UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO
FACULTAD DE INGENIERÍA ELÉCTRICA ELECTRÓNICA INFORMÁTICA Y MECÁNICA
ESCUELA PROFESIONAL DE INGENIERÍA INFORMÁTICA Y DE SISTEMAS

TESIS

DESARROLLO DE UN PROTOTIPO PARA ESCANEAR SUPERFICIES DIFÍCILES DE VISUALIZAR Y SU RECONSTRUCCIÓN MEDIANTE REDES NEURONALES

PRESENTADO POR:

BR. MADAI HANAMPA QUISPE
BR. JULIO CESAR MAMANI ARIAS

PARA OPTAR AL TITULO PROFESIONAL DE INGENIERO INFORMÁTICO Y DE SISTEMAS

ASESOR:
DR. LAURO ENCISO RODAS

CUSCO - PERÚ

2025

INFORME DE ORIGINALIDAD

(Aprobado por Resolución Nro.CU-303-2020-UNSAAC)

El que suscribe, **Asesor** del trabajo de investigación/tesis titulada: .....

"Desarrollo de un prototipo para escanear superficies difíciles de visualizar y su reconstrucción mediante redes neuronales"

Presentado por: Madai Hanampa Quispe DNI N° 47990432

presentado por: Julio Cesar Mamani Arias DNI N°: 47766770

Para optar el título profesional/grado académico de Ingeniero Informático y de Sistemas

Informo que el trabajo de investigación ha sido sometido a revisión por 1 veces, mediante el Software Antiplagio, conforme al Art. 6° del ***Reglamento para Uso de Sistema Antiplagio de la UNSAAC*** y de la evaluación de originalidad se tiene un porcentaje de 2 **%.**

Evaluación y acciones del reporte de coincidencia para trabajos de investigación conducentes a grado académico o título profesional, tesis

Porcentaje	Evaluación y Acciones	Marque con una (X)
Del 1 al 10%	No se considera plagio.	X
Del 11 al 30 %	Devolver al usuario para las correcciones.	
Mayor a 31%	El responsable de la revisión del documento emite un informe al inmediato jerárquico, quien a su vez eleva el informe a la autoridad académica para que tome las acciones correspondientes. Sin perjuicio de las sanciones administrativas que correspondan de acuerdo a Ley.	

Por tanto, en mi condición de asesor, firmo el presente informe en señal de conformidad y **adjunto** las primeras páginas del reporte del Sistema Antiplagio.

Cusco, 06 de Mayo de 2025

Firma

Post firma Lauro Enciso Rodas

Nro. de DNI 23853228

ORCID del Asesor 0000-0001-6266-9838

Se adjunta:

1. Reporte generado por el Sistema Antiplagio.
2. Enlace del Reporte Generado por el Sistema Antiplagio: **oid:** 27259:454159661

Madai Hanampa

Turnitin FinalTESIS FINALMadai 2025.pdf

Universidad Nacional San Antonio Abad del Cusco

Detalles del documento

Identificador de la entrega
trn:oid:::27259:454159661

Fecha de entrega
29 abr 2025, 10:26 p.m. GMT-5

Fecha de descarga
29 abr 2025, 10:49 p.m. GMT-5

Nombre de archivo
Turnitin FinalTESIS FINALMadai 2025.pdf

Tamaño de archivo
37.2 MB

128 Páginas

25.174 Palabras

138.276 Caracteres

2% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text
- Small Matches (less than 15 words)

Top Sources

1% Internet sources
0% Publications
1% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Dedicatorias

Este trabajo dedico a mi hija, mis abuelitos, mi mamá y mi esposo, por la gran enseñanza del día a día, la motivación e inspiración que me brindan.

Madai Hanampa Quispe

Dedico este trabajo a mi mamá, a mis hermanos y pareja, por el apoyo que siempre me dan y motivación a seguir adelante.

Julio cesar Mamani Arias

Agradecimiento

Agradecemos a Dios por la gran oportunidad de permitirnos terminar este proyecto pese a muchas dificultades. A nuestro asesor, el Dr. Lauro Enciso Rodas, por su orientación en la elaboración de este trabajo. Deseamos manifestar nuestro sincero agradecimiento a todas las personas que, de diversas maneras, contribuyeron al progreso de esta tesis. Agradecemos de manera especial a nuestros docentes, colegas y amigos por su apoyo inestimable, sus recomendaciones y, sobre todo, por haber sido una fuente continua de inspiración que nos motivó a culminar este proyecto con éxito. Gracias.

Abreviaturas

- **CNN** Convolutional neural network o ConvNet
- **GD** Gradient descent
- **IA** Inteligencia artificial
- **SVM** Support vector machine, maquina de vector soporte
- **RNN** Recurrent neural network
- **ReLU** Rectified Linear Unit
- **ML** Machine learning
- **LR** Logistic regression, regresion logistica
- **DMI2** Drone DJI Mini 2
- **RTH** Regreso automático o return to home.
- **WiFiUDP** Comunicación UDP en la red Wi-Fi del ESP32
- **LIDAR** Light Detection and Ranging
- **PCD** Point Cloud Data
- **PLY** formato PLY de archivo de nube de puntos
- **ROSbag** Archivo para almacenar datos en el sistema ROS
- **ECC** Estación de Control de Computadora
- **SOC** Sockets
- **THRD** Threading.º subprocesos
- **NP** Numpy, en consistencia con la convención en Python.
- **UDP** Protocolo de transmisión de datos.

Resumen

Cusco es una región con una amplia riqueza cultural, arqueológica y geográfica, conocida por albergar una de las civilizaciones más importantes de la historia: los Incas. Sin embargo, existen numerosas áreas aún desconocidas o poco estudiadas debido a su ubicación o accesibilidad, la dificultad de acceso o el temor que estas zonas generan entre los pobladores locales. Estas áreas, que incluyen cuevas, túneles, cavernas, y sitios arqueológicos, representan un desafío para los investigadores.

En este trabajo de investigación se presenta el desarrollo de un prototipo para la reconstrucción 3D de áreas de difícil acceso y escasa visibilidad, utilizando algoritmos de la librería Open3D y un enfoque basado en redes neuronales. Estas tecnologías permiten obtener modelos tridimensionales precisos a partir de nubes de puntos capturadas mediante un sensor LiDAR el cual con apoyo de un drone, explora el espacio tridimensional a través de coordenadas X, Y y Z. La problemática de explorar lugares inaccesibles, tanto a nivel físico como tecnológico, sigue siendo un desafío en la investigación, especialmente en regiones como Cusco, donde existen múltiples áreas que permanecen sin explorar.

El objetivo de este proyecto es proporcionar una herramienta que facilite la investigación en este área, permitiendo generar modelos 3D detallados que puedan servir como base para estudios arqueológicos, topográficos y de infraestructura.

El proceso de reconstrucción comienza con la captura de la nube de puntos, seguida de su procesamiento y análisis para estimar las normales de la superficie y generar una malla 3D detallada. Posteriormente, usamos redes neuronales basadas en autoencoders para mejorar la precisión y eficiencia de la reconstrucción de estas superficies. Así las aplicaciones de este trabajo incluyan la mejora en la exploración de lugares difíciles de alcanzar y la generación de modelos tridimensionales para una mejor comprensión y análisis de espacios ocultos.

Palabras Clave: *Prototipo, Reconstrucción 3D, Áreas de difícil acceso, Visibilidad escasa, Open3D, Redes neuronales, Modelos tridimensionales, Nubes de puntos, LiDAR, Dron, Estudios arqueológicos, Topográfia, Infraestructura, Normales, Malla 3D, Autoencoders, Exploración, Prototipo*

Abstract

Cusco is a region with vast cultural, archaeological, and geographical wealth, known for hosting one of the most important civilizations in history: the Incas. However, there are numerous areas that remain unexplored or poorly studied due to their location, accessibility challenges, or the fear these places generate among local inhabitants. These areas, which include caves, tunnels, caverns, and archaeological sites, pose a significant challenge for researchers.

In this research, we present the development of a prototype for 3D reconstruction of hard-to-reach areas with limited visibility, using algorithms from the Open3D library and a neural network-based approach. These technologies enable the generation of accurate 3D models from point clouds captured by a LiDAR sensor, which, supported by a drone, explores the three-dimensional space through X, Y, and Z coordinates. The challenge of exploring inaccessible locations, both physically and technologically, remains a significant issue in research, particularly in regions such as Cusco, where many areas remain unexplored.

The objective of this project is to provide a tool that facilitates research in this field, enabling the generation of detailed 3D models that can serve as a foundation for archaeological, topographic, and infrastructure studies.

The reconstruction process begins with point cloud capture, followed by its processing and analysis to estimate surface normals and generate a detailed 3D mesh. Subsequently, we use neural networks based on autoencoders to improve the accuracy and efficiency of surface reconstruction. The applications of this work include enhancing the exploration of hard-to-reach places and generating three-dimensional models for a better understanding and analysis of hidden spaces.

Keywords: Prototype, 3D Reconstruction, Hard-to-Reach Areas, Limited Visibility, Open3D, Neural Networks, Three-Dimensional Models, Point Clouds, LiDAR, Drone, Archaeological Studies, Topography, Infrastructure, Normals, 3D Mesh, Autoencoders, Exploration, Prototype.

Índice general

Índice de cuadros

Índice de figuras

Introducción

En el contexto actual, la exploración y documentación tridimensional de espacios desconocidos o de difícil acceso ha cobrado gran relevancia debido a su potencial impacto en diversas áreas de investigación y aplicaciones prácticas. Desde la arqueología y la preservación del patrimonio cultural, hasta la ingeniería civil, la minería y la gestión de desastres, contar con herramientas precisas y eficientes para mapear y reconstruir estos espacios permite acceder a información clave que antes era difícil o imposible de obtener. Sin embargo, los métodos tradicionales de documentación presentan limitaciones significativas, ya sea por los altos costos asociados, las barreras tecnológicas, o los riesgos inherentes que conllevan las inspecciones físicas en estas áreas.

En este contexto, el presente trabajo propone el diseño y desarrollo de un prototipo capaz de capturar y reconstruir tridimensionalmente estas superficies mediante el uso de un sensor LiDAR equipado en un dron. Este enfoque no solo permite acceder a lugares de difícil visualización, sino que también garantiza una mayor seguridad y precisión en la recolección de datos. El prototipo combina tecnologías avanzadas, como redes neuronales con autoencoders para la reconstrucción de superficies, proporcionando una solución.

El propósito de este desarrollo trasciende una aplicación específica, ya que sus potenciales usos abarcan áreas tan diversas como:

Exploración científica: Análisis detallado de cuevas, túneles y entornos naturales inaccesibles. Minería y geología: Inspección de túneles y galerías subterráneas, identificando riesgos estructurales. Arqueología y patrimonio cultural: Digitalización precisa de sitios arqueológicos complejos. Gestión de desastres: Evaluación de zonas afectadas por derrumbes o catástrofes naturales.

Además, este prototipo busca contribuir al avance del estado del arte en reconstrucción tridimensional, integrando métodos de procesamiento de nubes de puntos con redes neuronales basadas en autoencoders, que permiten mejorar la reconstrucción El presente trabajo de investigación propone la siguiente estructura:

- **Capítulo 1: Aspectos generales** Este capítulo presenta el marco introductorio del proyecto, incluyendo el contexto y la motivación para desarrollar un prototipo de reconstrucción tridimensional. Se explicará la relevancia del tema para explorar áreas desconocidas o de difícil acceso y su aplicabilidad en diversas áreas de

investigación. También se definirán los objetivos principales, tanto general como específicos, y se establecerán los alcances y limitaciones del proyecto.

- **Capítulo 2: Marco Teórico** Este apartado aborda los fundamentos conceptuales y teóricos necesarios para el desarrollo del prototipo. Se describirán las tecnologías y métodos actuales relacionados con la reconstrucción tridimensional y la captura de datos mediante nubes de puntos. Se analizarán conceptos clave como LiDAR, redes neuronales y algoritmos de reconstrucción, proporcionando la base científica para sustentar el diseño del prototipo.

- **Capítulo 3: Desarrollo del proyecto** Este capítulo detalla el diseño y construcción del prototipo. Se describen las herramientas y tecnologías utilizadas, como el sensor LiDAR, los algoritmos de procesamiento de nubes de puntos y los frameworks de reconstrucción 3D. Además, se incluyen las etapas de implementación, desde la captura de datos hasta la integración de la interfaz gráfica y las funcionalidades para visualizar los resultados.

- **Capítulo 4: Proceso experimental y resultados** En esta sección, se documentan las pruebas realizadas con el prototipo en diferentes entornos y condiciones. Se presentan los datos obtenidos, los análisis realizados y la validación de los resultados frente a objetivos planteados. Además, se discuten los desafíos encontrados durante el proceso experimental y cómo fueron abordados.

- **Capítulo 5: Conclusiones y Recomendaciones** El capítulo final resume las principales conclusiones del proyecto, destacando los logros alcanzados y el impacto potencial del prototipo en áreas de investigación e industrias específicas. También se presentan recomendaciones para optimizar el sistema y sugerencias para futuros trabajos, incluyendo mejoras técnicas, nuevas aplicaciones y líneas de investigación complementarias.

Parte I

Aspectos Generales

Capítulo 1

Aspectos Generales

1.1. Antecedentes

Actualmente hay herramientas que realizan la reconstrucción tridimensional con tecnologías altamente costosas, pero también existen investigaciones que realizaron en esta área como son los casos de:

- ***"Implementación de un sistema de mapeo y localización a un robot hexápodo enfocado en la exploración del entorno y monitoreo de temperatura"*** (Alvarado, 2019)

  El autor presenta un trabajo en el cual propone realizar un móvil de localización y mapeo de estructuras utilizando métodos probabilísticos, el uso de filtro de partículas para que el móvil pueda detectar objetos a su alrededor, toda la información será capturada y almacenada en un archivo que posteriormente será visualizado en un computador, el autor enfoca que el funcionamiento se verá enfocado en la robustez que obtenga de los resultados.

- ***"Reconstrucción de mapas utilizando Escaneo Láser para la navegación de un robot móvil"*** (Lopéz, 2016)

  Este proyecto presenta un trabajo de 2 partes, la primera el diseño y desarrollo de un móvil, y la segunda parte es la de integrar los diferentes componentes y sensores electrónicos para la captura de datos y comunicación del móvil hacia el computador. El autor también presenta un avance con el desarrollo de una interfaz con lo cual podrá realizar el monitoreo en tiempo real del móvil. Para la navegación emplea diversos algoritmos para la reconstrucción de mapas por el método de segmentación, el autor uso un sensor LIDAR el cual cumple con la función captar las distancias entre el sensor y la superficie, no presenta fallas en superficies no reflectantes y con ángulos de inclinación.

- ***"Técnicas de Medición Tridimensional Aplicables a Robótica Móvil Usando Proyección de Luz Estructurada"*** (Isáis, 2015)

  Realiza un proyecto que consta de 2 partes, la primera la de realizar un móvil con ruedas que será controlado para capturar los datos que mediante el uso de sensores láser podrá captar objetos a su alrededor, en la segunda parte el autor desarrolla una interfaz gráfica en la cual podrá visualizar en tiempo real la orientación y posición del carrito con ruedas el cual realizará y mostrará un mapa del lugar en el que se encuentre el móvil realizando así un escaneo del ambiente. Este trabajo usó para las pruebas un sensor de distancia RPLIDAR el cual no presenta dificultades como las de un sensor ultrasónico.

1.2. Planteamiento del problema

1.2.1. Descripción del problema

El avance tecnológico y la Inteligencia Artificial han revolucionado múltiples áreas del conocimiento, convirtiéndose en herramientas indispensables tanto para la vida cotidiana como para investigaciones que, en el pasado, enfrentaban limitaciones significativas. Estos avances han permitido logros como la reconstrucción de rostros históricos, la recreación de viviendas destruidas tras desastres naturales, la modelación exacta de ciudades, e incluso la creación de nuevos espacios tridimensionales a partir de imágenes bidimensionales. Sin embargo, en regiones como Cusco, estas tecnologías aún no han sido plenamente aprovechadas para resolver problemáticas locales específicas.

En el Cusco, una región rica en historia y cultura, existen numerosos lugares, túneles y estructuras subterráneas que, hasta el día de hoy, permanecen sin explorar debido a barreras de accesibilidad. Muchas de estas zonas presentan entradas extremadamente pequeñas, lo que impide el acceso humano directo. Además, la antigüedad y fragilidad de estas estructuras aumentan el riesgo de daños si se utilizan métodos invasivos de exploración. Por otro lado, los equipos necesarios para realizar investigaciones no invasivas, como los escáneres LiDAR y drones especializados, pueden facilitar proyectos locales o investigaciones arqueológicas y científicas en la región.

Esta problemática no se limita a Cusco. En muchas otras partes del mundo existen ambientes desconocidos o de difícil acceso, como túneles subterráneos, cavernas, zonas arqueológicas ocultas o sitios ubicados en terrenos peligrosos, que no han sido adecuadamente explorados debido a limitaciones tecnológicas o económicas. Estas barreras restringen la capacidad de los investigadores para descubrir, analizar y preservar valiosa información histórica, geológica o cultural.

En este contexto, se plantea como objetivo desarrollar un prototipo que permita superar estas limitaciones. Este prototipo está diseñado para apoyar la exploración y re-

construcción tridimensional de ambientes inaccesibles, a travéz de un enfoque asequible y eficiente. Consiste en la implementación de un dispositivo móvil equipado con un sensor LiDAR y controlado con Arduino, que, mediante la recopilación de datos de distancias y ángulos, genera una nube de puntos con coordenadas tridimensionales (x, y, z). Esta etapa inicial de escaneo es crucial para obtener una representación básica del espacio explorado.

Posteriormente, estos datos serán procesados utilizando librerías de Python, como la biblioteca Open3D y uso de redes neuronales para mejorar la calidad de la reconstrucción tridimensional. Estas tecnologías permiten optimizar la interpretación y visualización de los datos, generando modelos más precisos y detallados de los espacios explorados.

El desarrollo de este prototipo no solo tiene como finalidad la exploración de lugares inaccesibles en Cusco, sino que también abre la puerta a aplicaciones en diversas áreas, como:

- **Arqueología y conservación del patrimonio cultural:** Exploración de sitios arqueológicos ocultos o frágiles sin riesgo de daños estructurales.
- **Geología y minería:** Mapeo seguro de túneles y galerías subterráneas para identificar riesgos o evaluar condiciones estructurales.
- **Gestión de desastres naturales:** Reconstrucción de zonas afectadas por deslizamientos de tierra, terremotos o inundaciones.
- **Exploración científica:** Estudio de cavernas, sistemas de túneles y otras formaciones naturales de difícil acceso.

Con esta propuesta, se busca abordar las limitaciones tecnológicas y económicas que enfrentan las investigaciones en Cusco y otras regiones con características similares. Además, este proyecto pretende democratizar el acceso a herramientas avanzadas de reconstrucción tridimensional, haciéndolas más asequibles y accesibles para proyectos locales y regionales.

1.3. Formulación del problema

1.3.1. Problema general

¿Es factible desarrollar un prototipo de escaneo 3D, basado en redes neuronales, que permita reconstruir superfícies de difícil visualización a un costo significativamente menor al de las tecnologías actuales?

1.3.2. Problemas específico

- *Adquisición de hardware*, dado que sensores de alta precisión y estaciones totales profesionales pueden superar los $20 000 USD.
- *Licenciamiento de software*, pues las licencias de fotogrametría y procesamiento de nubes tienen cuotas anuales elevadas.
- *Gastos operativos y de mantenimiento*, ya que el calibrado periódico y los consumibles encarecen el uso en campo.

1.4. Objetivos

1.4.1. Objetivo General

Desarrollar un prototipo para escanear superficies de difícil visualización y su reconstrucción mediante redes neuronales

1.4.2. Objetivos Específicos

- Identificar instrumentos para capturar nubes de puntos e identificar el más adecuado para superficies de díficil visualización.
- Diseñar un prototipo que permita capturar nube de puntos en superficies de difícil visualización.
- Reconstrucción de las superficies mediante algoritmos de reconstrucción y redes neuronales.

1.5. Alcances y Limitaciones

1.5.1. Alcances

- Identificación de instrumentos adecuados para la captura de nubes de puntos: El proyecto incluirá la selección y análisis de herramientas tecnológicas, como sensores LiDAR y sistemas basados en Arduino, que sean eficaces para capturar datos en áreas de difícil acceso.
- Desarrollo de un prototipo funcional: Se diseñará e implementará un prototipo que integre sensores y controladores para la captura de nubes de puntos. El prototipo será probado en escenarios controlados y simulaciones representativas de lugares inaccesibles.

- Procesamiento de nubes de puntos y reconstrucción 3D: Se implementarán algoritmos de procesamiento y redes neuronales para reconstruir superficies tridimensionales a partir de datos capturados, utilizando herramientas como Python y Open3D.
- Enfoque en entornos específicos: El prototipo estará diseñado para explorar túneles pequeños, espacios arqueológicos y otras áreas de difícil visualización en la región de Cusco. El trabajo se limitará a reconstrucciones basadas en las condiciones del entorno local.
- Contribución académica y tecnológica: El proyecto ofrecerá una solución inicial y escalable para la reconstrucción de espacios inaccesibles, con potencial aplicación en otras áreas como minería, arqueología y conservación patrimonial.

1.5.2. Limitaciones

- Los altos costos de los sensores son una limitación por lo que debemos usar otros más económicos los cuales no son tan precisos.
- Poco conocimiento de electrónica, para la integración de circuitos con el móvil.
- Alcance del hardware: El prototipo estará limitado por las capacidades técnicas y económicas de los componentes seleccionados, como el sensor LiDAR y el controlador Arduino. Instrumentos de mayor precisión o alcance podrían no ser accesibles dentro del presupuesto.
- Resolución de la nube de puntos: La calidad de las nubes de puntos generadas dependerá de las especificaciones del sensor utilizado y de los movimientos bruscos que pueda tener el drone por un mal manejo, lo que podría limitar la precisión en la reconstrucción de superficies.
- Escenarios de prueba restringidos: Las pruebas del prototipo se realizarán en ambientes cerrados, se realizarón pruebas en ambientes abiertos, pero el sensor LIDAR tiene una distancia de 14 m de radio. Otro limitante en los escenarios, para el dron, en el Cusco se tiene un poligono de seguridad alrededor del aeropuerto Velazco Astete, el cual indica que no se puede volar al drone en este margen.
- Limitaciones computacionales: Los algoritmos de reconstrucción y redes neuronales estarán restringidos por los recursos computacionales disponibles, lo que podría afectar la velocidad o la resolución del procesamiento de los datos.

1.6. Justificación

El desarrollo de este prototipo tiene como propósito principal convertirse en una herramienta innovadora para la exploración de superficies de difícil visualización, tales como cuevas, cavernas, túneles, edificios colapsados o áreas de difícil acceso para el ser humano. Estas áreas, debido a su inaccesibilidad o peligros inherentes, han sido tradicionalmente difíciles de documentar y analizar, lo que limita el avance en campos como la arqueología, la geología, la minería, y la gestión de desastres.

El uso de robots y drones representa una solución eficiente y segura, ya que permite acceder y operar en entornos hostiles o inhabitables. Esto es especialmente relevante para la región del Cusco, que cuenta con una vasta riqueza arqueológica y geológica, incluyendo túneles y espacios subterráneos que permanecen inexplorados debido a su difícil acceso o a las limitaciones tecnológicas y económicas actuales. La implementación de este prototipo contribuirá a superar estos desafíos, facilitando el mapeo, la documentación y el análisis de estos lugares, lo cual es crucial para la investigación científica y la conservación patrimonial.

Además, el proyecto tiene aplicaciones potenciales en escenarios de emergencia, como la evaluación de estructuras después de fenómenos naturales, permitiendo obtener datos clave para la toma de decisiones en situaciones críticas. Esto no solo tiene un impacto local, sino también global, ya que la tecnología desarrollada puede adaptarse a otros contextos y regiones con características similares.

En este sentido, el presente proyecto busca no solo explorar y resolver problemáticas locales, sino también contribuir al avance de la ciencia y la tecnología en el ámbito de la reconstrucción tridimensional, ofreciendo una solución accesible y escalable que integre técnicas modernas de captura de datos y algoritmos avanzados de reconstrucción.

1.7. Metodología

Para este proyecto la metodología estará basada en el método descriptivo basándonos en Sampieri, R. (2014), la cual implica seguir una serie de pasos consecutivos y ordenados para llegar al objetivo planteado, los cuales son:

- Definición de requisitos y objetivos: Se establecen los requisítos y objetivos del proyecto, cuyo propósito es diseñar un vehículo móvil capaz de desplazarse en diferentes tipos de terreno. Este vehículo esta equipado con un sensor LIDAR, que se encargará del escaneo y captará una nube de puntos en el plano XY, complementando el plano Z con la ruta que sigue el móvil. De esta manera, se obtendrá una nube de puntos en un espacio tridimensional.

- Adquisición de datos: Una vez diseñado el prototipo, se procede a llevar a campo,

es decir a áreas de difícil acceso para la adquisición de la nube de puntos en 3D. Se realizan pruebas y ajustes para asegurar una obtención precisa y confiable de la nube de puntos en estas superficies.

- Procesamiento de datos: Se lleva a cabo el procesamiento de los datos capturados con el fin de limpiar y filtrar ruidos, así como eliminar puntos irrelevantes. Este proceso mejora la calidad de la nube de puntos y se desarrolla mediante algoritmos de filtrado de ruido, la eliminación de outliers (valores extremos que se desvían significativamente de la tendencia general de un conjunto de datos).

- Se utiliza redes neuronales de tipo Autoencoders para aumentar la densidad de puntos manteniendo la estructura original de la nube de puntos y de esta forma sea mas compacta y detallada.

- Posteriormente, se aplican algoritmos para el cálculo de las normales de la nube de puntos a partir de una línea referencia, basados en modelos matemáticos, paso crucial para la mejor reconstrucción.

- Reconstrucción de superficies: Se utilizó algoritmos y librerías de python como: open3d, algoritmo de Poison y malla triangular y así se logra la reconstrucción 3D final.

- Evaluación y ajuste: Se evalúa el rendimiento del prototipo mediante métricas apropiadas, como el error de reconstrucción y la precisión geométrica. Estos ajustes son en función de los resultados obtenidos y repitiendo el ciclo de entrenamiento y evaluación.

- Implementación y pruebas: Se finalizó con la implementación del prototipo con apoyo de un computador personal y se realizó las pruebas exhaustivas en diferentes escenarios y condiciones para verificar la eficacia del prototipo.

1.8. Cronograma de Actividades

Cuadro 1.1: Cronograma de actividades

	2023					2024							
	JUL.	AGO.	SEP.	OCT.	NOV.	DIC.	ENE.	FEB.	MAR.	ABR.	MAY.	JUN.	JUL.
1. REVISIÓN BIBLIOGRÁFICA													
2. INVESTIGACIÓN SOBRE REDES CONVOLUCIONALES													
3. DESARROLLO DE PLAN DE TESIS													
4. INVESTIGACIÓN DE METODOS DE RECONSTRUCCION 3D													
5. DISEÑO DE PROTOTIPO													
6. ADQUISICIÓN DE DATOS													
7. PREPROCESAMIENTO DE DATOS													
8. SEGMENTACIÓN DE LA NUBE DE PUNTOS													
9. RECONSTRUCCIÓN DE SUPERFICIES													
10. EVALUACIÓN DE RESULTADOS													
11. REDACCIÓN DE TESIS													

Fuente Propia

1.9. Costo y presupuesto

Los gastos en este proyecto se describirán en la siguiente tabla:

COMPONENTES	COSTO	FINANCIAMIENTO
Procesador Ryzen 5600 X	S/. 700	PROPIO
Disco duro NVME 2.0 1TB	S/. 400	PROPIO
Placa madre Asus TUF GAMING B450M	S/. 600	PROPIO
Tarjeta de video RTX 3070 Ti msi	S/. 4200	PROPIO
Fuente de poder 750W 80 PLUS	S/. 450	PROPIO
Sensor LIDAR	S/. 400	PROPIO
DRON DJI Avata 2	S/. 2300	PROPIO

Cuadro 1.2: Costos y presupuesto

Parte II

Marco Teorico

Capítulo 2

Marco conceptual

2.1. Identificación de instrumentos para la captura de la nube de puntos

La captura de nubes de puntos es un proceso que permite registrar información tridimensional de un entorno o superficie en forma de un conjunto de puntos en el espacio, definidos por coordenadas (x, y, z). Estas nubes de puntos se utilizan como base para generar modelos tridimensionales, permitiendo analizar, medir y reconstruir superficies físicas de manera precisa. Este proceso ha cobrado relevancia en diversas áreas como la arqueología, ingeniería civil, minería y cartografía, debido a su capacidad para representar entornos complejos o de difícil acceso.

Para capturar estas nubes de puntos, se emplean diferentes tipos de instrumentos y tecnologías que permiten recolectar información espacial con precisión. Entre los más utilizados se encuentran los sensores LiDAR, las cámaras estéreo y RGB-D, y los métodos basados en fotogrametría. La elección del instrumento adecuado depende de factores como el entorno, la precisión requerida y las limitaciones del proyecto, tales como costos o accesibilidad.

2.1.1. Instrumento de Captura: LiDAR

El sensor **LiDAR (Light Detection and Ranging)** es una herramienta avanzada que permite capturar información tridimensional de superficies u objetos mediante pulsos de luz láser. Este instrumento mide distancias basándose en el *Tiempo de Vuelo* (*Time-of-Flight*) de los pulsos láser y genera nubes de puntos tridimensionales. La ecuación principal para calcular la distancia d es:

$$d = \frac{c \cdot t}{2} \tag{2.1}$$

donde:

- c es la velocidad de la luz ($3 \times 10^8\,\mathrm{m/s}$),
- t es el tiempo que tarda el pulso en viajar hacia el objeto y regresar al sensor.

Transformación de Coordenadas

Los datos obtenidos por el LiDAR inicialmente están en coordenadas polares o esféricas (r, θ, ϕ), donde:

- r es la distancia medida,
- θ es el ángulo horizontal,
- ϕ es el ángulo vertical.

Estos datos se convierten a coordenadas cartesianas (x, y, z) mediante las siguientes ecuaciones:

$$x = r \cdot \cos(\theta) \cdot \sin(\phi) \tag{2.2}$$

$$y = r \cdot \sin(\theta) \cdot \sin(\phi) \tag{2.3}$$

$$z = r \cdot \cos(\phi) \tag{2.4}$$

Patrones de Escaneo

Los sensores LiDAR utilizan diferentes patrones de escaneo para cubrir el área de interés, tales como:

- Escaneo lineal o circular,
- Escaneo tipo espirógrafo, el cual es eficiente para maximizar la cobertura.

Estos patrones generan grandes cantidades de datos que requieren técnicas de registro y alineación para combinar múltiples nubes de puntos. Entre los algoritmos más comunes se encuentran:

- **ICP (Iterative Closest Point)**: Minimiza las diferencias entre dos nubes de puntos superpuestas.
- **NDT (Normal Distributions Transform)**: Transforma las nubes de puntos en distribuciones gaussianas para simplificar su registro.

Segmentación y Reconstrucción

Posteriormente, las nubes de puntos se segmentan y clasifican para identificar objetos o características específicas. Esto se logra mediante:

- Métodos geométricos basados en la densidad y normal de los puntos.
- Redes neuronales como *PointNet* y *DeepLab3D*, que aprenden patrones complejos en datos tridimensionales.

Aplicaciones del LiDAR

El LiDAR se utiliza ampliamente en áreas como:

- Arqueología: Para mapear zonas inexploradas.
- Ingeniería civil: Monitoreo de construcciones y terrenos.
- Cartografía: Generación de mapas topográficos de alta resolución.
- Exploración subterránea: Captura de túneles o cavernas inaccesibles.

2.1.2. Variedades de LIDAR

Existen diferentes tipos de Lidar, cada uno con sus correspondientes especificaciones:

LDROBOT D300 Lidar Kit

El **LDROBOT D300 Lidar Kit** es un sensor LiDAR económico con un rango de detección de **0.05 a 12 metros** y una precisión de **±2 mm**. Su frecuencia de operación es de **12 Hz**, lo que lo hace adecuado para proyectos educativos y de robótica que no requieren una actualización rápida de los datos. Este sensor es una opción accesible con un costo aproximado de **$150**, y se utiliza comúnmente en aplicaciones de bajo costo o proyectos DIY (Do It Yourself).

Especificaciones:

- **Rango de detección**: 0.05 - 12 m
- **Precisión**: ±2 mm
- **Frecuencia**: 12 Hz
- **Costo**: $150
- **Aplicaciones**: Robótica, investigación educativa, mapeo de pequeñas áreas.

Figura 2.1: Sensor LDROBOT LIDAR KIT
Fuente:https://eu.robotshop.com/de/products/ldrobot-d300-lidar-kit

Velodyne VLP-16

El **Velodyne VLP-16** es un sensor LiDAR de alta gama utilizado principalmente en vehículos autónomos y aplicaciones de mapeo a gran escala. Con un rango de detección de **0.5 a 100 metros** y una precisión de **±3 cm**, ofrece una frecuencia de operación entre **10 y 20 Hz**. Su precio es más elevado, oscilando entre **$4,000 y $8,000**, lo que lo convierte en una opción adecuada para investigaciones avanzadas y desarrollos industriales.

Especificaciones:

- **Rango de detección**: 0.5 - 100 m
- **Precisión**: ±3 cm
- **Frecuencia**: 10 - 20 Hz
- **Costo**: $4,000 - $8,000
- **Aplicaciones**: Vehículos autónomos, mapeo de grandes áreas, monitoreo industrial.

Figura 2.2: Velodyne VLP-16
Fuente:https://www.techinsights.com/products/ddt-1902-808

RPLIDAR A1

El **RPLIDAR A1** es un sensor LiDAR de bajo costo, con un rango de detección de **0.15 a 12 metros** y una precisión de **±2 %**. Su frecuencia de operación varía entre **5 y 10 Hz**, lo que lo hace adecuado para aplicaciones de mapeo a pequeña escala o vehículos autónomos simples. Su bajo precio, de **$100 a $150**, lo convierte en una opción popular para proyectos de robótica y sistemas de navegación de bajo presupuesto.

Especificaciones:

- **Rango de detección**: 0.15 - 12 m
- **Precisión**: ±2 %
- **Frecuencia**: 5 - 10 Hz
- **Costo**: $100 - $150
- **Aplicaciones**: Robótica, vehículos autónomos a pequeña escala, mapeo básico.

Figura 2.3: Sensor RPLIDAR A1
Fuente:https://www.amazon.com/RPLIDAR-A1-Omnidirectional-High-Speed-Acquisition/dp/B0B6B5MWSJ

Hokuyo UTM-30LX

El **Hokuyo UTM-30LX** es un sensor LiDAR de rango medio, con un alcance de **0.1 a 30 metros** y una precisión de **±30 mm**. Su frecuencia de operación es de **40 Hz**, lo que lo hace adecuado para aplicaciones en las que se requiere una actualización rápida de los datos. Este sensor tiene un costo que varía entre **$3,500 y $4,500** y se utiliza principalmente en robótica avanzada y vehículos autónomos.

Especificaciones:

- **Rango de detección**: 0.1 - 30 m
- **Precisión**: ±30 mm
- **Frecuencia**: 40 Hz
- **Costo**: $3,500 - $4,500
- **Aplicaciones**: Robótica avanzada, vehículos autónomos, navegación en tiempo real.

Figura 2.4: Sensor Hokuyo UTM-30LX
Fuente:https://www.hokuyo-aut.jp/search/single.php?serial=169

Ouster OS1-16

El **Ouster OS1-16** es un sensor LiDAR de alta gama que ofrece un rango de detección de **0.5 a 120 metros** y una precisión de **±3 cm**. Su frecuencia de operación varía entre **10 y 20 Hz**, lo que lo hace ideal para aplicaciones de vehículos autónomos y mapeo en tiempo real. Con un costo de **$6,000 a $10,000**, este sensor es adecuado para aplicaciones industriales y de investigación avanzada.

Especificaciones:

- **Rango de detección**: 0.5 - 120 m
- **Precisión**: ±3 cm
- **Frecuencia**: 10 - 20 Hz
- **Costo**: $6,000 - $10,000
- **Aplicaciones**: Vehículos autónomos, robótica industrial, mapeo de grandes áreas.

Figura 2.5: Sensor Ouster OS1-16
Fuente:https://www.ebay.com/itm/355817239101

2.1.3. Cámaras estéreo y RGB-D

Las cámaras estéreo y RGB-D son tecnologías utilizadas para la captura de nubes de puntos tridimensionales. Estas cámaras permiten obtener representaciones 3D de entornos complejos mediante la captura de imágenes desde diferentes ángulos o combinando imágenes RGB con información de profundidad.

Cámaras Estéreo

Las cámaras estéreo funcionan imitando la visión binocular humana, utilizando dos cámaras situadas en posiciones ligeramente separadas. Al capturar dos imágenes desde diferentes perspectivas, el sistema puede calcular la disparidad entre los puntos y, mediante triangulación, generar una representación tridimensional de la escena. La disparidad (d) entre las imágenes se relaciona con la profundidad (Z) del punto 3D a través de la ecuación:

$$Z = \frac{f \cdot B}{d}$$

donde f es la distancia focal de las cámaras, B es la base estereoscópica (distancia entre las cámaras) y d es la disparidad entre los puntos correspondientes en las dos imágenes.

Cámaras RGB-D

Las cámaras RGB-D, como las de *Intel RealSense* o *Microsoft Kinect*, van un paso más allá al capturar simultáneamente imágenes RGB y mapas de profundidad. Estas cámaras pueden utilizar diversas tecnologías, como el *Time-of-Flight (ToF)* y la *luz estructurada*, para medir la distancia a los objetos y generar nubes de puntos precisas.

- Time-of-Flight (ToF) Las cámaras ToF miden el tiempo que tarda un pulso de luz en reflejarse desde un objeto y regresar al sensor, permitiendo calcular la distancia de cada punto en la escena. Esta técnica proporciona mediciones de profundidad muy precisas y es ideal para entornos de alta complejidad.
- Luz Estructurada El método de luz estructurada proyecta un patrón de luz sobre la escena y mide cómo se deforma dicho patrón al incidir sobre las superficies. Esto permite calcular la profundidad de cada punto y crear una nube de puntos densa y detallada.

Ejemplos de Cámaras RGB-D

- *Intel RealSense*: Estas cámaras ofrecen modelos como el D415 y D435, que combinan imágenes RGB con mapas de profundidad obtenidos mediante tecnología de código de patrón y triangulación. - *Microsoft Kinect*: Originalmente diseñado para videojuegos, el Kinect se ha utilizado ampliamente en investigación para la captura de datos 3D mediante la proyección de patrones de luz estructurada.

Estas cámaras tienen diversas aplicaciones en áreas como la reconstrucción de entornos 3D, la robótica, la realidad aumentada y la exploración de superficies inaccesibles.

Aplicaciones

Las tecnologías de cámaras estéreo y RGB-D son ampliamente utilizadas en campos como: - Reconstrucción 3D de entornos complejos e inaccesibles. - Robótica y navegación autónoma. - Realidad Aumentada (AR) y Realidad Virtual (VR).

2.1.4. Fotogrametría

La fotogrametría es una técnica de captura de datos tridimensionales que utiliza imágenes fotográficas para obtener medidas precisas de objetos y superficies. A través de la fotogrametría, se pueden reconstruir modelos 3D a partir de imágenes 2D, aprovechando las coordenadas espaciales de los puntos de referencia en las imágenes tomadas desde diferentes ángulos. Esta técnica es muy útil en la reconstrucción de nubes de puntos, especialmente en situaciones donde otros sensores no son viables (Mikhail, Bethel, y McGlone, 1974).

Principio de Funcionamiento

El principio básico de la fotogrametría se basa en la triangulación. Al capturar varias imágenes de un objeto desde diferentes perspectivas, se pueden identificar puntos comunes en las imágenes. La geometría del sistema de cámaras permite calcular la posición 3D de esos puntos en el espacio (Jiang y Zhang, 2017). La triangulación se realiza utilizando las coordenadas de los puntos correspondientes en dos imágenes y la geometría del sistema de cámaras.

Fotogrametría y Reconstrucción de Nubes de Puntos

En la fotogrametría, se emplean algoritmos para encontrar puntos de interés comunes entre imágenes (puntos de coincidencia) y calcular las coordenadas tridimensionales de esos puntos. Para generar nubes de puntos, es necesario tener al menos dos imágenes, pero a mayor cantidad de imágenes, se logra una mayor precisión en la reconstrucción. Esta técnica ha sido utilizada extensivamente en áreas como la cartografía, la arqueología, y la ingeniería civil (Jiang y Zhang, 2017).

Herramientas y Software

Existen diversas herramientas de fotogrametría como *Agisoft Metashape* y *Pix4D*, que permiten procesar imágenes para generar nubes de puntos detalladas y modelos 3D. Estas herramientas utilizan algoritmos avanzados de procesamiento de imágenes para mejorar la precisión y resolución de las nubes de puntos generadas (Mikhail y cols., 1974).

2.1.5. Sensores complementarios

En la captura de nubes de puntos, se utilizan diferentes tipos de sensores complementarios que mejoran la precisión y eficiencia del proceso. Estos sensores, cuando se combinan con tecnologías como LiDAR o cámaras RGB-D, pueden proporcionar información adicional valiosa que ayuda a reconstruir superficies complejas y de difícil acceso.

Sensores de Ultrasonido y Radar

Los sensores de ultrasonido y radar destacan como tecnologías útiles en situaciones donde los métodos ópticos, como las cámaras o los sensores láser, pueden fallar debido a condiciones ambientales adversas, como niebla, oscuridad o alta humedad. Estas tecnologías funcionan utilizando ondas mecánicas (ultrasonido) o electromagnéticas (radar) para detectar objetos y medir distancias.

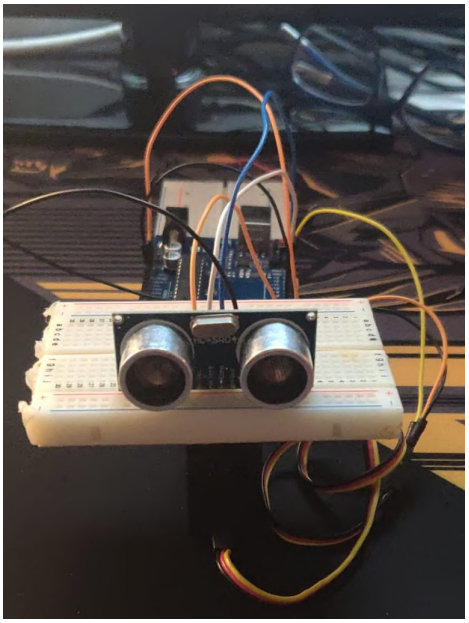

Figura 2.6: Sensor de tipo ultrasónico
Fuente:https://blogs.etsii.urjc.es/dseytr/radar-por-ultrasonidos/

El ultrasonido emplea ondas acústicas de alta frecuencia que rebotan en los objetos cercanos, calculando su distancia con base en el tiempo que tardan en regresar al sensor (Manohar y Singh, 2022). En contraste, los sensores de radar operan mediante ondas electromagnéticas, ofreciendo la ventaja de ser menos sensibles a condiciones atmosféricas extremas, lo que los convierte en herramientas confiables para aplicaciones de detección a mayor escala y en tiempo real (W. Zhang y Kim, 2023). Estas tecnologías son complementarias y pueden integrarse con otros sistemas para mejorar la percepción ambiental en aplicaciones como la navegación autónoma, la robótica o el monitoreo industrial.

Figura 2.7: Sensor de tipo radar
Fuente:https://www.elion.es/productos/sensores/radar/

Cámaras de Alta Velocidad

Las cámaras de alta velocidad se utilizan como herramientas clave en aplicaciones donde es necesario capturar imágenes de objetos en movimiento a gran velocidad. Estas cámaras son capaces de registrar datos en intervalos de tiempo extremadamente cortos,

permitiendo la obtención de imágenes precisas para la reconstrucción en tiempo real de superficies dinámicas o en rápida transición (Bernal y Liu, 2023).

En el contexto de la reconstrucción 3D, las cámaras de alta velocidad complementan otros métodos de captura de nubes de puntos, como los sistemas LiDAR o los sensores basados en luz estructurada. Estas tecnologías generan datos tridimensionales mediante el escaneo de superficies, ofreciendo alternativas versátiles según las condiciones de uso y las características del entorno (Wulf y Carter, 2022). La integración de múltiples métodos de captura en un sistema puede mejorar significativamente la calidad y precisión de los modelos tridimensionales generados, especialmente en escenarios con objetos móviles o entornos cambiantes.

Figura 2.8: Cámaras de alta velocidad
Fuente:https://telemetry.com.pe/index.php/producto/la-camara-de-alta-velocidad-mas-popular-de-la-industria/

Fusión de Sensores

La fusión de datos de diferentes sensores complementarios mejora la calidad de la nube de puntos, ya que cada tipo de sensor tiene sus propias fortalezas y limitaciones. La combinación de LiDAR, cámaras RGB-D, sensores inerciales y otros permite superar las limitaciones individuales de cada tecnología, generando modelos 3D más precisos y completos (?, ?).

2.2. Componentes del prototipo

2.2.1. Hardware

1. **Dron:** En este proyecto se utilizó el dron **DJI Avata 2**, un dispositivo compacto y potente, diseñado para ofrecer una experiencia de vuelo inmersiva, ideal para la captura de datos de alta calidad en condiciones dinámicas. El DJI Avata 2 cuenta con un diseño ligero y un sistema de transmisión de alta calidad, lo que lo convierte

en una herramienta adecuada para aplicaciones como la captura de imágenes, videos en alta definición y experimentos de alta velocidad.

El dron tiene las siguientes especificaciones clave:

- **Dimensiones:** 185 × 212 × 64 mm.
- **Peso:** 377 gramos.
- **Cámara:** Sensor CMOS de 1/1.3 pulgadas, con una lente de 12 mm f/2.8 y un campo de visión de 155 grados. La resolución de video es de hasta 4K a 50/60 fps y la de foto es de hasta 4000 × 3000 píxeles.
- **Velocidad máxima:** 28.8 km/h en modo Normal, 57.6 km/h en modo Sport y 97.2 km/h en modo Manual.
- **Tiempo máximo de vuelo:** Hasta 23 minutos.
- **Alcance máximo de transmisión:** 13 km en modo FCC.
- **Estabilización de imagen:** Rocksteady 3.0.
- **Memoria interna:** 46 GB.

Para más detalles, se puede consultar la página oficial de DJI (DJI, 2023).

Figura 2.9: Drone DJI Avata 2
Fuente:https://drondji.com/producto/dji-avata-2-fly-more-combo-1-bateria/

2. **Esp32:** En este proyecto, el ESP32 cumple un papel importante en la comunicación inalámbrica entre los diferentes componentes del sistema, específicamente entre el dron, el sensor LiDAR y la estación de control. A continuación se detalla su funcionalidad:

   a) **Gestión de la comunicación inalámbrica:** El ESP32 es un microcontrolador con capacidades de conectividad Wi-Fi y Bluetooth. En este proyecto,

se utiliza para transmitir los datos capturados por el sensor LiDAR al sistema de control terrestre en tiempo real. Dado que el sensor LiDAR genera grandes cantidades de datos, el ESP32 puede encargarse de establecer un canal de comunicación confiable y de baja latencia.

b) **Sincronización de datos:** El ESP32 asegura que los datos provenientes del LiDAR sean transmitidos de manera sincronizada con el sistema de control del dron. Esto es importante para que la nube de puntos capturada se corresponda con las coordenadas precisas del dron en cada momento. Esta sincronización permite una reconstrucción más precisa.

c) **Interfaz entre los sistemas de captura y procesamiento:** El ESP32 actúa como un intermediario entre el sensor LiDAR y la computadora de control. Dado que el sensor genera datos en tiempo real, el ESP32 los recibe y los envía de manera eficiente a la computadora para su procesamiento inmediato.

d) **Telemetría:**En algunos casos, el ESP32 también podría ayudar en la transmisión de datos de telemetría del dron, como su posición, altura y orientación, para complementar los datos de la nube de puntos y asegurar que la reconstrucción 3D esté georreferenciada con precisión.

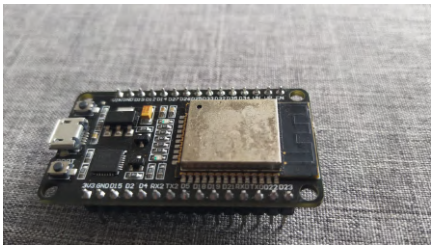

Figura 2.10: ESP32.
Fuente:Propia

3. **Sensor LiDAR:** El sensor LiDAR es un componente fundamental en este proyecto, ya que permite la captura de datos tridimensionales del entorno mediante la emisión de pulsos láser y el análisis del tiempo que estos tardan en regresar tras reflejarse en superficies. A continuación, se describen las principales funciones del sensor LiDAR en este contexto:

   a) **Captura de la nube de puntos:** El sensor es responsable de generar una nube de puntos tridimensional que representa las coordenadas espaciales del entorno escaneado. Esta nube de puntos es esencial para la reconstrucción 3D precisa de las superficies, proporcionando un mapa detallado del área escaneada. Recuperado de (DDRobot, 2023).

   b) **Escaneo de superficies de difícil acceso:** Una de las ventajas principales del uso del LiDAR en este proyecto es su capacidad para escanear áreas de

difícil acceso, como túneles o cavidades, sin riesgo para los humanos. Montado en el dron, el sensor permite recopilar datos desde una distancia segura, lo que lo hace ideal para ambientes peligrosos o inaccesibles. Recuperado de (DDRobot, 2023).

c) **Medición precisa de distancias:** El LiDAR proporciona mediciones con una precisión que varía entre milímetros y centímetros, dependiendo del modelo utilizado. Esta precisión es crítica para obtener una representación detallada del entorno, necesaria para la exactitud en la reconstrucción 3D. Recuperado de (DDRobot, 2023).

d) **Captura de datos en tiempo real:** Mientras el dron realiza su vuelo preprogramado, el sensor LiDAR captura datos en tiempo real, lo que permite una recolección rápida y eficiente de la información del entorno. Esta capacidad es esencial para la creación de modelos tridimensionales dinámicos y en tiempo real. Recuperado de (DDRobot, 2023).

e) **Alta tasa de captura de puntos:** El sensor utilizado en este proyecto tiene una tasa de captura elevada, capaz de generar una gran cantidad de puntos en un corto período de tiempo, lo que es fundamental para la precisión del mapeo. El LiDAR puede tener un rango efectivo que puede alcanzar varios metros, lo que permite escanear áreas grandes sin la necesidad de múltiples vuelos o maniobras complicadas. Recuperado de (DDRobot, 2023).

f) **Alto rango de escaneo:** El modelo de LiDAR empleado puede detectar objetos a distancias de hasta 10 metros, dependiendo del entorno y las condiciones de vuelo. Esto es útil para cubrir superficies extensas sin necesidad de realizar múltiples recorridos. Recuperado de (DDRobot, 2023).

Figura 2.11: Sensor LIDAR usado.
Fuente:Propia

4. **Computadora de control:** La computadora de control es una estación de proce-

samiento equipada con un procesador Ryzen 5600 RTX 3070 ti, 32 GB de RAM, disco NVME M2 1TB para el almacenamiento rápido de los datos capturados. El sistema operativo utilizado es Window 10 y cuenta con las bibliotecas necesarias, como Open3D, PCL (Point Cloud Library), y ROS (Robot Operating System) para la interpretación y procesamiento de los datos de la nube de puntos.

2.2.2. Software

Arduino

Se trata de un software de código abierto que posibilita el desarrollo de proyectos electrónicos interactivos. Se puede considerar como un entorno de programación que simplifica tanto la redacción como la carga de código en la placa. Este entorno se fundamenta en el lenguaje de programación Wiring, lo que facilita la escritura de código y su posterior carga en placas como Arduino o ESP32.

Python

Python es un lenguaje de programación de alto nivel, interpretado y de uso general que ha adquirido gran popularidad en múltiples campos, incluyendo el desarrollo web, la ciencia de datos, la inteligencia artificial, la automatización, entre otros. Su creador, Guido van Rossum, lo presentó por primera vez en 1991. Este lenguaje se distingue por su claridad y simplicidad, lo que lo convierte en una opción accesible para quienes desean aprender y escribir código.

- **Sintaxis sencilla:** Para (Lutz, 2013), Python presenta una sintaxis sencilla y directa que facilita a los programadores la articulación de ideas utilizando un menor número de líneas de código en comparación con otros lenguajes de programación.
- **Gran comunidad y bibliotecas:** Para (McKinney, 2017), menciona que Python dispone de una vasta comunidad de usuarios y una rica variedad de bibliotecas, lo que permite la incorporación de múltiples funcionalidades sin requerir la programación desde el inicio.
- **Portabilidad:** Como se indica en la referencia de van (van Rossum y Drake, 2009), Python es un lenguaje de programación que opera en múltiples plataformas, permitiendo su ejecución en diversos sistemas operativos, tales como Windows, macOS y Linux.
- **Aplicaciones Comunes:** (Pedregosa y cols., 2011), se señala que Python goza de una notable popularidad en el campo de la ciencia de datos y el aprendizaje automático, debido a la eficacia de sus robustas bibliotecas, tales como NumPy y scikit-learn.

Bibliotecas y herramientas

Para el desarrollo del software, se utilizan varias bibliotecas en Python y Arduino, todas actualizadas a las versiones mencionadas, que incluyen:

- `Socket` (versión 3.10.6): Esta biblioteca es fundamental para establecer la comunicación entre el dron y la estación terrestre. Permite la transmisión de datos en tiempo real, asegurando que la información capturada por el sensor LiDAR sea recibida y procesada adecuadamente en la computadora de control como menciona (Python Software Foundation, s.f.-b).

- `Threading` (versión 3.10.6): Según (Python Software Foundation, s.f.-c), esta biblioteca se utiliza para manejar múltiples tareas en paralelo. Al permitir la ejecución simultánea de diferentes hilos, facilita el procesamiento en tiempo real de los datos de la nube de puntos, mejorando la eficiencia del sistema .

- `Numpy` (versión 1.26.3): Esta biblioteca es esencial para la manipulación eficiente de datos numéricos. Con su capacidad para manejar arreglos multidimensionales, según (Harris, Millman, van der Walt, y cols., 2020), `numpy` permite realizar operaciones matemáticas complejas sobre las nubes de puntos de manera eficiente.

- `Open3d` (versión 0.16.0): Esta biblioteca se utiliza para el procesamiento y visualización de nubes de puntos. Proporciona herramientas para la reconstrucción 3D, cálculo de normales y otras operaciones relacionadas con datos tridimensionales, lo que es fundamental para la interpretación de los datos capturados por el sensor LiDAR (Zhou, Park, y Koltun, 2018).

- `Plyfile` (versión 1.0.3): Esta biblioteca permite la lectura y escritura de archivos en formato PLY, que es comúnmente utilizado para almacenar nubes de puntos. Facilita la interoperabilidad entre diferentes aplicaciones y herramientas que manejan datos 3D (Davies, 2017).

- `Torch` (versión 2.0.1): Esta biblioteca se utiliza para integrar redes neuronales en el procesamiento de datos. Permite aplicar técnicas de aprendizaje profundo para mejorar la calidad de la reconstrucción de superficies a partir de nubes de puntos (Paszke, Gross, Massa, Lerer, y cols., 2019)

- `TensorFlow` (versión 2.15.0): Esta biblioteca es fundamental para la implementación de modelos de aprendizaje profundo. Se utiliza para tareas de procesamiento y análisis de datos en tiempo real, aplicando técnicas de inteligencia artificial para mejorar la precisión de la reconstrucción 3D de acuerdo (TensorFlow, s.f.).

- `Math` (versión 3.10.6): La biblioteca `math` es utilizada para realizar operaciones matemáticas básicas y avanzadas necesarias en el procesamiento de datos del sensor

LiDAR y otros cálculos numéricos dentro del sistema según (Python Software Foundation, s.f.-a).

- `ComputeLidarData`: Este es un módulo personalizado desarrollado específicamente para procesar los datos del sensor LiDAR. Incluye funciones para filtrar, normalizar y transformar los datos de acuerdo a los requisitos del sistema, facilitando el análisis y la visualización de la información capturada.

Entorno de desarrollo: Se utiliza `Visual Studio Code` como entorno de desarrollo para programar y depurar el software en Python, y Arduino IDE para la programación del ESP32.

2.3. Robótica móvil y navegación autónoma

La robótica móvil es una subdisciplina clave dentro de la robótica que se centra en diseñar y desarrollar sistemas robóticos capaces de desplazarse de manera autónoma en entornos diversos. Este campo combina elementos de mecánica, electrónica, informática e inteligencia artificial para crear soluciones versátiles que abarcan desde vehículos autónomos hasta robots exploradores de terreno y aplicaciones de servicio en interiores.

Definición y relevancia de la robótica móvil:

La robótica móvil se distingue por su capacidad de movimiento, lo que permite a los robots realizar tareas en entornos dinámicos. A diferencia de los robots estacionarios, los sistemas móviles enfrentan desafíos relacionados con la navegación, percepción y adaptación al medio ambiente. Estos sistemas son esenciales en aplicaciones como exploración, logística y rescate, donde los entornos pueden ser impredecibles y no estructurados (Siegwart, Nourbakhsh, y Scaramuzza, 2011a).

Aplicaciones prácticas de la robótica móvil

- **Vehículos autónomos:** Los vehículos autónomos son quizás el ejemplo más conocido de robótica móvil. Empresas como Tesla y Waymo han desarrollado tecnologías avanzadas de navegación autónoma, integrando sensores de alta precisión y algoritmos de aprendizaje profundo para garantizar la seguridad y eficiencia en el tráfico.
- **Exploración y rescate:** En misiones de rescate o exploración de áreas inaccesibles para los humanos, los robots móviles equipados con sensores LIDAR y sistemas SLAM son indispensables. Estos dispositivos pueden cartografiar terrenos desconocidos y enviar datos en tiempo real a operadores remotos.

- **Logística y Almacenamiento:** Robots móviles como los utilizados por Amazon en sus almacenes emplean navegación autónoma para optimizar el movimiento de productos, mejorando la eficiencia operativa.

Desafíos y perspectivas futuras

- **Entornos No Estructurados:** Uno de los mayores desafíos en la robótica móvil es la navegación en entornos impredecibles, como áreas urbanas densas o terrenos accidentados. Los avances en inteligencia artificial y sensores continúan ampliando las capacidades de los robots en estos escenarios.
- **Computación en Tiempo Real:**Procesar grandes cantidades de datos sensoriales en tiempo real sigue siendo un reto técnico importante. La integración de hardware más potente y algoritmos optimizados es esencial para superar estas limitaciones (Thrun, Burgard, y Fox, 2005).

2.4. Procesamiento de datos

2.4.1. Mapeo

Para Pérez J. (2020):

El término mapeo se refiere tanto al acto como al resultado de la acción de mapear. Este verbo, utilizado en el contexto biológico, se relaciona con la representación gráfica de las diversas partes que componen un todo (...). En un sentido más amplio, se puede afirmar que mapear conlleva la elaboración de un mapa o la representación de una estructura o sistema en un formato gráfico que se asemeja a un mapa. De esta manera, se puede inferir que mapear constituye el proceso de organización espacial de un objeto.

2.4.2. Nube de puntos

La nube de puntos se entiende al conjunto de puntos que por la ubicación de sus posición en un sistema de coordenadas estos varían de acuerdo al valor que poseen ya sea de color, profundidad, etc. Estos generalmente se usan para representar la estructura en un espacio tridimensional del objeto o superficies. Esta nube de puntos son generadas por diversos scanner 3D o sensores 3d como el mando Kinect y demás, El propósito de la generación de nubes son diversas como la creación de modelos CAD tridimensionales, la medición de precisión, para usar en el control de productos. El uso de estos sensores Lidar tienen muchas aplicaciones como la verificación en la fabricación de piezas si estas cumplen con las especificaciones requeridas. Tolerancia de diseños, o en la parte médica en el diagnóstico por imágenes. Esta nube de puntos es un punto de partida para la

digitalización de objetos que posteriormente se usarán para crear mallas poligonales Nube de Puntos. (2015). kripkit.com.

2.4.3. Ruido

Según (López Cuevas, 2022) en cualquier recopilación de datos, pueden surgir valores atípicos que distorsionan la muestra, los cuales se conocen como ruido. Este ruido puede presentarse con diferentes frecuencias. La eliminación del ruido en una nube de puntos es beneficiosa para mejorar la calidad de la información. La reducción del ruido es una etapa clave en el proceso de simplificación de datos. Una forma de abordarlo es estableciendo un umbral que permita identificar valores comunes y diferenciar el ruido de manera efectiva. Este enfoque facilita el desarrollo de un proceso para la eliminación de ruido.

2.4.4. Normalización de puntos

La normalización de nubes de puntos es un proceso crucial en la reconstrucción 3D que busca mejorar la calidad de los datos adquiridos. Este proceso implica ajustar la nube de puntos a un sistema de coordenadas estándar, lo que facilita la comparación y el análisis posterior. Además, la normalización ayuda a reducir la variabilidad de las nubes de puntos, que puede ser causada por factores como la posición del sensor, la iluminación y el ruido ambiental de acuerdo con (Y. Zhang y et al., 2019). Para posteriormente, calcular las normales.

Importancia de cálculo de normales

Calcular las normales de superficie es un paso esencial en la reconstrucción de nubes de puntos. Las normales representan la orientación de las superficies en cada punto de la nube, lo que permite comprender mejor la geometría del objeto que se está modelando. Sin las normales, la reconstrucción de la superficie puede resultar inexacta, ya que la falta de información sobre la orientación de las superficies puede llevar a errores en la representación de la forma del objeto según (R. Rusu y Cousins, 2011). Además, las normales son fundamentales para diversas tareas en el procesamiento de nubes de puntos, como:

- **Generación de Mallas:** Las normales son utilizadas en algoritmos de triangulación para crear mallas 3D a partir de nubes de puntos, asegurando que la malla refleje correctamente la geometría del objeto.
- **Sombreado y Renderizado:** Para (Ouhbi y et al., 2020), en gráficos por computadora, las normales son necesarias para calcular la iluminación de las superficies, lo que contribuye a un renderizado más realista.

- **Segmentación y Clasificación** La información de las normales puede ser utilizada para identificar diferentes partes de un objeto o para clasificar las superficies según su orientación.

2.4.5. Aplicaciones prácticas de la exploración 3D

- **Robótica móvil y navegación Autónoma** Como mencionamos anteriormente esta es una de las aplicaciones que en la robótica móvil, es crucial para permitir a los robots entender y navegar en su entorno. Utilizando sensores como LIDAR o cámaras estéreo, los robots pueden generar representaciones tridimensionales precisas de su entorno, lo que les permite tomar decisiones informadas sobre cómo desplazarse, evitar obstáculos y realizar tareas específicas (Siegwart et al., 2011). Esto es especialmente útil en entornos complejos o desconocidos, donde el robot no tiene un mapa predefinido.

  Además, los robots de exploración autónomos, como aquellos utilizados en misiones espaciales o de rescate, se benefician enormemente de la capacidad de crear mapas 3D del terreno o de estructuras, lo que les permite planificar rutas de manera más eficiente y segura.

- **Cartografía y modelado de entornos complejos** La creación de modelos 3D de entornos complejos, como ciudades, túneles o zonas afectadas por desastres, es otra aplicación importante. Los sensores 3D, como LIDAR o fotogrametría, permiten mapear grandes áreas y generar modelos tridimensionales detallados. Estos modelos se pueden utilizar para la planificación urbana, la rehabilitación de infraestructuras y el análisis de daños después de desastres naturales (R. B. Rusu, Blodow, y Beetz, 2011). En el contexto de los túneles o zonas subterráneas, la exploración 3D también se usa para evaluar condiciones de seguridad, identificar posibles riesgos y optimizar las rutas de acceso.

- **Reconstrucción de paisajes para realidad aumentada (AR) y realidad virtual (VR)** En la industria del entretenimiento, la exploración 3D se utiliza para crear entornos virtuales realistas. Utilizando datos 3D generados por sensores como LIDAR, es posible crear representaciones precisas de paisajes, edificios y objetos para su uso en aplicaciones de realidad aumentada (AR) y realidad virtual (VR). Estas tecnologías se aplican en áreas como la educación, los videojuegos, el turismo virtual y las simulaciones de entrenamiento (Thrun y cols., 2005).

  Por ejemplo, en los videojuegos, los paisajes y los escenarios 3D pueden ser modelados a partir de datos de exploración 3D, proporcionando una experiencia inmersiva para los jugadores. En la educación, los modelos 3D permiten la visualización interactiva de conceptos complejos como anatomía, historia o geografía.

- **Inspección y mantenimiento de infraestructuras** Los sistemas 3D también se utilizan para la inspección y el mantenimiento de infraestructuras, como puentes, edificios y presas. Utilizando robots equipados con sensores 3D, es posible crear modelos detallados de las estructuras y evaluar su estado sin necesidad de intervenir físicamente. Estos modelos 3D permiten identificar áreas de daño o desgaste, lo que facilita el mantenimiento predictivo y mejora la seguridad (Siegwart, Nourbakhsh, y Scaramuzza, 2011b). Esta aplicación es particularmente útil en lugares de difícil acceso o en situaciones donde el tiempo y la seguridad son críticos, como en la inspección de líneas eléctricas o plataformas petroleras.

- **Medicina y cirugía asistida por computadora** En el ámbito médico, la exploración 3D se utiliza para crear modelos precisos de órganos y estructuras anatómicas a partir de imágenes médicas como resonancias magnéticas (RM) o tomografías computarizadas (TC). Estos modelos pueden ser utilizados para planificar procedimientos quirúrgicos, guiar cirujanos durante intervenciones y crear prótesis personalizadas (Z. Zhang, 2012). La simulación 3D también se emplea en la formación de médicos, permitiéndoles practicar procedimientos en un entorno virtual antes de realizar la cirugía real.

- **Conservación del patrimonio cultural** La preservación digital de patrimonio cultural es otro campo en el que la exploración 3D juega un papel fundamental. A través de técnicas de escaneo 3D, se pueden crear representaciones digitales de monumentos, esculturas y artefactos históricos. Esto no solo permite la preservación del patrimonio en su forma digital, sino que también facilita el acceso remoto a estas obras, incluso en caso de que sufran daños o destrucción(Carlevaris, Della Corte, y Russo, 2018).

- **Agricultura de precisión** En la agricultura de precisión, la exploración 3D se utiliza para mapear terrenos agrícolas y estudiar la distribución de cultivos. A través del análisis de datos 3D obtenidos mediante drones o satélites, los agricultores pueden obtener información precisa sobre el estado de sus cultivos, identificar áreas que requieren atención y optimizar el uso de recursos como agua y fertilizantes (R. B. Rusu y cols., 2011). Este tipo de tecnología también permite la automatización de tareas agrícolas mediante robots autónomos que navegan y trabajan en los campos.

2.4.6. Procesamiento y representación de datos 3D

El procesamiento de datos 3D juega un papel crucial en la creación de modelos tridimensionales precisos, necesarios para diversas aplicaciones en robótica y visión por

computadora. Este proceso implica la captura de datos de sensores, la eliminación de ruido, la alineación de nubes de puntos y la representación final en forma de modelos 3D.

- **Adquisición de datos 3D** Los datos 3D se pueden adquirir mediante una variedad de sensores. El LIDAR (Light Detection and Ranging) es uno de los más comunes en robótica debido a su capacidad para mapear el entorno con alta precisión utilizando pulsos láser. A través de este sensor, es posible obtener datos de distancia que se traducen en nubes de puntos, las cuales representan la geometría del entorno (Z. Zhang, 2012).

  Además de los sensores LIDAR, las cámaras estéreo también juegan un papel importante. Al capturar imágenes desde dos ángulos ligeramente diferentes, estas cámaras permiten estimar la profundidad de los objetos en una escena, generando mapas tridimensionales (Siegwart y cols., 2011b).

- **Filtrado y procesamiento de nubes de puntos** Una vez adquiridos los datos, se aplica un procesamiento para mejorar la calidad de las nubes de puntos. El filtrado de ruido es uno de los primeros pasos en este proceso. Técnicas como el filtrado de media móvil y el uso de algoritmos como RANSAC (Random Sample Consensus) ayudan a eliminar los puntos que no corresponden al objeto o al entorno real (Thrun y cols., 2005).

  El registro de nubes de puntos es otra fase crítica. Este paso se utiliza para alinear múltiples nubes de puntos adquiridas desde diferentes perspectivas o momentos. El algoritmo de Iterative Closest Point (ICP) es ampliamente utilizado para optimizar esta alineación, asegurando que las nubes de puntos se integren adecuadamente para formar una representación precisa (R. B. Rusu y cols., 2011).

- **Representación y Visualización de Datos 3D** Después de procesar los datos, los resultados se representan visualmente utilizando mallas 3D, que están formadas por vértices, aristas y caras. Estas mallas se crean a partir de las nubes de puntos utilizando algoritmos de triangulación, como la triangulación de Delaunay, que conecta los puntos en una serie de triángulos para formar una superficie continua (Carlevaris y cols., 2018).

  Además, se emplean técnicas de reconstrucción de superficies, como la reconstrucción de Poisson, para convertir las nubes de puntos dispersas en superficies más densas y continuas. Estas técnicas son fundamentales para obtener modelos 3D detallados de alta calidad (Curless y Levoy, 1996).

- **Aplicaciones de los datos 3D** Los modelos 3D resultantes tienen múltiples aplicaciones. En la robótica móvil, los modelos 3D se utilizan para la planificación de rutas, ya que permiten a los robots comprender su entorno y evitar obstáculos de

manera eficiente. También se aplican en la navegación autónoma, donde los robots utilizan estos modelos para tomar decisiones sobre cómo moverse en entornos complejos (Siegwart et al., 2011). En aplicaciones de rescate o exploración, los robots utilizan datos 3D en tiempo real para reconstruir áreas desconocidas y proporcionar información precisa para las tareas de intervención (R. B. Rusu y cols., 2011).

2.4.7. Malla de triángulos 3D

La reconstrucción de un objeto en un entorno tridimensional implica la aplicación de diversas técnicas, entre las cuales se destaca el uso de mallas de triángulos 3D. Esta técnica tiene como finalidad representar en la memoria del ordenador las características de forma y volumen de un objeto, a partir de un conjunto de puntos que han sido previamente obtenidos mediante métodos de mapeo. El proceso de reconstrucción se divide en cinco etapas: posición, registro, integración, segmentación y ajuste. Esta última etapa es crucial, ya que proporciona el modelo computacional del objeto o del entorno a representar (Yvert et al., 2005). La fase más significativa de este procedimiento es la reconstrucción 3D, específicamente en el ajuste de superficies, donde se genera el modelo matemático o la representación computacional del objeto o entorno a representar, según lo indicado en Wikipedia (2022).

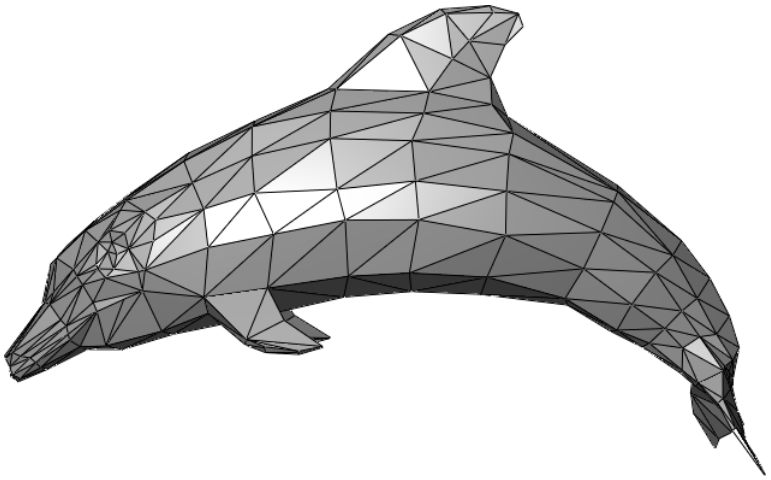

Figura 2.12: Malla de triángulos 3D
Fuente: https://es.wikipedia.org/wiki/Malla_poligonal

2.5. Algoritmos de Reconstrucción 3D

La reconstrucción 3D es un campo crucial en áreas como la visión por computadora, la fotogrametría, la robótica y el modelado 3D. Existen diferentes enfoques para realizar reconstrucciones tridimensionales, que varían según los tipos de datos disponibles (nubes de puntos, imágenes, etc.) y el resultado esperado (superficies suaves, mallas, volúmenes,

etc.). A continuación, se presentan algunos de los algoritmos más relevantes utilizados en la reconstrucción 3D, explicados de manera detallada.

2.5.1. Reconstrucción a partir de Nubes de Puntos

Una de las fuentes más comunes para la reconstrucción 3D son las nubes de puntos, las cuales son colecciones de puntos en un espacio tridimensional, representando una superficie o estructura de un objeto.

Poisson Surface Reconstruction (PSR)

El algoritmo de *Poisson Surface Reconstruction* (PSR) es un método ampliamente utilizado en la reconstrucción de superficies 3D a partir de nubes de puntos dispersas. Este enfoque se basa en resolver una ecuación diferencial de Poisson para interpolar la superficie a partir de los puntos de la nube de forma suave y continua. El principio fundamental de este algoritmo es que la normal de cada punto de la nube es usada como condición de frontera para la ecuación de Poisson, generando una superficie coherente.

El algoritmo fue propuesto por Kazhdan (Kazhdan, Bolitho, y Hoppe, 2006) permite obtener superficies con alta calidad y precisión, incluso cuando los datos de entrada presentan inconsistencias o puntos faltantes. La resolución de la ecuación de Poisson se logra mediante un modelo numérico basado en el uso de técnicas de discretización de mallas, donde se minimizan las inconsistencias en la superficie resultante. El proceso involucra la solución de un sistema de ecuaciones lineales en el espacio tridimensional, lo que se realiza típicamente mediante el método de la *factorización de matrices* o el uso de técnicas de *gradientes conjugados* para optimizar la superficie reconstruida.

Este método es especialmente útil cuando se trata de nubes de puntos que contienen ruido o huecos en los datos. La ecuación de Poisson se ajusta de manera que estos huecos se rellenen suavemente, creando una superficie continua y realista sin necesidad de un modelo de malla explícito. Esta característica es fundamental para la reconstrucción de superficies a partir de datos obtenidos de sensores como LiDAR, que pueden estar incompletos o contener ruido debido a la calidad de la captura de los puntos.

Una de las características clave de este algoritmo es su dependencia de la normal de los puntos en la nube. Las normales son cruciales para la formulación del problema de Poisson. Estas se pueden estimar a partir de la nube de puntos utilizando métodos como los de *K vecinos más cercanos* (KNN), lo que permite calcular una orientación local de la superficie en cada punto. Luego, a través de la ecuación de Poisson, se busca interpolar una función potencial que describe la superficie en términos de una aproximación suave a los datos. El proceso completo se resuelve numéricamente con un enfoque iterativo para obtener la superficie más ajustada.

Para resolver el sistema de Poisson, se utiliza la técnica de *matrices dispersas*, ya que la

matriz del sistema es muy grande y es fundamental mantener la eficiencia computacional. El uso de métodos de factorización directa o gradientes conjugados ayuda a garantizar que el algoritmo sea escalable para nubes de puntos grandes.

El algoritmo ha sido implementado en bibliotecas de código abierto como *MeshLab* y *Open3D*, que proporcionan interfaces para su integración en aplicaciones de reconstrucción 3D.

Ball Pivoting Algorithm (BPA)

El *Ball Pivoting Algorithm* (BPA) es un algoritmo que se utiliza en la reconstrucción de superficies 3D a partir de nubes de puntos dispersas. A diferencia de otros métodos, el BPA simula el proceso físico de un "pivotaje"de una esfera sobre los puntos de la nube, donde la esfera se mueve a lo largo de los puntos y conecta aquellos que están lo suficientemente cerca unos de otros. Este algoritmo genera una malla 3D al hacer girar la esfera sobre los puntos y formando las conexiones necesarias entre ellos.

El BPA fue desarrollado por Bernardini (Bernardini, Mittleman, y Rushmeier, 1999) en 1999 y ha demostrado ser altamente efectivo en la reconstrucción de superficies a partir de nubes de puntos densas. Su eficiencia radica en la capacidad de generar mallas rápidamente y con alta precisión, incluso en escenarios con una gran cantidad de puntos. Este algoritmo es particularmente útil en aplicaciones de escaneo 3D y modelado digital, donde se requiere una reconstrucción precisa de la geometría de un objeto a partir de datos obtenidos por sensores como LiDAR o fotogrametría.

El BPA es especialmente eficaz para manejar nubes de puntos con alta densidad, ya que el método de pivoteo es capaz de crear una malla de superficie rápidamente sin requerir demasiados cálculos computacionales. Además, este enfoque es menos sensible al ruido presente en los datos de entrada, lo que lo hace robusto ante imperfecciones en las nubes de puntos, como la presencia de puntos atípicos o dispersos. La simplicidad y la rapidez de su ejecución han hecho que el BPA sea ampliamente utilizado en la industria del modelado 3D y en aplicaciones de escaneo digital.

Greedy Triangulation

El *Greedy Triangulation* (Triangulación Codiciosa) es un algoritmo simple y eficiente utilizado en la reconstrucción de superficies 3D. Este enfoque trabaja de manera iterativa conectando puntos cercanos entre sí para formar triángulos, lo que genera una malla que representa la superficie 3D. A medida que se agregan triángulos a la malla, el algoritmo busca siempre el par de puntos más cercanos que aún no han sido conectados, lo que da lugar a una solución rápida y de bajo costo computacional.

El algoritmo de triangulación codiciosa fue propuesto por (Attene y Falcidieno, 2006), ha demostrado ser eficaz en la generación de modelos 3D simples pero efectivos. A pesar

de su simplicidad y rapidez, este algoritmo tiene limitaciones en cuanto a su capacidad para manejar nubes de puntos muy ruidosas o dispersas, ya que no realiza una filtración de ruido durante el proceso de triangulación. Sin embargo, cuando se aplica a nubes de puntos densas y bien distribuidas, puede generar mallas de buena calidad en poco tiempo.

El enfoque codicioso es particularmente útil en aplicaciones donde la velocidad es una prioridad y se pueden aceptar ciertas simplificaciones en la precisión de la reconstrucción de la superficie. Este algoritmo ha sido ampliamente utilizado en la creación de modelos 3D a partir de datos obtenidos por sensores como LiDAR o fotogrametría, especialmente cuando se necesita una solución rápida y eficiente para la reconstrucción.

2.5.2. Reconstrucción a partir de Imágenes

Cuando se cuenta con múltiples imágenes de una escena desde diferentes ángulos, se pueden emplear diversos métodos para reconstruir el modelo 3D de manera eficiente.

Structure from Motion (SfM)

El algoritmo de *Structure from Motion* (SfM) es una técnica fundamental en la reconstrucción 3D a partir de imágenes 2D. Este enfoque permite estimar la estructura tridimensional de una escena a partir de un conjunto de imágenes tomadas desde diferentes posiciones. Además, el algoritmo también estima las posiciones y orientaciones de las cámaras que capturaron esas imágenes. El proceso se lleva a cabo mediante la extracción de puntos de correspondencia entre las imágenes y la resolución de un problema de estimación de la estructura y la cámara de manera simultánea.

El enfoque SfM fue popularizado por Snavely (Snavely, Seitz, y Szeliski, 2008) en 2008, y desde entonces ha sido ampliamente utilizado en aplicaciones de fotogrametría, modelado 3D, y reconstrucción de entornos urbanos a partir de fotografías disponibles en línea. El algoritmo es especialmente efectivo cuando se dispone de un gran número de imágenes de alta calidad, lo que permite la reconstrucción de modelos detallados a partir de datos visuales sin necesidad de equipos costosos o especializados.

El SfM es una de las técnicas más utilizadas en la creación de mapas 3D de grandes áreas, como ciudades o paisajes, ya que permite construir modelos tridimensionales de entornos complejos utilizando únicamente fotografías. Es particularmente útil en escenarios donde no es práctico o posible utilizar sensores especializados como LiDAR. A pesar de su efectividad, el algoritmo puede ser sensible a la calidad y la cantidad de las imágenes de entrada, así como a la presencia de puntos de vista limitados o condiciones de iluminación adversas.

Multiview Stereo (MVS)

El enfoque de *Multiview Stereo* (MVS) es una técnica utilizada para la reconstrucción de escenas 3D densas a partir de múltiples vistas de una misma escena. A diferencia de *Structure from Motion* (SfM), que se enfoca en estimar la estructura general de la escena y la localización de las cámaras, MVS se especializa en calcular la profundidad de cada punto en la escena a partir de las imágenes disponibles, lo que permite generar una nube de puntos mucho más densa y detallada. Este enfoque se basa en la correspondencia de puntos entre diferentes vistas y en la optimización de la estimación de la profundidad a través de métodos como la minimización de la energía.

El algoritmo de MVS ha demostrado ser particularmente efectivo en aplicaciones que requieren una alta precisión en la reconstrucción de superficies, tales como la creación de modelos 3D detallados para entornos urbanos o la restauración de escenas históricas. Una de las ventajas clave de MVS es su capacidad para generar nubes de puntos de alta calidad, lo que lo convierte en una herramienta esencial para la creación de modelos 3D precisos en áreas como la fotogrametría y la ingeniería inversa. El uso de MVS ha sido fundamental en la creación de modelos 3D a partir de fotografías aéreas y la restauración de monumentos o edificios históricos, ya que permite obtener una representación tridimensional detallada sin la necesidad de equipos costosos como LiDAR. Sin embargo, uno de los desafíos de MVS es la necesidad de una gran cantidad de imágenes de calidad para garantizar una reconstrucción precisa, lo que puede ser un desafío en entornos donde las vistas disponibles son limitadas (Yuan, Liu, y Yang, 2018).

2.5.3. Reconstrucción con Redes Neuronales y Aprendizaje Profundo

Con el avance de las técnicas de aprendizaje profundo, las redes neuronales se han convertido en herramientas fundamentales para la reconstrucción 3D. Los *autoencoders* y redes especializadas como *PointNet* han demostrado ser eficaces para generar modelos 3D a partir de datos incompletos o con ruido, destacándose especialmente en escenarios con puntos de entrada limitados o desordenados.

PointNet

PointNet es una arquitectura de red neuronal profunda diseñada específicamente para trabajar directamente con nubes de puntos 3D, lo que elimina la necesidad de transformar los puntos en representaciones como mallas o vóxeles. En lugar de eso, PointNet procesa las nubes de puntos tal como están, extrayendo características espaciales y geométricas esenciales para realizar tareas de clasificación, segmentación y reconstrucción 3D. Esta red fue propuesta por Qi et al. en 2017 y ha demostrado ser eficaz en la clasificación y

segmentación de datos 3D, estableciendo un nuevo estándar para los métodos que operan sobre nubes de puntos (Qi, Su, Mo, y Guibas, 2017).

PointNet puede operar directamente sobre conjuntos de puntos desordenados y no estructurados, lo que lo hace ideal para una variedad de aplicaciones de reconstrucción 3D. Este enfoque ha abierto nuevas posibilidades para generar modelos 3D a partir de datos desordenados y ruidosos, haciendo que la reconstrucción sea más accesible en situaciones complejas.

Autoencoders para Reconstrucción 3D

Los *autoencoders* son redes neuronales utilizadas para aprender representaciones comprimidas de los datos. En el contexto de la reconstrucción 3D, los autoencoders pueden aprender una representación latente de las nubes de puntos o imágenes, que luego pueden ser utilizadas para reconstruir la estructura tridimensional. Este enfoque ha demostrado ser especialmente útil para generar modelos 3D de formas complejas, incluso cuando los datos están incompletos o contienen ruido (Wang, Li, y Liu, 2017).

Los autoencoders tienen la ventaja de ser capaces de trabajar con representaciones latentes, lo que les permite ser más eficientes en escenarios con datos limitados o inexactos. Esta capacidad los convierte en una herramienta poderosa en la reconstrucción 3D, especialmente en aplicaciones donde la calidad de los datos no es ideal.

2.5.4. Reconstrucción a partir de Profundidad Estimada

En áreas como la robótica, los sensores de profundidad, tales como los utilizados en dispositivos Kinect o LIDAR, permiten obtener información detallada sobre la geometría 3D de una escena. Estos datos se pueden fusionar para crear representaciones tridimensionales precisas y completas.

Depth Map Fusion

La fusión de mapas de profundidad es una técnica que combina varios mapas de profundidad obtenidos desde diferentes vistas para construir una representación 3D detallada de la escena. Este enfoque es especialmente útil cuando se cuenta con sensores de profundidad como Kinect o LIDAR, ya que permite crear modelos 3D densos y precisos a partir de la información proporcionada por estos sensores (Newcombe y cols., 2011).

El método ha sido clave en aplicaciones como la navegación autónoma de robots y el mapeo 3D en tiempo real. A través de la fusión de los mapas de profundidad generados, se pueden crear modelos precisos de entornos reales, lo que es esencial en tareas de localización y mapeo en robótica.

En resumen, los métodos para la reconstrucción 3D presentan una amplia variedad de enfoques, cada uno con ventajas y limitaciones. La elección del algoritmo adecuado

depende de los datos disponibles y los requisitos específicos de la aplicación. La combinación de técnicas tradicionales, como la interpolación de Poisson, con enfoques basados en redes neuronales, como PointNet, ha permitido avanzar significativamente en la precisión y eficiencia de los modelos 3D. A medida que la tecnología continúa evolucionando, se espera que estos métodos sean más accesibles y eficaces, ofreciendo nuevas oportunidades en campos como la robótica, la medicina y el diseño industrial.

2.6. Inteligencia Artificial

La inteligencia artifical esta siendo uno de los temas que da mucho de hablar en este tiempo, por lo que su avance esta siendo de forma exponencial. Para Hardy T. (2001) la inteligencia artificial tiene como propósito el estudio y análisis del comportamientohumano y transmitirlo a un computador haciendo que este simule las actividades intelectuales del hombre. De acuerdo a Valbuena R. (2021) define como:

Üna ciencia experimental de crecimiento exponencial, donde se modela computacionlmente la inteligencia humana".

2.7. Redes Neuronales

De acuerdo con Matich D. (2001), una red neuronal representa una innovadora modalidad de computación, basada en un modelo biológico que imita el funcionamiento del cerebro humano. Las redes neuronales artificiales consisten en estructuras interconectadas que envían señales desde la entrada hasta la salida.

Las características fundamentales que emulan estas redes neuronales artificiales se pueden sintetizar en tres conceptos clave: procesamiento paralelo, distribución y adaptabilidad, según lo expuesto por Del Brío et al. (2001).

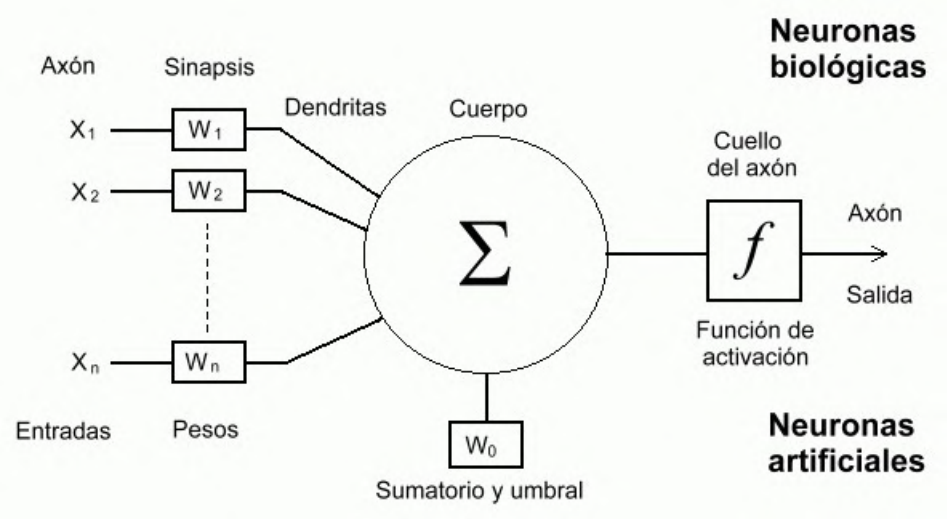

Figura 2.13: Equivalencia entre neurona biológica y una neurona artificial.
Fuente:https://www.um.es/LEQ/Atmosferas/Ch-VI-3/F63s4p3.html

2.7.1. Elementos básicos de una red neuronal

1. **Función de entrada (input function)**

   Para Matich, D. (2001): La neurona considera a todos los valores de entrada como uno, este es llamado como entrada global; por lo que el problema está dado en cómo combinar este conjunto de entradas dentro de la entrada global. Lo cual se logra con la función de entrada, esta función se describe como:

   $$\text{input i} = (\text{ini1} \bullet \text{ w i1})^*(\text{ini2} \bullet \text{w i2 })^* \ldots (\text{inin} \bullet \text{w in })$$

2. **Función de activación (activation function)** Para Freire, E. et al (2019) indica que una función de activación nos da una salida el cual se genera mediante una neurona dada, una entrada o conjunto de entradas. Cada capa conforma que conforma la red tiene una función de activación que permite reconstruir o predecir. Las funciones de activación de clasifica en:

   a) **Función lineal:** Conocida también como función identidad, donde la entrada es igual a la salida, si se tiene una red de varias capas se aplica la función lineal estamos hablando de la regresión lineal.

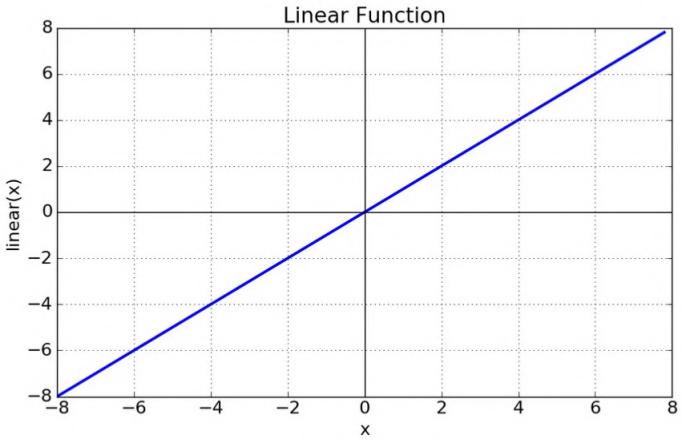

Figura 2.14: Función lineal.
Fuente: https://bootcampai.medium.com/redes-neuronales-13349dd1a5bb

b) **Función no lineal:** Entre estas funciones tenemos:

1) **Función umbral:** También conocida como escalón, indica que si x es menor de cero la neurona va a ser cero pero si es mayor será igual a 1.

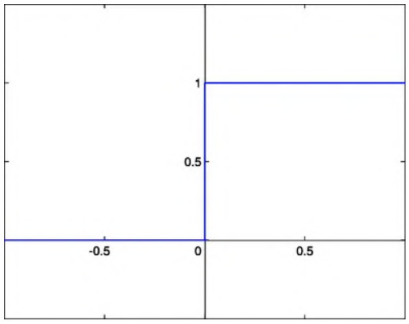

Figura 2.15: Función no lineal umbral.
Fuente:https://bootcampai.medium.com/redes-neuronales-13349dd1a5bb

2) **Función sigmoide**: Esta función cumple con la siguiente ecuación:

$$f(x) = \frac{1}{1-e^{-x}}$$

Es la función más adecuada para la última capa y clasificar datos en dos categorías. Presenta el siguiente gráfico:

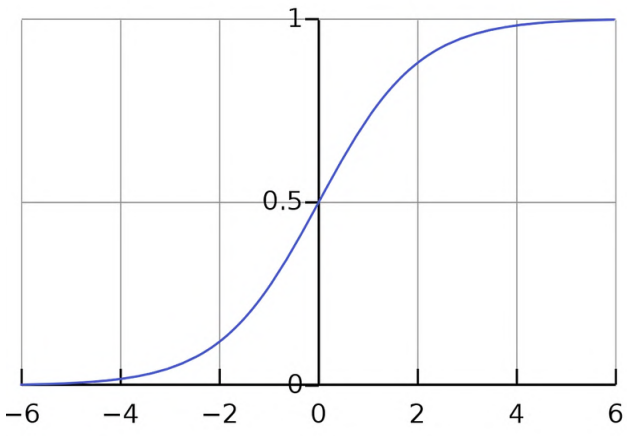

Figura 2.16: Función no lineal sigmoide.
Fuente:https://bootcampai.medium.com/redes-neuronales-13349dd1a5bb

3) **Función Tangente Hiperbólica (tanh)**: Según Calvo, J. (2020) nos indica que esta función es similar a la Sigmoide, la cual produce salidas en escala de [-1,+1]. Es una función continua.

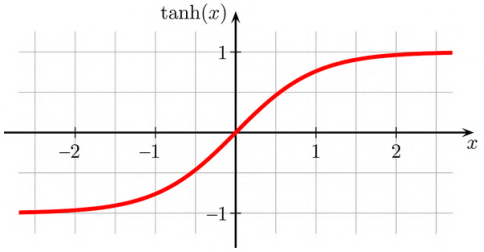

Figura 2.17: Función no lineal tangente.
Fuente:https://bootcampai.medium.com/redes-neuronales-13349dd1a5bb

El cual cumple con la siguiente ecuación:

$$f(x) = \frac{2}{1+e^{-2x}} - 1$$

4) **Función Relu**: Como indica Freire, E. et al (2019) esta función es la más usada ya que permite el aprendizaje muy rápido. Si sus entradas son menores a 0 entonces la salida es 0 y si es mayor su salida es el mismo valor de la entrada. Como se indican en las siguientes ecuaciones:

$$relu(x) = max(0, x)$$
$$relu'(x) = 1.(x > 0)$$

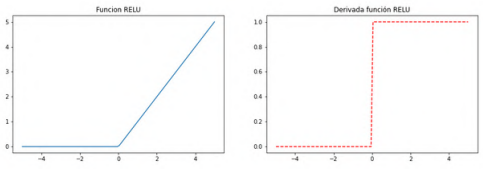

Figura 2.18: Función no lineal Relu.
Fuente:https://www.europeanvalley.es/noticias/crear-red-neuronal-desde-las-matematicas/

5) **Función Softmax** Esta función se utiliza principalmente en problemas de clasificación multiclase, donde el objetivo es asignar a cada instancia una probabilidad de pertenecer a cada una de las clases posibles. A diferencia de funciones como *Sigmoide* o *Tanh*, Softmax normaliza las salidas en un rango $[0, 1]$, garantizando que la suma de todas las probabilidades sea igual a 1. Esto permite interpretar sus resultados directamente como distribuciones de probabilidad.
Según Freire, E. et al. (2019), la función se define matemáticamente como:

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

donde:

- $\mathbf{z} = (z_1, z_2, \ldots, z_K)$ es el vector de entradas (logits) para cada clase.
- z_i es la entrada correspondiente a la clase i-ésima.
- K es el número total de clases.
- $\sigma(\mathbf{z})_i$ representa la probabilidad asignada a la clase i.

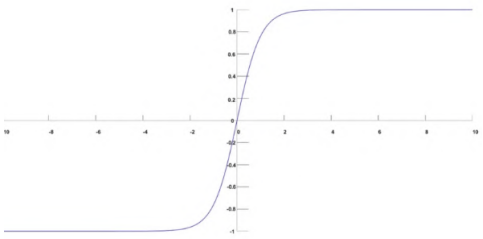

Figura 2.19: Función no lineal softmax.
Fuente: https://bootcampai.medium.com/redes-neuronales-13349dd1a5bb

c) **Función de salida (output function)**: Para el autor Matich, D. (2001):

El último elemento esencial para el funcionamiento de una neurona es la función de salida. Esta función es crucial, ya que establece el valor que se transmitirá a las neuronas conectadas. En caso de que la función de activación no alcance un umbral específico, no se enviará ninguna

señal a la neurona siguiente. Generalmente, no se permite cualquier valor como entrada para una neurona; por lo tanto, los valores de salida se restringen a un rango que puede ser [0, 1] o [-1, 1]. Además, estas salidas pueden adoptar formas binarias, como {0, 1} o {-1, 1}.

Según Matich (2001), la función de salida de una neurona binaria puede definirse como:

$$f(u) = \begin{cases} 1, & u \geq \theta, \\ 0, & u < \theta, \end{cases} \tag{2.5}$$

donde:

- u es la suma ponderada de entradas más el sesgo: $u = \sum_i w_i x_i + b$.
- θ es el umbral de activación.

2.7.2. Tipos de redes neuronales según su topología

Según Ordoñez, J. (2008). Las redes neuronales son clasificados en:

a) **Redes neuronales monocapa** Corresponde a la red neuronal más sencilla consta de una sola capa. Según imagen:

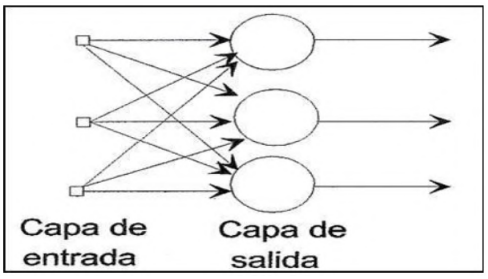

Figura 2.20: Red neuronal monocapa
Fuente: http://grupo.us.es/gtocoma/pid/pid10/RedesNeuronales.htm

b) **Redes neuronales multicapa** Corresponde a una red neuronal que consta de un conjunto de capas intermedias entre la cada de entrada y salida. Ver la siguiente imagen:

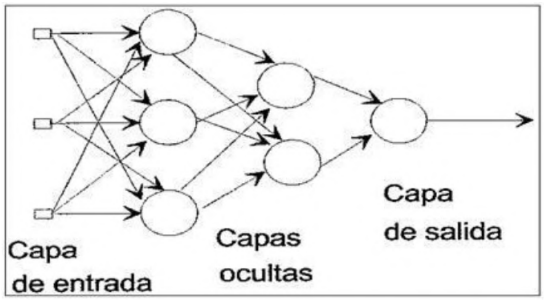

Figura 2.21: Red neuronal multicapa
Fuente: http://grupo.us.es/gtocoma/pid/pid10/RedesNeuronales.html

c) **Redes neuronales recurrente** Este tipo de red se distingue de las anteriores por la presencia de lazos de retroalimentación. Dichos lazos pueden involucrar neuronas de distintas capas, neuronas dentro de la misma capa o incluso conexiones dentro de una única neurona. Esta configuración la convierte en una herramienta particularmente idónea para analizar la dinámica de sistemas no lineales.

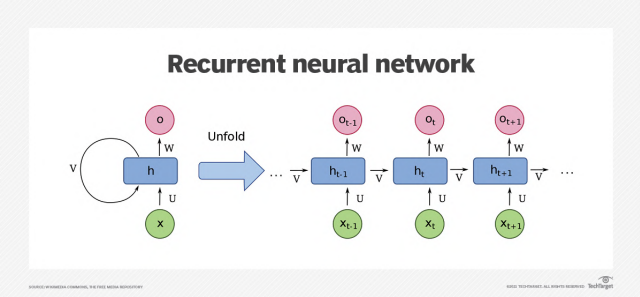

Figura 2.22: Red neuronal recurrente
Fuente: https://es.linkedin.com/pulse/redes-neuronales-recurrentes-rnn-en-el-procesamiento-de-david-d

2.8. Redes Convolucionales

De acuerdo con Durán, J. (2017), las redes neuronales convolucionales operan de manera análoga a las neuronas del sistema biológico, específicamente en el córtex visual, donde las neuronas poseen campos receptivos que reaccionan a estímulos en áreas específicas. En este contexto, cada neurona oculta se conecta únicamente a un pequeño subconjunto de elementos de la imagen. La operación de convolución implica realizar productos y sumas entre la imagen de entrada y un filtro (kernel), generando así un mapa de características. Una de las principales ventajas de las redes convolucionales radica en su capacidad para extraer información de la misma región de la imagen, lo que permite identificar características estacionarias y redu-

ce tanto el número de conexiones como la cantidad de parámetros que deben ser entrenados, en comparación con las redes multicapa completamente conectadas.

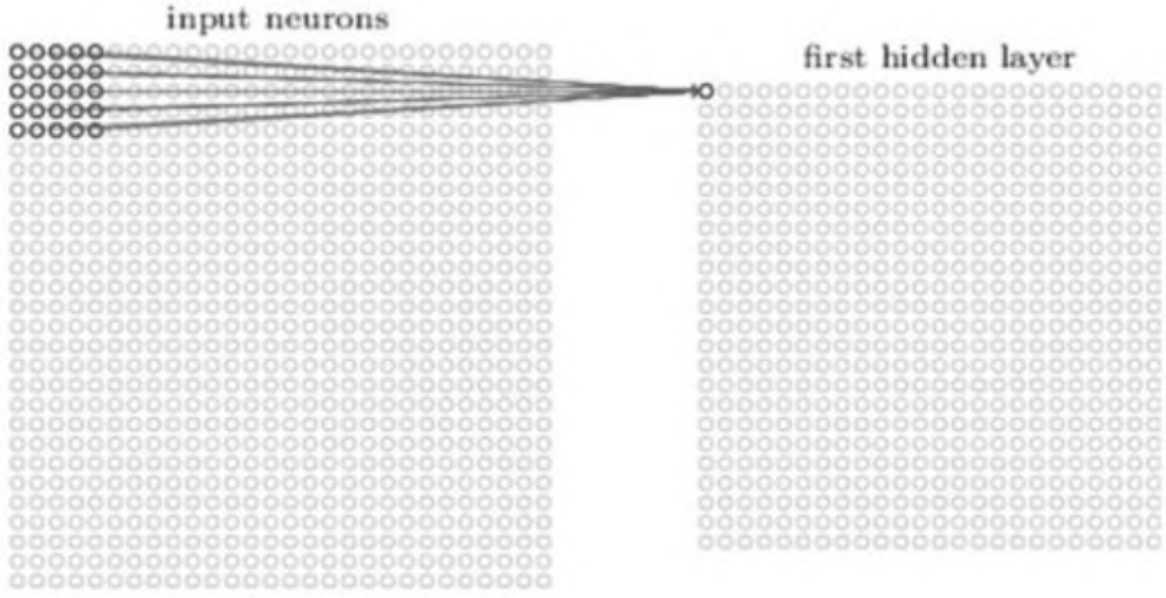

Figura 2.23: Funcionamiento de una red neuronal convolucional
Fuente:https://biblus.us.es/bibing/proyectos/abreproy/91338/descargar_fichero/TFG+Jaime+Dur%C3%A1n+Su%C3%A1rez.pdf

2.9. Autoencoders

Los Autoencoders son un tipo específico de red neuronal diseñada para aprender representaciones comprimidas de los datos de entrada. Su estructura típica consiste en dos componentes: un codificador (encoder) que comprime la entrada en una representación de menor dimensión y un decodificador (decoder) que intenta reconstruir la entrada a partir de esta representación comprimida (Goodfellow, Bengio, y Courville, 2016).

Este tipo de red neuronal no requiere etiquetas en sus datos, por lo que se considera un método de aprendizaje no supervisado. Los Autoencoders se utilizan ampliamente en aplicaciones de reducción de dimensionalidad, eliminación de ruido y generación de datos, donde la compresión y reconstrucción de datos son necesarias (L. Zhang y Yang, 2018).

2.9.1. Arquitectura y funcionamiento de los autoencoders

Estan compuestos por dos partes principales:

- **Encoder (Codificador):** El codificador es la parte de la red que transforma los datos de entrada en una representación de baja dimensión, conocida como espacio latente. Este proceso de compresión permite que el Autoencoder extraiga características relevantes y reduzca la cantidad de información redundante (Vincent, Larochelle, Bengio, y Manzagol, 2008).
- **Decoder (Decodificador):** El decodificador toma la representación comprimida en el espacio latente y la expande para intentar reconstruir los datos originales. El objetivo es que la salida reconstruida sea lo más parecida posible a la entrada (Kingma y Welling, 2014).

La función de pérdida de los Autoencoders se basa en el error de reconstrucción, que mide la diferencia entre la entrada y la salida reconstruida. Una función de pérdida comúnmente utilizada es el Error Cuadrático Medio (Mean Squared Error, MSE), que penaliza las diferencias entre la entrada original y la reconstrucción.

2.9.2. Variante de autoencoders

A lo largo de los años, los Autoencoders han evolucionado y se han desarrollado diversas variantes que se especializan en tareas específicas:

- **Denoising Autoencoders (DAE):** Los Denoising Autoencoders fueron propuestos por (Vincent y cols., 2008) como una forma de mejorar la capacidad de los Autoencoders para aprender representaciones robustas. La idea básica es añadir ruido a los datos de entrada y luego entrenar el modelo para reconstruir la versión "limpia"del dato original. Esto permite que el Autoencoder aprenda a ignorar el ruido y capture la estructura subyacente de los datos.
- **Variational Autoencoders (VAE):** Los Variational Autoencoders, propuestos por (Kingma y Welling, 2014), introducen un enfoque probabilístico. En lugar de aprender una única representación comprimida, el VAE aprende una distribución de probabilidad que permite generar nuevos datos similares a los datos de entrada. Esto los hace útiles en tareas de generación de datos, como imágenes o secuencias de audio.
- **Sparse Autoencoders:** Los Sparse Autoencoders aplican una penalización en la representación latente para inducir una representación esparsa, donde solo unos pocos nodos están activados. Esta técnica es útil para extraer las características más importantes y reducir la redundancia en los datos (Ng, 2011).

2.9.3. Aplicaciones de los Autoencoders

Los Autoencoders tienen una amplia gama de aplicaciones, que van desde la reducción de dimensionalidad hasta la generación de datos. Algunos ejemplos clave incluyen:

- **Reducción de dimensionalidad:** Al igual que el Análisis de Componentes Principales (PCA), los Autoencoders pueden comprimir los datos a una representación de menor dimensión, útil para visualización y análisis de datos de alta dimensión (Hinton y Salakhutdinov, 2006).
- **Eliminación de ruido:** Los Denoising Autoencoders son efectivos para eliminar ruido de los datos, como en la restauración de imágenes corruptas o con ruido (Vincent et al., 2008).
- Generación de datos: Los Variational Autoencoders se utilizan en tareas de generación de contenido, como en la creación de imágenes sintéticas, audio o incluso texto en diversos dominios (Kingma y Welling, 2014).
- **Reconstrucción de datos 3D:** En aplicaciones de reconstrucción en 3D, como nubes de puntos, los Autoencoders pueden aprender a modelar la estructura espacial de los datos, lo que es útil para tareas de reconstrucción detallada de objetos o entornos (Achlioptas, Diamanti, Mitliagkas, y Guibas, 2018).

2.9.4. Relevancia de los Autoencoders en la Inteligencia Artificial

Dentro del campo de la inteligencia artificial, los Autoencoders son una herramienta poderosa para el análisis y la síntesis de datos complejos. Como técnica de aprendizaje profundo, los Autoencoders permiten descubrir patrones significativos en los datos sin la necesidad de etiquetas, lo cual es valioso en aplicaciones de gran escala y en entornos con datos no estructurados (LeCun, Bengio, y Hinton, 2015).

En comparación con otros métodos de reconstrucción, los Autoencoders son especialmente útiles cuando los datos tienen estructuras complejas que requieren un aprendizaje flexible y adaptable. Esto los convierte en una alternativa eficaz para mejorar la precisión de reconstrucción y reducir errores en comparación con métodos tradicionales como el algoritmo de Poisson.

2.10. Visión Artificial

La visión artificial es un área de la Inteligencia Artificial que emplea diversas técnicas para adquirir, procesar y analizar información proveniente de imágenes digitales. Los seres humanos tenemos la capacidad de percibir el entorno en tres dimensiones y de identificar objetos junto con sus características a través de la vista. Por lo tanto, el objetivo primordial de la visión artificial es otorgar a las computadoras la habilidad de interpretar imágenes de manera similar a como lo hace un ser humano. Según De la Cruz D. y Alonso I. (2021), las principales actividades de la visión artificial pueden ser clasificadas en:

- **Procesamiento de imágenes digitales**: Su objetivo principal es analizar y describir el contenido de una imagen digital, facilitando su reconocimiento.
- **Visión computacional**: Busca dotar a las computadoras con la capacidad de imitar la percepción visual humana, como lo mencionan De la Cruz D. y Alonso I. (2021).

Según la Universidad de Cambridge, la visión artificial emplea tres enfoques principales para analizar los datos:

- **Reconocimiento**: Identifica la ubicación de los objetos en una imagen y determina qué son.
- **Reconstrucción**: Se refiere a la técnica que genera una forma tridimensional de los objetos en una imagen.
- **Registro**: Implica alinear modelos, comparando elementos de una imagen con patrones previamente establecidos en busca de coincidencias.

Figura 2.24: Tareas de reconocimiento, reconstrucción y registro sobre nubes de puntos.
Fuente:Universidad de Cambrigde

Parte III

Desarrollo del proyecto

Capítulo 3

Desarrollo del prototipo:

El prototipo desarrollado tiene como objetivo escanear y reconstruir superficies de difícil acceso utilizando un dron equipado con un sensor. El prototipo está diseñado para capturar una nube de puntos en tres dimensiones del área de interés y posteriormente, aplicar algoritmos de procesamiento para obtener una reconstrucción detallada. En este capítulo se detallan los componentes, la metodología de desarrollo y las pruebas realizadas para la validación del prototipo. En la siguiente imagen de describe cada etapa desarrollada en este proyecto de investigación:

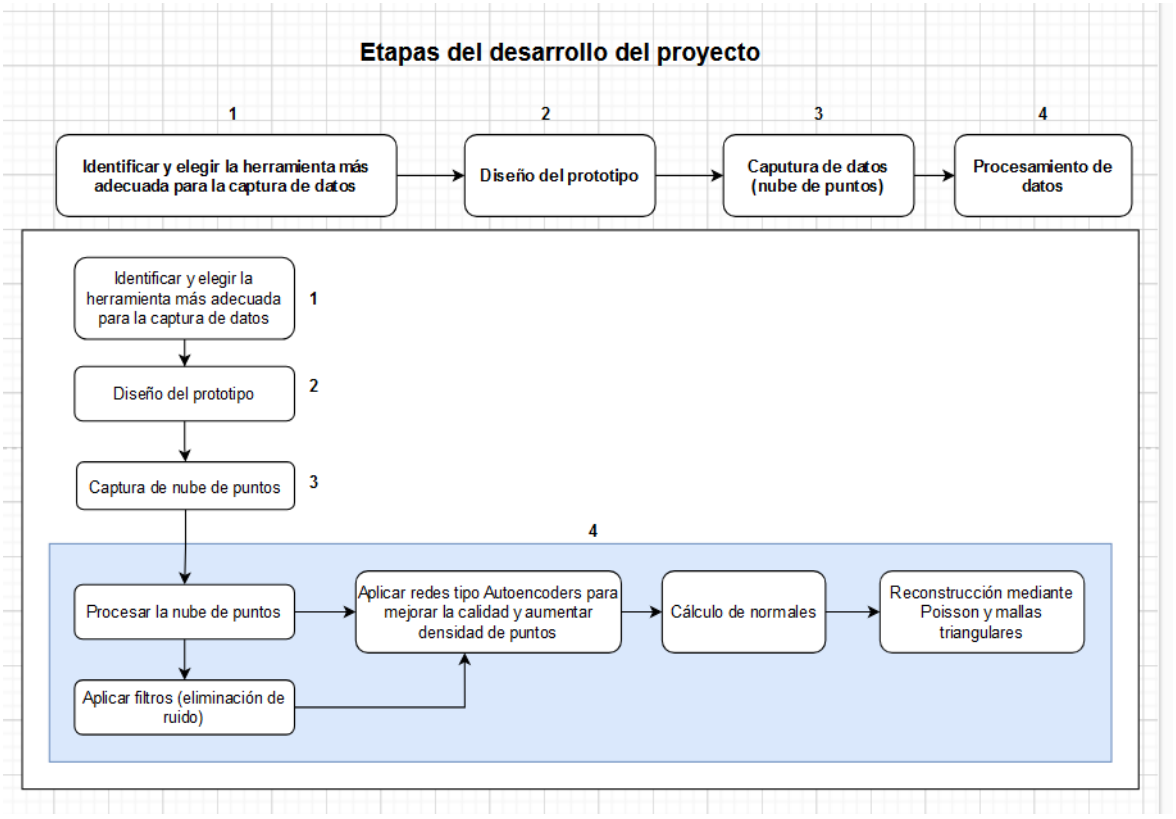

Figura 3.1: Etapas del desarrollo del Proyecto
Fuente:Propia

3.1. Identificación de la herramienta para la captura de nube de puntos

Para este proceso, tuvimos varias herramientas considerando a estos los diferentes tipos de sensores que pudimos ver y entre los sensores LIDAR, se encontre diferentes tipos con capacidades y especificaciones diferentes, para lo cual realizamos una tabla comparativa entre las especificaciones de cada uno, ventajas y desventajas y así de esta forma poder elegir al más adecuado para este prototipo:

Cuadro 3.1: Comparativa entre LIDAR, Sensores de Ultrasonido, Cámaras RGB-D y Estéreo

Tecnología	**Nombre**	**Rango de Detección**	**Precisión**	**Velocidad / Frecuencia**	**Costo Aproximado**	**Dimensión de Captura**	**Ambientes con Baja Luz**	**Peso Aproximado**
LIDAR	LDROBOT D300 Lidar Kit	0.05 - 12 m	±2 mm	12 Hz	$150	2D	Si	50 g
LIDAR	Velodyne VLP-16	0.5 - 100 m	±3 cm	10 - 20 Hz	$4,000 - $8,000	3D	Si	830 g
LIDAR	RPLIDAR A1	0.15 - 12 m	±2%	5 - 10 Hz	$100 - $150	2D	Si	200 g
LIDAR	Hokuyo UTM-30LX	0.1 - 30 m	±30 mm	40 Hz	$3,500 - $4,500	2D	Si	370 g
LIDAR	Ouster OS1-16	0.5 - 120 m	±3 cm	10 - 20 Hz	$6,000 - $10,000	3D	Si	1.3 kg
Ultrasónico	HC-SR04	2 - 400 cm	±3 mm	˜40 Hz	$2 - $10	1D	No	15 g
Cámaras RGB-D	Intel RealSense D455	0.2 - 10 m	±1% prof.	30 - 90 FPS	$250 - $500	3D	No	72 g
Cámaras Estéreo	ZED 2	0.5 - 20 m	±1% prof.	15 - 100 FPS	$449 - $700	3D	No	170 g

Después de todo este análisis, elegimos el LDROBOT D300 lidar Kit, considerando que es uno de los más adecuados para la reconstruccion más precisa en 3D, que tiene un rango de detección desde 0.05 a 12 m, una precisión de +/- 2mm , de un costo no tan eleveado , un peso aproximado 50g que seran necesarios para que el dron pueda lograr una estabilidad al momento del vuelo , siendo asi la mejor opcion entre todos los sensores que se estudiaron .

3.2. Diseño del prototipo

3.2.1. Metodología de Captura de Datos

a) **Lógica del código para la captura de datos:** El sensor LiDAR, junto con el dron y un ESP32, permite la captura de datos en tiempo real. El ESP32 gestiona la comunicación inalámbrica entre el dron y la estación de control, enviando los datos de la nube de puntos a la computadora de procesamiento. El código en Python captura los datos desde el sensor, procesándolos y almacenándolos en la computadora de control en formato .pcd (Point Cloud Data).

b) **Descripcion del proceso:** Durante el vuelo, el dron sigue una ruta predefinida y el sensor LiDAR realiza mediciones tridimensionales continuas del entorno. Los datos capturados son enviados y procesados en tiempo real, generando una nube de puntos del área escaneada.

c) **Formato de datos** Los datos capturados se almacenan en formato .pcd o .ply para nubes de puntos y en formato ROSbag para la sincronización y almacenamiento de todos los datos del vuelo. El formato .pcd o .ply permite un fácil acceso y manipulación de las nubes de puntos en el procesamiento posterior.

En este sistema, se utiliza un microcontrolador ESP32 equipado con un sensor LiDAR para capturar datos. Estos datos son transmitidos mediante el protocolo UDP (User Datagram Protocol) a una estación terrestre, la cual ejecuta un script en Python para recibir, procesar y almacenar los datos recibidos.

La configuración del ESP32 permite la transmisión de los datos LiDAR en tiempo real a través de una red Wi-Fi, utilizando la librería `WiFiUDP` para el manejo de la comunicación en el protocolo UDP. El código se encarga de leer los datos del sensor a través de los pines de comunicación serial (UART) y enviar los paquetes al servidor en la estación terrestre.

3.2.2. Algoritmo implementado en el Esp32

El algoritmo se encarga de recibir los datos empaquetados enviados por el sensor LIDAR a la estación de trabajo mediante el protocolo UDP, estos paquetes constan de cadenas de 94 caracteres en hexadecimal, el cual se debe identificar cada inicio y fin de la cadena para así poder procesarla en la estación de trabajo. Para la comunicación serie requerimos de:

$$\begin{aligned} \text{Datos leídos:} \quad & D = \{d_1, d_2, \ldots, d_n\} \\ \text{Buffer temporal:} \quad & B = \{b_1, b_2, \ldots, b_n\}, \quad n = \text{número de bytes disponibles} \\ \text{Conversión a hexadecimal:} \quad & H = \{h_1, h_2, \ldots, h_n\}, \quad h_i = \text{Hex}(d_i) \end{aligned}$$

Donde D: son los bytes que son enviados por el sensor y son leidos por el esp32 mediante el puerto serial, B representa un buffer temporal de bytes que almacena los datos por leer temporalmente y H es la cadena final en caracteres hexadecimales enviados a la estación de trabajo.

Para la construcción de cadena y envío UDP

$$\begin{aligned} \text{Cadena construida:} \quad & C = \text{Concatenación}(\{h_1, h_2, \ldots, h_k\}), \quad k = \text{datos seleccionados} \\ \text{Transmisión:} \quad & U = \text{UDP.println}(C, \text{IP}, \text{Puerto}) \end{aligned}$$

Donde C: es la concatenacion de todos caracteres convertidos en hexadecimal, donde k es el tamaño de la cadena, UDP.println seria la línea de comando para el envío o la transmisión de datos, la IP vendría ser la ip de la estación de trabajo y la comunicación esta dada mediante el puerto: 4210.

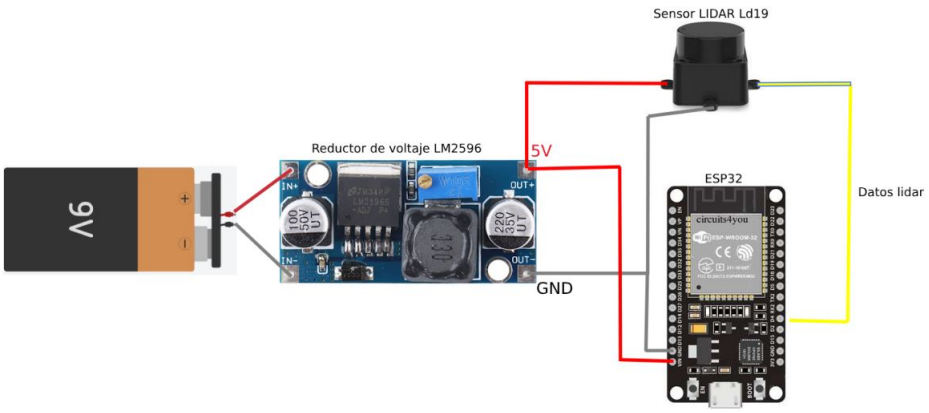

Figura 3.2: Diagrama de conexión LIDAR ESP32 Convertidor de voltajes
Fuente:Propia

Conexión WiFi

$$S(t) = \begin{cases} S_1, & \text{si WiFi.status()} \neq \text{WL\_CONNECTED} \\ S_2, & \text{si WiFi.status()} = \text{WL\_CONNECTED} \end{cases}$$

Explicación del proceso

- **Inicialización de la red y el protocolo UDP:** El ESP32 se conecta a la red Wi-Fi especificada (`ssid` y `password`) y configura un puerto UDP para transmitir datos a la estación de trabajo.
- **Captura de datos desde el sensor LiDAR:** Se leen los datos del sensor LiDAR a través del puerto serial configurado en los pines RXD1 y TXD1, convirtiéndolos a formato hexadecimal.
- **Transmisión de datos:** Cuando se recibe un paquete de datos LiDAR completo (94 caracteres), estos son enviados al servidor usando el protocolo UDP.

Recepción y almacenamiento de la nube de puntos

En el servidor, un script en `Python` recibe los datos enviados por el ESP32 y los almacena para su posterior procesamiento.

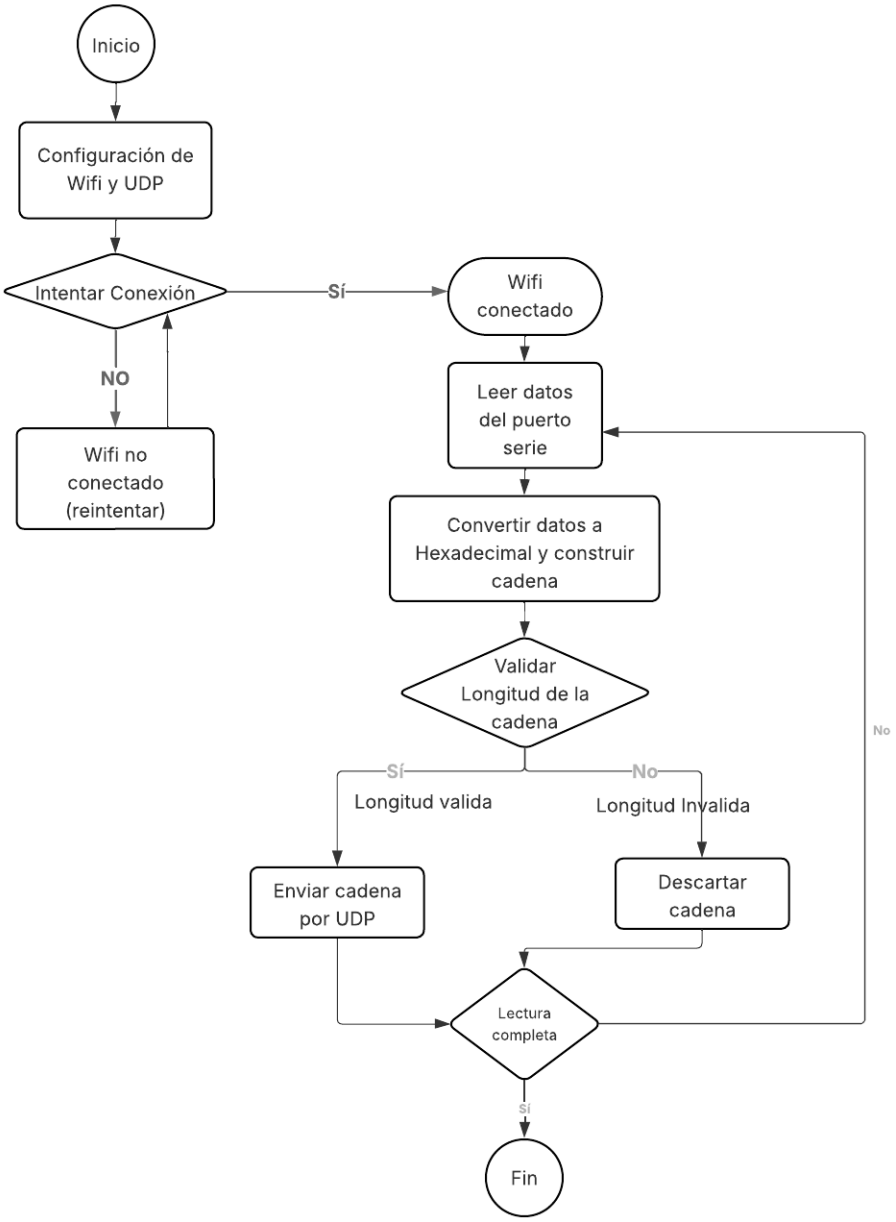

Figura 3.3: Flujograma de captura y envio de datos
Fuente:Propia

3.3. Proceso de reconstrucción 3D a partir de la nube de puntos

3.3.1. Eliminación de valores atípicos estadísticos en nubes de puntos 3D

El proceso de eliminación de valores atípicos estadísticos se basa en el cálculo de la distancia de cada punto en la nube de puntos a sus vecinos más cercanos, utilizando

la métrica de distancia euclidiana en el espacio 3D. El algoritmo sigue los siguientes pasos:

a) **Cálculo de distancias:** Para cada punto de la nube, se calcula la distancia a sus vecinos más cercanos. Esto se realiza utilizando la distancia euclidiana.

b) **Promedio y desviación estándar:** Para cada punto, se calcula el promedio (μ) de las distancias a sus vecinos más cercanos y la desviación estándar (σ) de esas distancias.

c) **Criterio para identificar outliers:** Si la distancia de un punto a sus vecinos está por encima de un umbral definido por la fórmula:

$$\text{distancia al punto} > \mu + \sigma \cdot \text{std\_ratio}$$

donde:

- μ es el promedio de las distancias a los vecinos,
- σ es la desviación estándar de esas distancias,
- y `std_ratio` es un factor ajustable que determina cuán lejos debe estar un punto para ser considerado un outlier.

d) **Filtrado de outliers:** Los puntos cuya distancia supera este umbral son considerados atípicos y se eliminan de la nube de puntos.

Este proceso se aplica a las nubes de puntos generadas por sensores LiDAR u otros dispositivos 3D, ayudando a mejorar la calidad de los datos antes de realizar análisis como la reconstrucción 3D o el análisis geométrico.

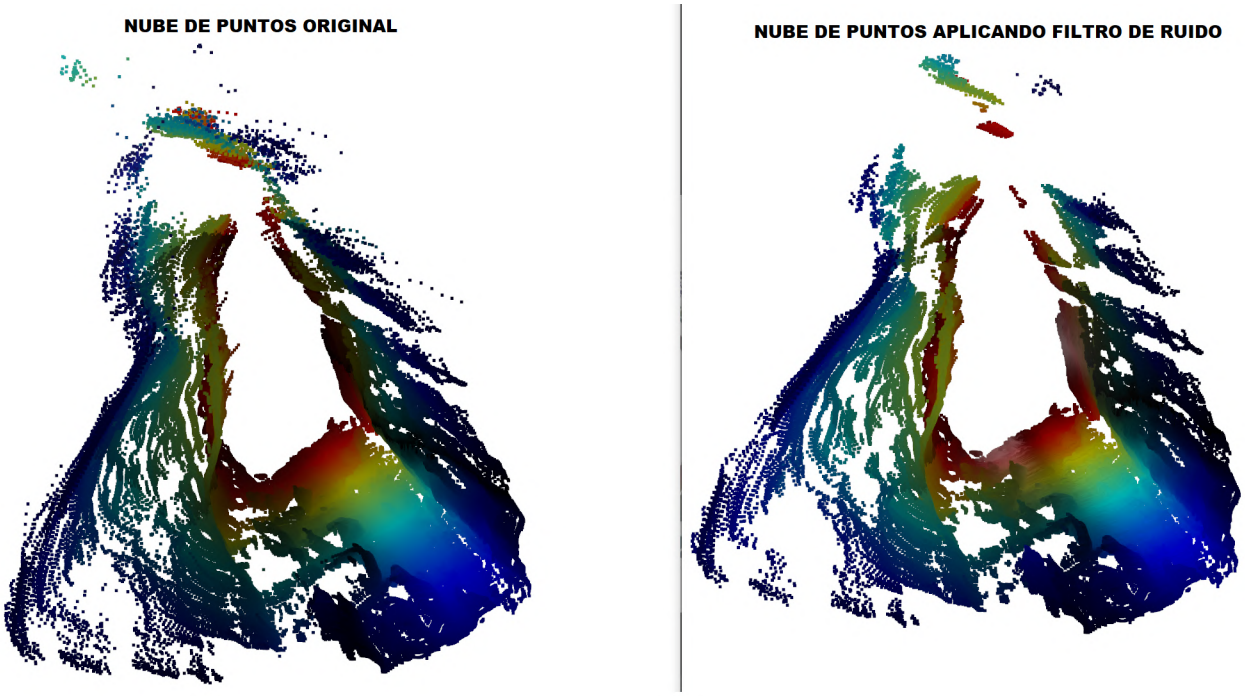

Figura 3.4: Filtrado de ruido mediante outliers.
Fuente:Propia

3.4. Uso de redes neuronales

El presente capítulo describe el proceso de implementación de un modelo basado en redes neuronales tipo Autoencoder para la reconstrucción y aumento de densidad de nubes de puntos 3D, a partir de datos recolectados por el prototipo . El enfoque busca superar las limitaciones de métodos tradicionales, como el algoritmo de reconstrucción por Poisson, mediante la utilización de aprendizaje profundo.

3.4.1. Introducción a los autoencoders

Un Autoencoder es una red neuronal diseñada para aprender una representación compacta (*latent space*) de los datos de entrada y luego reconstruirlos con alta fidelidad. Se compone de dos partes principales:

- **Encoder**: Transforma los datos originales en una representación de baja dimensión que captura las características esenciales del conjunto de datos.
- **Decoder**: Reconstruye los datos originales a partir de la representación latente generada por el Encoder, con la posibilidad de generar puntos adicionales para aumentar la densidad.

En este proyecto, el Autoencoder se entrena para trabajar con nubes de puntos 3D, mejorando su reconstrucción y aumentando su densidad para obtener una representación más detallada y precisa del espacio original.

3.4.2. Métrica de evaluación: Chamfer Distance

La métrica de *Chamfer Distance* mide la similitud entre dos nubes de puntos, evaluando la distancia promedio entre los puntos más cercanos de cada nube. Es utilizada como función de pérdida para guiar el entrenamiento del modelo, minimizando las discrepancias entre la reconstrucción y los datos originales.

La **Chamfer Distance (CD)** es una métrica comúnmente utilizada para evaluar la diferencia entre dos nubes de puntos en tareas como reconstrucción 3D. En el caso de este trabajo, se utiliza para medir el error entre la nube de puntos reconstruida por la red de autoencoders y la nube de puntos original.

3.4.3. Modelo Matemático

Dada una nube de puntos X y una nube de puntos Y, la **Chamfer Distance** se define como:

$$d_{\text{CD}}(X, Y) = \sum_{x \in X} \min_{y \in Y} \|x - y\|_2^2 + \sum_{y \in Y} \min_{x \in X} \|x - y\|_2^2$$

Donde:

- $X = \{x_1, x_2, ..., x_n\}$ es la nube de puntos reconstruida (predicha).
- $Y = \{y_1, y_2, ..., y_m\}$ es la nube de puntos original (ground truth).
- $\|x - y\|_2$ representa la **distancia euclidiana** entre dos puntos x y y.
- El operador mín selecciona el punto más cercano en la otra nube.

3.4.4. Explicación Conceptual

- **Búsqueda del punto más cercano**: Para cada punto en la nube de puntos reconstruida (X), se encuentra el punto más cercano en la nube de puntos original (Y). Esto se repite de manera simétrica para cada punto de Y hacia X.
- **Suma de distancias**: La **Chamfer Distance** suma las distancias cuadradas entre los puntos y sus respectivas correspondencias más cercanas. Al hacerlo en ambas direcciones, se asegura una medida simétrica entre las dos nubes.

- **Interpretación**: Un valor pequeño de Chamfer Distance indica que las dos nubes de puntos son similares, mientras que un valor grande sugiere que la reconstrucción tiene un mayor error respecto a la nube de puntos original.

3.4.5. Arquitectura del autoencoder

El Autoencoder es una red neuronal de aprendizaje no supervisado formada por dos componentes simétricos:

- **Encoder**, que comprime la información de entrada en un espacio latente de baja dimensión
- **Decoder**, que expande dicha representación para reconstruir (y en nuestro caso, aumentar) la nube de puntos original.

Esta arquitectura fue seleccionada frente a otras alternativas (por ejemplo, redes convolucionales 3D) por los siguientes motivos:

- **Ausencia de dataset etiquetado**: En los entornos arqueológicos y naturales de Cusco no existe una base de datos volumétrica o voxelizada de todos los posibles escenarios.

  Entrenar un 3D-CNN requeriría cientos o miles de ejemplos etiquetados, lo cual es inviable dada la complejidad y variabilidad de cada sitio.
- **Aprendizaje no supervisado y generalización**, Con un autoencoder podemos usar cualquier colección de nubes de puntos (incluso parciales) sin necesidad de etiquetas.

  El "cuello de botella" (bottleneck) fuerza a la red a aprender rasgos geométricos esenciales (curvaturas, densidades), lo cual promueve su capacidad para reconstruir y rellenar huecos en escenarios inéditos.
- **Aumento de densidad**,El decoder no solo reconstruye, sino que "interpola" puntos adicionales en zonas con baja densidad, mejorando la fidelidad de la nube 3D resultante.

El Autoencoder es una red neuronal que consta de dos partes principales: **el Encoder** y **el Decoder**. A continuación, se explica cómo funciona cada componente:

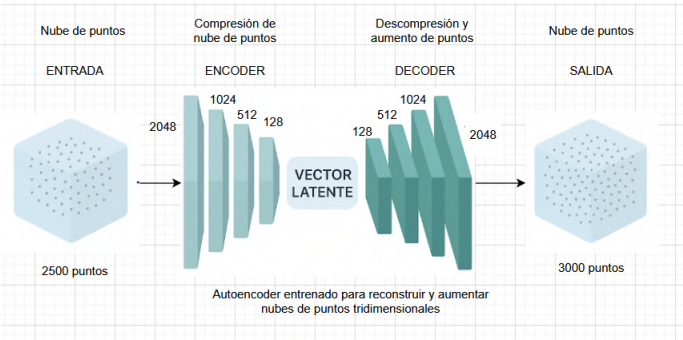

Figura 3.5: Diagrama de la red neuronal
Fuente:Propia

3.4.6. Encoder

El **Encoder** reduce la dimensionalidad de la nube de puntos 3D. Utiliza capas densas para mapear los puntos 3D a un espacio latente comprimido. Cada capa aplica la activación `LeakyReLU`, lo que ayuda a mitigar el problema de los gradientes nulos y mejora la capacidad de la red para aprender representaciones útiles en el espacio comprimido. El código de la implementación del encoder es el siguiente:

3.4.7. Decoder

El **Decoder** toma la representación latente comprimida por el Encoder y la reconstruye en la nube de puntos original. Al igual que el Encoder, el Decoder consta de varias capas densas, todas con activaciones `LeakyReLU`, lo que permite una reconstrucción más precisa. A continuación, se presenta el código del Decoder:

3.4.8. Inicialización de Pesos

El entrenamiento de la red se estabiliza utilizando una inicialización adecuada de los pesos. En este caso, se utiliza la inicialización `Xavier Uniform`, que es una técnica comúnmente utilizada en redes profundas. Esta técnica asegura una distribución uniforme adecuada de los pesos, lo que ayuda a prevenir problemas como el desvanecimiento o la explosión de gradientes. Esto permite que la red comience el proceso de entrenamiento con un buen punto de partida para aprender representaciones significativas. El código para la inicialización de pesos es el siguiente:

Descripción de `nn.Linear` en PyTorch

En redes neuronales, las capas densas o totalmente conectadas implementadas con la clase `nn.Linear` en PyTorch aplican una transformación lineal de la forma:

$$y = xW^T + b$$

donde:

- x: es el vector de entrada (o matriz, si se procesan múltiples vectores en paralelo).
- W: es la matriz de pesos que contiene los parámetros aprendidos por la capa.
- W^T: es la **transposición** de la matriz W, necesaria para garantizar la compatibilidad dimensional en la multiplicación.
- b: es el vector de sesgo (*bias*), también aprendido durante el entrenamiento.
- y: es el vector o matriz resultante, que corresponde a la salida de la capa tras aplicar la transformación.

Razón para utilizar W^T: En capas densas, la multiplicación de matrices entre la entrada x y los pesos W debe respetar las reglas de compatibilidad dimensional. Por ejemplo:

- Si x es un vector de tamaño $1 \times$ `in_features`,
- y W tiene dimensiones `out_features` $\times$ `in_features`,

entonces se utiliza W^T (dimensiones `in_features` $\times$ `out_features`) para que la multiplicación matricial xW^T sea válida. El resultado será un vector y con dimensiones $1 \times$ `out_features`.

3.4.9. Flujo Completo de la Red

La arquitectura completa del autoencoder sigue un flujo de entrada, compresión y reconstrucción de los datos. La siguiente tabla nos da a aconocer la arquitectura completa diseñada:

Cuadro 3.3: Arquitectura del Autoencoder

Fase	**Capa**	**Dimensión (entrada → salida)**	**Activación**
Encoder	Capa 1	2048 → 1024	LeakyReLU
	Capa 2	1024 → 512	LeakyReLU
	Capa 3	512 → 128	LeakyReLU
	Latent space	128	—
Decoder	Capa 1	128 → 512	LeakyReLU
	Capa 2	512 → 1024	LeakyReLU
	Capa 3	1024 → 2048	LeakyReLU
	Salida	2048 → $(x,y,z)^*$	Lineal

La descripción de la tabla anterior se detalla en el siguiente texto:

- **Entrada**: Se toma la nube de puntos 3D original.
- **Compresión (Encoder)**: La nube de puntos es procesada por el Encoder, que reduce la dimensionalidad a un espacio latente comprimido.
  - **Primera capa densa** (`nn.Linear(input_dim, 2048)`):
    - **Input:** La entrada del encoder es una nube de puntos 3D, donde `input_dim` es el número de características de cada punto.

    La primera capa tiene 2048 unidades. Esto amplía la capacidad de la red para capturar relaciones no lineales entre los puntos. La función de activación es LeakyReLU con un `negative_slope=0.2`. Esta activación ayuda a evitar el problema de "vanishing gradient"que puede surgir con ReLU tradicional, permitiendo que la red aprenda de manera más eficiente. La activación LeakyReLU permite que pasen valores negativos pequeños, lo que mantiene las conexiones activas durante el entrenamiento.
  - **Segunda capa densa** (`nn.Linear(2048, 1024)`): La salida de la primera capa se pasa a la segunda capa con 1024 unidades, reduciendo la dimensionalidad gradualmente. La función de activación sigue siendo LeakyReLU con un `negative_slope=0.2`.
  - **Tercera capa densa** (`nn.Linear(1024, 512)`): Esta capa reduce aún más las dimensiones y continúa el proceso de abstracción de la información relevante de la nube de puntos.
  - **Capa final (Latent Space)** (`nn.Linear(512, latent_dim)`): Finalmente, la red codifica la entrada en un espacio de características latente (`latent_dim`). Este es el tamaño reducido de la representación comprimida, que es mucho más pequeña que la entrada original. Este espacio latente es donde se almacenan las características más importantes de la nube de puntos.
- **Decoder**: El Decoder toma la representación latente del encoder y la reconstruye de nuevo en el espacio original de la nube de puntos. El Decoder consta de una secuencia de capas densas que aumentan progresivamente la dimensionalidad hasta llegar de nuevo a `input_dim`, la dimensión original de la nube de puntos.

- **Primera capa densa** (`nn.Linear(latent_dim, 512)`):
  - **Input:** La representación comprimida en el espacio latente (`latent_dim`).
  - **Salida:** La red comienza a expandir la representación comprimida, primero a 512 dimensiones. La activación es LeakyReLU con `negative_slope=0.2`.

- **Segunda capa densa** (`nn.Linear(512, 1024)`): La salida de la primera capa se pasa a la segunda capa con 1024 unidades, aumentando gradualmente las dimensiones. La función de activación sigue siendo LeakyReLU con un `negative_slope=0.2`.
- **Tercera capa densa** (`nn.Linear(1024, 2048)`): El decoder sigue aumentando las dimensiones hasta alcanzar 2048, acercándose cada vez más a la entrada original.
- **Capa final** (`nn.Linear(2048, input_dim)`): Finalmente, la salida es una reconstrucción de la nube de puntos original con las mismas dimensiones que la entrada. Esto es lo que se espera que sea la salida del autoencoder: una reconstrucción lo más parecida posible a los puntos 3D iniciales. La activación final puede ser una función lineal si los valores de salida deben estar en un rango continuo (como las coordenadas 3D).

3.4.10. Entrenamiento del autoencoder

El modelo se entrena minimizando la *Chamfer Distance*, utilizando el optimizador Adam con un programador de tasas de aprendizaje (*StepLR*). Durante el entrenamiento, se monitorea la pérdida para evaluar el progreso y ajustar el modelo según sea necesario.

3.4.11. Aumento de densidad

El modelo entrenado se utiliza para generar nubes de puntos más densas. Esto se logra mediante la generación de puntos adicionales a partir de la representación latente, preservando las características originales del objeto.

3.5. Métricas de evaluación

Una vez definida la arquitectura del autoencoder, es necesario establecer las métricas que cuantifican la calidad de la reconstrucción y el aumento de densidad en las nubes de puntos 3D. A continuación se presentan las principales:

3.5.1. Chamfer Distance

Definición

$$d_C(X,Y) \;=\; \frac{1}{|X|}\sum_{x\in X} \min_{y\in Y} \|x-y\|^2 \;+\; \frac{1}{|Y|}\sum_{y\in Y} \min_{x\in X} \|y-x\|^2$$

Interpretación Mide la distancia promedio de cada punto de una nube al conjunto opuesto, penalizando tanto huecos como solapamientos en ambas direcciones.

Justificación

- **Diferenciable:** Permite incorporar la métrica directamente en la función de pérdida durante el entrenamiento.
- **Resistente a outliers moderados:** Un único punto muy desviado no domina la medida, a diferencia de la Hausdorff Distance.
- **Eficiente en GPU:** Puede calcularse en paralelo para lotes grandes de nubes de puntos.

3.5.2. Error cuadrático medio (MSE)

Definición

$$\mathrm{MSE}(X,Y) \;=\; \frac{1}{N}\sum_{i=1}^{N} \|x_i - y_i\|^2$$

donde se asume correspondencia uno a uno entre puntos de ambas nubes.

Interpretación Cuantifica el error medio de posición entre puntos emparejados. Es sencillo de calcular cuando la correspondencia está garantizada.

Limitaciones

- Requiere que las nubes de puntos estén ordenadas o alineadas punto a punto.
- Muy sensible a emparejamientos incorrectos o reordenamientos de los puntos.

3.5.3. Otras métricas

- **Hausdorff Distance:** Define la máxima distancia punto–conjunto. Es útil para garantizar que ninguna parte de la nube reconstruida esté muy alejada, pero puede verse sesgada por valores extremos.

- **Earth Mover's Distance (EMD):** Calcula el mínimo "coste" para transformar una nube en otra, considerando la distribución de masa. Proporciona una correspondencia más rica, pero su cómputo es significativamente más costoso.

En la siguiente table se detalla las diferencias principales entre estas métricas:

Cuadro 3.4: Comparación de métricas de evaluación

Métrica	**Definición**	**Qué mide**	**Ventajas**	**Limitaciones**
Chamfer Distance	$d_C(X,Y) = \frac{1}{\lvert X\rvert}\sum_{x\in X}\min_{y\in Y}\lVert x-y\rVert^2 + \frac{1}{\lvert Y\rvert}\sum_{y\in Y}\min_{x\in X}\lVert y-x\rVert^2$	Distancia promedio bidireccional punto→conjunto.	Diferenciable; resistente a outliers moderados; eficiente en GPU.	Puede suavizar detalles muy locales.
Mean Squared Error (MSE)	$\mathrm{MSE}(X,Y) = \frac{1}{N}\sum_{i=1}^{N}\lVert x_i-y_i\rVert^2$	Error medio punto–punto (requiere correspondencia).	Fácil de implementar; optimización directa.	Sensibilidad a desequilibrios y reordenamientos.
Hausdorff Distance	$d_H(X,Y) = \max\{\sup_{x\in X}\inf_{y\in Y}\lVert x-y\rVert, \sup_{y\in Y}\inf_{x\in X}\lVert y-x\rVert\}$	Máxima discrepancia punto→conjunto.	Detecta el peor caso local; útil en control de calidad.	Extremadamente sensible a outliers.
Earth Mover's Distance (EMD)	*Coste mínimo de transporte entre distribuciones.*	Divergencia de distribuciones completas.	Modela correspondencias globales de forma óptima.	Cómputo intensivo; escasa escalabilidad.

3.5.4. Selección de la métrica principal

Para nuestro caso —reconstrucción y densificación de nubes 3D irregulares y sin correspondencia exacta— la **Chamfer Distance** se adopta como métrica de referencia por su equilibrio entre diferenciabilidad, robustez a outliers y eficiencia computacional. En el Capítulo 4 se presentan los resultados comparativos que confirman su superior correlación con la calidad visual y geométrica de las reconstrucciones.

3.6. Cálculo de las Normales

3.6.1. Cálculo del vector de referencia

El cálculo de las normales en una nube de puntos es esencial en diversas aplicaciones de la geometría computacional, análisis de superficies, y simulación de iluminación. Una **normal** es un vector perpendicular a la superficie en cada punto, que describe la orientación de la superficie en ese punto específico.

Este proceso se basa en los métodos presentados por diversos investigadores en el campo de la reconstrucción 3D y la geometría computacional, como el trabajo realizado por *R. Zhang* y *M. Kazhdan* en el cálculo de normales en nubes de puntos.

1. Lectura de la Nube de Puntos

La nube de puntos está formada por un conjunto de puntos $P = \{p_1, p_2, \ldots, p_n\}$ en el espacio tridimensional. Cada punto p_i tiene coordenadas (x_i, y_i, z_i) en R^3, lo que describe su posición en el espacio 3D. Estos puntos se leen generalmente desde un archivo en formato PLY, un formato comúnmente utilizado en la representación de nubes de puntos.

2. Definición de Puntos de Inicio y Final

Para el cálculo de la normal, se seleccionan dos puntos de referencia P_{inicio} y P_{final}. Estos puntos definen una línea de recorrido en el espacio tridimensional que será utilizada para el cálculo de las normales:

$$P_{\text{inicio}} = (x_{\text{inicio}}, y_{\text{inicio}}, z_{\text{inicio}}) = (0, 0, 0)$$

$$P_{\text{final}} = (x_{\text{final}}, y_{\text{final}}, z_{\text{final}}) = (0, 0, z_{\max})$$

Estos puntos de referencia son utilizados para definir la dirección de la línea de recorrido sobre la que se proyectan las normales de la nube de puntos.

3. Cálculo del Vector de Recorrido

El vector de recorrido $\mathbf{v}$ conecta los puntos de inicio y final de la línea de recorrido. Se calcula como la diferencia entre las coordenadas de estos dos puntos:

$$\mathbf{v} = P_{\text{final}} - P_{\text{inicio}} = (x_{\text{final}} - x_{\text{inicio}}, y_{\text{final}} - y_{\text{inicio}}, z_{\text{final}} - z_{\text{inicio}})$$

Este vector se utiliza para definir la dirección del recorrido. Para normalizar el vector $\mathbf{v}$, se divide entre su magnitud $||\mathbf{v}||$:

$$\hat{\mathbf{v}} = \frac{\mathbf{v}}{||\mathbf{v}||}$$

donde la norma $||\mathbf{v}||$ es dada por la raíz cuadrada de la suma de los cuadrados de sus componentes:

$$||\mathbf{v}|| = \sqrt{v_x^2 + v_y^2 + v_z^2}$$

4. Conversión de la Nube de Puntos a un Array de Coordenadas

Una nube de puntos tridimensional se puede representar en términos de sus coordenadas cartesianas (x_i, y_i, z_i), asociadas a cada punto p_i. Formalmente, cada punto se expresa como un vector:

$$\mathbf{p}_i = (x_i, y_i, z_i)$$

El conjunto completo de puntos se organiza en una estructura matricial o un array de coordenadas, facilitando su procesamiento computacional mediante operaciones vectoriales y matriciales.

Para calcular una **normal** en cada punto p_i, se sigue el siguiente procedimiento:

a) **Definición del vector relativo al inicio:** Se define el vector $\mathbf{v}_i$ que conecta el punto de referencia inicial P_{inicio} con el punto p_i. Este vector se obtiene como:

$$\mathbf{v}_i = \mathbf{p}_i - P_{\text{inicio}}$$

donde:

- $\mathbf{p}_i$ es el vector del punto actual.
- P_{inicio} es un punto fijo de referencia.

b) **Proyección sobre el vector de recorrido:** Se proyecta el vector $\mathbf{v}_i$ sobre el vector de recorrido $\mathbf{v}$, para aislar su componente paralela. La proyección $\mathbf{v}_{\parallel}$ se calcula como:

$$\mathbf{v}_{\parallel} = \left(\frac{\mathbf{v}_i \cdot \mathbf{v}}{||\mathbf{v}||^2} \right) \mathbf{v}$$

donde:

- $\mathbf{v}_i \cdot \mathbf{v}$ es el producto punto entre $\mathbf{v}_i$ y $\mathbf{v}$.
- $||\mathbf{v}||$ es la norma (magnitud) del vector $\mathbf{v}$.

Esta operación permite obtener la parte de $\mathbf{v}_i$ que es colineal con $\mathbf{v}$.

c) **Cálculo de la componente normal:** La componente normal $\mathbf{n}_i$ en el punto p_i se calcula restando la componente paralela del vector original:

$$\mathbf{n}_i = \mathbf{v}_i - \mathbf{v}_{\parallel}$$

Esta diferencia extrae la parte de $\mathbf{v}_i$ que es ortogonal al vector de recorrido $\mathbf{v}$.

d) **Normalización del vector normal:** Finalmente, para obtener un vector unitario (de magnitud uno), se normaliza $\mathbf{n}_i$:

$$\hat{\mathbf{n}}_i = \frac{\mathbf{n}_i}{||\mathbf{n}_i||}$$

donde $||\mathbf{n}_i||$ representa la magnitud del vector $\mathbf{n}_i$.

Este procedimiento se repite para cada punto p_i de la nube, generando así un conjunto de normales unitarias asociadas a cada posición espacial.

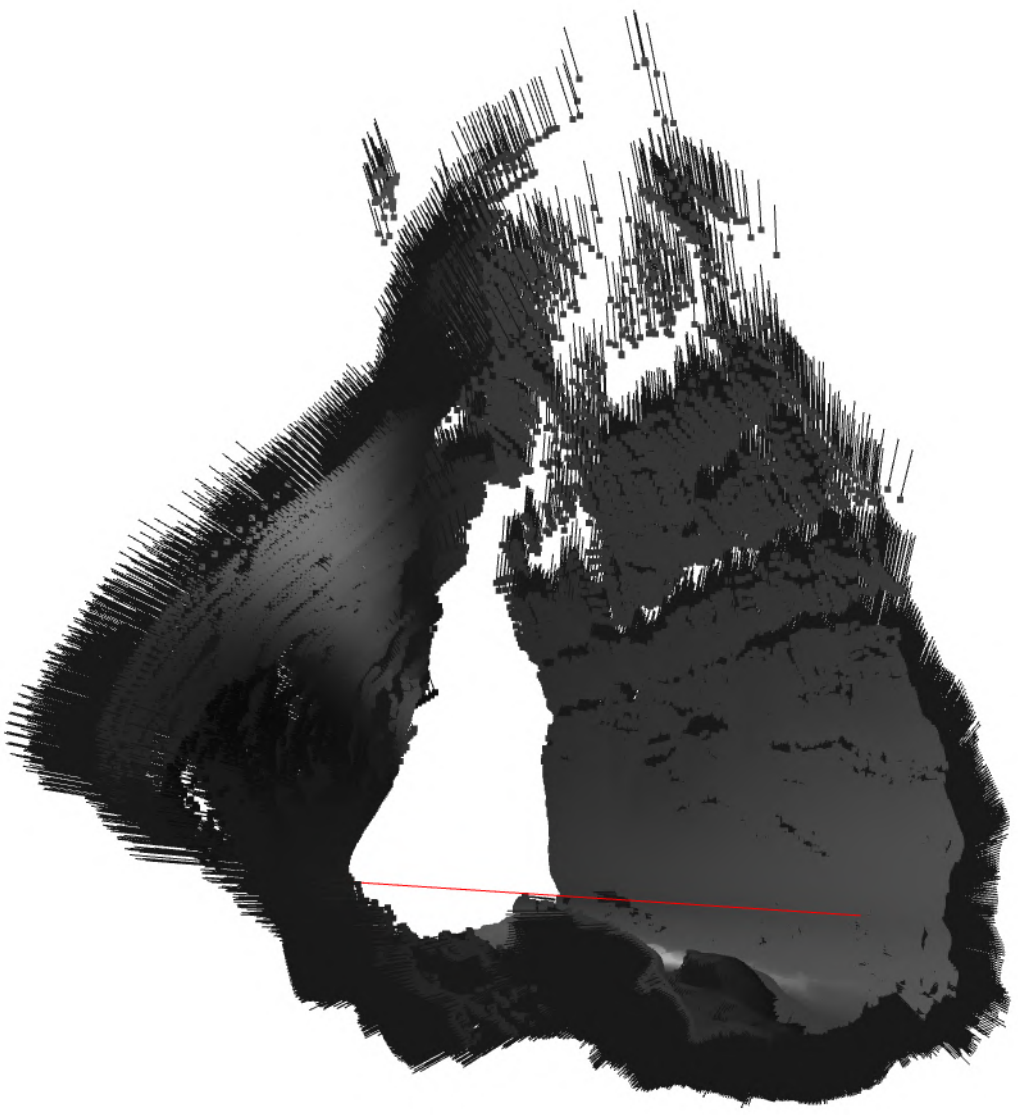

Figura 3.6: Normales con el cálculo de la línea de referencia.
Fuente:Propia

3.6.2. Asignación de Normales a la Nube de Puntos

Una vez calculadas las normales para todos los puntos, estas se asignan a la nube de puntos. Cada punto p_i en la nube tiene ahora una normal asociada $\hat{\mathbf{n}}_i$, representando la orientación de la superficie en ese punto específico.

3.6.3. Visualización de las Normales

La visualización de las normales es crucial para entender la orientación de las superficies de la nube de puntos. En la visualización, cada normal $\hat{\mathbf{n}}_i$ se representa como un vector que parte del punto p_i y apunta en la dirección de la normal.

3.6.4. Almacenamiento de la Nube de Puntos con Normales

Finalmente, la nube de puntos, junto con sus normales calculadas, se guarda en un archivo en formato PLY para su posterior análisis o visualización.

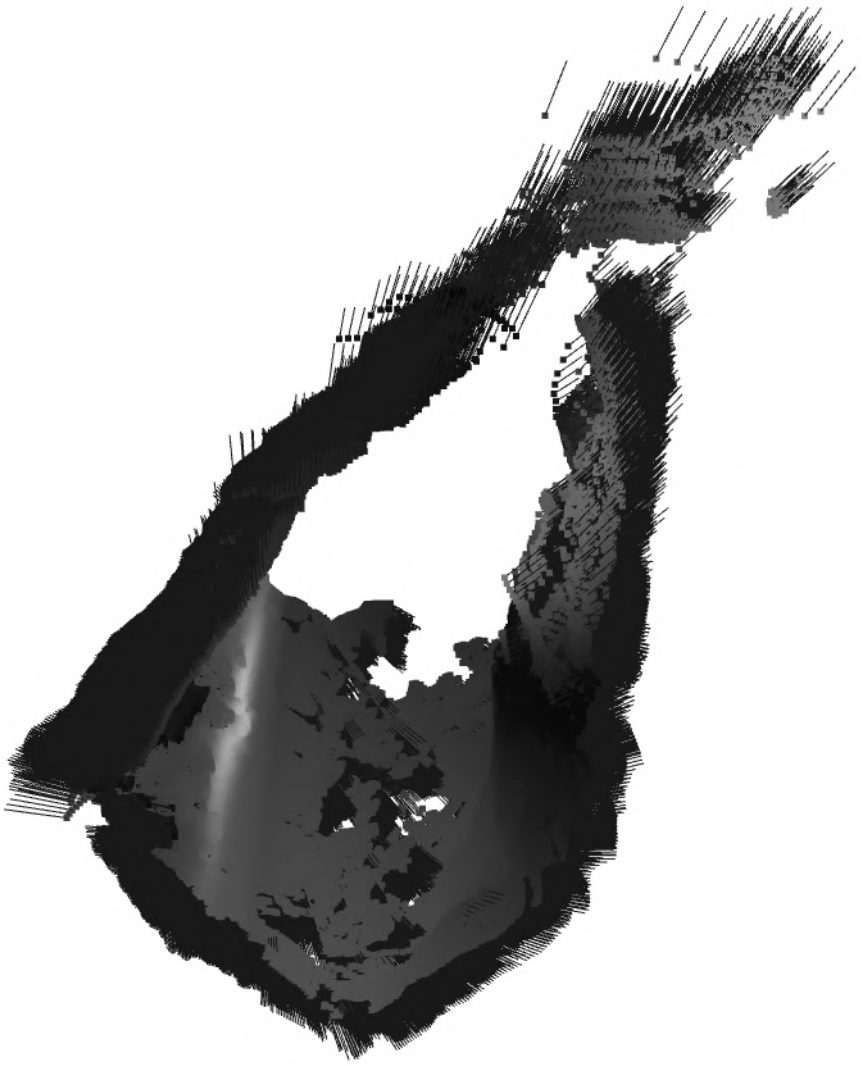

Figura 3.7: Cálculo de las normales.
Fuente:Propia

Esta etapa es muy importante para una reconstrucción 3D. Siendo entre ellas las siguientes razones:

- **Representación geométrica:** Las normales proporcionan información sobre la orientación de las superficies en el espacio. Sin esta información, es difícil comprender cómo se configura la geometría del objeto representado por la nube de puntos.
- **Mejora en la visualización:** Al visualizar nubes de puntos, las normales ayudan a crear representaciones más realistas de las superficies. Las normales se utilizan en técnicas de sombreado que simulan la iluminación en función de

la orientación de la superficie, lo que mejora significativamente la estética y la comprensión visual de los modelos 3D.

- **Aplicaciones en reconstrucción:** En procesos de reconstrucción 3D, como el mapeo y la creación de modelos, las normales son esenciales para realizar operaciones como la detección de bordes, segmentación y alineación de superficies. Sin normales precisas, los resultados de la reconstrucción pueden ser inexactos o incompletos.
- **Interacción en Simulaciones:** En simulaciones físicas y renderizado, las normales son necesarias para calcular interacciones con luces y sombras, así como para simular el comportamiento físico de objetos. Esto es particularmente importante en campos como la robótica, la realidad virtual y los videojuegos para trabajos futuros de esta investigación.
- **Facilitan el análisis Geométrico:** Las normales permiten realizar análisis geométricos avanzados, como calcular curvaturas y realizar segmentaciones en base a la orientación de las superficies. Esto es útil en aplicaciones de ingeniería, análisis de estructuras y reconocimiento de formas.

3.7. Algoritmo de reconstrucción Poisson

Para la reconstrucción de la superficie a partir de la nube de puntos, se utilizó el algoritmo de Poisson. Este algoritmo permite generar una malla tridimensional continua a partir de las normales calculadas en los puntos. La reconstrucción de Poisson fue seleccionada por su capacidad de producir mallas suaves y detalladas, incluso en áreas con datos dispersos o con ruido. Además, se emplearon técnicas de filtrado para reducir el ruido y mejorar la precisión en áreas complejas.

- **Carga de la nube de puntos:** Aquí se carga una nube de puntos desde un archivo PLY utilizando Open3D. La nube de puntos debe haber sido preprocesada previamente y contener normales, ya que el algoritmo de Poisson las utiliza para generar la malla.
- **Reconstrucción de la superficie con el algoritmo de Poisson:** En esta parte, se realiza la reconstrucción de la superficie con el algoritmo de Poisson. El parámetro `depth=9` controla la resolución de la reconstrucción (cuanto mayor sea el valor, más detallada será la malla). La función `create_from_point_cloud_poisson` devuelve tanto la malla como las densidades de los puntos.

  El algoritmo de Poisson resuelve la siguiente ecuación diferencial:

$$\Delta\phi = \nabla \cdot V$$

Donde:

- ϕ es la superficie que se busca reconstruir.
- V es el campo vectorial que proviene de las normales de la nube de puntos.
- Δ es el operador laplaciano.

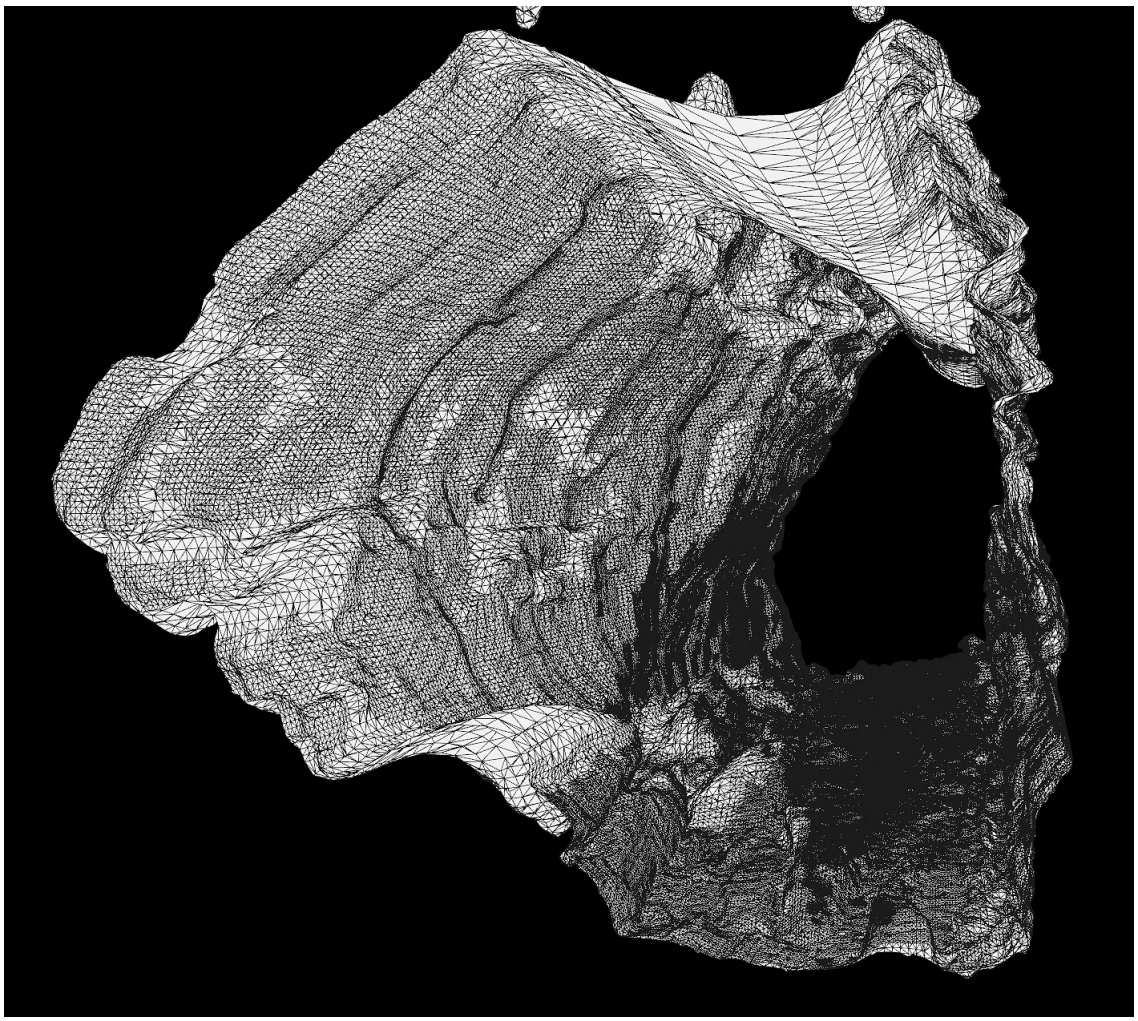

Figura 3.8: Nube de puntos reconstruido.
Fuente:Propia

Este algoritmo permite obtener una representación visual efectiva de la superficie a partir de la nube de puntos, facilitando el análisis y la interpretación de la geometría tridimensional.

Parte IV

Proceso experimental y resultados

Capítulo 4

Resultados

El diseño completo del prototipo

Figura 4.1: Drone con sensor lidar y esp32
Fuente: Propia

Figura 4.2: Drone en vuelo durante las pruebas en el Balcón del Diablo, ubicado en la comunidad de Chakan, provincia de Cusco, a 3800 m.s.n.m.

Fuente: Propia. Fotografía tomada en el Balcón del Diablo, Chakan, Cusco, Perú.
Ubicación geográfica: 13°29'08.0"S, 71°58'15.0"W

4.0.1. Interfaz gráfica

Para la demostración del prototipo se creo una interfaz gráfica en python para la manipulación amigable del código desarrollado y el montaje del móvil para la captura de datos que se visualizarán en las imagenes siguientes:

Diseño del menú principal

En este interfaz nos permite realizar el paso a paso de todo lo analizado en el desarrollo para nuestra reconstrucción: El botón captura de datos nos permite empezar con la toma de la nube de puntos, antes debe estar previamente conectado el sensor lidar al esp32 para poder emitir los datos, con esto se monta al drone para que pueda captar en el escenario definido.

Figura 4.3: Interfaz del prototipo
Fuente: Propia

Captura de datos

En este interfaz nos permite realizar el paso a paso de todo lo analizado en el desarrollo para nuestra reconstrucción: El botón captura de datos nos permite empezar con la toma de la nube de puntos, antes debe estar previamente conectado el sensor lidar al esp32 para poder emitir los datos.

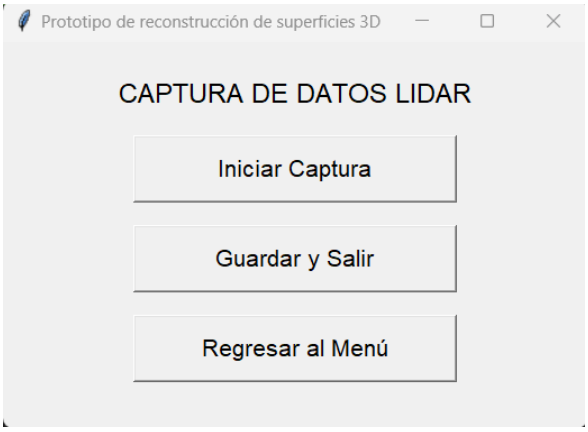

Figura 4.4: Interfaz del submenú de captura de datos
Fuente: Propia

Para el proceso de captura de datos ingresamos a iniciar captura y prosigue el sensor a capturar los datos para guardarlos en un archivo .ply en nuestro computador.

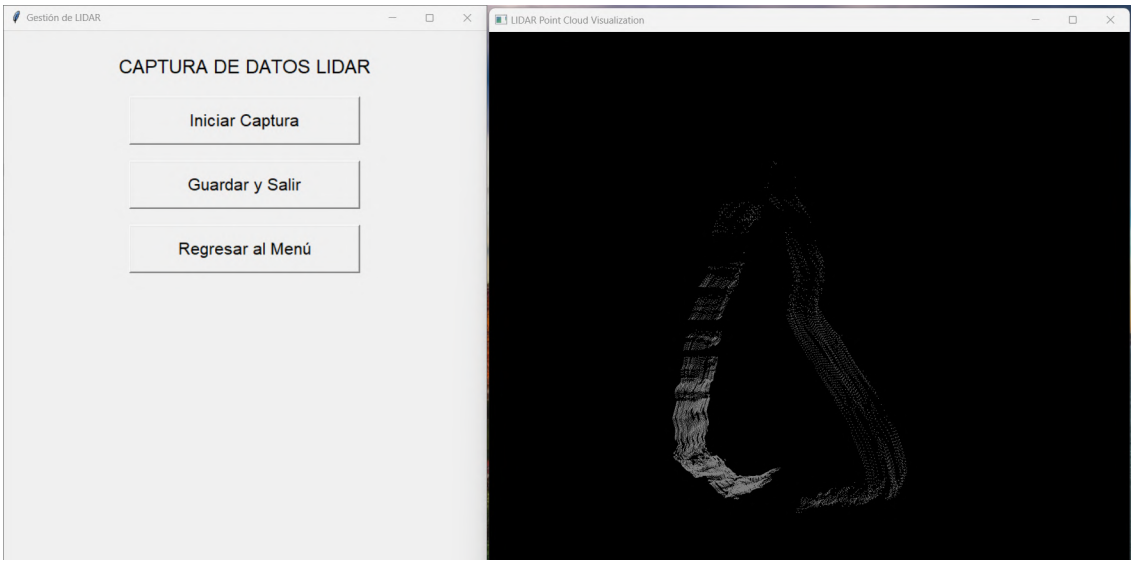

Figura 4.5: Interfaz del prototipo para la captura de datos
Fuente: Propia

Proceso de captura de datos:

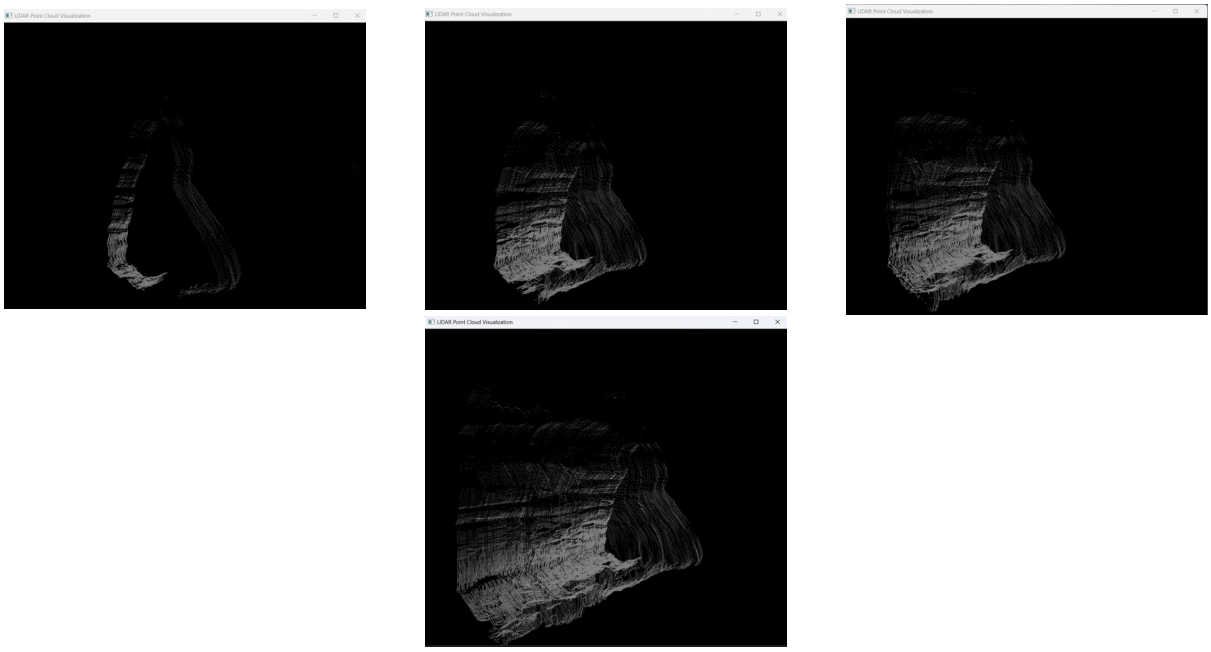

Figura 4.6: Proceso de captura de puntos
Fuente: Propia

4.1. Procesamiento de datos

Interfaz para el procesamiento de datos

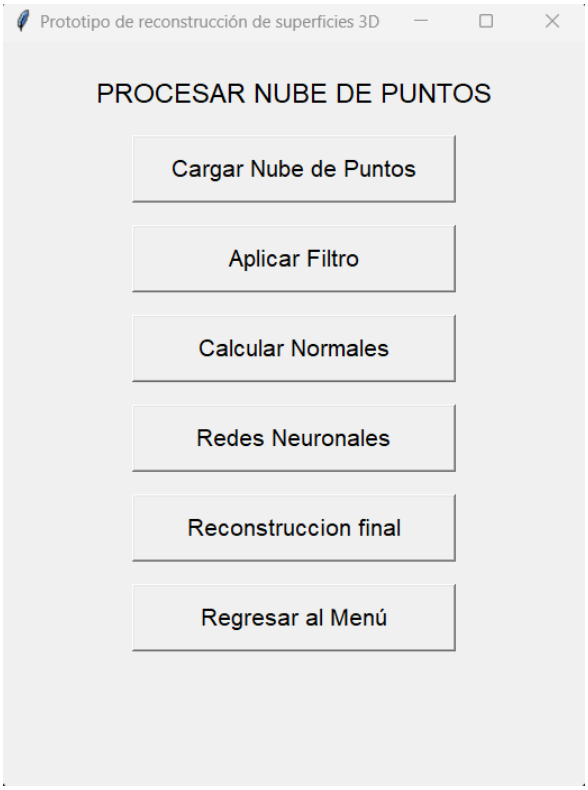

Figura 4.7: Interfaz del prototipo para procesamiento de datos
Fuente: Propia

Pasos del procesamiento

Cargar nube de puntos

Seleccionamos la nube de puntos para el procesamiento de datos:

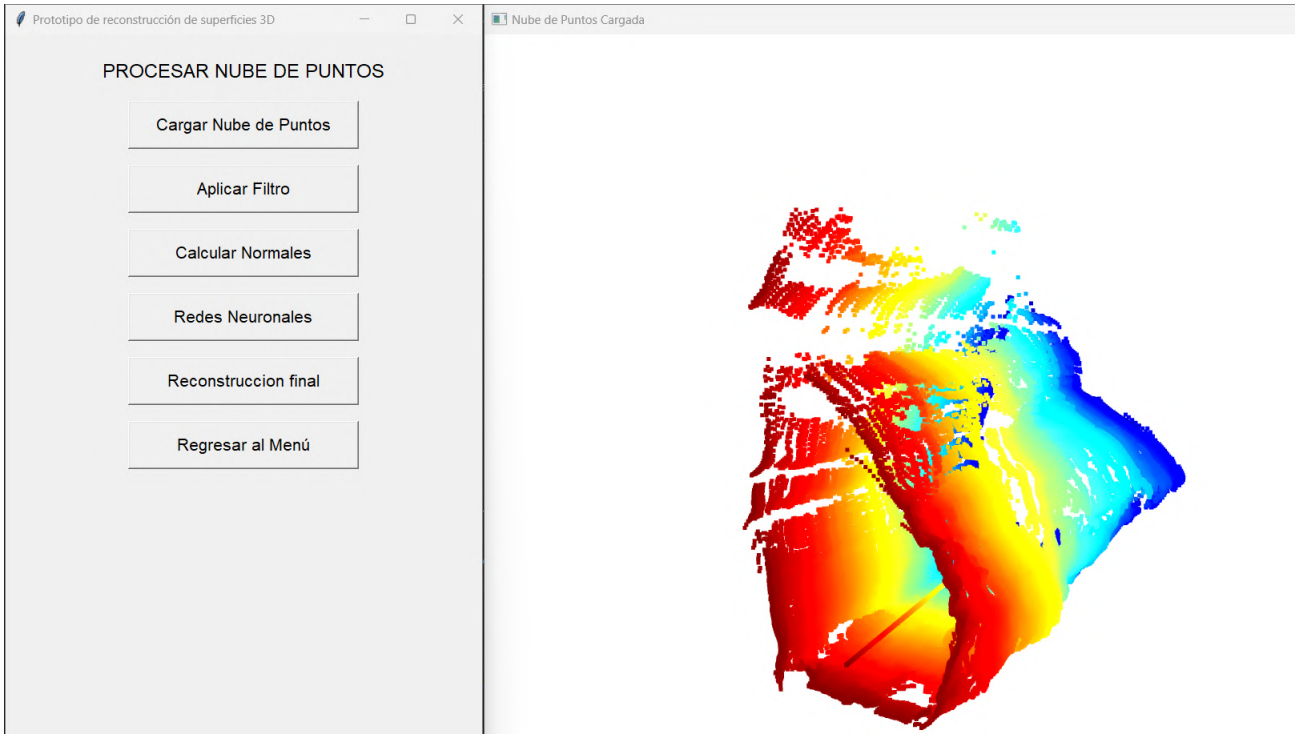

Figura 4.8: Selección de la nube a procesar
Fuente: Propia

Aplicar filtro

Esta es una técnica para la aplicación de filtro para reducción de ruido:

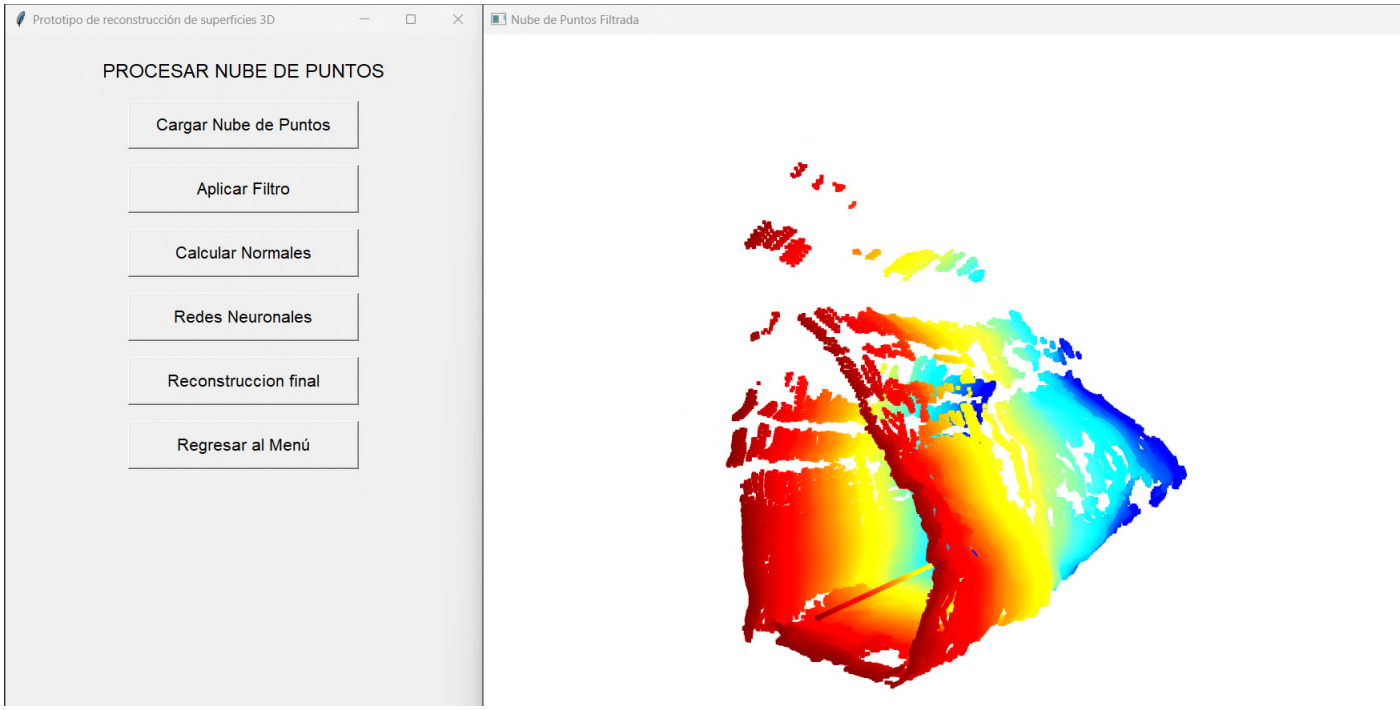

Figura 4.9: Nube de puntos después de la aplicación del filtro
Fuente: Propia

Cálculo de normales

Esta este paso de cálcula las normales de la nube a tratar después del filtrado de ruido.

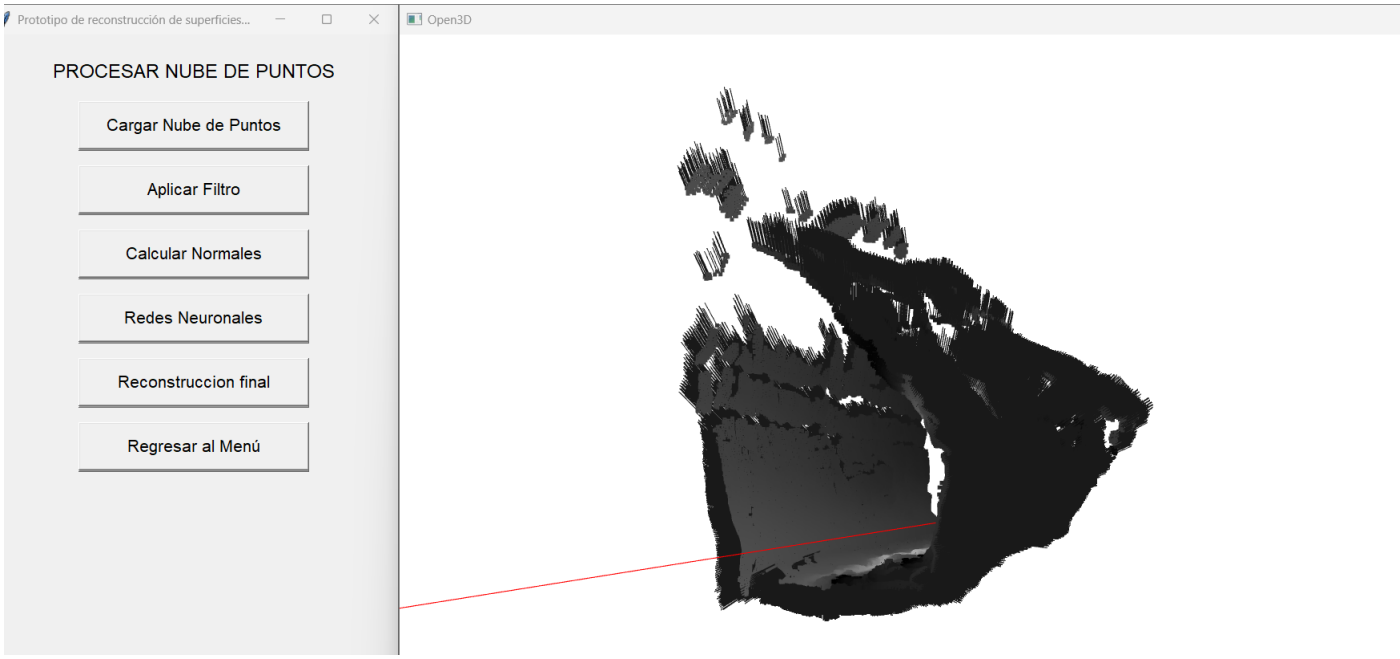

Figura 4.10: Nube de puntos con línea de referencia para cáculo de normales
Fuente: Propia

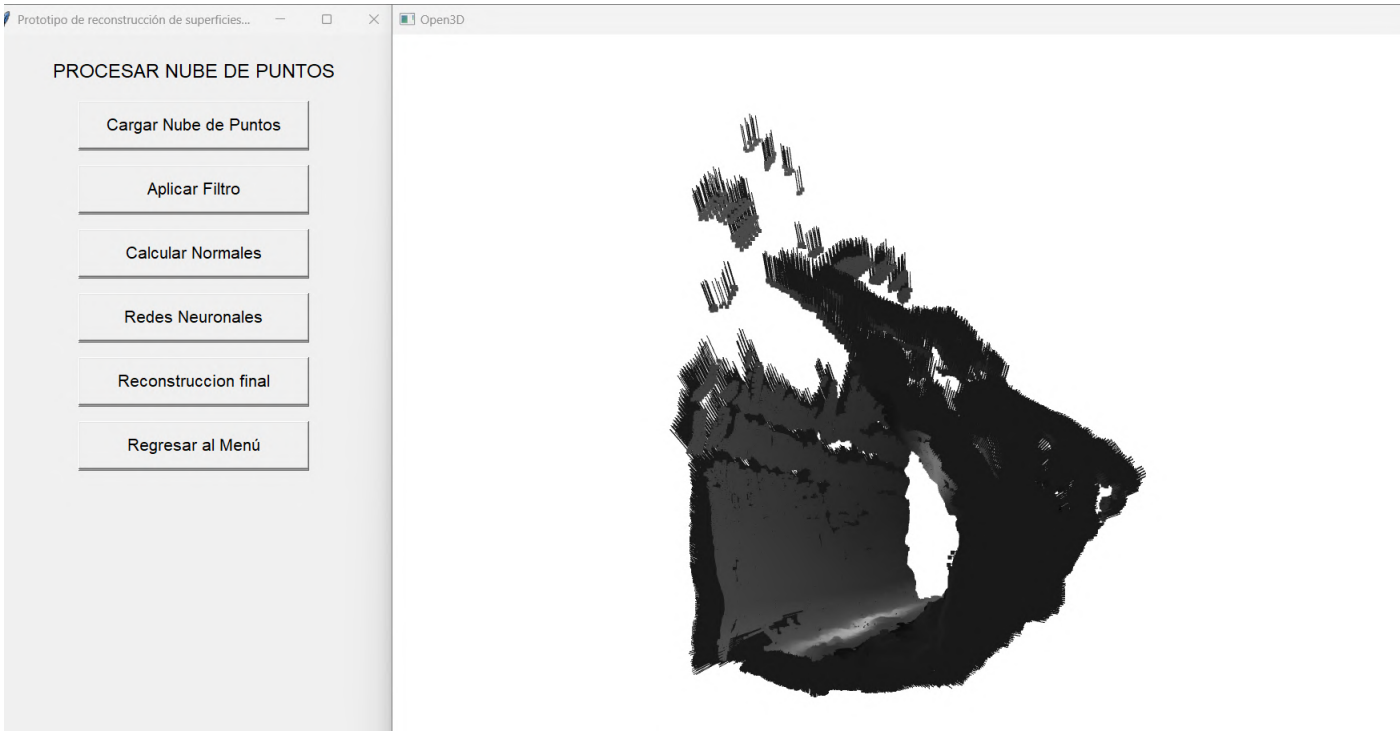

Figura 4.11: Nube de puntos con sus respectivas normales
Fuente: Propia

Imágenes de como se veían con el cálculo de normales en diferentes escenarios:

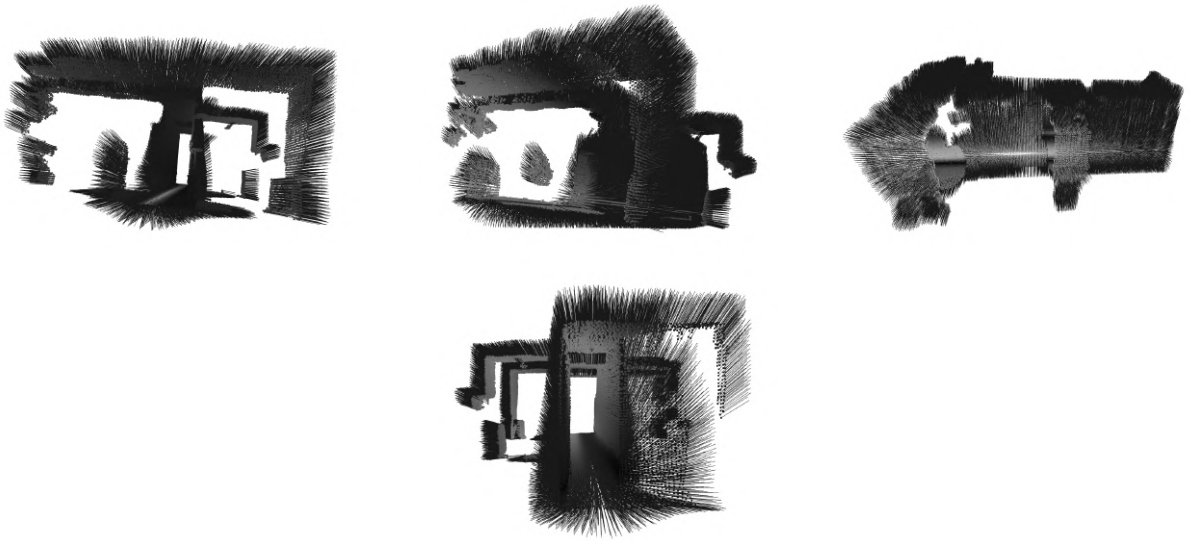

Figura 4.12: Proceso de cálculo de normales en escenario 1
Fuente: Propia

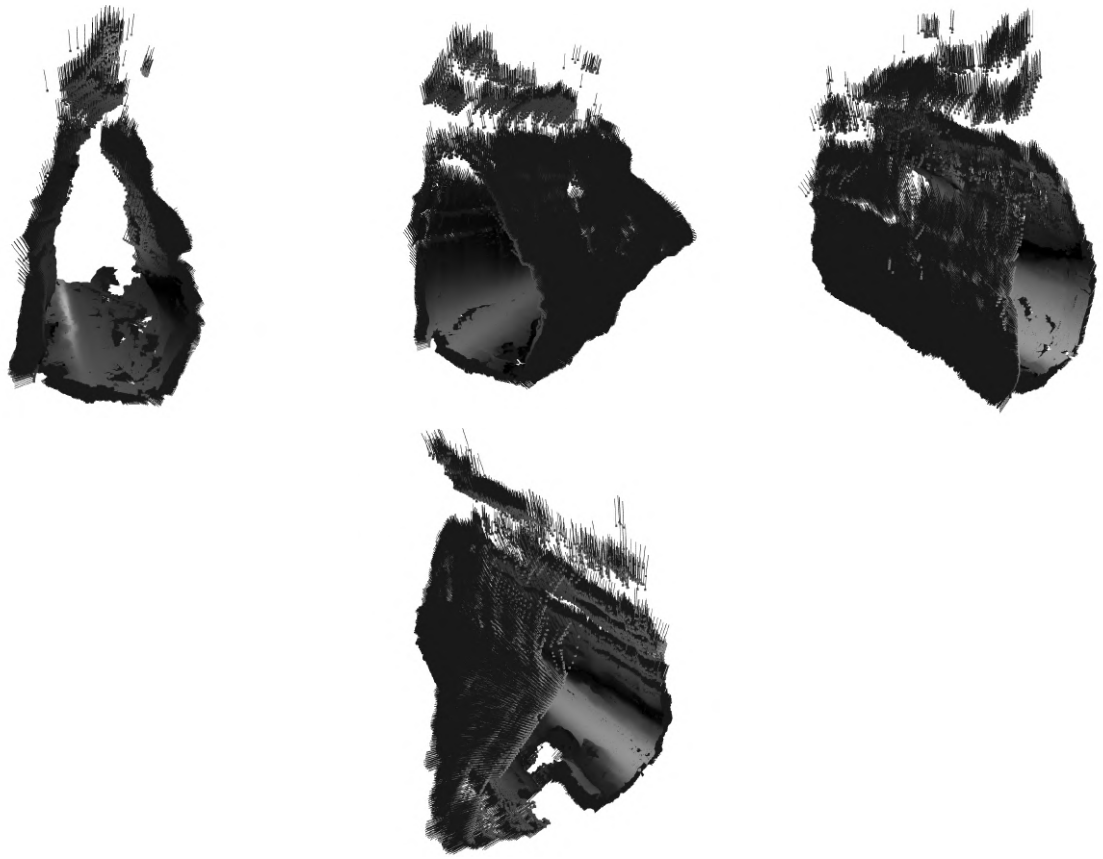

Figura 4.13: Proceso de cálculo de normales en escenario 2: Cueva balcón del diablo
Fuente: Propia

4.2. Reconstrucción final

- **Aumento de densidad con redes neuronales** Para este proceso, mostraremos como fue el cambio en la reconstrucción mediante las redes neuronales con Autoencoders, con el modelo desarrollado en el capítulo de Desarrollo, para lo cual el modelo funciona correctamente dando un margen de error mínimo conseguido de: 0.5 que se mostrara más adelante en las métricas de evaluación. En esta imagen se puede visualizar que los puntos blancos son la nube inicial de puntos y los puntos rojos son las nubes de puntos agregadas.

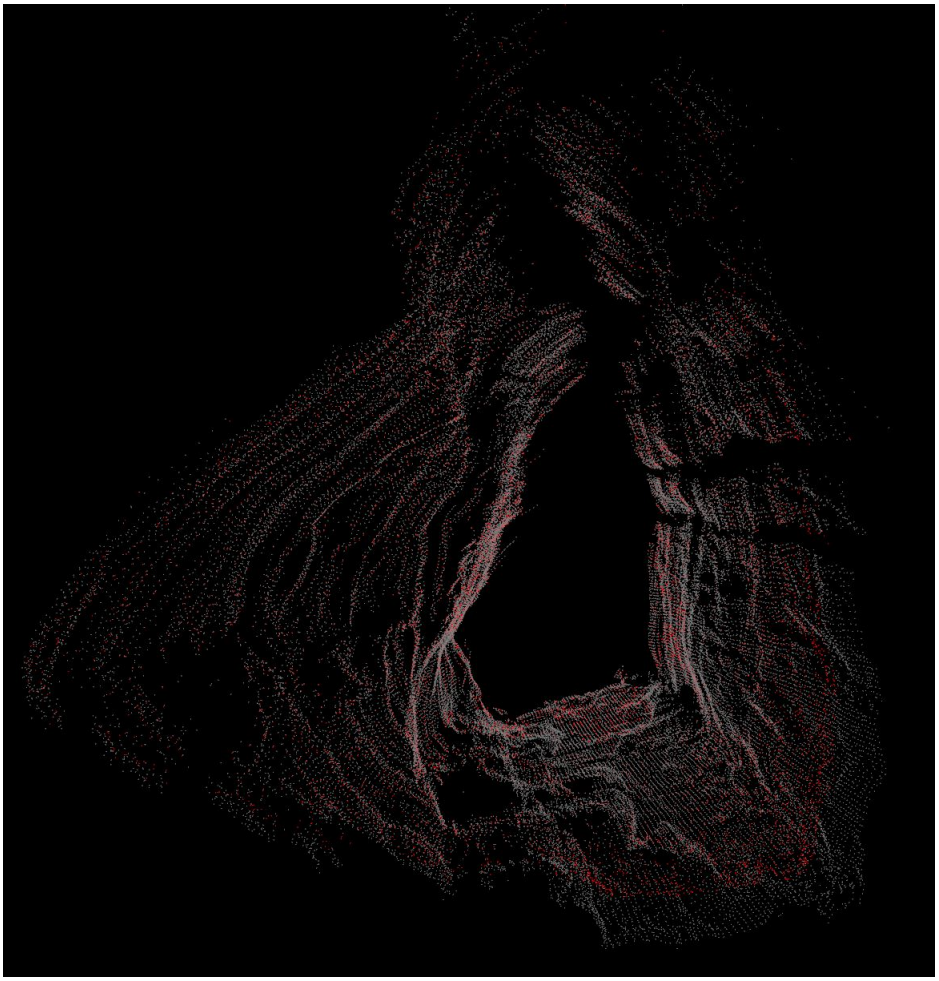

Figura 4.14: Nube de puntos con aumento de densidad
Fuente: Propia

- **Proceso de reconstrucción 3D con Poisson y malla de polígonos** En este paso se procede a la reconstrucción mediante la librería open3D. A continuación se muestra los resultados del escenario: Balcón del diablo.

Figura 4.15: Imágenes reales del balcón del diablo (Coordenadas: 13°30'45.0"S 71°58'33.0"W, Cusco, Perú)

Fuente: Propia

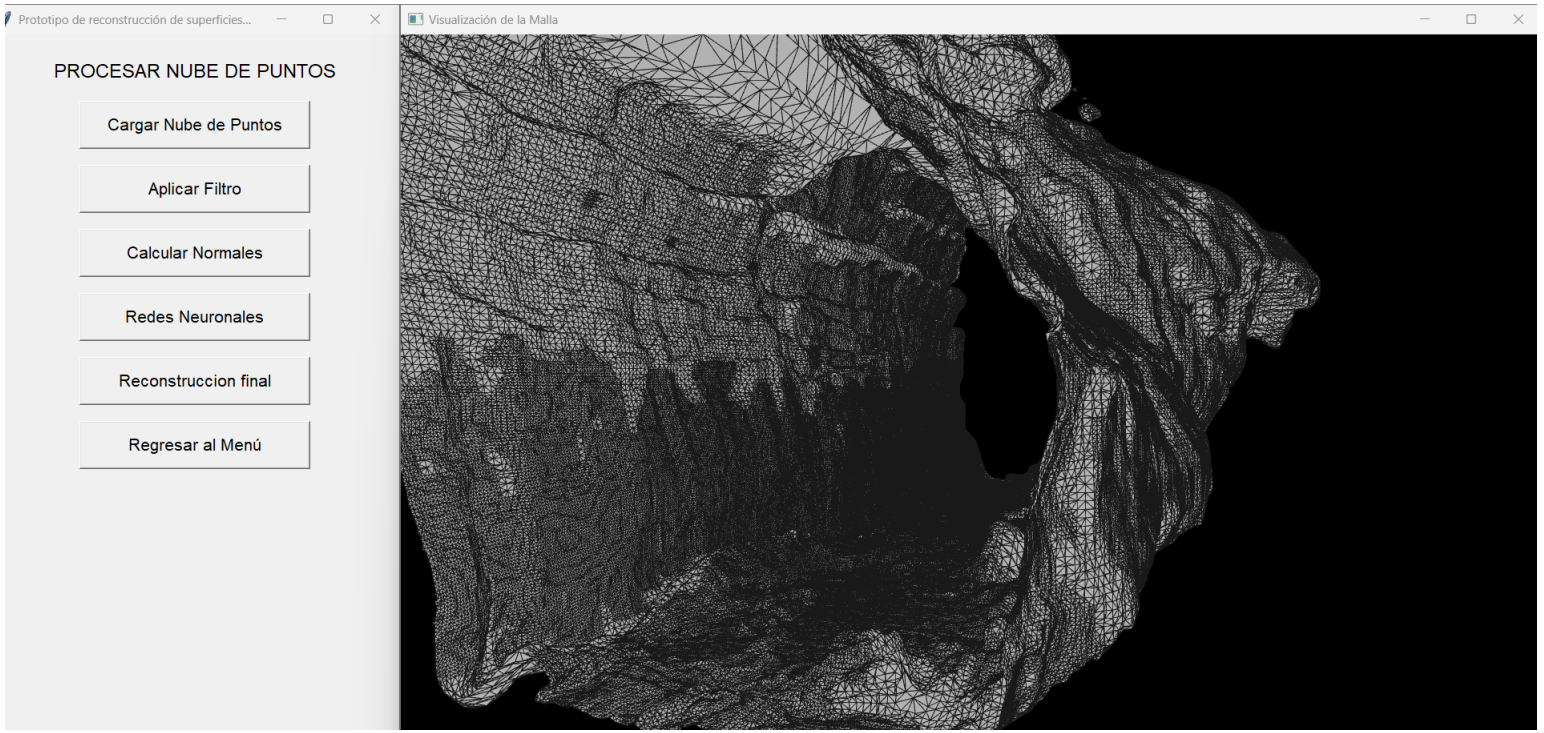

Figura 4.16: Reconstrucción final después de la aplicación de redes neuronales
Fuente: Propia

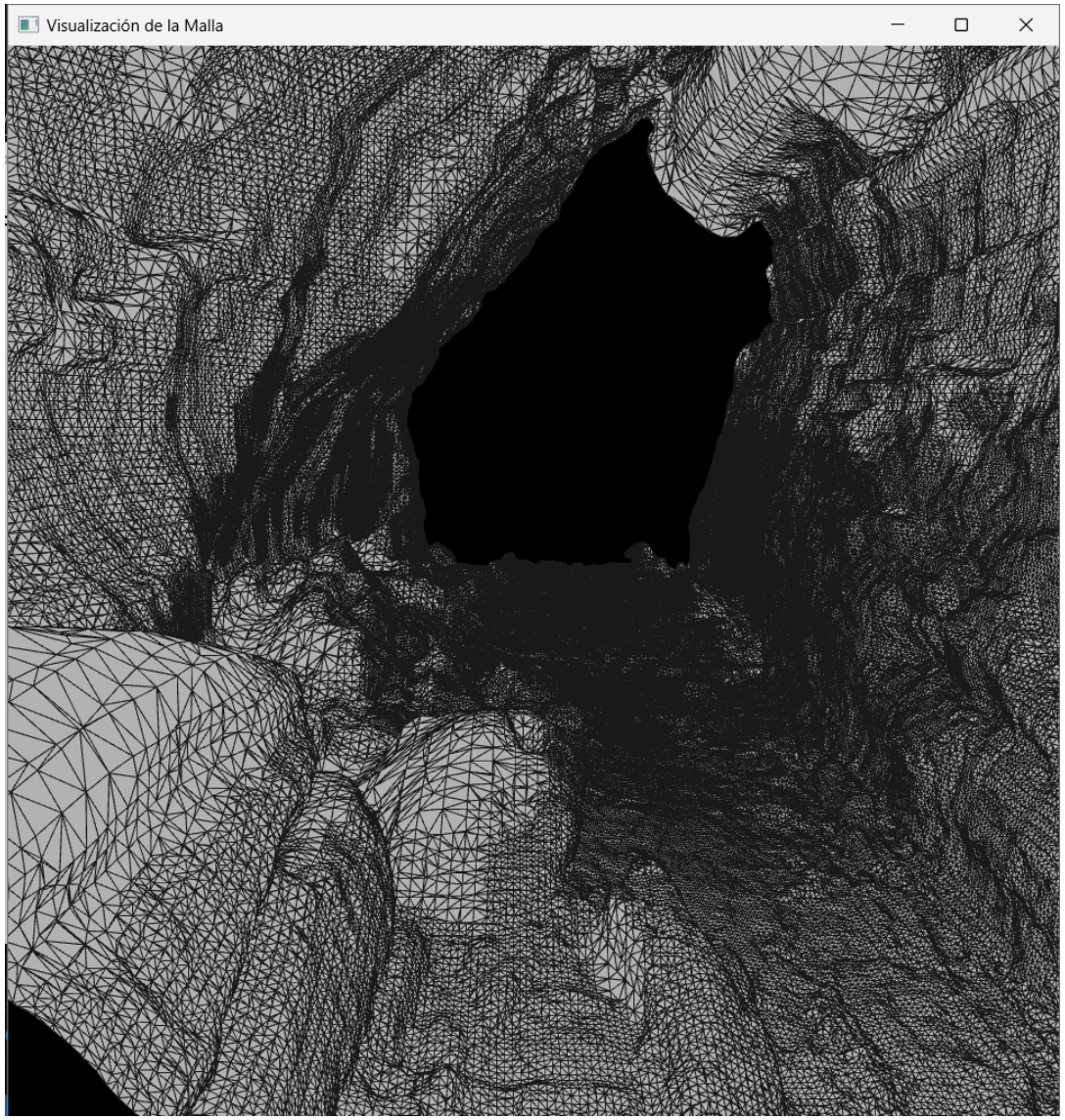

Figura 4.17: Reconstrucción final
Imagen reconstruida del balcón del diablo (Coordenadas: 13°30'45.0"S 71°58'33.0"W, Cusco, Perú)

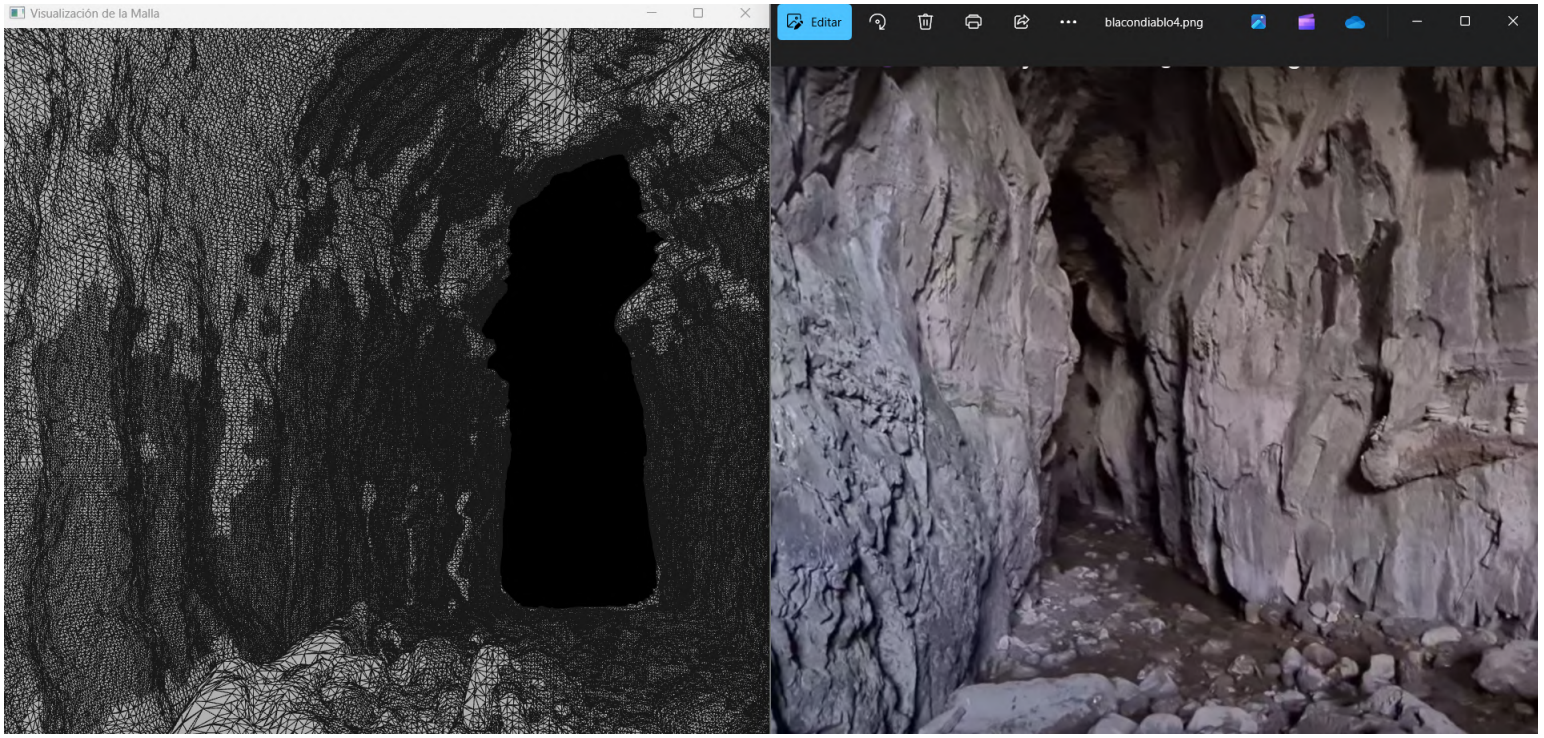

Figura 4.18: Reconstrucción final vs Imagen original
Ubicación geográfica: 13°29'08.0"S, 71°58'15.0"W **Fuente**: Propia

4.3. Métricas de evaluación del modelo

Los resultados obtenidos mediante el Autoencoder se comparan con los generados por el método de Poisson, evaluando la densidad y precisión en la reconstrucción. La métrica de Chamfer Distance se utiliza para validar cuantitativamente las diferencias.

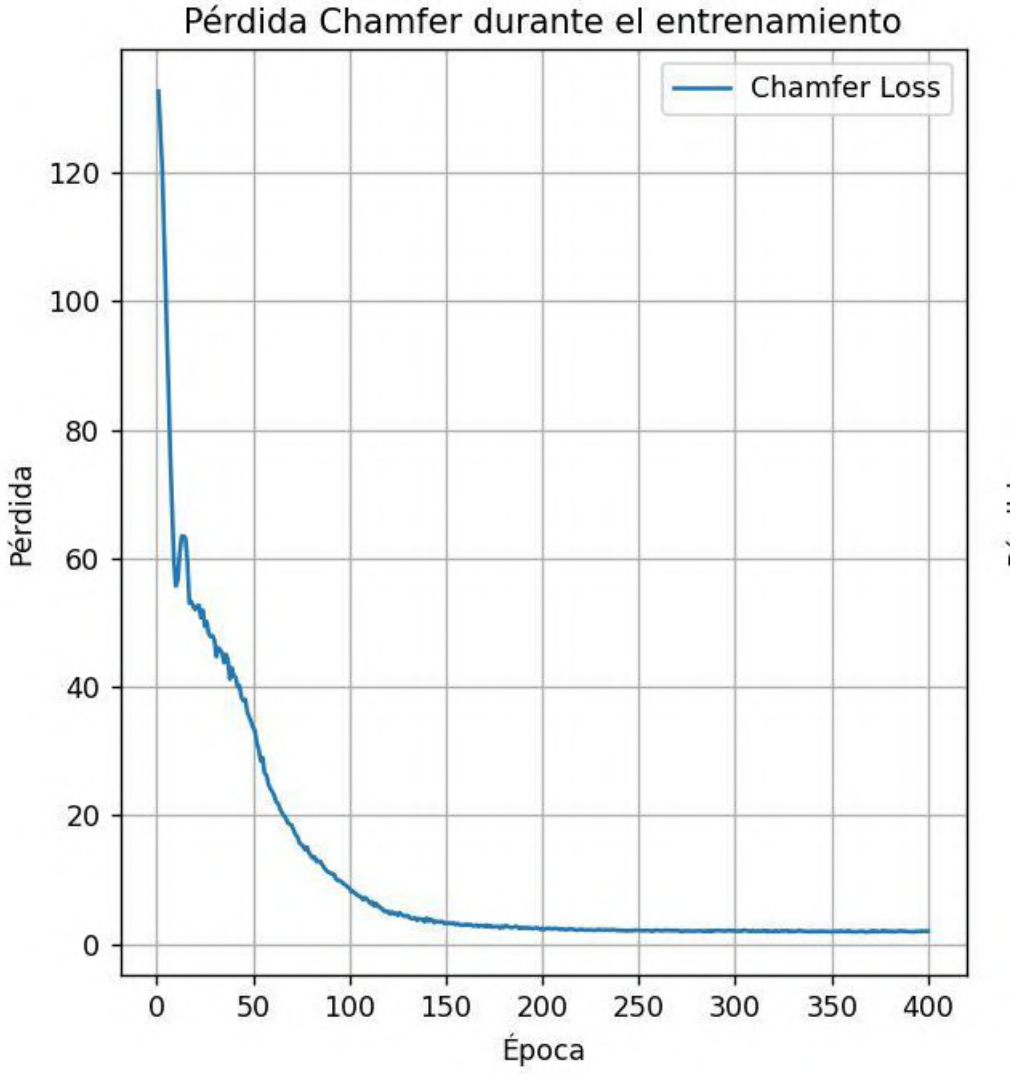

Figura 4.19: Margen de error resultado de métrica Chamfer Distance
Fuente: Propia

- **Precisión**: El Autoencoder presenta una menor pérdida de detalles finos en comparación con Poisson.
- **Densidad**: La generación de puntos adicionales por el Autoencoder es coherente con la distribución original.

Curvas de pérdida y convergencia del modelo

Durante el entrenamiento grabamos la pérdida de reconstrucción en función a las épocas para ver como el modelo mejora con el tiempo. Esto nos da una idea más clara de como el modelo esta aprendiendo, para ello utilizamos la librería de matplotlib para graficarla mediante el siguiente algoritmo:

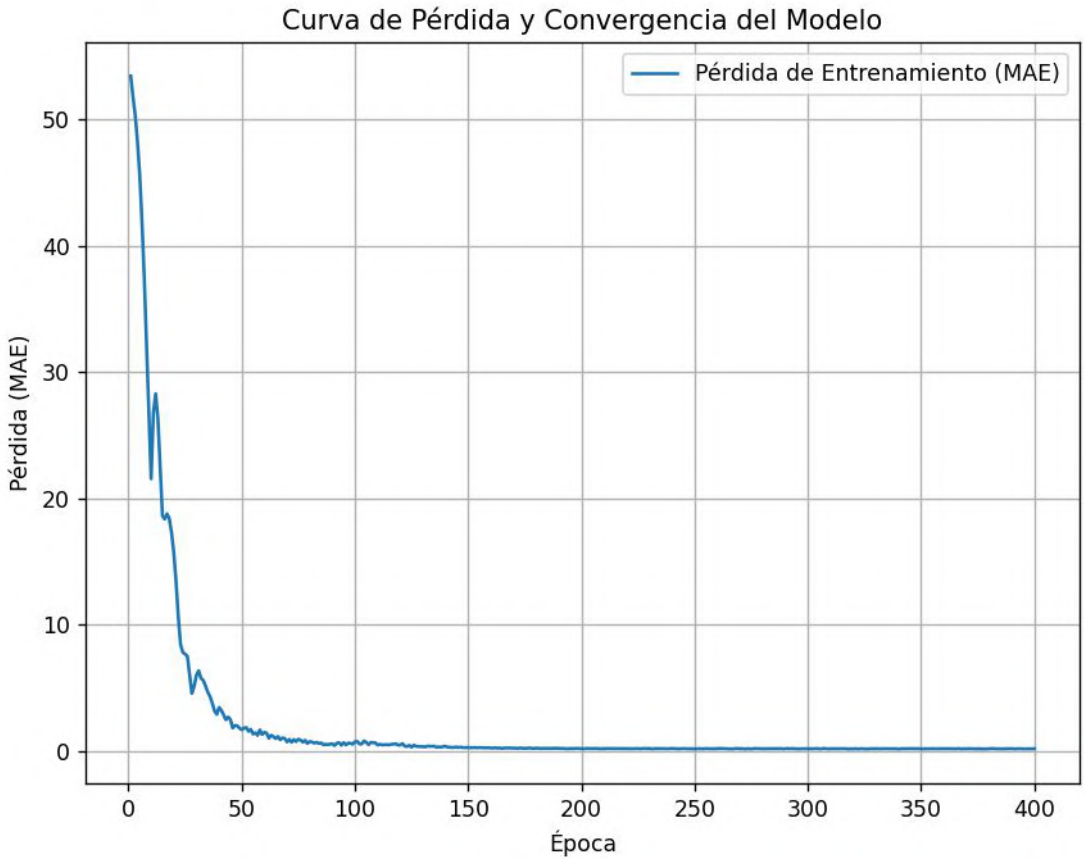

Figura 4.20: Gráfica de las curvas de pérdida y convergencia del modelo
Fuente: Propia

4.3.1. Comparación con CHAMFER DISTANCE vs MSE

A continuación se muestra una tabla con las principales características de cada métrica empleada para evaluar la reconstrucción y aumento de densidad:

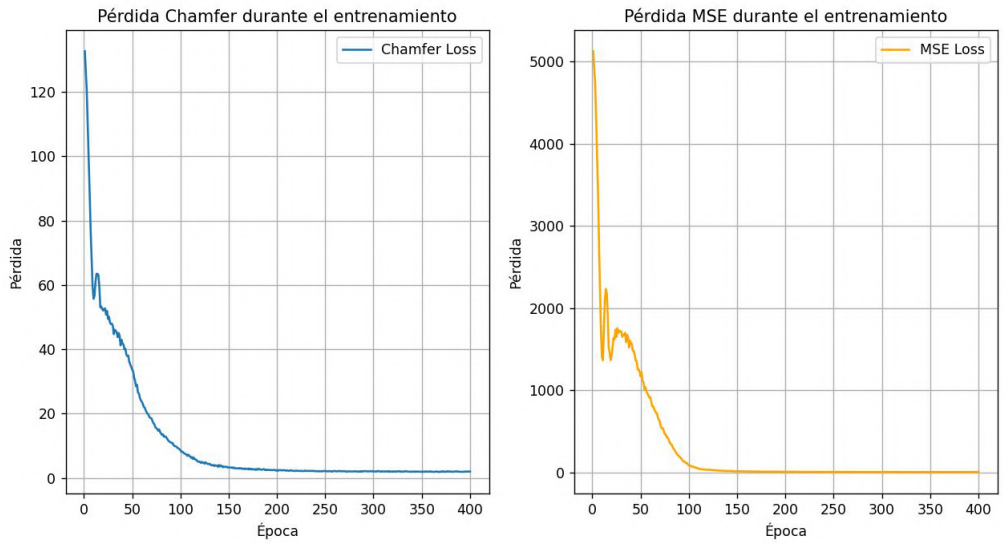

Figura 4.21: Métrica de evaluación: Chamfer Distance vs MSE
Fuente: Propia

La Figura anterior muestra la evolución de dos métricas de error utilizadas durante el proceso de entrenamiento: Chamfer Loss y MSE Loss, en función de las épocas.

Chamfer Loss (izquierda): Se observa una caída abrupta durante las primeras 50 épocas, pasando de un valor inicial superior a 130 hasta valores cercanos a 20. Posteriormente, la pérdida continúa disminuyendo de manera progresiva, estabilizándose alrededor de 1 después de la época 150. Esta tendencia sugiere que el modelo logra una mejora continua en la reconstrucción de la geometría de las nubes de puntos, capturando cada vez mejor la forma global y local de los objetos.

MSE Loss (derecha): Esta métrica también presenta una fuerte disminución inicial, partiendo de valores superiores a 5000. A pesar de mostrar algunas oscilaciones en las primeras 50 épocas, la pérdida desciende de manera más abrupta que Chamfer, alcanzando valores mínimos cercanos a 0 antes de la época 150. Esto indica que, a nivel de correspondencia punto a punto (cuando existe un emparejamiento directo entre los puntos), el modelo converge rápidamente hacia una solución precisa.

Por lo tanto, ambas curvas reflejan una buena convergencia del modelo. La Chamfer Loss, al depender de distancias punto-conjunto y no requerir correspondencia directa, tiende a ser más suave y menos oscilante. Por su parte, el MSE, aunque más sensible a desalineaciones, demuestra una rápida adaptación cuando las correspondencias son consistentes.

En conjunto, estas gráficas validan que el modelo mejora su capacidad de reconstrucción tridimensional tanto en términos de forma global (Chamfer) como de precisión local (MSE).

Aunque MSE puede funcionar bien en condiciones ideales (puntos ordenados y emparejados), Chamfer Distance es preferida en tareas con nubes de puntos no estructuradas o generadas, donde la correspondencia exacta no está garantizada. Es una métrica más natural para evaluar formas que para medir exactitud punto a punto. Además que Chamfer Distance mide la proximidad estructural entre dos formas en el espacio 3D, evalúa cuánto se parece la forma reconstruida a la original, lo cual es crucial en autoencoders, donde queremos que la estructura general de la nube se conserve, no solo los valores individuales.

Conclusiones

Entre los principales logros del proyecto, se destacan:

- Se ha revisado varias herramientas e instrumentos para la captura de la nube de puntos, para ello escogimos LDROBOT 360 LIDAR KIT, siendo uno de los sensores que el margen de error es mínimo y con las especificaciones necesarias para el diseño de nuestro prototipo.
- Se diseño satisfactoriamente a nivel de hadware y software el prototipo para el escaneo de superficies de díficil visualización.
- Se logró la reconstrucción tridimensional, aplicando redes de autoencoder que mejoró la densidad de la nube de puntos y posterior a elló se uso algoritmo de reconstrucción Poisson y mallas triangulares para su representación logrando la reconstrucción tridimensional propuesta.
- Por último, el presente trabajo presenta varias características innovadoras que lo hacen único en el contexto de la región de Cusco y en el ámbito de las tecnologías geoespaciales. Las principales aportaciones innovadoras de esta investigación son:
  - Tecnología no disponible en la región: Se introduce un enfoque que no está disponible localmente, llenando un vacío importante en las metodologías tradicionales de levantamiento topográfico y geoespacial.
  - Uso de drones para la recolección de datos: Se emplea tecnología avanzada de drones, lo que permite llegar a zonas de difícil acceso, reduciendo tiempos y costos en comparación con los métodos tradicionales.
  - Accesibilidad económica: Se propone una solución de bajo costo, lo que facilita la implementación de tecnologías avanzadas en regiones con limitaciones económicas, democratizando el acceso a estas herramientas para diversas instituciones y proyectos.

Desafíos y Limitaciones

Durante el desarrollo, surgieron algunos desafíos que marcan las limitaciones actuales del sistema y apuntan hacia futuras investigaciones:

- Optimización del procesamiento en tiempo real: La gran cantidad de datos generados requiere optimización adicional para ser procesada en tiempo real de forma eficiente, lo cual es crucial para misiones prolongadas en terrenos de difícil acceso.
- Capacidad limitada por parte del sensor LIDAR de hasta 6 m de radio para la captura,la cual no puede exceder de 12 m, considerando que los ambientes para los cuales son propuestos en esta investigación, son áreas de dificil acceso para el ser humano, pero que el dron podrá acceder.
- Poca referencia bibliográfica en relación a redes neuronales de tipo Autoencoders.
- Esta investigación no solo respondería a los problemas incialmente planteadas, sino también tiene aplicaciones multidisciplinarias:
  - **Arqueología**: Exploración de sitios frágiles sin daño estructural,
  - **Geología**: Análisis de túneles y zonas subterráneas o difícil acceso para el ser humano.
  - **Gestión de desastres**: Reconstrucción de áreas afectadas.

Recomendaciones

- **Selección adecuada del sensor LiDAR:** La calidad de los datos depende en gran medida del sensor. Se recomienda evaluar cuidadosamente parámetros como rango de medición, resolución angular, frecuencia de muestreo, precisión y peso para asegurar compatibilidad con la plataforma (dron) y adecuación al tipo de escenario a escanear.
- **Mejorar la autonomía del dron:** Se recomienda optimizar el consumo energético del dron o explorar la posibilidad de utilizar baterías de mayor capacidad para incrementar el tiempo de vuelo y así cubrir áreas más grandes sin interrupciones.
- **Optimizar el procesamiento en tiempo real:** A medida que los entornos a escanear se vuelven más complejos, es crucial mejorar los algoritmos de procesamiento para reducir el tiempo necesario en la generación de nubes de puntos y la reconstrucción de superficies, lo que puede lograrse con técnicas avanzadas de paralelización y procesamiento distribuido.
- **Implementar mejoras en la transmisión de datos:** Aunque el uso de `socket` ha mostrado buenos resultados, es recomendable explorar tecnologías de transmisión de datos más robustas, como el uso de redes 5G o Wi-Fi de alta velocidad, que permitan mayor estabilidad en la comunicación entre el dron y la estación terrestre.
- **Ampliar el uso de redes neuronales:** Se sugiere continuar investigando en el uso de redes neuronales para mejorar no solo la reconstrucción de superficies, sino también para filtrar el ruido en las nubes de puntos, lo que incrementará la calidad y utilidad de las reconstrucciones en entornos complejos.
- **Validación en diferentes entornos:** Es importante probar el prototipo en una variedad de escenarios reales para verificar su eficacia en diferentes condiciones ambientales y geográficas, como túneles, cañones, y otras áreas de difícil acceso.

Trabajos futuros

- **Integración de Texturizado y Coloración:** Con esto se propone incluir representación de textura y color, integrando datos RGB de sensores RGB-D para así representar modelos 3D más realistas.
- **Optimización de Procesamiento en Tiempo Real:** Investigar métodos para reducir el tiempo de procesamiento durante la reconstrucción, empleando técnicas más eficientes o algoritmos paralelizados para procesadores gráficos (GPU).
- **Evaluación Multimodal para Escenarios Complejos:** Proponer un sistema que combine el uso de múltiples sensores (LIDAR, RGB-D,IMU), para capturar ambientes dinámicos y mejorar la precisión de la reconstrucción 3D en entornos díficiles como áreas oscurar o con superficies reflectantes.
- **Reconstrucción escalable para grandes áreas:** Investigar métodos para dividir automáticamente grandes nubes de puntos en regiones manejables, construirlas individualmente y unirlas sin pérdida de precisión. También desarrollar herramientar para mapear áreas extensas y abiertas.
- **Simulación para Realidad Virtual y Aumentada:** Extender la reconstrucción para crear modelos optimizados para aplicaciones de realidad virtual (VR) o aumentada (AR). Proponer herramientas para interactuar con las reconstrucciones en VR, permitiendo exploraciones inmersivas y análisis detallados.

Referencias

Achlioptas, P., Diamanti, O., Mitliagkas, I., y Guibas, L. (2018). Learning representations and generative models for 3d point clouds. En *International conference on machine learning (icml).* Descargado de `https://arxiv.org/abs/1707.02392`

Alvarado, C. (2019). Implementación de un sistema de mapeo y localización a un robot hexápodo enfocado en la exploración del entorno y monitoreo de temperatura. *Universidad Politécnica Salesiana, Ecuador*.

Attene, M., y Falcidieno, B. (2006). A graph-based algorithm for 3d surface reconstruction. *Computers & Graphics*, *30*(3), 442–451. doi: 10.1016/j.cag.2005.12.002

Bernal, J., y Liu, M. (2023). High-speed cameras in dynamic surface reconstruction: Applications and advancements. *Journal of Imaging Technology*, *12*(1), 45–63.

Bernardini, F., Mittleman, J., y Rushmeier, H. (1999). The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, *5*(4), 349–359. doi: 10.1109/2945.817034

Carlevaris, M., Della Corte, G., y Russo, G. (2018). *Applications of 3d data processing.* CRC Press.

Curless, B., y Levoy, M. (1996). Volumetric scene reconstruction from range images. En *Acm siggraph conference proceedings* (pp. 303–312).

Davies, D. (2017). Plyfile: Python module for reading and writing ply files [Manual de software informático]. Descargado de `https://pypi.org/project/plyfile/` (Accedido: 2024-10-11)

DDRobot. (2023). *Lidar sensor specifications and applications.* `https://www.ddrobotec.com/lidar-sensor-specs/`. (Accedido: 2024-10-08)

DJI. (2023). *Dji avata 2 specifications.* Descargado de `https://www.dji.com/es/avata-2/specs?utm_source=chatgpt.com` (Accedido el 10 de junio de 2024)

Goodfellow, I., Bengio, Y., y Courville, A. (2016). *Deep learning.* MIT Press. Descargado de `https://www.deeplearningbook.org/`

Harris, C. R., Millman, K. J., van der Walt, S. J., y cols. (2020). Array programming with numpy. *Nature*. Descargado de `https://numpy.org/doc/` (Accedido: 2024-10-14)

Hinton, G. E., y Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*(5786), 504–507. doi: 10.1126/science.1127647

Isáis, C. A. G. (2015). *Técnicas de medición tridimensional aplicables a robótica usando proyección de luz estructurada.* (Tesis de Master, Centro de investigaciones en Óptica, A.C). Descargado de https://cio.repositorioinstitucional.mx/jspui/bitstream/1002/429/1/16407.pdf

Jiang, H., y Zhang, J. (2017). Photogrammetry for 3d reconstruction and mapping. *Journal of Applied Remote Sensing*, *11*(4), 1144-1160.

Kazhdan, M., Bolitho, M., y Hoppe, H. (2006). Poisson surface reconstruction. *ACM Transactions on Graphics (TOG)*, *25*(3), 1–13. doi: 10.1145/1141911.1141912

Kingma, D. P., y Welling, M. (2014). Auto-encoding variational bayes. En *International conference on learning representations (iclr).* Descargado de https://arxiv.org/abs/1312.6114

LeCun, Y., Bengio, Y., y Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. doi: 10.1038/nature14539

Lopéz, J. (2016). *Reconstrucción de mapas utilizando escaneo láser para la navegación de un robot móvil* (Tesis de Master, Universidad Politénico Nacional). Descargado de https://tesis.ipn.mx/bitstream/handle/123456789/20141/Reconstrucci%C3%B3n%20de%20mapas%20utilizando%20escaneo%20l%C3%A1ser%20para%20la%20navegaci%C3%B3n%20de%20un%20robot%20m%C3%B3vil.pdf

Lutz, M. (2013). *Learning python: Powerful object-oriented programming.* O'Reilly Media.

López Cuevas, E. (2022). *Análisis y mejora de nubes de puntos* (Tesis de Master, Universidad Politécnica de Madrid). Descargado de https://oa.upm.es/70664/1/TFG_Junio22_Lopez_Cuevas_Enrique.pdf

Manohar, R., y Singh, A. (2022). Ultrasonic sensors for obstacle detection: Applications and advancements. *Sensors Journal*, *22*(3), 456–478.

McKinney, W. (2017). *Python for data analysis: Data wrangling with pandas, numpy, and ipython.* O'Reilly Media.

Mikhail, E. M., Bethel, J. S., y McGlone, J. C. (1974). *Introduction to modern photogrammetry.* New York: John Wiley & Sons, Inc.

Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., y Davison, A. J. (2011). Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. En *Proceedings of the ieee international symposium on mixed and augmented reality (ismar)* (pp. 127–136). doi: 10.1109/ISMAR.2011.6092378

Ng, A. (2011). *Sparse autoencoder.* Descargado de https://cs.stanford.edu/~acoates/cs294a/sparseAutoencoder_2011new.pdf (CS294A Lecture No-

tes)

Ouhbi, A., y et al. (2020). Normal estimation for point clouds: A review. *Computers, Materials and Continua*, *64*(3), 1375–1390. doi: 10.32604/cmc.2020.09759

Paszke, A., Gross, S., Massa, F., Lerer, A., y cols. (2019). Pytorch: An imperative style, high-performance deep learning library [Manual de software informático]. Descargado de `https://pytorch.org/docs/` (Accedido: 2024-10-12)

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Dubourg, V. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Python Software Foundation. (s.f.-a). math — mathematical functions [Manual de software informático]. Descargado de `https://docs.python.org/3/library/math.html` (Accedido: 2024-10-11)

Python Software Foundation. (s.f.-b). socket — low-level networking interface [Manual de software informático]. Descargado de `https://docs.python.org/3/library/socket.html` (Accedido: 2024-10-14)

Python Software Foundation. (s.f.-c). threading — thread-based parallelism [Manual de software informático]. Descargado de `https://docs.python.org/3/library/threading.html` (Accedido: 2024-10-14)

Qi, C. R., Su, H., Mo, K., y Guibas, L. J. (2017). Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 652–660. doi: 10.1109/CVPR.2017.16

Rusu, R., y Cousins, S. (2011). 3d point cloud processing: A survey. En *Proceedings of the ieee international conference on robotics and automation (icra)* (pp. 1–5). IEEE.

Rusu, R. B., Blodow, N., y Beetz, M. (2011). Real-time 3d object recognition and localization for autonomous robotic systems. *Robotics and Autonomous Systems*, *59*(1), 65–78.

Siegwart, R., Nourbakhsh, I. R., y Scaramuzza, D. (2011a). *Introduction to autonomous mobile robots* (2nd ed.). MIT Press.

Siegwart, R., Nourbakhsh, I. R., y Scaramuzza, D. (2011b). *Introduction to autonomous mobile robots* (2nd ed.). MIT Press.

Snavely, N., Seitz, S. M., y Szeliski, R. (2008). Modeling the world from internet photo collections. *International Journal of Computer Vision*, *80*(2), 189–210. doi: 10.1007/s11263-007-0107-y

TensorFlow. (s.f.). Tensorflow: An end-to-end open-source machine learning platform [Manual de software informático]. Descargado de `https://www.tensorflow.org/api_docs/` (Accedido: 2024-10-11)

Thrun, S., Burgard, W., y Fox, D. (2005). *Probabilistic robotics*. MIT Press.

van Rossum, G., y Drake, F. L. (2009). *Python 3 reference manual*. CreateSpace Independent Publishing Platform.

Vincent, P., Larochelle, H., Bengio, Y., y Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. En *International conference on machine learning (icml)* (pp. 1096–1103). doi: 10.1145/1390156.1390294

Wang, Z., Li, H., y Liu, J. (2017). A deep learning-based approach for 3d point cloud reconstruction. En *Proceedings of the ieee international conference on computer vision (iccv)* (pp. 2478–2487). doi: 10.1109/ICCV.2017.267

Wulf, J., y Carter, E. (2022). Lidar and structured light: Comparison and integration for 3d point cloud generation. *3D Imaging Systems Journal*, *14*(4), 301–320.

Yuan, Y., Liu, S., y Yang, L. (2018). Multi-view stereo: A review. *Computer Vision and Image Understanding*, *171*, 1–22. doi: 10.1016/j.cviu.2018.03.005

Zhang, L., y Yang, X. (2018). Real-time 3d vision and applications with intel realsense cameras. *Journal of Computer Vision and Image Processing*, *5*(2), 35-44.

Zhang, W., y Kim, S. (2023). Radar technology in adverse weather conditions: Challenges and solutions. *Journal of Advanced Sensing*, *10*(2), 34–50.

Zhang, Y., y et al. (2019). A comprehensive review on point cloud registration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *41*(7), 1641–1658. doi: 10.1109/TPAMI.2018.2871881

Zhang, Z. (2012). Microsoft kinect sensor and its effect. *IEEE MultiMedia*, *19*(2), 4–10.

Zhou, Q.-Y., Park, J., y Koltun, V. (2018). Open3d: A modern library for 3d data processing [Manual de software informático]. Descargado de `https://www.open3d.org/docs/release/` (Accedido: 2024-10-14)

Anexos

Fotografías del proceso de pruebas

Pruebas realizadas

Diseño del prototipo y muestra antes de las pruebas

Se preparó el prototipo para sus primeras pruebas, calibrando el desplazamiento con el que recorrerá, realizando las conexiones necesarias para su funcionamiento:

Figura 4.22: Diseño del prototipo
Fuente: Propia

Pruebas de vuelo

Se realizó las pruebas de vuelo y que todo este funcionando antes de entrar al escenario de prueba.

Figura 4.23: Pruebas de vuelo
Fuente: Propia

Reconstrucción con diferentes algoritmos de reconstrucción

Se capturó las datas y realizarón pruebas con otros algoritmos de reconstrucción
Algoritmo de Alpha shape

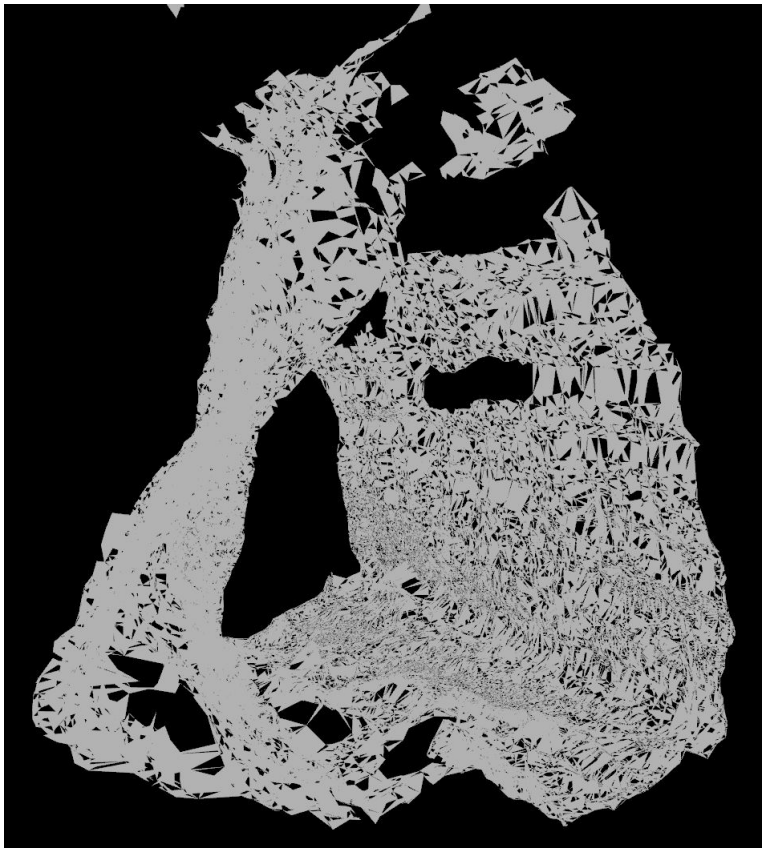

Figura 4.24: Reconstrucción con Alpha Shape
Fuente: Propia

Algoritmo de Ball Pivoting

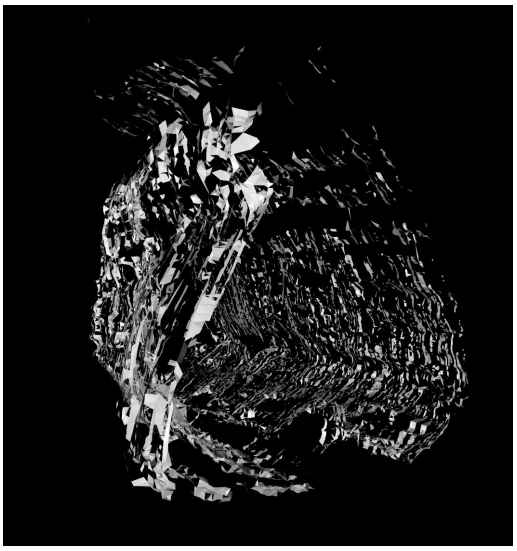

Figura 4.25: Reconstrucción con ball pivoting
Fuente: Propia

Algoritmo de Poisson sin normales ni redes neuronales

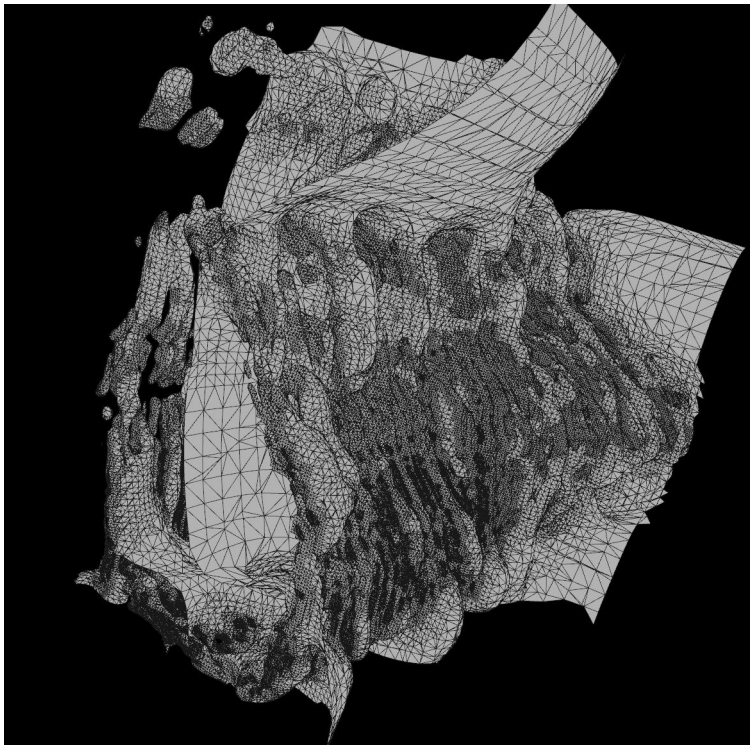

Figura 4.26: Primeras capturas balcón del diablo sin normales y redes neuronales
Fuente: Propia

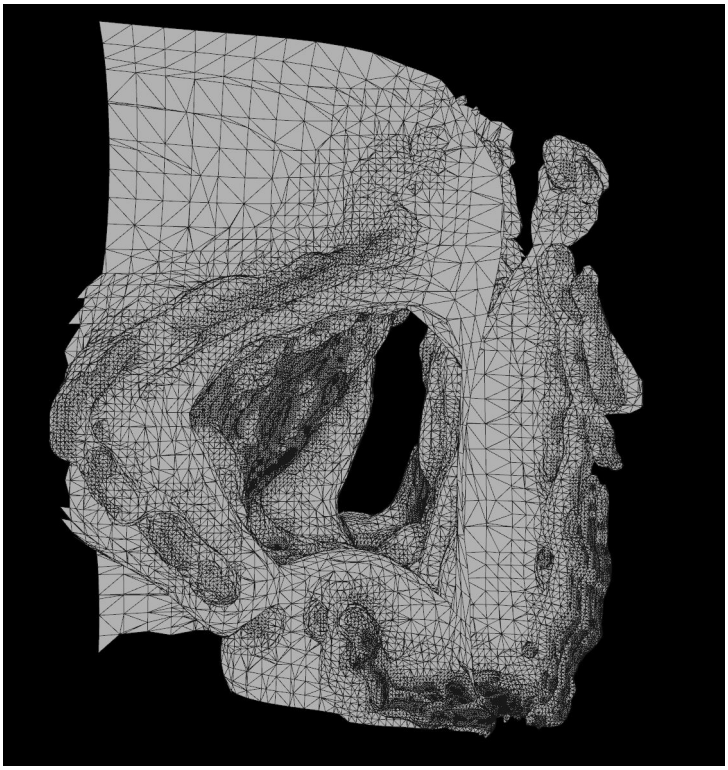

Figura 4.27: Primeras capturas balcón del diablo sin normales y redes neuronales - segunda vista

Fuente: Propia

Escenarios de prueba

Se realizó las pruebas en diferentes ambientes cerrados, de aproximadamente 5x5 metros, no se realizo pruebas en lugares abiertos puesto que no son propuestos en este proyecto de investigación, considerando lugares inaccesibles para el ser humano.

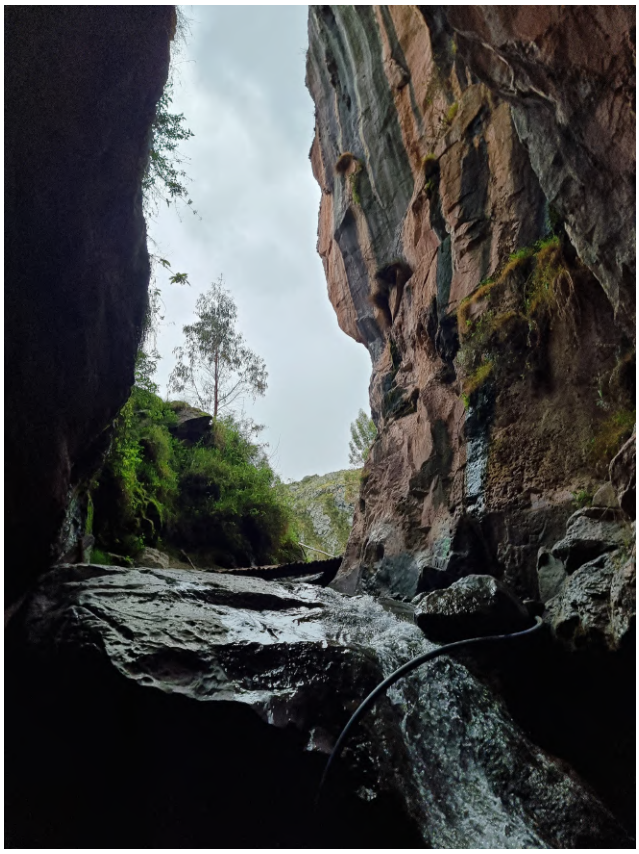

Figura 4.28: Escenario de pruebas vista 1
Fuente: Propia

Figura 4.29: Escenario de pruebas vista 2
Fuente: Propia

Pruebas realizadas en cueva balcón del diablo

Se realizarón diferentes prubeas, para poder calibrar el tiempo de recorrido, duración de las baterias, distancia de desplazamientos y también pruebas de manejo del drone.

Figura 4.30: Toma de datos
Fuente: Propia

Aplicación de normales mediante algoritmo propio vs algoritmo de open3D

Se probo con otros algoritmos para el cálculo de normales entre los cuales usamos uno que viene dentro de la librería de open3D, en la siguiente imagen se puede visualizar la diferencia y razón por la cual no fue considerada en este proyecto:

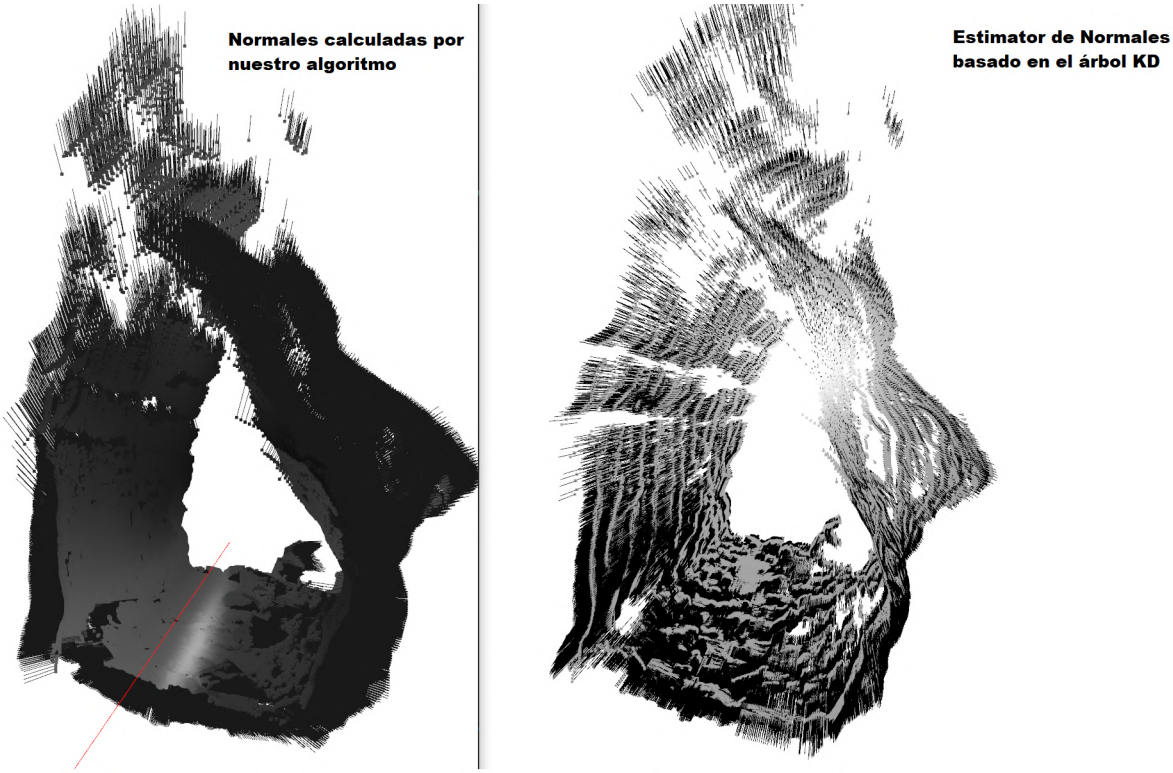

Figura 4.31: Aplicación de normales
Fuente: Propia

Librería de Pytorch 3D

También se pudo investigar diferentes avances que se viene desarrollando por parte de la librería Pytorch, el cuál viene desarrollando la reconstrucción mediante mallas 3D y aprendizaje profundo.

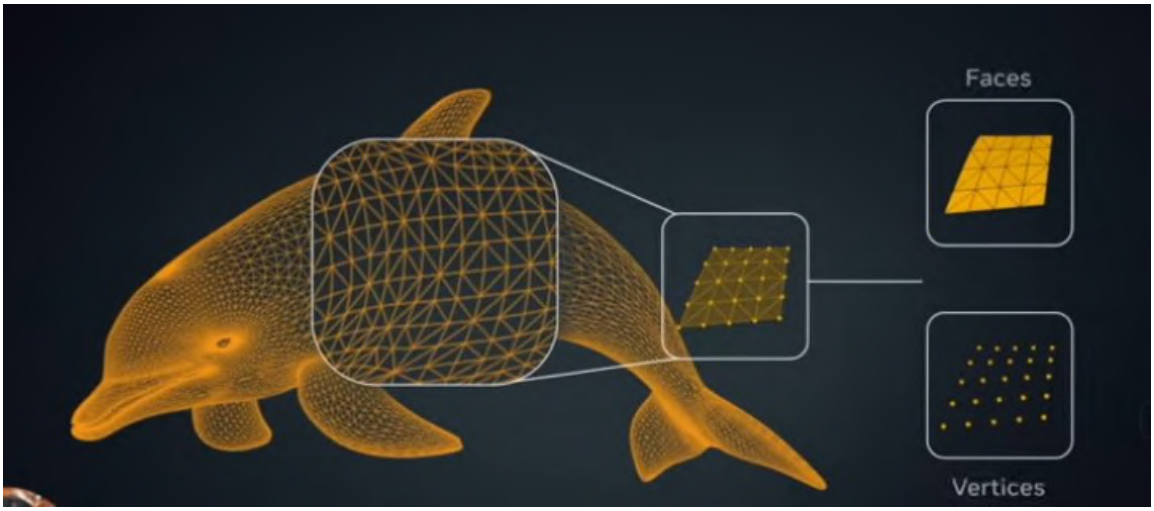

Figura 4.32: Reconstrucción de objetos mediante redes neuronales y malla de triángulos desde tecnología de Pytorch 3D

Fuente: https://ai.meta.com/blog/-introducing-pytorch3d-an-open-source-library-for-3d-deep-learning/