Localización y construcción de mapas voxelizados en robótica móvil empleando cámaras RGB-D

- **Autor:**
- Pablo Vázquez Vera
- **Tutores:**
- José Raúl Ruiz Sarmiento
- Jose Luis Matez Bandera

Índice

Introducción: Breve explicación sobre el problema abordado

Objetivos: Resultados que se desea obtener

Metodología: Fases de desarrollo

Implementación: Detalles sobre el flujo de trabajo

Resultados: Métricas para validar que se cumplen los objetivos

Conclusiones: Interpretación general de los resultados

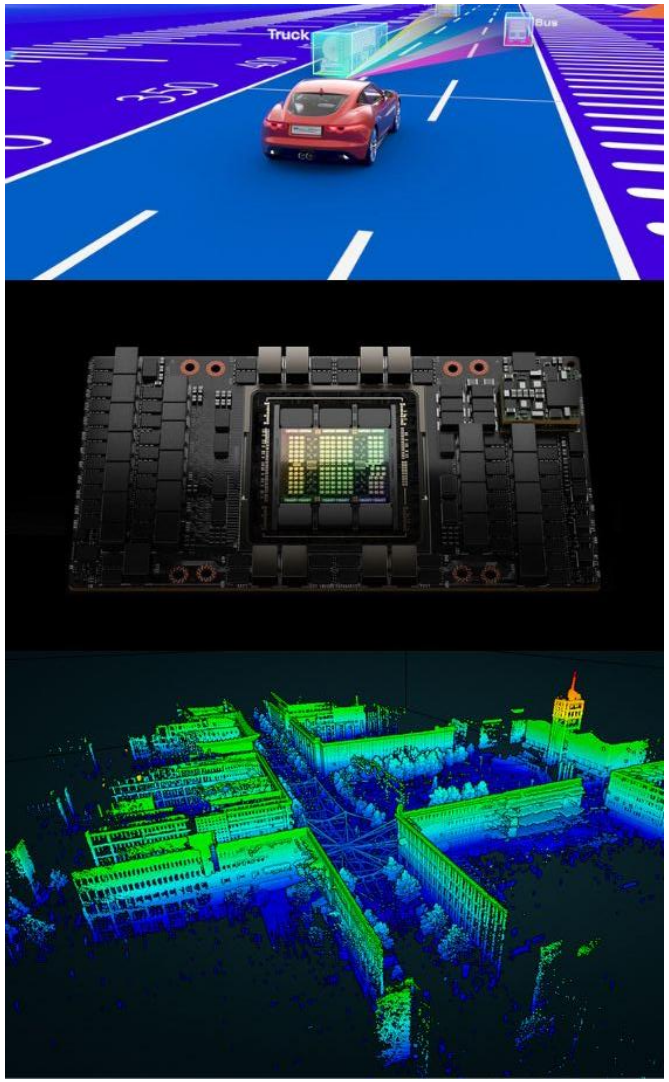
Introducción - *Panorama*

El mundo ha visto como en la ultima década, la robótica móvil vista en la ciencia ficción cada vez esta mas cerca de convertirse en realidad. Desde vehículos autónomos hasta robots humanoides, esta incipiente revolución cambiara el mundo tal y como lo conocemos.

Introducción - *Tecnologías impulsoras*

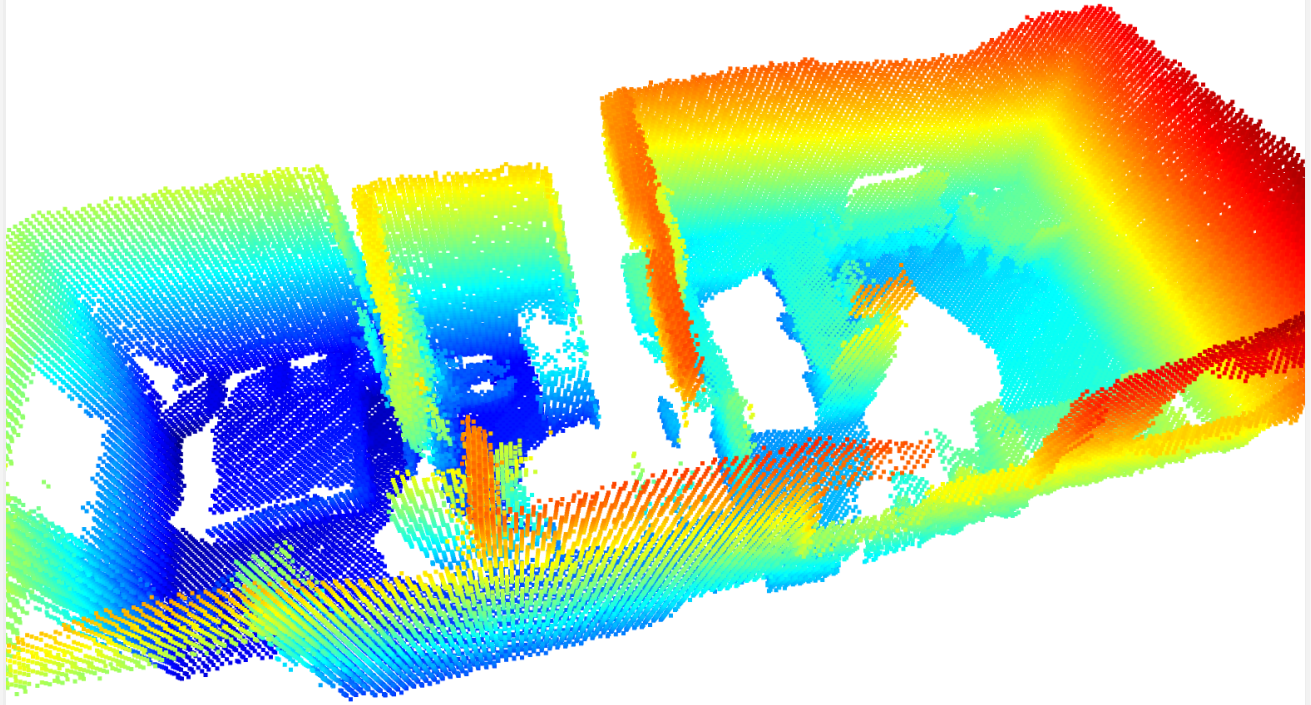
La robótica móvil ha ampliado sus horizontes de forma exponencial en la última década gracias a una combinación de factores:

- **Desarrollo de nuevos sensores más precisos:**
  - Cámaras 3D, LIDAR o sensores ultrasónicos.
- **Inteligencia Artificial y aprendizaje automático:**
  - Implementación de algoritmos de visión por computador capaces de reconocer objetos, personas y obstáculos.
- **Mapeo y localización (SLAM - Simultaneous Localization and Mapping):**
  - Técnicas avanzadas de SLAM permiten a los robots orientarse y construir mapas de entornos complejos si depender de GPS.
- **Mayor capacidad de computación:**
  - Hardware miniaturizado especializado como GPU y TPU a bordo de los robots.
- **Conectividad y sistemas distribuidos:**
  - Uso de 5G, Wi-Fi de baja latencia y redes mesh para controlar enjambres de robots.
- **Software y entornos de simulación:**
  - Frameworks como ROS2 han estandarizado el desarrollo robótico



Introducción - *Problemática*

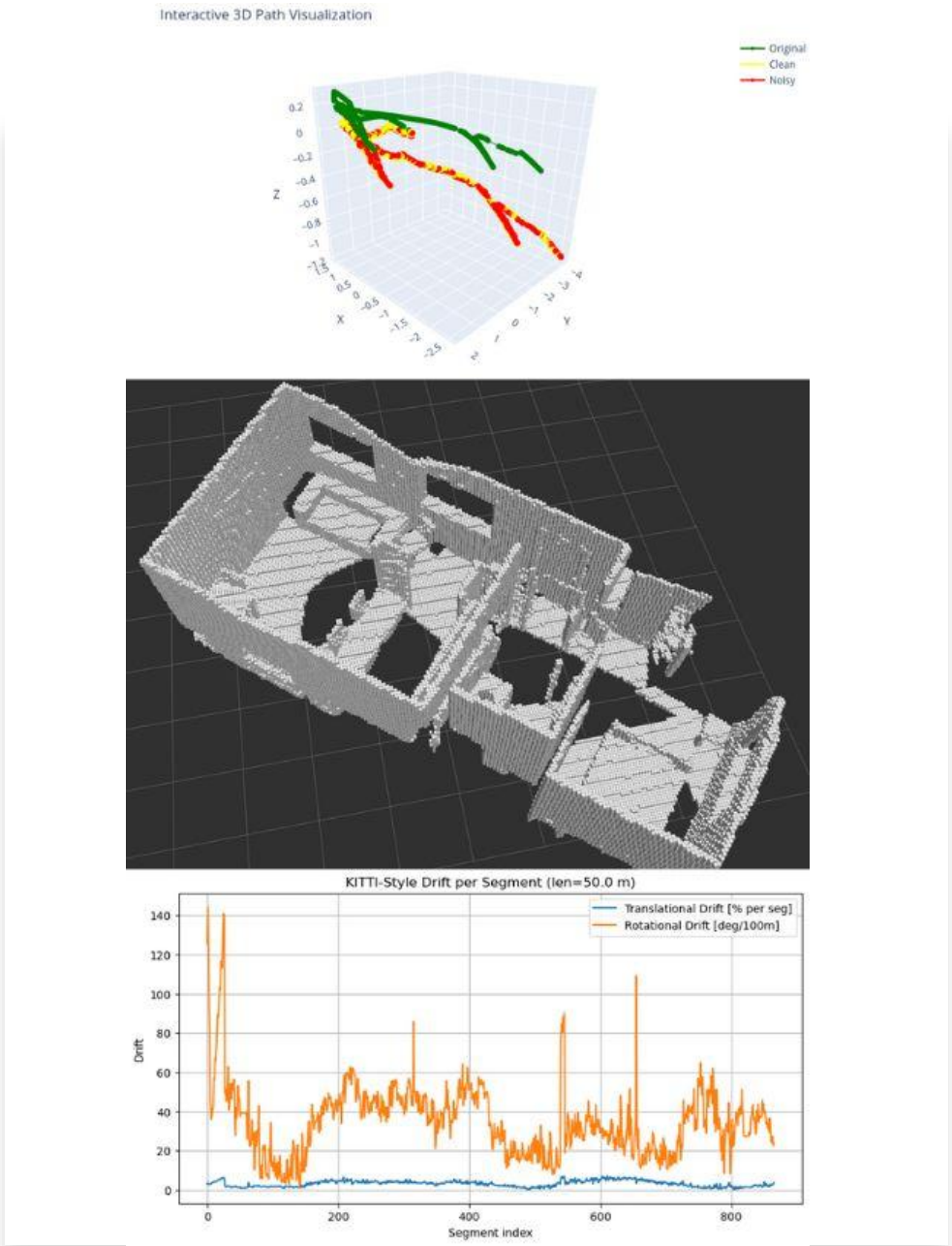
- El problema que abordamos en este proyecto es el propio de **SLAM**, es decir, intentaremos localizarnos en el entorno en el que nos encontramos a la vez que construimos un mapa de este. Para ello, usaremos como única fuente de datos, aquella producida por un sensor **RGB-D**.
- Todo esto se llevará a cabo en un **entorno simulado**, pero para probar la robustez del sistema desarrollado, se añadirá un error considerable a la entrada (nube de puntos).



Objetivos

Los objetivos de este proyecto son los siguientes:

- Calcular la pose del sistema
- Generar un mapa voxelizado del entorno (comúnmente conocido como mapa)
- Evaluar la calidad de la pose calculada por el sistema
- Evaluar la calidad del mapa generado por el sistema

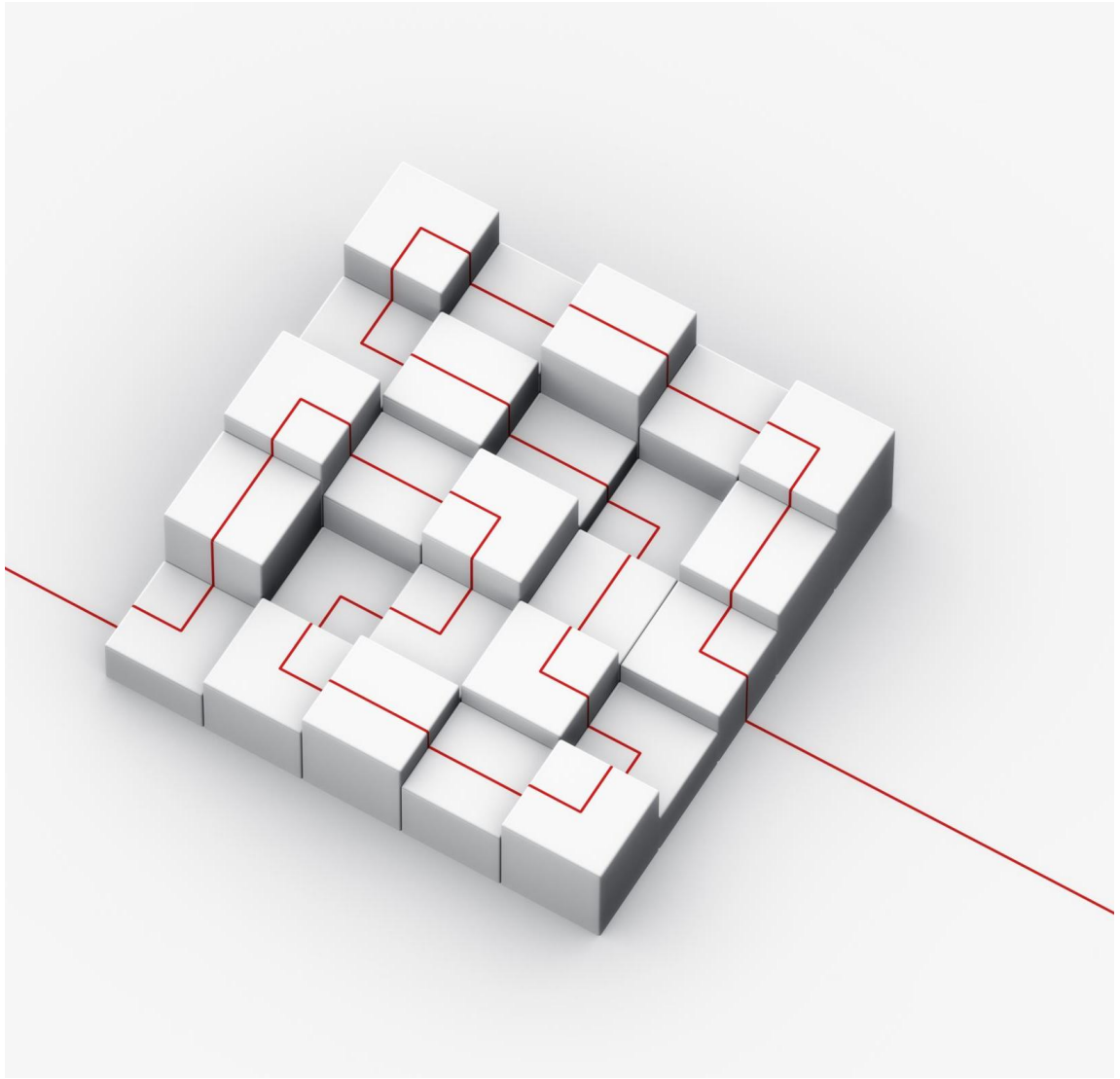


Metodología

El modelo de desarrollo usado ha sido:

- **Desarrollo incremental:** El software se construye en partes pequeñas donde cada una aporta una funcionalidad completa

Para esto, se ha creado un repositorio en la plataforma GITHUB y se han creado ramas para implementar cada una de las funcionalidades, que posteriormente se han fusionado con la rama principal.



Implementación - Conceptos Clave

Antes de entrar en los detalles de la implementación, es conveniente explicar algunos conceptos clave que facilitan la comprensión de cada paso. Estos son:

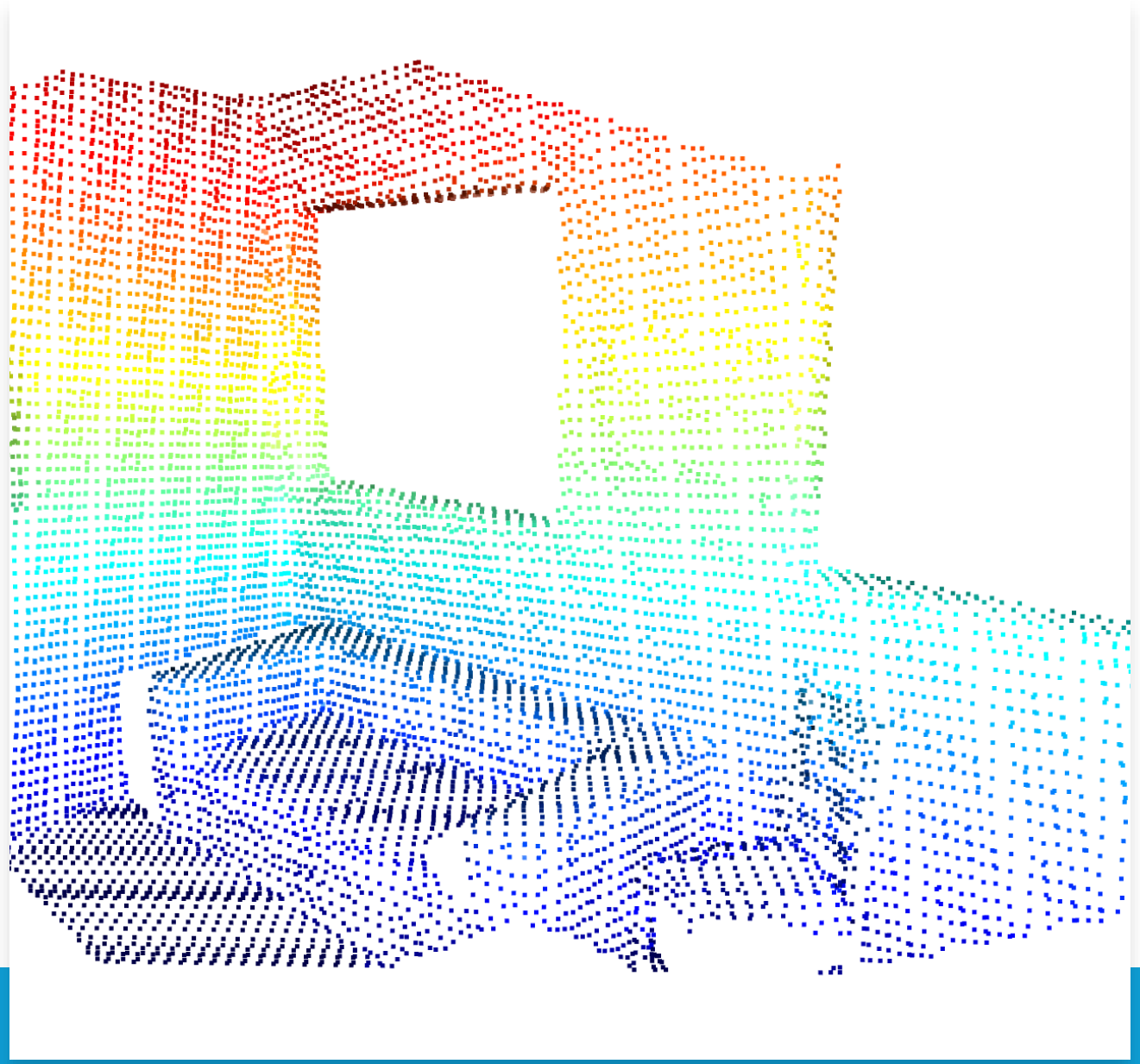
- Entrada
- Pose
- Mapa
- SLAM
- ICP
- ROS2

Implementación - Entrada

Entender la estructura de los datos que se maneja es crucial para poder sacar el máximo partido de estos. Los datos son proporcionados por:

- **Sensor RGB-D:** Este sensor incluye una cámara RGB y un sensor de profundidad. De esta forma puede entregar:
    - **Imagen RGB:** Tres matrices 2D (una para cada canal: rojo, verde, azul), donde cada una representa la intensidad del canal para cada pixel
    - **Nube de puntos:** Conjunto de puntos en coordenadas 3D (x, y, z).

La nube de puntos con la que tratamos puede contener errores, de esta forma, probamos la robustez del sistema . Conjunto de puntos en coordenadas 3D (x, y, z).



Implementación - Pose

La pose hace referencia a dos conceptos:

- Posición: Ubicación en el espacio (normalmente en coordenadas (x, y, z))
- Orientación: Dirección hacia la que esta orientado el objeto

La pose será representada por una matriz homogénea 4x4 que combina:

- Traslación (posición)
- Rotación (orientación)

Usar este tipo de matrices tiene muchas ventajas entre las que se encuentran:

- Unifican rotación y traslación
- Permiten encadenar movimientos fácilmente
- Son matemáticamente consistentes
- Son estándar en el desarrollo de robots y visión
- Facilitan la inversión y el cambio de marcos de referencia

$$T = \begin{bmatrix} R_{3\times3} & t_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix}$$

Donde:

- R es la matriz de rotación
- T es el vector de traslación
- La ultima fila es la parte homogénea

Implementación - Mapa

Un mapa voxelizado, comúnmente conocido como mapa, es una representación tridimensional donde el espacio se divide en una rejilla de pequeños cubos llamados voxeles. Cada voxel contiene la información sobre su ocupación. De esta forma, un mapa voxelizado se define bajo la siguiente función:

$$M\colon \mathbb{Z}^3 \rightarrow \{0,1,-1\}$$

Donde:

- $\mathbb{Z}^3$ representa los índices enteros de la rejilla 3D (i, j, k).
- $M\ (i, j, k)$ da el estado del voxel:
  - 0: No ocupado
  - 1: Ocupado
  - -1: Desconocido

Un parámetro importante de los mapas voxelizados es su resolución, es decir, el tamaño de cada voxel (volumen que representa)

Implementación - SLAM

SLAM es una técnica que permite al robot construir un mapa del entorno mientras se localiza dentro de él, al mismo tiempo.

En SLAM se suele implementar alguna técnica de cierre de bucle, es decir, que cuando el sistema reconoce que ha regresado a un lugar ya visitado, corrige el error acumulado en la trayectoria y el mapa.

Existe una variante de SLAM que no contempla el cierre de bucle, y es el tópico de interés de este proyecto para así reducir la cantidad de recursos necesarios para la ejecución. Además, sería posible aplicar el cierre de bucle a posteriori para corregir el mapa.

En general, SLAM se basa en una estimación bayesiana donde se busca estimar la probabilidad conjunta de:

$$p(x_t, m|z_{1:t})$$

Donde x_t es la posición del robot en el momento t, m es el mapa, $z_{1:t}$ la secuencia de observaciones.

Implementación - ICP

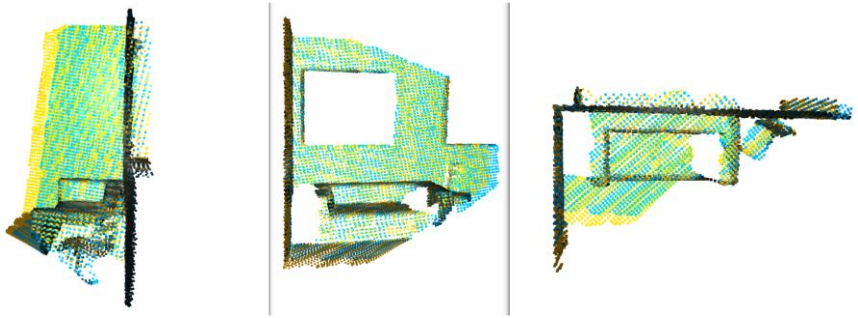
ICP (Iterative Closest Point) es un algoritmo de registro de nubes de puntos que se utiliza para alinear dos conjuntos de datos 3D, es decir, encuentra la mejor transformación (rotación, traslación) que alinea la nube de entrada a la nube de referencia.

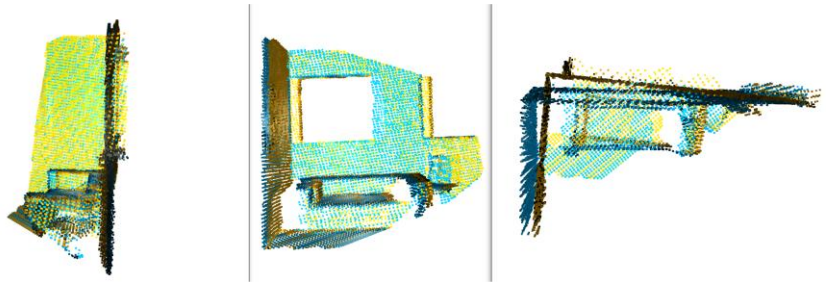
En resumen, su funcionamiento es el siguiente:

1. **Emparejamiento:** Para cada punto de la nube de entrada se encuentra el punto mas cercano en la nube de referencia.
2. **Estimación:** Se calcula la transformación óptima que minimiza la distancia entre los pares de puntos.
3. **Aplicación:** Se aplica la transformación a la nube de entrada.
4. **Iteración:** Se repiten estos pasos hasta que el cambio entre iteraciones sea mínimo (convergencia).

La implementación usada para este proyecto es proporcionada por la librería Open3D. Un detalle muy importante de esta implementación es la posibilidad de elegir distintos métodos de convergencia:

- **Punto a Punto:** Si basa en minimizar la distancia euclídea directa entre pares de puntos correspondientes.
- **Punto a Plano:** Minimiza la distancia entre un punto de la nube de entrada y el plano tangente de su punto de correspondencia en la nube de referencia (necesita del calculo de normales).





Implementación - ROS2

ROS2 (Robot Operating System 2) es un framework de software de código abierto diseñado para el desarrollo de aplicaciones robóticas. Este ofrece las siguientes características:

- **Comunicación distribuida:** Permite la comunicación en tiempo real entre nodos.
- **Modularidad:** Permite dividir un sistema robótico en múltiples nodos independientes.
- **Reutilización de componentes:** Gran cantidad de paquetes reutilizables.
- **Soporte para sistemas críticos:** Seguridad, calidad de servicio y ejecución en tiempo real.
- **Multiplataforma:** Soporte para Linux, Windows, MacOS y lenguajes como Python y C++.
- **Interoperabilidad:** Integración con simuladores.

Implementación – Nodos diseñados

Gracias a ROS2 podemos dividir las funcionalidades del sistema en diferentes nodos que nos permitirán ejecutar de forma flexible nuestro sistema en un entorno de simulación. Los nodos diseñados son los siguientes:

1. Nodo de gestión de datos de entrada
2. Nodo de adición de ruido (Opcional)
3. Nodo de localización
4. Nodo de mapeo

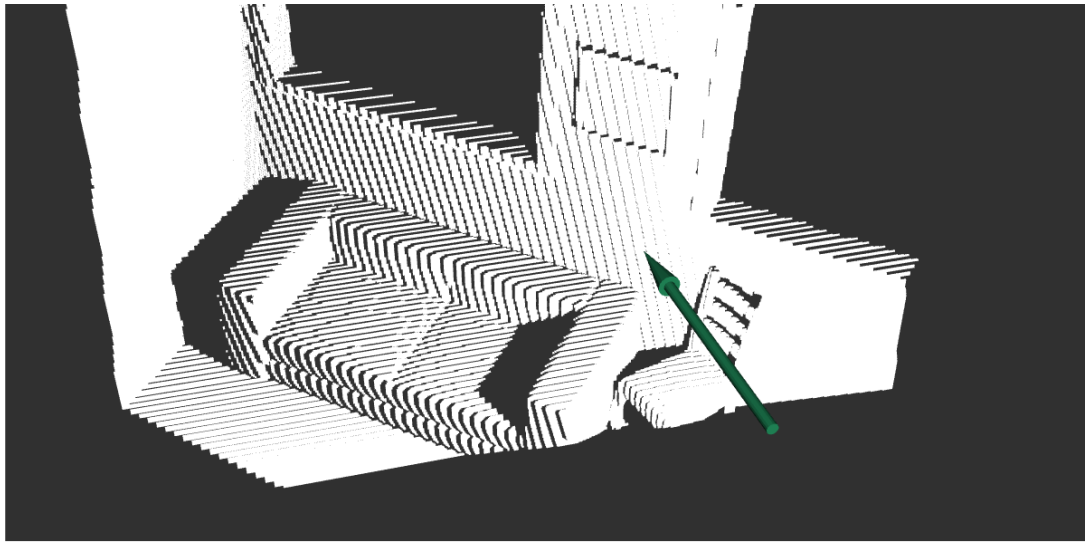
Todos estos nodos se comunican entre sí a través de un sistema de publicador/subscriptor y se sincronizan para asegurar la correcta ejecución del sistema.

Implementación – Nodo de entrada

Los datos de entrada se encuentran almacenados en una grabación rosbag (archivo que contiene los mensajes publicados durante su periodo de grabación).

Este nodo se encarga de:

1. Leer el archivo rosbag hasta que encuentra un mensaje publicado por el sensor RGB-D
2. Se deserializa este mensaje y se transforma al formato estándar de ROS2 para nubes de puntos (PointCloud2)
3. Se publica el mensaje a través el tópico “input_pcl”
4. Se espera a que el resto de los nodos hayan procesado la nube para volver a iniciar el proceso



Implementación – Nodo de adición de ruido

Este nodo es el encargado de añadir ruido a las nubes de puntos publicadas en el tópico “input_node”. Para ello, el nodo realiza las siguientes transformaciones:

1. **Mascara de probabilidad:** Se asigna una probabilidad aleatoria $u_i \sim \mathcal{U}(0,1)$ para cada punto i de la nube. Aquellos con probabilidad $p > u_i$ se le aplicara error.

2. **Distribución normal:** Se genera una variable aleatoria $n \sim \mathcal{N}(\mu, \sigma^2)$

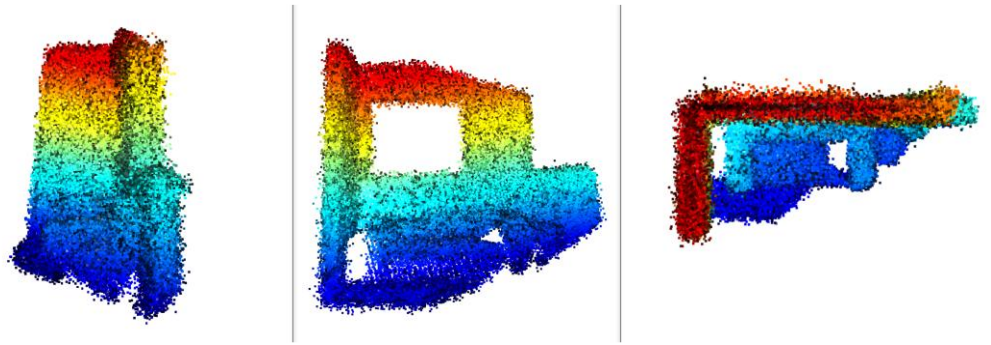
3. **Aplicación del ruido:** para cada punto seleccionado por la máscara modificando cada coordenada como:

$$x' = x + n_x \quad y' = y + n_y \quad z' = z + n_z$$

Donde $n_x, n_y, n_z \sim \mathcal{N}(\mu, \sigma^2)$ son muestras independientes de la distribución normal y (x', y', z') son las nuevas coordenadas

4. Se publica la nube de puntos ruidosa a través del tópico “noisy_pcl”

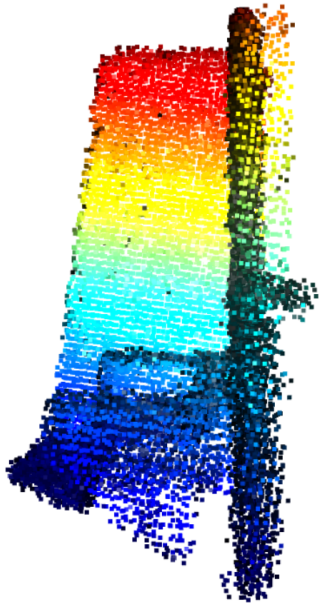
5. Se espera hasta recibir la siguiente nube de puntos

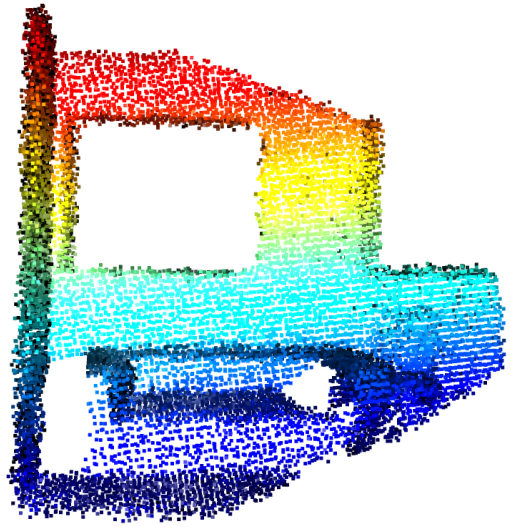


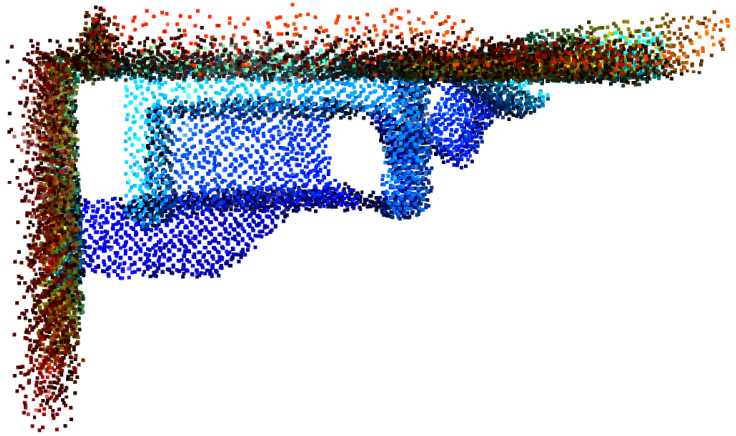
Implementación – Nodo de localización

El nodo de localización es el nodo central del sistema y se encarga tanto de calcular la pose del robot, como de evaluar que nubes de puntos son candidatas para fusionarse con el mapa. El funcionamiento de este nodo lo podemos dividir en dos fases:

1. **Primera llamada:** En esta llamada se inicializan la mayoría de parámetros necesarios para las sucesivas llamadas.
2. **Llamadas sucesivas:** En estas llamadas se recorre la lógica necesaria para calcular la variación de pose y si estamos ante una nube de puntos candidata para fusionarse al mapa







Implementación - Preprocesamiento

Esta función prepara la nube de puntos de entrada para que el resto de las funciones puedan usarla correctamente. Las transformaciones aplicadas son:

1. **Tipo de datos:** De PointCloud2 a ”open3d.geometry.PointCloud()”
2. **Submuestreo:** Se agrupan los puntos según la resolución de la rejilla del mapa
3. **Eliminación de outliers:** Se eliminan los puntos que se distancian mucho de sus vecinos (posibles outliers)
4. **Traslación al origen:** Las nubes de puntos recibidas vienen referenciadas a un sistema de coordenadas (0,0,0), para facilitar la convergencia de ICP, la desplazamos hasta la pose obtenida en el paso anterior
5. **Calculo de normales:** Necesario para realizar emparejamiento ICP “punto a plano”.

Implementación – Fotogramas Clave

Dada la naturaleza numérica y heurística de ICP, todos los emparejamientos contendrán un mínimo error, por esto, si componemos todas las nubes con sus respectivas transformaciones, la calidad del mapa degenerará rápidamente.

Para evitar esto, se definirán dos tipos de fotogramas clave:

- **Fotograma clave no perteneciente al mapa:** Nube que esta suficientemente alejado del fotograma clave anterior, pero que no añade nueva información
- **Fotograma clave perteneciente al mapa:** Nube que añade la suficiente información al mapa como que sea interesante fusionarla con el mismo

Implementación – Nodo de localización

En la **primera llamada** se llevan a cabo las siguientes operaciones:

1. **Preprocesamiento**
2. **Almacenamiento de la primera nube de puntos:** Se usará para calcular el desplazamiento respecto de la siguiente nube
3. **Almacenamiento del primer fotograma clave y del mapa:** Se almacenan en caso para comprobar si las siguientes nubes recibidas son candidatas para formar parte del mapa

Implementación – Nodo de localización

En las **llamadas sucesivas** se lleva a cabo el siguiente proceso:

1. **Preprocesamiento**
2. **Lógica ICP**
    1. **ICP fotograma a fotograma:** Se calcula ICP entre el nuevo fotograma y el anterior
        1. Para que este calculo sea aceptado se comprueba que:
            - Fitness > Umbral mínimo
            - RMSE < Umbral máximo
            - Distancia < Umbral máximo
            - Ángulo < Umbral máximo
    2. **ICP fotograma a fotograma clave:** Si el emparejamiento anterior no fue exitoso o han pasado n fotogramas desde el ultimo fotograma clave se realiza ICP entre el fotograma actual y el ultimo fotograma clave.
        1. Si el emparejamiento cumple una de las siguientes condiciones, se convierte en un nuevo fotograma clave:
            - Distancia > Umbral mínimo
            - Ángulo > Umbral mínimo
            1. Si cumple esto, alguno de estas condiciones, se comprueba si añade nueva información al mapa, entonces se calcula ICP fotograma a mapa. Si cumple la siguiente condición, no solo será un fotograma clave, si no que además formara parte del mapa:
                - Ratio de novedad > Umbral mínimo
3. **Publicación de la nueva información:**
    1. Si se ha encontrado un nuevo fotograma clave, se publica la nube de puntos y la pose
    2. Si no se ha encontrado un nuevo fotograma clave, se publica únicamente la pose

Implementación – Node de mapeo

Este nodo hace uso de la librería bonxai, que implementa una estructura jerárquica de voxeles dispersa, es decir, no es necesario reservar memoria para todos los voxeles que representen grandes volúmenes.

Se ha elegido esta librería por su eficiencia en memoria.

El funcionamiento de este nodo es el siguiente:

1. Espera a recibir una nube de puntos
2. Cuando recibe una nube de puntos la integra en el mapa
3. Publica el mapa actualizado

Resultados

En este apartado se definen métricas para evaluar la calidad del la pose calculada y la calidad del mapa resultante.

En las métricas se comparan los valores reales (ground-truth) respecto a los valores de las ejecuciones con ruido y sin ruido.

El valor real ha sido extraído directamente de la grabación rosbag capturada en el entorno de simulación.

Métricas sobre la localización

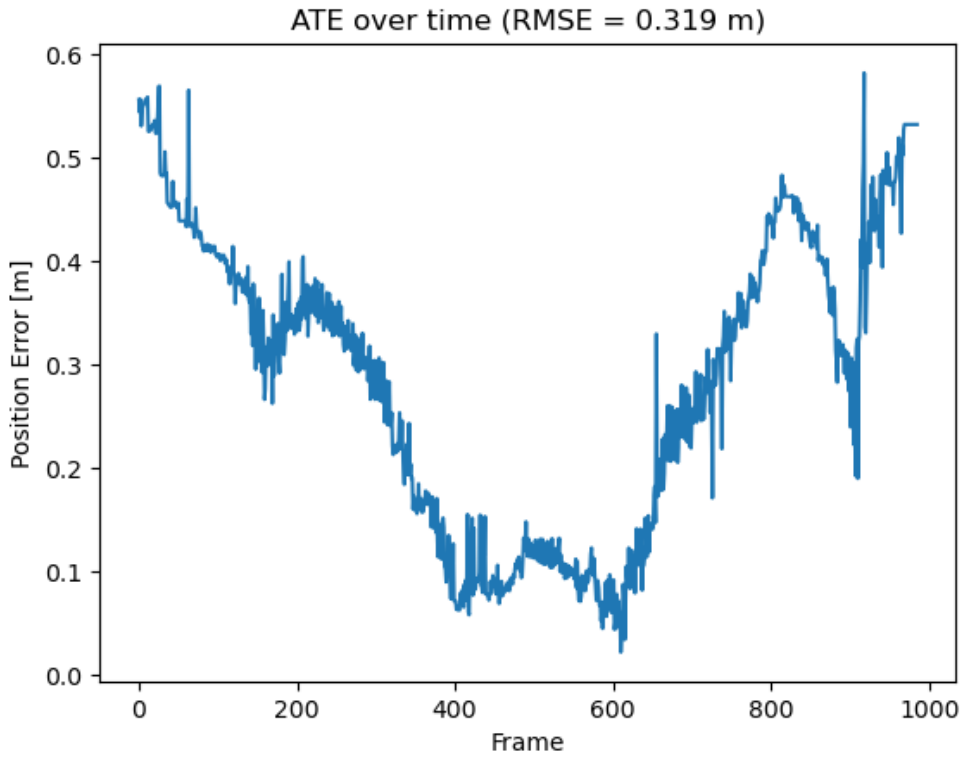
ATE (Absolute Trajectory Error)

Esta métrica se encarga de evaluar el error acumulado de la trayectoria en coordenadas globales.

Especialmente importante en escenarios sin cierre de bucle ya que revela de manera directa la deriva acumulada.

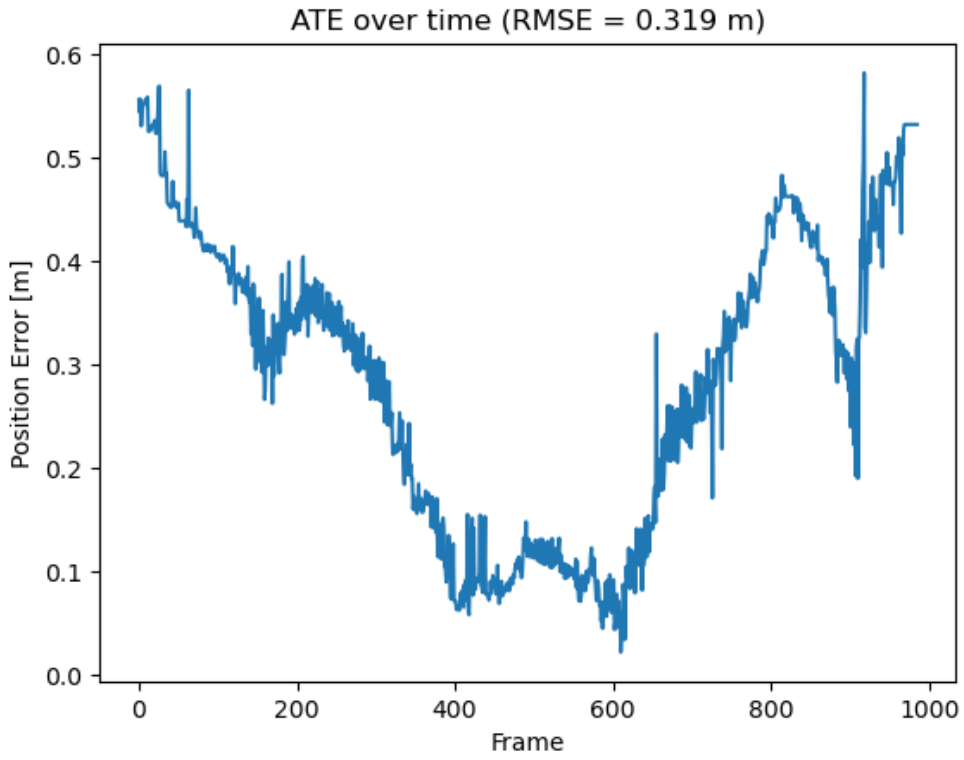
ATE – Ejecución sin ruido

- Valores obtenidos:
  - ATE (RMSE): 0.319 metros
  - ATE (Mediana): 0.312 metros
  - ATE (Máximo): 0.582



ATE – Ejecución con ruido

- Valores obtenidos:
  - ATE (RMSE): 0.314 metros
  - ATE (Mediana): 0.306 metros
  - ATE (Máximo): 0.589

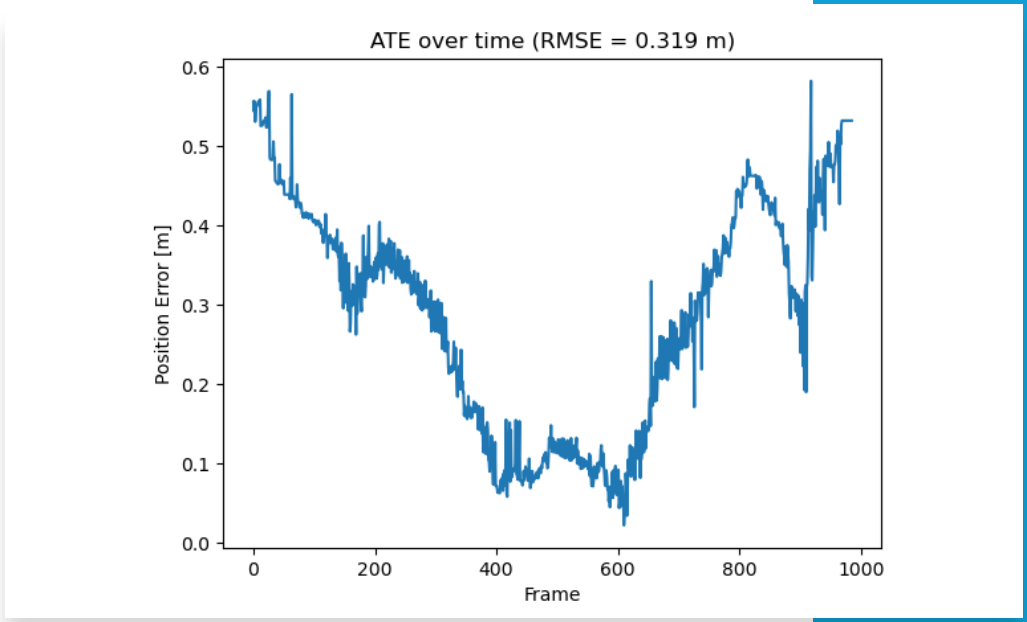


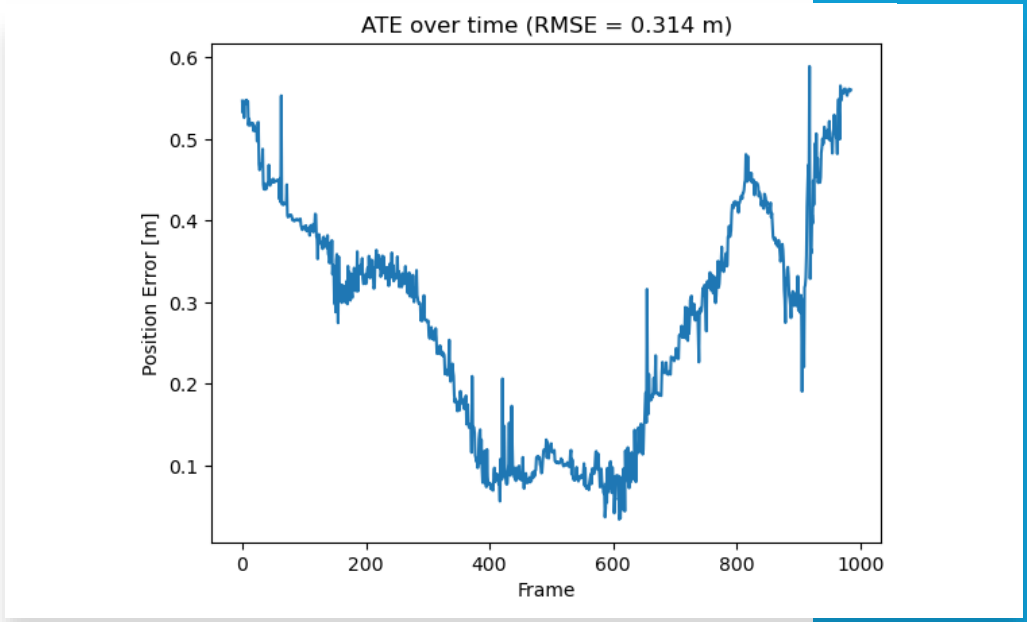
Interpretación de ATE

Los valores para ambas ejecuciones son muy similares, esto sugiere que el sistema es robusto frente al ruido.

Las pequeñas irregularidades vistas representan los momentos en los que se ancla de nuevo la pose al mapa.

Los grandes saltos vienen dados por irregularidades en el método de grabación de la pose.





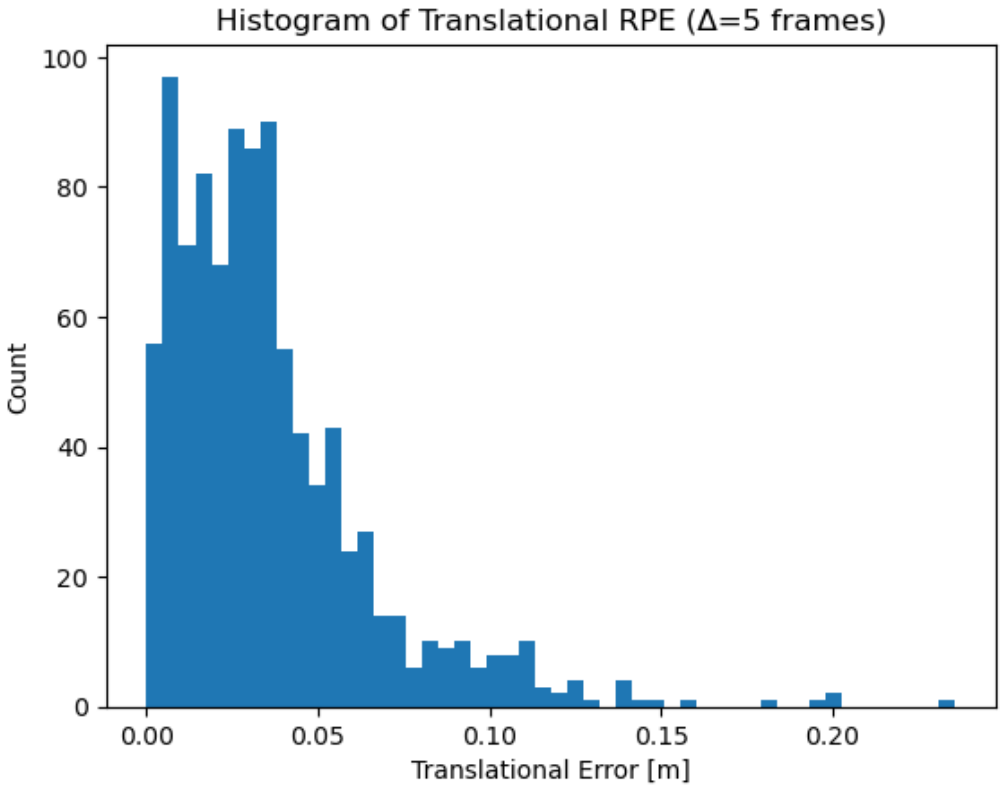
RPE (Relative Pose Error)

Esta métrica se encarga de evaluar la precisión local del movimiento relativo, es decir, nos aísla de la deriva global y se centra en pequeñas ventanas para calcular la deriva.

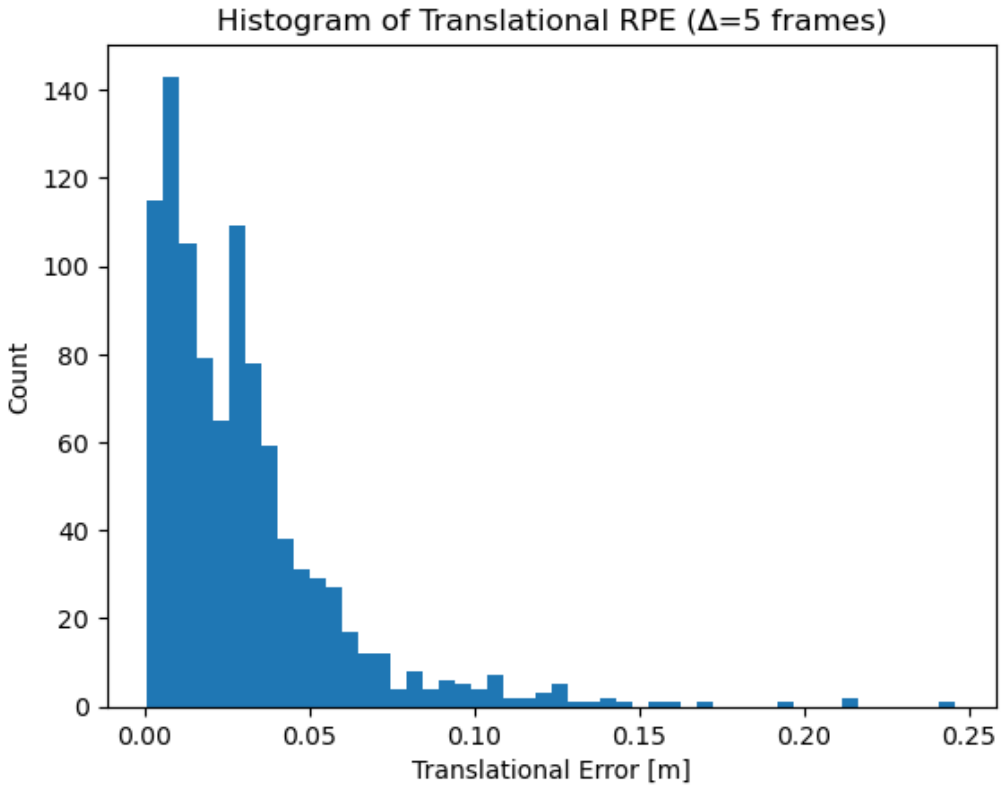
Útil para comprobar que, aunque la trayectoria global se haya desviado, se siguen haciendo estimaciones coherentes

RPE – Gráficos

Ejecución sin ruido



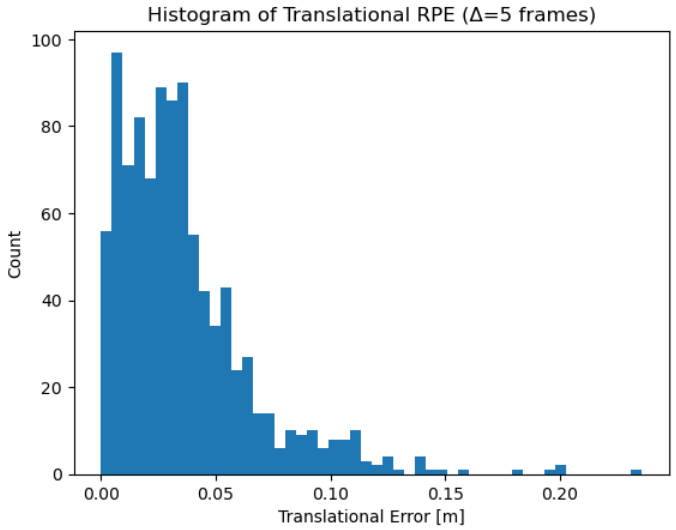
Ejecución con ruido

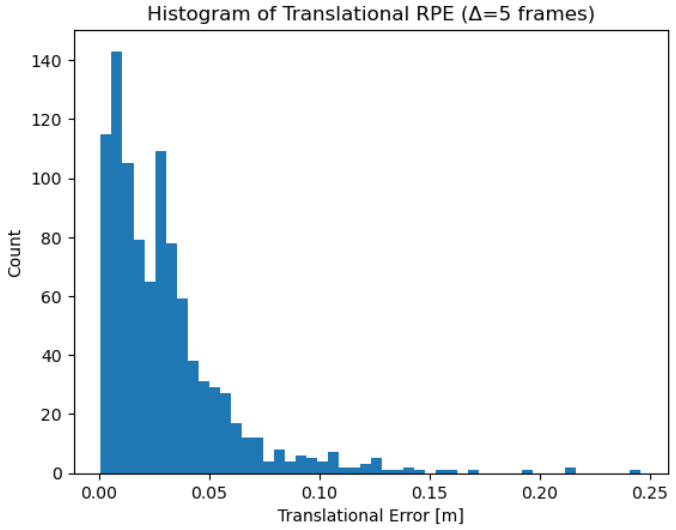


RPE - Interpretación

Los valores para ambas ejecuciones son muy similares, lo que muestra buena consistencia local.

Este valor se mantiene contenido en gran parte gracias a que el sistema se vuelve a anclar al mapa cuando se supera cierta distancia recorrida o cada n nubes recibidas





KITTY-stily Drift

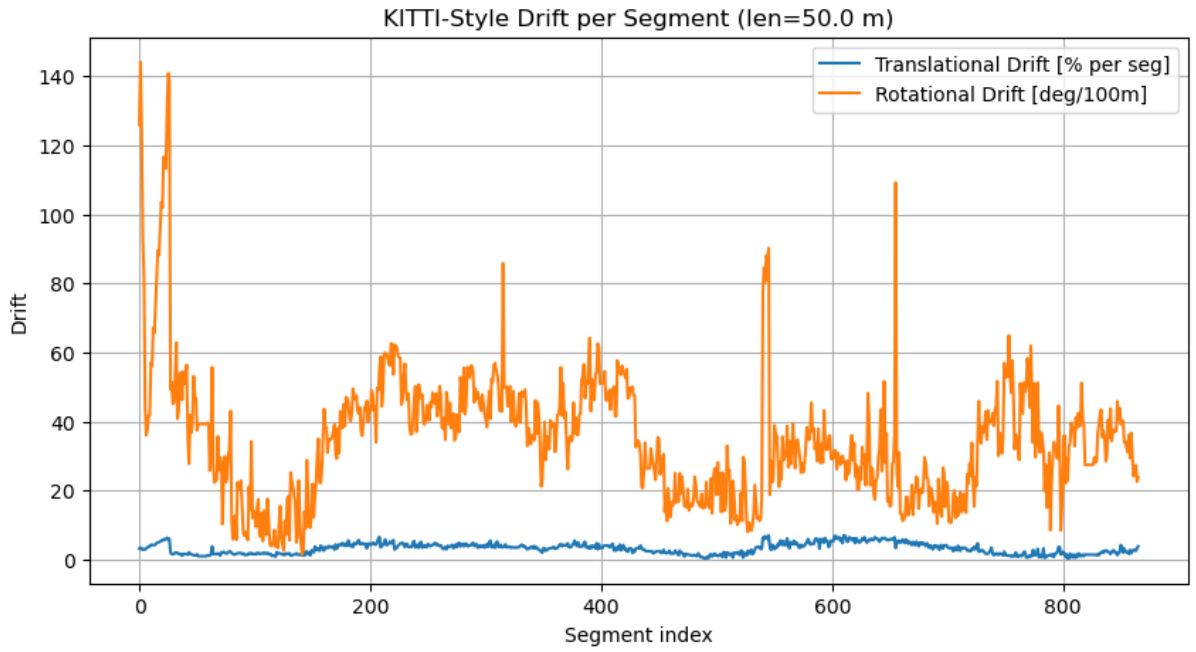
La métrica KITTY evalúa la deriva sobre tramos de longitud L. Para cada uno de estos segmentos, se calcula el error relativo de traslación y rotación.

Esta métrica ofrece una visión normalizada por distancia recorrida de la deriva, lo que permite comparar secuencias de distinta longitud.

KITTY – Ejecución sin ruido

Los valores obtenidos son:

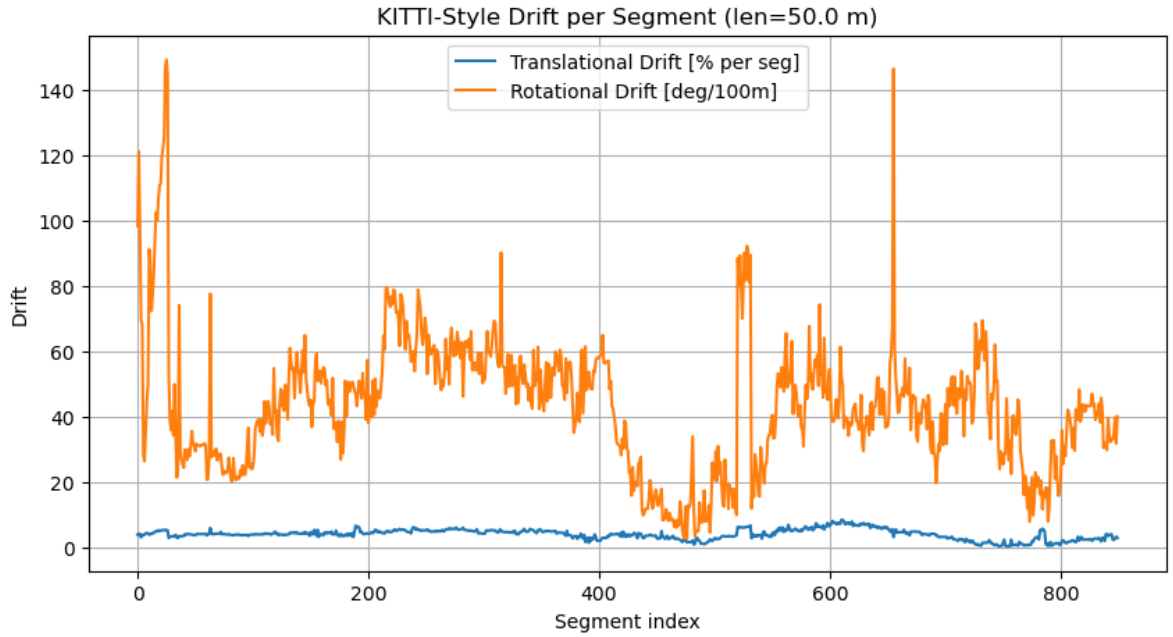
- Derrape traslacional, medio: 3.15%.
- Derrape traslacional, mediana: 3.21%.
- Derrape rotacional, medio: 34º/100 metros.



KITTY – Ejecución con ruido

Los valores obtenidos son:

- Derrape traslacional, medio: 4.13%.
- Derrape traslacional, mediana: 4.22%.
- Derrape rotacional, medio: 43º/100 metros.

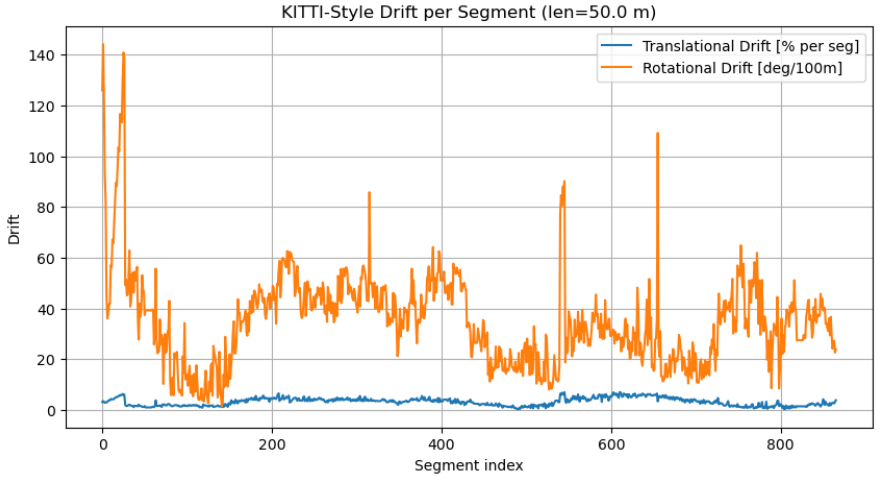


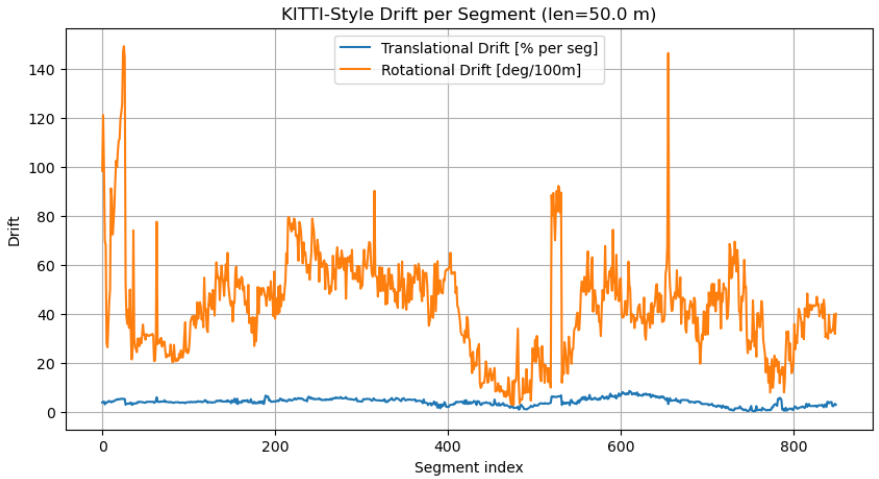
KITTY - Interpretación

Los resultados de KITTY indican que el sistema mantiene una tasa de acumulación de error razonable en ambas ejecuciones.

La mediana cerca de la media indica que los errores no están dominados por unos pocos segmentos atípicos.

El derrape traslacional sufre mas cuando se añade ruido



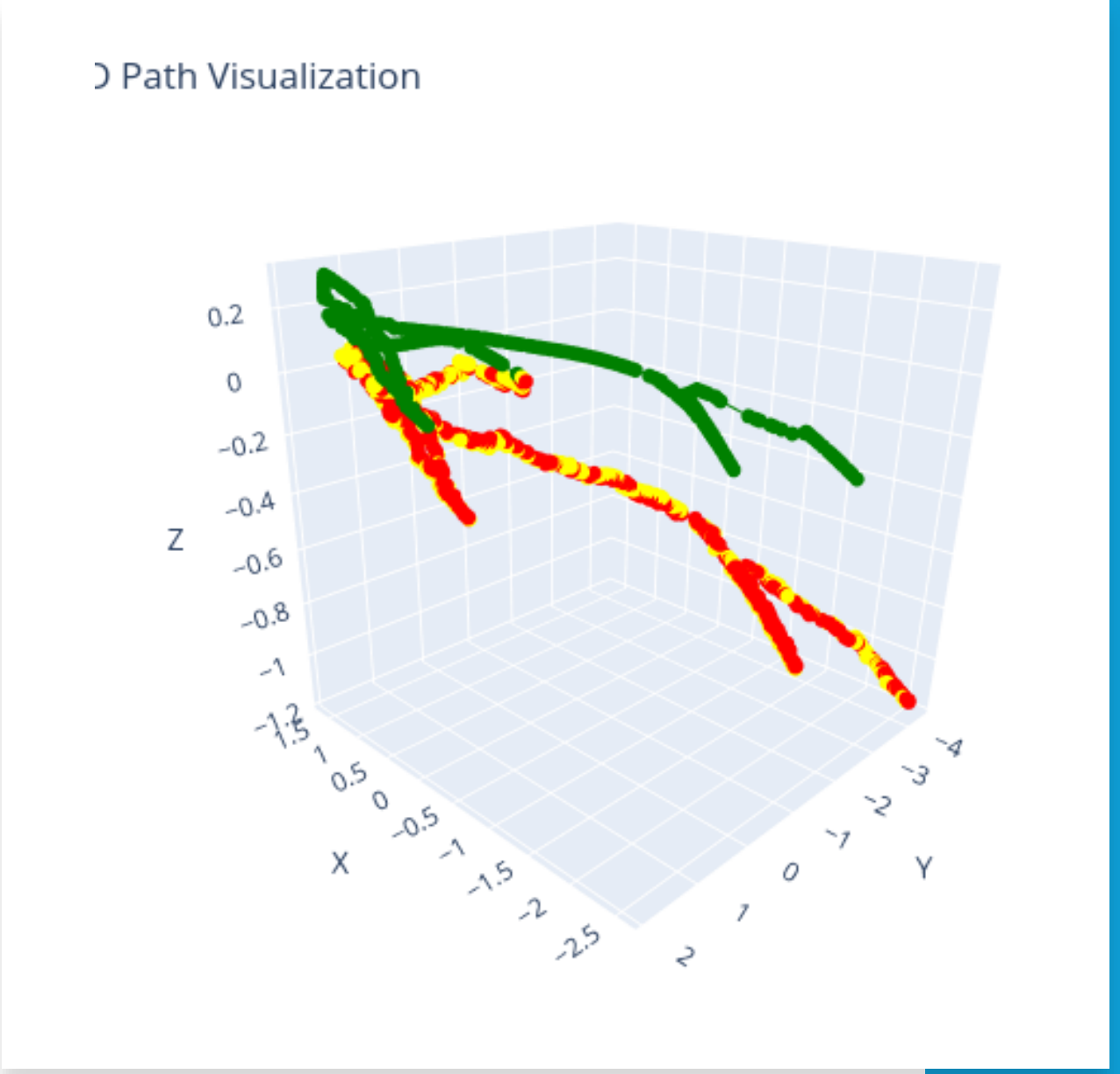


Métricas de localización - Resumen

En resumen, las métricas proporcionan una visión completa del rendimiento del sistema:

- ATE: Revela la precisión global de la trayectoria
- RPE: Evalúa la consistencia local del movimiento
- KITTY: Normaliza la deriva por distancia recorrida

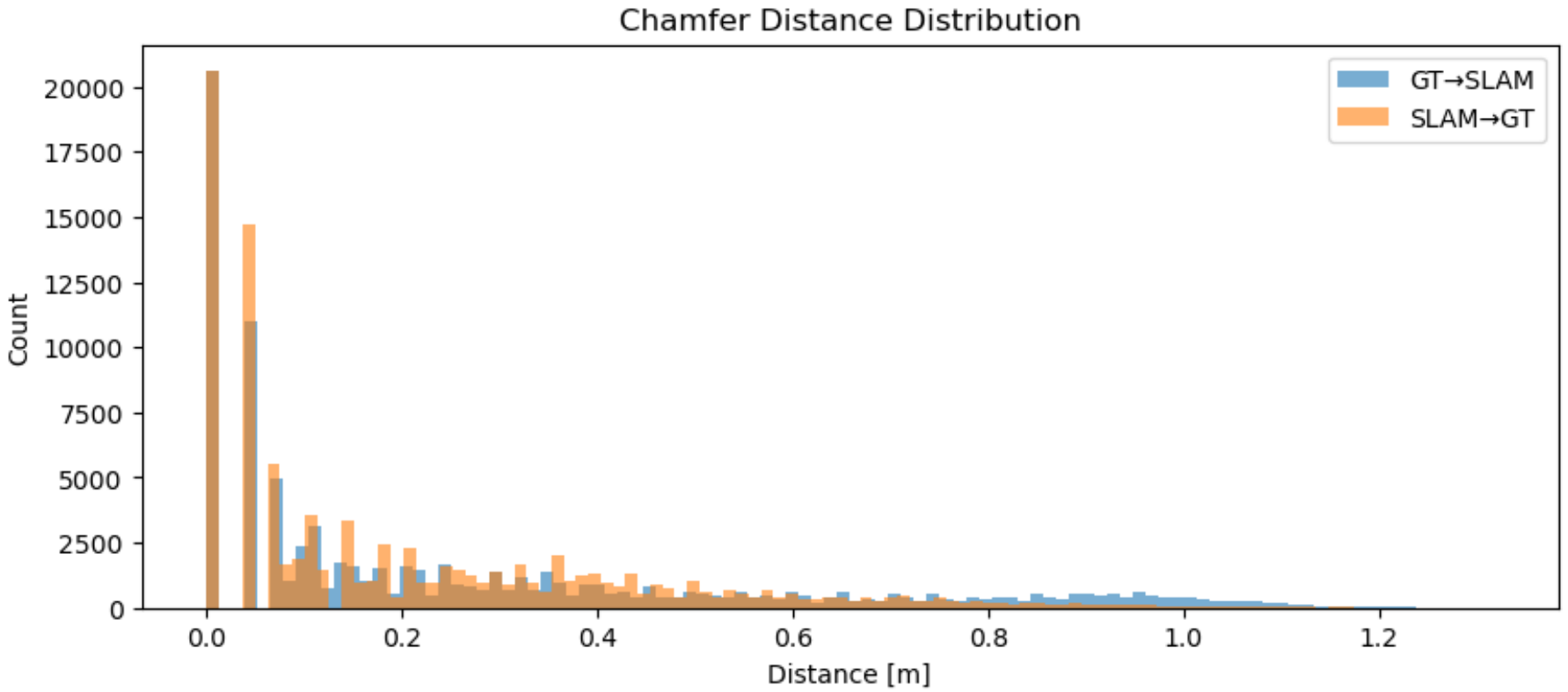
Haciendo una inspección visual de las trayectorias podemos ver como ambas trayectorias calculadas son muy similares y se van alejando de la original a medida que se avanza



Métricas de mapeo

Chamfer distance (Bidireccional)

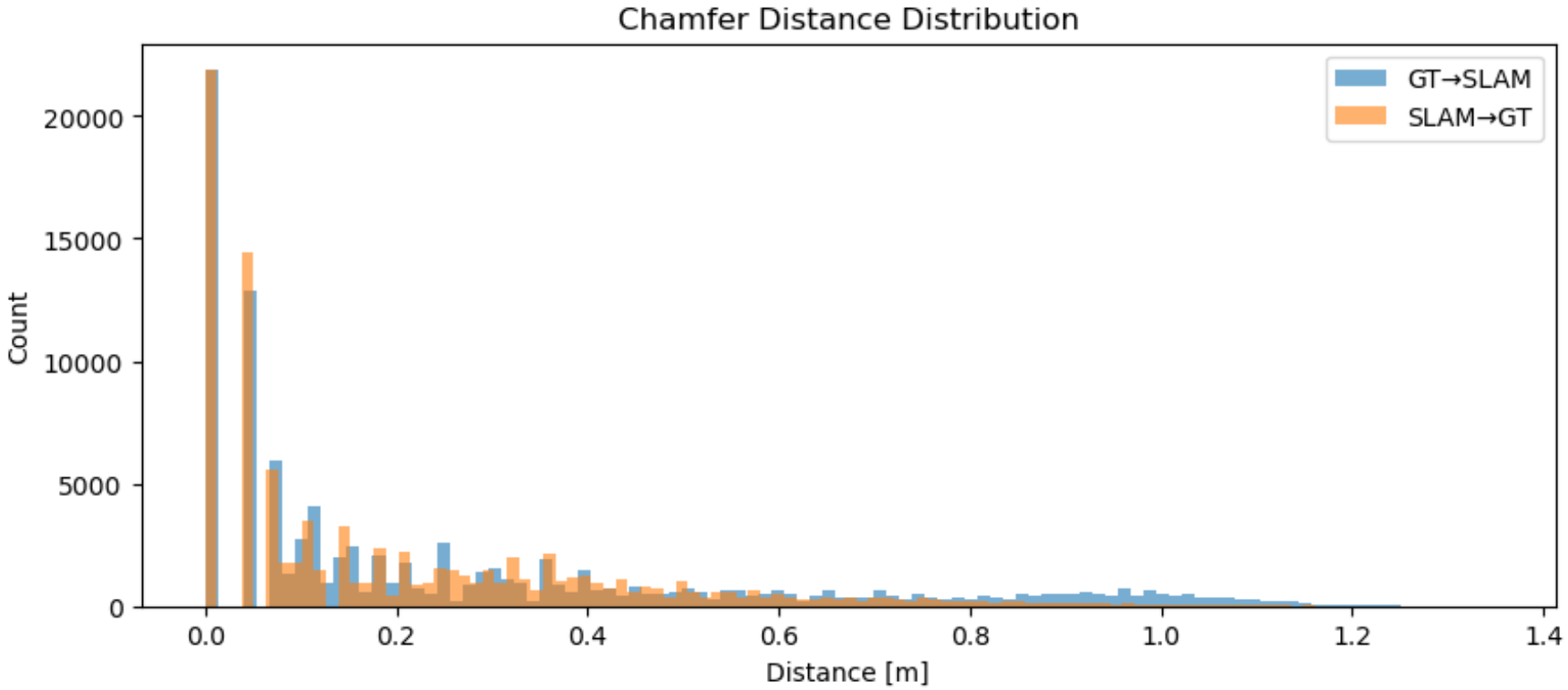
Esta métrica calcula para cada punto del conjunto, la distancia al punto mas cercano del otro conjunto. La forma en la que se calcula, evalua tanto la cobertura de la reconstrucción (si faltan regiones) como la exactitud (si se agregaron puntos espurios).



Chamfer – Ejecucion sin ruido

Los valores obtenidos son:

- Media Original->Calculado: 0.273 metros.
- Media Calculado->Original: 0.216 metros.
- Media Chamfer bidireccional: 0.122 metros.



Chamfer – Ejecucion con ruido

Los valores obtenidos son:

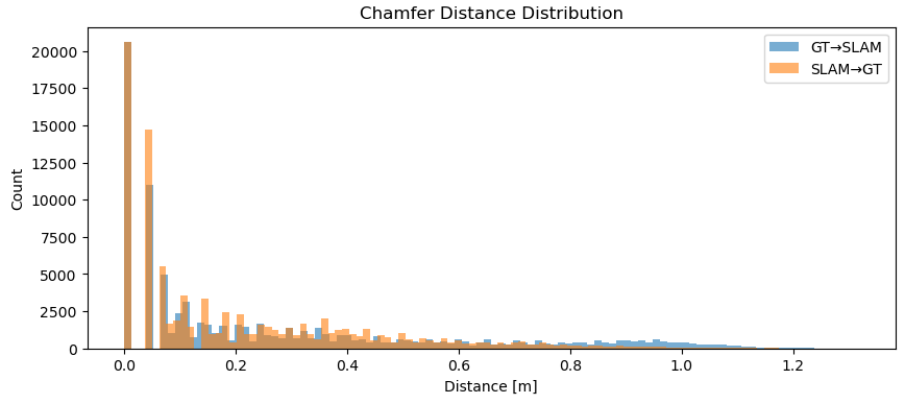
- Media Original->Calculado: 0.274 metros.
- Media Calculado->Original: 0.209 metros.
- Media Chamfer bidireccional: 0.122 metros.

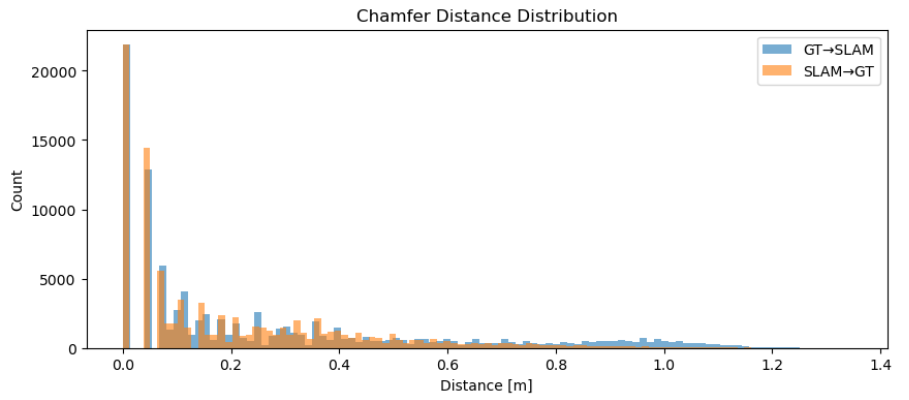
Chamfer - Interpretación

Los resultados indican que se logra una reconstrucción del entorno con cobertura y precisión razonables.

La media tanto en una dirección como en otra sugiere que la mayoría de los puntos están bien representados.

La media bidireccional indica que el promedio de los puntos del mapa estan a una distancia razonable de los puntos de referencia





F-Score

La métrica F-Score combina:

- Precisión: Proporción de las predicciones positivas que fueron realmente correctas
- Sensibilidad: Proporción de los positivos reales que fueron detectados

F-Score

Los valores obtenidos son en la ejecución sin ruido:

- F-score , τ = 0.2 metros:
  - Precisión: 0.567
  - Sensibilidad: 0.592
  - F-score: 0.579
- F-score , τ = 0.1 metros:
  - Precisión: 0.428
  - Sensibilidad: 0.447
  - F-score: 0.438
- F-score, τ = 0.05 metros:
  - Precisión: 0.323
  - Sensibilidad: 0.328
  - F-score: 0.325

Los valores obtenidos son en la ejecución con ruido:

- F-score , τ = 0.2 metros:
  - Precisión: 0.573
  - Sensibilidad: 0.601
  - F-score: 0.586
- F-score , τ = 0.1 metros:
  - Precisión: 0.427
  - Sensibilidad: 0.459
  - F-score: 0.442
- F-score, τ = 0.05 metros:
  - Precisión: 0.314
  - Sensibilidad: 0.338
  - F-score: 0.325

F-Score - Interpretación

Los resultados del F-SCORE indican que el sistema logra un equilibrio razonable entre precisión y sensibilidad.

Usando como referencia el umbral de 0.2 metros, se observa una precisión del 56,7% y una sensibilidad del 59,2% lo que sugiere que la mayoría de los puntos reconstruidos están cerca de los puntos de referencia y cubren una parte significativa de la escena.

A medida que hacemos el umbral mas estricto se observa como bajan los valores como es de esperar.

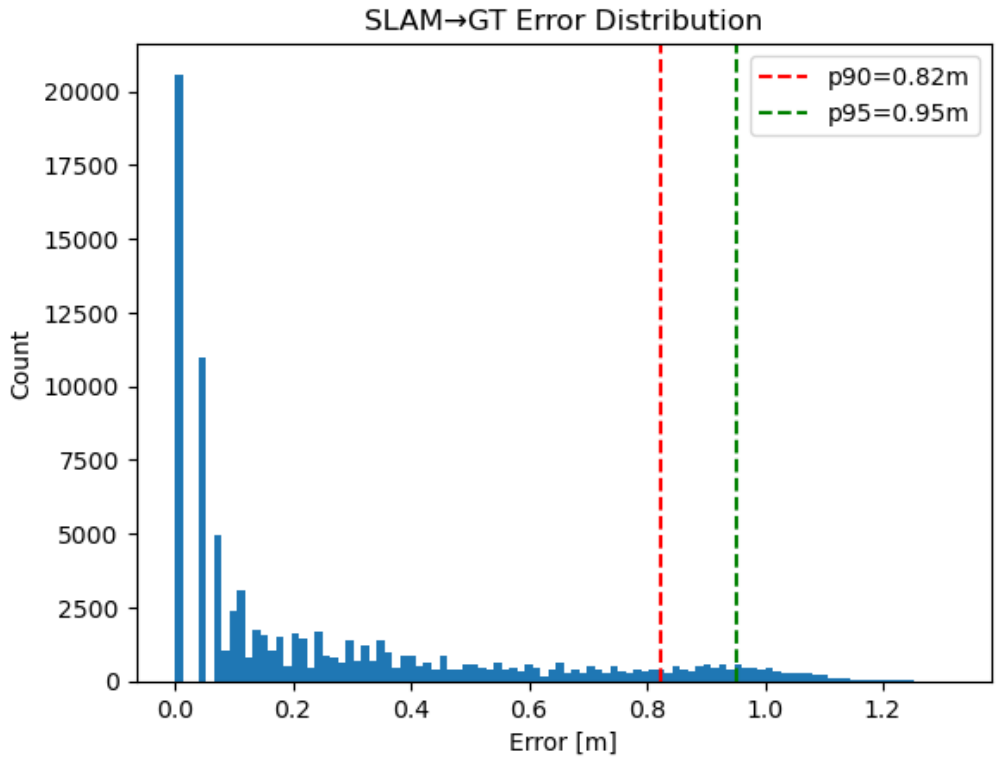
Distribución de percentiles

Esta métrica se encarga de analizar como de distribuyen las diferencias entre el mapa generado y el de referencia expresada en error por voxel, esto permite detectar si los errores se concentran en pocas regiones o si están repartidos de forma uniforme.

Distribución de percentiles – Ejecución sin ruido

Los valores obtenidos son:

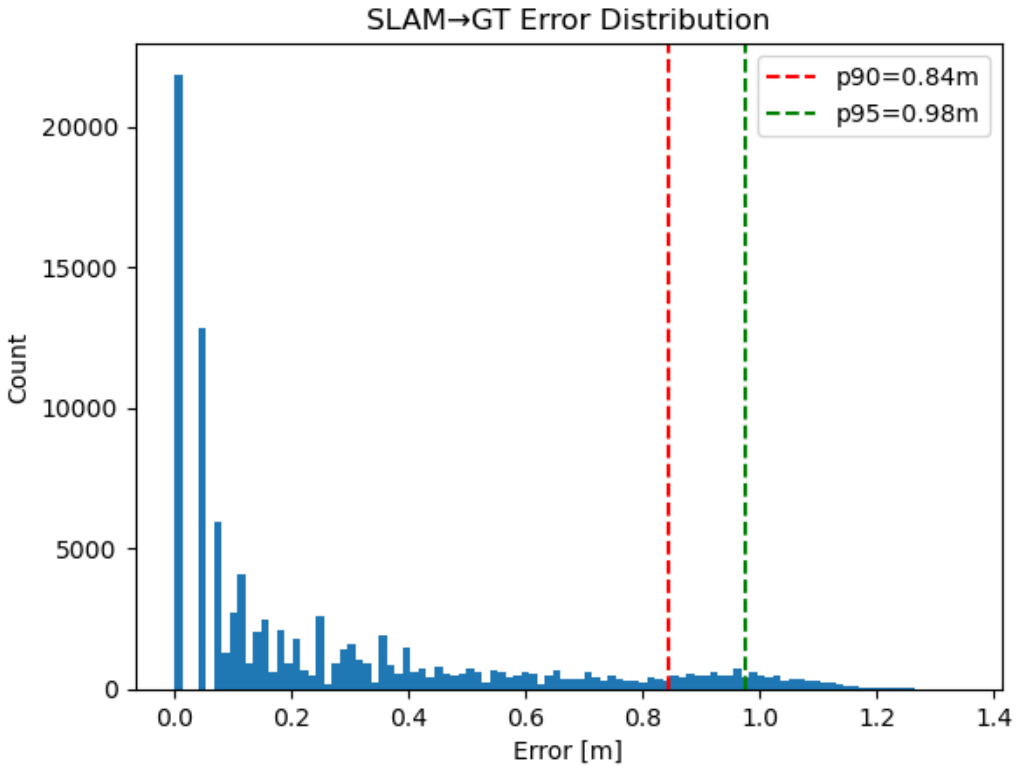
- Media: 0.141 metros.
- P90: 0.825 metros.
- P95: 0.953 metros.
- P99: 1.083 metros.



Distribución de percentiles – Ejecución sin ruido

Los valores obtenidos son:

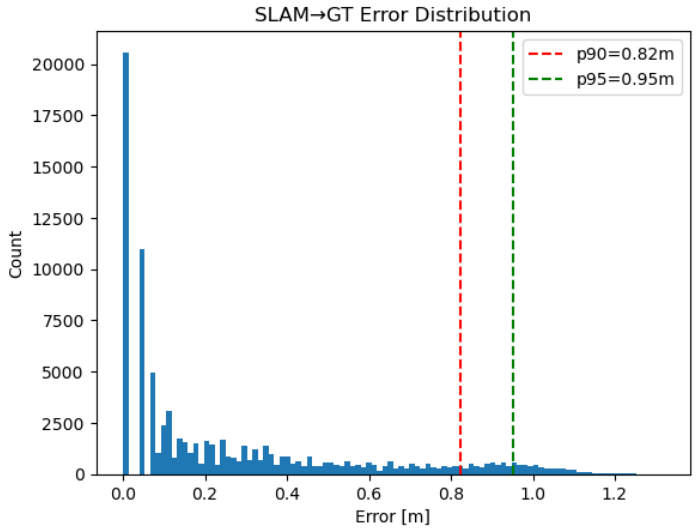
- Media: 0.141 metros.
- P90: 0.844 metros.
- P95: 0.976 metros.
- P99: 1.108 metros.

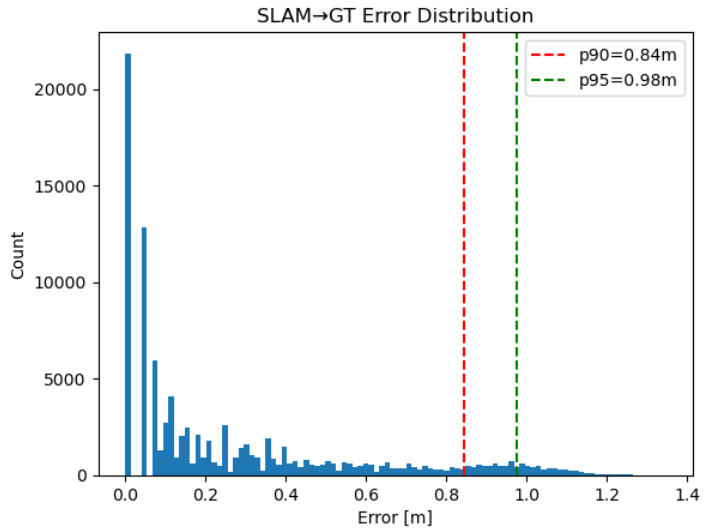


Distribución de percentiles - Interpretación

Los resultados muestran que se consigue una representación razonable del entorno.

La media de 0.141 metros sugiere que la mayoría de los puntos del entorno están bien representado. Sin embargo, en los percentiles 90, 95, 99 se puede ver como hay puntos alejados del mapa original, estos no son outliers si no resultado del drift acumulado



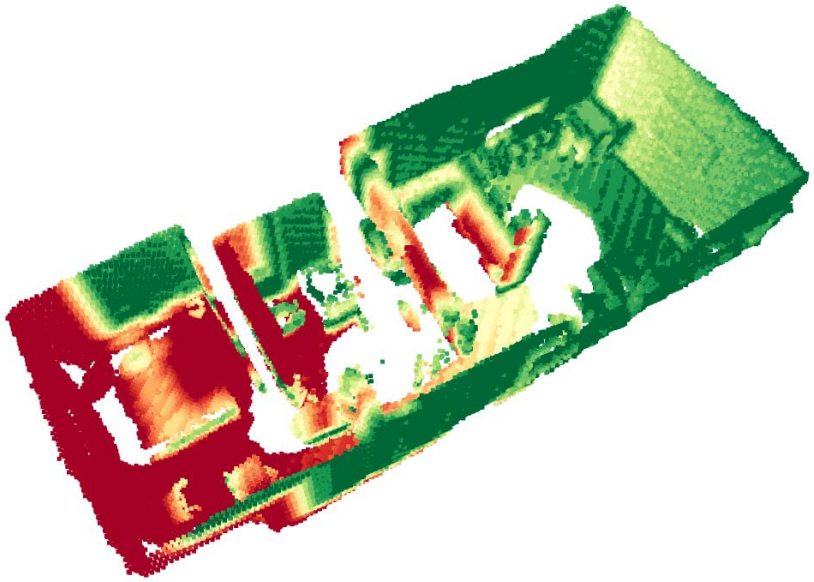


Métricas de mapeo - Resumen

En resumen, las métricas de mapeo proporcionan una visión completa del rendimiento del sistema:

- Chamfer: Evalúa tanto la precisión como la cobertura
- F-Score: Combina precisión y sensibilidad bajo distintos umbrales de tolerancia
- Distribución de probabilidades: Ofrece una visión detallada de la variabilidad del error en todo el volumen

Mapa de calor: Distancia de los puntos del mapa calculado al original



Conclusiones

En este proyecto se ha podido comprobar como haciendo uso de diferentes técnicas, es posible realizar SLAM sin cierre de bucle obteniendo unos resultados positivos incluso con ruido.

Esta implementación podría mejorarse con funcionalidades fuera del alcance de este proyecto como:

- Añadir cierre de bucle en tiempo de ejecución o a posteriori
- Integración de técnicas de aprendizaje automático para mejorar la correspondencia entre nubes
- Incorporación de otro tipo de sensores para mejorar la precisión

Muchas gracias por su atención