprendizaje automático mediante la minimización de una función de costo.
- **Red neuronal**: Es un modelo computacional que emula el sistema nervioso biológico utilizado en el aprendizaje automático y la inteligencia artificial.
- **Densificación automatizada**: Es un proceso automatizado que aumenta la densidad de puntos en una nube de puntos tridimensional.
- **Tabla hash multirresolución**: Es una estructura de datos que utiliza una tabla hash para almacenar y acceder a datos de manera eficiente, con diferentes niveles de granularidad.
- **Quaternion**: Es una representación matemática de orientación en el espacio tridimensional que se utiliza en gráficos por computadora y en robótica.
- **Vector de traslación**: Es un vector que representa el desplazamiento de un objeto en el espacio tridimensional.

A.2. Versiones de Software

Versiones de los diferentes software utilizados durante todo el proyecto.

- Entorno de Python (.venv): 3.11.8
- FFMPEG: 5.1.2
- COLMAP: 3.9.1
- PrusaSlicer: 2.7.4
- Insant NGP: RTX 3000 Series
- Visual Studio Code: 1.90

A.3. Dependencias

Las dependencias necesarias para poder utilizar el programa del proyecto, son las siguientes:

- Disponer de una tarjeta gráfica NVIDIA.
- Tener un entorno de Python versión 3.11.8
- Tener instalado el ejecutable de **Instant NGP**, disponible para las siguientes versiones de la tarjeta gráfica:

  Si se dispone de una Tarjeta gráfica GTX 1000 series: Link

Si se dispone de una Tarjeta gráfica GTX 2000 series: Link

Si se dispone de una Tarjeta gráfica GTX 3000 series o superior: Link

A.4. Configuración

Para poder empezar a utilizar el programa, simplemente hay que seguir los siguientes pasos:

Crear un entorno de Python nuevo, dentro de la carpeta del repositorio descargado de **Github**:

```
python -m venv .venv
```

Seguidamente activamos el entorno creado, con el comando:

```
.\.venv\Scripts\activate
```

Finalmente instalar las dependencias del repositorio utilizando el comando:

```
pip install -r requirements.txt
```

Una vez todo listo puede ejecutarse el programa, este debe ejecutarse dentro de la estructura del proyecto (al mismo nivel que el fichero main.py) mediante el comando:

```
python .\main.py -r <Ruta que contiene el .exe de Insant NGP>
```

Apéndice B

Manual de Usuario

En este último capítulo del apéndice, se explica de forma detallada como funciona el software desarrollado. Además de ofrecer consejos de como ser utilizado.

B.1. Fase 1: Segmentación de imágenes

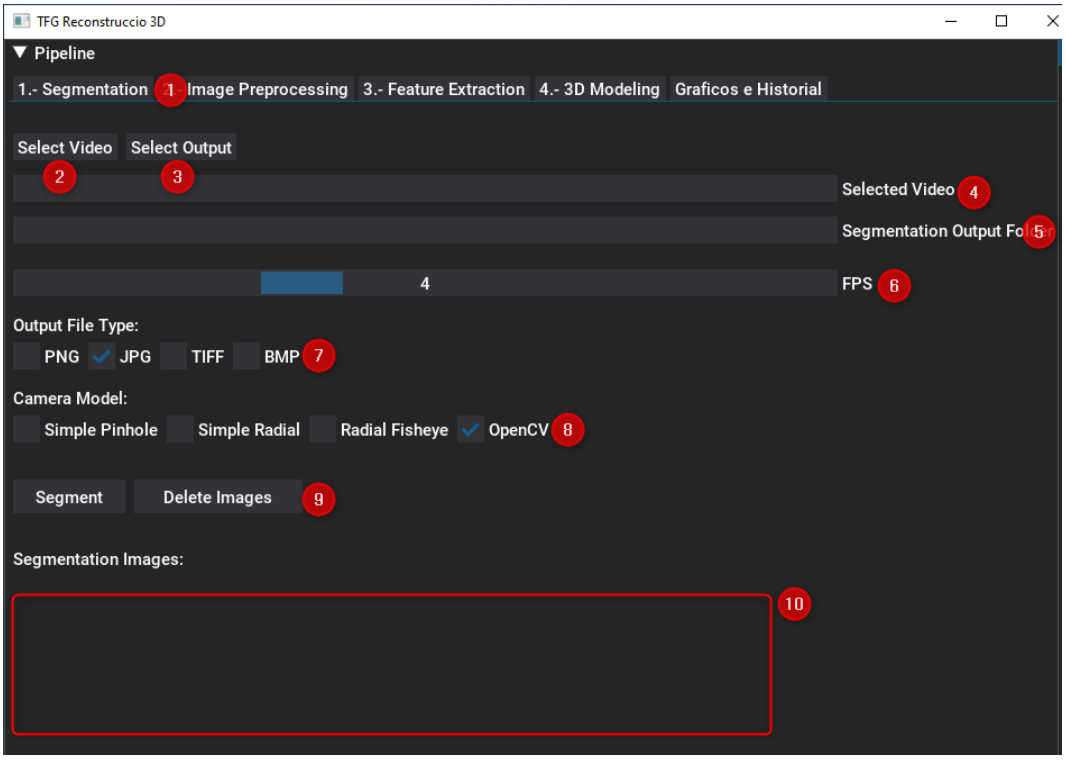


Figura B.1: GUI del Proceso de Segmentación

Flujo de trabajo de la Figura B.1:

1. Por defecto, el programa se inicia en la pestaña de segmentación, como su propio nombre indica, es donde se realiza el paso inicial de segmentar el vídeo de partida.

2. En primera instancia debe seleccionarse el vídeo original, esto se hace utilizando el botón marcado. El selector está configurado solamente subir archivos *.mp4, *.mov, *.avi.

3. Subido el vídeo, debe seleccionarse una carpeta destino donde se guardaran los frames resultantes del proceso. El selector está configurado exclusivamente para seleccionar carpetas.

4. Aquí se indica la ruta del vídeo seleccionado.

5. Aquí se indica la ruta de la carpeta de salida seleccionada.

6. Mediante el slider se selecciona el número de frames por segundo a los que se pasara el vídeo, a mayor número, más fotogramas saldrán. Los fotogramas de salida, automáticamente se adaptan a un formato de color de 24 bits máximo de profundidad, esto es debido a que la herramienta COLMAP no soporta una profundidad mayor.

7. Dependiendo de si interesa más una salida ligera, pero con más perdida, o viceversa, se puede elegir el formato de los frames resultantes.

8. En este apartado se elige la configuración de la cámara empleada para captar las imágenes o el vídeo.

    - **Simple Pinhole:** Utilizar este modelo de cámara, si sus imágenes no están distorsionadas a priori. Estos utilizan uno y dos parámetros de distancia focal, respectivamente.

    - **Simple Radial:** Este debería ser el modelo de cámara de elección, si los intrínsecos son desconocidos y cada imagen tiene una calibración de cámara diferente, por ejemplo, en el caso de fotos de Internet. Este es una version simplificada del modelo OpenCV

    - **Radial Fisheye:** Este modelo de cámara es para las lentes de ojo de pez.

    - **OpenCV:** Utilizar este modelo de cámara si se conoce a priori los parámetros de calibración.

9. Si todo lo anterior ha sido configurado adecuadamente, puede iniciarse el proceso de segmentación del vídeo utilizando el botón "*Segment*". En el caso de que el número de frames no haya sido el adecuado o las imágenes resultantes tienen algún defecto, pueden borrarse utilizando el botón "*Delete Images*".

10. Finalizado el proceso de segmentación, automáticamente se cargarán los frames en el visor, para que se pueda revisar la calidad de estos.

B.2. Fase 2: Preprocesamiento de las imágenes

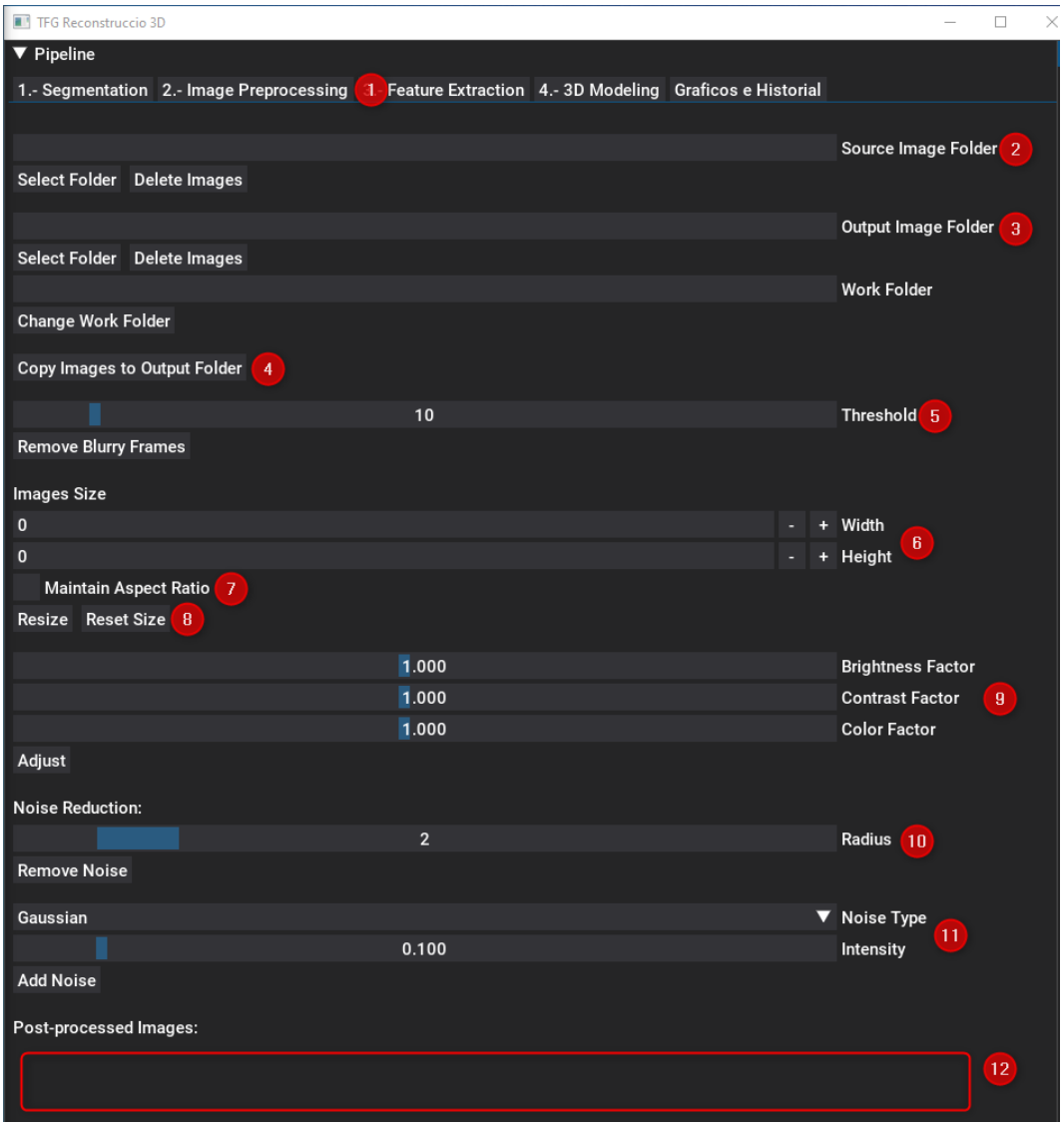


Figura B.2: GUI del Proceso de Preprocessing

Flujo de trabajo de la Figura B.2:

1. Terminado el proceso de segmentación, se puede pasar a la siguiente pestaña, en esta se podrán modificar las características de las imágenes resultantes.

2. Si en el paso anterior se configuró la carpeta de output, en esta pestaña se configurará automáticamente como "*Source Image Folder*".

3. A continuación, debe configurarse la carpeta de salida para la modificación de las imágenes, la idea fundamental de trabajo, es tener una carpeta origen donde las imágenes se utilizaran de base y las modificaciones se realizaran sobre la carpeta de salida.

4. Para inicializar el proceso deben copiarse las imágenes de origen en la carpeta de salida.

5. Esta primera función tiene la finalidad de eliminar aquellas imágenes más borrosas con el fin de eliminar ruido. Como mayor sea el umbral seleccionado, mayor serán las imágenes eliminadas.

6. La siguiente función sirve para redimensionar el tamaño de la imagen.

7. Está la opción de mantener el ratio original para no crear deformaciones.

8. Además, en caso de ser necesario, se puede resetear el tamaño original de las imágenes.

9. También existe la posibilidad de modificar las componentes básicas de las imágenes, entre estas son: Factor de brillo, Factor de contraste y Factor de color.

10. En caso de existir imágenes con ruido, se puede aplicar una reducción de este, mediante el uso de filtros gaussianos.

11. Finalmente, por temas de investigación también se ha añadido la función de insertar ruido, ya sea ruido gaussiano o de tipo sal y pimienta.

12. Cada vez que se modifiquen las imágenes se irán reflejando los cambios en su correspondiente área.

B.3. Fase 3: Extracción de características

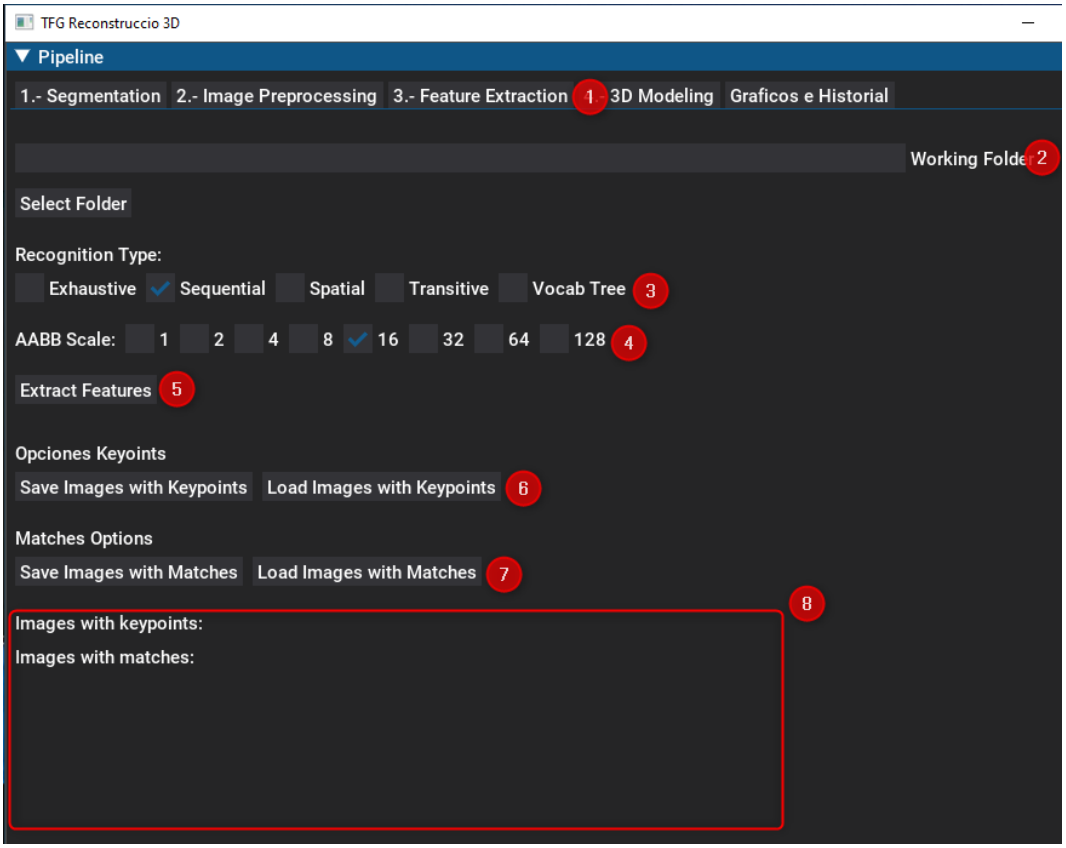


Figura B.3: GUI del Proceso de Feature Extraction

Flujo de trabajo de la Figura B.3:

1. Pre procesadas todas las imágenes puede iniciarse la extracción de las características de la escena.

2. Debe configurarse el entorno de trabajo donde se guardaran todos los datos de la extracción. Se recomienda que el directorio de trabajo este dentro de la carpeta de salida de las imágenes preprocesadas, de este modo cada test queda recogido en carpetas distintas.

3. En este punto, se debe seleccionar el modo de reconocimiento de las imágenes, esto define en como el proceso identificara los frames correlativos para representar la escena.

    - **Exhaustive:** Si el número de imágenes del conjunto de datos es relativamente bajo 1 - 200, este modo de comparación debería ser lo suficientemente rápido y ofrece los mejores resultados de reconstrucción. Aquí, cada imagen se compara con todas las demás.

- **Sequential:** Este modo es útil si las imágenes se adquieren en orden secuencial, por ejemplo, mediante una cámara de vídeo. En este caso, los fotogramas consecutivos se solapan visualmente y no es necesario cotejar exhaustivamente todos los pares de imágenes. En su lugar, las imágenes capturadas consecutivamente se comparan entre sí.
- **Spatial:** Este modo de correspondencia compara cada imagen con sus vecinas espaciales más cercanas. Las ubicaciones espaciales pueden establecerse manualmente en la gestión de la base de datos. Por defecto, COLMAP también extrae la información GPS de EXIF y la utiliza para la búsqueda de vecinos espaciales más cercanos. Si se dispone de información precisa de localización previa, este es el modo de comparación recomendado.
- **Transitive:** Este modo de correspondencia utiliza las relaciones transitivas de las correspondencias de características ya existentes para producir un gráfico de correspondencia más completo. Si una imagen A coincide con una imagen B y B coincide con C, este emparejador intenta emparejar A con C directamente.
- **Vocab Tree:** En este modo, cada imagen se compara con sus vecinas visuales más próximas mediante un árbol de vocabulario con reordenación espacial. Este es el modo de comparación recomendado para grandes colecciones de imágenes, varios miles.

4. El AABB scale, se trata de la configuración interna, la más importante de Instant NGP. Esta especifica la extensión de la escena, con un valor predeterminado de 16; es decir, la escena se escala de forma que las posiciones de la cámara se encuentren a una distancia media de 16 unidades del origen. Cuanto menor sea el valor, mayor la velocidad de entrenamiento. Ajustando AABB scale a una potencia mayor de 2 (hasta un máximo de 128), el modelo NeRF extenderá los rayos a un cuadro delimitador mucho mayor.

5. Una vez configurado todo, ya se puede inicial el proceso de reconocimiento del entorno.

6. Una vez se ha realizado la extracción de todos los datos y se ha creado el fichero transforms.json, pueden visualizarse los resultados con la finalidad de analizar la calidad de estos, además de guardarlos.

7. Además de extraer los keypoints del resultado, también pueden guardarse los matches encontrados entre las diferentes imágenes, por temas de optimización y espacio, se ha limitado el número de matches que se seleccionaran, para ello de las parejas de coincidencias encontradas se eligen las 100 primeras (con mayores coincidencias) y de estas, se enseñan 10 muestras aleatorias, de otra forma sería imposible apreciar los resultados correctamente en las imágenes.

8. Al guardar y cargar los keypoints o las coincidencias, estas serán mostradas por pantalla en el área remarcada de la imagen anterior.

B.4. Fase 4: Modelaje 3D

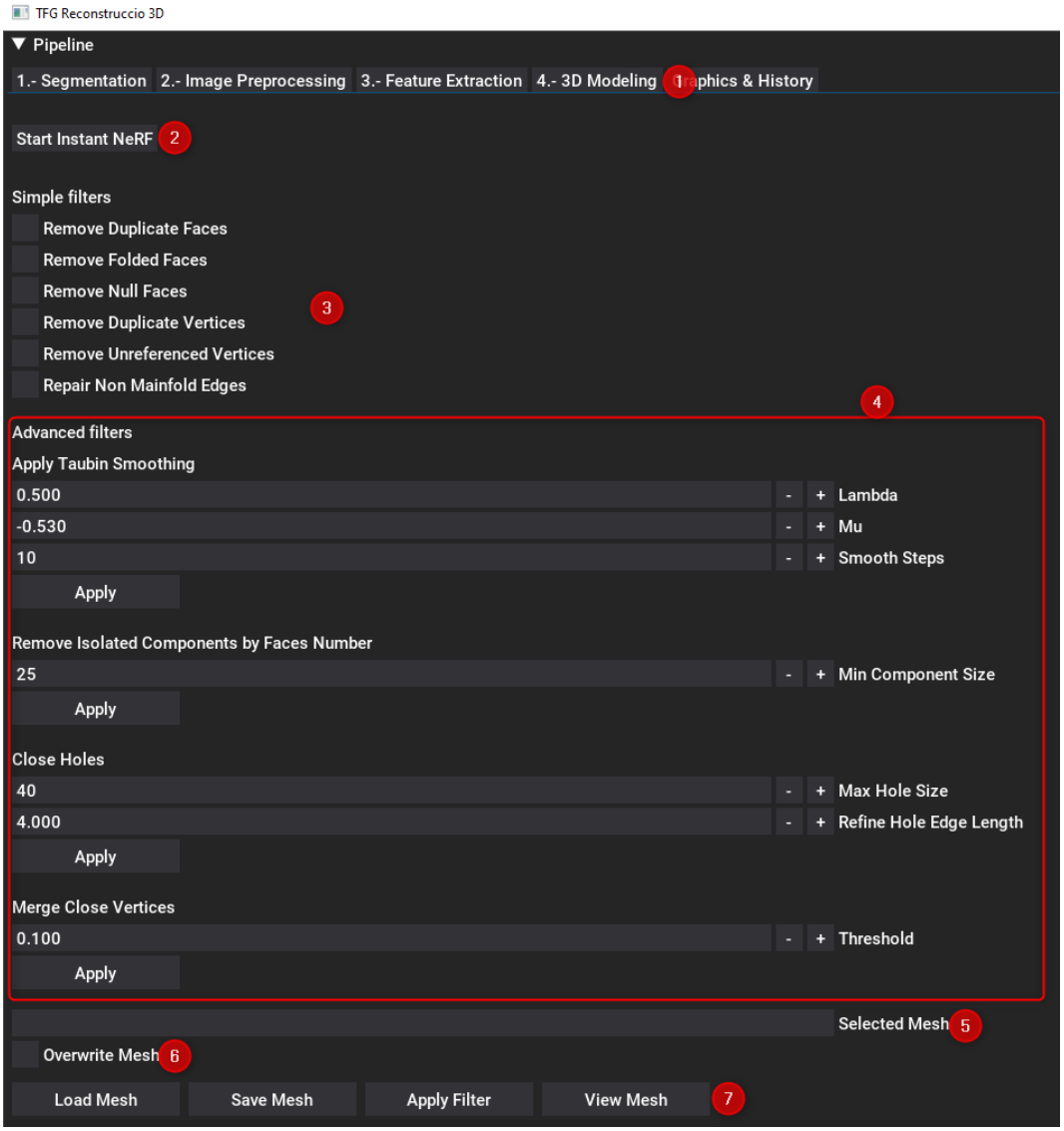


Figura B.4: GUI del Proceso de Reconstrucción 3D

Flujo de trabajo de la Figura B.4:

1. Extraídos y condensados todos los datos de la escena necesarios como input para la red neuronal de Instant NGP puede iniciarse el proceso de renderización.

2. Al darle al botón, automáticamente se iniciará **`Instant NGP`**, con los datos extraídos de la escena. Hay que tener en cuenta, que el directorio de trabajo esté configurado adecuadamente (este debe ser el que contiene el documento **transforms.json**).

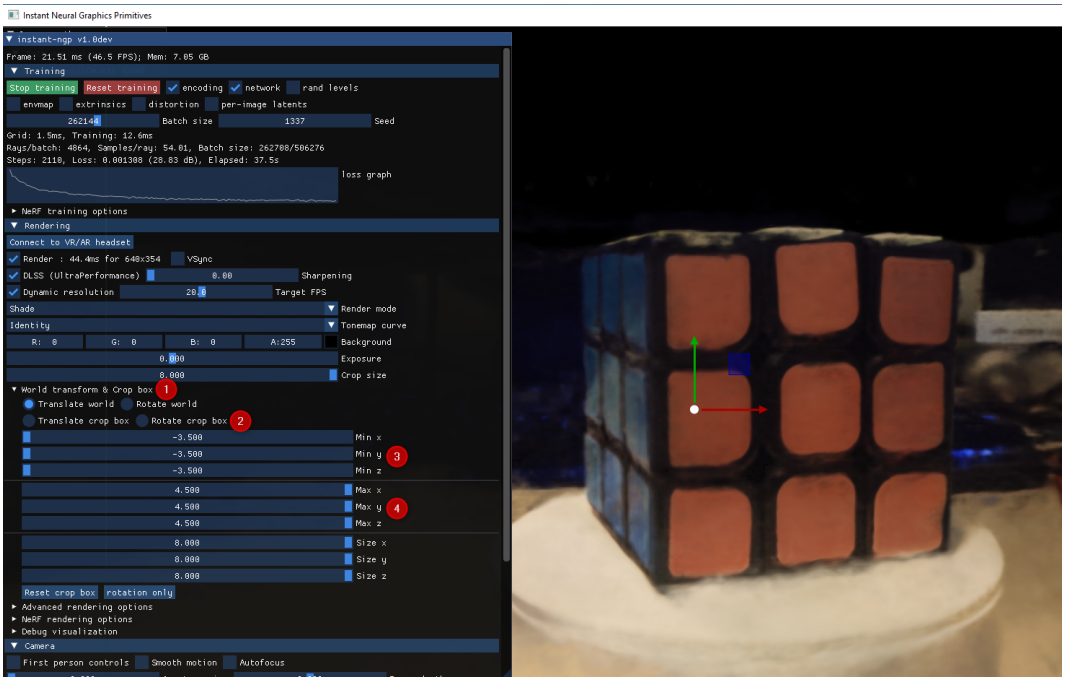


Figura B.5: Interfaz de Instant NGP

Dentro de la interfaz, más allá de las diferentes configuraciones que se pueden realizar (estás ya están optimizadas para el rendimiento más eficiente), los parámetros que se tiene que modificar son los siguientes:

- En el apartado de **World transform & Cropbox** deben modificarse los diferentes parámetros (3, 4) del tamaño de Mundo, para ello con la opción de Translate world seleccionado debe reducirse la escena a la unidad mínima que contenga el objeto a imprimir.

- Una vez reducido, si es necesario, puede rotarse la caja (2) que contiene el mundo, para intentar ajustarla lo máximo posible, cuanto más ajustada, menor ruido habrá en el objeto final.

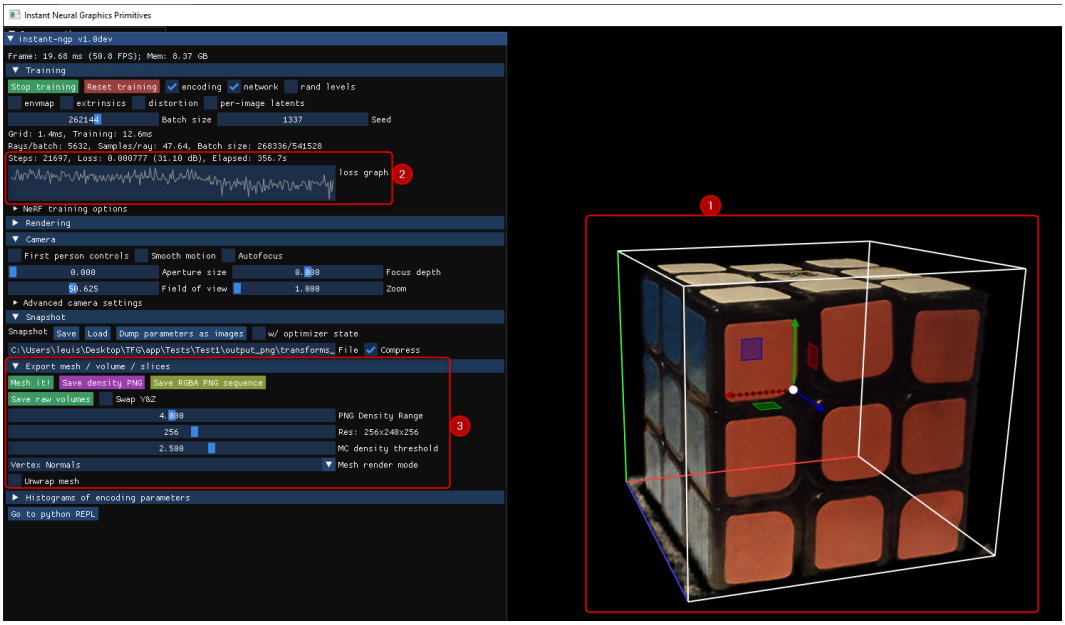


Figura B.6: Exportar mesh Instant NGP

a) Reducido el tamaño del mundo al mínimo posible.

b) Es importante verificar el error producido durante la renderización, ya que este indica la calidad del resultado. Lo recomendable es esperar hasta que la gráfica se pare, esto significa que se ha reducido el error al máximo. Dependiendo del número de imágenes proporcionadas como input, esta tarea puede demorar hasta 30 minutos, aunque la calidad del resultado final no variará significativamente. En condiciones óptimas, el resultado puede alcanzarse en tan solo 3 minutos con una calidad muy parecida a la final.

c) Alcanzada la calidad de renderizado deseada, pueden exportarse los datos a formato mallado (mesh), esto se realiza mediante el Botón: **Mesh it!**, en el apartado de **Export mesh / volume / slice**, el objeto .obj será guardado en la ruta que indica el input de la parte de abajo del apartado actual.

3. Exportados los datos a formato **.obj**, se procede a la eliminación de ruido de estos mediante el uso de la librería de **MeshLab**. Se deben seleccionar aquellos filtros que interesen más dependiendo del resultado final del objeto escaneado y la complejidad de este.

4. En el apartado de Filtros Avanzados, el funcionamiento es algo distinto. Se configura y se ejecuta de forma individual cada función, ya que esta es más compleja.

5. En este input aparece el nombre de la malla seleccionada para el refinamiento.

6. Si se desea sobrescribir los datos del objeto exportado, podemos habilitar la opción **Overwrite Mesh** (5)

7. Para poder visualizar la malla exportada, simplemente debe seleccionarse mediante el uso del botón **Load Mesh** y posteriormente visualizarlo con el botón **View Mesh** (6)

B.5. Fase 4: Gráficos e Historial

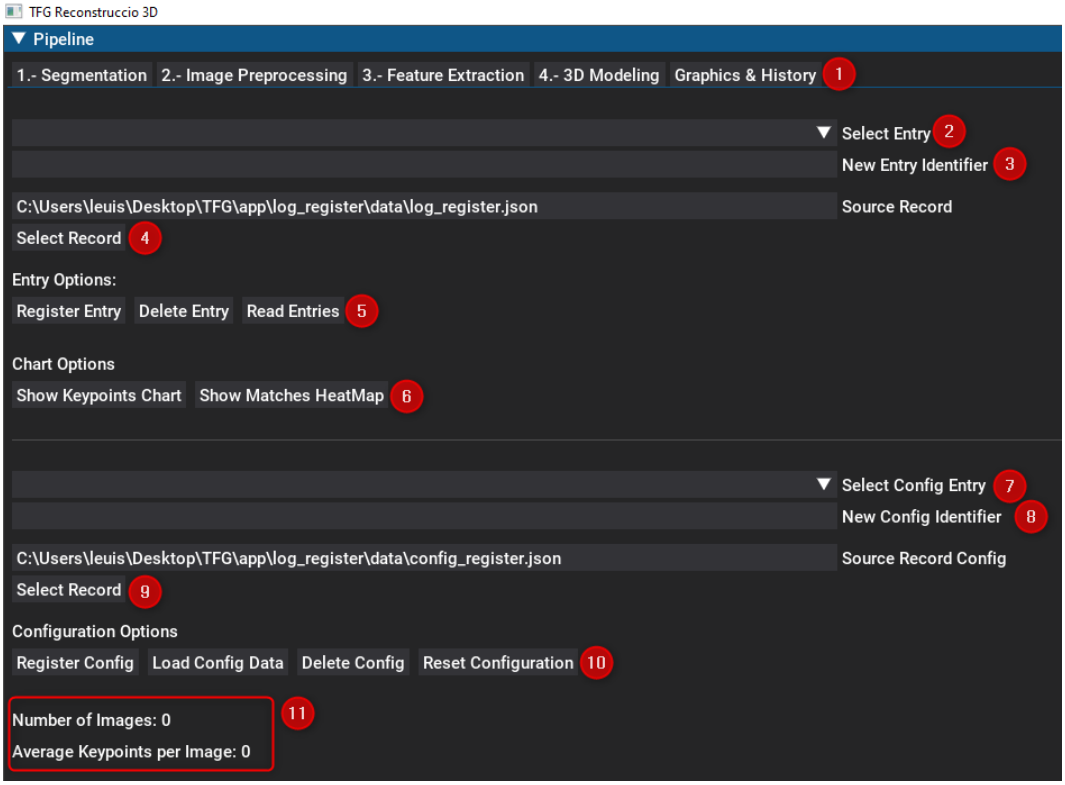


Figura B.7: GUI del apartado de Gráficos e Historial

Flujo de trabajo de la Figura B.7:

1. Finalmente, la última pestaña del programa, en esta se recogen las ejecuciones y las configuraciones guardadas.

2. Si existe una entrada en el registro de ejecuciones, aparecerá en la lista de identificaciones. También se utiliza para especificar la entrada de la que se extraerá los datos de ejecución de los botones del apartado (6).

3. En el caso de querer registrar una nueva entrada debe especificarse el identificador.

4. Si se desea tener diferentes registros de ejecuciones, puede elegirse otro origen.

5. Opciones disponibles para la entrada. Crear una nueva. Eliminar la seleccionada. Leer de nuevo el registro de ejecuciones.
6. Se lee el identificador seleccionado en el desplegable y se genera la gráfica de puntos clave y el mapa de calor de las coincidencias.
7. Si existe una entrada en el registro de configuraciones, aparecerá en la lista de identificaciones. También se utiliza para especificar la entrada de la que se aplicara la configuración (10).
8. En el caso de querer registrar una nueva configuración debe especificarse el identificador.
9. Se puede modificar la fuente de registros de donde extraer las configuraciones.
10. Opciones disponibles para el registro de configuraciones. Registrar una nueva configuración. Cargar en el pipeline la configuración seleccionada. Eliminar una configuración. Resetear la configuración a por defecto.
11. Al cargar una entrada del registro de ejecuciones, se configuran los datos básicos de esta.

Bibliografía

[1] Autodesk, Inc., Autodesk fusion 360, 2024.

[2] Ultimaker B.V., Ultimaker cura, Blender Foundation, Watermolenweg 2, Netherlands, 2023.

[3] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia, Meshlab: an open-source mesh processing tool, European Interdisciplinary Cybersecurity Conference, 2008.

[4] Blender Online Community, Blender - a 3d modelling and rendering package, Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018.

[5] Xian-Feng Han, Hamid Laga, and Mohammed Bennamoun, Image-based 3d object reconstruction: State-of-the-art and trends in the deep learning era, IEEE transactions on pattern analysis and machine intelligence **43** (2019), no. 5, 1578–1604.

[6] Jonathan Hoffstadt and Preston Cothren, Dearpygui, 2024.

[7] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis, 3d gaussian splatting for real-time radiance field rendering, ACM Transactions on Graphics **42** (2023), no. 4, 1–14.

[8] Prusa Research, Prusaslicer, 2024.

[9] Johannes L. Schönberger, Colmap: A general-purpose structure-from-motion and multi-view stereo software, 2024.

[10] Gabriel Taubin, A signal processing approach to fair surface design, Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, 1995, pp. 351–358.

[11] Visual Computing Lab - ISTI - CNR, Meshlab: The open source system for processing and editing 3d triangular meshes, `http://www.meshlab.net/`, 2023, Accessed: 2023-06-08.