ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

CARRERA TELECOMUNICACIONES

"DISEÑO DE UN SISTEMA DE RECONSTRUCCIÓN AMBIENTAL 3D INDOOR QUE PERMITA DETERMINAR LA TRAYECTORIA DE UN ROBOT MÓVIL MEDIANTE ANÁLISIS Y PROCESAMIENTO DE IMÁGENES"

Trabajo de Integración Curricular
Tipo: Proyecto Técnico

Presentado para optar al grado académico de:
INGENIERO EN ELECTRÓNICA, TELECOMUNICACIONES Y REDES

AUTOR:

NELSON STALIN SHAGÑAY RUIZ

Riobamba – Ecuador

2023

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

CARRERA TELECOMUNICACIONES

"DISEÑO DE UN SISTEMA DE RECONSTRUCCIÓN AMBIENTAL 3D INDOOR QUE PERMITA DETERMINAR LA TRAYECTORIA DE UN ROBOT MÓVIL MEDIANTE ANÁLISIS Y PROCESAMIENTO DE IMÁGENES"

Trabajo de Integración Curricular
Tipo: Proyecto Técnico

Presentado para optar al grado académico de:
INGENIERO EN ELECTRÓNICA, TELECOMUNICACIONES Y REDES

AUTOR: NELSON STALIN SHAGÑAY RUIZ

DIRECTOR: ING. OSWALDO GEOVANNY MARTINEZ GUASHIMA MSc.

Riobamba – Ecuador

2023

Yo, NELSON STALIN SHAGÑAY RUIZ, declaro que el presente Trabajo de Integración Curricular es de mi autoría y los resultados del mismo son auténticos. Los textos en el documento que provienen de otras fuentes están debidamente citados y referenciados.

Como autor asumo la responsabilidad legal y académica de los contenidos de este Trabajo de Integración Curricular; el patrimonio intelectual pertenece a la Escuela Superior Politécnica de Chimborazo.

Riobamba, 31 de marzo de 2023

Nelson Stalin Shagñay Ruiz
1723371439

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

CARRERA TELECOMUNICACIONES

El Tribunal del Trabajo de Integración Curricular certifica que: El Trabajo de Integración Curricular; Tipo: Proyecto Técnico, **"DISEÑO DE UN SISTEMA DE RECONSTRUCCIÓN AMBIENTAL 3D INDOOR QUE PERMITA DETERMINAR LA TRAYECTORIA DE UN ROBOT MÓVIL MEDIANTE ANÁLISIS Y PROCESAMIENTO DE IMÁGENES"**, realizado por el señor: **NELSON STALIN SHAGÑAY RUIZ**, ha sido minuciosamente revisado por los Miembros del Tribunal del trabajo de titulación, el mismo que cumple con los requisitos científicos, técnicos, legales, en tal virtud el Tribunal Autoriza su presentación.

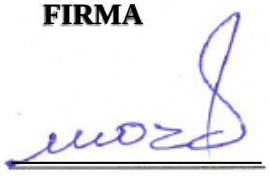
	FIRMA	**FECHA**
Ing. Franklin Geovanni Moreno Montenegro **PRESIDENTE DEL TRIBUNAL**		2023-03-31
Ing. Oswaldo Geovanny Martínez Guashima **DIRECTOR DE TRABAJO DE INTEGRACIÓN CURRICULAR**		2023-03-31
Ing. Verónica Elizabeth Mora Chunllo **ASESORA DEL TRABAJO DE INTEGRACIÓN CURRICULAR**		2023-03-31

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE ANEXOS

RESUMEN

El objetivo de este Trabajo de Integración Curricular fue diseñar e implementar un sistema de reconstrucción 3D de entornos indoor, que permita definir la trayectoria de un robot móvil teleoperado, para lo cual se implementó el robot móvil que contiene una cámara MYNT EYE S1030 para la captura de imágenes que, mediante la técnica de visión estéreo, poder reconstruir un escenario indoor, el funcionamiento del sistema se determinó mediante un análisis de aleatoriedad. La reconstrucción de entornos y posterior cálculo de la trayectoria se definió por etapas, las cuales fueron: Análisis, Diseño, Implementación, Evaluación y Mantenimiento del sistema, donde se definen subetapas para el algoritmo de reconstrucción programado en MATLAB. Para el análisis de aleatoriedad y comparativo se decidió aplicar distintas pruebas estadísticas como prueba de Shapiro, prueba F y prueba T de Student para evaluar el sistema, de lo cual se obtuvo que, el sistema difiere en medidas comparado con la herramienta comercial COLMAP; en el análisis de exactitud con el entorno real, se tiene que el sistema difiere, pero en menor escala haciendo un cálculo de error absoluto, dando un error promedio de 3.27 metros con respecto al valor real, y finalmente, se hizo una reconstrucción del escenario indoor variando parámetros de este. Se concluye que el sistema da resultado aceptables y no altamente confiables debido a la cámara y la calidad de las imágenes capturadas, ya que la creación de las nubes de puntos genera ruido, por lo que el algoritmo ICP de unión de nubes de puntos sobrepone las nubes de puntos que hacen que se pierda poco a poco las dimensiones reales del escenario. Se recomienda complementar la técnica de visión estéreo con sensores de profundidad para llegar a obtener valores próximos a los reales y así mejorar la reconstrucción de entornos.

Palabras clave: <VISION ESTÉREO>, <RECONTRUCCIÓN 3D>, <ROBOT MÓVIL>, <MATLAB (SOFTWARE)>, <COLMAP (SOFTWARE)>, <CALIBRACIÓN>, <PUNTO ITERATIVO MÁS CERCANO (ICP)>, <ALGORITMO ICP>.

SUMMARY

The present Curricular Integration Work had the objective of designing and implementing a 3D reconstruction system of interior environments, which allows defining the trajectory of a teleoperated mobile robot, for which the mobile robot that contains a MYNT EYE S1030 camera was implemented to capture images that, through the stereo vision technique, to be able to reconstruct an indoor scenario, the operation of the system was determined by means of a randomness analysis. The reconstruction of environments and subsequent calculation of the trajectory was defined by stages, which were: Analysis, Design, Implementation, Evaluation and Maintenance of the system, where substages are defined for the reconstruction algorithm programmed in MATLAB. For the randomness and comparative analysis, it was decided to apply different statistical tests such as the Shapiro test, the F test and the Student's T test to evaluate the system, from which it was obtained that the system differs in measurements compared to the commercial tool COLMAP; In the analysis of accuracy with the real environment, it is found that the system differs, but on a smaller scale, making an absolute error calculation, giving an average error of 3.27 meters with respect to the real value, and finally, a reconstruction of the scenario was made. indoor by varying its parameters. It is concluded that the system gives acceptable and not highly reliable results due to the camera and the quality of the captured images, since the creation of the point clouds generates noise, so the ICP algorithm for joining the point clouds overlaps the point clouds that gradually lose the real dimensions of the stage. It is recommended to complement the stereo vision technique with depth sensors to obtain values close to the real ones and thus improve the reconstruction of environments.

Keywords: <STEREO VISION>, <3D RECONSTRUCTION>, <MOBILE ROBOT>, <MATLAB (SOFTWARE)>, <COLMAP (SOFTWARE)>, <CALIBRATION>, <ITERATI CLOSEST POINT (ICP)>, <ICP ALGORITHM>.

INTRODUCCIÓN

Gracias al avance tecnológico hemos logrado realizar procesos cada vez más complejos, como lo es la reconstrucción 3D, la cual nos permite recrear objetos reales o escenarios de forma virtual y mostrarlos a través de una computadora manteniendo sus características físicas. Este proceso lleva tiempos en estudio por los que existen diversos métodos para hacer la reconstrucción 3D con el fin de llegar a ser lo más fiel al objeto o escenario real para luego darle un propósito al escenario virtual, como lo es la exploración espacial o la tele operación en zonas de alto riesgo humano.

El fin de este Trabajo de Integración Curricular es la reconstrucción de escenarios en 3 dimensiones con la fidelidad lo más cercanas posibles al escenario real. Está fidelidad depende de las características de las cámaras estereoscópicas al capturar datos en los escenarios de ambiente controlado, así como del software para procesar las imágenes. La reconstrucción de escenarios 3D empieza con la obtención de datos del escenario real, para lo cual, una manera de obtener estos datos es el del uso de imágenes capturadas por cámaras estéreo debidamente calibradas sobre un robot móvil, llamando a este proceso como visión estéreo, se continua mediante la rectificación de estas imágenes, creación de mapas de disparidad, creación de nubes de puntos, unión de nubes de puntos y cálculo de trayectoria, para finalmente, analizar el funcionamiento del sistema.

CAPITULO I

1 MARCO REFERENCIAL

1.1 Antecedentes

La forma de percibir una imagen se ha modificado con respecto al comienzo de nuestra sociedad, esto debido al surgimiento de la imagen digital. Es completamente distinto a como se la conocía debido a que ha adquirido una independencia. Se la puede modificar sin alterar sus resultados físicos, mantiene su originalidad, aunque se le haya hecho variedad de modificaciones, y cada una se vuelve independiente de la otra. Es distinta su forma de captura. No necesita de una impresión para poder observarla ya que es un archivo electrónico que se puede almacenar en una computadora o una unidad de almacenamiento, es decir, sin la necesidad de cobrar su forma material en una hoja (Luna 2003, p. 4).

La visión por computadora abarca varias técnicas de reconstrucción, una de ellas la técnica de reconstrucción tridimensional mediante imágenes y se lo ha estudiado por una gran cantidad de tiempo. Así se puede observar ya que Lawrence G. Roberts, se registra como la primera publicación de reconstrucción de objetos en 3D en base a imágenes 2D (Roberts 1963, p. 14), ahí define, de manera detallada y mecanizada, las primeras reglas y supuestos de percepción de profundidad. Este trabajo, puede nombrarse el comienzo de la visión por computadora y que lo llevó a obtener su título de doctorado.

(Vila Ubieto 2009, p. 10), entrega una propuesta para la reconstrucción de objetos sencillos, en base de imágenes obtenidas a partir de 2 cámaras en forma de un arreglo paralelo; dicha tarea la realizó desarrollando un algoritmo en Matlab. Por otra parte, este algoritmo reconstruye objetos, de una forma adecuada, que contengan una geometría recta. Dicho sistema no está desarrollado para objetos con una geometría redonda.

(Carabias et al. 2010, p. 36), entrega una propuesta de desarrollo de una aplicación la cual obtenga la reconstrucción en 3D del entorno por el cual navega el robot móvil autónomo, en base a 2 imágenes tomadas por diferentes cámaras, y en base de la información recolectada por estas, planear una ruta del robot. Este algoritmo desarrollado por Carabias tenía algunos problemas, debido al momento de reconstruir el entorno, esta se hallaba desplazada con respecto al entorno original, pero el algoritmo reconoce si hay o no obstáculos y en base a esa información planear la navegación del robot móvil.

1.2 Formulación del problema

¿De qué manera diseñar un sistema de reconstrucción ambiental 3D indoor que permita determinar la trayectoria de un robot móvil mediante análisis y procesamiento de imágenes?

1.3 Sistematización del problema

¿Cómo revisar y elaborar un estado del arte sobre los métodos y tecnologías de reconstrucción 3D mediante la técnica de visión estereoscópica?

¿Mediante qué parámetros realizar la clasificación de robots móviles para la posterior selección de un robot móvil?

¿Cuál es el mejor diseño de una composición cámaras sobre el robot móvil que permitan obtener una visión de 360 grados al escenario indoor a reconstruir?

¿De qué manera implementar un prototipo adaptable al diseño para permitir obtener las imágenes en un escenario indoor a tiempos discretos de posición del robot móvil?

¿Qué parámetros de operación son los correctos en las cámaras estereográficas que nos permitan obtener las imágenes adecuadas?

¿Cómo realizar el procesamiento digital de las imágenes obtenidas por el robot móvil para la reconstrucción 3D del escenario y trazar la ruta recorrida del mismo?

¿Mediante que métodos evaluar el correcto funcionamiento del sistema de reconstrucción 3D de escenarios indoor?

1.4 Justificación teórica

La variedad de aplicaciones en diversas áreas de sistemas de reconstrucción en tres dimensiones va creciendo, por ejemplo: En ingeniería, la alteración de productos, ingeniería inversa, medición sin entrar en contacto de cualquier objeto; para la creación de prótesis en la medicina o tomografías axiales computarizas en tres dimensiones; para el análisis de piezas muy frágiles en la arqueología; aplicaciones para mejora de la ergonomía como el diseño de calzado a medida, el modelado 3D para tiendas en línea o entrenamiento. Dependiendo la aplicación, es de gran importancia entender la teoría que se encuentra detrás, así como optar por herramientas que ayuden a la crear nuevas aplicaciones que faciliten resolver problemas o entenderlas de mejor manera (Vargas 2020, p. 14).

Debido a la importancia que representan estos sistemas, es obligada el desarrollo de estos que puedan se asequibles a bajos costos y sirvan de apoyo a poblaciones de bajos recursos.

Existen varias técnicas de reconstrucción tridimensional, varias de las cuales se nombran a continuación.

- Técnica de visión artificial activa: esta técnica emplea la obtención de varias posiciones de un objeto mediante una cámara de video móvil para el objeto a reconstruir. La cámara se mantiene en constante movimiento, eso debido a mantener correspondencia entre el plano del objeto y de la cámara.
- Telemetría Laser: esta técnica consiste un objeto envuelto por un royo luminoso. Mediante métodos de triangulación o de medidas de los tiempos de vuelo del rayo en varios puntos, se puede estimar las profundidades del objeto para posteriormente realizar su reconstrucción 3D.
- Técnicas de luz estructurada: estas técnicas son las más aplicadas en la industria; permitiendo reconstruir un objeto en 3 dimensiones, a través del análisis de la deformación que sufre el rayo luminoso que se emitió hacia el entorno u objeto a reconstruir. Esta técnica requiere la obtención de imágenes que contengan la deformación de los rayos emitidos hacia el objeto mediante una o varias cámaras.
- Técnica de visión artificial Multi-Vistas: Esta técnica se basa en la obtención de la información tridimensional mediante la existencia de 2 o más perspectivas del mismo punto, se lo realiza de esta manera, debido a que el procesamiento digital de estas imágenes hace que se pierda el eje asociado a la profundidad del punto capturado.

Las técnicas de visión Multi-Vistas, según la cantidad de imágenes, se clasifican en:

- Visión artificial estereoscópica (2 cámaras)
- Visión artificial Trifocal (3 cámaras).
- Visión artificial n-focal (n cámaras).

En la tabla 1-1 se muestra las diferentes ventajas y desventajas que presenta cada técnica de reconstrucción tridimensional al ser aplicadas en la navegación de objetos móviles.

Tabla 1-1: Comparativa entre las diferentes técnicas de reconstrucción 3D

	VENTAJAS	DESVENTAJAS
Visión Estereoscópica	• Construye el entorno rápidamente. • Económico computacionalmente	• Correspondencia entre pixeles de 2 imágenes
Visión activa	• Extrae la información 3D de un objeto en movimiento. • Económico	• Los Parámetros de calibración cambian. • Las cámaras deben moverse adecuadamente en el entorno
Telemetría Laser	• Preciso	• Requiere ambientes de luz controlada y de la geometría del objeto
Luz Estructurada	• Preciso • Construye el entorno rápidamente y con exactitud.	• Requiere ambientes de luz controlada y de la geometría del objeto. • Costoso computacionalmente

Fuente: (Narváez y Herrera 2015, p. 14).
Realizado por: Narváez y Herrera, 2015

Ilustración 1-1: Etapas para obtener un entorno 3D de un entorno indoor, escenario real y escenario reconstruido.

Fuente: (Vargas 2020, p. 25)

En la Ilustración 1-1 se puede observar las etapas iniciales del escenario real para la adquisición de imágenes y reconstrucción tridimensional de este entorno mediante el procesamiento digital de imágenes y alteración de tonalidades.

Hablando de experimentos basados en robótica móvil, estos requieren un especial interés, debido a que representan a sistemas reales difíciles, que agrupan diferentes conceptos. Para esto se debe enfocarse en la realización de varios experimentos de manera remota que conllevan a elaborar un laboratorio remoto para un robot móvil.

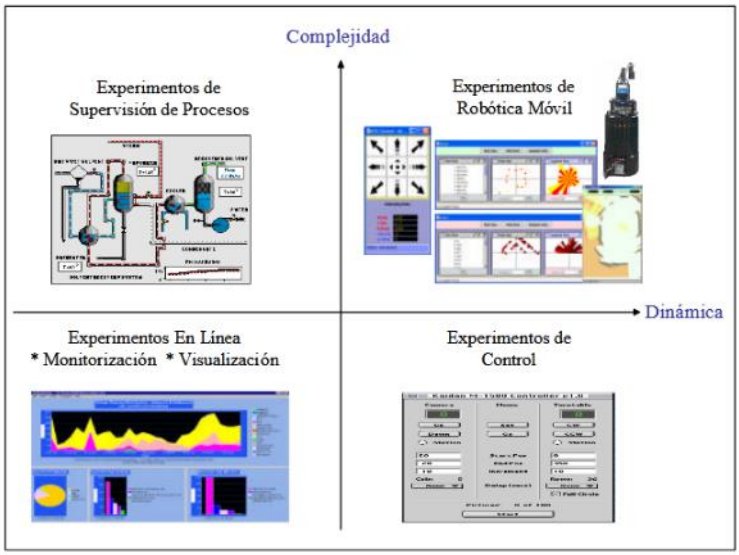


Ilustración 2-1: Tipos de experimentos remotos

Fuente: Montalvo Martínez, 2016

Como se muestra en la Ilustración 2-1, los experimentos remotos se pueden clasificar en los siguientes tipos:

- Experimentos como monitorización o visualización. Estos son los más fáciles de implementar, debido a que son experimentos con poca dinámica y escasa complejidad.
- Experimentos de control de procesos, que, aun siendo bajos en lo que a una de una dinámica importante se refiere, son bastante más complejos que los anteriores por la cantidad de variables.
- Experimentos de control. Este debe tener un buen estudio del retraso temporal que impone el internet, ya que esta define una alta dinámica que define su dificultad. Se debe hacer en tiempo real el muestreo de variables de salida para posteriormente hacer, en tiempo real, el control de las variables de entrada para no desestabilizar el sistema.
- Experimentos en robótica móvil. Estos son más difíciles de desarrollar, debido a su alta dinámica que genera una alta complejidad debido a la gran cantidad de variables a controlar. Estos robots móviles se los adapta a un gran número de sensores y actuadores solo los cuales poder actuar.

A partir de la información anteriormente expuesta, se propone elegir la técnica de visión artificial estereoscópica, como la más adecuada para la reconstrucción de entornos 3D; esto debido a las diversas ventajas que esta tiene frente a las demás como los son: rápida construcción del entorno, la funcionabilidad de la técnica tanto en ambiente controlados como

nos controlados, el bajo costo computacional de sus algoritmos y finalmente su alta precisión en la medición de coordenadas de puntos 3D sobre un robot móvil

1.5 Justificación aplicativa

Al momento de diseñar la presente propuesta, esta usa un método de calibración de cámaras y la información para crear la vista estereoscópica, esta información vendrá de múltiples imágenes captadas por las cámaras necesarias proporcionando así un aumento de la información obtenida. La presente propuesta de tesis se enfoca en la reconstrucción en tercera dimensión de un entorno indoor mediante imágenes estereográficas tomadas en traslación por el robot móvil para luego determinar su trayectoria sobre el escenario mapeado haciendo uso del color y/o la textura de este siempre que sea posible. Esta diversidad de imágenes y múltiples vistas permite una reconstrucción más completa del entorno.

Para cumplir con lo antes mencionado, el primer paso es la Investigación de los diferentes tipos de robots móviles disponibles en el mercado donde se pueda clasificar estos dependiendo las necesidades, para luego hacer la selección de un robot móvil a control remoto que disponga de las partes principales del robot y que permitan diseñar la estructura que contenga a las cámaras, tal es el caso del robot de chasis de tanque inteligente presentado en la Figura 3-1.

Ilustración 3-1: Chasis de tanque inteligente para coche con Control

Fuente: (Aliexpress, 2021)

Después de seleccionado el robot móvil se procede a diseñar un sistema de reconstrucción de entornos indoor en 3D por medio de la utilización de al menos 2 cámaras que forman parte de la cámara estéreo Mynt eye s1030, las cuales se disponen en la facultad di informática y electrónica, que son capaces de capturar imágenes al mismo tiempo por medio de sus dos lentes, con una resolución de 752x480 a 60 FPS se pueden obtener imágenes claras incluso en

ambientes de poca luz. El Angulo de visión de estas cámaras en cada eje es de 146, 122 y 76 grados horizontal y vertical lo que permite tener un amplio enfoque en objetos pequeños y medianos, además de su gran alcance desde el medio metro hasta las18 metros. Estas cámaras se encontrarán en el robot móvil de manera que se pueda obtener una visión de 360 grados alrededor del objeto para así determinar parámetros como profundidad y distancia desde el objeto móvil hacia los objetos que conforman el entorno indoor, entre otros parámetros geométricos. Queda de manifiesto la utilización de más cámaras según sea necesario.

Una vez realizado el diseño se analizará las imágenes obtenidas mediante el arreglo para configurar los parámetros de operación de las cámaras como resolución, claridad, calibración, triangulación y otros parámetros que permitan la captura de imágenes de excelente calidad.

De las imágenes obtenidas con los parámetros óptimos de operación, estás imágenes digitales capturadas se procesan por medio de un algoritmo que se acople al mapeo de escenarios y a los recursos disponibles por el diseñador, entre estos tenemos tanto las herramientas de matlab, OPENCV o la del propio fabricante Mynt de las cámaras, con el propósito de calcular el mapa de disparidad entre pares estéreo, con esto se obtendrán la profundidad de los objetos escaneados, también el de generar nubes de puntos a partir de mapas de disparidad, mediante este proceso se unirán todos los mapas de disparidad provenientes del entorno a reconstruir, se segmentará los mapas tomando lo que se necesita en base al escenario. Para finalizar, se unirán las nubes de puntos para crear un modelo en tercera dimensión del escenario y trazar la ruta seguida. Todo este proceso se ve involucrado en las diferentes etapas del procesamiento de imágenes como se observan en la Figura 4. En base a los resultados obtenidos se podrá proponer un análisis de visión en tiempo real mediante visión artificial para el robot en movimiento.

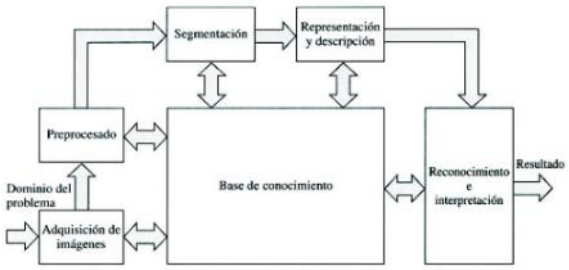


Ilustración 4-1: Etapas involucradas en el procesamiento de imágenes

Fuente: (Vargas 2020, p. 28)

Posteriormente se dispondrá a implementar un prototipo del sistema de reconstrucción de entornos 3D de entornos indoor mediante la unión de procesos antes mencionados. Para la implementación ya se debe contar con el modelo del robot móvil a acoplar la cámara.

Finalmente, se evaluará el correcto funcionamiento del sistema de reconstrucción de entornos 3D indoor, de esta manera se pondrá a prueba para determinar la funcionalidad del diseño y para este objetivo se evaluará el sistema mediante una comparativa del escenario reconstruido variando parámetros de luz, complejidad y ruta del robot móvil, también se evaluará mediante un análisis estadístico de aleatoriedad para analizar los resultados de la reconstrucción bajo los mismos parámetros.

La estereoscopia tiene muchas aplicaciones y la reconstrucción de ambientes es una de ellas, con esta podremos determinar la forma virtual un objeto, entorno de una manera más rápida, ahorrando tiempo y recursos humanos.

1.6 OBJETIVOS

1.6.1 Objetivo general

Realizar el diseño de un sistema de reconstrucción ambiental 3D indoor que permita determinar la trayectoria de un robot móvil mediante análisis y procesamiento de imágenes.

1.6.2 Objetivos específicos

- Revisar y elaborar un estado del arte sobre los métodos y tecnologías de reconstrucción 3D mediante la técnica de visión estereoscópica.
- Investigar la clasificación de robots móviles para la posterior selección de un robot móvil adecuado.
- Diseñar una composición de cámaras sobre el robot móvil que permitan obtener una visión de 360 grados al escenario indoor a reconstruir.
- Implementar un prototipo adaptable al diseño para permitir obtener imágenes en un escenario indoor a tiempos discretos de posición del robot móvil.
- Analizar las imágenes estereográficas obtenidas hasta encontrar los parámetros óptimos que permitan obtener las imágenes adecuadas.
- Realizar el procesamiento digital de las imágenes obtenidas por el robot móvil para la reconstrucción 3D del escenario y trazar la ruta recorrida del mismo.
- Evaluar el correcto funcionamiento del sistema de reconstrucción 3D mediante un análisis comparativo y un análisis de aleatoriedad sobre el escenario.

CAPITULO II

2 MARCO TEORICO

2.1 Estado del arte

En el caso (Córdova et al. 2020) se desarrolló un sistema de reconstrucción 3D catadióptrico estéreo, compuesta por dos cámaras catadióptricas, capaz de generar reconstrucciones semidensas basadas en DeepMatching con restricciones epipolares. El método genera reconstrucciones 3D precisas para entornos interiores y exteriores. En desventaja con respecto al presente trabajo consiste en la portabilidad ya que su gran tamaño y fragilidad, lo que lo hace inadecuado para situaciones de la vida real. Por otro lado, una ventaja del sistema es que DeepMatching proporciona significativamente más puntos característicos que los detectores de esquinas o de puntos característicos como el algoritmo Harris–Stephens usado en este proyecto.

Por otro lado, (Huang et al. 2018) presentan un sistema llamado DeepMVS, que se define como una red neuronal convolucional profunda (ConvNet) para la reconstrucción estéreo multivista. Tomando una cantidad arbitraria de imágenes como entrada, primero producen un conjunto de volúmenes de barrido plano y usan la red DeepMVS propuesta para predecir mapas de disparidad de alta calidad. Una desventaja con respecto al proyecto presente es que, en términos de cálculo computacional, usan GPU´s de alta capacidad de procesamiento como los donados directamente por NVIDIA por que el sistema no es de fácil acceso, en términos de costos, con respecto al presente proyecto. La ventaja es sus mapas de disparidad son de alta calidad lo que significa una reconstrucción igualmente de alta calidad y con pocas perdidas en las características del entorno.

Siguiendo, (Jung et al. 2022) proponen un sistema de reconstrucción 3D mediante el uso de una cámara estéreo y un sensor de tiempo de vuelo (ToF) mediante las imágenes estéreo obtienen información de color y profundidad del entorno en reconstrucción y mediante el sensor ToF generan más mapas de profundidad con valores exactor del escenario real para mejorar los resultados de profundidad y así mejorar los resultados al reconstruir el escenario, uniendo de mejor manera estas nubes de puntos. Una clara ventaja del sistema mencionado y el sistema propuesto en el presente documento es el uso del sensor ToF que nos brinda, lo llamado *ground truth* que proporciona datos exactos del mundo real y por lo cual sus resultados mejoran en gran medida, la desventaja va en términos de costos, ya que esos sensores tienen un alto precio en el mercado convencional.

Finalmente, (Jin, Ko & Lee 2018) describen la reconstrucción de entornos indoor en 3D mediante el uso de una sola cámara estéreo montado en un Dron y el cálculo de la posición del dron mediante el uso solo de las imágenes capturadas por el dron, esto lo hacen usando los algoritmos SGM para el cálculo de los mapas de profundidad y algoritmo IGP para la unión de la nube de puntos.

2.2 Clasificación de robots móviles

Los robots móviles se definen como un sistema distribuido compuesto por un equipo de entidades computacionales móviles, que de forma básica se dividen en 3 partes, hardware donde se definen las funciones de este robot, sistema de movimiento (sistema de llantas, sistema de piernas, etc.) y un chasis donde se ensamblan estos subsistemas. Estos robots se pueden clasificar en robots móviles autónomos y robots móviles no autónomos.

2.2.1 Robots Móviles Autónomos

Estos tipos de robots móviles son capaces ejercer un tipo de trabajo específico o varios, dependiendo del tipo de asistencia que tengan, estas asistencias pueden ser robótica que solo puede ejercer un trabajo ya programado con anterioridad o tener una asistencia mediante inteligencia artificial, lo cual le permite tomar decisiones en base a al ambiente que los rodea o se le instruya a hacer.

Uno de los robots verdaderamente autónomos más conocidos y posiblemente más prolíficos del mercado actual es el Roomba. Si bien Roomba es un producto de consumo, sus capacidades y funcionalidades se trasladan al almacén y al espacio industrial y han hecho que los AMR sean una tecnología mucho más accesible. El Roomba puede tomar decisiones y actuar en base a lo que percibe en su entorno. Se puede colocar en una habitación, solo, y hará su trabajo sin ayuda ni supervisión de una persona.

Ilustración 1-2: Robot Roomba

Fuente: (irobot, 2020)

Las ventajas que estos tipos de robots pueden tener son:

- Se encuentran activos por el tiempo que sus baterías se lo permitan y sin la necesidad de supervisión humana.
- Incrementan la productividad según la tarea que realicen.
- Reducen el tiempo en que se realizan sus procesos.
- Tienen alta flexibilidad y son capaces de adaptarse al entorno.

Las desventajas que presentan son:

- Pueden ser hackeados por medio de un virus que los puede dejar fuera de operación e interrupción de la productividad.
- Comportamientos erróneos comprometen la seguridad del personal involucrado en la operación.
- Si se equivoca aplicando el software se puede producir una desventaja grave.

2.2.2 Robots móviles no autónomos

Estos robots tienen la característica y desventaja de depender de personal humano para su funcionamiento, pero tiene la ventaja de poder agregarle módulos para un manejo más sencillo y de gran comodidad, como lo puede ser el manejo inalámbrico, esto se lo puede hacer mediante la agregación de módulos WI-FI o Bluetooth.

Estos robots son más simples y sencillos de implementar, ya que depende de módulos para funcionar y con la facilidad de acceder a un microprocesador como lo es Arduino, se puede llegar a programar de forma rápida. Ejemplo de esto tenemos los robots para trabajar a tele operación, ya que se los controla a distancia.

2.3 Visión estereoscópica

El procesamiento de imágenes en los sistemas informáticos digitales suele considerar la información visual como una secuencia de fotogramas. Estos fotogramas son de cámaras que capturan la realidad durante un corto período de tiempo.

El procesamiento de video digital tiene que procesar cada cuadro para obtener un resultado de filtro o detectar una característica en la entrada. La visión artificial clásica comenzó utilizando una única cámara como sensor del sistema para realizar un tratamiento de cada uno de los fotogramas obtenidos por esa cámara. Este método proporcionó un entorno controlado, pero

carece de ciertos aspectos de la visión humana, como la visión 3D, el cálculo de distancias, las trayectorias, etc.

Hoy en día, la humanidad ha experimentado un gran avance en el campo de la visión artificial. Este avance está relacionado con la introducción de un mayor número de cámaras en escena. Intentando imitar la visión humana, los investigadores suelen trabajar con un sistema de dos cámaras, llamado sistema de visión estéreo. En la visión estéreo, los algoritmos existentes utilizan fotogramas de dos cámaras digitales y los procesan. (Bhatti ,2008 p. 14)

2.3.1 Visión artificial clásica

El objetivo de la visión por computadora es procesar imágenes adquiridas con cámaras para producir una representación de objetos en el mundo. Ya existe una serie de sistemas de trabajo que realizan partes de esta tarea en dominios especializados. Por ejemplo, un mapa de una ciudad o una cadena montañosa se puede producir de forma semiautomática a partir de un conjunto de imágenes aéreas. Un robot puede usar los varios cuadros de imagen por segundo producidos por una o dos cámaras de video para producir un mapa de su entorno para la planificación de rutas y la evitación de obstáculos. Un sistema de inspección de circuito impreso puede tomar una imagen por placa en una cinta transportadora y producir una imagen binaria que marque posibles puntos de soldadura defectuosos en la placa.

Existen diferentes modelos con los que trabajar en visión artificial. En un principio, las investigaciones buscaron aplicaciones industriales utilizando un sistema de vista única con una sola cámara. Estos sistemas tienen muchas limitaciones debido a que solo tienen un punto de vista de la escena. Un avance importante fue implementar sistemas con múltiples puntos de vista: se pueden usar múltiples cámaras o una cámara en movimiento. Con esta modificación, las aplicaciones industriales experimentaron una gran mejora en su eficiencia: con múltiples cámaras se puede reconstruir un escenario en tres dimensiones y se pueden solucionar los errores anteriores que producía el uso de una sola cámara (sin conocimiento en profundidad). Sin embargo, el principal objetivo de los investigadores en esta área es tratar de imitar el comportamiento y la funcionalidad de la visión humana. Es por ello que en el área de la visión artificial hay una gran cantidad de investigadores trabajando con sistemas de visión estéreo, donde se utiliza un modelo de dos cámaras.

En visión artificial, el modelo de dos cámaras se inspira en el modelo biológico de la propia estereovisión, donde gracias a la distancia entre los ojos se puede estimar la profundidad. Esto corresponde a la tercera dimensión. El hecho de la distancia entre los dos ojos produce una

disparidad entre las visiones obtenidas de cada ojo. (Bhatti, 2008, p. 13) Según la Ilustración 2-2 hay un desfase entre la información de cada ojo. En resumen, los dos ojos ven la escena de manera similar, pero con cierto desplazamiento y este desplazamiento es inversamente proporcional a la distancia entre los ojos y el objeto en sí.

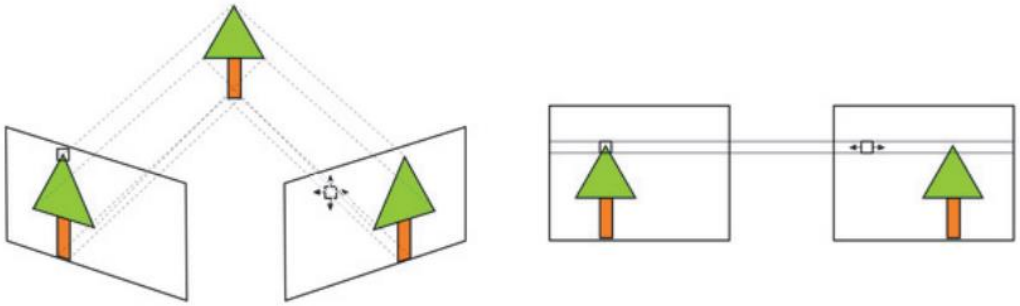


Ilustración 2-2: Disparidad en estéreo visión

Fuente: (Bhatti 2008, p. 14)

Otro aspecto inherente a los sistemas de visión estereoscópica es su geometría. Se puede elegir en función de la geometría de los ejes ópticos: paralelos o convergentes. El sistema visual humano funciona con ejes convergentes, por lo que los ojos se centran en los objetos de interés. Cuando el objeto está al lado, hay una convergencia de ejes sobre ese objeto. Por el contrario, si el objeto está situado a cierta distancia, casi no hay convergencia de ojos. En este caso es común suponer que los ejes ópticos están en forma paralela.

Cuando se utiliza un sistema de visión estereoscópica, dos de los pasos comunes en el proceso de video son la adquisición de imágenes y el modelado del sistema de cámara. (Bhatti, 2008, p. 15) Una descomposición más detallada del proceso de visión estereoscópica se puede ver en la Ilustración 3-2.

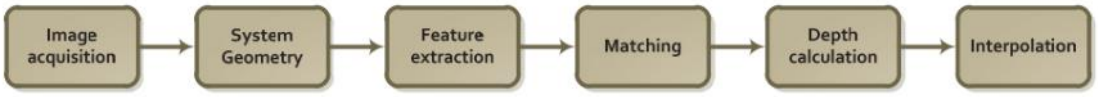


Ilustración 3-2: Pasos en el proceso de estéreo visión

Realizado por: (Bhatti 2008, p. 14)

2.3.1.1 Adquisición de imagen

Este paso se lo puede hacer de varias maneras diferentes. El par de imágenes estéreo se los puede tomar de manera simultánea (al mismo tiempo) o tomadas mediante un intervalo de tiempo fijo entre imágenes. Se debe tener en cuenta el momento del día, las condiciones atmosféricas u otra variable que haya cambiado la escena considerada a reconstruir, si se hace la toma de imágenes en intervalos de tiempos grandes. Un factor que se considera importante para

la toma de imágenes es la aplicación que tiene el sistema. Es diferente un sistema cartográfico mediante un vehículo aéreo que toma imágenes, usualmente de baja resolución, mediante una única cámara solo varios tipos de terrenos, a un sistema de vehículo terrestre autónomo en la que las imágenes capturadas deben ser de mayor resolución si su objetivo es el evitar obstáculos, donde el objetivo principal es la identificación de objetos más que regiones y, por lo general, suelen tener dos cámaras para tomar imágenes simultáneamente. Tipicamente se determina primero la aplicación del sistema para definir el tipo de escena sobre la que se trabaja. Estas escenas se pueden clasificar en 2 grupos: Escenas que contienen elementos los cuales fueron realizados por el ser humano, como carreteras, edificios; y escenas naturales con superficies, como los son terrenos, lisos u ondulados, montañas, agua y follaje (Martínez, 2010, p. 13).

2.3.1.2 Geometría del sistema

Una geometría del sistema es una representación de los atributos geométricos y físicos más importantes de las cámaras utilizadas para la visión estéreo. Este modelo puede tener una componente relativa, la cual relaciona el sistema de coordenadas de una cámara con el de la otra, y es independiente de la escena, y también puede tener una componente absoluta, la cual relaciona el sistema de coordenadas de una de las cámaras con un sistema de coordenadas fijo de la escena. El modelo más usado es uno con dos cámaras que tienen sus ejes ópticos paralelos, siendo la distancia que los separa la línea base. Quedando sus ejes ópticos perpendiculares a la línea base, y sus líneas de exploración o líneas epipolares paralelas a la línea base. Las líneas epipolares son líneas que unen la imagen izquierda y la imagen derecha de un mismo punto. Cualquier punto del espacio tridimensional unido a los dos centros de proyección de las cámaras define un plano, llamado plano epipolar. La intersección de un plano epipolar con el plano de proyección de una cámara define una línea epipolar. Para todos los puntos, cuyas proyecciones izquierdas estén contenidas en una misma línea epipolar en la imagen izquierda, sus proyecciones derechas deben estar también contenidas sobre una misma línea epipolar en la imagen derecha, y viceversa. Como se ha dicho anteriormente, el principal problema al realizar visión estereoscópica es encontrar la correspondencia entre los puntos de las imágenes. Al realizar esta correspondencia se debe efectuar una búsqueda en dos dimensiones, tanto en el eje X como en el eje Y. Cada punto de la imagen derecha al que se quiere emparejar con un punto de la imagen izquierda, se selecciona un conjunto de vecinos (vecindario) en torno al píxel con las mismas coordenadas (sistema de referencia relativo de las cámaras) en la imagen izquierda. Los píxeles que constituyen este vecindario tienen coordenadas que difieren del píxel de la imagen derecha tanto en la componente X como en la componente Y. Para reducir la complejidad de este proceso se puede llegar a una búsqueda unidimensional, reduciéndose su vecindario a una única dimensión. Para conseguir esta reducción en la complejidad

computacional, basta con situar y orientar las cámaras de forma que sólo exista un desplazamiento horizontal entre ellas, los ejes ópticos deben ser paralelos y los ejes de abscisas (eje X) de cada una de las cámaras deben de ser coincidentes como sucede en la Ilustración 4-2. Al conseguir esta distribución y orientación de las cámaras, éstas sólo tendrán desplazamiento en sentido horizontal, y se dice que las imágenes están alineadas horizontalmente. Bajo esta situación, las líneas epipolares que definen el plano epipolar son coincidentes, consiguiendo simplificar la búsqueda del emparejamiento a recorrer las imágenes por filas. Con esta geometría se obtiene la denominada restricción epipolar, que ayuda a limitar el espacio de búsqueda de correspondencias, de manera que en el sistema de ejes paralelos convencional todos los planos epipolares originan líneas horizontales al cortarse con los planos de las imágenes. En un sistema con la geometría anterior se obtiene un valor de disparidad d, para cada par de puntos emparejados $P_I(X_I, Y_I)$ y $P_D(X_D, Y_D)$ dado por $d = X_I – X_D$. Con el valor de disparidad para cada punto de la imagen se construye un matriz o mapa de disparidad, en el que cada punto de la imagen contiene su valor de disparidad. A partir de este mapa de disparidad se logrará construir el mapa de profundidades, que es el objetivo final de todo sistema estereoscópico. (Martínez, 2010, p. 15)

En la Ilustración 4-2, puede ver todos los elementos antes mencionados. Hay tres sistemas de referencia, uno absoluto y dos asociados a la cámara. El sistema absoluto, que pertenece a la escena real 3D observada, la proporciona el origen O y los ejes {X, Y, Z}. El sistema relativo de la cámara izquierda está dado por el O_I original y los ejes $\{X_I, Y_I, Z_I\}$ y el cuadro que falta es el cuadro de la cámara derecha, donde su origen es O_D y los ejes $\{X_D, Y_D, Z_D\}$.

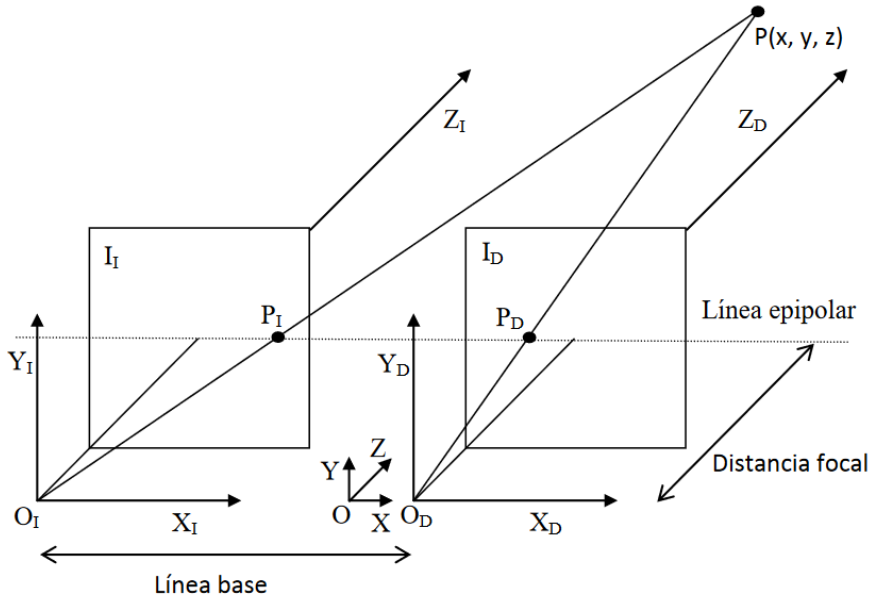


Ilustración 4-2: Correspondencia de un sistema estero

Fuente: (Martínez 2010, p. 24)

A pesar de la teoría expuesta anteriormente, en la realidad muy pocas veces las imágenes que forman el par estereoscópico se encuentran alineadas horizontalmente. Además, en el proceso de adquisición se introducen deformaciones en las imágenes que degradarán la precisión en la medida de profundidad a menos que sean corregidas. Dos ejemplos de estas deformaciones son la distorsión radial y la distorsión tangencial. Necesitándose en este caso realizar un proceso previo de calibración de cámaras con el fin de corregir dichas anomalías. (Pajares y De la Cruz, 2008, p. 26)

2.3.1.3 Extracción de características

En el paso de extracción de características se obtienen elementos identificativos de la imagen. De estos elementos, a su vez se tienen que extraer algunos atributos, los cuales se utilizarán en el siguiente paso, correspondencia de características. Por lo tanto, este paso está muy ligado al de correspondencia y, al ser el paso de correspondencia el más importante de todos, primero se suele decidir qué método utilizar al realizar la correspondencia entre imágenes y según las características que se empleen, éstas serán las que se extraigan de las imágenes. Existen dos clases de técnicas para determinar la correspondencia entre dos imágenes estereoscópicas: las técnicas basadas en el área ("area-based") y las técnicas basadas en las características ("feature-based"). (Martínez, 2010, p. 17)

En las técnicas de estereovisión basadas en el área se busca correlación cruzada entre patrones de intensidad en la vecindad local o vecindario de un píxel en una imagen, con patrones también de intensidad en una vecindad correspondiente a un píxel en la otra imagen del par estereoscópico. Por tanto, las técnicas basadas en el área utilizan la intensidad de los píxeles como característica esencial.

Para las técnicas de estereovisión basadas en las características se toman representaciones simbólicas obtenidas de las imágenes de intensidad en vez de utilizar las intensidades directamente. Algunas de las características cuyo uso está más extendido son: puntos de borde aislados, cadenas de puntos de bordes o regiones delimitadas por bordes.

Aparte de los bordes, las regiones son otra de las primitivas que pueden utilizarse en el proceso de visión estereoscópica. Una región es una zona de la imagen, que habitualmente está asociada con una determinada superficie en la escena 3-D y delimitada por bordes.

2.3.1.4 Emparejamiento

En el paso de la correspondencia de imágenes, bien utilizando píxeles o características, se debe determinar para un punto del espacio tridimensional, cuál es su proyección en cada imagen del par estereoscópico. Al comienzo del proceso de correspondencia ya se tienen los vectores con los atributos de las características consideradas. Con estos vectores, comparando los valores que toman sus atributos se debe establecer una correspondencia local entre características. Esta correspondencia se determina mediante alguna métrica que proporcione cuál es el grado de similitud para dos vectores de atributos, como por ejemplo la distancia Euclídea, la distancia de Mahalanobis u otras distancias. Tras realizar esta correspondencia local, se debe comprobar su consistencia, para lo cual se comienza otro proceso de correspondencia, pero en este caso de naturaleza global. Ambos procesos de correspondencia utilizan propiedades de la realidad física, que son formuladas en términos de restricciones. (Pajares y De la Cruz 2008, p. 30) Las restricciones utilizadas se enuncian a continuación:

- Epipolar: las imágenes de una misma entidad 3D deben proyectarse sobre la misma línea epipolar. Esta restricción se deriva de la geometría de las cámaras y requiere que las cámaras estén alineadas. (Reid y Beardsley 1996)
- Semejanza: las dos imágenes de la misma entidad 3D deben tener propiedades o atributos similares.
- Unicidad: para cada característica en una imagen debe haber una única característica en la otra imagen, salvo que se produzca una oclusión y no haya correspondencia de alguna característica.
- Orden posicional: dadas dos características en una determinada imagen, por ejemplo, la izquierda, situada una a la derecha de la otra, están restricción supone que este mismo orden se mantiene en la imagen derecha para sus respectivas características homólogas. Debido a la geometría también se debe verificar que, para una misma característica el valor de su coordenada x en la imagen derecha debe ser menor que su coordenada x en la imagen izquierda. Dado el par de imágenes estereoscópicas de la Figura 4-2, supongamos que el punto a y la estrella b de la imagen izquierda se corresponden con el punto c y la estrella d de la imagen derecha, como se tiene que el punto está más a la izquierda que la estrella en la imagen izquierda, $xa < xb$, también debe de estarlo en la imagen derecha, $xc < xd$. Además, la coordenada horizontal del punto y de la estrella en la imagen derecha debe de ser que la coordenada horizontal en la imagen izquierda, $xc < xa$ y $xd < xb$.

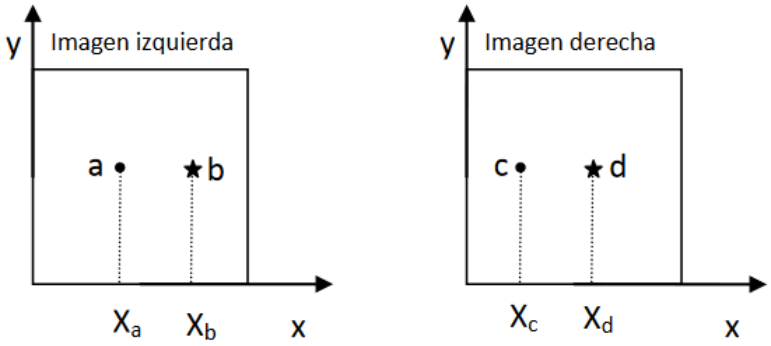


Ilustración 5-2: Restricción sobre el orden posiciona

Realizado por: (Martínez 2010, p. 29)

- Continuidad de la disparidad: asume que las variaciones de disparidad en la imagen son generalmente suaves, es decir que si consideramos un mapa de disparidad éste se presenta continuo salvo en unas pocas discontinuidades. Este principio también aparece bajo distintas formas y a veces con alguna pequeña variación, tal es el caso de Disparidad Diferencial Mínima en (Medioni y Nevatia 1985) o Continuidad "figural" en (Pollard, Mayhew y Frisby 1985).
- Relaciones estructurales: supone que los objetos están formados por aristas, vértices o superficies con una cierta estructura y una disposición geométrica entre dichos elementos

Como ya se ha expresado en varias ocasiones, la correspondencia es el paso más complejo del proceso de la visión estereoscópica. Al ser el paso más complicado es también el aspecto más estudiado y repetitivo en la literatura de la estereovisión. La dificultad para establecer la correspondencia entre puntos o características de un par de imágenes estereoscópicas proviene de la naturaleza del propio sistema. Las imágenes que forman el par estereoscópico son tomadas por un sistema de visión estéreo, donde la diferencia que existe entre la imagen izquierda y derecha es el ángulo o posición con las que fueron realizadas. Teóricamente esta es casi la única diferencia que se produciría, pero en la práctica las condiciones de iluminación pueden ser ligeramente diferentes o incluso existe la posibilidad de que aparezcan reflejos en una imagen y que la otra imagen esté ausente de ellos. A su vez se suele producir un fenómeno adicional que complica las cosas de manera importante, son las oclusiones, donde una característica de una imagen puede ocultarse en la otra del par estereoscópico. Además, los sistemas convencionales utilizan dos o más cámaras, dichas cámaras aun siendo de las mismas características técnicas presentan diferencias intrínsecas debido al distinto comportamiento de los componentes ópticos y electrónicos. (Martínez, 2010, p. 21)

Debido a los anteriores factores, un mismo punto y su vecindad en la escena 3-D puede proyectarse en cada una de las imágenes con diferentes valores de intensidad o incluso estar presente en una imagen y ausente en la otra imagen si se produce una oclusión. Esto es por lo que el proceso de correspondencia estereoscópica es el paso más difícil y complicado dentro del proceso de visión estereoscópica.

Se mencionó en el paso relativo a la "extracción de las características" que existen dos tipos de técnicas para realizar la correspondencia estereoscópica: las técnicas basadas en el área y las técnicas basadas en las características. Ambas técnicas tienen sus ventajas y desventajas, y dependiendo de cuáles sean las restricciones a las que nos enfrentamos se debe elegir una u otra técnica. Habrá ocasiones en las que la técnica a utilizar venga más o menos impuesta (por ejemplo, si en el entorno a estudiar se dispone de unos bordes perfectamente definidos, utilizando una técnica basada en las características se podría realizar la correspondencia con relativa facilidad) y, sin embargo, en otras ocasiones será difícil decantarse por unas u otras, o incluso se puede realizar una mezcla de las dos.

2.3.1.5 Cálculo de profundidad

Luego del proceso de emparejamiento, el sistema tiene las correspondencias entre los elementos que aparecen en una de las proyecciones con los elementos de la otra. Resuelto este problema, el cálculo de la profundidad es un problema relativamente fácil, que consiste únicamente en una simple triangulación. Sin embargo, en algunas ocasiones, la ejecución de este proceso revela algunas no correlaciones obtenidas de los resultados del paso anterior. Estos errores se deben a la falta de precisión o a la falta de fiabilidad de los resultados.

Gracias a las restricciones epipolares las proyecciones de un objeto tridimensional en ambas cámaras son bien conocidas si la geometría del sistema se ha definido correctamente en el segundo paso. Considerando una relación geométrica con semejanzas de triángulos, si dos proyecciones concretas reflejadas en cada cámara se relacionan con un mismo punto tridimensional (resuelto en el proceso de emparejamiento), se pueden calcular las coordenadas de este objeto en el espacio y, con ellas, la tercera coordenada (Z) es conocida por lo que la profundidad también. (Bhatti 2008, p. 16) Luego de este proceso, se obtiene un mapa de profundidad de la escena como se observa en la Ilustración 6-2.

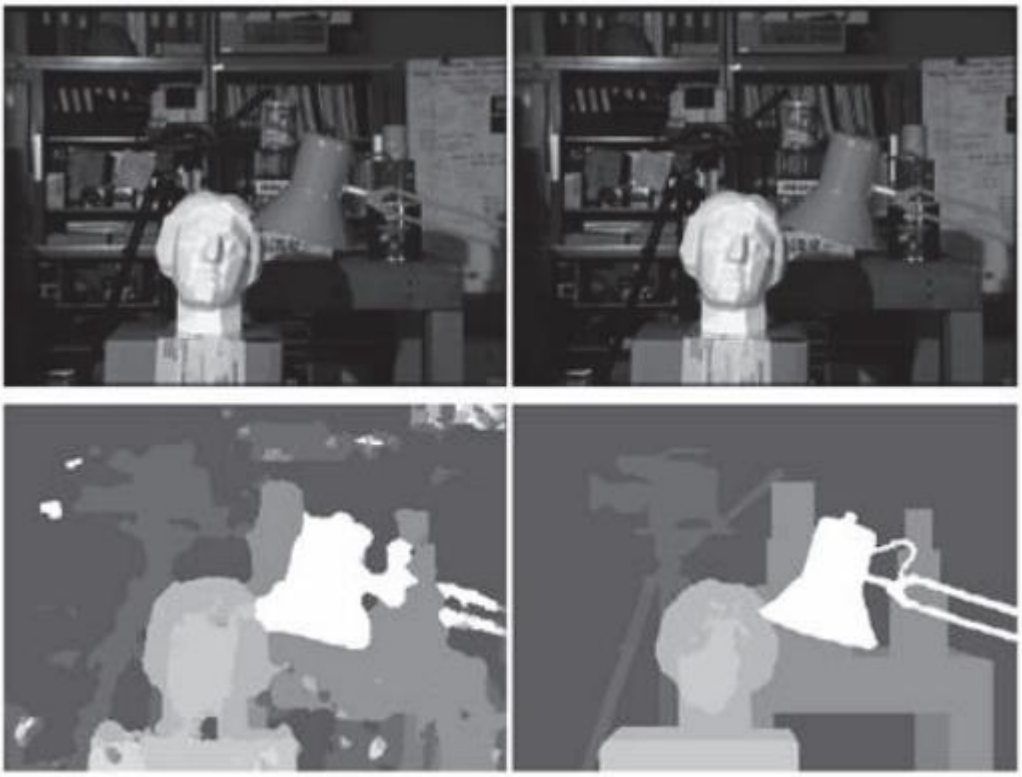


Ilustración 6-2: Mapa de disparidad y mapa de profundidad para un escenario estéreo concreto

Fuente: (Bhatti 2008, p. 16)

Considerando una relación geométrica de semejanza de triángulos, las coordenadas del punto de la escena P(X, Y, Z) pueden deducirse fácilmente observando la Figura 6-2, obteniendo los resultados dados por la ecuación (1-2). La Ilustración 7-2 es el mismo sistema y situación que se daba en el Figura 3-2, pero observado desde arriba. En este caso se denota la distancia de la línea base con la letra b y la distancia focal, idéntica en las dos cámaras, con la letra f. En la figura, P es un punto de la escena con coordenadas (x, y, z) según el sistema de coordenadas del mundo real; P_I es la proyección del punto P en la imagen izquierda y tiene por coordenadas (X_I,Y_I), según el sistema de coordenadas relativo de la cámara izquierda; y P_D es la proyección del punto P en la imagen derecha, que tiene por coordenadas (X_D,Y_D), según el sistema de coordenadas relativo de la cámara derecha. Con la letra d se representa la disparidad del punto, que es el desplazamiento horizontal que se producen, $d = X_I - X_D$.

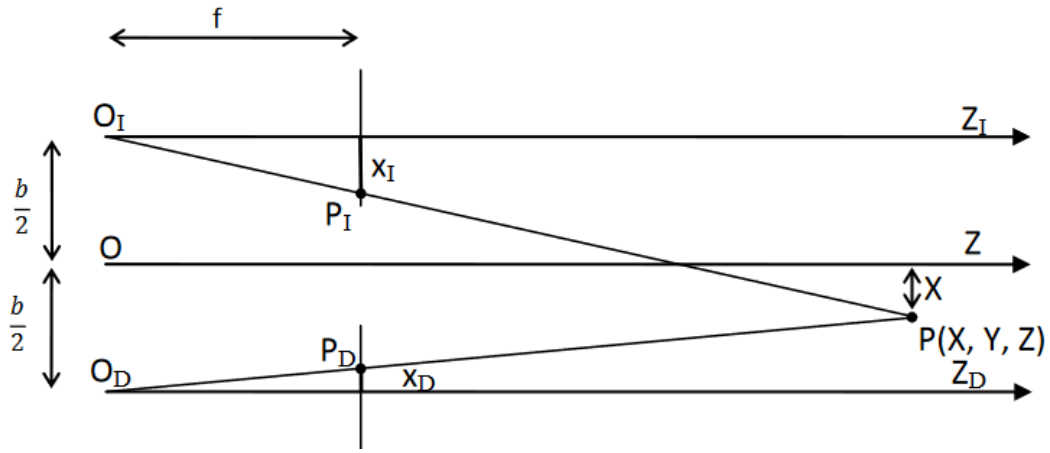


Ilustración 7-2: Geometría de dos cámaras en estéreo con ejes ópticos paralelos desde una perspectiva superior.

Fuente: (Martínez 2010, p. 36)

Se deduce a partir de la ecuación (1-2), cuando se utiliza esta geometría, la profundidad Z, es inversamente proporcional a la disparidad de la imagen y para una profundidad dada, a mayor b mayor d, lo que sugiere que la línea base puede incrementarse para mejorar la exactitud de la profundidad medida, pero ello lleva consigo el hecho de que ahora ambas imágenes tienen menos características comunes, es decir menos bordes o regiones procedentes de los objetos de la escena, debido a las desapariciones y oclusiones de las imágenes de dichos objetos. (Reid y Beardsley, 1996)

$$\text{Imagen Izquierda: } \frac{b/_2+x}{z} = \frac{x_I}{f} => x_I = \frac{f}{z}\left(x + \frac{b}{2}\right)$$

$$\text{Imagen Derecha: } \frac{b/_2+x}{z} = \frac{x_D}{f} => x_D = \frac{f}{z}\left(x + \frac{b}{2}\right) \qquad (1\text{-}2)$$

$$\boldsymbol{d = x_I - x_D = \frac{f.b}{d} => z = \frac{f.b}{d}}$$

2.3.1.6 Interpolación

Este paso no siempre se aplica, depende de los mecanismos utilizados en el resto de los pasos y del problema de aplicación que el sistema esté tratando de resolver, ya que en algunos casos los resultados obtenidos al final del proceso de cálculo de profundidad son suficientes (mapa de profundidad). En otros casos, los resultados muestran una gran cantidad de puntos tridimensionales con su correspondencia en ambas cámaras, pero para hacer un proceso de interpolación estos puntos no son suficientes. (Bhatti 2008, p. 16)

Uno de los métodos más sencillos que se utilizan para resolver el problema de la interpolación es la interpretación del mapa de disparidad obtenido en los pasos anteriores, como se observa en la Ilustración 6-2.

2.4 Cámara estéreo

Es una cámara que tiene dos lentes emparejados separados aproximadamente a la misma distancia que los ojos de una persona, de modo que dos imágenes que se verán en un estereoscopio o se proyectarán para dar una impresión estereoscópica se pueden tomar simultáneamente

2.4.1 Calibración de cámaras

La calibración de la cámara es un proceso de encontrar los parámetros intrínsecos y extrínsecos de una cámara o un subconjunto de estos.

Se ha dedicado especial interés al desarrollo de métodos rápidos de calibración para cámaras simples. Por ejemplo, un método permite la calibración de la cámara utilizando un patrón muy simple que se puede obtener de una impresora láser.

2.4.1.1 Métodos de calibración estándar

Los métodos de calibración clásicos se basan en patrones de calibración especialmente preparados, es decir, objetos con dimensiones y posiciones conocidas en un determinado sistema de coordenadas. Luego, las características, como esquinas, líneas, etc., se extraen de una imagen del patrón de calibración. Por lo general, los objetos de calibración se eligen para tener características prominentes, que son fáciles de localizar y medir sin ambigüedades de sus posiciones. (Cyganek y Siebert 2009, p.94) Un -simple tablero de ajedrez puede servir para este propósito, como se observa en la Ilustración 8-2.

Ilustración 8-2: Tablero de ajedrez como patrón de calibración de cámara

Fuente: (Cyganek y Siebert 2009, p. 94)

Existe un gran número de métodos de cálculo de los parámetros internos y externos de la cámara. La mayoría de ellos se basan en las fórmulas presentadas (2-2)–(3-2).

$$P_c = R(P_w - T) \tag{2-2}$$

Donde P_c expresa la ubicación de un punto P en el sistema de coordenadas de la cámara, P_w es su ubicación en el sistema de coordenadas externo, R representa la matriz de rotación y T es la matriz de traslación entre los orígenes de esos dos sistemas de coordenadas.

$$\gamma p = MP \tag{3-2}$$

Donde γ es un escalar, p es una imagen del punto P bajo la transformación M realizada por la cámara estenopeica.

2.4.1.2 Calibración fotométrica

La mayoría de los métodos de procesamiento de imágenes asumen la existencia de un fotosensor con una característica lineal de la señal de salida con respecto a la intensidad de la luz captada por el sensor. Los siguientes dos fenómenos relacionados con los sistemas de adquisición de imágenes reales requieren una calibración adecuada.

1. La corrección gamma: Las cámaras reales, aunque dotadas de dispositivos CCD bastante lineales, suelen contener el llamado circuito de corrección gamma. Se emplea para una representación de señal adecuada para una pantalla de salida (como la pantalla de un monitor). En este tipo de cámaras, para obtener una imagen sin distorsiones, es necesario realizar un proceso inverso a la corrección gamma. Sin embargo, esto es posible si se conocen de antemano los parámetros de esta corrección.
2. Polarización en condiciones de luz cero. El dispositivo CCD genera electrones incluso si todo el dispositivo está situado en una cámara totalmente oscura. Esta es una generación térmica que provoca una salida distinta de cero incluso sin luz incidente. El nivel de esta señal se denomina nivel inicial de negro. Aunque en la mayoría de las aplicaciones esto no provoca no linealidades, debe tenerse en cuenta al diseñar un método de procesamiento de imágenes.

En los sistemas de estereovisión se requiere una calibración fotométrica adicional que consiste en la ecualización del nivel medio de amplificación de las dos cámaras. Tal calibración debería

eliminar cualquier diferencia de intensidades en las imágenes estéreo de salida. De lo contrario, se puede esperar un aumento de las llamadas coincidencias falsas, especialmente si se utilizan métodos de coincidencia simples.

2.4.1.3 Auto calibración

Se ha dedicado mucha investigación a responder a la pregunta de si es posible calibrar una cámara únicamente a partir de secuencias de imágenes tomadas por la cámara. La solución a este problema, conocida como auto calibración de la cámara, permite determinar los parámetros intrínsecos de la cámara. Aunque no se trata de una calibración completa de la cámara, permite la reconstrucción de la escena hasta un cierto factor de escala lo cual es suficiente en muchas aplicaciones de visión artificial. Por lo tanto, los métodos de auto calibración de la cámara permiten el cálculo de los parámetros intrínsecos de la cámara en función de las coincidencias entre series de imágenes de la misma escena, tomadas por una sola cámara, pero con parámetros de visualización modificados, como la posición de la cámara (traslación, rotación o ambas), su distancia focal o una combinación de estos

2.4.2 MYNT EYE S1030-IR-120/Mono

La MYNTEYE S1030-IR-120/Mono es una cámara estéreo monocromática visible e infrarroja (IR), con una unidad de medición inercial (IMU) integrada de 6 ejes y un ángulo visual alto. Tiene un obturador global y las cámaras están sincronizadas por hardware. El fabricante no especifica si las mediciones de las cámaras están sincronizadas con la medición de la IMU y el disparador del hardware. El producto proporciona un kit de desarrollo de software (SDK) que se usó en el sistema operativo Ubuntu 16.04 para interactuar con la cámara y registrar datos usando los paquetes ROS proporcionados. Robot Operating System (ROS) es "un metasistema operativo de código abierto" para robots. Brinda los servicios que normalmente brinda un sistema operativo y un conjunto de herramientas para ejecutar código en múltiples máquinas. (Cappellaro, 2019, p. 9).

La cámara puede utilizada para la detección de profundidad precisa con un rango flexible entre 0,5 y 18 metros. Rendimiento optimizado en condiciones normales de luz o condiciones de poca luz. Precisión con un amplio campo de visión a 146 grados. La IMU de seis ejes combinada con la sincronización de fotogramas proporciona una precisión de menos de un milisegundo. Paquete completo con SDK simple de integrar. Fácil desarrollo y rápida integración con los datos de profundidad creados a través del sensor EYE S. La combinación de la amplia línea de base y el rango flexible de hasta veinte metros proporciona una solución versátil para

aplicaciones logísticas, donde las esquinas son estrechas, las variables en movimiento son muchas y el requisito de precisión de los datos de profundidad es alto. MYNT EYE proporciona una tasa de continuidad de respuesta rápida mientras la plataforma permanece en movimiento. Para navegación o escaneo volumétrico, en áreas con poca iluminación, los componentes IR pueden ofrecer información de profundidad secundaria consistente, creando flexibilidad. También el evitar obstáculos requiere continuidad y una alta tasa de llenado. Los drones y plataformas similares con alta movilidad y peso ligero que pueden cambiar la dirección del movimiento rápidamente se benefician del mapa de profundidad de 60 FPS de MYNT EYE y un amplio campo de visión, que cubre más área y variables con información de profundidad confiable. En comparación con una solución de módulo multisensorial que cubre la misma área, el ligero MYNT EYE produce un mapa de profundidad de mayor definición con una distorsión mínima a través de un obturador global. Los proyectores IR mejoran la precisión al reducir el ruido espacial y el ruido temporal. Los datos adicionales añaden precisión y mejoran el resultado de alimentación para la medición volumétrica. Las aplicaciones de reconocimiento facial, identificación de objetos, seguimiento de movimiento, mapeo y AR y VR se pueden integrar a la perfección. La IMU de menos de un milisegundo y la velocidad de sincronización de cuadros agregan ventajas adicionales para el uso de MYNT EYE al reducir el ruido temporal. (MYNTAI, 2019) En la Ilustración 9-2 se muestra la cámara y la Tabla 1-2 muestra un resumen de sus características.

Tabla 1-2: MYNTEYE S1030-IR datasheet

Model	S1030-IR-120/Mono
Size	**PCB dimension 149mm × 24 mm total dimension 165×31.5×29.6 mm**
Frame Rate	**10/15/20/25/30/35/40/45/50/55/60 FPS**
Resolving Power	**752x480**
Depth Resolving Power	**|Base on CPU/GPU Up to 752×480@60FPS**
IR detectable range	**Up to 3m**
Visual Angle	**D:140°H:120°V:75°**
Color Pattern	**Monochrome**
Pixel Size	**6.0 × 6.0 μm**
Shutter Speed	**17 milliseconds**
Baseline	**120.0 mm**
Camera Lens	**M6 Camera Attachment**
Focal Length	**2.1mm**
Motion Perception	**6 Axis IMU**
Control Function	**Exposure/Shutter/Brightness/IR**

IMU Frequency	**100/200/250/333/500 Hz**
Cameras Synchronization Precision	**<1ms (up to 0.05ms)**
Working Distance	**0.8-5m+**
Scanning Mode	**Global shutter**
Power	**3.5W @ 5V DC from USB**
Output data format	**Raw data**
Data transfer Interface	**USB 3.0**
Weight	**196g**
UVC MODE	**YES**

Fuente: (Cappellaro 2019, p. 36)

Ilustración 9-2: Cámara MYNT EYE S1030-IR-120/Mono

Fuente: (MYNTAI. 2019)

2.5 Software para el procesamiento de imágenes

2.5.1 SDK

Un kit de desarrollo de software (SDK) es un conjunto de herramientas proporcionado usualmente por el fabricante de una plataforma de hardware, un sistema operativo (SO) o un lenguaje de programación. (RedHat, 2020)

Los SDK permiten que los desarrolladores de software creen aplicaciones para esa plataforma, ese sistema o ese lenguaje de programación específicos. Imagínelos como un kit de herramientas, algo así como la bolsa de herramientas que se incluye con un mueble para armar por su cuenta, pero aplicado al desarrollo de aplicaciones. De esa manera, usted tiene las piezas fundamentales (es decir, las herramientas de desarrollo) que necesita para llevar a cabo el trabajo. El contenido del kit depende de cada fabricante.

Por lo general, un SDK básico incluye un compilador, un depurador y varias interfaces de programación de aplicaciones (API), aunque también puede incluir otros elementos:

- Documentación
- Bibliotecas
- Editores
- Entornos de tiempo de ejecución o desarrollo
- Herramientas de prueba o análisis
- Controladores
- Protocolos de red

Un SDK útil incluye todos los elementos que un desarrollador podría necesitar al momento de crear aplicaciones nuevas para el producto específico y su ecosistema. Algunos SDK también incluyen un ejemplo o proyecto de prueba básico para que los desarrolladores puedan comenzar a trabajar cuanto antes.

2.5.2 OpenCV

Es una librería software open-source de visión artificial y machine learning. OpenCV tiene una licencia BSD, lo que permite utilizar y modificar el código, tiene una comunidad de más de 47000 personas y más de 7 millones de descargadas. Es una librería muy usada a nivel comercial, desde Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota, Applied Minds, VideoSurf, Zeitera. La librería tiene más de 2500 algoritmos, que incluye algoritmos de machine learning y de visión artificial para usar. Estos algoritmos permiten identificar objetos, caras, clasificar acciones humanas en vídeo, hacer tracking de movimientos de objetos, extraer modelos 3D, encontrar imágenes similares, eliminar ojos rojos, seguir el movimiento de los ojos, reconocer escenarios. Se usa en aplicaciones como la detección de intrusos en vídeos, monitorización de equipamientos, ayuda a navegación de robots, inspeccionar etiquetas en productos, etc. OpenCV está escrito en C++, tiene interfaces en C++, C, Python, Java y MATLAB interfaces y funciona en Windows, Linux, Android y Mac OS. (Gracia, 2013)

2.5.3 Visual Studio

Visual Studio es un entorno de desarrollo integrado (IDE) para el sistema operativo Windows. Este IDE soporta diversos lenguajes de programación como Visual C++, Visual C#, Visual J# y Visual Basic.Net entre otros. (Patiño, 2013, p. 53)

Está basado en BASIC (Beginner's All-purpose Symbolic Instruction Code), un lenguaje de programación de alto nivel, que puede ser tanto interpretado como compilado, no estructurado, y de fácil aprendizaje, Visual Basic .NET es un lenguaje de programación orientado a objetos que cuenta con los beneficios que le brinda .NET Framework, el modelo de programación diseñado para simplificar la programación de aplicaciones en un entorno sumamente distribuido. Desde Visual Studio 2015 su nuevo IDE, viene preparado para desarrollar aplicaciones para Windows, pero también para Android, iOS y Windows Phone. Esto permite desarrollar en C# nativo sin necesidad de usar Java por ejemplo, por medio de XAMARIN. También podemos crearlas usando HTML y JavaScript.

Para poder trabajar con este IDE hay que mencionar algunos aspectos importantes de su interfaz.

1. Explorador de soluciones. En esta ventana se visualizan los archivos que están incluidos en el proyecto.
2. Barra de herramientas. En esta barra se encuentran todas las funciones para modificar los archivos y las propiedades del proyecto.
3. Visor de archivos. En esta ventana se visualiza el archivo para poder editarlo.
4. Reporte de acciones. En esta ventana se muestran los mensajes generados después de un proceso de compilación

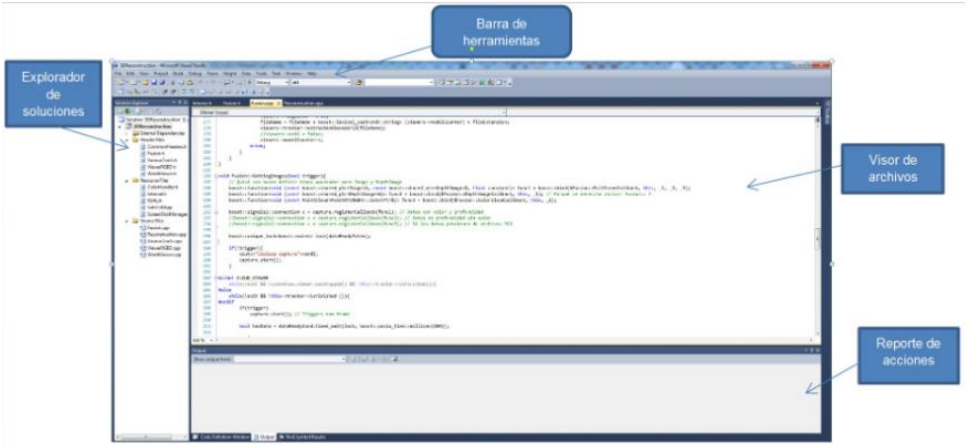


Ilustración 10-2: Interfaz de Visual Studio

Fuente: (Patiño 2013, p. 54)

2.5.4 Matlab

Matlab es una abreviatura del término «Laboratorio de matrices», ya que inicialmente se denominó lenguaje de programación de matrices. Es un lenguaje de programación de cuarta generación. Fue diseñado por Cleve Moler con el objetivo de encontrar una forma alternativa de realizar álgebra lineal y computación numérica. (Velazquez 2020)

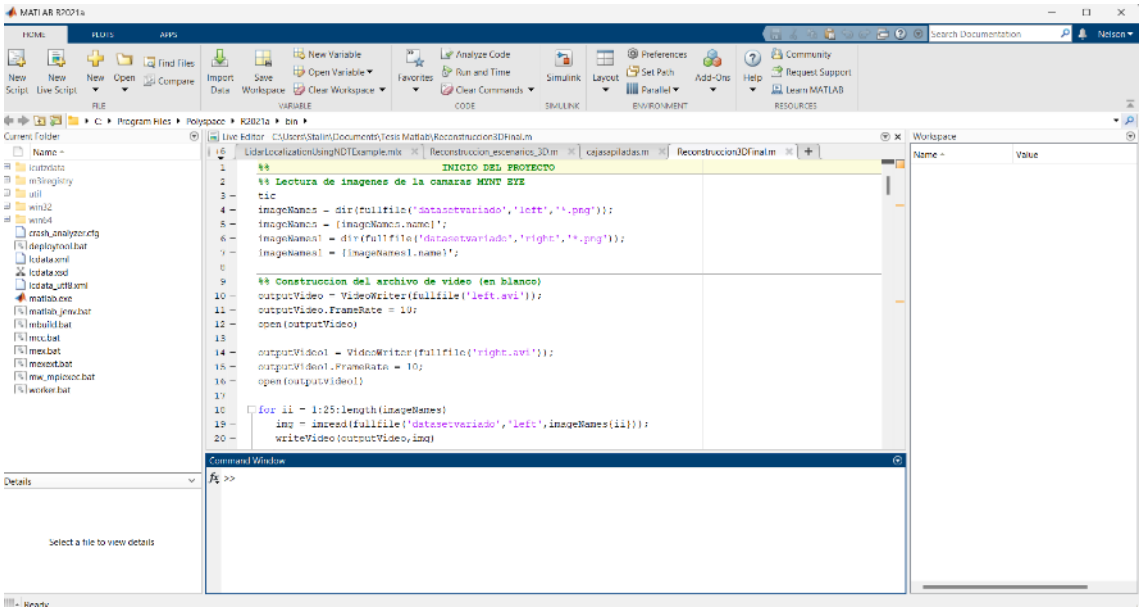


Ilustración 11-2: Interfaz MATLAB

Realizado por: Shagñay Nelson, 2023

Algunos beneficios:

- Facilita el desarrollo de la simulación científica gracias a la biblioteca incorporada.
- La funcionalidad se puede ampliar enormemente agregando cajas de herramientas.
- Alta eficiencia de codificación y productividad, ya que no requiere un compilador para su ejecución.
- Ideal para desarrollar aplicaciones de investigación científica.
- Matlab Coder permite convertir código para usar en otros lenguajes de programación, como C ++, Java y Python.
- Plataforma independiente.

CAPITULO III

3 METODOLOGÍA

Este capítulo detalla las etapas para ejecutar el proyecto como lo son la ejecución del proyecto, el diseño del sistema, definición de las etapas del proyecto, desarrollo del algoritmo e implementación del sistema de reconstrucción.

La metodología puesta en práctica para el proyecto es la definida metodología de Cascada, debido a que el proyecto se centra, básicamente, en la elaboración de un algoritmo para la reconstrucción de escenarios en 3D para ambientes indoor. La metodología Waterfall es un proceso de desarrollo secuencial de proyectos que suele utilizarse en el desarrollo de software. Esta metodología concibe el trabajo en un conjunto de etapas que deben ejecutarse una tras otra. Su nombre viene dado por las diferentes fases que componen el proyecto, ya que deben colocarse una encima de otra siguiendo un orden concreto y estricto de arriba hacia abajo. (Digital Talent Agency, 2018, p. 4)

La metodología en cascada se divide en 5 etapas o fases fundamentales, las cuales comienzan por la etapa de análisis que, define un plan a llevar para desarrollar el sistema propuesto así como especificar los requisitos a cumplirse en el mismo, pasando por la etapa de diseño caracterizada por definir un diseño adecuado con el fin de cumplir con el objetivo principal del proyecto, la etapa de implementación donde se define el algoritmo así como el desarrollo del sistema para la captura de datos con los cuales el algoritmo trabaja, llegando a la etapa de verificación se integran tanto el sistema como el algoritmo para realizarse pruebas de funcionamiento, llegando al final con la etapa de mantenimiento, el cual recoge resultados y realiza mejoras, de ser el caso, al analizar estos resultados. Estas etapas se pueden distinguir en la ilustración 1-3.

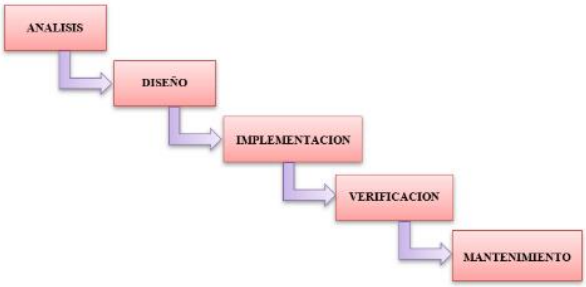


Ilustración 1-3: Metodología de Cascada

Realizado por: Shagñay Nelson, 2022

3.1 Análisis del sistema

En esta fase se hace un análisis de cuáles son los requisitos que debe cumplir el sistema con el fin de tener un sistema en óptimas condiciones. En torno a la problemática planteada en el capítulo 1, donde se plantearon preguntas que buscaban una necesidad a ser satisfechas con el fin de cumplir con los objetivos propuestos. Cumplir con cada una de las preguntas, relacionadas con una etapa de la metodología propuesta, supone el desarrollo del proyecto. Por ende, la fase de análisis busca definir las funciones y características que debe tener el sistema en base a la problemática para que sea un sistema operacional y los requisitos que debe cumplir este sistema de reconstrucción ambiental.

3.1.1 Características y funciones del sistema de reconstrucción de entornos indoor

- Reconstruye un escenario completo capturado mediante imágenes dentro de su trayectoria en forma de lazo.
- Con la capacidad de unir las diferentes nubes de puntos generadas por cada imagen tomada en lapsos de tiempos distintos para generar una nube de puntos global que representan el escenario.
- Determina su trayectoria recorrida automáticamente, una vez realizada la reconstrucción del escenario, dentro de este escenario.
- De rápida implementación y mantenimiento con un sistema sencillo de administrar, utilizar y ajustar de acuerdo con los requerimientos del escenario a reconstruir.
- Basado únicamente en la técnica de visión estereoscópica, característica proporcionada por las cámaras MYNTEYE S1030, mediante la determinación de mapas de profundidad en base a las imágenes obtenidas.
- Sistema compuesto por una cámara estéreo sobre un robot móvil controlado inalámbricamente para la captura de imágenes.
- Robot móvil de bajo costo, pero de baja complejidad de operación y mantenimiento con capacidad de movimiento limitado por una batería recargable.
- Utilizable en entornos no muy amplios debido a la limitación de las cámaras MYNTEYE S1030.
- Resolución del escaneado de hasta 752x480 pixeles, resolución dada por las características de las cámaras.
- No automático, el sistema no permite la adquisición de imágenes directamente desde el algoritmo planteado, se hace usos de varias herramientas.

- Utilizable en ambientes con distinta iluminación, un sistema con capacidad de reconstruir ambientes con distintos niveles de intensidad luminosa.

3.1.2 Requisitos del sistema de reconstrucción de entornos indoor

- El sistema debe basarse en la técnica de reconstrucción por visión estereoscópica debido al uso de cámaras estéreo y usando el método de correspondencia de puntos que implica 2 imágenes haciendo referencia a los mismos objetos de una parte específica del entorno indoor.
- La determinación de la trayectoria del robot móvil debe estar ligada a la técnica de reconstrucción, quiere decir que busca relación entre diferentes nubes de puntos generadas mediante la técnica de reconstrucción para determinar el movimiento del robot entre la correspondiente nube de puntos y la siguiente.
- La cámara estéreo debe estar correctamente calibrada con la finalidad de minimizar errores como distorsión e irregularidades generados por defectos de fábrica de la cámara en cuestión.
- Un sistema de reconstrucción de entornos indoor con bajo costo en comparación con los sistemas profesionales de reconstrucción debido a las características de las cámaras disponibles en la ESPOCH así como de dispositivos tecnológicos disponibles en el país.
- Fácil de manejo para permitir un acceso a personas comunes su uso, con una interfaz amigable con el usuario que permita la utilización y manejo de cada recurso del sistema.
- Diseño simple de sencilla implementación y mantenimiento donde se pueda hacer la captura de múltiples imágenes seguidas de todo un entorno mientras se controla al robot móvil, además de permitir un fácil mantenimiento de cámaras.
- Capacidad de portabilidad para poder movilizar el sistema a distintos entornos y que no ponga en riesgo la integridad de cada componente.
- Dispositivo compatible con un sistema operativo compatible con Matlab y SDK de las cámaras basadas en OpenCV 3.4.1, que son las herramientas usadas para el procesamiento y adquisición de imágenes

3.1.3 Análisis de los requisitos para la reconstrucción de entornos indoor

El sistema de reconstrucción de entornos indoor mediante el análisis y procesamiento de imágenes, usa una única técnica de reconstrucción que es la técnica de visión estereoscópica el cual simula la visión humana, para lo cual se debe determinar los parámetros de calibración de la cámara estéreo a usarse para corregir problemas como forma de pixeles y tipos de distorsión como lo son la distorsión radial y tangencial vistas en el capítulo anterior.

Debe ser un sistema de bajo costo y de sencilla realización, pero con resultados lo suficientemente buenos como para identificar los objetos que componen el entorno en reconstrucción una vez digitalizado, también debe ser un sistema sencillo de manipular para que una persona sea capaz de manejarlo sin tener un nivel alto en procesamiento de imágenes y programación. El algoritmo debe ser capaz de leer las imágenes capturadas por la cámara para luego determinar la nube de puntos a partir de estas, unir las diferentes nubes de puntos generadas para generar una sola nube de puntos y, por último, hacer un análisis correspondiente de relación de las distintas nubes de puntos para determinar la trayectoria del robot móvil. Por último, el sistema debe ser compatible con diversos sistemas operativos, evitando así que un solo computador sea el encargado de hacer funcionar el sistema.

3.2 Diseño

Para poder desarrollar esta etapa de la metodología, se abordan 3 temas principales como lo son: la técnica de reconstrucción que se usó, las fases de operación del sistema de reconstrucción de entornos y el diseño del sistema de reconstrucción de entornos indoor, eso temas de desarrollan para poder entender y resolver la problemática propuesta en el capítulo 1, partiendo desde ¿Cuál es el mejor diseño de una composición cámaras sobre el robot móvil que permitan obtener una visión de 360 grados al escenario indoor a reconstruir?.

Según el marco teórico descrito en el capítulo 2, se señalan algunas formas y métodos de conversión de objetos físicos a digitales. La técnica más relevante debido a que es la técnica usada para el proyecto es una técnica pasiva descrita como la técnica de visión estereoscópica, la cual consiste en la triangulación de puntos extraídos del par de imágenes obtenidos que tienen ventaja de ser enteramente tridimensionales, ya que es la técnica con la cual se basa la vista humana para la percepción de profundidad.

Una vez entendido el funcionamiento de la técnica de visión estéreo y en base a hacer uso de herramientas tecnológicas adquiridas por la Escuela Superior Politécnica de Chimborazo para uso de los estudiantes, como es este caso las cámaras MYNT EYE S1030, las cuales tienen el objetivo de ser usadas principalmente para la aplicación de la técnica de visión estéreo para la virtualización de objetos, se procede a desarrollar el sistema de reconstrucción en base a esta técnica.

3.2.1 Diseño físico del sistema de reconstrucción de entornos indoor

Para la implementación del sistema de reconstrucción de entornos indoor, es necesario un diseño que satisfaga las características de este, para lo cual, en la etapa de análisis se enlistó ciertas características que el sistema debe cumplir, entre los cuales estaba el diseño de un robot móvil sobre el cual va montada la cámara estéreo, este debiendo ser de bajo costo, pero sencillo de manejo. Para lo cual, se optó por un diseño de chasis de robots móviles sencillo, pero adaptable, sobre el cual se monta un stick para acoplar la cámara y nos dé una altura prudencial sobre el suelo de 40 cm y con 10 grados de inclinación de la cámara sobre el eje horizontal, con la finalidad de poder observar de mejor manera el escenario a reconstruir y, al mismo tiempo, no tan elevado para evitar un desbalanceo en el robot. Para esto, la Ilustración 2-3 nos muestra el prototipo del sistema de reconstrucción de entornos indoor.

Ilustración 2-3: Prototipo del sistema de reconstrucción

Realizado por: Shagñay Nelson, 2022

Cabe acotar que el ensamble de estos tres elementos que son el robot móvil, el stick y la cámara, junto a la computadora que recolecta los datos, llegan componer el sistema de reconstrucción de entornos indoor con superficies planas por la misma manera de ser un sistema sencillos con elementos básicos de construcción.

3.2.2 Etapas del sistema de reconstrucción

Los diferentes subprocesos que lleva a cabo el proceso principal del sistema de reconstrucción de entornos indoor se estructura de manera ordenada, empezando desde la obtención de imágenes mediante el sistema físico observado en la Ilustración 2-3 hasta la evaluación del sistema de reconstrucción. El proceso se divide en subproceso que, de forma secuencial, para el sistema se muestran en la Ilustración 3-3.

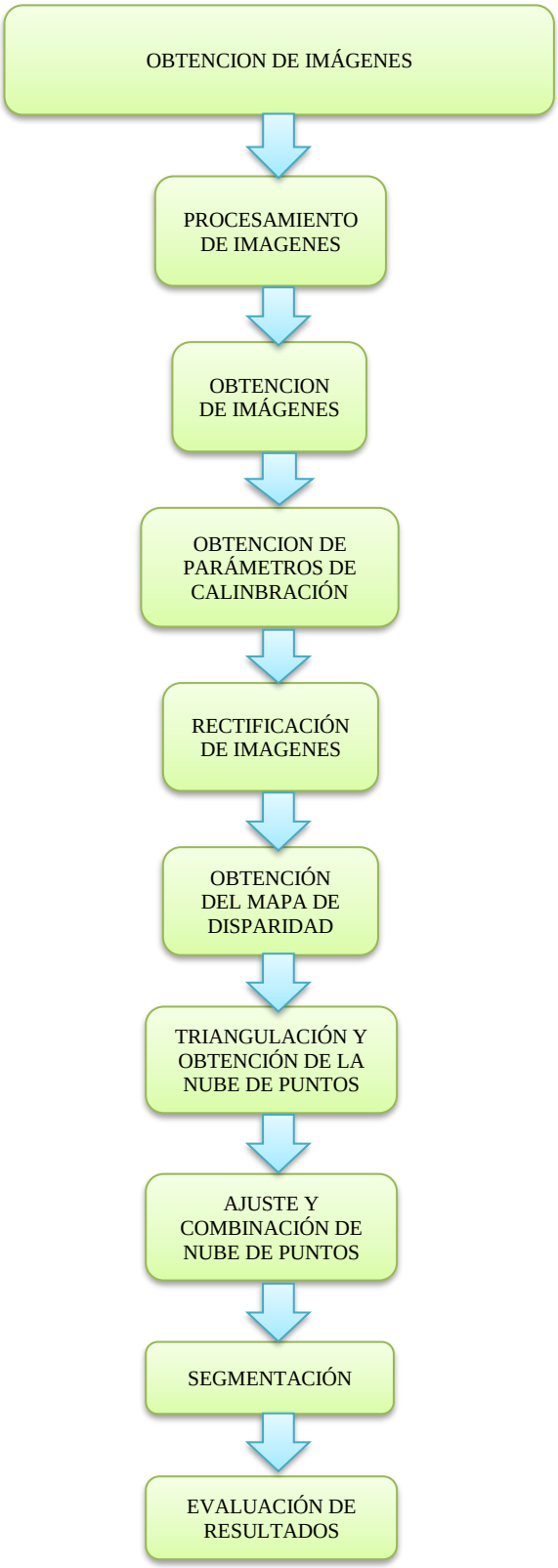


Ilustración 3-3: Subprocesos del sistema de reconstrucción de entornos.

Realizado por: Shagñay Nelson, 2023

3.2.2.1 Obtención de imágenes

La obtención de imágenes es el primer subproceso en el proceso de reconstrucción de entornos y, por ende, uno de los más importantes. Para obtener estas imágenes se usan las cámaras estéreo MYNT EYE S1030-IR-120/Mono fabricada por MYNT donde sus características se presentaron en la tabla 1-2. El control de Hardware de las cámaras se lo hace mediante 2 herramientas que son SDK (Kit de Desarrollo de Software) provista por el fabricante de las cámaras y el compilador de código C++, Visual Studio 2017.

Por recomendación del fabricante, no se hace uso de máquinas virtuales debido a problemas al virtualizar de los controladores para los puertos USB 3.0 debido a que estas cámaras requieren un canal con alta capacidad de transmisión de datos tanto para la comunicación y el envío de información desde las cámaras hasta el computador.

El fabricante proporciona un SDK de 3 niveles basando sus dependencias en *OpenCV*, las cuales son:

- api: Interfaz superior con dependencias de *OpenCV*.
- device: Interfaz entre capas sin dependencias de *OpenCV*.
- uvc: Interfaz inferior sin dependencias de *OpenCV*.

En el proyecto se hace uso de la interfaz superior API ya que es donde se encuentran las aplicaciones desarrolladas por el fabricante y entre esas el proyecto record que es el que hace uso este proyecto para la captura de imágenes. Para el desarrollo de aplicaciones se puede optar por deshabilitar las librerías de *OpenCV* buscando el archivo *Option.cmake* y colocando la opción *with API* a *OFF*.

Como se mencionó en el párrafo anterior, para la captura de imágenes se hace uso del proyecto record desarrollado por MYNTAI en 2019. Este código fue desarrollado en *OpenCV* por lo que se hace uso de esta herramienta para ejecutarlo y consta de dos archivos que son record.cc y dataset.cc. El archivo dataset.cc se encarga de crear los directorios donde se guardan distintos datos como lo son la imagen de cámara izquierda, la imagen de la cámara derecha, la disparidad y la profundidad calculada por la toma de cada par de imágenes estéreo mediante la función GetStreamWriter, así como de nombrar cada dato, enumerarlos mediante la función SaveStreamData y, al final, guardarlos en los directorios creados en formato .png. Este archivo el llamado desde el archivo record mediante la función *dataset.SaveStreamData()*. El archivo record, captura los datos como imágenes tanto de la cámara izquierda como de la cámara derecha, así como la disparidad y la profundidad desde el momento que se manda a ejecutar el

proyecto ya que usas la función *Start* al principio y no está condicionada de ningún modo, se debe configurar el número de imágenes por segundo (fps) en el archivo, donde el valor mínimo aceptado es de 10 fps, llama al archivo dataset mediante su librería. La funcion *WaitForStream()* llama a los diferentes datos ya mencionados y con la función *GetStreamDatas()* se capturan los datos, dependiendo del argumento que contengan como Stream::LEFT, capturara las imágenes de la cámara izquierda. Para finalizar, se llama a al archivo dataset mediante la función *dataset.SaveStreamData()* para guardar los datos. Una vez hecha la toma de datos requerido se detiene la ejecución del proyecto al presionar la tecla "q", "Q" o "ESC".

Del proyecto record, los datos a usar son solo las imágenes, no rectificadas, de la cámara derecha e izquierda con una resolución de 752x480 pixeles, las cuales se muestran un ejemplo de estas en la Ilustración 4-3.

Ilustración 4-3: Imágenes tomadas por la cámara estéreo MYNT EYE S1030 Izquierda y Derecha

Realizado por: Shagñay Nelson, 2023

3.2.2.2 Lectura de imágenes mediante MATLAB

Mediante el uso de la herramienta matemática MATLAB se hace, lo que conlleva, todo el procesamiento de las imágenes, las cuales deben ser importadas desde el directorio donde las guarda el proyecto record. No se puede hacer la captura de imágenes directamente desde MATLAB debido a que el fabricante no ha desarrollado la compatibilidad con esta herramienta y solo se lo puede hacer mediante el SDK del fabricante y de Visual Studio.

Para importar las imágenes y sea de mayor facilidad, se debe mover estas al directorio contenedor del proyecto en MATLAB llamado Reconstruccion_escenarios_3D.m. Se indica la ubicación de estas mediante la función *dir()* y como argumento de esta función el nombre de la carpeta contenedora así como la extensión de las imágenes, en este caso, *.png*, como se muestra en la sintaxis siguiente:

```
imageNames = dir(fullfile('dataset1','left','*.png'));
```

La variable *imageNames* contiene los nombres de cada imagen en la carpeta en un arreglo en MATLAB. En la Ilustración 5-3 muestra la carpeta contenedora de las imágenes tomadas por la cámara estéreo, ya movidas a la carpeta del proyecto creado en MATLAB.

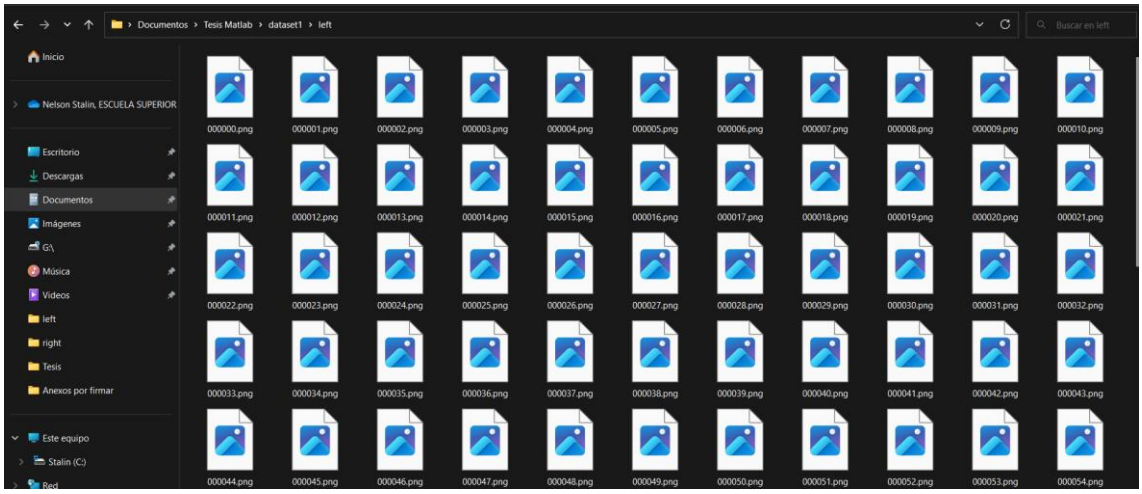


Ilustración 5-3: Carpeta contenedora de imágenes.

Realizado por: Shagñay Nelson, 2023

A partir de las imágenes ya leídas en MATLAB, se procede a convertirlas en video con extensión *.avi* para mejor manejo de estas, por lo que mediante las siguientes líneas de código se procede a la creación del video.

mess

```
outputVideo = VideoWriter(fullfile('left.avi'));
outputVideo.FrameRate = 10;
open(outputVideo)
for ii = 1:length(imageNames)
   img = imread(fullfile('dataset1','left',imageNames{ii}));
   writeVideo(outputVideo,img)
end
close(outputVideo)
```

Las funciones usadas se enumeran a continuación:

- *VideoWriter()*: Crea un archivo de video con extensión *.avi* para poder escribir sobre él.
- *Fullfile()*: Toma el directorio donde se encuentra creado el proyecto e indica que se cree el archivo de video en ese lugar y como parámetro de entrada el nombre del archivo.
- *open(outputVideo):* Se abre al archivo de video para poder modificarlo y agregarle las imágenes tomadas.
- *ciclo for:* Un ciclo de repetición usado para ir leyendo una por una las imágenes de la carpeta contenedora e ir agregándolas al archivo de video.
- *Imread():* Lee una imagen de la carpeta contenedora que va cambiando según se vuelva a iniciar el *ciclo For.*
- *writeVideo():* Agrega las imágenes al archivo de video una por una dependiendo del *ciclo For.*
- *close(outputVideo):* Cierra el archivo de video una vez agregadas todas las imágenes de la carpeta contenedora.

El resultado es la creación de un video con la extensión *.avi* llamado *left* contenedor de cada una de las imágenes que se las puede manipular de mejor manera y poder observar los datos obtenidos de las cámaras que el proyecto *record* no nos da. La extensión *.avi* es usado por su facilidad de agregar datos de video como imágenes de forma secuencial y audio de ser el caso, por lo que a este tipo de archivos se los llama como contenedores. Se procede a hacer lo mismo con las imágenes tomadas por la cámara derecha y se genera un archivo de video con nombre *right*. El resultado de este proceso se muestra en la Ilustración 5-4.

Ilustración 6-3: Video generado mediante MATLAB y las imágenes capturadas

Realizado por: Shagñay Nelson, 2023

Para la lectura de estas imágenes, se hiso uso de los archivos de videos creados, por lo que la siguiente sintaxis nos muestra la foma en que se manipuló cada frame de los videos creados.

```
readerLeft = VideoReader(videoFileLeft);
readerRight = VideoReader(videoFileRight);
frameLeft = readFrame(readerLeft);
frameRight = readFrame(readerRight);
```

Donde mediante la función *VideoReader()*, crea un objeto que lee cada componente del archivo y sus características como lo son los fps, las imágenes que lo componen tipo de archivo y demás variables no necesarias, de donde, mediante el uso de la función *readFrame*() se captura, por defecto, la primera imagen que compone a cada archivo de video. En la Ilustración 7-3 se muestra la imagen guardada en la variable *frameLeft* y los objetos y variables creados en el workspace de MATLAB.

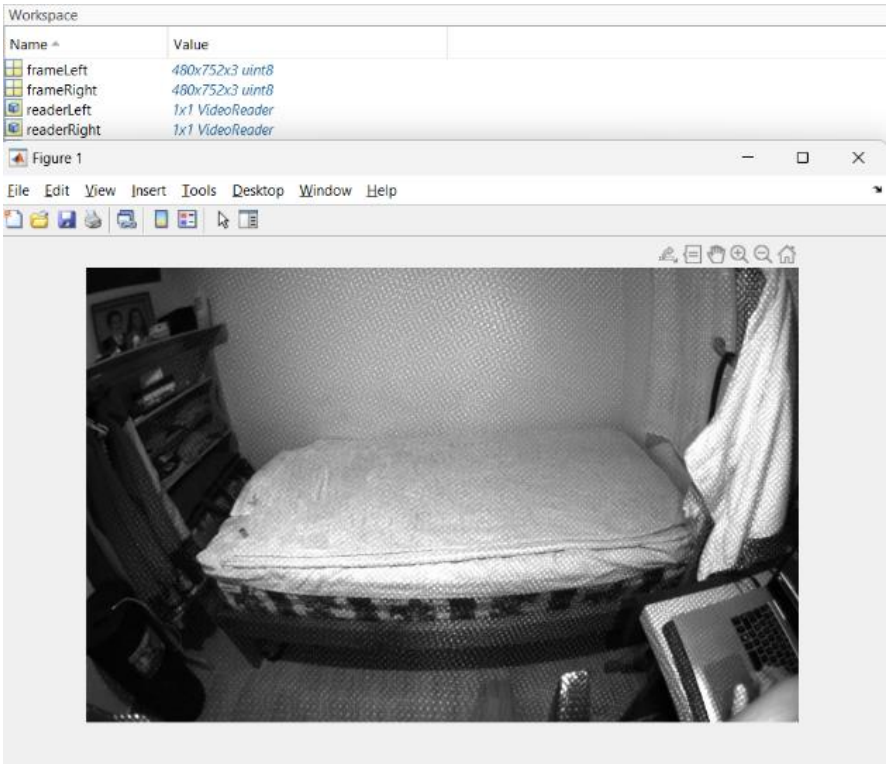


Ilustración 7-3: Imagen capturada del archivo de video *left* y variables contenedoras en el workspace de MATLAB

Realizado por: Shagñay Nelson, 2023

Las imágenes contenidas en la variable *frameLeft* y *frameRight* son de tipo RGB por lo que esta se contiene en 3 matrices bidireccionales, una matriz por color primario que define a la imagen. El tipo de dato uint8 define 8 bits para cada píxel representados por enteros de 0 a 255, es decir, un byte por píxel.

3.2.2.3 Obtención de parámetros para rectificación de imágenes

Para la rectificación de las imágenes, se debe obtener los parámetros de calibración de la cámara MYNT EYE S1030, por lo cual se hiso uso de una herramienta didáctica de calibración de cámaras monoculares y estéreo llamado *Stereo Camera Calibrator* con los cuales calcula los parámetros de calibración y así poder exportarlos al proyecto de reconstrucción. Cabe decir que existen otros métodos de calibración de las cámaras como los son la herramienta ROS que trabaja en el sistema operativo LINUX, pero se la evitó para no complicar el proyecto ya que todo el proyecto se encuentra desarrollado en WINDOWS. La aplicación es fácil de usar, debido a que acepta la entrada de imágenes estéreo con un patrón de calibración, es este caso se usó la patron "*checkerboard*" el cual es parecido a una tabla de ajedrez, aunque se pude usar otros patrones de calibración como los son "*AprilTag*" o cuadrículas circulares simétricas y asométricas. MATLAB define un proceso lineal de pas-os para calibrar cámaras, entre estas las cámaras estereo, esto se pude observar en la Ilustración 8-3.

Ilustración 8-3: Proceso de pasos a seguir para calibrar cámaras en MATLAB

Fuente: (MATHWORKS, 2022)

Este proceso sirve como ayuda al momento de calibrar cámaras, ya que presentan una serie de consejos a seguir al utilizar la herramienta y obtener parámetros de calibración bueno o mínimamente aceptables. La aplicación necesita identificar un punto de origen único, por lo que se aconseja usar un patrón de calibración sin ambigüedad de 180^0, es decir, el patrón no debe ser el mismo al girarlo en 180^0. Para satisfacer esta condición, en el caso del patrón de "checkboard" se debe se debe tener un número para de cuadrados en horizontal y un número impar de cuadrados en vertical, tal como se muestra en la Ilustración 9-3.

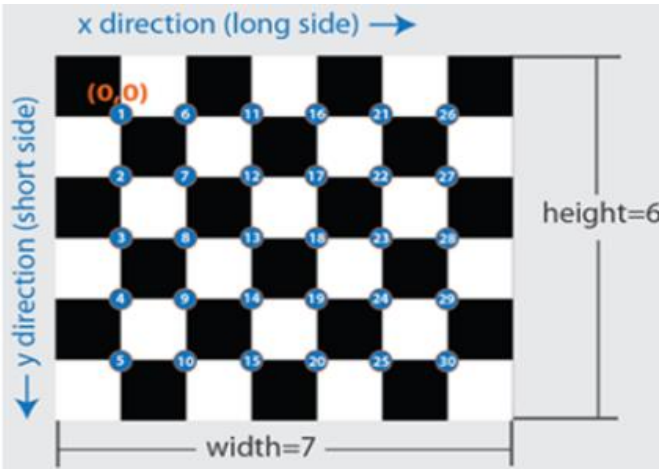


Ilustración 9-3: Patrón de calibración "*checkerboard*"

Fuente: (MATHWORKS, 2022)

Otro requerimiento necesario para que el patrón sea aceptado es que el patrón tenga 2 cuadrados negros en esquinas continuas y en las otras 2 esquinas continuas, 2 cuadrados blancos, esto le permite a la herramienta detectar la orientación del patrón de calibración y el origen. La herramienta asigna el lado más largo al eje X y el lado más corto al eje Y.

En la calibración de la cámara MYNT EYE S1030 se hizo uso de un patrón de calibración de "*Checkerboard*" de 9x8 cuadros, cada cuadro con una medida de 25mm por lado. Por otro lado, el patrón de calibración de estar totalmente plano y sobre una superficie sin deformaciones, con el fin de evitar tomas de imágenes con el patrón deforme, lo que conlleva a tener parámetros de calibración no funcionales.

En la preparación de la cámara y captura de imágenes del patrón de la calibración MATLAB presenta algunas reglas y consejos que permiten una mejor obtención de imágenes del patrón de calibración, las cuales son:

- Para capturas de imágenes en cámaras con enfoque automático, se recomienda desactivar este y configurar el enfoque según la aplicación.
- Colocar el patrón de calibración a una distancia tal de la cámara de modo de que este ocupe al menos el 20% de la imagen.
- Capturar las imágenes de tal manera que el patrón sea visible para ambas cámaras.
- No cambiar la configuración interna de las cámaras como zoom, contraste u otras entre imágenes, esto cambiará la distancia focal al momento de calibrar.
- Para una mayor precisión en la calibración es recomendable utilizar un mínimo de 10 a 20 pares de imágenes.
- Utilizar imágenes sin comprimir o imágenes en formato de compresión sin pérdidas como PNG.
- La distancia que hay entre la cámara y el patrón de calibración es la distancia en la que se mide y se reconstruyen los objetos.
- Colocar la cámara en un ángulo inferior a 45^0 con respecto al plano de la cámara como se observa en la Ilustración 10-3.
- Las oposiciones de las imágenes deben tener distintas orientaciones relativas a la cámara.
- Para cámaras con distorsión radial es necesario que, al capturar las imágenes con el patrón de calibración, estás se tomen en las periferias horizontales y verticales debido a que la distorsión de la lente aumenta radialmente desde el centro de la imagen y no siempre es uniforme en todo el marco de la imagen.

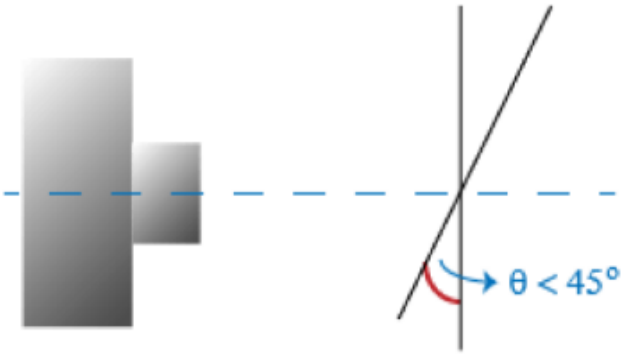


Ilustración 10-3: Obtención de imágenes con respecto al plano de la cámara

Fuente: (MATHWORKS, 2022)

La captura de las imágenes del patrón de calibración se las hace de la misma manera que se explicó en con anterioridad mediante el uso de Visual Studio, solamente que, en este caso, la cámara permanece de manera estática mientras el patrón de calibración se mueve en distintas posiciones y distancias con tal de satisfacer las reglas establecidas anteriormente y una vez se tiene la cantidad de imágenes se procede al uso de la herramienta "*Stereo Camera Calibrator*"

Para abrir la herramienta "Stereo Camera Calibrator" hay que dirigirse a la pestaña "APPS", luego bajamos a el apartado de "IMAGE PROCESSING AND COMPUTER VISION" o también se lo puede abrir desde el Command Window en MATLAB con solo escribir el comando de "stereoCameraCalibrator", donde se despliega una ventana que al dar en la opción "Add Images" se despliega otra ventana que nos pedirá indicar el directorio de las imágenes del patrón de calibración, tanto de la cámara izquierda y derecha, después nos pide indicar el tamaño de los cuadrados del patrón de calibración, que en este caso es de 25 mm. En la Ilustración 11-3 se muestran las ventanas de la herramienta "Stereo Camera Calibrator".

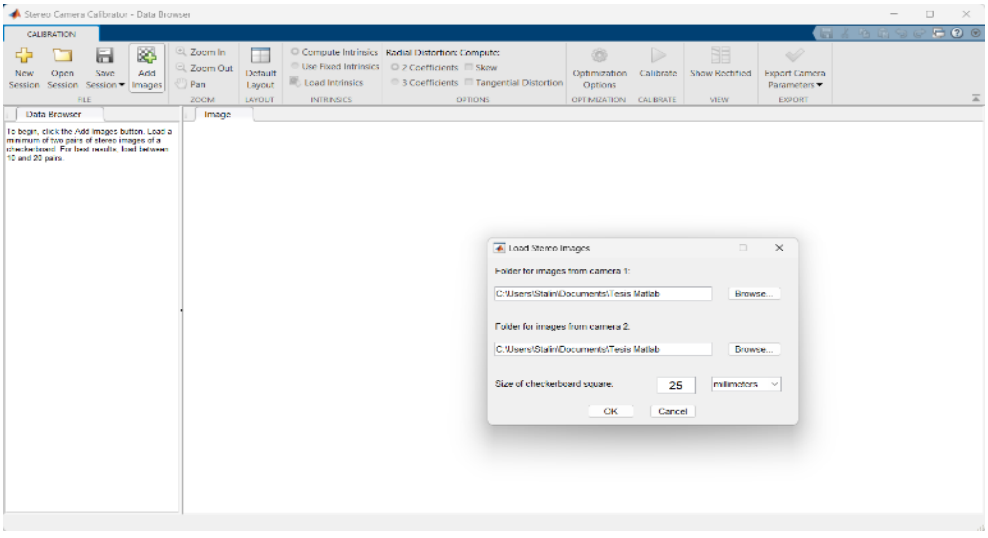


Ilustración 11-3: Ventana de la herramienta "*Stereo Camera Calibrator*"

Realizado por: Shagñay Nelson, 2023

Una vez seleccionadas las imágenes, se procede a cargarla en la herramienta por lo que esta empieza a asociar y asociarlas en pares de imágenes que se pueden observar en la parte izquierda de la ventana, durante el proceso esta herramienta elimina las imágenes que pueden ser duplicadas o muy borrosas, también se las puede eliminar de forma manual de ser el caso. Este proceso puede tomar su tiempo dependiendo del patrón, el número de cuadros y su tamaño. Las imágenes aceptadas por la herramienta se las pude ver la Ilustración 12-3, donde al seleccionar un par de imágenes, estas nos muestran con círculos verdes la intercepción de los cuadrados y con un cuadro amarillo el punto donde se origina la imagen, también se puede observar los ejes X y Y asignados.

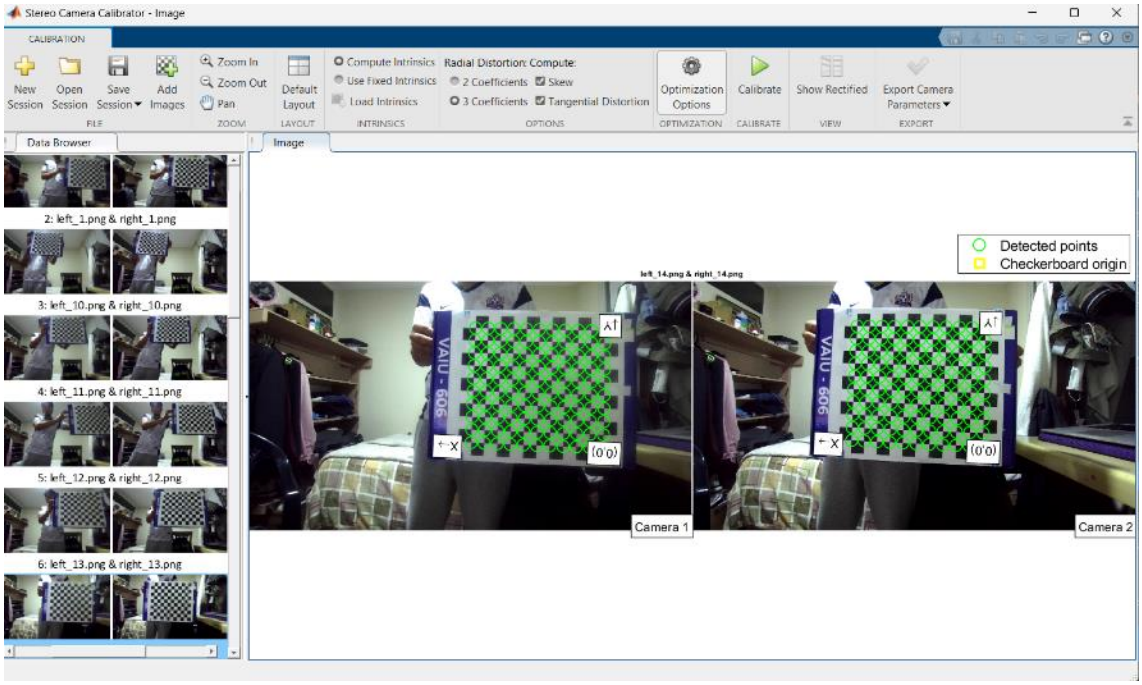


Ilustración 12-3: Par de imágenes estéreo capturados por la herramienta "*Stereo Camera Calibrator*"

Realizado por: Shagñay Nelson, 2023

Como muestra la Ilustración 12-3, en la parte superior de la ventana se pudo encontrar las opciones de calibración, estas permiten ajustar y seleccionar el número de parámetros a obtener, por lo que se decidió obtener los 3 parámetros de distorsión radial debido a la curvatura en las periferias de las imágenes obtenidas, también se ha seleccionado la obtención de los datos de distorsión tangencial debido a que el fabricante de la cámara no presenta ningún dato con respecto a la alineación de los sensores de imagen y, finalmente, se ha marcado la opción de "*Skew*" con la finalidad de corregir errores que pueden hacerse presentes en los píxeles. En la opción de "*Optimization Options*" se puede agregar parámetros iniciales de distorsión intrínseca y de distorsión radial, pero como hay eso valores dados por el fabricante, se los deja en blanco.

Una vez se hiso lo anteriormente explicado, se procedió a calibrar y se espera a que culmine el proceso, el cual dio resultados como se pueden observar en la Ilustración 13-3 donde podemos observar las imágenes con los puntos re proyectados, una estadística de errores y un diagrama físico de la ubicación del patrón de calibración por cada imagen tomada.

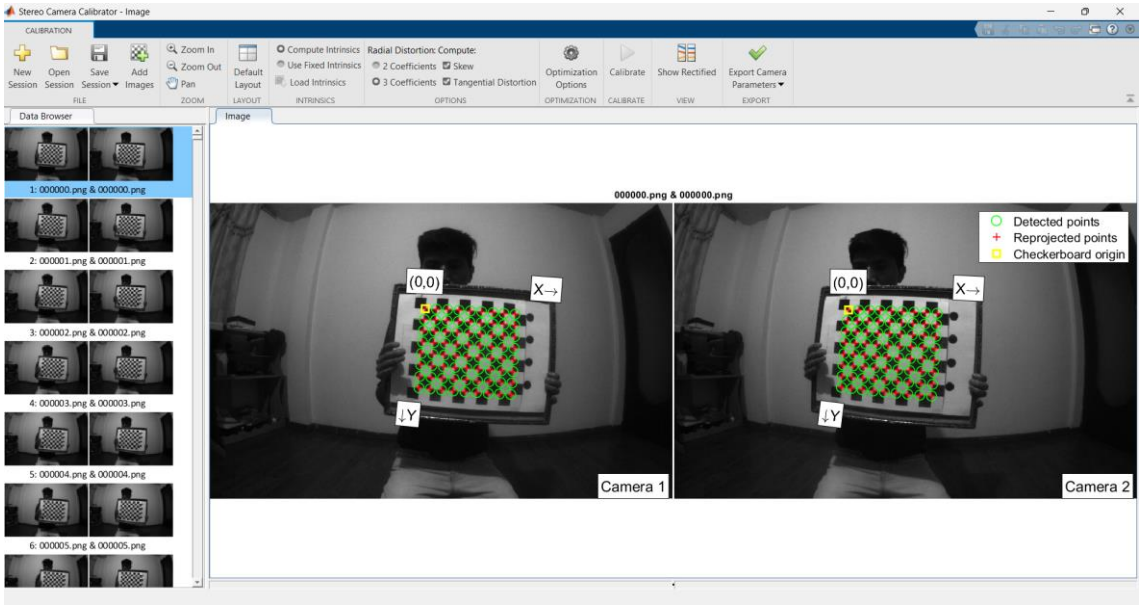


Ilustración 13-3: Imágenes re proyectadas, cuadro estadístico de errores y diagrama físico de patrón de calibración

Realizado por: Shagñay Nelson, 2023

Para disminuir el promedio de error de proyección de cada píxel en los parámetros obtenidos, se analizó el resultado planteado en el gráfico estadístico de error que se muestra en la Ilustración 14-3, donde esta gráfica muestra el error por cada par de imágenes estéreo. Si existen uno o varios pares de imágenes con un margen de error de proyección superior a la media que se muestra con una línea entrecortada, estos pares afectan a los parámetros calculados, por lo que, para mayor precisión, se los fue eliminando, yendo a la sección *Data Browser* e identificando el par, se lo elimina y se vuelve a recalibrar, para volver a hacer el análisis de la gráfica de error de proyección y se repitió el proceso de ser necesario.

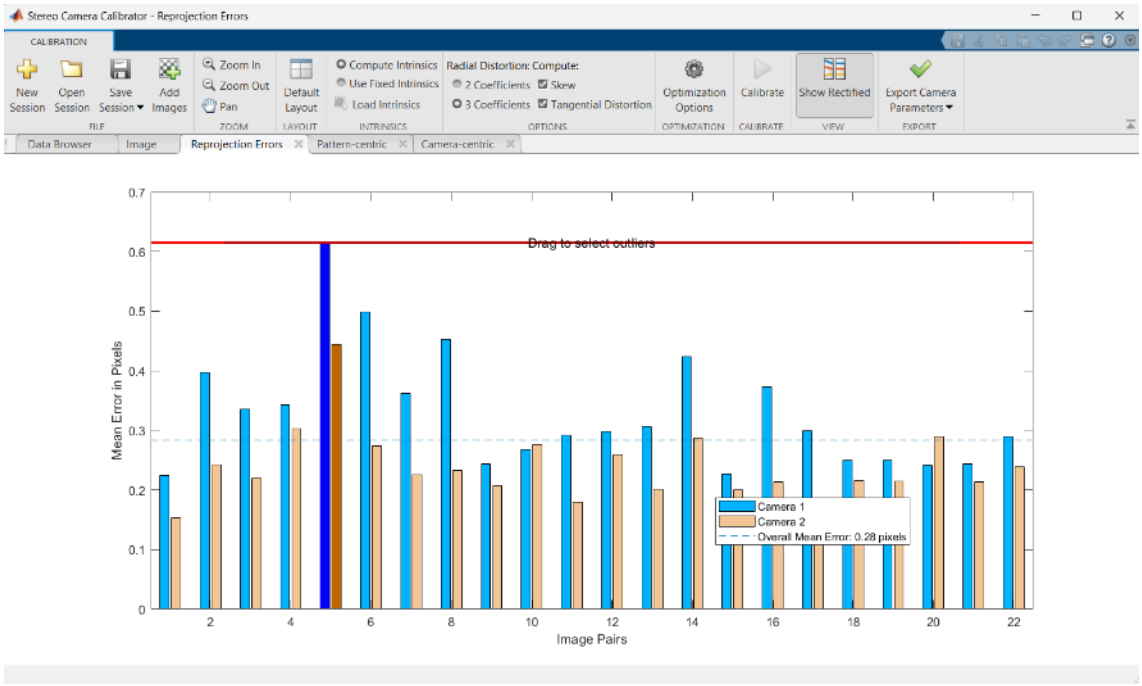


Ilustración 14-3: Cuadro estadístico de errores de re-proyección

Realizado por: Shagñay Nelson, 2023

Mediante el análisis del diagrama de cámara céntrica y patrón estático también se pueden mejorar los resultados de los parámetros de calibración, esto se lo hizo eliminando pares que hayan tenido patrones con una inclinación mayor a 45^0 o, si a su vez, hay cámaras que se encuentran detrás del patrón que puede ocurrir por algún error en los pares de imágenes o error de calibración. En la Ilustración 15-3 se pueden observar estos diagramas a detalle.

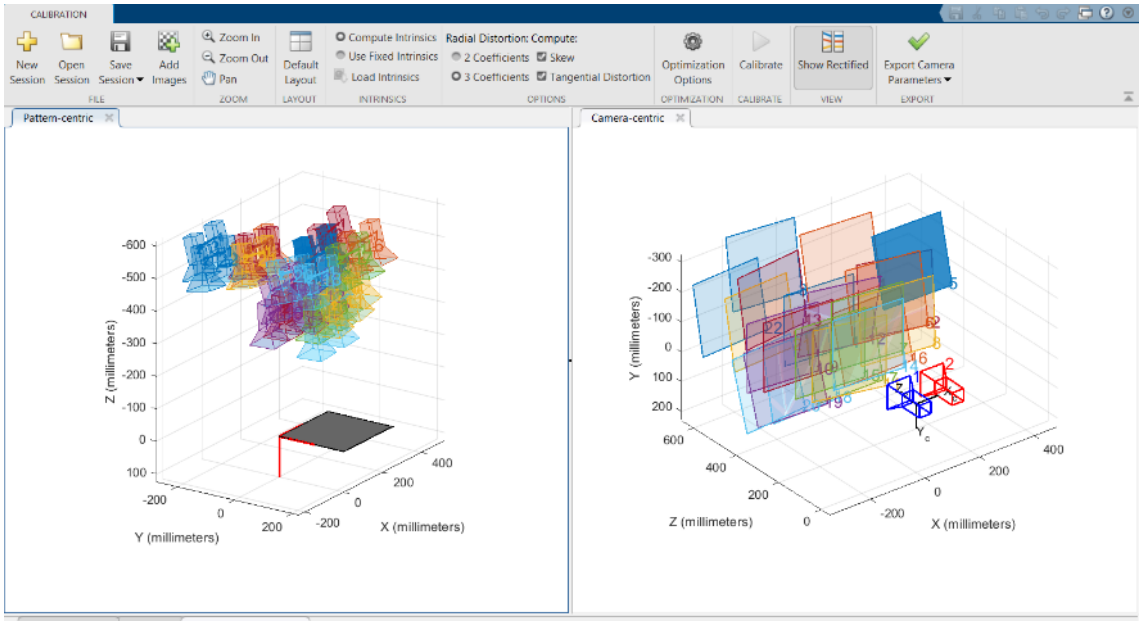


Ilustración 15-3: Diagramas de patrón estático y cámara céntrica

Realizado por: Shagñay Nelson, 2023

Para finalizar con la calibración, se analizó el resultado de la calibración al mostrar el par de imágenes ya rectificadas y verificando si este par de imágenes se encuentran alineados horizontalmente mediante líneas trazadas en horizontal que verifiquen se la alineación. Esto se puede analizar seleccionando la opción "*Show Rectified*" que se encuentra en la parte superior de la ventana como se observa en la Ilustración 16-3.

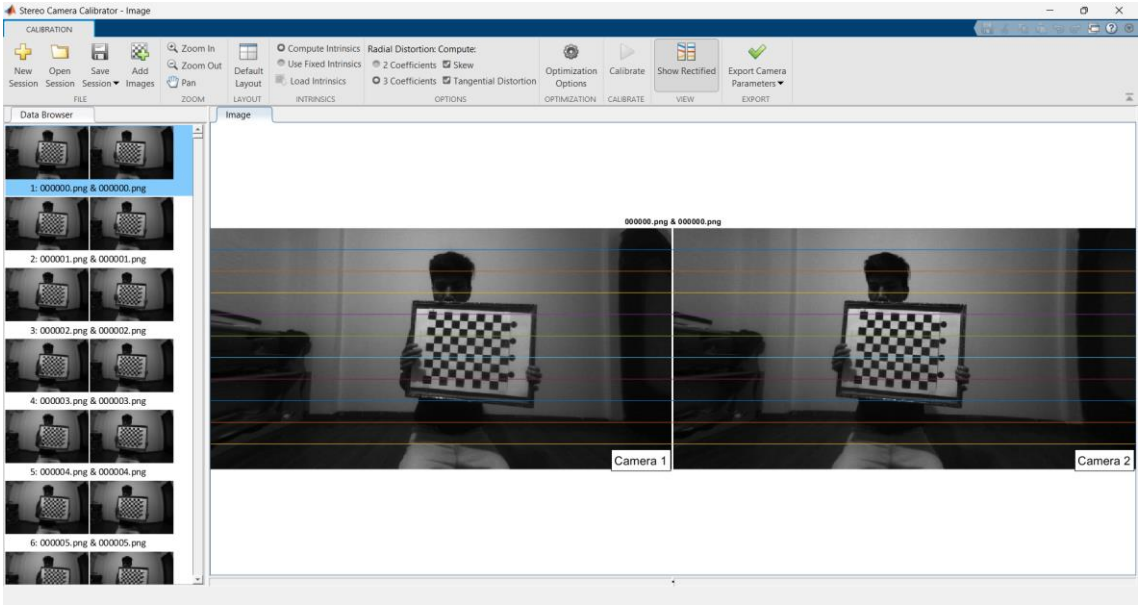


Ilustración 16-3: Imágenes rectificadas por "*Stereo Camera Calibrator*"

Realizado por: Shagñay Nelson, 2023

Se analizaron las imágenes rectificadas y mediante la eliminación o agregación de pares de imágenes estéreo se pudo calcular los parámetros de calibración de la mejor manera, por lo que se procedió a exportarlas al *workspace* del proyecto principal, donde se pudo observar los distintos parámetros calculados como lo son el número de patrones, parámetros de cada cámara individualmente, media de error de re-proyección, matriz de traslación, unidades de medición, etc.

3.2.2.4 Rectificación de imágenes estéreo

Una vez importadas los parámetros de calibración en el proyecto y con las imágenes cargadas, se procedió a la rectificación de estas, para lo cual, MATLAB tiene una función específica para esto la cual es *rectifyStereoImages*, que acepta por parámetros de entrada una imagen de la cámara izquierda y una imagen de la cámara derecha a ser rectificadas y los parámetros de calibración anteriormente exportados, como se explica en el proceso de calibración. La salida de esta función es de 2 imágenes de tamaño 548x1024 píxeles en formato tipo *uint8.* La rectificación de imagen estéreo proyecta imágenes en un plano de imagen común de tal manera que los puntos correspondientes tienen las mismas coordenadas de fila. Esta proyección de imagen hace que la imagen parezca como si las dos cámaras estuvieran en paralelo debido a esto

las imágenes resultantes varían de tamaño a las iniciales. La sintaxis de esta función se muestra a continuación.

[frameLeftRect, frameRightRect] = ...
rectifyStereoImages(frameLeft, frameRight, stereoParamsmil);

Las variables *frameLeftRect* y *frameRightRect* es donde se almacenan las imágenes, derecha e izquierda, rectificadas, mientras que las variables *frameLeft* y *frameRight* contienen las imágenes no rectificadas y la variable *steroParamsmil* contiene los parámetros de calibración de la cámara estéreo. Este proceso junto a la calibración son las bases del proyecto debido a que la rectificación realiza el recorte y el ajuste de posición de los píxeles con el objetico de que cada píxel de coordenada *(x1, y1)* de la imagen 1 correspondan a al píxel con coordenadas *(x2, y2)* de la imagen 2, como se observó en la Ilustración 16-2.

En la Ilustración 17-3 se puede observar una comparativa entre las imágenes rectificadas y sin rectificar, donde, a primera vista, se puede observar una alineación en las periferias de su marco a diferencia de las imágenes no rectificadas que en su periferia forman una curva debido a la distorsión radial.

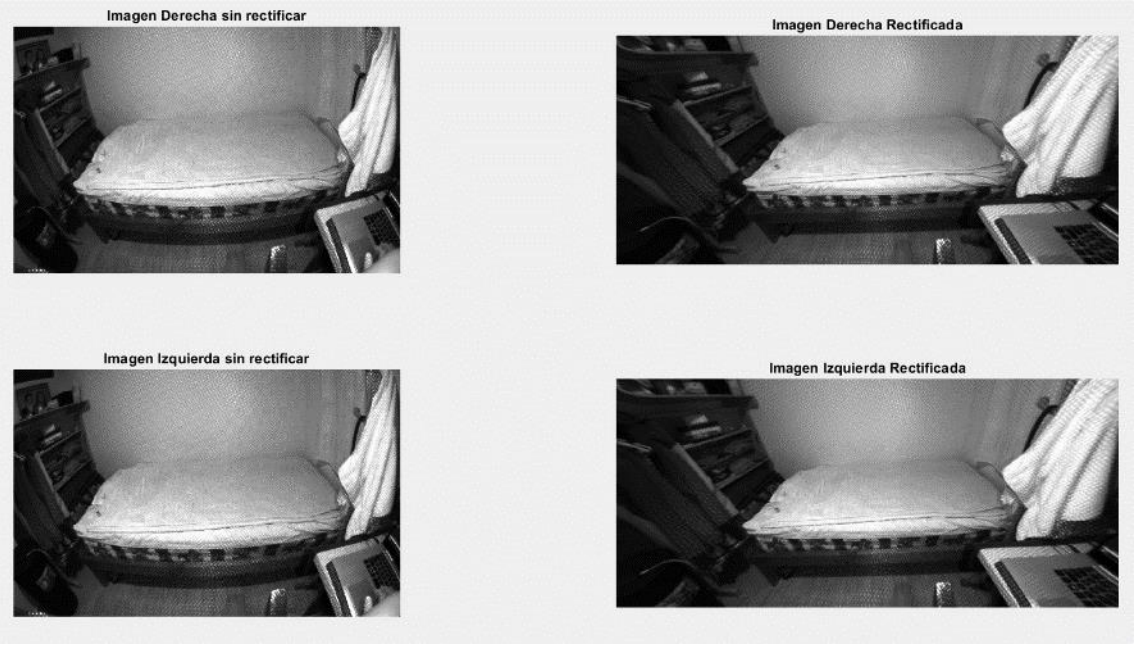


Ilustración 17-3: Comparativa Imágenes rectificadas y sin rectificar

Realizado por: Shagñay Nelson, 2023

Para ver de mejor manera la rectificación de las imágenes, se sobre posicionó las imágenes rectificadas y sin rectificar respectivamente y se analiza píxeles característicos de estas imágenes para comprobar que estén alineadas horizontalmente en el caso de las imágenes rectificadas y como no se encuentran alineadas las imágenes no rectificadas. Esto se puede observar en la Ilustración 18-3 donde las zonas rojas hacen alusión a la imagen de la derecha y las zonas azules hacen alusión a la imagen de la izquierda.

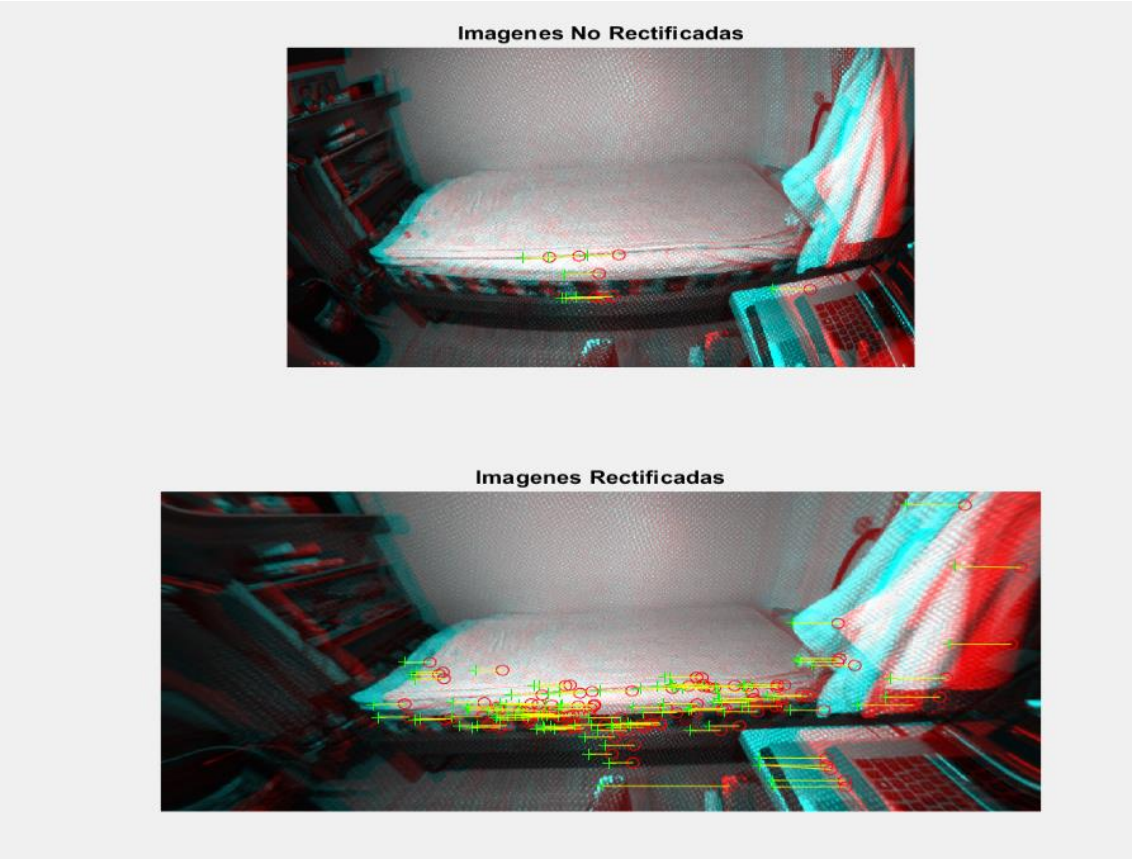


Ilustración 18-3: Visualización de alineación de imagen rectificada en comparación a la imagen no rectificada.

Realizado por: Shagñay Nelson, 2023

Los píxeles característicos de las imágenes son destacados en ambas imágenes donde al hacer la sobre posición se comparó estos píxeles característicos, donde, al unirlos con una línea se puede observar cómo hay una alineación horizontal de estos en la parte de las imágenes rectificadas y como esta alineación no se observa en la sobre posición de las imágenes no rectificadas ya que la línea trazada no se encuentra alineada correctamente con el eje horizontal.

3.2.2.5 Disparidad de imágenes estéreo

Una vez rectificadas las imágenes, se procede a encontrar la disparidad de estas que se define como la diferencia relativa de la posición de 2 imágenes, para lo cual se dispone de la función *disparitySGM*, esta función utiliza el método *semi global matching*, el cual se basa en un algoritmo de visón por computadora para la búsqueda de mapas de disparidad densos a partir de un par de imágenes estéreo rectificadas, donde mide la similitud de cada píxel de una imagen estéreo con cada píxel dentro del subconjunto de la otra imagen estéreo. Este método es de los más usados debido a sus favorables resultados y su comparativa con el tiempo de computo que suele ser menor a otros algoritmos. Para su comparativa se hizo la comparativa con un segundo método que proporciona MATLAB, este es conocido como *block matching* y puede ser llamado mediante la función *disparityBM*, el algoritmo consiste en la segmentación de la imagen en bloques de tamaño fijo y buscar la mejor coincidencia de cada bloque con la imagen de referencia. Entonces, el bloque con mayor coincidencia será el bloque con menor error de estimación y ese bloque será seleccionado como el bloque de coincidencia. En la Ilustración 19-3 se puede observar los dos métodos empleados y sus mapas de disparidad respectivamente.

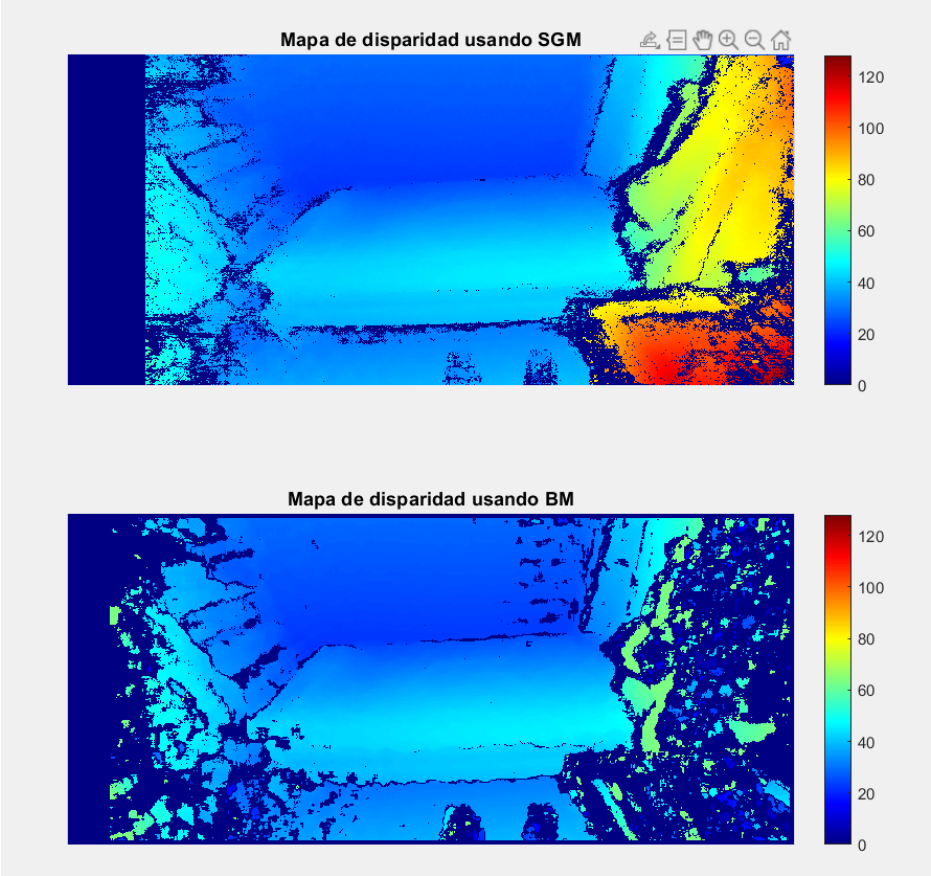


Ilustración 19-3: Mapa de disparidad mediante el método SGM y BM

Realizado por: Shagñay Nelson, 2023

Como se observa en la Ilustración 19-3, el mapa de disparidad por el método SGM da mejores resultados que el método BM por lo que se decidió usar la función *disparitySGM* para la reconstrucción de entornos. La sintaxis que provocan estos resultados se muestra a continuación.

```
frameLeftGray  = rgb2gray(frameLeftRect);
frameRightGray = rgb2gray(frameRightRect);
disparityRange = [0 128];
disparityMap1 = disparitySGM(frameLeftGray, frameRightGray);
disparityMap2 = disparityBM(frameLeftGray, frameRightGray);
```

De donde la variable *distarityRange*, guarda los rangos de disparidad mínimos y máximos, estos valores deben ser divisible para 16, enteros y hasta negativos de ser el caso. Los métodos SGM y BM requieren parámetros de entrada las imágenes, tanto derecha e izquierda, rectificadas y transformadas a escalas de grises, para esto se hace uso de la función rgbtogray*()* y se guarda en las variables *frameLeftGray* y *frameRightGray.*

3.2.2.6 Reconstrucción y segmentación del escenario

Teniendo el mapa de disparidad de un escenario en concreto, se procedió a encontrar la profundidad la cual es inversamente proporcional a la disparidad, lo que significa que, a mayor disparidad entre el par de imágenes estéreo, menor será la profundidad y viceversa. Para esto se hizo uso de la función reconstructScene*()*, la cual es una función propia de MATLAB la cual usa un mapa de disparidad y los parámetros de calibración de la cámara como parámetros de entrada y tiene como salida una matriz con las dimensiones de las imágenes rectificadas, esta matriz representa las dimensiones en el mundo de 3 dimensiones de los puntos de la escena a reconstruir. Para observar gráficamente estas dimensiones se procede a el uso de la función *pointCloud()*, esta función nos ofrece como salida una nube de puntos que puede ser observada por un visor de imágenes en 3D, para lo que se hace uso del visor *pcplayer* permitiéndonos orientas el objeto 3D y hasta ubicar en pantalla el visor. La Ilustración 20-3 nos muestra el resultado de esta función

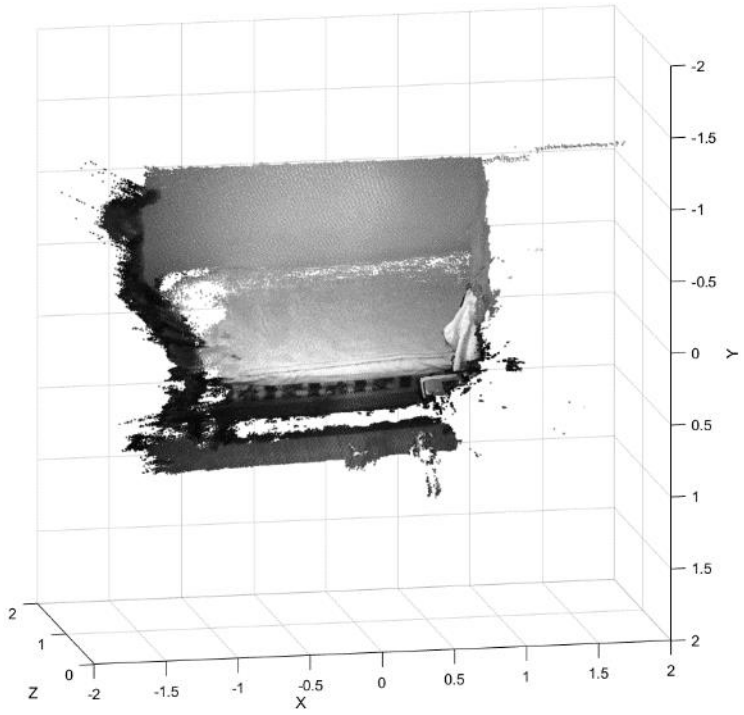


Ilustración 20-3: Nube de puntos de un escenario en 3 dimesiones

Realizado por: Shagñay Nelson, 2023

Teniendo en cuenta el par de imágenes y el escenario donde se las tomó, los procesos antes explicados permiten reconstruir gran parte de este, por lo que, no está demás recalcar que se debe tomar el tiempo necesario en el proceso de rectificación y obtención de los parámetros de calibración ya que estos procesos permiten que haya correspondencia entre pixeles del par de imágenes estéreo. Esto se hace con el fin de calcular la resta relativa entre las dos imágenes, lo que se conoce como disparidad y de esta depende los cálculos de profundidad que hace el algoritmo. La sintaxis del proceso de reconstrucción se presenta a continuación.

```
points3D = reconstructScene(disparityMap1, stereoParamsmil);
points3D = points3D ./ 1000;
ptCloud = pointCloud(points3D, 'Color', frameLeftRect);
```

En la variable *points3D* se guarda matriz resultante de la función *reconstrcutScene()* y se divide para 1000 para pasar las dimensiones guardadas en la matriz a milímetros y por último, se crea la nube de puntos con estas dimensiones gracias a la función *pointCloud()* que se almacena en la variable *ptclud*, donde, a parte la matriz de dimensiones del escenario se puede ingresar una imagen como referencia de color para el escenario, por lo que se ingresó la imagen izquierda antes de transformarla a escala de grises en el proceso de encontrar el mapa de disparidad como referencia.

Para la segmentación de la nube de puntos, se optó por el uso de la función *thresholdPC(),* la cual tiene como parámetros de entrada una matriz de límites en los ejes X, Y y Z y la nube de puntos a segmentar para luego comparar cada punto de la nube de puntos con la matriz de límites, los valores que no estén dentro de este rango se les asigna un valor de *NAN*. Se puede observar en la Ilustración 21-3 como se segmenta la nube de puntos para poder eliminar el ruido dentro de esta nube u poder obtener el escenario lo óptimo posible.

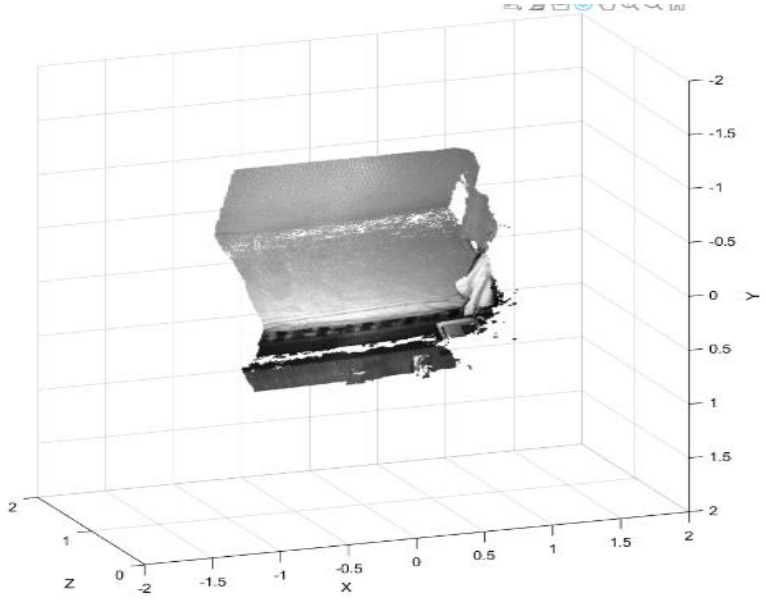


Ilustración 21-3: Nube de puntos de escenario reconstruido y segmentado para eliminar el ruido presente.

Realizado por: Shagñay Nelson, 2023

Mediante la segmentación se puede obtener una nube de puntos más limpia y poder mejorar los resultados finales. La sintaxis usada para el proceso de segmentación se presenta a continuación.

```
th = [-0.7 1;-1 1;0 2];
points3D = thresholdPC(points3D,th);
```

La matriz contenedora de los limites se guarda en la variable *th* y mediante la función *thresholdPC()* y la nube de puntos determinada en el proceso de reconstrucción permiten la segmentación de la nube de puntos la cual será guardada en la misma variable *points3D*.

Para finalizar, el proceso de reconstrucción y segmentación del escenario a reconstruir se lo repite las veces necesarias, dependiendo del par de imágenes estéreo que se tengan, esto se hace mediante el uso de un ciclo *while* y como parámetros de control se lo hace repetir mientras se va

leyendo en forma ordenada las imágenes que componen tanto el video *left* y *right*, donde la sintaxis que realiza este proceso se lo presenta a continuación:

```
while hasFrame(readerLeft) && hasFrame(readerRight)
   i=i+1;
   frameLeft = readFrame(readerLeft);
   frameRight = readFrame(readerRight);
   [frameLeftRect, frameRightRect] = ...
      rectifyStereoImages(frameLeft, frameRight, stereoParamsmil);
   frameLeftGray  = rgb2gray(frameLeftRect);
   frameRightGray = rgb2gray(frameRightRect);
    disparityMap1 = disparitySGM(frameLeftGray, frameRightGray);
   points3D = reconstructScene(disparityMap1, stereoParamsmil);
   points3D = points3D ./ 1000;
   points3D = thresholdPC(points3D,th);
   ptCloud = pointCloud(points3D, 'Color', frameLeftRect);
   ContenedorPT{i}=ptCloud;
end
```

De esta manera, se crea las nubes de puntos de cada par de imágenes estéreo que representan el escenario a reconstruir y se los guarda en la variable *Contenedor* para posterior unión de estas nubes de puntos.

3.2.2.7 Unión de nube de puntos

Una vez obtenidas todas las nubes de puntos y guardadas, se procede a unirlas, para lo cual se creó un algoritmo con diferentes funciones que nos permiten calcular la rotación y traslación de cada nube de puntos en función de la nube anterior a que se está analizando. Para este proceso se hace primero el uso de la función *pcdownsample()* que preprocesó los datos reduciendo la muestra con un filtro de cuadrícula de cuadro y estableció el tamaño del filtro de cuadrícula en 2 cm. El filtro de cuadrícula divide el espacio de la nube de puntos en cubos. Los puntos dentro de cada cubo se combinan en un solo punto de salida promediando sus coordenadas X, Y, Z, esto se hace con el fin de disminuir la carga de proceso de las nubes de puntos al unirlas debido a que contienen una gran cantidad de puntos y para el cálculo de rotación y traslación no es necesario analizar todos los puntos. La sintaxis de reducción de muestras se le muestra a continuación.

```
ptCloudRef = ContenedorPT{1};
ptCloudCurrent = ContenedorPT{2};
gridSize1 = 0.1;
fixed = pcdownsample(ptCloudRef, 'gridAverage', gridSize1);
moving = pcdownsample(ptCloudCurrent, 'gridAverage', gridSize1);
```

Primero, se guardó las primeras nubes de puntos contiguas en 2 variables diferentes las cuales son *ptcloudRef* y *ptcloudCurrent*, luego se procede a la reducción de muestras en ambas nubes de puntos dividendo las nubes de puntos en cubos 2 centímetros y se redujeron a un punto de muestra por cada cubo, esto se lo hace mediante el valor del parámetro de entrada *gridSize* este valor en metros y luego se guarda estas nuevas nubes de puntos en las variables *fixed* y *moving*. El resultado se lo muestra en la ilustración 22-3.

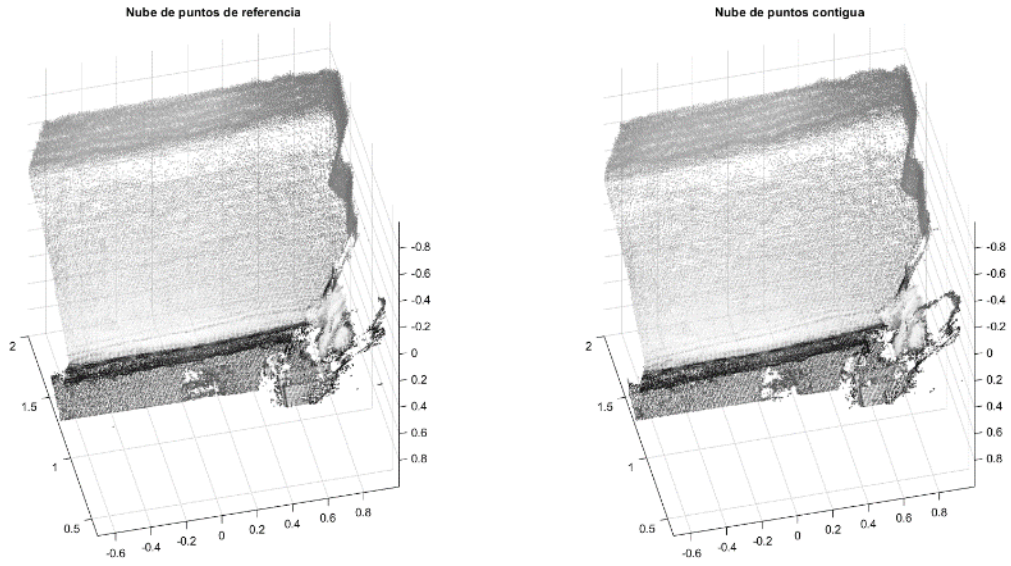


Ilustración 22-3: Reducción de numero de muestras en nubes de puntos contiguas del escenario a reconstruir

Realizado por: Shagñay Nelson, 2023

Para alinear las dos nubes de puntos, usamos el algoritmo ICP para estimar la transformación rígida tridimensional en los datos reducidos. Usamos la primera nube de puntos como referencia y luego aplicamos la transformación estimada a la segunda nube de puntos original. Necesitamos fusionar la nube de puntos de la escena con la nube de puntos alineada para procesar los puntos superpuestos. Se hace el uso de la función *pcregistericp()* para encontrar la transformación rígida para alinear la segunda nube de puntos con la primera nube de puntos y luego se hace uso de la función *pctransform()* para transformar la segunda nube de puntos en el sistema de coordenadas de referencia definido por la primera nube de puntos. Ahora se puede crear la escena unida mediante el uso de la función *pcmerge()* que une la nube de puntos de referencia y la nube de puntos contigua ya alineada a la nube de puntos de referencia donde la

región superpuesta se filtra mediante un filtro de cuadrícula de caja de 0,5 cm y se puede aumentar el tamaño de combinación para reducir el requisito de almacenamiento de la nube de puntos de la escena resultante y disminuir el tamaño del filtro de cuadrícula para aumentar la resolución de la escena. El escenario resultante se muestra en la Ilustración 23-3.

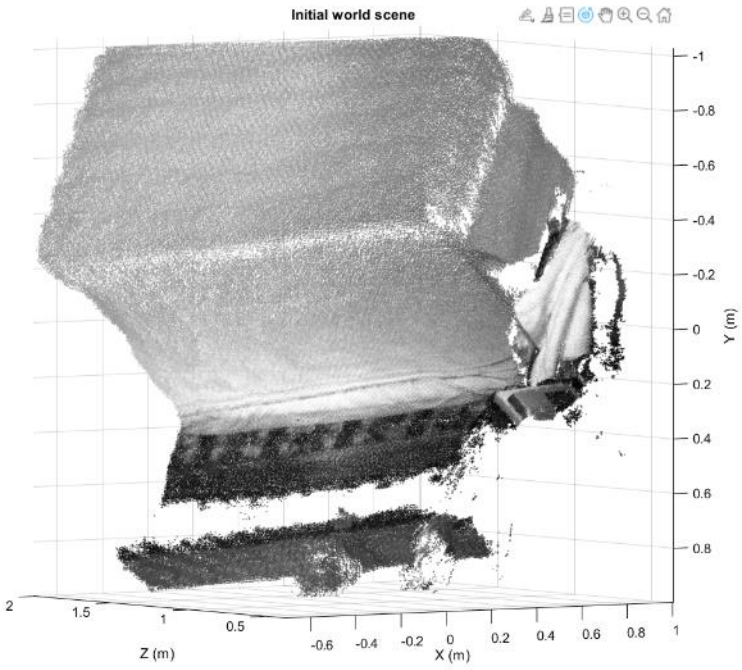


Ilustración 23-3: Unión 2 nubes de puntos contiguas.

Realizado por: Shagñay Nelson, 2023

La unión de nubes de puntos se logra mediante la sintaxis que se muestran a continuación.

```
tform = pcregistericp(moving, fixed, 'Metric','pointToPlane','Extrapolate', true);
ptCloudAligned = pctransform(ptCloudCurrent,tform);
mergeSize = 0.005;
ptCloudScene = pcmerge(ptCloudRef, ptCloudAligned, mergeSize);
```

La función *pcregister()* tiene como salida un objeto que contiene tanto la rotación y traslación calculada de la nube de puntos contigua con respecto a la nube de puntos de referencia y esta se la guarda en la variable *tform*, los parámetros de entrada son tanto la nube de puntos de referencia y la contigua. La métrica que en este caso se hiso uso de *pointToPlane* que puede reducir el número de iteraciones a procesar. Sin embargo, esta métrica requiere pasos algorítmicos adicionales dentro de cada iteración. La métrica *pointToPlane'* mejora el registro de superficies planas. Se establece también que extrapolen estas nubes de puntos con el fin trazar una ruta en el espacio de estado de registro. Mediante la función *pctransform()* se aplican los valores de rotación y traslación almacenados en la variable *tform*, esta nube de puntos

alineada se la guarda en la variable *ptCloudAligned* y mediante la función *pcmerge* se hace la unión de las nubes de puntos con un filtro de cuadrícula de 0.5 cm para guardarla en la variable *ptCloudScene*.

El proceso se repite con todas las nubes de puntos restantes y uniéndolas una a una en el escenario reconstruido, para lo cual se hace uso de un ciclo *for* va desde las siguientes nubes restantes hasta la última nube de puntos almacenada en el arreglo *Contenedor*. La sintaxis para la unión completa de nube de puntos se muestra a continuación.

```
accumTform = tform;
for i = 3:length(ContenedorPT)
   ptCloudCurrent = ContenedorPT{i};
   fixed = moving;
   moving = pcdownsample(ptCloudCurrent, 'gridAverage', gridSize1);
   tform = pcregistericp(moving, fixed, 'Metric','pointToPlane','Extrapolate', true);
   Contenedortransformada{i}=tform;
   accumTform = affine3d(tform.T * accumTform.T);
   ptCloudAligned = pctransform(ptCloudCurrent, accumTform);
   ptCloudScene = pcmerge(ptCloudScene, ptCloudAligned, mergeSize);
end
angle = -pi/6;
A = [1,0,0,0; 0, cos(angle), sin(angle), 0; 0, -sin(angle), cos(angle), 0; 0 0 0 1];
ptCloudScene = pctransform(ptCloudScene, affine3d(A));
```

Las líneas de la sintaxis se explican a continuación

- *acumTform:* sirve para guardar el sistema de coordenadas del escenario en general.
- *For:* Ciclo creado para poder aplicar el proceso de unión en cada nube de puntos restante del escenario en reconstrucción.
- *ptCloudCurrent = ContenedorPT{i}:* Reemplaza la nube de puntos contigua a la siguiente nube de puntos almacenada en la variable *Contenedor.*
- *fixed = moving:* reemplaza la nube de puntos contigua anterior almacenada en la variable *moving* en la nube de puntos de referencia y se la almacena en la variable *fixed.*
- *moving = pcdownsample(ptCloudCurrent, 'gridAverage', gridSize1):* Aplica una reducción de muestras a la nueva nube de puntos contigua.

- *tform = pcregistericp(moving, fixed, 'Metric','pointToPlane','Extrapolate', true):* aplica el algoritmo ICP a la nueva nube de puntos de referencia y contigua para calcular la rotación y traslación de la nube de puntos contigua a la de referencia.
- *Contenedortransformada{i}=tform:* Almacena los valores de traslación y rotación individual entre las nuevas nubes de referencia y la contigua por cada repetición.
- *accumTform = affine3d(tform.T * accumTform.T):* Transforma cada nube de puntos al sistema de coordenadas de referencia. Esta transformación se lo hace mediante una multiplicación de transformaciones por pares entre los valores de traslación de el sistema de coordenadas de referencia y la traslación calculada entre la nueva nube de puntos de referencia y contigua.
- *ptCloudScene = pcmerge(ptCloudScene, ptCloudAligned, mergeSize):* Une la nube de puntos acumulada con la nueva nude punto ya alineada al escenario en reconstrucción.
- *angle = -pi/6:* Ángulo de corrección del escenario reconstruido por elevación de la cámara con respecto al eje horizontal.
- *A = [1,0,0,0; 0, cos(angle), sin(angle), 0; 0, -sin(angle), cos(angle), 0; 0 0 0 1]:* Matriz de corrección del escenario para hacer que el plano de tierra sea paralelo al plano X-Z en el escenario mostrado en pantalla.
- *ptCloudScene = pctransform(ptCloudScene, affine3d(A)):* Aplicación de la matriz de corrección al escenario reconstruido.

El resultado de unir las nubes de puntos del escenario a reconstruir, que para este caso es el de un dormitorio, se muestra en la Ilustración 24.3.

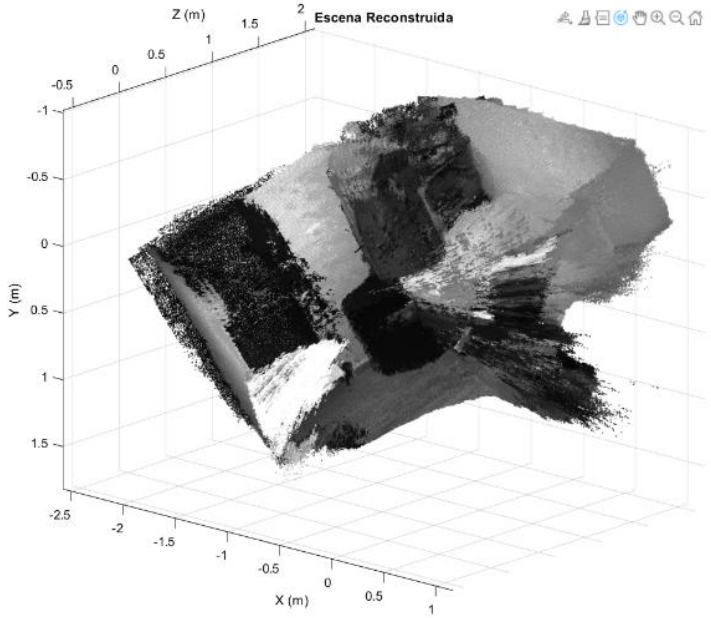


Ilustración 24-3: Escena reconstruida de un dormitorio.

Realizado por: Shagñay Nelson, 2023

3.2.2.8 Determinación de la trayectoria del robot móvil

Una vez hecha la reconstrucción del entorno, se procedió a calcular la trayectoria del robot móvil y esto se lo hiso mediante el análisis de cada nube de puntos que conformar el entorno reconstruido y el vector de traslación de cada nube con respecto al entorno. Para esto, se hiso uso de la función *pcmapndt()* la cual crea un mapa de transformación de distribuciones normales (NDT) a partir de un mapa de nube de puntos prediseñado del entorno. El mapa NDT es una representación comprimida y eficiente en memoria adecuada para la localización. El objeto convierte el mapa de nube de puntos en un conjunto de vóxeles (nombre que hace alusión al mismo concepto de los píxeles en una imagen 2D, pero en un objeto en 3D), cada uno representado por una distribución normal 3D. Se hace uso de otras funciones como la función *selectSubmap()* la cual sirve para seleccionar un mapa dentro de todo el entono y agilitar el análisis, también se hace uso de la función *findPose()* la cual localizar una nube de puntos dentro de un mapa utilizando el algoritmo de transformación de distribuciones normales (NDT) y por último se hace uso de la función *isInsideSubmap()* la cual Comprueba si la posición de consulta está dentro del submapa seleccionado. La Ilustración 25-3 muestra el resultado del proceso explicado.

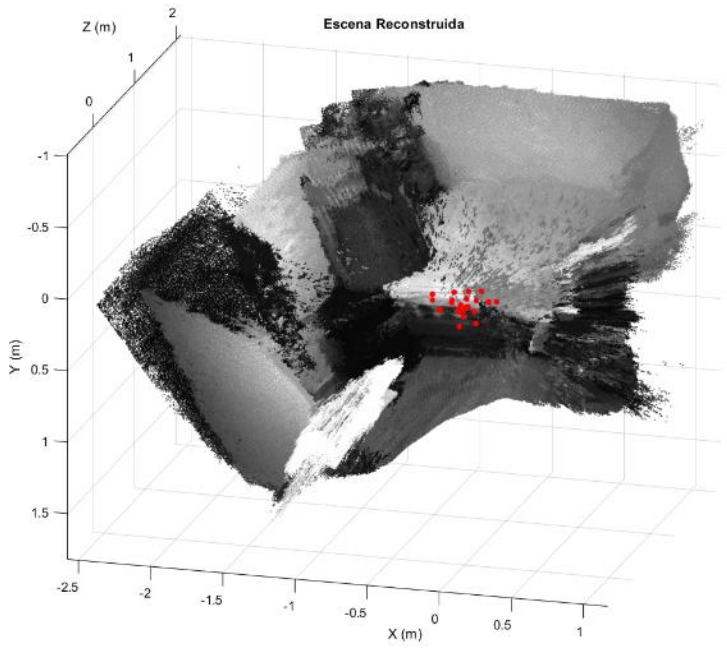


Ilustración 25-3: Trayectoria calculada del robot móvil

Realizado por: Shagñay Nelson, 2023

Para mostrar la trayectoria recorrida por el robot móvil se procedió a ilustrar con círculos de color rojo, dentro del escenario, la ubicación de la cámara al instante que se toman cada par estéreo de imágenes que conforman y que conforman la nube de puntos con el escenario reconstruido. Al irse calculando y mostrando en el escenario la ubicación de la cámara, el conjunto de puntos rojos ya nos muestra un recorrido hecho por el robot móvil.

3.3 Implementación del sistema de reconstrucción

Para la implementación, se la hizo pensando en un sistema de bajo costo, por lo que, se hizo la elección de un chasis más parecido al de la sección del diseño y que sea de fácil adquisición en las tiendas electrónicas de la ciudad de Riobamba. Luego se procedió a montar el circuito de control del robot entre las 2 placas de policarbonato que conforman el chasis del robot móvil, donde este circuito de control está compuesto de un Arduino uno, un módulo de potencia para el control de los motores de las llantas del robot móvil y un módulo Bluetooth para conectar al teléfono celular el robot móvil y poder controlarlo mediante una aplicación ya diseñada por terceros. Encima de la placa superior del chasis se monta una base cilíndrica de una altura de 30 cm para poder montar la cámara encima de la base cilíndrica, esta base, en la parte superior, tiene un desnivel de 10^0 con respecto al eje horizontal y un tornillo para poder acoplar la cámara de manera segura, tal y como se presentó en el diseño. En la Ilustración 26-3 se muestra la implementación del sistema de reconstrucción.

Ilustración 26-3: Implementación del sistema de reconstrucción

Realizado por: Shagñay Nelson, 2023

Cabe resaltar que el sistema se implementó en base al diseño y con el fin de movilizarse por entornos indoor, por lo cual no se posee motores de gran potencia y llantas para terrenos irregulares, y con un robot de no gran altura, con el fin de tomar imágenes de escenarios indoor habituales de casa comunes que no sobrepasan alturas de 2.5m.

3.4 Evaluación del sistema de reconstrucción.

Al momento de evaluar el sistema de reconstrucción, se deben establecer indicadores para la realización de pruebas estadísticas, donde se deben establecer, primeramente, las necesidades o expectativas a para evaluar el desempeño del sistema de reconstrucción implementado en la

sección anterior. En la tabla 1-3 se muestran los objetivos principales a cumplir por un sistema a partir del método de posicionamiento que usa.

Tabla 1-3: Objetivos de desempeño según el método de posicionamiento.

Objetivo Principal	Métodos de posicionamiento	Principales limitaciones
Velocidad, simplicidad	Geometría (Características Naturales)	Precisión, Características y desempeño dependen del objeto
Flexibilidad	Geometría + Objetivos (Híbrido)	Comprometido con la exactitud
Confiabilidad, precisión	Objetivos	Objetivos fijos en el objeto

Fuente: (Allard y Lavoie 2014, p. 7)
Realizado por: Shagñay Nelson, 2023

En la Tabla 1-3 se observan los 5 objetivos principales los cuales son: Velocidad, Simplicidad, Flexibilidad, Confiabilidad y Precisión, teniendo en cuenta que el sistema de reconstrucción basa el posicionamiento en la geometría, que quiere decir que al momento de reconstruir usa características naturales del entorno para hacer el proceso de reconstrucción, por lo que, según la tabla, un factor importante que se evaluó es la velocidad.

Entonces, analizando la Tabla 1-3, se tomaron en cuenta tanto la velocidad, precisión y exactitud como indicadores para la evaluación del sistema, por lo que estos indicadores se definen a continuación.

- Velocidad: Tiempo medido que toma el sistema, divido en 4 etapas principales que son a lectura de los datos tomados, creación de las nubes de puntos a partir de los pares estéreo, unión de las nubes de puntos hasta generar el escenario, determinación de la trayectoria del robot móvil. En este caso, las mediciones se tomaron en segundos.
- Precisión: Variación de medidas entre los distintos datos tomados sobre un mismo escenario al tenerlo reconstruido totalmente. Estas variaciones de mediadas se tomaron en centímetros.
- Exactitud: Variación de medidas entre los datos tomados des escenario reconstruido y el valor de medidas del escenario en el mundo real. Estas diferencias de medidas se las tomó en centímetros.

Así mismo se decidió evaluar el sistema mediante la variación de parámetros físicos del escenario como lo son la intensidad de luz, complejidad del escenario mediante la agregación de objetos gradualmente y variación de la ruta del robot móvil. Esto se hace con el propósito de saber a qué parámetros específicos, el sistema reconstruye de mejor manera o se desenvuelve mejor.

Estos resultados se los muestra y analiza de forma detallada en el Capítulo 4.

3.4.1 Configuración de parámetros de la imagen

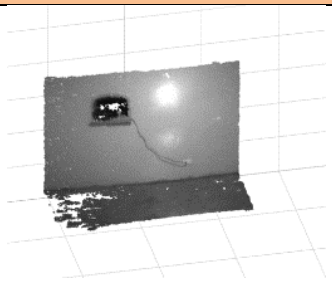
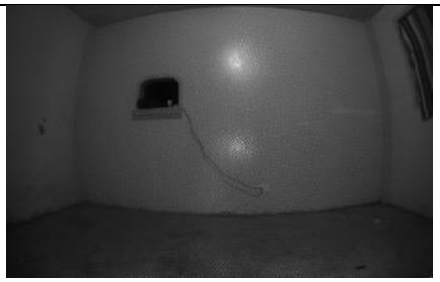
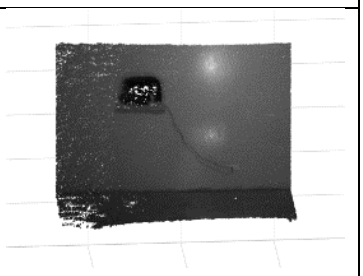
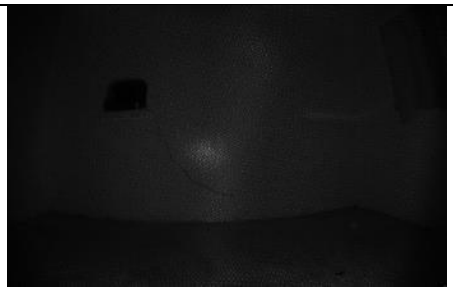
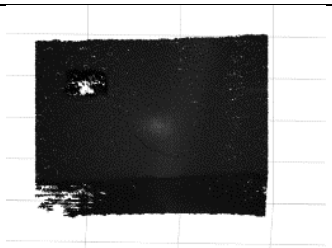
Para mejorar la reconstrucción de un entorno se debe determinar los mejores parámetros para la captura de la imagen, de los cuales se los puede manipular ya sea mediante las cámaras como son los parámetros de brillo, contraste e infrarrojo, así como parámetros externos, como lo es la intensidad de luz. Entonces, para validar los valores correctos de brillo, contraste, infrarrojo e intensidad luminosa nos centramos en la etapa final de reconstrucción.

Para identificar los valores adecuados dependiendo de los parámetros ya definidos, se hizo mediante la captura de imágenes variando cada parámetro a la vez.

3.4.1.1 Determinación del parámetro de intensidad luminosa

Para determinar el valor adecuado de intensidad luminosa para una mejor reconstrucción se debe saber que este valor se mide en lux, y al tratarse de entornos indoor se tomó en cuenta la variación de este parámetro para la captura de imágenes en torno a un entorno luminoso debido a un foco incandescente, entorno con poca intensidad luminosa debido a un foco ahorrador y, finalmente, un lugar sin luz. Se debe determinar los valores de infrarrojo, brillo y contraste por defecto. Los valores de lux en cada caso fueron medidos gracias una herramienta instalada en el móvil llamada medidor de luz. Estos valores y sus resultados se presentan el Tabla 2-3.

Tabla 2-3: Variación de intensidad luminosa

Intensidad luminosa (lux)	Imagen	Nube de puntos
420		
190		
0		

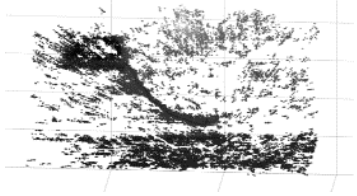
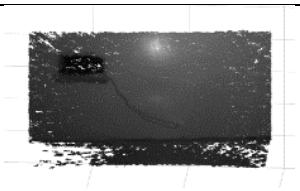
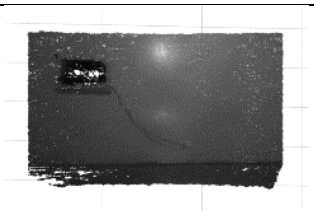
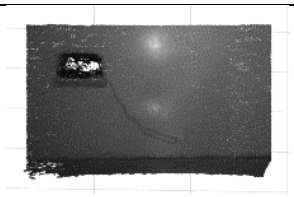
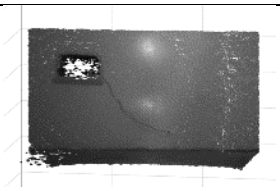
Realizado por: Shagñay Nelson, 2023

De los resultados obtenidos mediante la Tabla 2-4, se puede determinar que para una mejor reconstrucción se debe hacer la captura de las imágenes con una intensidad luminosa media, es decir a 190 lux.

3.4.1.2 Determinación del parámetro de infrarrojo

Para determinar el valor adecuado de infrarrojo se lo hizo mediante la variación de este parámetro que puede ir desde 0 hasta 160, este parámetro es fundamental al reconstruir debido que agrega más características a las imágenes que nos sirven al momento de la rectificación y el procesado de las imágenes. Este distinguir la variación de este valor, se hizo mediante saltos de 40 puntos, con valores de intensidad luminosa de 190 lux y demás parámetros por defecto. La Tabla 3-3 nos muestra los resultados de estas variaciones.

Tabla 3-3: Variación del parámetro de Infrarrojo

Infrarrojo	Imagen	Nube de puntos
0		
40		
80		
120		
160		

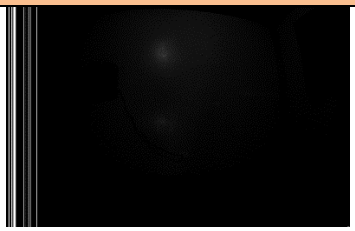
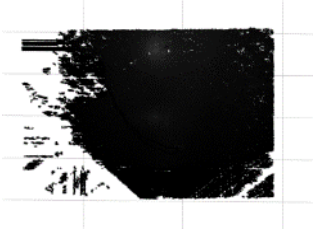
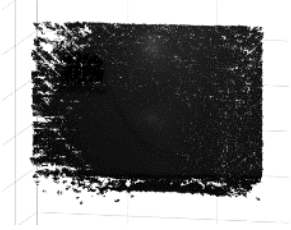
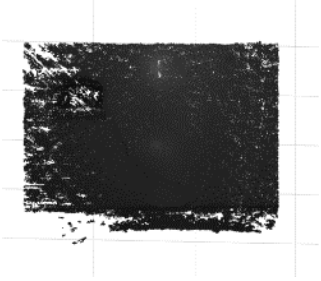
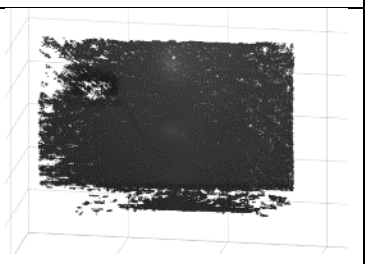
Realizado por: Shagñay Nelson, 2023

Según los resultados obtenidos en la Tabla 3-3, se decidió por el valor de 120 para infrarrojo, ya que nos da resultados mejores que con los demás valores infrarrojo.

3.4.1.3 Determinación del parámetro de contraste

Para determinar el valor correcto de contraste, se debe saber que el contraste ajusta la diferencia entre tonos claros y obscuros de una imagen. Este valor se lo puede variar desde 0 hasta 254 y para observar su variación se hicieron pasos de 85. Estos resultados se muestran en la Tabla 4-3.

Tabla 4-3: Variación del parámetro de contraste

Contraste	Imagen	Nube de puntos
0		
85		
170		
254		

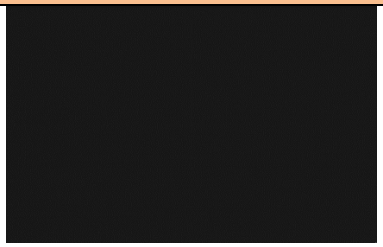
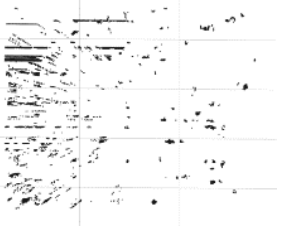
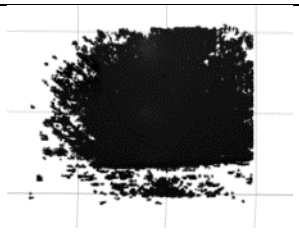
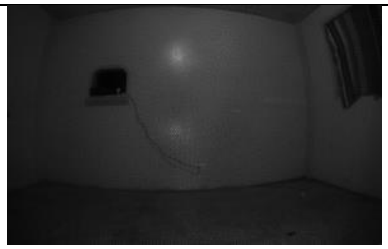
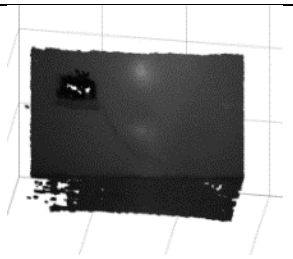
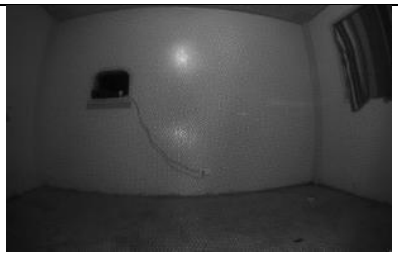
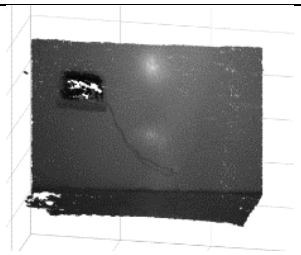
Realizado por: Shagñay Nelson, 2023

Como se observa en la Tabla 4-3, se pude decir que a mayor contraste mejor es la reconstrucción, por lo que se selecciona el valor de 254 como valor por defecto para la reconstrucción.

3.4.1.4 Determinación del parámetro de brillo

Para determinar el valor del brillo que se ajuste a una buena reconstrucción se vario este parámetro que va desde 0 hasta 240, por lo que varió en pasos de 80 para poder determinar la diferencia entre valores. Estos resultados se muestran en la Tabla 5-3.

Tabla 5-3: Variación del parámetro de brillo

Brillo	Imagen	Nube de puntos
0		
80		
160		
240		

Realizado por: Shagñay Nelson, 2023

Como podemos observar, a mayor brillo, la reconstrucción cobra mejor calidad, por lo que se optó por hacer uso del valor máximo de brillo, en este caso de 240.

En resumen, se optó por los valores de 120 en brillo, 190 en intensidad luminosa, 254 en contraste y 240 en brillo, con esos valores se garantiza la mayor calidad posible al momento de la toma de imágenes.

3.4.2 Determinación de los parámetros de calibración

Como se observó en la etapa de calibración, los parámetros se calculan mediante la herramienta de *stereo visión* por lo que aquí se muestran los resultados de ese proceso, para lo cual se identificó la cámara según su código identificador y se tiene como código clave a la cámara con

el valor de 21. La Tabla 6-3 nos muestra las matrices: fundamental, esencial, rotación, traslación y error de re-proyección.

Tabla 6-3: Parámetros de calibración para cámara 21

Cámara	Matriz fundamental	Matriz esencial	Matriz rotacional	Matriz de traslación	Error de re-proyección
21	-3.64954316507 31e-09 -4.71470415577 74e-06 0.00036283999 2495049 7.35536136199 094e-06 -2.44242136761 17e-07 0.33040231147 8265 -0.00082716071 380672 -0.33096207887 169 -0.17525639759 121	-0.00047782704 5805467 -0.61737894150 8668 -0.28977307156 8478 0.96421019518 7865 -0.03155211412 68698 120.56 0160099459 0.35895168675 6924 -120.562247785 020 -0.03294021359 86423	0.99999569842962 4 -0.00057516329181 4209 0.0028761 6227574103 0.00057592685086 9575 0.9999997 99132381 -0.00026465724955 2481 -0.00287600947687 831 0.0002663 12570192759 0.9999958 28814853	-120.56243879 6195 -0.2896098433 66878 0.6174 55712880116	0.43909666 3520233

Realizado por: Shagñay Nelson, 2023

3.5 Mantenimiento del sistema de reconstrucción y mejoras.

Para el mantenimiento del robot, se tomó en cuenta la autonomía del sistema, la cual se basa en una batería LiPo compuesto por 2 celdas, la cual nos da un voltaje de salida de 7.4 voltios y un promedio de descarga de 1600 mA por hora, la cual, mediante pruebas nos da un promedio de uso de trabajo de 2 horas continuas antes de volver a cargar la batería por lo que n el mantenimiento se debe tener en cuenta el estado de la batería así como su tiempo de vida que en promedio se promete hasta 300 ciclos de carga. Cuenta con un peso de 1.2 kg por lo que es un sistema portable y de fácil implementación, ya que el único elemento extraíble es la cámara estéreo.

Otro punto importante es el uso de las cámaras, que para darle mantenimiento se recomienda el hacer la calibración de esta con un periodo de 1 vez cada 6 meses con el fin de corregir error que se pueden presentar por su uso continuo y alguna alteración de los parámetros de esta.

Para mejor el sistema se debe tener en cuenta la variedad de baterías recargables y su método de recarga, ya que la batería LiPo cuenta con su propio cargador el cual tiene su sistema de plugs de carga y solo sirve para baterías LiPo, por lo que se puede optar por un cambio en baterías con carga universal como podrían ser tipo micro USB y tipo C. La cámara estéreo también supone un punto de mejora del sistema ya que hay distintos tipos de cámaras estéreo que ofrecen mejor resolución y con captura de imágenes a color, que hasta permiten compatibilidad con la herramienta de MATLAB.

CAPITULO IV

4 RESULTADOS, DISCUSIÓN Y ANÁLISIS

Los resultados que se muestran van acorde a los indicadores de evaluación ya explicados y acorde al número de pruebas que, en este caso, se establecieron a 30 pruebas comparativas simples. En dichas pruebas se tomaron mediciones de los parámetros de funcionamiento del sistema de reconstrucción en base a los resultados obtenidos, para posteriormente ser contrastados con medidas entre los diferentes números de pruebas y también se los contrastó con medidas del entorno real.

Con lo anterior indicado, se planteó la estimación de una media poblacional en cada proceso que logra realizarlo con éxito el sistema de reconstrucción, para lo cual con 30 pruebas se logra hacer la medición correcta de los indicadores planteados, y posteriormente, permitir su evaluación a través de la prueba T de Student que se basa en el contraste de dos muestras pareadas y normales. Las pruebas se las realizaron en el escenario que se muestra en la Ilustración 1-4.

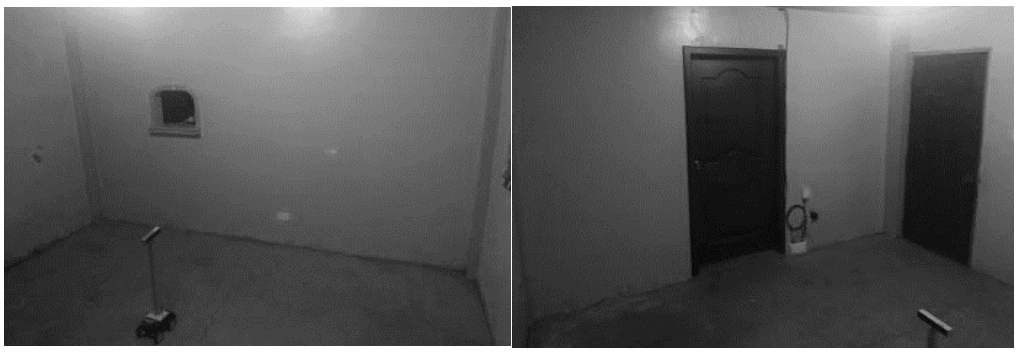


Ilustración 1-4: Escenario seleccionado para la elaboración de pruebas

Realizado por: Shagñay Nelson, 2023

El escenario se reconstruyó de forma normal, así como lo podemos observar en la Ilustración 2-4, donde podemos observar el escenario y la trayectoria del robot al momento de tomar las imágenes.

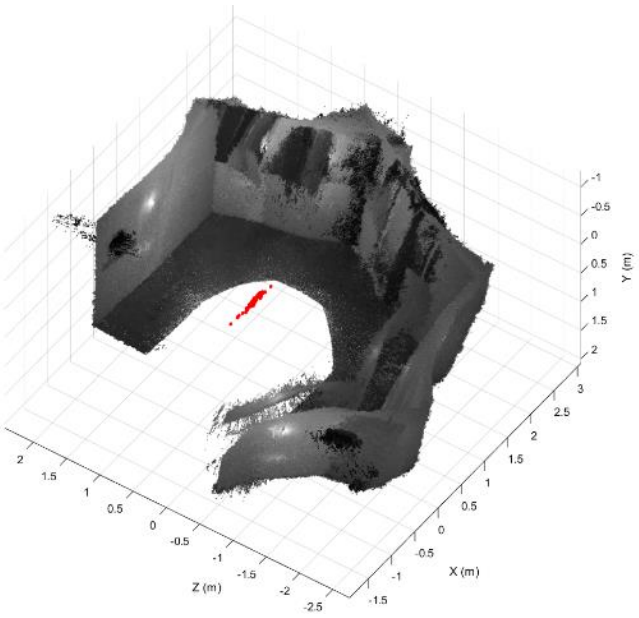


Ilustración 2-4: Escena reconstruida y trayectoria calculada

Realizado por: Shagñay Nelson, 2023

Este entorno se escogió debido a sus mejores resultados visuales al momento de reconstruir ya que es un cuarto vacío y que, al final, no provoca mucho ruido comparado con el mismo entorno, pero con agregación de objetos que complican la reconstrucción.

4.1 Post-procesamiento de la nube de puntos

Como se observó, se tiene una escena con mucho ruido por lo que se procedió a eliminar la mayoría de ruido posible mediante la herramienta llamada MeshLab. Para esto, se tuvo que exportar el escenario como un archivo PLY, una extensión de archivos para programas de diseño CAD.

4.1.1 Procesamiento dentro de MeshLab

Se procedió a importarlo dentro de la herramienta MeshLab, para eso nos fuimos a la pestaña: File → Import Mesh, navegamos hasta el directorio contenedor del archivo PLY y seleccionamos abrir, como lo muestra la Ilustrración 3-4.

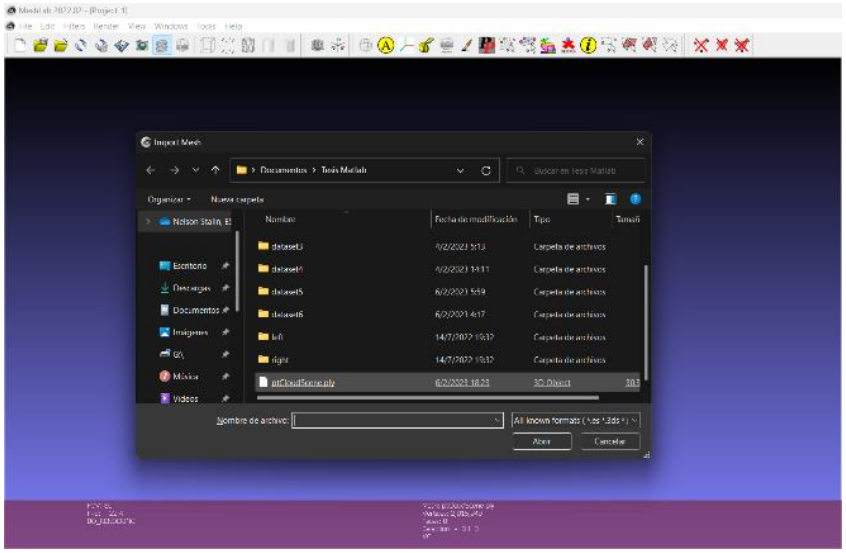


Ilustración 3-4: Importación de archivo PLY a MeshLab

Realizado por: Shagñay Nelson, 2023

Ya abierto el archivo, se procede a eliminar el ruido dependiendo de la persona y evitando eliminar parte del escenario que no es ruido, por lo cual, se debe hacer un análisis minucioso de la nube de puntos generada y reconocer los datos que se dispersan demasiado. Para hacer esto se usa la herramienta: Select Vertexes y se selecciona de forma manual. Tal y como se ve en la Ilustración 4-4.

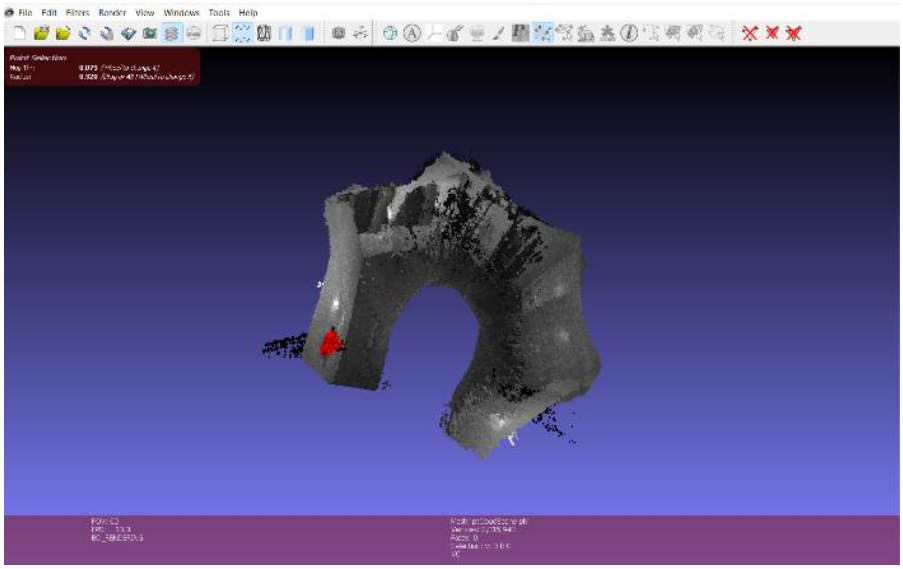


Ilustración 4-4: Selección de datos dispersos al escenario

Realizado por: Shagñay Nelson. 2023

Ya teniendo seleccionados los datos dispersos, se procede a eliminarlos utilizando la herramienta Filter Select. Esto lo podemos observar en la Ilustración 5-4.

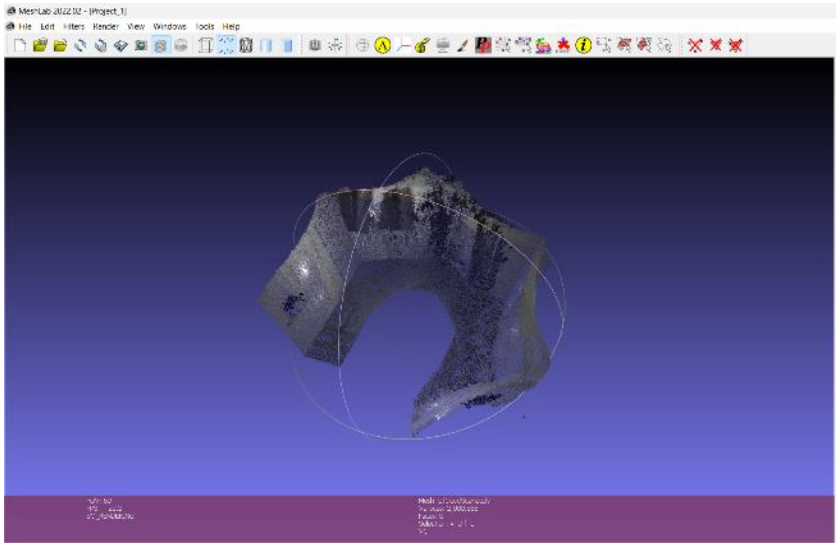


Ilustración 5-4: Eliminación de puntos dispersos

Realizado por: Shagñay Nelson, 2023

Siguiendo con el proceso de la nube, se deben aplicar cálculos de normales a los puntos en la nube, ´para lo que se ingresa a la pestaña: Filters → Normals, Curvatures and Orientations → Compute normals for point sets. Esto lo podemos ver en la Ilustración 6-4.

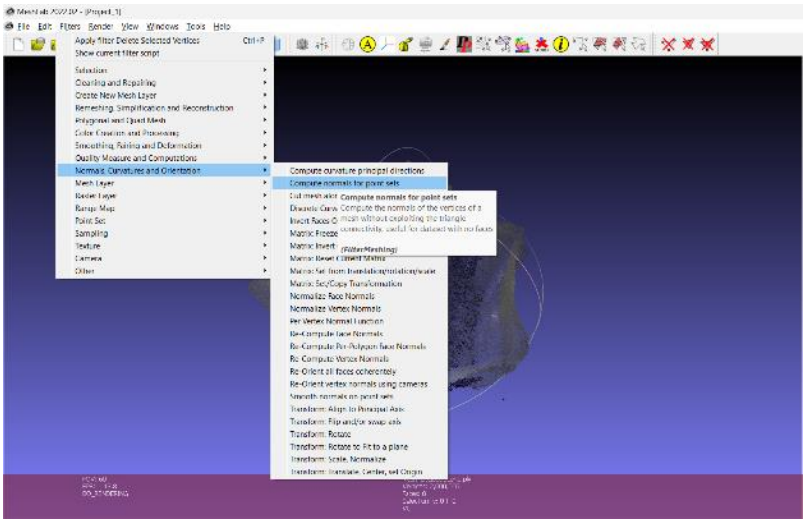


Ilustración 6-4: Cálculo de normales para los puntos de la nube

Realizado por: Shagñay Nelson, 2023

A continuación, se procede a calcular con los parámetros por defecto. Los resultados se los puede observar en la Ilustración 7-4.

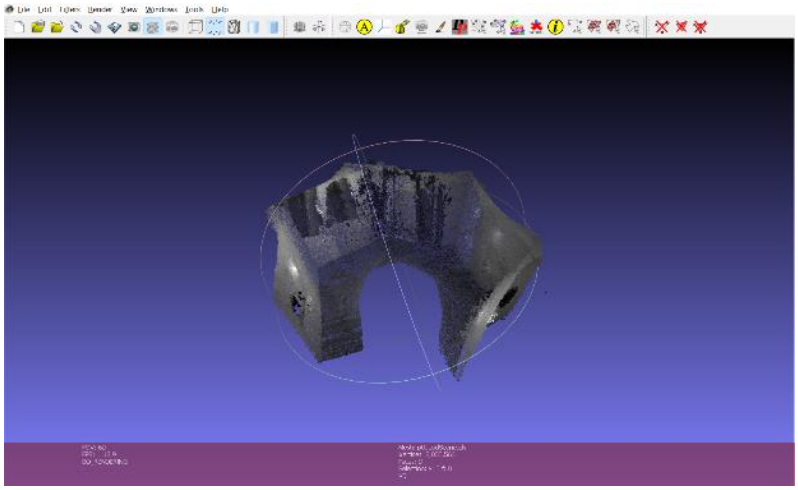


Ilustración 7-4: Resultados de los cálculos de las normales a los puntos

Realizado por: Shagñay Nelson, 2023

Prosiguiendo, se tienen que muestrear los puntos, para lo cual, nos vamos a la pestaña: Filters → Sampling → Point cloud simplification donde se aplican los parámetros por defecto. La Ilustración 8-4 nos muestra los resultados de la simplificación de puntos.

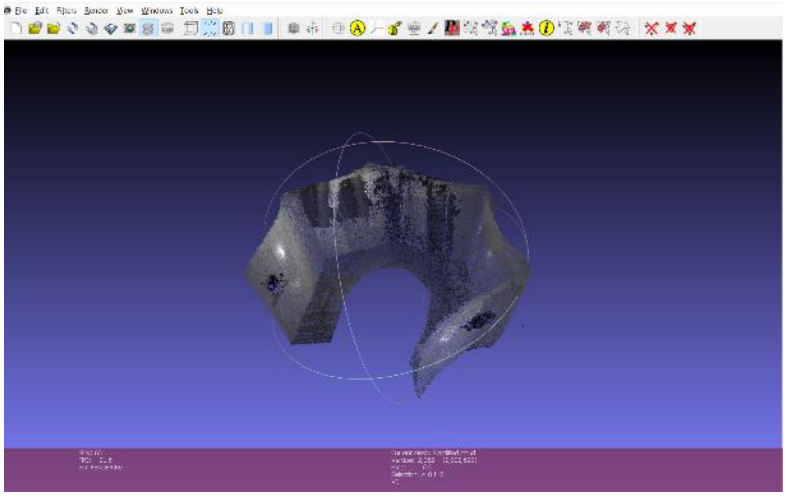


Ilustración 8-4: Simplificación de puntos

Realizado por: Shagñay Nelson, 2023

Se procede a recalcular las normales a los puntos, como se muestra en la Ilustración 9-4.

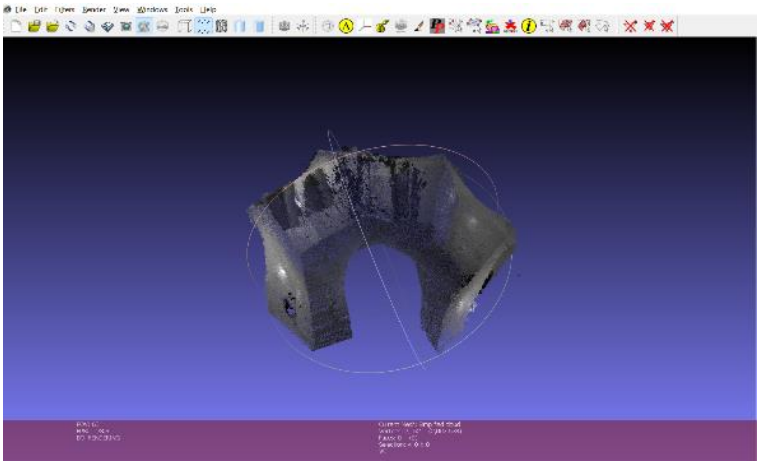


Ilustración 9-4: Recalculo de las normales de los puntos

Realizado por: Shagñay Nelson, 2023

Para Finalizar, se tiene el escenario filtrado lo máximo posible que mejoran los resultados obtenidos después de aplicar el algoritmo de reconstrucción en MATLAB.

4.2 Pruebas de aleatoriedad del sistema

Al momento de realizar las distintas pruebas, con la meta de determinar los tiempos de ejecución del algoritmo de reconstrucción en MATLAB; se hace uso del escenario mostrado al inicio del capítulo y se realizan las mediaciones 30 gracias a la función *tik tok()* de MATLAB, el cual nos muestra los tiempos de ejecución en la ventana de comandos.

4.2.1 Mediciones de tiempo requerido por etapa

En esta evaluación se lo hizo por etapas para poder determinar de mejor manera el algoritmo, por lo que se dividen en 4 etapas, la de importación de datos a MATLAB, Rectificación de imágenes y generación de nubes de puntos, Unión de nubes de puntos y Determinación de la trayectoria del robot móvil.

4.2.1.1 Tiempo de importación de datos a MATLAB

Este tiempo hace referencia a la lectura de los datos del directorio señalado, lectura de las imágenes, creación de los videos contenedores de las imágenes.

Tabla 1-4: Tiempos capturados para la importación de datos a MATLAB

No de Pruebas	*Tiempo (seg)*
1	15,432863
2	16,857281
3	14,843271
4	14,358690
5	14,320129
6	14,180019
7	14,184175
8	16,695040
9	14,292843
10	14,792768
11	14,719639
12	14,696164
13	14,562380
14	14,790960
15	14,849611
16	14,574031
17	14,822353
18	25,288919
19	13,698186
20	13,584509
21	13,528411
22	15,443732
23	13,578932
24	13,675876
25	13,856258
26	13,388428
27	13,460387
28	13,532261
29	13,530062
30	13,528626
PROMEDIO	**14,768893**

Realizado por: Shagñay Nelson, 2023

En este proceso se obtuvo un promedio de 14.77 seg, lo cual es una etapa de lectura de datos que toma poco tiempo ya que no hay un alto proceso que implica la lectura de datos.

4.2.1.2 Tiempo de rectificación y creación de nubes de puntos

Este tiempo hace referencia desde la importación de los parámetros de calibración, rectificación de imágenes, cálculo del mapa de disparidad y creación de las nubes de puntos por para de imágenes estéreo.

Tabla 2-4: Tiempos de rectificación y creación de nubes de puntos

No de Pruebas	*Tiempo (seg)*
1	31,931688
2	32,962964
3	30,107937
4	33,129874
5	29,373201
6	29,816848
7	29,131704
8	32,637000
9	29,542941
10	30,249144
11	32,074798
12	31,085433
13	31,841390
14	30,664561
15	30,586076
16	31,256568
17	30,847102
18	32,034865
19	23,742497
20	23,686676
21	24,236917
22	24,386929
23	23,304608
24	23,572255
25	23,724498
26	24,074813
27	23,507313
28	23,753691
29	23,795003
30	24,115534
PROMEDIO	**28,172494**

Realizado por: Shagñay Nelson, 2023

El promedio de tiempo de esta etapa es de 28.17 seg, que indican que la creación de las nubes de puntos no lleva mucho tiempo de procesado.

4.2.1.3 Tiempo de unión y costura de las nubes de puntos

En este proceso se toma en consideración la unión de las diferentes nubes de puntos mediante el algoritmo ICP y el cálculo de las matrices de rotación y traslación entre la nube de referencia y la nube subsiguiente.

Tabla 3-4: Tiempos de unión y costura de las nubes de puntos.

No de Pruebas	*Tiempo (seg)*
1	39,164048
2	39,763594
3	37,948650
4	41,327886
5	37,300571
6	38,011584
7	36,825359
8	37,945047
9	39,508549
10	38,478127
11	38,859387
12	38,890336
13	39,082884
14	38,150861
15	38,652062
16	39,822472
17	38,364837
18	32,664194
19	31,200272
20	30,891564
21	30,792790
22	30,953290
23	30,616104
24	31,295520
25	30,830826
26	30,951313
27	31,033779
28	30,996241
29	30,769082
30	31,099618
PROMEDIO	**35,406362**

Realizado por: Shagñay Nelson, 2023

El promedio de tiempo que conlleva esta etapa es de 35.40 seg, lo que nos indica que tampoco lleva mucho tiempo la unión y costura de las nubes de puntos para recrear el entorno indoor.

4.2.1.4 Tiempo de cálculo de la trayectoria del robot móvil

Esta etapa se ve involucrado el proceso de submapeo del entorno ya reconstruido y el cálculo de la posición del robot móvil al momento de la captura del par de imágenes estéreo.

Tabla 4-4: Tiempos de cálculo de la trayectoria del robot móvil

No de Pruebas	***Tiempo (seg)***
1	566,969898
2	565,364134
3	568,241035
4	583,702809
5	566,591051
6	563,543864
7	570,557344
8	557,860804
9	588,569471
10	584,086352
11	594,279657
12	587,941385
13	586,714977
14	587,206189
15	583,476209
16	583,039859
17	562,322595
18	488,791339
19	470,485614
20	466,903234
21	469,816982
22	467,291646
23	468,359648
24	468,552264
25	467,501232
26	468,498841
27	466,928334
28	466,670569
29	469,375463
30	469,879734
PROMEDIO	**530,317418**

Realizado por: Shagñay Nelson, 2023

Esta etapa es tiene un promedio de tiempo de 530.32 seg, que equivale a 8.84 minutos, lo que indica que es la etapa que más procesamiento de los datos conlleva.

4.2.2 Tiempo total estimado de importación y procesamiento de las imágenes

Mediante la suma de los tiempos de cada etapa implicada en la reconstrucción del entorno antes planteadas, se puede llevar a estimar el tiempo en promedio en que se demora el sistema en reconstruir un en entorno, el cual variará dependiendo del entorno a reconstruir, pero nos da una idea de velocidad de procesamiento del sistema. Donde el tiempo total se lo calcula de la siguiente manera:

$t_{total} = t_{importación} + t_{creación} + t_{unión} + t_{cálculo\_de\_trayectoria}$

$t_{total} = 14.77 + 28.17 + 35.40 + 530.32$

$t_{total} = 608.66$ seg

$t_{total} = 10$ min 14 seg

Así se determina de un promedio de 10 minutos y 14 segundos para la reconstrucción de un entorno.

4.2.3 Pruebas de precisión del sistema

Para el análisis de resultados en función de la precisión del sistema, si hace un contraste de precisión de los resultados obtenidos por el sistema y los resultados obtenidos mediante el uso de un software externo llamado Colmap, que tiene la capacidad de generar nubes de puntos aplicando el método de la fotogrametría.

Una vez obtenidos las nubes de puntos, se los hace un post procesamiento en la herramienta Meshlab, con lo que se mejora la nube de puntos eliminando ruido y también se tienen herramientas de medición de longitudes de los objetos. Para esta comparativa se hizo de mediciones del perímetro reconstruido de un mismo escenario repetidamente y mediante las herramientas de medición de Meshlab se tomaron los datos en ambos casos y se calcula tanto la media de medidas entre las pruebas y la desviación estándar como medida de prueba de precisión de los datos, para luego hacer un contraste entre ambos resultados.

4.2.3.1 Cálculo de medidas obtenidas entre Colmap y el prototipo desarrollado

Una vez realizadas las pruebas, los resultados se muestran, a continuación, en la Tabla 5-4

Tabla 5-4: Medidas de perímetros, del escenario según el sistema y Colmap

No pruebas	Matlab (m)	Colmap (m)
1	10,5885	14,44039
2	9,93076	14,01645
3	9,87666	14,95431
4	9,87533	15,2146
5	10,33583	14,26154
6	10,23851	13,98154
7	9,89954	14,9142
8	9,86928	13,51343
9	10,225234	13,87496
10	9,26761	14,52486
11	9,6751	14,6743
12	10,049	14,1061
13	10,09126	14,6347
14	9,74726	14,43501
15	9,883291	15,4783
16	11,085797	14,6512
17	10,25774	14,37916
18	9,26133	14,1473
19	9,605101	14,4278
20	9,60958	14,26107
21	9,40864	13,8461
22	10,69234	13,9514
23	10,67184	14,53551
24	10,03004	14,81173
25	10,24292	15,24078
26	9,89905	15,8994
27	9,79873	12,8974
28	9,707995	14,23223
29	10,02457	14,89745
30	9,5161	15,23116
Promedio	9,978831267	14,481146
Desviación estándar	0,421681568	0,611988994

Realizado por: Shagñay Nelson, 2023

Como podemos observar, en cuanto a precisión individual, la desviación estándar dice cuan dispersos se encuentran las mediciones entre sí. Entonces el sistema tiene una desviación estándar de 0.42168, cual significa que la mayoría de los datos se encuentran dispersos ±0.42168 metros con respecto a la media de 9.978831 metros. Lo mismo con los valores tomados en Colmap.

Entonces, al ver la diferencia de medidas entre el sistema desarrollado y la herramienta Colmap, se hizo un contraste mediante la comparación de medidas, de la siguiente manera.

Se busca comprobar si la precisión del sistema desarrollado se asemeja o no por la herramienta Colmap que es una herramienta de código abierto por lo que mucha gente colaboró en el desarrollo de esta herramienta y, por lo tanto, sus resultados son aceptables y sirve para hacer una buena comparativa en busca del porcentaje de confianza del sistema.

Primeramente, se toman los datos del sistema desarrollado y Colmap para su procesado mediante los siguientes pasos.

1. Establecimiento de la hipótesis nula e hipótesis alternativa:
   H_0: μS - μC = 0 (La diferencia entre el sistema y Colmap es nula o no significativa)
   H_1: μS < μC (La media del perímetro medido por Colmap es mayor a la media del perímetro medido por el sistema).

2. Establecer el requerimiento de los datos para realizar la prueba T para el contraste de pruebas en muestras pareadas:
   - Normalidad de los datos: Los datos deben ajustarse a una distribución normal.
   - Igualdad de varianza (homcedasticidad): Los datos pueden, o no, cumplir la igualdad de varianza, pero, se debe comprobar el valor de esta condición, ta que esta sirve como parámetro de entrada en la prueba T

3. Determinar la normalidad de los datos a través de una prueba de Shapiro dentro de Excel y usando la herramienta *XrealStats*, como se puede observar en la Tabla 6-4.

Tabla 6-4: Prueba de Shapiro entre los datos del sistema y Colmap

Shapiro-Wilk Test		
	Sistema	Colmap
W-stat	0,9649223	0,985383849
p-value	0,41100158	0,943538086
alpha	0,05	0,05
normal	yes	yes

Realizado por: Shagñay Nelson, 2023

4. De los datos arrojados en el paso 3, el valor más importante a analizar es el p-value (probabilidad que los datos obtenidos sean estadísticamente significativos)

- Se determina un que si el p-value va sobre el 0.05, los datos se distribuyen de forma normal, y si p-value es menor a 0.05 se determina que los datos no tienen una distribución normal.
- El p-value en los datos del sistema desarrollado es de 0.411, por lo tanto, los datos del sistema tienen una distribución normal.
- El p-value en los datos del sistema desarrollado es de 0.9435, por lo tanto, los datos del sistema tienen una distribución normal.

5. Al cumplirse que los datos tienen una distribución normal, se procede a determinar si las varianzas de sus poblaciones son iguales a través de la prueba F, para esto se lo hace en Excel y se lo representa en la tabla 7-4.

Tabla 7-4: Prueba F para determinar la igualdad de varianza en las colecciones de datos

F-Test Two-Sample for Variances		
	Sistema	*Colmap*
Mean	9,9788313	14,481146
Variance	0,1778153	0,374530529
Observations	30	30
df	29	29
F	0,4747686	
P(F<=f) one-tail	0,0246226	
F Critical one-tail	0,5374	

Realizado por: Shagñay Nelson, 2023

6. Analizando el resultado obtenido en el paso 5, se determina que, en valor F es menor que el valor crítico de F, por lo que se acepta la hipótesis nula donde se acepta la igualdad de variancia.

7. Ya conociendo la igualdad de varianzas, se procede a hacer la prueba T para medias de muestras pareadas. Estos resultados se observan en la Tabla 8-4.

Tabla 8-4: Prueba T en Excel para datos del sistema y Colmap

T Test: Two Paired Samples					
SUMMARY			Alpha	0,05	
Groups	*Count*	*Mean*	*Std Dev*	*Std Err*	*t*
Matlab (m)	30	9,978831267	0,421682		
Colmap (m)	30	14,481146	0,611989		
Difference	30	-4,50231473	0,740626	0,13521925	-33,296404
T TEST					
	p-value	*t-crit*	*lower*	*upper*	*sig*
One Tail	5,8057E-25	1,699127027			yes
Two Tail	1,1611E-24	2,045229642	-4,77887	-4,2257603	yes

Realizado por: Shagñay Nelson, 2023

8. Analizando los resultados, se tiene que el valor p es menor que 0.05 por lo que se rechaza la hipótesis nula y se acepta la hipótesis alternativa, queriendo decir que hay menor promedios entre las medidas del sistema y Colmap.

Además, mediante la Ilustración 10-4 se puede observar la dispersión de datos obtenidos por sistema y los datos obtenidos por Colmap.

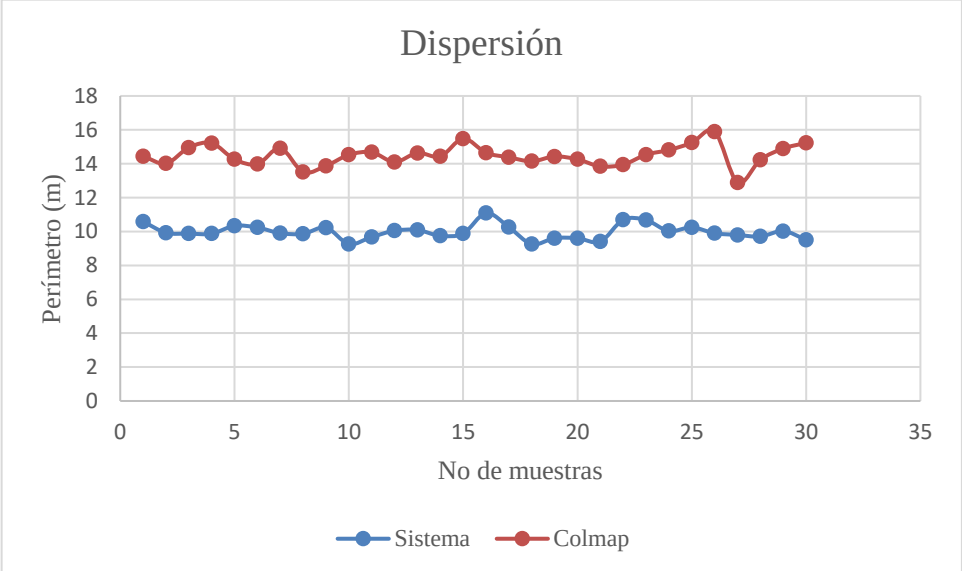


Ilustración 10-4: Dispersión entre el conjunto de datos del sistema y Colmap

Realizado por: Shagñay Nelson, 2023

4.2.3.2 Cálculo de error absoluto entre los datos obtenidos por el sistema y Colmap

Se determinó las diferencias de errores entre las medidas tomadas por el sistema desarrollado y Colmap para hallar el error absoluto. Estos valores se los muestra en la Tabla 9-4.

Tabla 9-4: Cálculo de error absoluto

No pruebas	Error entre sistema y Colmap (m)
1	3,85189
2	4,08569
3	5,07765
4	5,33927
5	3,92571
6	3,74303
7	5,01466
8	3,64415
9	3,649726
10	5,25725
11	4,9992
12	4,0571
13	4,54344
14	4,68775
15	5,595009
16	3,565403
17	4,12142
18	4,88597
19	4,822699
20	4,65149
21	4,43746
22	3,25906
23	3,86367
24	4,78169
25	4,99786
26	6,00035
27	3,09867
28	4,524235
29	4,87288
30	5,71506
Media muestral	4,502314733
Desviación estándar	0,740626317

Realizado por: Shagñay Nelson, 2023

Analizando los resultados obtenidos se tiene que:

- Hay un promedio de error entre medidas de 4.5 metros, lo cual difiere bastante y da baja confianza a sistema en comparación con Colmap.
- La desviación estándar es de 0.74 metros, lo que quiere decir que de la media de error que es 4.5 m, la mayoría de los datos se encuentran en ese rango entre ± 0.74 m

4.2.3.3 Cálculo de error porcentual entre los datos obtenidos por el sistema y Colmap

La Tabla 10-4 presenta el error porcentual para las mediciones tomadas.

Tabla 10-4: Cálculo de Error porcentual

No pruebas	Error entre sistema y Colmap (%)
1	26,67441807
2	29,14924963
3	33,95442518
4	35,0930685
5	27,52655043
6	26,77122835
7	33,62339247
8	26,96687666
9	26,30440736
10	36,19484112
11	34,06772384
12	28,76131603
13	31,04566544
14	32,47486493
15	36,14743867
16	24,33522851
17	28,66245316
18	34,53641331
19	33,42643369
20	32,61669706
21	32,04844686
22	23,36009289
23	26,58090428
24	32,28312965
25	32,79267859
26	37,73947445
27	24,02554003
28	31,78865856
29	32,70949055
30	37,52215852
Media muestral	30,97277556
Desviación estándar	4,100948034

Realizado por: Shagnay Nelson, 2023

Analizando los resultados, se puede decir que:

- El error porcentual medio de los datos obtenidos por el sistema y los datos de Colmap, representan hasta un 30%

4.2.4 Mediciones de exactitud

Otro parámetro de medición para evaluar el sistema de reconstrucción es mediante la medición de exactitud, para esto se compara el valor real del escenario reconstruido con los valores medidos mediante los diferentes números de pruebas. En este apartado, para medir la exactitud, se hace referencia a valores como el error absoluto y la desviación estándar dará determinar la dispersión de los datos.

4.2.4.1 Comparación de medidas obtenidas por el sistema y la medida real

Tabla 11-4: Medidas de perímetros del escenario real vs las mediciones del sistema

No pruebas	Sistema	Real
1	10,5885	13,25
2	9,93076	13,25
3	9,87666	13,25
4	9,87533	13,25
5	10,33583	13,25
6	10,23851	13,25
7	9,89954	13,25
8	9,86928	13,25
9	10,225234	13,25
10	9,26761	13,25
11	9,6751	13,25
12	10,049	13,25
13	10,09126	13,25
14	9,74726	13,25
15	9,883291	13,25
16	11,085797	13,25
17	10,25774	13,25
18	9,26133	13,25
19	9,605101	13,25
20	9,60958	13,25
21	9,40864	13,25
22	10,69234	13,25
23	10,67184	13,25
24	10,03004	13,25
25	10,24292	13,25
26	9,89905	13,25
27	9,79873	13,25
28	9,707995	13,25
29	10,02457	13,25
30	9,5161	13,25
Promedio	9,97883127	13,25
Desviación estándar	0,42168157	0

Realizado por: Shagñay Nelson, 2023

De los valores obtenidos, se puede interpretar de la siguiente manera, mediante el promedio obtenido de los valores medidos, se tiene que la medida media es de 9.97 metros y que su desviación estándar es de ±0.4216 metros con referencia al valor medio, queriendo decir esto

que la mayoría de los valores medidos se encuentran dentro del rango representado por la desviación estándar y en torno al valor promedio medido. Esta información de los datos tomados por el sistema se muestra mediante una gráfica de dispersión como se observa en la Ilustración 11-4.

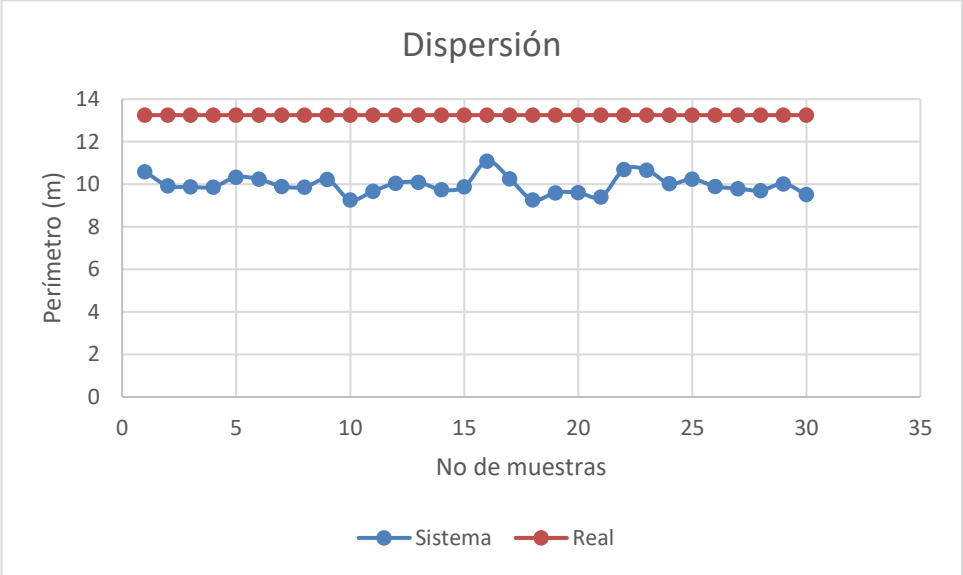


Ilustración 11-4: Dispersión de los datos medidos por el sistema desarrollado con respecto al valor real.

Realizado por: Shagñay Nelson, 2023

4.2.4.2 Cálculo de error absoluto entre medidas obtenidas por el sistema y el valor real

La Tabla 12-4 muestra los valores de errores por prueba hecha con respecto al valor real, así como el valor medio de error absoluto y su desviación estándar, medidas que si toman para determinar el error medio que supone al aplicar el prototipo y como estos errores se distribuyen con respecto al valor medio.

Tabla 12-4: Error absoluto entre el valor real y el valor medido del escenario reconstruido

No pruebas	Error absoluto
1	2,6615
2	3,31924
3	3,37334
4	3,37467
5	2,91417
6	3,01149
7	3,35046
8	3,38072
9	3,024766
10	3,98239
11	3,5749
12	3,201
13	3,15874
14	3,50274
15	3,366709
16	2,164203
17	2,99226
18	3,98867
19	3,644899
20	3,64042
21	3,84136
22	2,55766
23	2,57816
24	3,21996
25	3,00708
26	3,35095
27	3,45127
28	3,542005
29	3,22543
30	3,7339
Promedio	3,271168733
Desviación estándar	0,421681568

Realizado por: Shagñay Nelson, 2023

De los datos obtenidos, se tiene que:

- El valor promedio de error absoluto entre los datos medidos y el valor real es de 3.2711 metros, lo cual, es dimensiones grandes, esto representa un valor de error considerable.
- La desviación estándar es de ±0.4216 metros, lo que quiere decir que la mayoría de los errores medidos se distribuyen es ese rango.

4.2.4.3 Análisis en torno a los valores recabados

Analizando los resultados obtenidos, se puede decir que:

- El análisis estadístico de aleatoriedad nos indicó que los valores medidos con el sistema desarrollado difieren con los valores medidos mediante Colmap que es una herramienta mucho más desarrollada y exacta. Por lo que la prueba T nos sirvió para comprobar esa diferencia de valores.
- La estimación de errores entre el sistema vs Colmap y el sistema vs el valor real, nos llevan a determinar una confiabilidad aceptable del sistema desarrollado, en lo que a términos de precisión y exactitud se refiere.

4.3 Pruebas de reconstrucción del sistema

A continuación, se muestran una serie de experimentos de reconstrucción en base a un mismo entorno, pero variando el número de objetos pertenecientes al escenario y que han sido reconstruidos con el sistema desarrollado, empezando desde el entorno vacío hasta un entorno completamente cubierto por objetos como lo serían con un entorno normal.

4.3.1 Escenario con cajas cuadradas

Al escenario se le agregaron 3 objetos cuadrados, de los cuales se tomaron como referencia para el cálculo de error en la reconstrucción.

Ilustración 12-4: Escenario con cajas cuadradas

Realizado por: Shagñay Nelson, 2023

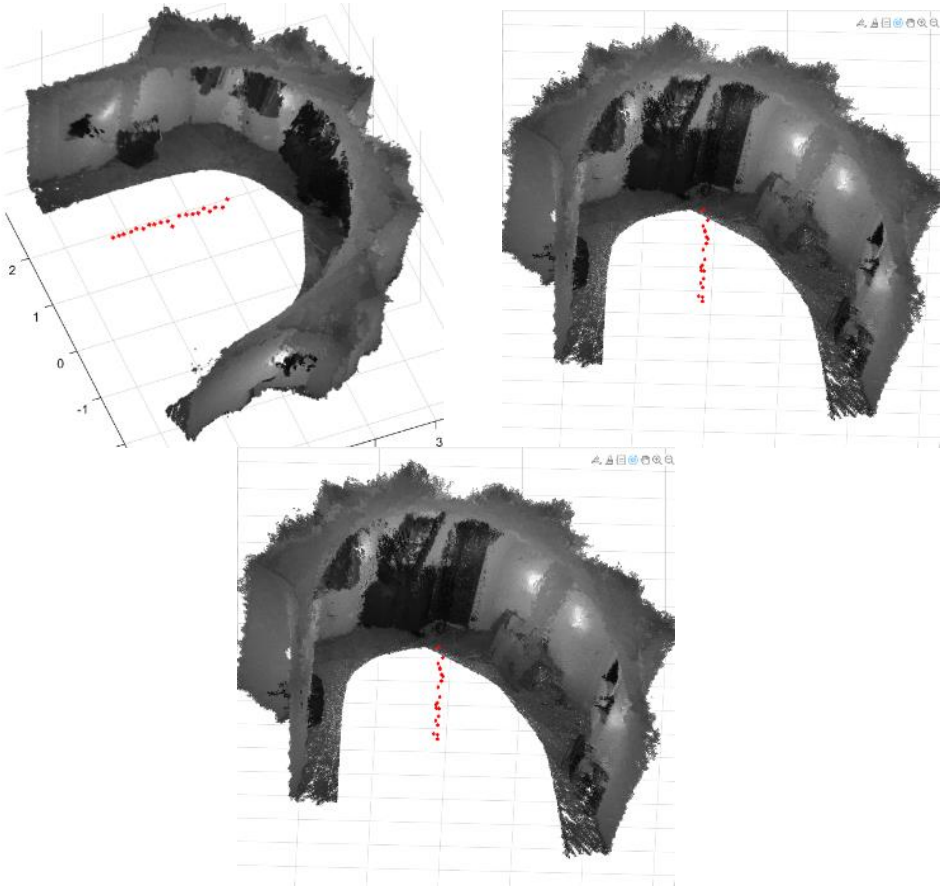


Ilustración 13-4: Escenario reconstruido y trayectoria calculada

Realizado por: Shagñay Nelson, 2023

Tabla 13-4: Error de medidas entre las alturas de los objetos reales y los reconstruidos

Objetos	Real (cm)	Reconstruido (cm)	Error %
1	60	58.32	2.8
2	45	46.76	3.9
3	75	71.47	4.7

Realizado por: Shagñay Nelson, 2023

4.3.2 Reconstrucción con objetos circulares

Al escenario se le agregaron, a parte de los objetos cuadrados, objetos cilindricos para su reconstrucción.

Ilustración 14-4: Escenario con objetos cilíndricos

Realizado por: Shagñay Nelson, 2023

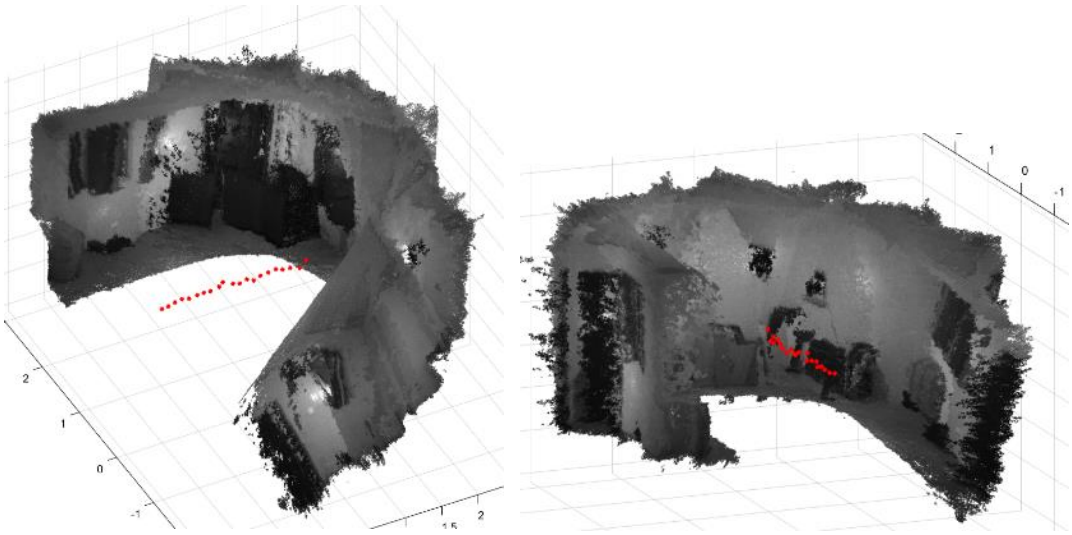


Ilustración 15-4: Escenario con objetos cilíndricos reconstruidos

Realizado por: Shagñay Nelson, 2023

Tabla 14-4: Cálculo de error de alturas de objetos cilíndricos

Objetos	Real (cm)	Reconstruido(cm)	Error %
1	60	67.77	12.95
2	30	12.36	58.8
3	22	17.65	19.77

Realizado por: Shagñay Nelson, 2023

4.3.3 Reconstrucción con objetos de diferentes formas

Para finalizar, se agregaron objetos de varias formas al escenario indoor, con el objetivo de dificultar la reconstrucción.

Ilustración 16-4: Escenario con objetos de distintas formas

Realizado por: Shagñay Nelson, 2023

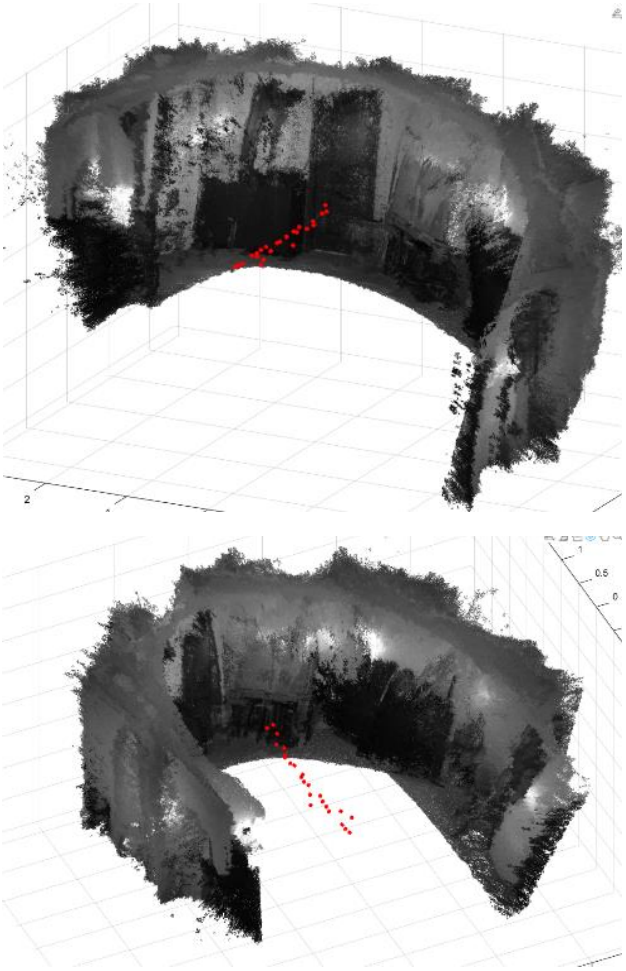


Ilustración 17-4: Escenario con objetos variados reconstruido a 2 vistas

Realizado por: Shagñay Nelson, 2023

Tabla 15-4: Error calculado de medidas reales y medidas reconstruidas

Objetos	Real (cm)	Reconstruido (cm)	Error %
1	80	82.91	3.63
2	125	113.54	9.1
3	90	100.69	11.87

Realizado por: Shagñay Nelson, 2023

4.3.4 Análisis en torno a los escenarios reconstruidos

Al momento de reconstruir el escenario con objetos cuadrados, se tuvo un error porcentual entre las medidas de las alturas reales de los objetos y las medidas de los objetos reconstruido bajos, esto se debe a que las cajas, al momento de que el algoritmo halla las características entre pares de imágenes estéreo, la rectificación de imágenes nos permiten una geometría epipolar más sencilla al momento de generar los mapas de disparidad y de profundidad, por lo que sus reconstrucciones arrojaron errores bajos, caso contrario con los objetos cilíndricos, con el los cuales el error porcentual aumentó, dándonos a saber que el algoritmo no encuentra, de forma precisa, las características entre pares estéreo. Para finalizar, los resultados con objetos variados arrojaron menor porcentaje de error en objetos grandes como el armario agregado, y también tomar en cuenta que el error se calculó de objetos de forma parecidas a las cuadradas, por lo que el error no se ve muy presente en la Tabla 15-4.

CONCLUSIONES

- Se revisaron varias investigaciones sobre avances en la reconstrucción de entornos mediante visión estéreo como única técnica de reconstrucción y complementada con otras técnicas, de las cuales se realizó un estado del arte en base a esas investigaciones que permitieron conocer el estado actual de la tecnología en lo que a reconstrucción de entornos en tres dimensiones se refiere

- Se investigó sobre las categorías de robot móviles destacando 2 principalmente, la de robots móviles autónomos que no necesitan supervisión humana para su funcionamiento y no autónomos que son operados a distancia y son los usados para áreas donde la teleoperación es necesaria, como la exploración de terrenos a distancia

- Se diseñó un sistema que permite la reconstrucción de entornos indoor en 3D mediante el uso de la técnica de visión estéreo, permitiendo obtener tiempos de reconstrucción relativamente bajos, en comparación a técnicas que emplean sensores, redes neuronales, y hasta Deep learning para complementar la técnica de visión estéreo, que son más precisos, pero con tiempos de cómputo más altos.

- Se implementó un sistema prácticamente sencillo, de robot móvil y de una cámara estéreo para la adquisición del entorno, a 360 grados, que no conlleva tanta complejidad en su programación y permite su control a distancia para mayor facilidad al momento de la recolección de imágenes, lo cual hace al sistema fácil de implementarse y portátil para reconstrucción de entornos indoor.

- Se variaron parámetros en el entorno como la intensidad luminosa, así como parámetros de las cámaras como lo son el brillo, intensidad del sensor infrarrojo, y contraste hasta que se encontró los parámetros óptimos que permitieron obtener las imágenes con la mayor calidad posible para su posterior procesamiento

- Se trazó la trayectoria del robot móvil implementado solo procesamiento de imágenes y la técnica de visión estéreo, donde sus tiempos de cómputo son los que más consumen al momento de ejecutar el algoritmo de reconstrucción, esto debido a que se analiza el escenario completamente reconstruido y se compara con las nubes de puntos individuales para poder encontrar el posicionamiento del robot.

- Los resultados muestran que el sistema de reconstrucción por visión estéreo no es un sistema lo suficientemente completo para la reconstrucción de escenarios, esto se debe a

que depende mucho de que ámbitos como la calidad de las imágenes obtenidas, errores de calibración de la cámara que producen ruido en las nubes de puntos que hacen no muy eficiente al algoritmo ICP calcular las transformadas de rotación y traslación de una nube de puntos con respecto a otra haciendo que cada vez que se aplique el algoritmo, los valores reales varíen, como pudimos observar en los resultados de precisión y exactitud del sistema. Además, los resultados de complejidad del escenario muestran que el número de objetos o sus formas no son un gran factor al momento de reconstruir el escenario debido a que el error entre los objetos medidos no se distancia mucho unos de los otros, por lo que un sistema de reconstrucción de entornos depende mucho de la cantidad de datos se puedan recabar del entorno real, de ahí que se usen técnicas complementarias para la obtención de estos datos.

RECOMENDACIONES

- El sistema hace uso de la cámara MYNT EYE S1030 para la obtención del par de imágenes estéreo, estás cámaras ofrecen una baja calidad de las imágenes tomadas y en blanco y negro, por lo que se pierde de muchos datos del entorno al momento de filtrar y segmentar ruido de las nubes de puntos, por lo que, se recomienda el uso de otras cámaras que permitan corregir estos errores para poder mejorar los resultados obtenidos.

- Además, se recomienda el uso de técnicas que complementen la técnica de visión estéreo, esto mejorará mucho más los resultados en cuestión de precisión y exactitud, debido a que estas técnicas complementarias permiten obtener información más exacta del entorno, como lo son láseres, que permiten obtener las dimensiones exactas del entorno y permiten un mapa de profundidad más exacto.

- Se recomienda la implementación de un sistema inalámbrico de recolección de datos, complementando al uso unas cámaras diferentes, ya que en este sistema no se lo logró, debido a 2 razones, la primera, que las cámaras usadas cuentan con una fuente información limitada, debido a que el soporte técnico de la empresa que provee estas cámaras cerró, y resultó altamente complicado el hacerlas funcionar y , segundo porque en los objetivos no se planteó el diseño del sistema inalámbrico para la recolección de datos y envío a la estación base.

- Se hizo uso de MATLAB para el procesamiento de las imágenes debido al conocimiento aprendido de este lenguaje de programación y lo familiarizado que se encuentra el estudiante de telecomunicaciones con este lenguaje, pero se recomienda, para un uso ilimitado de herramientas que van con el procesamiento de imágenes y facilitar el trabajo en esta área el uso de otro lenguaje de programación como Phyton que tiene a su disposición un cúmulo de librerías PCL (point cloud libraries).

BIBLIOGRAFIA

ALLARD, P.-H. & LAVOIE, J.-A. Differentiation of 3D scanners and their positioning method when applied to pipeline integrity, 2014, pp. 7.

BHATTI, A. *Stereo Vision*. [En línea], 2008. [Consulta: 2022-12-15] Disponible en: https://www.researchgate.net/publication/236455177, ISBN 978-953-7619-22-0.

CAPPELLARO, R. Comparison of Stereo Visual Inertial Odometry Algorithms for - Unmanned Ground Vehicles [En línea] (Trabajo de titulación). (Maestría) Politécnico de Torino, Torino, Italia. 2019, p. 9 [Consulta: 2022-11-11] Disponible en: https://webthesis.biblio.polito.it/11648/1/tesi.pdf.

CARABIAS, D.M.; et al. Sistema de Visión Estereoscópica para Navegación Autónoma de vehículos no tripulados, 2010. [en línea]. [Consulta: 2023-01-16]. Disponible en: https://pdfslide.net/documents/sistema-de-vision-estereoscopica-para-navegacion-autonoma-de-vehiculos.html.

CÓRDOVA, D.; et al. Three-Dimensional Reconstruction of Indoor and Outdoor Environments Using a Stereo Catadioptric System Applied Sciences, 2020. [en línea], vol. 10, no. 24, pp. 18. [Consulta: 15 febrero 2023]. ISSN 2076-3417. DOI 10.3390/app10248851. Disponible en: https://www.mdpi.com/2076-3417/10/24/8851.

CYGANEK, B.; & SIEBERT, J.P. *An Introduction to 3D Computer Vision Techniques and Algorithms: Cyganek/An Introduction to 3D Computer Vision Techniques and Algorithms*, 2009. [en línea]. Chichester, UK: John Wiley & Sons, Ltd. [Consulta: 5 mayo 2022]. ISBN 978-0-470-69972-0. Disponible en: http://doi.wiley.com/10.1002/9780470699720.

DIGITAL TALENT AGENCY. *ModeloWaterfall* [en línea] 2018., pp. 4. [Consulta: 2023-01-19]. Disponible en: https://www.dtagency.tech/cursos/metodologias_gestion_proyectos/tema_1-ModeloWaterfall.pdf.

GRACIA, L. ¿Qué es OpenCV? *Un poco de Java* [blog]. [Consulta: 2022-07-14]. Disponible en: https://unpocodejava.com/2013/10/09/que-es-opencv/.

HUANG, P.-H.; et al. DeepMVS: Learning Multi-View Stereopsis. *Proceedings of the IEEE Conference on Computer Vision and Pattern* Recognition, 2018 [en línea]. [Consulta: 2023-02-15]. Disponible en: https://openaccess.thecvf.com/content_cvpr_2018/html/Huang_DeepMVS_Learning_Multi-View_CVPR_2018_paper.html.

JIN, Y.-H.; et al. An Indoor Location-Based Positioning System Using Stereo Vision with the Drone Camera. *Mobile Information Systems*, 2018. [en línea], vol. 2018. [Consulta: 15 febrero 2023]. ISSN 1574-017X. DOI 10.1155/2018/5160543. Disponible en: https://www.hindawi.com/journals/misy/2018/5160543/.

JUNG, S.; et al. 3D Reconstruction Using 3D Registration-Based ToF-Stereo Fusion. *Sensors (Basel, Switzerland)*,2022, vol. 22, no. 21. ISSN 1424-8220. DOI 10.3390/s22218369.

LUNA, I.S. La historia de la imagen o una imagen para la historia. *Cuicuilco*, 2003. [en línea], vol. 10, no. 29, p. 4. [Consulta: 2023-02-16]. ISSN 1405-7778,. Disponible en: https://www.redalyc.org/articulo.oa?id=35102912.

MARTÍNEZ, M.M. Técnicas de visión estereoscópica para determinar la estructura tridimensional de la escena [En línea] (Trabajo de titulación). (Maestría) Universidad Complutense de Madrid, Madrid, España. 2010, pp. 97. [Consulta: 2022-09-19] Disponible en: https://eprints.ucm.es/id/eprint/11350/1/T%C3%A9cnicas_de_visi%C3%B3n_estereosc%C3%B3pica_para_determinar_la_estructura_tridimensional_de_al_escena.pdf

MEDIONI, G.; & NEVATIA, R. Segment-based stereo matching. *Computer Vision, Graphics, and Image Processing*, 1985. [en línea], vol. 31, no. 1, pp. 2-18. [Consulta: 14 junio 2022]. ISSN 0734-189X. DOI 10.1016/S0734-189X(85)80073-6. Disponible en: https://www.sciencedirect.com/science/article/pii/S0734189X85800736.

MYNTAI, MYNT EYE Standard. *MYNT EYE*, 2019. [blog]. [Consulta: 6 mayo 2022]. Disponible en: https://www.mynteye.com/products/mynt-eye-stereo-camera.

NARVÁEZ, J.E.S.; & HERRERA, A.M.Z. Artificial Estereoscópico, Para Aplicaciones En Navegación Robótica [En línea] (Trabajo de titulación). (Ingeniería) Universidad de San Buenaventura Seccional Medellín, Medellín, Colombia. 2015. p. 14 [Consulta: 2020-12-16] Disponible en: http://bibliotecavirtualoducal.uc.cl/vufind/Record/oai:localhost:10819-4155

PAJARES, G.; & DE LA CRUZ, J.M. *Visión por computador. Imágenes Digitales y Aplicaciones*, 2008. *2ª Edición*. Paracuellos del Jarama (Madrid): RA-MA S.A. Editorial y Publicaciones. ISBN 978-84-7897-831-1.

PATIÑO, R., *Reconstrucción 3d A Partir De Imágenes Rgb-D En Ambientes Controlados*, [en línea] (Trabajo de titulación). (Maestría) Instituto Politécnico Nacional, México D.F., México. 2013. p. 53 [Consulta: 2020-12-26]. Disponible en: https://tesis.ipn.mx/bitstream/handle/123456789/20982/Reconstrucci%C3%B3n%203D%20a%20partir%20de%20im%C3%A1genes%20RGB-D%20en%20ambientes%20controlados.pdf?sequence=1&isAllowed=y.

POLLARD, S.B.; et al. A Stereo Correspondence Algorithm Using a Disparity Gradient Limit. *Perception*, 1985. [en línea], vol. 14, no. 4. [Consulta: 14 junio 2022]. ISSN 0301-0066. DOI 10.1068/p140449. Disponible en: https://doi.org/10.1068/p140449.

REDHAT. ¿Qué es un SDK?. [blog]. [Consulta: 14 junio 2022]. Disponible en: https://www.redhat.com/es/topics/cloud-native-apps/what-is-SDK.

REID, I.D.; & BEARDSLEY, P.A. Self-alignment of a binocular robot. *Image and Vision Computing* [en línea]1996, vol. 14, no. 8, [Consulta: 14 junio 2022]. ISSN 0262-8856. DOI 10.1016/0262-8856(96)01095-5. Disponible en: https://www.sciencedirect.com/science/article/pii/0262885696010955.

ROBERTS, L.G.. *Machine perception of three-dimensional solids* [en línea]. (Trabajo de titulación). (Doctorado) Massachusetts Institute of Technology, Massachusetts, Estados Unidos de América. 1963. p. 14 [Consulta: 16 febrero 2023]. Disponible en: https://dspace.mit.edu/handle/1721.1/11589.

VARGAS, R. Dispositivo De Reconstrucción Tridimensional Remoto Mediante Luz Estructurada Y Visión Estereoscópica Para Ambientes No Controlados [En línea] (Trabajo de titulación). (Doctorado) Universidad Autónoma del Estado de Morelos, Cuernavaca, México. 2020. pp. 14 - 28 [Consulta: 18 de enero del 2023]. Disponible en: http://riaa.uaem.mx/xmlui/bitstream/handle/20.500.12055/1189/BEVRRB03T.pdf?sequence=1&isAllowed=y

VELAZQUEZ, M. *Reclu IT*. [blog]. [Consulta: 14 junio 2022]. Disponible en: https://recluit.com/que-es-matlab/#.YqiMqOzMK3A.

VILA UBIETO, K. Reconstrucción 3d De Modelos Utilizando Técnicas De Visión Artificial [en línea] (Trabajo de titulación). (Ingeniería) Universidad Pontificia Comillas, Madrid, España. 2009. p. 9 [Consulta: 16 febrero 2023]. Disponible en: https://silo.tips/download/reconstruccion-3d-de-modelos-utilizando-tecnicas-de-vision-artificial.

ANEXOS

Anexo A: Código del proyecto record en Visual Studio para la captura de datos

Archivo Record:

```
include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include "mynteye/logger.h"
#include "mynteye/device/device.h"
#include "mynteye/device/utils.h"
#include "mynteye/util/times.h"
#include "dataset.h"
#include "mynteye/api/api.h"

MYNTEYE_USE_NAMESPACE

int main(int argc, char *argv[]) {
  glog_init _(argc, argv);
  auto &&api = API::Create(argc, argv);
  if (!api) return 1;
  auto request = api->GetStreamRequest();
  request.fps = 10;
  api->ConfigStreamRequest(request);
  api->EnableMotionDatas();
  api->EnableStreamData(Stream::DEPTH);
  api->Start(Source::ALL);
  const char *outdir;
  if (argc >= 2) {
    outdir = argv[1];
  } else {
    outdir = "./dataset";
  }
  tools::Dataset dataset(outdir);

  cv::namedWindow("frame");
```

```
std::size_t img_count = 0;
std::size_t imu_count = 0;
auto &&time_beg = times::now();
while (true) {
  api->WaitForStreams();
          auto &&left_datas = api->GetStreamDatas(Stream::LEFT);
          auto &&right_datas = api->GetStreamDatas(Stream::RIGHT);
  auto &&depth_data = api->GetStreamData(Stream::DEPTH);
  auto &&disparity_data = api->GetStreamData(Stream::DISPARITY);
  img_count += left_datas.size();

  auto &&motion_datas = api->GetMotionDatas();
  imu_count += motion_datas.size();
  cv::Mat img;
  if (left_datas.size() > 0 && right_datas.size() > 0) {
    auto &&left_frame = left_datas.back().frame_raw;
    auto &&right_frame = right_datas.back().frame_raw;
    if (right_frame->data() && left_frame->data()) {
      if (left_frame->format() == Format::GREY) {
        cv::Mat left_img(
            left_frame->height(), left_frame->width(), CV_8UC1,
            left_frame->data());
        cv::Mat right_img(
            right_frame->height(), right_frame->width(), CV_8UC1,
            right_frame->data());
        cv::hconcat(left_img, right_img, img);
      } else if (left_frame->format() == Format::YUYV) {
        cv::Mat left_img(
            left_frame->height(), left_frame->width(), CV_8UC2,
            left_frame->data());
        cv::Mat right_img(
            right_frame->height(), right_frame->width(), CV_8UC2,
            right_frame->data());
        cv::cvtColor(left_img, left_img, cv::COLOR_YUV2BGR_YUY2);
        cv::cvtColor(right_img, right_img, cv::COLOR_YUV2BGR_YUY2);
        cv::hconcat(left_img, right_img, img);
      } else if (left_frame->format() == Format::BGR888) {
```

```
      cv::Mat left_img(
          left_frame->height(), left_frame->width(), CV_8UC3,
          left_frame->data());
      cv::Mat right_img(
          right_frame->height(), right_frame->width(), CV_8UC3,
          right_frame->data());
      cv::hconcat(left_img, right_img, img);
    } else {
      return -1;
    }
    cv::imshow("frame", img);
  }
}
if (img_count > 10 && imu_count > 50) {  // save
  // save Stream::LEFT
  for (auto &&left : left_datas) {
    dataset.SaveStreamData(Stream::LEFT, left);
  }
  // save Stream::RIGHT
  for (auto &&right : right_datas) {
    dataset.SaveStreamData(Stream::RIGHT, right);
  }
  // save Stream::DEPTH
  if (!depth_data.frame.empty()) {
    dataset.SaveStreamData(Stream::DEPTH, depth_data);
  }
  // save Stream::DISPARITY
  if (!disparity_data.frame.empty()) {
    dataset.SaveStreamData(Stream::DISPARITY, disparity_data);
  }
  for (auto &&motion : motion_datas) {
    dataset.SaveMotionData(motion);
  }
  std::cout << "\rSaved " << img_count << " imgs"
            << ", " << imu_count << " imus" << std::flush;
}
```

```
    char key = static_cast<char>(cv::waitKey(1));
    if (key == 27 || key == 'q' || key == 'Q') {  // ESC/Q
      break;
    }
  }
  std::cout << " to " << outdir << std::endl;
  auto &&time_end = times::now();
  api->Stop(Source::ALL);
  float elapsed_ms =
      times::count<times::microseconds>(time_end - time_beg) * 0.001f;
  LOG(INFO) << "Time beg: " << times::to_local_string(time_beg)
            << ", end: " << times::to_local_string(time_end)
            << ", cost: " << elapsed_ms << "ms";
  LOG(INFO) << "Img count: " << img_count
            << ", fps: " << (1000.f * img_count / elapsed_ms);
  LOG(INFO) << "Imu count: " << imu_count
            << ", hz: " << (1000.f * imu_count / elapsed_ms);
  return 0;
}
```

Anexo B: Código de Proyecto ctrl_infrared

```
#include <opencv2/highgui/highgui.hpp>
#include "mynteye/logger.h"
#include "mynteye/api/api.h"

MYNTEYE_USE_NAMESPACE

int main(int argc, char *argv[]) {
  auto &&api = API::Create(argc, argv);
  if (!api) return 1;
  bool ok;
  auto &&request = api->SelectStreamRequest(&ok);
  if (!ok) return 1;
  api->ConfigStreamRequest(request);
  Model model = api->GetModel();
  if (model == Model::STANDARD) {
```

```
    // ir control: range [0,160], default 0
    api->SetOptionValue(Option::IR_CONTROL, 120);

    LOG(INFO) << "Set IR_CONTROL to "
              << api->GetOptionValue(Option::IR_CONTROL);
  }
  if (model == Model::STANDARD210A ||
   model == Model::STANDARD2 || model == Model::STANDARD200B) {
    LOG(INFO) << "Sorry, MYNTEYE-S210A S2000 S200B don't support ir control";
    return 0;
  }
  api->Start(Source::VIDEO_STREAMING);
  cv::namedWindow("frame");
  while (true) {
    api->WaitForStreams();

    auto &&left_data = api->GetStreamData(Stream::LEFT);
    auto &&right_data = api->GetStreamData(Stream::RIGHT);

    cv::Mat img;
    cv::hconcat(left_data.frame, right_data.frame, img);
    cv::imshow("frame", img);

    char key = static_cast<char>(cv::waitKey(1));
    if (key == 27 || key == 'q' || key == 'Q') {  // ESC/Q
      break;
    }
  }

  api->Stop(Source::VIDEO_STREAMING);
  return 0;
}
```

Anexo C: Código de reconstrucción de entornos y cálculo de trayectoria en Matlab

```
%%                    INICIO DEL PROYECTO
%% Lectura de imagenes de la camaras MYNT EYE
tic
imageNames = dir(fullfile('datasetvariado','left','*.png'));
imageNames = {imageNames.name}';
imageNames1 = dir(fullfile('datasetvariado','right','*.png'));
imageNames1 = {imageNames1.name}';

%% Construccion del archivo de video (en blanco)
outputVideo = VideoWriter(fullfile('left.avi'));
outputVideo.FrameRate = 10;
open(outputVideo)

outputVideo1 = VideoWriter(fullfile('right.avi'));
outputVideo1.FrameRate = 10;
open(outputVideo1)

for ii = 1:25:length(imageNames)
   img = imread(fullfile('datasetvariado','left',imageNames{ii}));
   writeVideo(outputVideo,img)
end
close(outputVideo)

for ii = 1:25:length(imageNames1)
   img1 = imread(fullfile('datasetvariado','right',imageNames1{ii}));
   writeVideo(outputVideo1,img1)
end
close(outputVideo1)

%% Create Video File Readers and the Video Player para leer los frames del video
videoFileLeft = 'left.avi';
videoFileRight = 'right.avi';

%reproduccion de video
videoreproduccion = VideoReader(videoFileLeft);
```

```
[height width channels] = size(readFrame(videoreproduccion));
player = vision.VideoPlayer('Position', [0,0,width height],'Name','CAMARA IZQUIERDA');
while hasFrame(videoreproduccion)
  frame = readFrame(videoreproduccion);
  step(player,frame);
  pause(1/videoreproduccion.frameRate)
end
release(player);
toc
%%                        NUBES DE PUNTOS
%% Read and Rectify Video Frames
tic
load stereoParamscam21.mat
% lectura de datos de los videos left y right
readerLeft = VideoReader(videoFileLeft);
readerRight = VideoReader(videoFileRight);
player = vision.VideoPlayer('Position', [20,200,740 560]);
figure;
frameLeft = readFrame(readerLeft);
frameRight = readFrame(readerRight);
[frameLeftRect, frameRightRect] = ...
   rectifyStereoImages(frameLeft, frameRight, stereoParamscam21);
subplot(2,2,1)
imshow(frameLeft)
title('Imagen Derecha sin rectificar')
subplot(2,2,2)
imshow(frameLeftRect)
title('Imagen Derecha Rectificada')
subplot(2,2,3)
imshow(frameLeft)
title('Imagen Izquierda sin rectificar')
subplot(2,2,4)
imshow(frameLeftRect)
title('Imagen Izquierda Rectificada')

%% Calculo de mapa de disparidad
frameLeftGray  = rgb2gray(frameLeftRect);
```

```
frameRightGray = rgb2gray(frameRightRect);

% disparityRange = [-16*1 256];
% disparityMap1 = disparity(frameLeftGray, frameRightGray,'BlockSize',5,
'DisparityRange',disparityRange,'UniquenessThreshold',11*1,'ContrastThreshold',0.05,'method','
SemiGlobal');
disparityRange = [0 64];
disparityMap1 = disparitySGM(frameLeftGray, frameRightGray,'DisparityRange'...
   ,disparityRange,'UniquenessThreshold',0);
disparityMap2 = disparityBM(frameLeftGray,
frameRightGray,'DisparityRange',disparityRange,'UniquenessThreshold',11*1 );
figure;
subplot(2,1,1)
imshow(disparityMap1, disparityRange);
title('Mapa de disparidad usando SGM');
colormap jet
colorbar
subplot(2,1,2)
imshow(disparityMap2, disparityRange);
title('Mapa de disparidad usando BM');
colormap jet
colorbar

%% Creacion nubes de puntos
readerLeft = VideoReader(videoFileLeft);
readerRight = VideoReader(videoFileRight);
points3D = reconstructScene(disparityMap1, stereoParamscam21);
points3D1 = reconstructScene(disparityMap2, stereoParamscam21);

% Convert to meters and create a pointCloud object
points3D = points3D ./ 1000;
th = [-1.6 2;-1.5 1;1.4 3];
points3D = thresholdPC(points3D,th);
ptCloud = pointCloud(points3D, 'Color', frameLeftRect);
ptCloud = pcdenoise(ptCloud, 'NumNeighbors', 8, 'Threshold', 1);
ptCloud = removeInvalidPoints(ptCloud);
```

```
% Create a streaming point cloud viewer
player3Dsim = pcplayer([-2, 4], [-2, 2], [0, 4], 'VerticalAxis', 'y', ...
    'VerticalAxisDir', 'down');
% Visualize the point cloud
view(player3Dsim, ptCloud);

% Process the Rest of the Video
i=0;
while hasFrame(readerLeft) && hasFrame(readerRight)
    i=i+1;
    % Read the frames.
    frameLeft = readFrame(readerLeft);
    frameRight = readFrame(readerRight);

    % Rectify the frames.
    [frameLeftRect, frameRightRect] = ...
        rectifyStereoImages(frameLeft, frameRight, stereoParamscam21);

    % Convert to grayscale.
    frameLeftGray  = rgb2gray(frameLeftRect);
    frameRightGray = rgb2gray(frameRightRect);

    % Compute disparity.
    disparityRange = [0 64];
    disparityMap1 = disparitySGM(frameLeftGray, frameRightGray,'DisparityRange'...
    ,disparityRange,'UniquenessThreshold',0);
%     disparityRange = [-16*1 256];
%     disparityMap1 = disparity(frameLeftGray, frameRightGray,'BlockSize',5,
'DisparityRange',disparityRange,'UniquenessThreshold',11*1,'ContrastThreshold',0.05,'method','
SemiGlobal');

    % Reconstruct 3-D scene.
    points3D = reconstructScene(disparityMap1, stereoParamscam21);
    points3D = points3D ./ 1000;
    points3D = thresholdPC(points3D,th);
    ptCloud = pointCloud(points3D, 'Color', frameLeftRect);
    ptCloud = pcdenoise(ptCloud, 'NumNeighbors', 8, 'Threshold', 1);
```

```
    ptCloud = removeInvalidPoints(ptCloud);
    view(player3Dsim, ptCloud);
    ContenedorPT{i}=ptCloud;
%     pause(2)
end
release(player);
toc
%%                          UNION DE NUBE DE PUNTOS
% Register Two Point Clouds
tic
ptCloudRef = ContenedorPT{1};
ptCloudCurrent = ContenedorPT{2};

gridSize1 = 0.1;
fixed = pcdownsample(ptCloudRef, 'gridAverage', gridSize1);
moving = pcdownsample(ptCloudCurrent, 'gridAverage', gridSize1);

tform = pcregistericp(moving, fixed, 'Metric','pointToPlane','Extrapolate', true);
Contenedortransformada{1}=tform;
Contenedortransformada{2}=tform;
ptCloudAligned = pctransform(ptCloudCurrent,tform);

% Unir las nubes de puntos
mergeSize = 0.01;
ptCloudScene = pcmerge(ptCloudRef, ptCloudAligned, mergeSize);

% Visualize the input images.
figure
subplot(2,2,1)
imshow(ptCloudRef.Color)
title('First input image','Color','w')
drawnow

subplot(2,2,3)
imshow(ptCloudCurrent.Color)
title('Second input image','Color','w')
drawnow
```

```
% % Visualize the world scene.
subplot(2,2,[2,4])
pcshow(ptCloudScene, 'VerticalAxis','Y', 'VerticalAxisDir', 'Down')
title('Initial world scene')
xlabel('X (m)')
ylabel('Y (m)')
zlabel('Z (m)')

% Acumulado de transformadas
accumTform = tform;
figure
hAxes = pcshow(ptCloudScene, 'VerticalAxis','Y', 'VerticalAxisDir', 'Down');
title('Updated world scene')
% Set the axes property for faster rendering
hAxes.CameraViewAngleMode = 'auto';
hScatter = hAxes.Children;

for i = 3:length(ContenedorPT)
    ptCloudCurrent = ContenedorPT{i};

    % Use previous moving point cloud as reference.
    fixed = moving;
    moving = pcdownsample(ptCloudCurrent, 'gridAverage', gridSize1);

    % Apply ICP registration.
    tform = pcregistericp(moving, fixed, 'Metric','pointToPlane','Extrapolate', true);
    Contenedortransformada{i}=tform;

    % Transform the current point cloud to the reference coordinate system
    % defined by the first point cloud.
    accumTform = affine3d(tform.T * accumTform.T);
    ptCloudAligned = pctransform(ptCloudCurrent, accumTform);

    % Update the world scene.
    ptCloudScene = pcmerge(ptCloudScene, ptCloudAligned, mergeSize);
```

```
    % Visualize the world scene.
    hScatter.XData = ptCloudScene.Location(:,1);
    hScatter.YData = ptCloudScene.Location(:,2);
    hScatter.ZData = ptCloudScene.Location(:,3);
    hScatter.CData = ptCloudScene.Color;
    drawnow('limitrate')
end
angle = -pi/6;
A = [1,0,0,0;...
     0, cos(angle), sin(angle), 0; ...
     0, -sin(angle), cos(angle), 0; ...
     0 0 0 1];
% ptCloudScene = pctransform(ptCloudScene, affine3d(A));
pcshow(ptCloudScene, 'VerticalAxis','Y', 'VerticalAxisDir', 'Down')
title('Escena Reconstruida')
xlabel('X (m)')
ylabel('Y (m)')
zlabel('Z (m)')
toc
pause(10)
pcwrite(ptCloudScene,"ptCloudScene.ply",PLYFormat="binary");
%%                          TRAYECTORIA DEL ROBOT
tic
voxelSize = 0.02;
ndtMap = pcmapndt(ptCloudScene,voxelSize);
center = Contenedortransformada{1}.Translation;
% sz = [70 50 20];
sz = [4 8 8];
ndtMap = selectSubmap(ndtMap,center,sz);
axis = [0.02 0.02 0.02 0 0 0];
distThresh = 0.2;
for n = 1:length(ContenedorPT)
    ptCloud = ContenedorPT{n};
    initPose = Contenedortransformada{n};

    poseTranslation = initPose.Translation;
    [isInside,distToEdge] = isInsideSubmap(ndtMap,poseTranslation);
```

```
    submapNeedsUpdate = ~isInside ...       % Current pose is outside submap
        || any(distToEdge(1:2) < distThresh);   % Current pose is close to submap edge

if submapNeedsUpdate
    ndtMap = selectSubmap(ndtMap,poseTranslation,sz);
end

% Localize the point cloud scan in the map.
currPose = findPose(ndtMap,ptCloud,initPose);
pos = [currPose.Translation(1:2) radius];
showShape('circle',pos,'Color','r');
hold on
% Display the position of the estimate as a circle.

% Pause to view the change.
pause(0.2)
end
toc
```

Anexo D: Programación del robot móvil en Arduino IDE

```
#include <AFMotor.h>
#include <SoftwareSerial.h>

SoftwareSerial BT1(9, 10);
AF_DCMotor motorUno(1, MOTOR12_64KHZ);
AF_DCMotor motorDos(2, MOTOR12_64KHZ);
AF_DCMotor motorTres(3, MOTOR34_64KHZ);
AF_DCMotor motorCuatro(4, MOTOR34_64KHZ);

char dato;
void setup() {
  BT1.begin(9600);
}

void loop() {
  if (BT1.available() > 0) {
```

```
    dato = BT1.read();
  }

  switch (dato) {
    case 'F':
      ADELANTE();
      break;
    case 'L':
      IZQUIERDA();
      break;
    case 'R':
      DERECHA();
      break;

    case 'B':
      REVERSA();
      break;

    case 'S':
      Stop();
      break;

  }
}

void ADELANTE() {
  motorUno.setSpeed(70);
  motorDos.setSpeed(70);
  motorTres.setSpeed(70);
  motorCuatro.setSpeed(70);
  motorUno.run(FORWARD);
  motorDos.run(FORWARD);
  motorTres.run(FORWARD);
  motorCuatro.run(FORWARD);
}

void REVERSA() {
```

```
  motorUno.setSpeed(70);
  motorDos.setSpeed(70);
  motorTres.setSpeed(70);
  motorCuatro.setSpeed(70);
  motorUno.run(BACKWARD);
  motorDos.run(BACKWARD);
  motorTres.run(BACKWARD);
  motorCuatro.run(BACKWARD);
}

void DERECHA() {
  motorUno.setSpeed(100);
  motorDos.setSpeed(100);
  motorTres.setSpeed(100);
  motorCuatro.setSpeed(100);
  motorUno.run(FORWARD);
  motorDos.run(RELEASE);
  motorTres.run(FORWARD);
  motorCuatro.run(RELEASE);
}

void IZQUIERDA() {
  motorUno.setSpeed(100);
  motorDos.setSpeed(100);
  motorTres.setSpeed(100);
  motorCuatro.setSpeed(100);
  motorUno.run(RELEASE);
  motorDos.run(FORWARD);
  motorTres.run(RELEASE);
  motorCuatro.run(FORWARD);
}
void Stop()
{
  motorUno.run(RELEASE); //stop the motor when release the button
  motorDos.run(RELEASE); //rotate the motor clockwise
  motorTres.run(RELEASE); //stop the motor when release the button
  motorCuatro.run(RELEASE); //stop the motor when release the button
```

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

DIRECCIÓN DE BIBLIOTECAS Y RECURSOS DEL APRENDIZAJE

UNIDAD DE PROCESOS TÉCNICOS

REVISIÓN DE NORMAS TÉCNICAS, RESUMEN Y BIBLIOGRAFÍA

Fecha de entrega: 31/ 03 / 2023

INFORMACIÓN DEL AUTOR/A (S)
Nombres – Apellidos: Nelson Stalin Shagñay Ruiz
INFORMACIÓN INSTITUCIONAL
Facultad: Informática y Electrónica
Carrera: Telecomunicaciones
Título a optar: Ingeniero en Electrónica, Telecomunicaciones y Redes
f. Analista de Biblioteca responsable: Ing. Fernanda Arévalo M.

0542-DBRA-UPT-2023