UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

IMPLEMENTACIÓN DE MAPEO Y LOCALIZACIÓN SIMULTÁNEA EN VEHÍCULOS AUTÓNOMOS UTILIZANDO FILTRO PHD Y VISIÓN COMPUTACIONAL

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

DIEGO MAURICIO PINCHEIRA IBAÑEZ

PROFESOR GUÍA:
MARTIN ADAMS

MIEMBROS DE LA COMISIÓN:
FELIPE INOSTROZA FERRARI
JAVIER RUIZ DEL SOLAR

SANTIAGO DE CHILE
2022

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: DIEGO MAURICIO PINCHEIRA IBAÑEZ
FECHA: 2022
PROF. GUÍA: MARTIN ADAMS

IMPLEMENTACIÓN DE MAPEO Y LOCALIZACIÓN SIMULTÁNEA EN VEHÍCULOS AUTÓNOMOS UTILIZANDO FILTRO PHD Y VISIÓN COMPUTACIONAL

El presente trabajo aborda la implementación de un sistema de Localización y Mapeo Simultáneo (SLAM) empleando información visual mediante el uso de Conjuntos Finitos Aleatorios (*Random Finite Sets, o RFS*) en el contexto de robótica, denominado como RFS-SLAM. Mediante una cámara RGB-D, es posible obtener información del entorno de tal manera de poder elaborar un mapa de este, y permitir que el dispositivo pueda ubicarse dentro de dicho mapa, sin necesidad de tener información previa.

El uso de Random Finite Sets en los algoritmos SLAM permiten que estos modelen entornos dinámicos sin necesidad de utilizar algoritmos de asociación de datos, cuya complejidad escala de manera exponencial al introducir nueva información proveniente de la trayectoria y el mapa, por lo que se plantea en este trabajo de memoria que el desempeño de algoritmos SLAM que se basan en RFS es más robusto al utilizar información visual frente a errores de detección.

Para la ejecución del algoritmo, se capturaron imágenes mediante una cámara RGB-D, capturando mediciones del entorno mediante descriptores ORB y la información provista por el sensor de profundidad de la cámara Kinect, estimando así la posición de objetos de interés relativos a la cámara a lo largo de una trayectoria. Dicha información se le entrega al algoritmo RFS-SLAM, que arroja tanto la trayectoria estimada como la ubicación de dichos elementos de interés en el espacio. Los resultados de dicha implementación permiten concluir un menor error a lo largo de la trayectoria en comparación a los comandos de movimiento realizados, a la vez de robustez de dicho resultado al aumentar la incertidumbre.

A mi Inti, mi rayito de sol, que sin saber
lo que es una carrera universitaria
estaría orgullosa de lo que he logrado.

Agradecimientos

Ha sido un largo camino en mi carrera, y estoy seguro que no habría llegado ni a la mitad de ella sin el apoyo incondicional de mis amigos y familia. Es gracias a mi mamá Paola, Katty, Manuel, Olivia, Elena, Lorna, Boris y a mis hermanas Emilia y Gabriela que logré sobrellevar los períodos más duros, dándome apoyo cuando más lo necesitaba.

Agradezco infinitamente a mi polola Valentina, que ha estado a mi lado estos últimos años de manera incondicional, siendo su amor y calidez parte íntegra de mis dias.

Me encuentro muy agradecido del cariño y del aguante de mis grandes amigos de plan común: Manuel Rodriguez, Marcial Benjamin D., Gianluca D'Agostino, Gerardo Flores, Felipe Mahu, Fernanda Canales, Paula Berrios, Sebastian Pizarro, Paolo Venegas, Pamela Canales, Polyn Cifuentes, que son parte importante de mi corazón. También agradezco de corazón todo el apoyo de mis amigas Baoyi Guo, Constanza Bustamante, Javiera Abrigo y Catalina Castillo, que me acompañaron y animaron en todo el período de especialidad.

Agradezco a mis compañeros de la Orquesta Beauchef, que me permitían escapar a través de la música y sus risas de tantos números y formulas. En particular, mis amigos Andrés Cárdenas, Felipe Cornejo, Camilo Carvajal, Felipe Gambardella, Almendra Huequelef, Joaquín Cruz, José Felipe F., Valentina Reyes, y a Fernando Aravena, que animaban mis sábados.

Agradezco el ánimo y respaldo de mis amigos queridos a lo largo de mi carrera: Rodriguito, Paul, Jocy, Pauli, Tomka, Jeremias, Chani, Abby, Seba S., Cota, Pablito, Vero, Vicente, Nico G., Jano y Alejandro Diaz. Su cariño y humor son una parte importante de mi vida, y los llevo siempre en mi corazón.

Agradezco mucho haber conocido a mis amigos de Coronel, con quien compartí horas y horas de diversión y cuyas risas y momentos me ayudaron a despejarme cuando más lo necesitaba: Aleeh, Enzo, Cote, Ronald, Maxi.

Por último, quiero agradecer a mi comisión, por tener paciencia con mis avances y guiarme a lo largo de este período.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Motivación

A medida que la robótica se ha ido integrando de manera más profunda al mundo, ha sido inevitable el surgimiento de algoritmos que permitan que un robot sea capaz de localizarse y desplazarse en un entorno que este no tiene registrado en su base de datos. Así, surgen los **algoritmos de localización y mapeo simultáneo** (Simultaneous Localization And Mapping, o SLAM), cuyo objetivo consiste en que un robot elabore un mapa del entorno y pueda a la vez ubicarse en él [1]. Las aplicaciones de dichos algoritmos son variadas y ya están presentes de manera cotidiana: la presencia de autos que no requieren conductor, aspiradoras robóticas, entre otros.

Para realizar dicha tarea, se requiere que el robot pueda interpretar el entorno a su alrededor [2]; de esta manera se ha dispuesto de sensores tales como LiDARs (Light Detection And Ranging)[1], que permiten una alta flexibilidad respecto a los entornos en los que se puede circular. Sin embargo, se presentan ciertas situaciones donde dicho uso de sensor recibe información ruidosa o errada, como por ejemplo, en situaciones con polvo o niebla. Como una alternativa, los algoritmos de Visual SLAM utilizan como entrada al sistema dispositivos que captan información visual del entorno mediante el uso de cámaras. De esta manera, mediante dicho flujo de información visual, se estima el mapa del entorno y la posición del robot en este mediante **elementos destacables presentes**, como lo son muebles dentro de una habitación; a medida que nueva información ingresa, se va actualizando la información presente, añadiendo además representaciones de lo que se observa a lo largo de la trayectoria [3].

A lo largo de los últimos años, han surgido métodos que permiten solucionar este problema utilizando tanto vectores como grafos, siendo ORB-SLAM (SLAM con el uso de características ORB, cuyas siglas significan *Oriented FAST*[2] *and Rotated BRIEF*[3]) [4] y RTAB-MAP (Real-Time Appearance-Based Mapping) [5] algunos de los algoritmos basados en dichas tecnologías respectivas. Sin embargo, estos algoritmos **presentan problemas de rendimiento frente a alto número de falsas alarmas y detecciones erróneas**; a la vez, debido al problema de asociación de información (*data association, o DA*), requieren complejos algo-

[1] Estos se conocen también como LaDARs (Laser imaging Detection And Ranging)
[2] FAST: Features from Accelerated Segment Test
[3] BRIEF: Binary Robust Independent Elementary Features

ritmos que permitan relacionar información en distintos instantes de tiempo [2].

Recientemente, se ha propuesto el uso de Random Finite Sets (RFS) para la resolución del problema SLAM [6]. Este posee la ventaja de trabajar mediante el uso de conjuntos finitos aleatorios, lo que evita el problema de asociación de información, otorgando así una **mayor robustez en la elaboración tanto de la trayectoria como del mapa frente a errores de detección**. Sin embargo, si bien se ha evaluado su desempeño desde un punto de vista teórico, sólo se ha comprobado de manera práctica con visión computacional de manera limitada [7] o utilizando LiDARs [8].

1.2. Descripción del proyecto

El trabajo a elaborar en la memoria abarca la implementación de un algoritmo SLAM utilizando Conjuntos Aleatorios Finitos (Random Finite Sets), empleando visión computacional y robótica, con el fin de estimar trayectorias ejecutadas, a la vez de generar un mapa del entorno de tales trayectorias.

Para ejecutar dicho algoritmo, es necesario recopilar previamente mediciones de un robot en movimiento que porte una cámara capaz de obtener la distancia del robot a elementos de interés existentes en su campo de visión. Una vez se han obtenido mediciones a lo largo de una trayectoria definida, se procederá a ingresar estas a un programa que ejecutará el algoritmo SLAM, obteniendo así la estimación tanto de la posición global de los elementos de interés observados, como también de la trayectoria seguida.

Para la implementación física, se utilizará una cámara RGB-D. Con esta, y utilizando un computador que permita la ejecución del algoritmo SLAM, se llevará a cabo la implementación a tiempo real antes mencionada en un entorno delimitado que permita un circuito cerrado.

1.3. Objetivos

1.3.1. Hipótesis

Dado el contexto actual de la robótica presentado anteriormente, se plantea como hipótesis de estudio que el uso de Random Finite Sets permite que algoritmos SLAM que utilicen visión computacional (Visual SLAM), poseen un **mejor desempeño frente a errores de detección en comparación con los comandos de movimiento**; además, adaptar tales algoritmos al uso de un **campo de visión** (en inglés *Field of View*, FoV) apropiado permitirán **mayor precisión en la asignación de probabilidad de detección**, contribuyendo así al aumento de su desempeño.

1.3.2. Objetivo general

De acuerdo a la hipótesis planteada anteriormente, se establece como objetivo general de este trabajo **adaptar un algoritmo SLAM que utilice Random Finite Sets (RFS-SLAM) con datos simulados, al uso de características provenientes del mundo real provenientes de una cámara RGB-D.**

1.3.3. Objetivos específicos

- Calibrar la cámara RGB-D a utilizar, calculando sus parámetros intrínsecos y extrínsecos.
- Grabar una base de datos pertinente a los experimentos utilizando una cámara RGB-D.
- Detectar objetos de interés en dicha base de datos mediante características visuales, y estimar la ubicación de estos en el espacio, generando mediciones.
- Implementar modelo de movimiento y modelo de medición en un algoritmo SLAM que utilice Random Finite Sets (RFS-SLAM).
- Adaptar el FoV del algoritmo al correspondiente de la cámara RGB-D en RFS-SLAM.
- Implementar métricas de error.

1.4. Alcances

Dada la naturaleza de la entrada descrita anteriormente, se delimitará la información a utilizar a aquella que pueda ser captada por un vehículo terrestre.

1.5. Estructura del informe

El informe se presenta bajo la siguiente estructura:

1. En el Capítulo 2 se ilustra el Marco Teórico, detallando las bases teóricas necesarias para la implementación del proyecto. Se presentan así conceptos como SLAM, Random Finite Sets, entre otros.
2. En el Capítulo 3, se postula la Metodología a seguir, describiendo los pasos a seguir para la implementación de la solución del problema.
3. En el Capítulo 4 se exponen los Resultados obtenidos.
4. En el Capítulo 5 se presenta la Discusión de los resultados obtenidos, argumentando los alcances de estos, a la vez de comentar las dificultades encontradas.
5. Finalmente, en el Capítulo 6 se detallan las Conclusiones del presente trabajo.

Capítulo 2

Marco Teórico

2.1. SLAM

Simultaneous Localization And Mapping, o SLAM, consiste en un problema en el ámbito de róbotica en el que un sistema móvil debe, en base a mediciones de los sensores presentes, estimar un mapa del entorno a la vez que evalúa la ubicación de este dentro de dicho mapa; a su vez, se conoce tambien como SLAM a la familia de algoritmos que ofrecen soluciones a dicho problema. Estos algoritmos son los principales métodos que poseen vehículos autónomos y robots para poder maniobrar en espacios que el sistema no posee información (o, en su defecto, posee información incompleta) [1].

2.1.1. Formulación matemática

La ejecución de SLAM se puede agrupar en tres pasos, que se operan en cada iteración del algoritmo [1]:

1. El robot se mueve a una determinada posición.
2. El robot observa y detecta elementos de interés (comúnmente llamados *landmarks*) ubicados en el entorno.
3. El robot compara los *landmarks* identificados actualmente con los registrados en las iteraciones anteriores del algoritmo.

Dado un tiempo t, se define al robot como un sistema de 6 grados de libertad en un espacio tridimensional, siendo sus componentes las coordenadas cartesianas X, Y, Z, y sus ángulos de navegación respecto a los ejes cartesianos (dirección/yaw, elevación/pitch, y ángulo de alabeo/roll).

La forma más habitual de definir el problema SLAM es formular este de acuerdo a **una distribución de probabilidad** [1]:

- Para ello, sea X_t la posición del robot en un tiempo t (definida típicamente a través de un vector), y $X_{0:T}$ la trayectoria del robot desde el tiempo inicial $t = 0$ hasta el tiempo $t = T$.

- A su vez, se define como un mapa $\mathcal{M}_t$ al conjunto de k elementos de interés (en inglés, ***landmarks***), $m_t^1, m_t^2, ..., m_t^{k'}$ presentes en un tiempo t.
- Por otra parte, el robot registra k' mediciones de dichos *landmarks* $z_t^1, z_t^2, ..., z_t^{k'}$, almacenando estos en un vector Z_t. La concatenación de todas las mediciones realizadas desde el tiempo inicial $t = 1$ hasta el tiempo $t = T$ se denota como $Z_{1:t}$.
- Con la información de la ubicación del robot y el mapa de su entorno, es posible definir el **estado del modelo** en un tiempo t como $Y_t = (X_t, \mathcal{M}_t)$.
- Además, el robot recibe comandos de control, que permiten que este se desplace a lo largo del entorno. Esto se representa para un tiempo t como U_t, siendo la concatenación desde el tiempo inicial $t = 1$ hasta el tiempo $t = T$ denotada por $U_{1:t}$.

Así, se define mediante la Ecuación 2.1 la **probabilidad conjunta del estado del robot y el mapa en un tiempo** t, dadas las observaciones y movimientos del robot realizados previamente [4] [7]:

$$p(X_{0:t}, \mathcal{M}_t | Z_{1:t}, U_{1:t}) \tag{2.1}$$

De esta manera, para resolver el problema SLAM, se debe estimar dicha distribución de probabilidad para cada instante de tiempo en la trayectoria; sin embargo, para aplicaciones en tiempo real, se vuelve redundante y computacionalmente costoso si se utilizan los valores desde el comienzo para estimar una trayectoria y mapa completo desde el principio. De esta manera, se aplican algoritmos de **filtrado**, que consisten en reutilizar la información obtenida en tiempos anteriores y actualizarla con la información proveniente de los sensores y odometría del tiempo actual t [7]. Así, se origina **Online SLAM**, que permite estimar la trayectoria y el mapa más recientes (es decir, en el tiempo t) [7][9] :

$$p(X_t, \mathcal{M}_t | Z_{1:t}, U_{1:t}) = \int ... \int \int p(X_{1:t}, \mathcal{M}_t | Z_{1:t}, U_{1:t}) dx_1 dx_2 ... dx_{t-1} \tag{2.2}$$

No obstante, si se cuenta con la información de sensores y movimiento grabada con anterioridad de la trayectoria, y no es necesario operar en tiempo real, es posible resolver dicho problema de manera completa, que se conoce como **Full SLAM** u **Offline SLAM** [9]. Para ello, se busca aquella trayectoria y mapa que maximicen la distribución de probabilidad de SLAM [6], es decir [7]:

$$X_{0:T}^*, \mathcal{M}_T^* = \underset{X_{0:T}, \mathcal{M}_T}{\operatorname{argmax}} \; p(X_{0:T}, \; \mathcal{M}_T | Z_{1:T}, U_{1:T}) \tag{2.3}$$

Por otra parte, retomando lo observado en la Ecuación 2.2, es posible abordar esta mediante el Teorema de Bayes [10]:

$$\begin{aligned} & p(X_t, \mathcal{M}_t | Z_{1:t}, U_{1:t}) \\ & \quad = \eta \; p(Z_t | X_t, \mathcal{M}_t) \int \int p(X_t, \mathcal{M}_t | U_t, X_{t-1}, \mathcal{M}_{t-1}) \\ & \qquad p(X_{t-1}, \mathcal{M}_{t-1} | Z_{1:t-1}, U_{1:t-1}) \; dX_{t-1} \; d\mathcal{M}_{t-1} \end{aligned} \tag{2.4}$$

[4] Aquí se debe destacar que, tanto u_t como z_t comienzan desde $t = 1$. Esto se debe a que primero se inicializa el estado del robot ($x_{t=0}$), luego se mueve el robot ($u_{t=1}$) y se realizan las observaciones ($u_{t=1}$), obteniendo así el estado del robot posterior ($x_{t=1}$).

Si se asume que el mapa es estático, es decir, $\mathcal{M}_i = \mathcal{M}_j = \mathcal{M}\ \forall i, j$, y además se incorpora el supuesto que el movimiento del robot es independiente al mapa generado, la Ecuación 2.4 se simplifica [10]:

$$p(X_t, \mathcal{M}|Z_{1:t}, U_{1:t}) \\ = \eta\ p(Z_t|X_t, \mathcal{M}) \int p(X_t|U_t, X_{t-1}) p(X_{t-1}, \mathcal{M}|Z_{1:t-1}, U_{1:t-1})\ dX_{t-1} \tag{2.5}$$

Así, se define la Ecuación 2.5 como un **filtro de Bayes**, que se agrupa en dos etapas [10]:

- **Predicción**: Aquí, se considera el estado anterior del sistema, es decir, $p(X_{t-1}, \mathcal{M}|$ $Z_{1:t-1}, U_{1:t-1})$; y se itera este con la transición de estado entre X_{t-1} y X_t dada por U_t, , permitiendo realizar una **estimación de la trayectoria del robot** dada dicha información. Esto se representa mediante el término a la **derecha** de la Ecuación 2.5:

$$\int p(X_t|U_t, X_{t-1}) p(X_{t-1}, \mathcal{M}|Z_{1:t-1}, U_{1:t-1})\ dX_{t-1} \tag{2.6}$$

  - Se observa también que el estado anterior del sistema permite establecer una **recursión** del filtro, dado los términos $p(X_t, \mathcal{M}|Z_{1:t}, U_{1:t})$ y $p(X_{t-1}, \mathcal{M}|Z_{1:t-1}, U_{1:t-1})$.
  - Por otra parte, el término $p(X_t|U_t, X_{t-1})$ se interpreta como el **modelo de movimiento** del sistema (dado el estado del robot previo y un comando de control, se actualiza el estado del robot).
  - Dado lo anterior, es posible interpretar la Ecuación 2.6 como una distribución de probabilidad a *priori*.

- **Corrección**: Esta sección del filtro se encarga de incluir la información de las mediciones tomadas del mapa en el tiempo actual, ajustando la predicción realizada anteriormente y actualizando el mapa, siendo esto formulado mediante la parte **izquierda** de la Ecuación 2.5:

$$\eta\ p(Z_t|X_t, \mathcal{M}) \tag{2.7}$$

  - El término $p(Z_t|X_t, \mathcal{M})$ corresponde al **modelo de medición** del sistema (dado el mapa y el estado del robot, se extraen mediciones de los elementos pertenecientes al mapa).
  - Finalmente, el término η corresponde a un factor que normaliza la ecuación y permite una correcta distribución de probabilidad.
  - Así, es posible interpretar la Ecuación 2.7 como una distribución de probabilidad a *posteriori*.

2.1.2. Frontend y Backend

El funcionamiento de SLAM se puede desglosar en dos partes: ***frontend*** y ***backend***. El primero consiste en la implementación de métodos que procesan la información proveniente de sensores, transformando esta en una representación que sea posible de procesar de manera matemática (como por ejemplo, una distribución de probabilidad de un determinado objeto ubicado en el entorno, restricciones en un problema de optimización, entre otros). Por otra parte, *backend* toma dicha representación matemática para resolver las ecuaciones necesarias

para obtener el estado del sistema [1].

El *frontend* de SLAM es posible dividirlo en subconjuntos de tareas específicas: la medición de movimiento del vehículo, la identificación de secciones de interés dentro de las mediciones del entorno, la correspondencia de características entre un instante t y el instante $t-1$, y la identificación de circuitos cerrados. Por otra parte, se tienen 3 familias de algoritmos que son las más usadas dentro de *backend* [1]:

- **Filtros de Kalman Extendidos** (*Extended Kalman Filter, EKF*) [9] [11]:
  - Los Filtros de Kalman representan las distribuciones de probabilidad del Filtro de Bayes (Ecuaciones 2.6 y 2.7) asumiendo distribuciones **Gaussianas**: dado un tiempo t, dichas distribuciones de probabilidad quedan representadas por un estimado de la media μ_t y la covarianza Σ_t; además, se asume que tanto los modelos de movimiento y medición son **lineales**.
  - Sin embargo, lo anterior no sirve cuando los modelos no son lineales; para ello, surge el **Filtro de Kalman Extendido**, que aborda dichas no-linealidades aproximando estas mediante un **desarrollo de series de Taylor**.
- **Filtros de partículas** [12] [13]:
  - Mediante un conjunto aleatorio de cúmulos de puntos, denominados **partículas**, se representa la probabilidad a posteriori del filtro de Bayes.
  - Cada una de las partículas representa una posible posición del robot en la ubicación de la misma; a medida que se itera el algoritmo, las partículas que presentan una mayor precisión propagan la información para las siguientes iteraciones del filtro, mientras que las restantes se les asigna un menor peso, o son directamente eliminadas.
  - Los pasos de dicho filtro de partículas son los siguientes [13]:
    1. Se inicializa un conjunto de partículas con disposición aleatoria y de igual peso, que representan la distribución de probabilidades del estado del robot.
    2. Cada partícula se actualiza con el estado del robot dado el comando de control.
    3. Se obtienen las mediciones, se actualiza el estado del robot con dicha información, y se idea un mapa local.
    4. Se ajusta el peso de cada partícula de acuerdo a la variación entre las mediciones previas y las actuales, a la vez de comparar el mapa local con el mapa global.
    5. Finalmente, se realiza un **remuestreo** de las partículas de acuerdo a sus pesos.
  - Cabe destacar que los pasos 1 y 2 corresponden a la **predicción del filtro de Bayes**, mientras que los pasos 3, 4 y 5 corresponden a la **corrección del fitro de Bayes**.
- **Algoritmos basados en grafos** [5] [14]:
  - Mediante este método, se busca elaborar un grafo que mejor represente la trayectoria del robot y el mapa dado las observaciones y controles de movimiento.
  - Para ello, se tiene que tanto la información de la ubicación y orientación del robot como las mediciones realizadas en ese punto se agrupan en **nodos**.

- Por otra parte, las **aristas** representan restricciones respecto al movimiento a lo largo de la trayectoria.
- Una vez se tienen los nodos y aristas, se busca optimizar estos tal que el grafo obtenido se ajuste lo mejor posible a las restricciones impuestas.
- Así, se le llama *front-end* a la construcción del grafo, mientras que se denomina *back-end* como la optimización del mismo [5].

2.1.3. Tecnologías y sensores utilizados para realizar mediciones

En robótica, con el fin de interactuar con su entorno, se requiere el uso de **sensores**, que permiten interactuar con el entorno mediante observaciones de objetos de interés. Para ello, se dispone principalmente de cuatro sistemas de sensores [2]:

- **Light Detection and Ranging Sensors (LiDAR)**: Este sistema corresponde al uso de señales electromagnéticas para realizar mediciones en base a las diferencias entre la señal emitida y la señal reflejada [2]. Para ello, se presentan dos principales métodos de cálculo de distancia:
  - **Time-of-Flight** [6]: En este método, un transmisor inicia una cuenta en un reloj interno, a la vez que emite un **pulso** de luz en la dirección objetivo. Este pulso de luz, al momento de rebotar en el blanco y retornar hacia el emisor, pasa por un fotodetector, que registra la diferencia entre la emisión y llegada de la onda electromagnética. Debido a que el láser utiliza pulsos de luz, y la velocidad de esta es conocida, la ecuación que determina la distancia del objeto se define mediante la Ecuación 2.8:

    $$d = \frac{c_m \cdot t}{2} \tag{2.8}$$

    Donde c_m representa la velocidad de la luz en el medio, y t el tiempo registrado en el reloj interno (es decir, cuanto se demoró el pulso de luz en ir al objetivo y posteriormente retornar).
  - **Desplazamiento de fase**: A diferencia del método anterior, donde se emite un pulso de luz, para este método se utiliza un **haz de luz sinusoidal continuo** con una frecuencia conocida y constante. De manera similar al método anterior, el haz de luz es emitido en dirección a un objetivo, y al momento de hacer contacto, una porción de dicha luz retorna de manera paralela y en sentido contrario a la onda emitida. Al momento de hacer contacto con el fotodetector, se registra la diferencia de fase entre la onda emitida y reflejada, permitiendo estimar la distancia del objetivo mediante la Ecuación 2.9:

    $$d = \frac{c_m \cdot \Delta\phi}{2\pi \cdot f} \tag{2.9}$$

    Donde c_m representa la velocidad de la luz en el medio, $\Delta\phi$ es la diferencia de fases, y f es la frecuencia del haz de luz emitido.

[5] No confundir con el *front-end* y *back-end* de los algoritmos SLAM.
[6] En español, se conoce como "tiempo de vuelo".

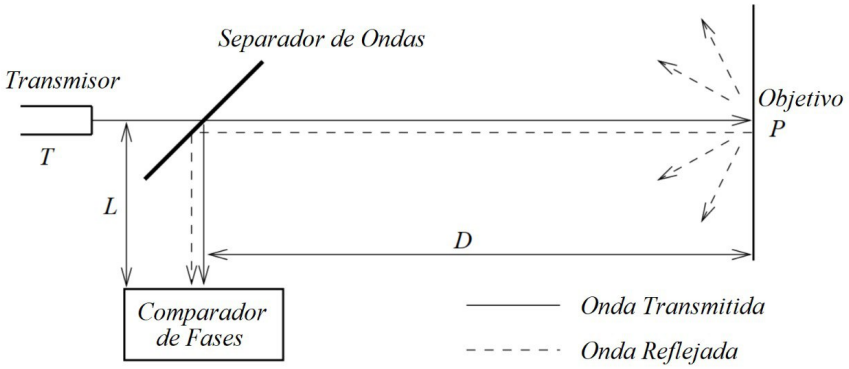

Figura 2.1: Imagen extraída y traducida de [2], donde se observa el funcionamiento de un LiDAR con desplazamiento de fase. Se emite un pulso láser en el transmisor al objetivo P. Este láser pasa a través de un separador de rayos, redirigiendo una fracción de este a un comparador de fases. Una vez que llega a su objetivo P, el láser se refleja en la superficie, siendo redirigido de vuelta también al lector de fases. Este último compara la diferencia de fases entre láser enviado y recibido para calcular la distancia aproximada D del punto P.

Estos sensores son uno de los más usados actualmente debido a su alta precisión, y al bajo costo computacional para el procesamiento de la entrada al sistema; sin embargo, son caros y presentan problemas graves de medición en determinadas condiciones atmosféricas o superficies (por ejemplo, una alta presencia de polvo en el ambiente, o al intentar realizar mediciones bajo el agua) [2].

- **Radio Detection and Ranging (RADAR)**: Los radares se caracterizan por el uso de **ondas de radio** para la localización de objetos en el entorno. Para ello, se emiten ondas electromagnéticas en la dirección objetivo, registrando la distancia de objetos en base a la onda retornada debido al contacto. Generalmente se componen de cuatro elementos: un transmisor, encargado de generar la onda electromagnética; una antena que permite emitir o recibir dichas ondas; un interruptor que se encarga de indiciar el modo de la antena, es decir, cuando emite o cuando recibe pulsos; y finalmente un receptor, que se encarga de procesar la señal recibida [15].

  Existen dos tipos principales de radares [15]:

  – **Radar de pulsos**: Tal como su nombre indica, se hace uso de **pulsos de ondas de radio** para la detección de objetos. De manera similar a LiDAR, se estima su distancia mediante ToF: con la antena en modo de transmisión, se emite un pulso de energía, a la vez de iniciar un contador interno y cambiar el modo de la antena a receptor. Una vez se registra el eco de la señal emitida, se evalúa la distancia del objeto mediante la Ecuación 2.10, similar a la Ecuación 2.8:

  $$d = \frac{c_m \cdot t}{2} \tag{2.10}$$

  Donde c_m representa la velocidad de la luz en el medio, y t el tiempo registrado en el contador interno (es decir, cuanto se demoró la onda de radio en ir al objetivo y posteriormente retornar).

– **Radar de onda continua**: Estos radares, denominados como CW-RADAR (*Continuous Wave RADAR)*, emiten de manera continua una onda con alta frecuencia. El tipo de CW-RADAR más utilizado es el radar de onda continua modulado en frecuencia (Frequency-Modulated Continuous Wave RADAR, o FMCW RADAR), que, como su nombre lo indica, **modula la frecuencia de la onda emitida**, permitiendo calcular la distancia de los objetos de acuerdo a la **diferencia de frecuencia** entre la onda emitida y la reflejada.

Se presenta a continuación la Figura 2.2, que ilustra los elementos claves para poder calcular la distancia de un objeto detectado por un radar [16]:

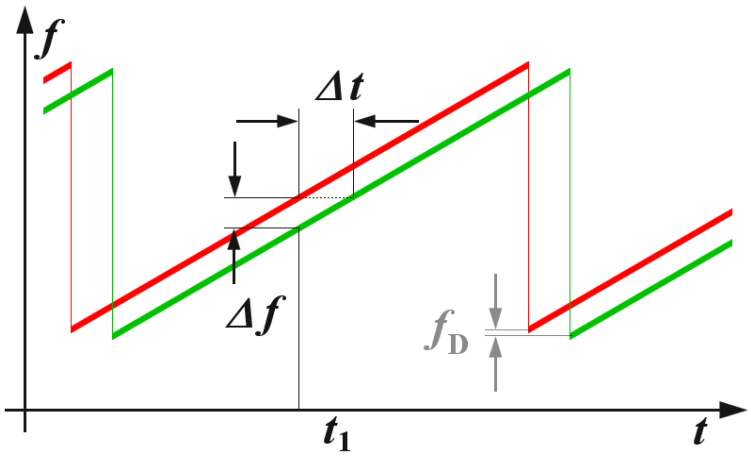

Figura 2.2: Imagen extraída de [16], donde se presenta un esquema de la diferencia entre señal transmitida y reflejada para efectos del cálculo de distancia, presentándose estas en color rojo y verde respectivamente. Se aprecia que existe un desfase entre dichas señales, representada por Δt; a su vez, este desfase genera una diferencia en frecuencia en un determinado tiempo: Δf. Finalmente, si el objeto detectado posee una velocidad racial respecto al radar, es necesario incluir un *offset* en la frecuencia dado por el **efecto Doppler**, que se representa en la figura como f_D.

Así, se hace uso de la Ecuación 2.11, que calcula la distancia del objeto en base a la información de la onda emitida y reflejada [15] [16]:

$$d = \frac{c_m|\Delta t|}{2} = \frac{c_m|\Delta f|}{2 \cdot \left(\frac{df}{dt}\right)} \tag{2.11}$$

Donde c_m corresponde a la velocidad de la luz en el medio, Δt representa el tiempo de propagación, Δf se asocia a la diferencia de frecuencia registrada, y finalmente $\frac{df}{dt}$ refleja la variación de frecuencia por unidad de tiempo, tal como se indica en la Figura 2.2.

- **Sound navigation and Ranging Sensors (SONAR)**: SONAR consiste en el uso de **ondas sonoras** para la detección de objetos en el entorno mediante el método de *Time-of-Flight* [2]:

  – Inicialmente el SONAR se comporta como transmisor, enviando una onda sonora

en un arco; al pasar el umbral de mínima distancia, cambia su modo a receptor, registrando posteriormente el tiempo en que las oscilaciones debido a la colisión por el contacto con los objetos retornan al SONAR.

- Este funcionamiento es similar al observado en LiDAR y RADAR, con la diferencia que, en vez de ocupar ondas electromagnéticas, se utiliza la velocidad del sonido en el medio de interés; así, se presenta la Ecuación 2.12, que guarda similitud a la Ecuación 2.8:

$$d = \frac{v \cdot t}{2} \tag{2.12}$$

  Donde v es la velocidad del sonido en el medio donde se propaga este, y t el tiempo registrado.

- Si bien la precisión de las mediciones realizadas es peor que los sistemas LiDAR en medios terrestres, SONAR presenta un desempeño óptimo en entornos acuáticos, por lo que es típicamente ocupado en vehículos autónomos que requieran operar bajo el agua.

- **Visión Computacional**: Finalmente, es posible utilizar cámaras con el fin de captar la información del entorno. Mediante la información visual proporcionada, es posible obtener características que permiten definir elementos de interés (conocidos como *landmarks*), permitiendo así la estimación del lugar [2].

  El uso de visión computacional ha tenido un alza importante en el último tiempo; si bien se proponía su uso desde hace muchos años atrás, el alto costo computacional requerido y la necesidad de una iluminación adecuada limitaban drásticamente su uso. A lo largo de la última década, debido al mejor rendimiento de equipos computacionales, implementación de algoritmos con mayor capacidad de optimización de procesamiento de datos, y el uso de cámaras estéreo y RGB-D, su aplicación se ha visto aumentada.

  Un punto importante a destacar es el uso de cámaras RGB-D en comparación con cámaras estéreo; si bien ambas ocupan dos lentes para captar información, las cámaras RGB-D poseen además un **sensor de profundidad**, lo que complementa el alcance del sistema de visión computacional. De esta manera, se reduce drásticamente la imprecisión al momento de detectar ciertos objetos de interés dentro del entorno.

2.2. Extracción de características visuales

Para procesar el entorno de manera visual, una forma de examinar la información entregada es extrayendo características notorias de los elementos que lo componen; es decir, ciertos descriptores presentes dentro de la imagen que permitan identificar tanto elementos presentes en la misma, como diferenciar entre dichos elementos. Así, es posible procesar el entorno en base a puntos de interés comunes en imágenes en una determinada secuencia de tiempo.

Los puntos de interés se agrupan en dos categorías [17]:

- **Bordes**: Son aquellas características que se fundamentan en la detección de contornos y la diferenciación tanto entre objetos adyacentes como entre los elementos y el fondo. Estas se relacionan tanto con la orientación de los objetos como con lo que los rodea.

- **Puntos de interés**(*keypoints*): Son aquellas características que, como su nombre indica, identifican sectores clave de objetos, como por ejemplo, esquinas de una ventana, bordes puntiagudos en las letras de señales de tránsito, entre otros. Mediante estas es posible asociar **mediciones** a objetos presentes en el entorno.

2.2.1. Descriptores ORB

ORB (Oriented FAST[7] and Rotated Brief) es un algoritmo que mezcla descriptores BRIEF (Binary Robust Independent Elementary Features) [18] con el detector de esquinas FAST [19], optimizando y mejorando a la vez el rendimiento de ambos [20].

Para obtener dichos puntos de interés (también conocidos como *keypoints*), se itera de la siguiente manera [20]:

1. En primer lugar se realiza una detección de *keypoints* utilizando FAST con un radio de 9 píxeles.

2. Con el objetivo de obtener un número N de *keypoints* deseados, se establece un umbral de tal manera que se capturan >N *keypoints*, y se ordena estos de acuerdo a la métrica de esquinas de Harris [21]. Luego, se seleccionan los N *keypoints* con mayor puntaje.

3. El procedimiento anterior se utiliza en una pirámide de escala de la imagen objetivo, calculando *keypoints* en cada nivel; así, se obtienen características "parcialmente" invariantes en escala.

Posteriormente, se obtiene la orientación del *keypoint* mediante la **intensidad del centroide** [22]:

1. Se calcula el *momento* m_{pq} del área del *keypoint* mediante la siguiente fórmula:

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y) \tag{2.13}$$

   con $I(x,y)$ la intensidad del píxel ubicado en (x,y).

2. Con dicha fórmula, se obtiene el *centroide* del *keypoint*:

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}\right) \tag{2.14}$$

3. Finalmente, es posible trazar una recta desde el centro del área del *keypoint* hacia el centroide, obteniendo así el siguiente ángulo [8]:

$$\theta = atan2(m_{01}, m_{10}) \tag{2.15}$$

Por otra parte, los descriptores BRIEF consisten en un vector de características formulado a partir de n test binarios, donde estos últimos corresponden a una función binaria cuyos valores dependen de la intensidad de los píxeles en el área de interés. Estos descriptores

[7] FAST: Features from Accelerated Segment Test
[8] atan2() corresponde a la arco tangente de dos parámetros.

presentan un bajo desempeño frente a rotaciones, por lo que se procede a utilizar en ORB una versión modificada de estos, rBRIEF [23]:

1. Se calcula el ángulo de los *keypoints* formulado previamente.
2. Para cada conjunto de n test binarios en (xi, yi), se define una matriz de 2 x n, S, que contiene las coordenadas de dichos píxeles.
3. Utilizando con ello el ángulo del *keypoint* theta, se encuentra su matriz de rotacion y se rota S, obteniendo Stheta.
4. Se discretiza el ángulo en incrementos de 2 pi / 30 (12 grados), y se construye una LUT de patrones BRIEF precalculados. De esta manera, siempre y cuando la orientación del *keypoint* sea consistente, el mismo conjunto de puntos Stheta será ocupado para calcular su descriptor.

Sin embargo, dicha orientación resulta en la pérdida de información del vector de características, principalmente de su varianza y una media cercana a 0.5; para solventar lo anterior, ORB utiliza un algoritmo avaro (*greedy search*) que itera sobre todos los posibles tests binarios, buscando aquellos que posean de manera simultánea una alta varianza y una media cercana a 0.5, además de no estar correlacionados. Así, se obtiene el vector de características rBRIEF.

Las ventajas de ORB respecto a BRIEF y FAST se resumen en los siguientes puntos [20]:

- Se calcula de forma más eficiente los descriptores BRIEF.
- Se incorpora un componente de orientación a FAST sin comprometer en exceso el costo computacional.
- Analiza la varianza y correlación entre descriptores BRIEF.
- Incorpora métodos que facilitan la distinción de características BRIEF no correlacionadas si se presenta una invarianza rotacional.

2.3. Random Finite Sets

De manera simplificada, es posible definir las mediciones realizadas por sensores como un conjunto aleatorio de elementos, teniendo cada uno de ellos cierta incertidumbre respecto a su ubicación real. Esto implica que, al momento de realizar mediciones, es necesario poder contar con un sistema que permita tanto incluir o eliminar elementos (debido a la presencia de falsas alarmas u omisión de la detección de objetos), como variar el orden de los mismos.

Si bien los *landmarks* se presentan en un orden concreto, esto se debe únicamente a como el sensor los ha reconocido; es posible que, tomando una ruta distinta, el sensor reconozca los mismos objetos pero en un orden completamente distinto. Se debe notar además que, independientemente del sentido en que se recorra el mapa, **los puntos de interés deberían ser los mismos** [6].

Dado lo planteado previamente, se observa que, al momento de estimar un mapa mediante un vector (o matriz) de características, era necesario contar con todas las posibles permutaciones de los *landmarks* dado su orden de visualización. Para abordar ello, surge de manera natural interpretar al conjunto de características que conforman un mapa como **un conjunto aleatorio**. Como ejemplo, sea un listado de detecciones realizadas en un plano 2D de la forma $z^1 = [r^1\theta^1]^T, z^2 = [r^2\theta^2]^T, ..., z^\xi = [r^\xi\theta^\xi]^T$, donde r y θ corresponden a la distancia y el ángulo de la detección relativa al sensor respectivamente. Representando dichas observaciones mediante un vector se tiene $Z = [z^1, z^2, ..., z^\xi]^T$, lo que vuelve delicada su operación al momento de implementar el algoritmo ya que no se puede variar su dimensionalidad de manera dinámica. En cambio, un conjunto definido como $Z = \{z^1, z^2, ..., z^\xi\}$ permite, por definición, añadir o remover elementos de manera dinámica [6]. Además, debido a que un conjunto no posee un orden determinado, no se presenta el problema mencionado en el párrafo anterior.

Dado las características mencionadas anteriormente, y la naturaleza aleatoria de las mediciones, esta forma de definir el mapa se denomina **conjunto finito aleatorio (*Random Finite Sets*, o *RFS*)** [6] [9].

2.3.1. Representación matemática de un mapa mediante RFS

Sea $\mathcal{M}_t$ la representación mediante RFS del mapa a estimar en un tiempo t. Este se define de acuerdo a las características en el mapa en dicho tiempo t: $M_t = \{m_t^1, m_t^2, ..., m_t^\xi\}$, donde m_t^i es una determinada característica presente en el mapa. Se obtiene así la representación del mapa en la iteración t de acuerdo a lo siguiente [24]:

$$\underbrace{\mathcal{M}_t}_{\text{Mapa est. actual}} = \underbrace{\mathcal{M}_{t-1}}_{\text{Mapa est. anterior}} \cup \left(\underbrace{FoV(X_t)}_{\text{Info. prov. del sensor}} \cap \bar{\mathcal{M}}_{t-1} \right), \tag{2.16}$$

siendo $\bar{\mathcal{M}}_{k-1}$ el conjunto de características (*features*) que **no** se encuentra presente en $\mathcal{M}_{t-1}$. Lo anterior se interpreta de manera intuitiva que, a medida que el sensor va entregando más información al mapa, este aumenta su conjunto de *features*.

2.3.2. Representación de las mediciones mediante RFS

Con el fin de integrar tanto el ruido de medición como la incertidumbre de detección en el algoritmo SLAM, se plantean las detecciones realizadas por un vehículo con ubicación X_t en un tiempo t como un RFS, denominando a este conjunto Z_t. Este se compone tanto de las características que se espera que estén presentes a la vez de las falsas detecciones realizadas [6] [24]:

$$Z_t = \bigcup_{m \in \mathcal{M}_t} \mathcal{D}_t(m, X_t) \cup \mathcal{C}_t(X_t), \tag{2.17}$$

donde $\mathcal{D}_t(m, X_t)$ es el RFS de las mediciones dado un *landmark* ubicado en m, y $\mathcal{C}_t(X_t)$ es el RFS de las falsos *landmarks* detectados (falsas alarmas), estando el vehículo en la posición

[9] Dado la facilidad de nombrar este tipo de conjuntos de acuerdo a la sigla RFS, se utilizará esta a lo largo del documento en vez de su nombre completo.

X_t [6]. Así, es posible observar que $Z_t = \{z_t^1, ..., z_t^\xi\}$ consiste en una cantidad aleatoria de observaciones z_t^i en cada iteración t.

Por otra parte, mediante el uso de estadísticas para conjuntos finitos [25], la verosimilitud de las mediciones Z_t dado la trayectoria $X_{0:t}$ y el mapa $\mathcal{M}_t$ se define de la siguiente manera [6]:

$$p(Z_t|X_t, \mathcal{M}_t) = \sum_{\mathcal{W} \subseteq Z_t} f_D(\mathcal{W}|\mathcal{M}_t, X_t) f_C(Z_t - \mathcal{W}|X_t), \tag{2.18}$$

donde $f_D(\mathcal{W}|\mathcal{M}_t, X_t)$ corresponde a la densidad de probabilidad del RFS de observaciones, y $f_C(Z_t - \mathcal{W}|X_t)$ correspondiente a la densidad de probabilidad de las detecciones erróneas (C_t). Así, se aborda la incertidumbre respecto a las mediciones mencionada anteriormente: el primer término encapsula la incertidumbre de la precisión de las mediciones y su ruido asociado, mientras que el segundo engloba la incertidumbre de las detecciones erróneas.

2.4. RFS-SLAM

Utilizando las observaciones y el mapa delimitados en la sección anterior, es posible redefinir la Ecuación 2.1 mediante Random Finite Sets, describiendo de esta manera el estado del robot en SLAM mediante RFS de acuerdo a la siguiente distribución de probabilidad [24]:

$$p(X_{0:t}, \mathcal{M}_t|Z_{1:t}, U_{1:t}), \tag{2.19}$$

donde $Z_{1:t}$ es el RFS correspondiente a las observaciones realizadas desde $t = 1$ hasta t, $U_{1:t}$ el conjunto de los movimientos realizados por el robot desde $t = 1$ hasta t, y $X_{0:t}$ la trayectoria del robot desde el estado inicial hasta su ubicación en tiempo t.

Mediante la aplicación de un filtro Bayesiano, es posible separar la Ecuación 2.19 en dos etapas: **Predicción** y **Corrección**, similar a lo observado en las Ecuaciones 2.6 y 2.7 respectivamente [24]:

$$\begin{aligned} p(X_{0:t}, \mathcal{M}_t|Z_{1:t}, U_{1:t}) &= g(X_t|X_{t-1}, U_t) \int f_{\mathcal{M}}(\mathcal{M}_t|X_t, \mathcal{M}_{t-1}) p(X_{t-1}, \mathcal{M}_{t-1}) \delta\mathcal{M}_{t-1} \\ &= \frac{p(Z_t|X_t, \mathcal{M}_{t-1})\ p(X_{0:t}, \mathcal{M}_t|Z_{1:t-1}, U_{1:t})}{p(Z_t|X_{0:t}, Z_{t-1})} \end{aligned} \tag{2.20}$$

donde $\int f_{\mathcal{M}}(\mathcal{M}_t|X_t, \mathcal{M}_{t-1})$ corresponde a la densidad dinámica de transición de mapa [10], $p(Z_t|X_{0:t}, Z_{t-1})$ es la verosimilitud del conjunto de mediciones, y $g(X_t|X_{t-1}, U_t)$ corresponde a la densidad de transición que estima la posición del robot en t dada la posición en $t-1$ y los comandos de movimiento dados en t.

[10] Un punto importante de esta integral es que es una **integral de conjuntos**, por lo que sus propiedades varían respecto a una integral convencional.

2.4.1. Factorización de RFS-SLAM

De acuerdo a lo presentado en FastSLAM [12], es posible también factorizar la Ecuación 2.19 en dos probabilidades condicionales [24]:

$$p(X_{0:t}, \mathcal{M}_t | Z_{1:t}, U_{1:t}) = p(X_{0:t} | Z_{1:t}, U_{1:t})\ p(X_{0:t}, \mathcal{M}_t | X_{0:t}, Z_{1:t}) \tag{2.21}$$

Donde:

- $p(X_{0:t}|Z_{1:t}, U_{1:t})$ corresponde a la **función de densidad de probabilidad de la trayectoria del vehículo**, de acuerdo a las mediciones realizadas y a los comandos de movimiento entregados. Esta estimación a su vez se puede definir de acuerdo a las mediciones realizadas con anterioridad:

  $$p(X_{0:t}|Z_{1:t}, U_{1:t}) = \eta\ p(Z_t|X_{0:t}, Z_{1:t-1})\ p(X_{0:t}|Z_{1:t-1}, U_{1:t}) \tag{2.22}$$

  Donde η es una constante de normalización, $p(Z_t|X_{0:t}, Z_{1:t-1})$ refleja la relación entre las mediciones actuales dado las mediciones realizadas anteriormente y la trayectoria del robot hasta el tiempo t, y $p(X_{0:t}|Z_{1:t-1}, U_{1:t})$ predice la trayectoria hasta dicho tiempo t dada las observaciones previas y los comandos de control, mediante la siguiente ecuación:

  $$p(X_{0:t}|Z_{1:t-1}, U_{1:t}) = p(X_t|X_{t-1}, U_{1:t})\ p(X_{0:t-1}|Z_{1:t-1}, U_{1:t-1}) \tag{2.23}$$

  Siendo $p(X_t|X_{t-1}, U_{1:t})$ la densidad de transición de posición del robot.

- $p(\mathcal{M}_t|X_{0:t}, Z_{1:t})$ consiste en **la función de densidad de probabilidad de los *landmarks*** dada la trayectoria del robot. Es posible definir dicha densidad de probabilidad mediante la siguiente Ecuación:

  $$p(\mathcal{M}_t|X_{0:t}, Z_{1:t}) = \frac{p(Z_t|X_{0:t}, \mathcal{M}_t)\ p(\mathcal{M}_t|X_{0:t}, Z_{1:t-1})}{p(Z_t|X_{0:t}, Z_{1:t-1})} \tag{2.24}$$

  Donde $p(Z_t|X_{0:t}, \mathcal{M}_t)$ corresponde al modelo de medición, y $p(Z_t|X_{0:t}, Z_{1:t-1})$ indica la relación entre las mediciones actuales dado las mediciones realizadas anteriormente y la trayectoria del robot hasta el tiempo t.

De acuerdo a [6] y [12], es posible estimar el término $p(X_{0:t}|Z_{1:t}, U_{1:t})$ utilizando un **filtro de partículas**; esto conlleva que la densidad $p(Z_t|X_{0:t}, Z_{1:t-1})$ permita ponderar los pesos de cada partícula. Según [6] y [12], para calcular dicho término es necesario poder **asociar la información** (en inglés *Data Association*, o *DA*), entre las mediciones Z_t y su predicción dado $X_{0:t}$ y $Z_{0:t-1}$; no obstante, debido a que se opera con conjuntos, no es posible establecer DA entre ellos, por lo que no es posible calcular computacionalmente $p(\mathcal{M}_t|X_{0:t}, Z_{1:t})$.

Una solución posible es realizar una aproximación mediante un **Filtro PHD**, que se detallará en la siguiente sección.

2.5. Filtro PHD

2.5.1. Definición matemática

Es posible aproximar ciertas densidades de probabilidades de RFS utilizando una propiedad importante de los mismos: su **momento de primer orden**. Para un RFS $\mathcal{M}_t$, se define su momento de primer orden, o *probability hyphotesis density* (**PHD**), como una función no negativa v_t tal que, para cada región S dentro del espacio de características [26]:

$$\int_S v_t(m)dm = \mathbb{E}[\mathcal{M}_t \cap S] \tag{2.25}$$

Si se asume el contexto de SLAM y $\mathcal{M}_t$ se define como un mapa de *landmarks*, se interpreta que la masa de la PHD entrega **la cantidad esperada** de *landmarks* $\hat{m}_t$ contenidas en el mapa M_t. Si ahora se supone que el RFS $\mathcal{M}_t$ sigue una distribución Poisson, es posible obtener directamente la densidad de probabilidad de $\mathcal{M}_t$ [26]:

$$p(\mathcal{M}_t) = \frac{\prod\limits_{m \in \mathcal{M}_t} v_t(m)}{exp(\int v_t(m)dm)} \tag{2.26}$$

2.5.2. Aproximación de verosimilitud de mediciones

Para poder abordar el cálculo de $p(Z_t|X_{0:t}, Z_{1:t-1})$, se aplica el Teorema de Bayes [24]:

$$p(Z_t|X_{0:t}, Z_{1:t-1}) = p(Z_t|X_{0:t}, \mathcal{M}_t)\frac{p(\mathcal{M}_t|X_{0:t}, Z_{1:t-1})}{p(\mathcal{M}_t|X_{0:t}, Z_{1:t})} \tag{2.27}$$

Esto entrega dos distribuciones que calculan la densidad de probabilidad de $\mathcal{M}_t$ de acuerdo a la trayectoria y las mediciones. Si se asume una distribución de Poisson, es posible ocupar la Ecuación 2.26 para aproximar dichas densidades [24]:

$$p(\mathcal{M}_t|X_{0:t}, Z_{1:t-1}) \approx \frac{\prod\limits_{m \in \mathcal{M}_t} v_{t|t-1}(m|X_{0:t})}{exp(\int v_t(m|X_{0:t})dm)} \tag{2.28}$$

$$p(\mathcal{M}_t|X_{0:t}, Z_{1:t}) \approx \frac{\prod\limits_{m \in \mathcal{M}_t} v_t(m|X_{0:t})}{exp(\int v_t(m|X_{0:t})dm)} \tag{2.29}$$

Lo anterior toma como supuesto que los *landmarks* son independientes e idénticamente distribuidos (i.i.d), y además, el número de *landmarks* se determina mediante una distribución de Poisson. Las aproximaciones realizadas permiten observar que **es posible utilizar este filtro PHD en la resolución del problema SLAM con RFS** [6]. Así, [6] y [26] sugieren utilizar una **estrategia de multi-características** establecida en [27] para el cálculo de las densidades de probabilidad [24].

2.6. Filtro GM-PHD con RFS SLAM

Para aproximar el mapa mediante un Filtro PHD, se define en primer lugar una suma ponderada de gaussianas como la función PHD a utilizar, y se aproxima la función de recursión de mapa mediante un filtro PHD definido por una mezcla de gaussianas (*Gaussian Mixture*, o *GM*), lo que se conoce como filtro GM-PHD; a su vez, la trayectoria y su recursión se aborda mediante un **filtro de partículas** [26]. Esto se conoce en su conjunto como **RB-PHD SLAM**.

El filtro GM-PHD posee tres etapas, que se describirán en las siguientes secciones.

2.6.1. Predicción

La sección del filtro GM-PHD encargada de la **predicción** para un tiempo t se define mediante la Ecuación 2.30 [26]:

$$v_{t|t-1}(m|X_{0:t}) = v_{t-1}(m|X_{0:t}) + b(m|X_t) \tag{2.30}$$

Donde $v_{t-1}(m|X_{0:t})$ es la GM-PHD del tiempo previo (es decir, para $t-1$), y $b(m|X_t)$ corresponde a la GM-PHD de la creación de nuevas Gaussianas que modelan nuevos *landmarks* (en otras palabras, permite representar *landmarks* que entran en el campo de visión de los sensores y no han sido observados previamente).

La Figura 2.3 [11] ilustra los pasos encargados de la predicción en el Filtro PHD [26]:

1. Se crean nuevas Gaussianas ubicadas en las posiciones de las mediciones en el tiempo $t-1$ mediante la aplicación de la función inversa del **modelo de medición del entorno** ($(h^{espacial})^{-1}$), y se les asigna un peso constante a cada una de ellas.
2. Cada Gaussiana del mapa anterior es propagada, por lo que, asumiendo un mapa estático, estas Gaussianas propagadas no sufren cambios.
3. Finalmente, las Gaussianas resultantes de los pasos 1 y 2 se suman, generando la predicción del mapa.

[11] Esta y las Figuras a continuación presentan una notación ligeramente diferente a la existente en este informe. Para una mayor equivalencia, se debe notar que $t = k$.

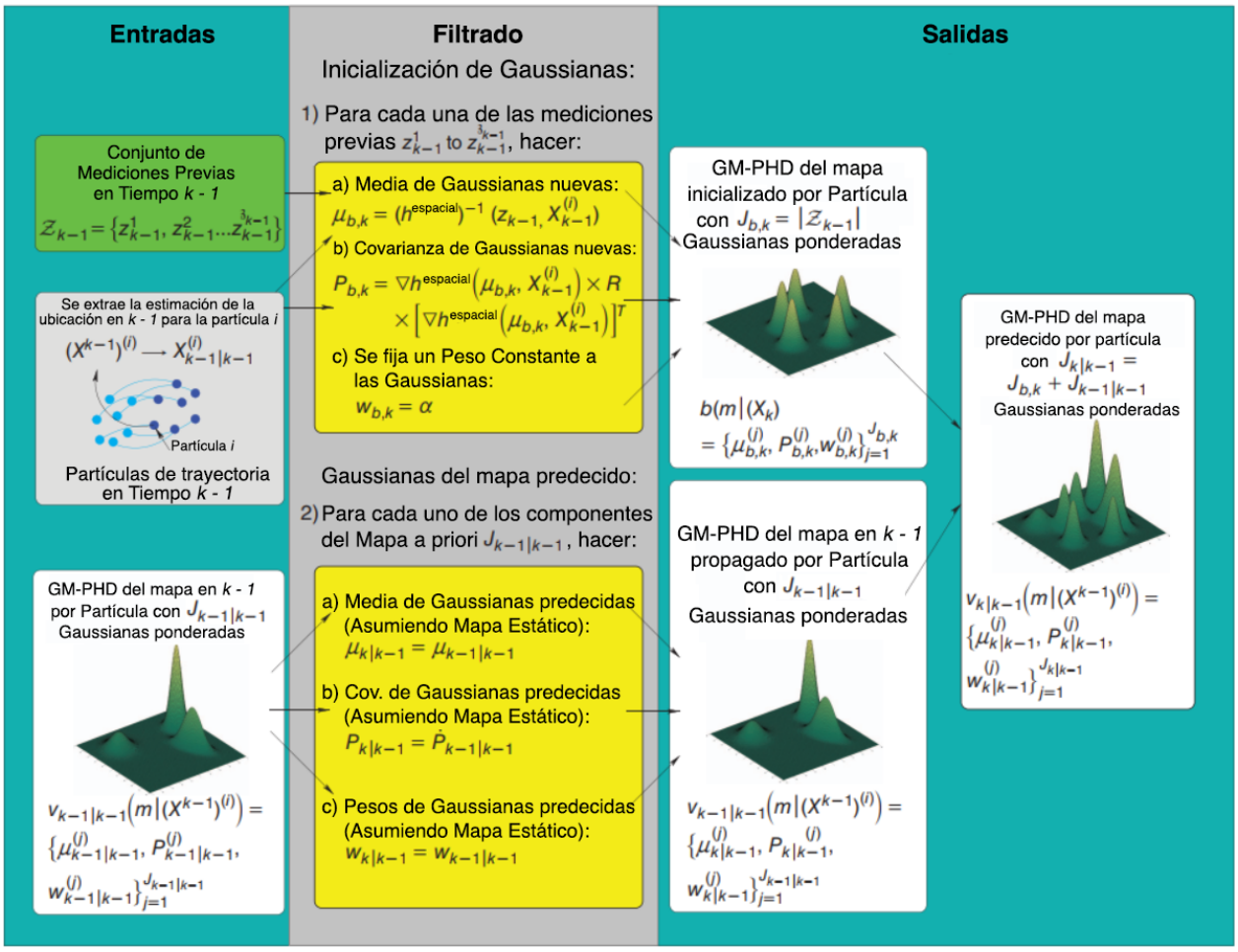

Figura 2.3: Imagen extraída y traducida de [26], que resume la sección del filtro PHD encargada de la predicción. Dado el conjunto de observaciones Z_{t-1}, y la trayectoria estimada en el tiempo $t-1$ de las N partículas, se genera una mezcla de gaussianas con peso constante. Estas se operan con las gaussianas ponderadas de la iteración anterior, originando una nueva serie de gaussianas ponderadas que corresponden a la **predicción del mapa de acuerdo al filtro de partículas**. Estas últimas se propagan hacia la sección de corrección con el fin de filtrar la información y así disminuir la incertidumbre acumulada.

2.6.2. Corrección

Por otra parte, la Ecuación 2.33 se encarga de la **corrección** del Filtro GM-PHD [26]:

$$v_t(m|X_{0:t}) = v_{t|t-1}(m|X_{0:t})(1 - P_D(m|X_t)) \tag{2.31}$$

$$+ v_{t|t-1}(m|X_{0:t}) \sum_{z \in Z_t} \frac{P_D(m|X_t)h(z|m, X_t)}{c_t(z|X_t) + \int P_D(\xi|X_t)\ h(z|\xi, X_t)\ v_{t|t-1}(\xi|X_{0:t})d\xi} \tag{2.32}$$

$$v_t(m|X_{0:t}) = v_{t|t-1}(m|X_{0:t}) \Bigg[1 - P_D(m|X_t) + \sum_{z \in Z_t} \frac{P_D(m|X_t)h(z|m, X_t)}{c_t(z|X_t) + \int P_D(\xi|X_t)\ h(z|\xi, X_t)\ v_{t|t-1}(\xi|X_{0:t})d\xi} \Bigg] \tag{2.33}$$

Donde $P_D(m|X_t)$ corresponde a la probabilidad de detectar el *landmark* m dado que la posición del robot en t es X_t, $c_t(z|X_t)$ es la PHD del RFS correspondiente a las falsas alarmas C_t (presente en la definición de Z_t de la Ecuación 2.17), y $h(z|m, X_t)$ el modelo de medición del entorno del sensor.

Se observa que la Ecuación 2.33 se subdivide en dos:

- El término de la Ecuación 2.31 se asocia a aquellas Gaussianas ponderadas de acuerdo a la probabilidad de *no-detección* [12].
- El término de la Ecuación 2.32 corresponde a aquellas Gaussianas actualizadas por la información proveniente de las observaciones realizadas en el tiempo t, en conjunto a sus probabilidades de detección.

La Figura 2.4 resume los pasos requeridos para la corrección del Filtro PHD [26]:

1. Se realiza una predicción de la ubicación del robot dado la posición anterior, los comandos de movimiento previos, y el ruido asociado al movimiento.
2. Se actualizan los parámetros de las Gaussianas de no-detección (media y covarianza), copiando las Gaussianas de la predicción y ponderándolas por la probabilidad de no existir detección.
3. Mediante la aplicación del Filtro de Kalman Extendido (EKF), la media y covarianza de las Gaussianas del mapa predecido se actualizan utilizando la información proveniente de cada una de las mediciones en t.
4. Se actualizan los pesos de las Gaussianas de detecciones[13] de acuerdo a su probabilidad de detección, su proximidad dada una cierta Distancia de Mahalanobis, y la PHD de falsas alarmas c_t.
5. Se mezclan los parámetros de las Gaussianas de no-detección y las Gaussianas de detección, resultando la GM-PHD $v_t^i(m|X_{0:t})$ para cada partícula i.
6. Finalmente, se selecciona aquellas Gaussianas que posean los mayores pesos, y se purga el resto de Gaussianas.

[12] No-detecciones: caso donde se debería haber detectado un objeto, pero no se presenta detección; es decir, un falso negativo.

[13] Detecciones: Caso de verdadero positivo, donde se detecta acertadamente un objeto.

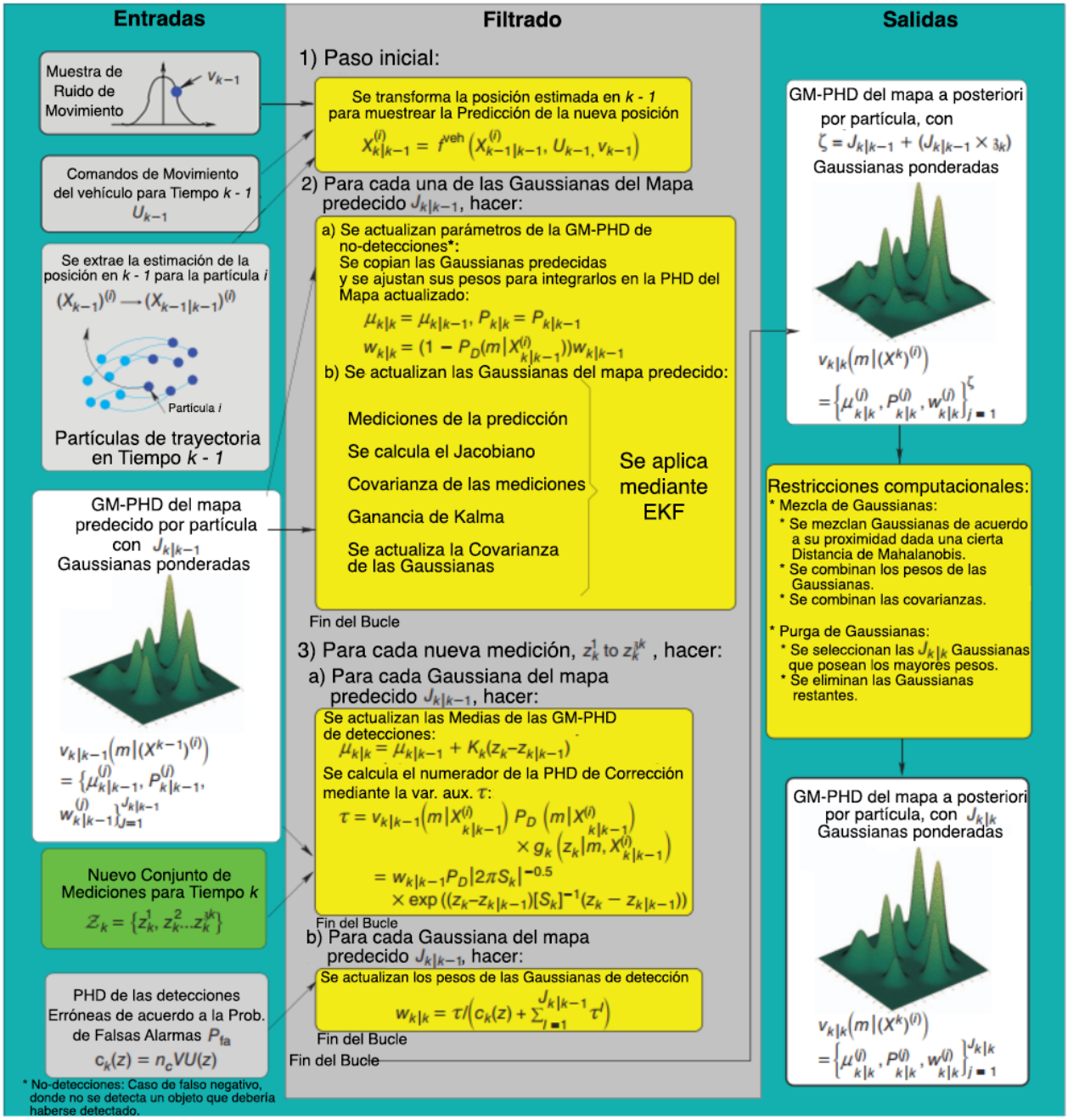

Figura 2.4: Imagen extraída y traducida de [26], que resume la sección del filtro PHD encargada de la corrección. Aquí, dada la información propagada en la sección de predicción, en conjunto con la trayectoria de las partículas, se filtran aquellas gaussianas que presentan poca correspondencia con la información de la trayectoria de las partículas, a la vez que se le da mayor relevancia a aquellas gaussianas con mayor verosimilitud respecto a la trayectoria de las partículas. Así, se "limpia" la mezcla de gaussianas, y se propaga a la predicción de la siguiente iteración.

2.6.3. Actualización de las partículas de trayectoria

El filtro PHD-SLAM estima la trayectoria del robot mediante un **conjunto de partículas**, que representan las posibles posiciones del robot para cada instante de tiempo. Los

pasos a seguir para la actualización de la trayectoria de dichas partículas se presentan a continuación, quedando además resumidos en la Figura 2.5 [26]:

1. Se realiza una predicción de la ubicación del robot dado la posición anterior, los comandos de movimiento previos y el ruido asociado al movimiento.
2. Se actualizan los pesos de las partículas de acuerdo a la verosimilitud del conjunto de las mediciones.
3. Se normalizan los pesos.
4. Finalmente, se realiza un remuestreo todas las partículas de acuerdo a sus respectivos pesos.

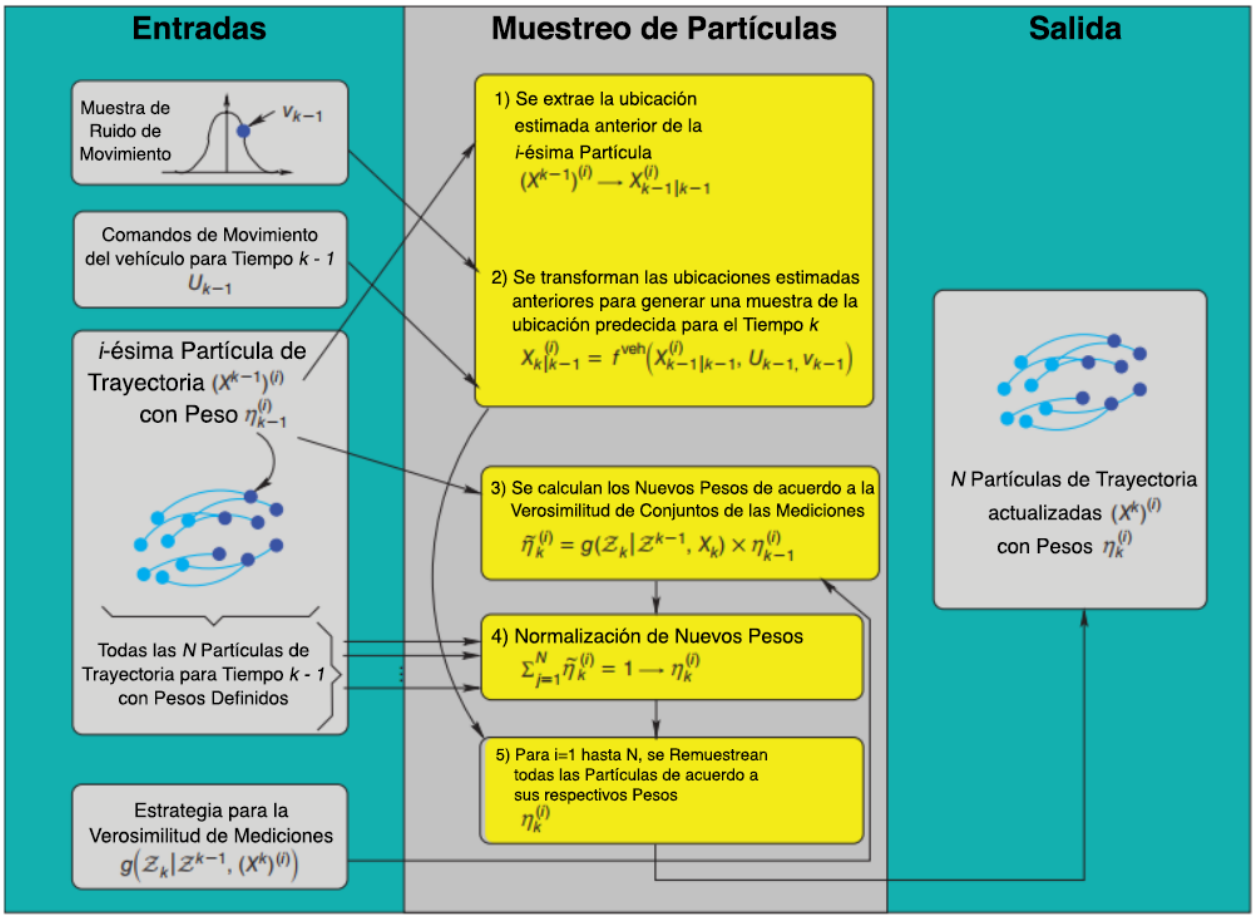

Figura 2.5: Imagen extraída y traducida de [26], que resume la sección encargada de la actualización de la trayectoria de las partículas. Dado los movimientos del vehículo U_{t-1} y la trayectoria de las N partículas, se utiliza la ubicación previa de las partículas y la verosimilitud de las observaciones para ponderar su importancia al momento de describir la probabilidad del sistema.

Capítulo 3

Metodología

La implementación del proyecto se realizó a cabo en un computador con el sistema operativo Ubuntu, debido a la alta compatibilidad de las librerías a utilizar con dicho sistema operativo.

La calibración de cámara y el proceso hasta la obtención de coordenadas 3D a partir de los elementos de interés se llevó a cabo mediante el lenguaje de programación Python. Por otra parte, se utilizó una librería de código abierto RFS-SLAM [28], que contiene el algoritmo de estudio a utilizar. Esta librería trabaja mediante datos simulados: genera un conjunto de *landmarks* en posiciones aleatorias, realizando mediciones de los mismos mediante un movimiento aleatorio. Debido a ello, se requirió modificar dicha librería a lo largo del proyecto para poder adecuarla a las especificaciones de este.

La metodología contempla la secuencia de pasos a continuación:

1. **Base de datos a utilizar**: Para el experimento, se utilizó una base de datos elaborada para la actividad en vez de captura de imágenes en tiempo real, debido a que se cuenta con la captura de mediciones de manera desacoplada del código RFS-SLAM[28]. Para ello, se capturó una secuencia de imágenes con las siguientes características:

    - Se capturó un total de **195 imágenes**.
    - Las imágenes se grabaron en una sala cerrada de 4 metros de ancho y 8 de largo, a lo largo de una mesa rectangular de 1.5 metros de ancho y 5 metros de largo. Las paredes de dicha sala son de vidrio, por lo que el entorno cuenta con una alta reflectividad.

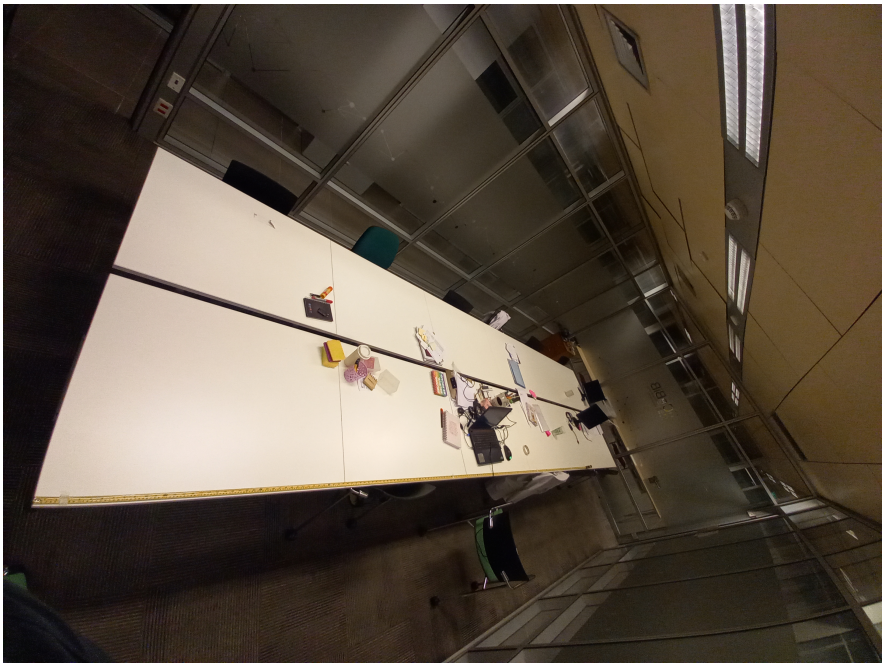

Figura 3.1: Vista elevada del setup. La trayectoria a seguir rodea la mesa, siendo su punto de partida la silla que se aprecia con el respaldo apuntando hacia la cámara.

- El circuito resultante resultó con dimensiones máximas de 2 metros de ancho y 6 de largo.
- La cámara se ubicó en el asiento de una silla con ruedas, con el fin de poder desplazar la cámara a lo largo de la trayectoria; por lo tanto, no se cuenta con un sistema de medición de movimiento integrado.

Figura 3.2: Setup de la silla.

- Los ejes de coordenadas se fijaron respecto a la posición inicial de la cámara, siendo X el eje horizontal desde el centro del plano de la cámara hacia su izquierda, Y el eje vertical (con valores positivos apuntando hacia el extremo superior del plano de la cámara), y Z el eje longitudinal, adquiriendo valores positivos a medida que se aleja uno de la cámara.
- La posición $(x, y, z) = (0, 0, 0)$ se establece en la posición inicial de la cámara, ubicada esta a un costado de la mesa, de manera paralela a uno de sus bordes largos.

Figura 3.3: Imagen de la cámara. Se aprecia su distancia en el punto inicial respecto a una de las esquinas de la mesa (270 cm).

- El recorrido ideal se diseñó en el suelo a lo largo de la mesa, estableciendo este como el **Ground Truth**.
- Se delimitó que, para las secciones rectas de la trayectoria, las capturas tomadas tengan una separación de 10 centímetros a lo largo del eje longitudinal.
- Con el fin de reducir al mínimo la variación en el eje horizontal, se dispuso de una guía a lo largo de la trayectoria a seguir, permitiendo que la variación en el eje horizontal a lo largo de los trayectos rectos sea despreciable.
- Por último, el experimento base se llevará a cabo con **600** partículas en RFS-SLAM; se realizarán adicionalmente experimentos con un menor número de partículas (20 y 200) para comparar su desempeño con el caso de estudio.

2. **Calibración de cámara**: Para poder operar los algoritmos es necesario contar con los parámetros **intrínsecos** y **extrínsecos** de una cámara RGB-D. Con estos es posible tanto revertir la distorsión provocada por cada uno de los lentes (debido a la curvatura de estos), como también de poder asociar píxeles de elementos comunes entre los lentes (es decir, establecer elementos en común visibles en ambos lentes). Los pasos para calibrar la cámara son los siguientes:

   - De manera preliminar, se debe contar con una imagen de **tablero de ajedrez**. Se debe saber de cuantos cuadrados es tanto de largo como de ancho, a la vez del largo en milímetros de los cuadrados.
   - Se deben tomar una serie de imágenes del tablero de ajedrez tanto con la cámara RGB como con la cámara de profundidad en modo infrarrojo de manera simultánea. Dichas imágenes deben ser con **el tablero en diferentes posiciones e inclinaciones**, con el fin de permitir mayor robustez de los parámetros calculados. Se utilizó

para este trabajo un total de 50 imágenes [14].

- Con las imágenes de calibración, y utilizando la información característica del tablero de ajedrez (sus dimensiones, tamaño de los cuadrados, y cantidad de cuadrados por largo y alto), se obtienen los **puntos de imagen** y **puntos de objeto**, los que permiten caracterizar el tablero de ajedrez a lo largo de la secuencia.
- Una vez se han obtenido los puntos de imagen y puntos de objeto, se hace uso de la función *calibrateCamera()* de la librería OpenCV [29], entregando tres elementos:
  - La matriz de **parámetros intrínsecos** M:

$$M = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \tag{3.1}$$

    Donde (f_x, f_y) corresponden a los largos focales en píxeles en el eje X e Y de la cámara, y (c_x, c_y) corresponde a las coordenadas del centro óptico. Se tiene una matriz de cámara propia para **cada uno de los lentes**.
  - La **matriz de rotación** $R^{3\times3}$ y el **vector de traslación** x^{trans}, que se denominan como **parámetros extrínsecos**. Mediante estos, es posible transformar entre el sistema de coordenadas global y el sistema de coordenadas de la cámara.
- Con los parámetros calculados, se corrobora su coherencia des-distorsionando las imágenes provenientes de los lentes: si estas imágenes son coherentes y los parámetros se encuentran dentro de los márgenes esperados, se procede con el resto de la metodología; si no, se procede nuevamente a la calibración de parámetros.

La cámara utilizada en este proyecto corresponde a una cámara **Kinect V1**, que contiene varios sensores, tales como se ilustran en la Figura 3.4:

[14] De manera experimental se ha observado que 30 imágenes entrega parámetros adecuados.

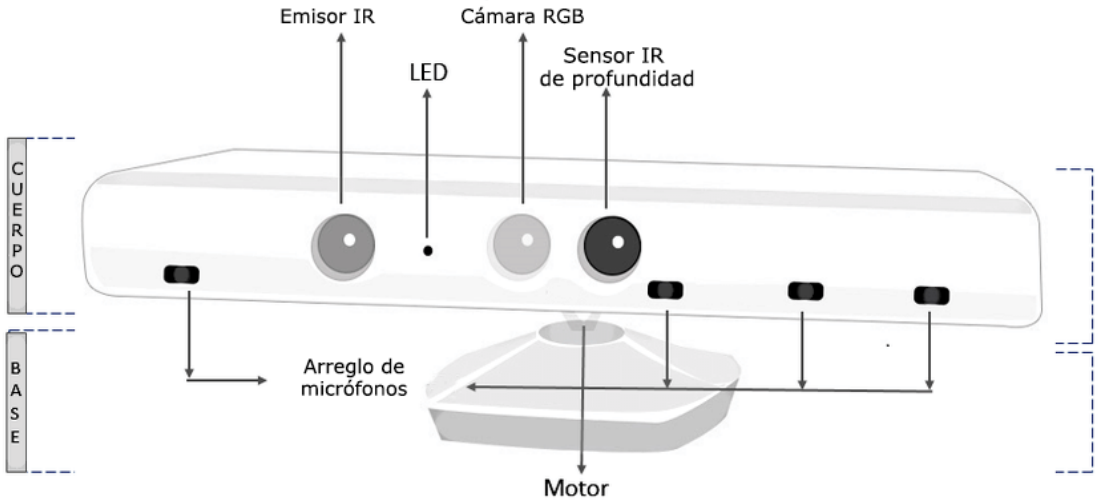

Figura 3.4: Imagen extraída y traducida de [30], donde se detalla la estructura interna de la Kinect. Se aprecia la cámara de color RGB, el emisor InfraRojo, y el sensor InfraRojo de profundidad (receptor InfraRojo), siendo estos dos últimos elementos los encargados de calcular la profundidad. Posee además un arreglo de cuatro micrófonos.

3. **Implementación de modelos de medición y movimiento**:

   - **Modelo de movimiento**: Para operar adecuadamente el algoritmo, es necesario instaurar un modelo de movimiento apropiado al experimento a realizar. Esto requiere delimitar los siguientes aspectos:
     - Definir el movimiento posible únicamente en un plano horizontal, fijando el desplazamiento vertical.
     - Establecer el paso en términos de distancia entre cada imagen capturada.
     - Definir una matriz de covarianza σ_{mv} pertinente al movimiento efectuado.

     La trayectoria en un tiempo k se encuentra delimitada por dos elementos:

     - El desplazamiento realizado a lo largo de los tres ejes cartesianos $\hat{X}, \hat{Y}, \hat{Z}$ mediante el siguiente vector de traslación :

     $$x_k^{trans} = (x_k, y_k, z_k)^T \tag{3.2}$$

     - La orientación a través de un **cuaternión**, que se describe mediante la Ecuación 3.3:

     $$\mathbf{q}_k = (q_k^x, q_k^y, q_k^z, q_k^w)^T \tag{3.3}$$

       Así, se tiene un modelo con siete coordenadas y seis grados de libertad.

   Para modelar los comandos de movimiento en la librería RFS-SLAM, se le añadió ruido a la trayectoria Ground Truth, utilizando como covarianza una matriz diagonal con los valores presentados en la Tabla 3.1. Además, se realizarán experimentos adicionales modificando dichos valores de covarianza; en particular, se verificará su desempeño al multiplicar los valores de covarianza de movimiento 1.5 y por 2 (es decir, si σ_{mv} es la covarianza, se evaluará su desempeño con $1.5\sigma_{mv}$ y $2\sigma_{mv}$).

Tabla 3.1: Valores de covarianza utilizados en la creación de la trayectoria a seguir por la cámara, siendo estos representativos de la **incertidumbre** de movimiento a lo largo de la trayectoria.

	Eje X	Eje Y	Eje Z	q_x	q_y	q_z	q_w
Valores de covarianza (σ_{mv})	1.0e-4	1.0e-5	1.0e-4	0.0	1.0e-4	0.0	1.0e-4
Valores de covarianza modificada ($1.5\sigma_{mv}$)	1.5e-4	1.5e-5	1.5e-4	0.0	1.5e-4	0.0	1.5e-4
Valores de covarianza modificada ($2\sigma_{mv}$)	2.0e-4	2.0e-5	2.0e-4	0.0	2.0e-4	0.0	2.0e-4

- **Modelo de medición**: La librería RFS-SLAM [28] genera *landmarks aleatorios*, calculando mediciones a partir de ellos a lo largo de una trayectoria (por defecto aleatoria). Por lo tanto, es necesario modificar el código para que, en vez de utilizar mediciones provenientes de *landmarks* simulados, reciba coordenadas 3D capturadas de manera externa. Lo anterior conlleva los siguientes pasos:
  - Importar en el código de C++ las coordenadas 3D provenientes de una secuencia de imágenes.
  - Eliminar la creación de *landmarks* simuladas en el algoritmo.
  - Reemplazar las mediciones para cada tiempo k con las coordenadas 3D calculadas.
  - Establecer una matriz de covarianza adecuada para las mediciones σ_{ms}.

  Para poder llevar a cabo lo anterior, se tienen las siguientes consideraciones [7]:
  - Sea $m^i = (m_x^i, m_y^i, m_z^i)^T$ la posición de la i-ésima *landmark* del mapa $\mathcal{M} = \{m^1, ..., m^i, ..., m^n\}$, y $m^{i'} = (m_x^{i'}, m_y^{i'}, m_z^{i'}, m_w^i)^T$ la representación de dicho *landmark* en coordenadas 3D homogéneas (con $m^i = ((m_x^{i'}/m_w^i), (m_y^{i'}/m_w^i), (m_z^{i'}/m_w^i))^T$).
  - Sea también $R_k^{3\times3}$ y x_k^{trans} la matriz de rotación de 3×3 y el vector de posición de la cámara en un tiempo k; y f el largo focal de la cámara.
  - Las coordenadas homogéneas 2D de las características en la cámara se encuentran descritas por la siguiente ecuación:

$$\begin{pmatrix} x_k^{im'} \\ y_k^{im'} \\ w_k \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} R_k^{3\times3} & x_k^{trans} \\ \mathbf{0}_{1\times3} & 1 \end{pmatrix} \begin{pmatrix} m_x^{i'} \\ m_y^{i'} \\ m_z^{i'} \\ m_w^i \end{pmatrix} \tag{3.4}$$

  Donde $(x_k^{im}, y_k^{im})^T = ((x_k^{im'}/w_k), (y_k^{im'}/w_k))$ corresponde a las coordenadas de las características visuales.

4. **Procesamiento de la imagen RGB y de profundidad**: Debido a que la posición de ambos lentes de la cámara son distintos, es necesario alinear las imágenes de profundidad con la imagen RGB [15]. Así, con un procesamiento de la información de profundidad, será posible determinar la posición en el espacio de un objeto de interés visto en la cámara RGB. Para ello, se realizan los siguientes pasos:

[15] En inglés este proceso se conoce como **depth registration**.

- Se corrige la distorsión de las imágenes de profundidad y RGB utilizando sus parámetros **intrínsecos**.
- Se utilizan los parámetros **extrínsecos** para rotar y trasladar la imagen de profundidad, alineando esta con la imagen RGB.

Así, una vez realizado lo anterior, se tendrá que cada píxel de la imagen RGB tendrá una asociación directa en la imagen de profundidad.

5. **Extracción de características**: Con la información visual proveniente de la cámara RGB, se utilizó en primera instancia un detector de esquinas FAST (Features from Accelerated Segment Test), permitiendo estos detectar **puntos clave** en dicha imagen.

   Una vez se verificó el funcionamiento adecuado del detector de esquinas FAST, se procedió a reemplazar este por el detector de puntos clave y descriptor ORB. Esto se debe a que los puntos clave capturados provienen de FAST, y presentan además las ventajas de ser filtrados mediante la métrica de esquinas de Harris y son "parcialmente" invariantes en escala; no obstante, se destaca que no se hizo uso de los descriptores rBRIEF generador por ORB. Por último, se utilizó un algoritmo de filtrado a los *keypoints* obtenidos por ORB, con el fin de hacer más homogénea la captura de información [31].

6. **Cálculo de coordenadas 3D a partir de la cámara de profundidad**: Con el fin de poder obtener la posición de píxeles de interés en el espacio 3D, se requerirá transformar los valores provenientes de la cámara de profundidad: Dichos valores usualmente se encuentran entre 0 y 2047, por lo que es necesario convertir estos a coordenadas cartesianas. Para ello, se realizan los siguientes pasos:

   a) Se convierten los valores de profundidad para que queden en metros mediante la siguiente ecuación [32]:

   $$d_m = \frac{1}{(-0.0030711016 \cdot d_{raw}) + 3.3309495161} \tag{3.5}$$

   Donde d_{raw} corresponde a valores de la imagen de profundidad entre 0 a 2048, y d_m corresponde a valores de la imagen de profundidad en metros.

   b) Posteriormente, se itera sobre cada coordenada de los *keypoints* (u, v) para obtener las coordenadas en metros x_m, y_m, y z_m utilizando las siguientes fórmulas [32]:

   $$z_m = d_m \tag{3.6}$$

   $$x_m = \frac{ax \cdot (u - c_x^d) \cdot z_m}{f_x^d} \tag{3.7}$$

   $$y_m = \frac{ax \cdot (v - c_y^d) \cdot z_m}{f_y^d} \tag{3.8}$$

   Donde:

   - f_x^d, f_y^d, c_x^d, y c_y^d corresponden a elementos de la matriz de parámetros intrínsecos de la cámara de profundidad descritos en la Ecuación 3.1.

- ax corresponde a un valor que determina la orientación de los ejes del plano de la imagen:
  - Si $ax = 1$, el eje X apunta hacia la derecha, Y hacia abajo y Z hacia delante.
  - Si $ax = -1$, el eje X apunta hacia la izquierda, Y hacia arriba, y Z hacia delante. En este proyecto se utilizará este valor.
- El origen $(0, 0, 0)$ se encuentra en el centro del lente del receptor InfraRojo (IR Depth Sensor en Figura 3.4).

c) Finalmente, se guardan las coordenadas 3D de los elementos detectados en disco duro para todas las imágenes de una secuencia determinada.

Para procesar la incertidumbre de dichas mediciones capturadas en RFS-SLAM, se les asignó una matriz de covarianza σ_{ms} diagonal, cuyos elementos σ^x_{ms}, σ^y_{ms}, y σ^z_{ms} se ilustran en la Tabla 3.2

Tabla 3.2: Valores de σ_{ms}, siendo estos representativos de la **incertidumbre** de las mediciones realizadas.

	Valor de Covarianza
σ^x_{ms}	5.0e-3
σ^y_{ms}	5.0e-3
σ^z_{ms}	5.0e-3

7. **Campo de visión de cámara y probabilidad de detección**:

- El algoritmo RFS-SLAM cuenta por defecto con una probabilidad de detección omni-direccional que le permite tomar mediciones en cualquier ángulo.
- Para poder adaptar correctamente el modelo de medición de la cámara al algoritmo, es necesario modificar el **Campo de Visión** (*Field of View, FoV*) de la simulación de tal forma que sea similar al de la Kinect.
- Si lo anterior no se realiza correctamente, se generará que, al momento de estimar *landmarks* en la trayectoria, estos sólo permanezcan en el mapa mientras la cámara los observe; una vez que estos no se encuentren en el FoV, se borrarán del mapa, impidiendo la elaboración adecuada del mismo, afectando así estimación de la trayectoria.

Dado la información obtenida en [33], se tiene en la Tabla 3.3 los siguientes ángulos horizontales y verticales para el FoV de la Kinect:

Tabla 3.3: Valores del FoV para la Kinect.

	Ángulos de FoV
Ángulo Horiz.	62°
Ángulo Vert.	48.6°

También [33] ilustra los rangos máximos y mínimos de profundidad de la Kinect, presentes en la Tabla 3.4 [16]:

[16] En la práctica, se observó que puede entregar valores entre 0.8 [m] y 4 [m], pero la mayor precisión la obtiene dentro del rango [1.2, 3.5].

Tabla 3.4: Rangos máximos y mínimos de la Kinect.

	Distancia [m]
Rango mínimo	3.5
Rango máximo	1.2

- Así, con dicha información, se diseña un campo de visión con forma de **tronco de pirámide**, tal como se muestra en las Figuras 3.5 y 3.6:

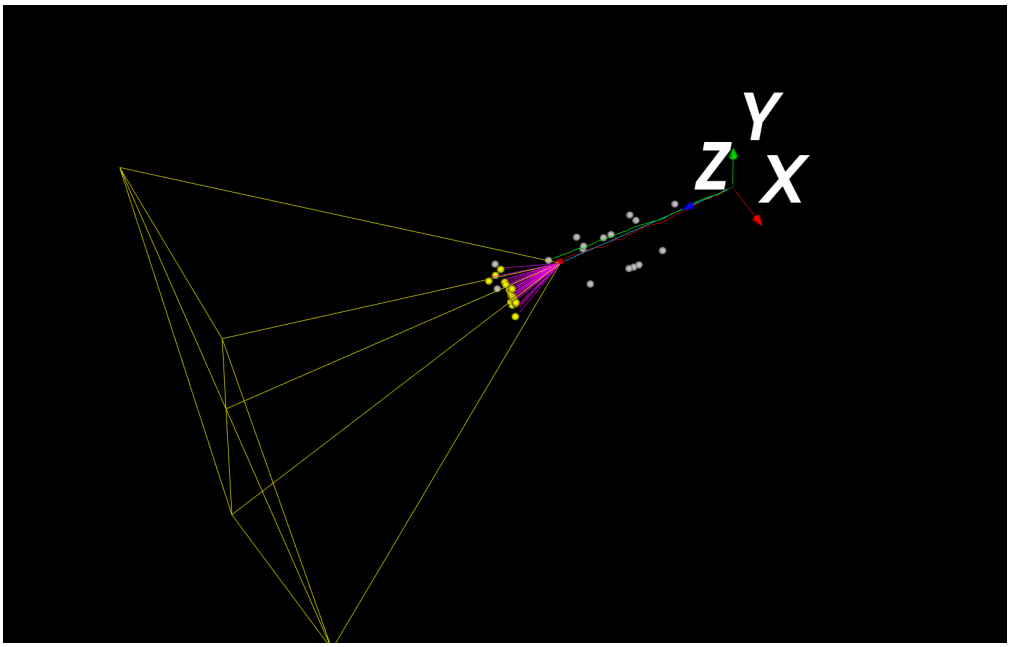

Figura 3.5: Imagen del algoritmo RFS-SLAM con el FoV funcional en color amarillo. En la imagen se aprecian *landmarks* delimitados como esferas blancas, *landmarks* localizados en el FoV como esferas amarillas, las mediciones como líneas moradas, la trayectoria *Ground Truth*, los comandos de movimiento y la trayectoria estimada como las rutas de color celeste, verde y rojo respectivamente.

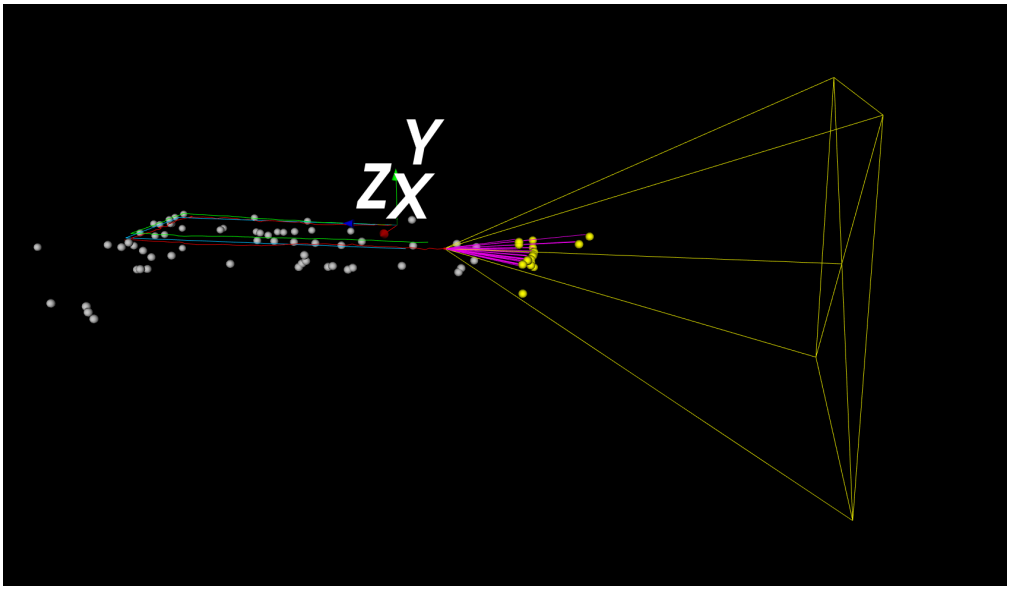

Figura 3.6: Imagen del algoritmo RFS-SLAM con el FoV funcional desde otro ángulo y en otro lugar de la trayectoria.

- Dado lo anterior, se determina que los elementos que se encuentran dentro del FoV poseen una probabilidad de detección distinta de cero, otorgada por el experimento. Por otra parte, los elementos que no se encuentran presentes en el FoV se les asigna una probabilidad igual a 0.

8. **Métrica de error de conjuntos**: Para poder medir la calidad de la estimación de los *landmarks* obtenidos por RFS-SLAM, es necesario contar con una métrica de estimación de error de conjuntos aleatorios. El código RFS-SLAM utilizado cuenta con las métricas *Optimal Sub-Pattern Assignment (OSPA)* [34], y *Cardinalized Optimal Lineal Assignment (COLA)* [35], las que permiten evaluar el desempeño del mapa generado.

Capítulo 4

Resultados

4.1. Experimento base

4.1.1. Trayectorias

Se presenta en la Figura 4.1 los siguientes elementos:

- En azul se presenta la trayectoria a seguir ideal de la cámara, denominada como **ground truth**.
- En verde se ilustran los movimientos realizados a la silla, siendo su concatenación llamada **dead reckoning**.
- Por último, en rojo se tiene la **estimación de la trayectoria de la cámara** realizada por RFS-SLAM.

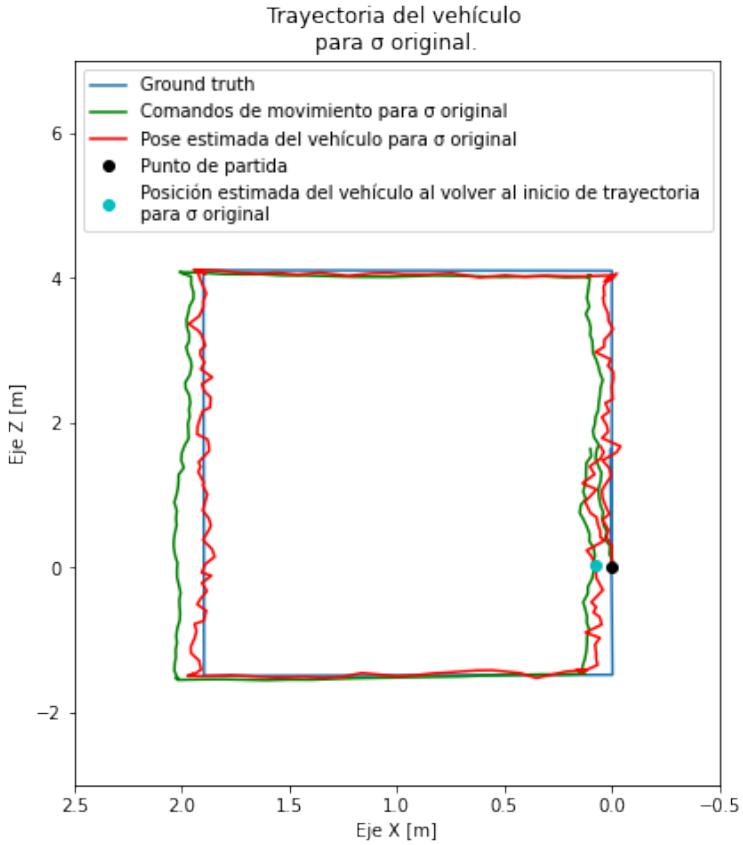

Figura 4.1: Comparación de la trayectoria ideal (*Ground truth*, en azul), los movimientos realizados a la silla (*Dead reckoning*, en verde) y la trayectoria estimada en rojo en los ejes X y Z. Se muestra además en negro el punto de partida de la cámara, y en celeste la posición estimada de la cámara al retornar a dicho punto.

Mediante la trayectoria elaborada, se logró capturar mediciones de posición de elementos de interés respecto a la cámara, e implementar estas exitosamente en el algoritmo RFS-SLAM.

Se compara en las Figuras 4.2, 4.3, y 4.4 la trayectoria realizada por la cámara en los ejes X, Y y Z respectivamente:

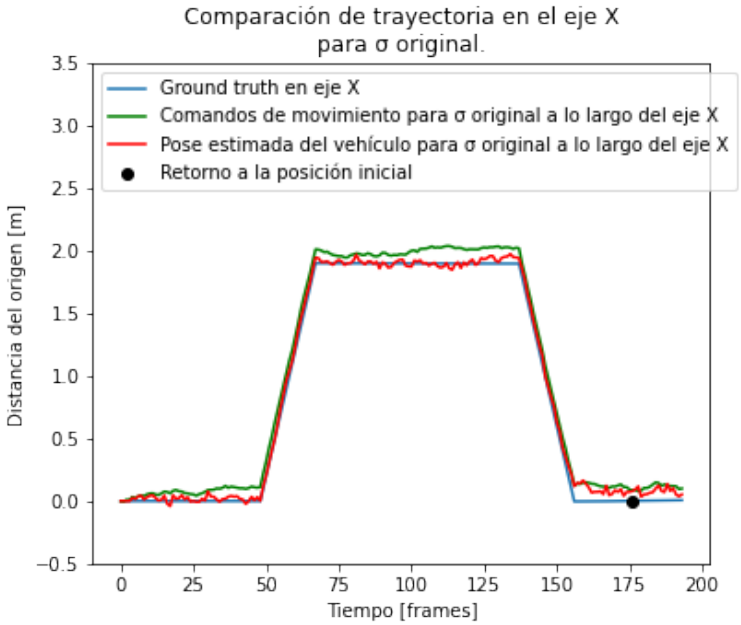

Figura 4.2: Comparación de la trayectoria ideal (*Ground truth*, en azul), los movimientos realizados a la silla (Dead reckoning, en verde) y la trayectoria estimada en rojo en el eje X. Se observa además en negro la posición ideal de la cámara en dicho eje al momento de volver al punto de origen.

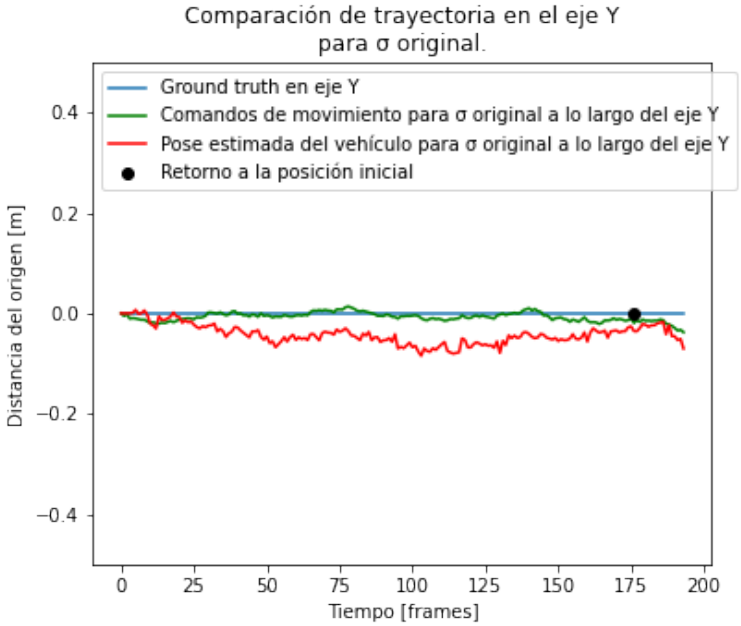

Figura 4.3: Comparación de la trayectoria ideal (*Ground truth*, en azul), los movimientos realizados a la silla (Dead reckoning, en verde) y la trayectoria estimada en rojo en el eje Y. Se observa además en negro la posición ideal de la cámara en dicho eje al momento de volver al punto de origen.

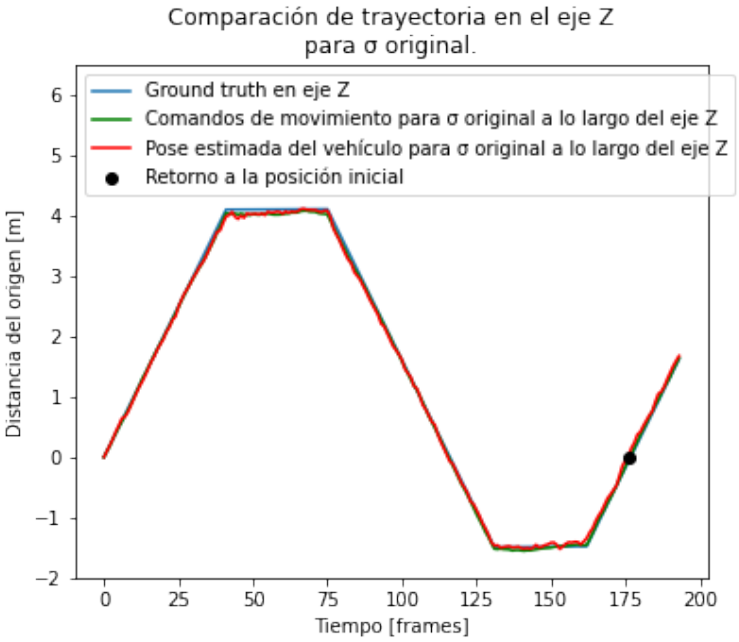

Figura 4.4: Comparación de la trayectoria ideal (*Ground truth*, en azul), los movimientos realizados a la silla (Dead reckoning, en verde) y la trayectoria estimada en rojo en el eje Z. Se observa además en negro la posición ideal de la cámara en dicho eje al momento de volver al punto de origen.

4.1.2. Estimaciones de error

Se tiene a continuación en la Figura 4.5 la estimación de error en distancia para cada uno de los ejes tanto del *dead reckoning* como de la trayectoria estimada:

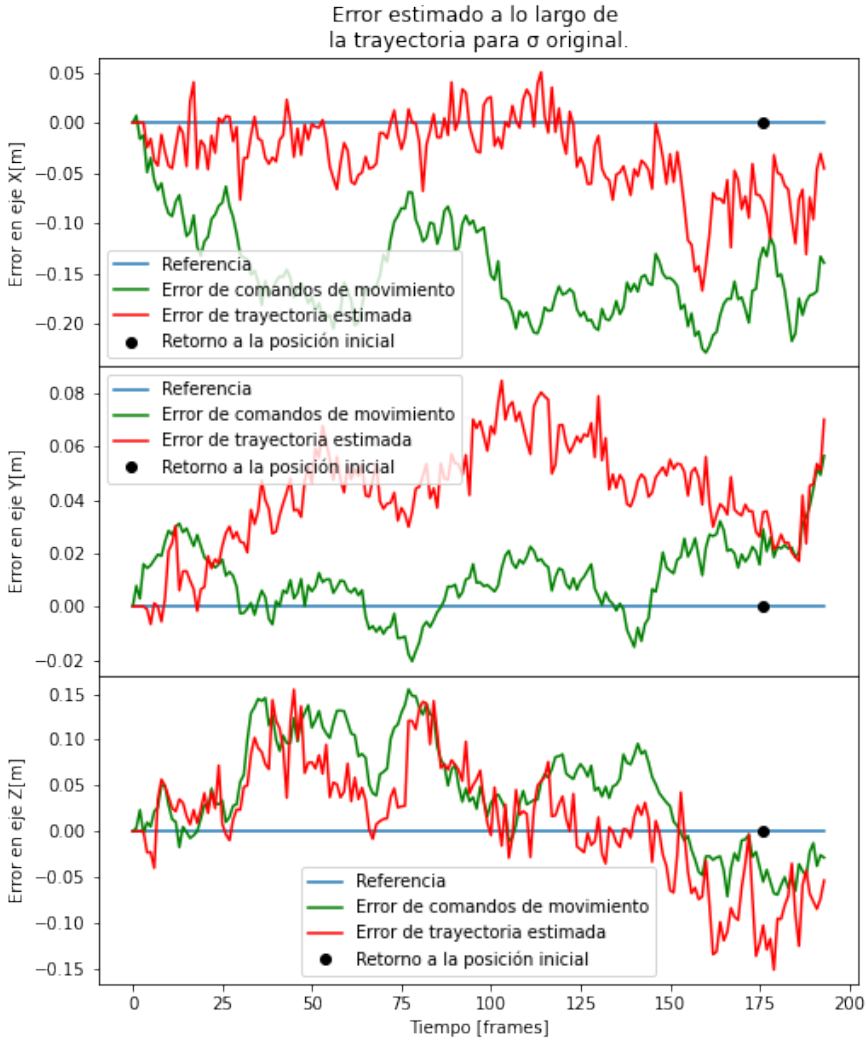

Figura 4.5: Errores a lo largo de la trayectoria en cada uno de los ejes cartesianos. Se observa que, con la excepción del eje Y, la tendencia de error de la trayectoria estimada se encuentra por debajo de los comandos de control de la cámara.

Por otra parte, en la Figura 4.6 se presenta la estimación de error de los cuaterniones:

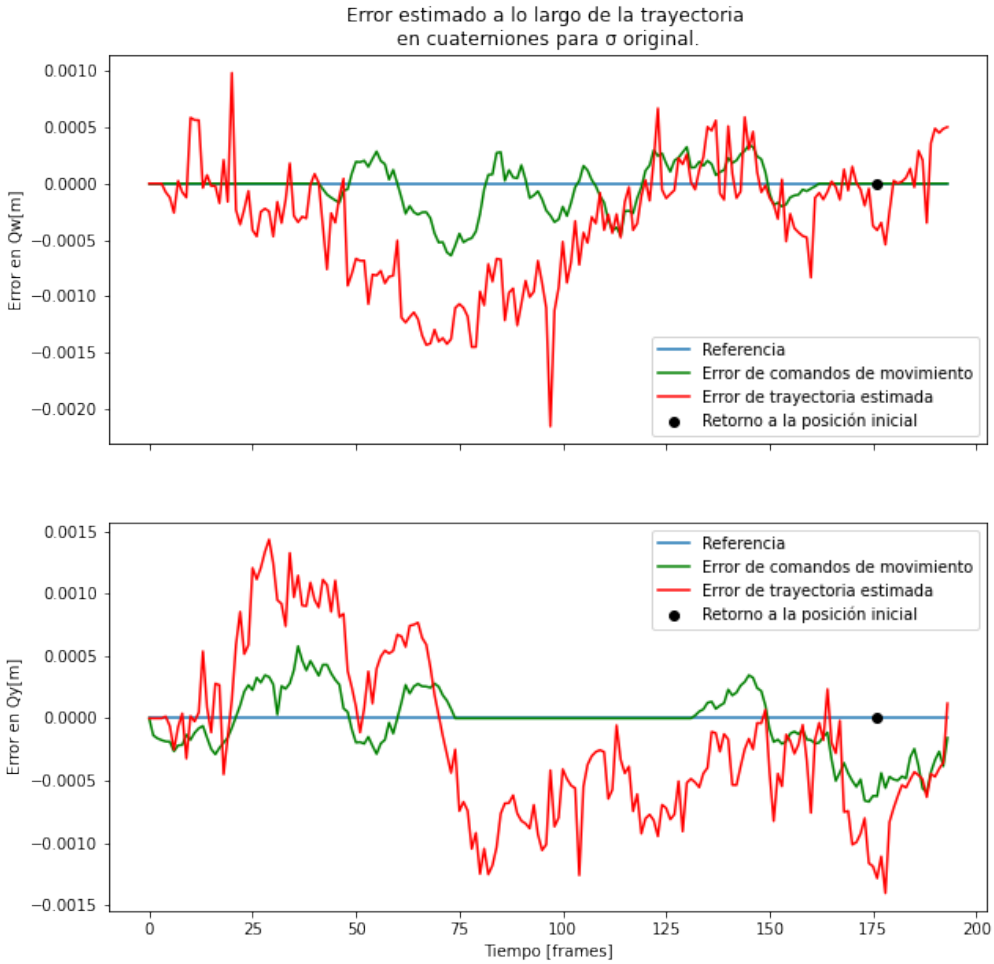

Figura 4.6: Errores a lo largo de la trayectoria para Q_w y Q_y. Se observa una diferencia moderada de la trayectoria estimada respecto a los comandos de control.

4.1.2.1. Mapa generado

Las Figuras 4.7, 4.8 y 4.9 ilustran el mapa estimado mediante RFS-SLAM:

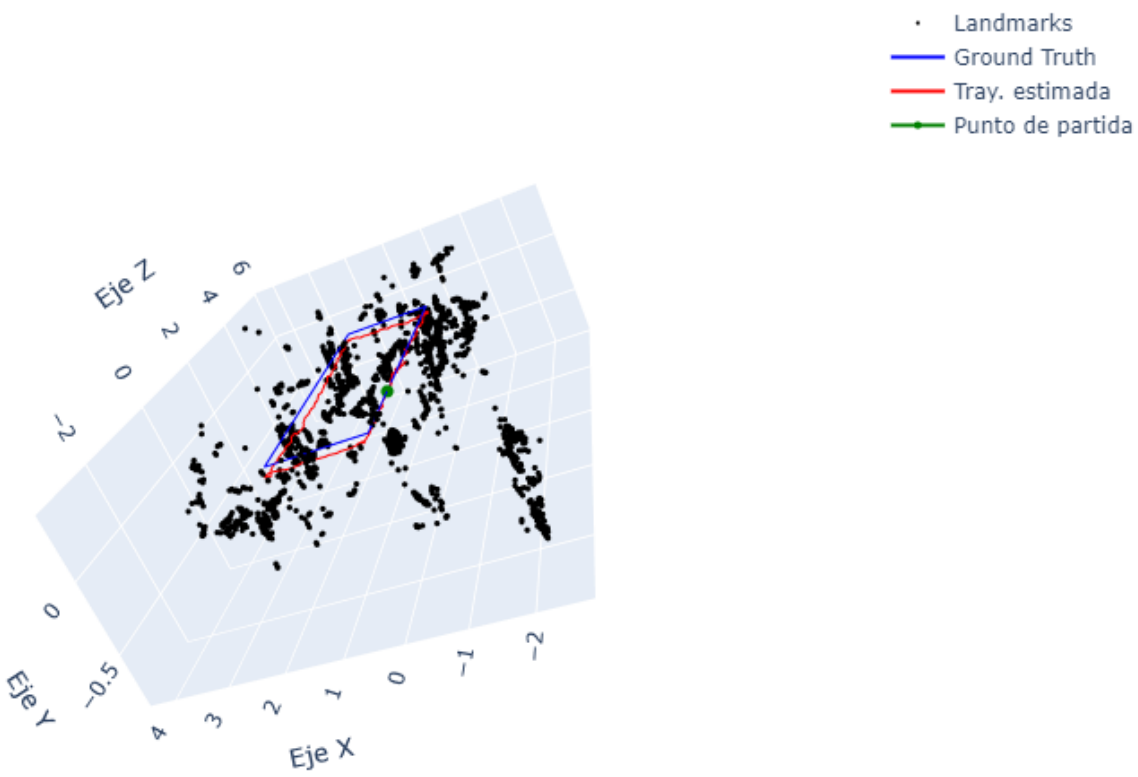

Figura 4.7: Gráfica del Mapa estimado por RFS-SLAM, en donde los *landmarks* son representados por puntos de color negro, *Ground Truth* en azul, la trayectoria estimada por RFS-SLAM en rojo, y el Punto de partida de la cámara (definido a la vez como el origen global del mapa) en verde.

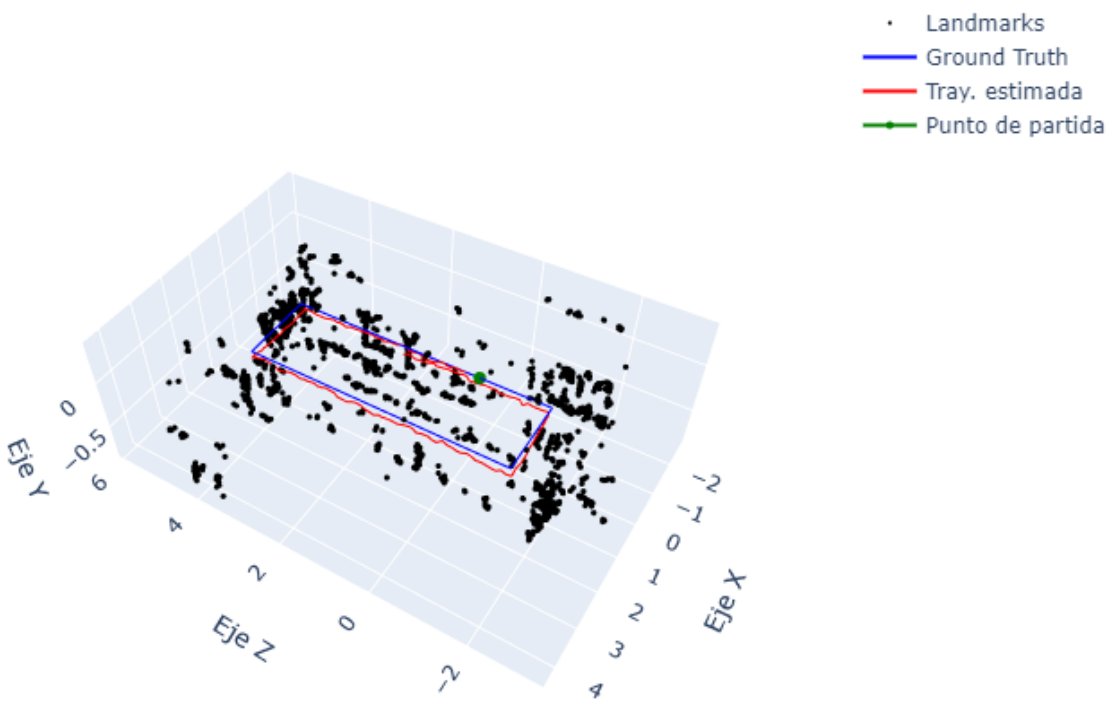

Figura 4.8: Gráfica del Mapa estimado por RFS-SLAM, en donde los *landmarks* son representados por puntos de color negro, *Ground Truth* en azul, la trayectoria estimada por RFS-SLAM en rojo, y el Punto de partida de la cámara (definido a la vez como el origen global del mapa) en verde.

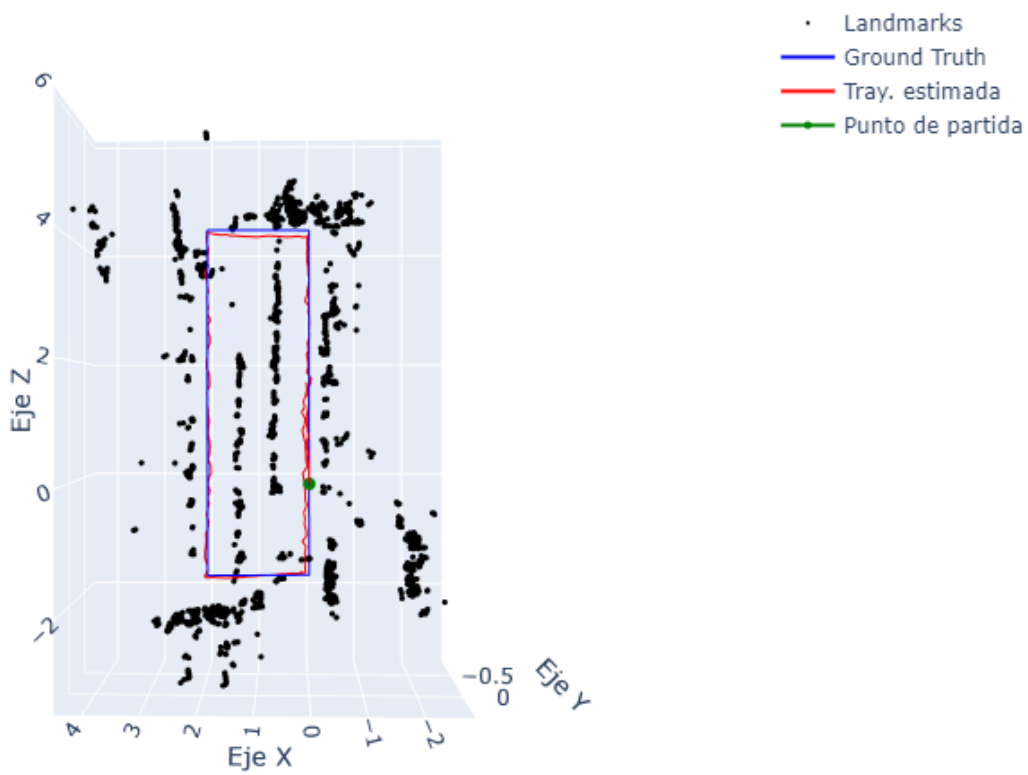

Figura 4.9: Gráfica del Mapa estimado por RFS-SLAM, en donde los *landmarks* son representados por puntos de color negro, *Ground Truth* en azul, la trayectoria estimada por RFS-SLAM en rojo, y el Punto de partida de la cámara (definido a la vez como el origen global del mapa) en verde.

Se registró un total de 13155 *landmarks*.

4.2. Resultados con diferentes valores de covarianza

Se ilustran a continuación los resultados producto de multiplicar por 1.5 y por 2 las covarianzas de movimiento.

4.2.1. 1.5 veces Covarianzas de movimiento

4.2.1.1. Trayectorias

La Figura 4.10 ilustra las tres trayectorias principales (*Ground Truth*, comandos de control y trayectoria estimada por RFS-SLAM), al amplificar la covarianza de movimiento por un factor de 1.5. A su vez, las Figuras 4.11, 4.12 y 4.13 representan de manera gráfica los valores de dichas trayectorias en los ejes X, Y y Z respectivamente.

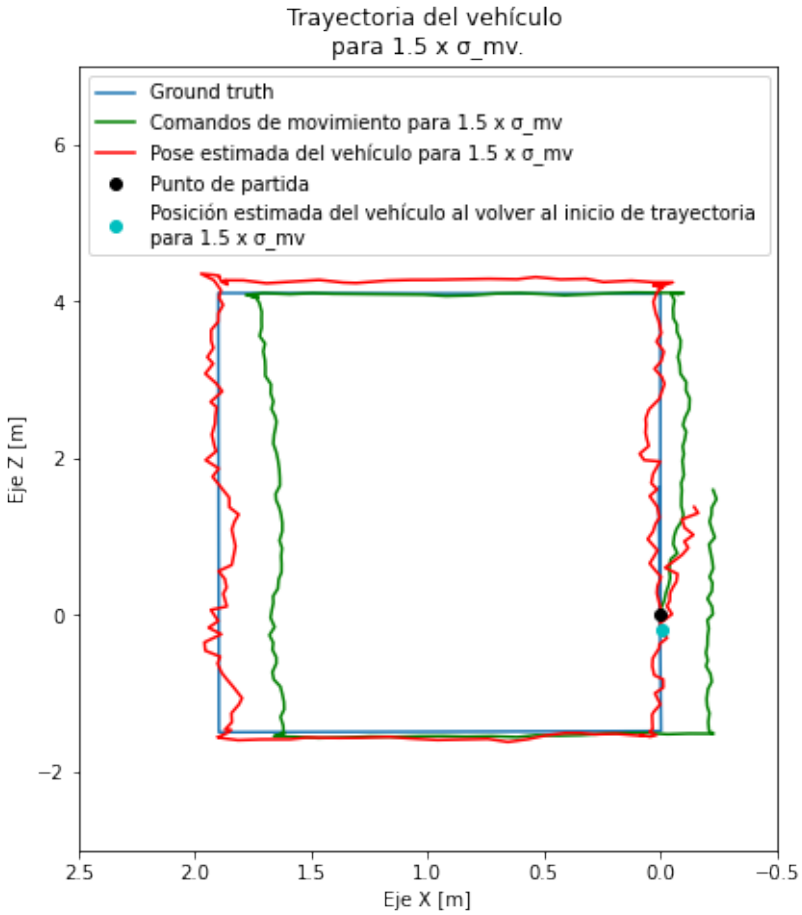

Figura 4.10: Comparación de *Ground Truth* (azul), comandos de movimiento (verde), y trayectoria estimada (rojo), a lo largo de los ejes X y Z al aumentar en 1.5 veces la covarianza descrita en la Tabla 3.1. Se muestra además en negro el punto de partida de la cámara, y en celeste la posición estimada de la cámara al retornar a dicho punto.

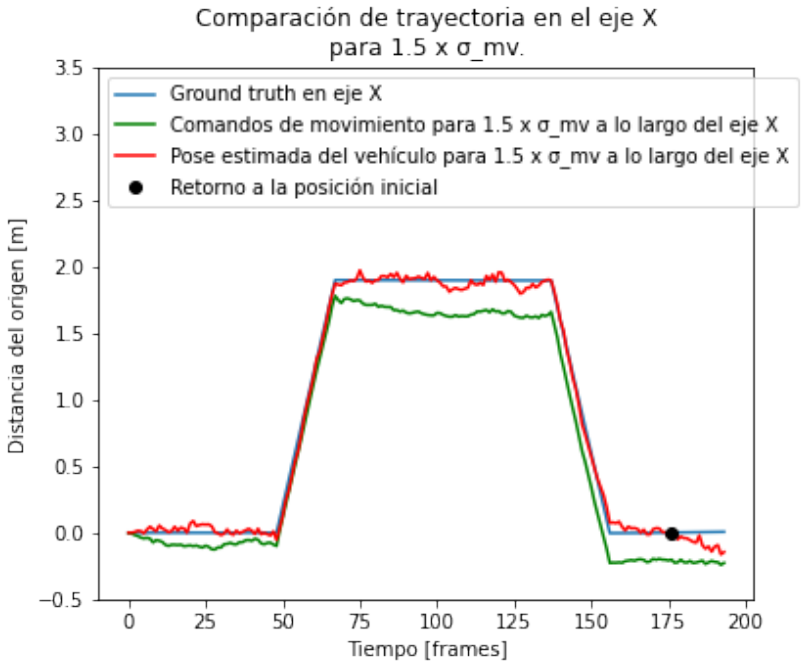

Figura 4.11: Comparación de *Ground Truth* (azul), comandos de movimiento (verde), y trayectoria estimada (rojo), a lo largo del eje X al aumentar en 1.5 veces la covarianza descrita en la Tabla 3.1. Se muestra además en negro la posición ideal de la cámara en dicho eje al momento de retornar al punto de origen.

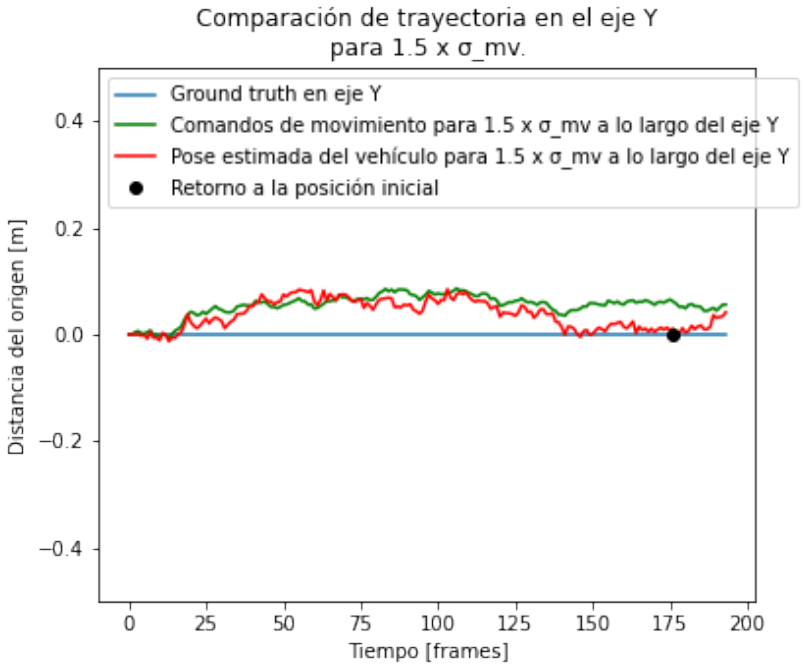

Figura 4.12: Comparación de *Ground Truth* (azul), comandos de movimiento (verde), y trayectoria estimada (rojo), a lo largo del eje Y al aumentar en 1.5 veces la covarianza descrita en la Tabla 3.1. Se muestra además en negro la posición ideal de la cámara en dicho eje al momento de retornar al punto de origen.

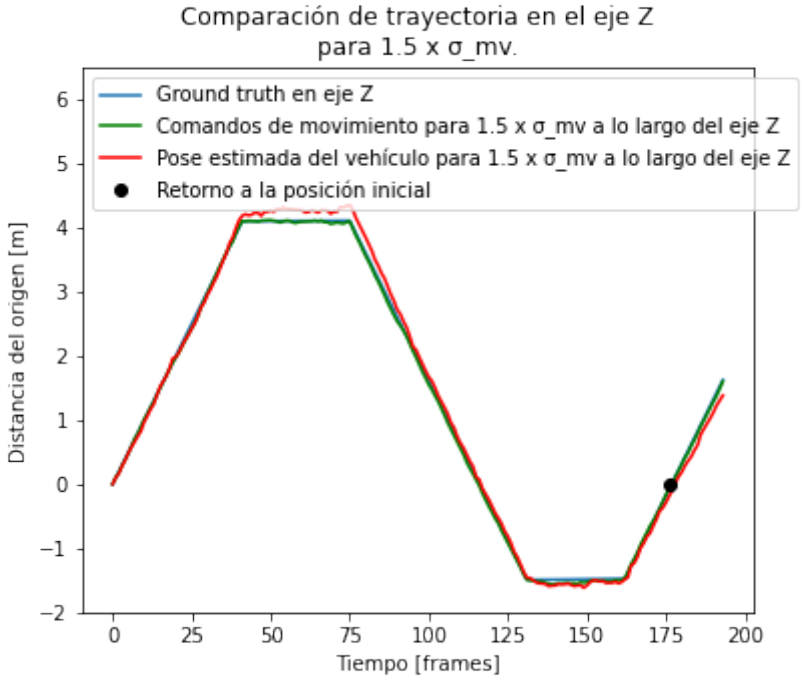

Figura 4.13: Comparación de *Ground Truth* (azul), comandos de movimiento (verde), y trayectoria estimada (rojo), a lo largo del eje Z al aumentar en 1.5 veces la covarianza descrita en la Tabla 3.1. Se muestra además en negro la posición ideal de la cámara en dicho eje al momento de retornar al punto de origen.

4.2.1.2. Estimaciones de error

Se tiene a continuación en la Figura 4.14 la estimación de error en distancia para cada uno de los ejes tanto del *dead reckoning* como de la trayectoria estimada:

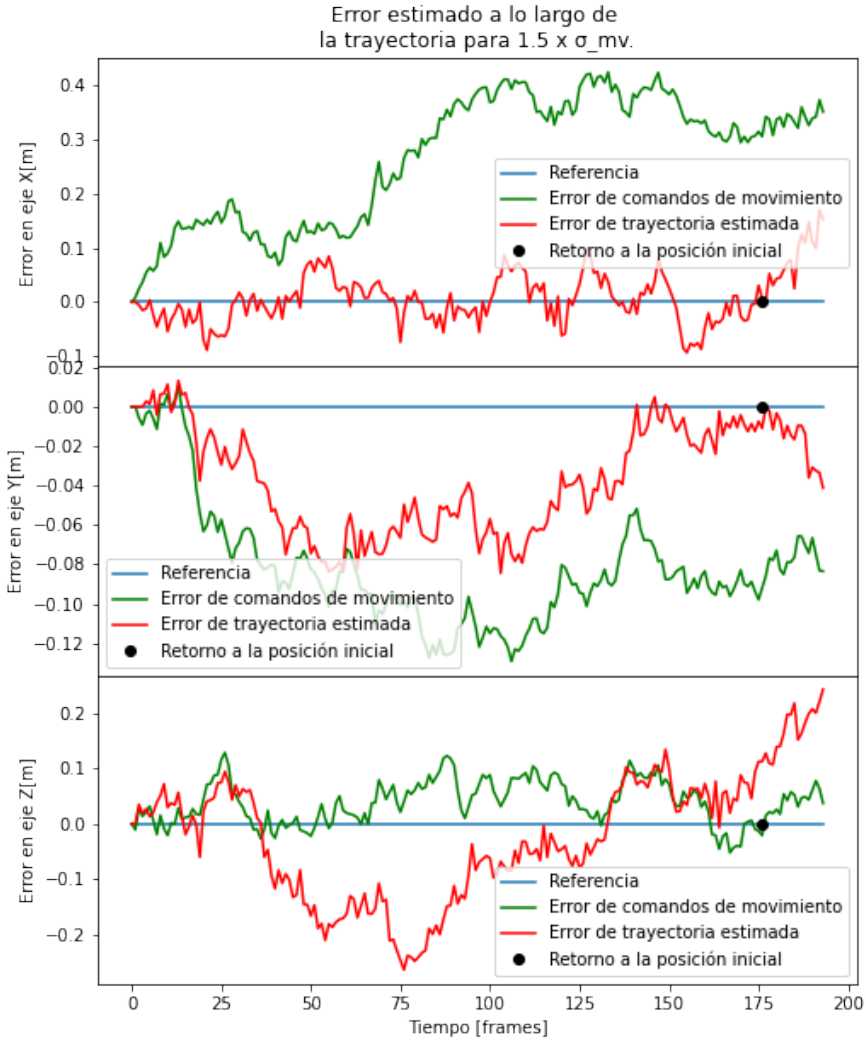

Figura 4.14: Errores a lo largo de la trayectoria en cada uno de los ejes cartesianos al aumentar en 1.5 veces la covarianza descrita en la Tabla 3.1. Se observa que la tendencia de error de la trayectoria estimada se encuentra por debajo de los comandos de control de la cámara de manera predominante con la excepción del eje Z.

Por otra parte, en la Figura 4.15 se presenta la estimación de error de los cuaterniones:

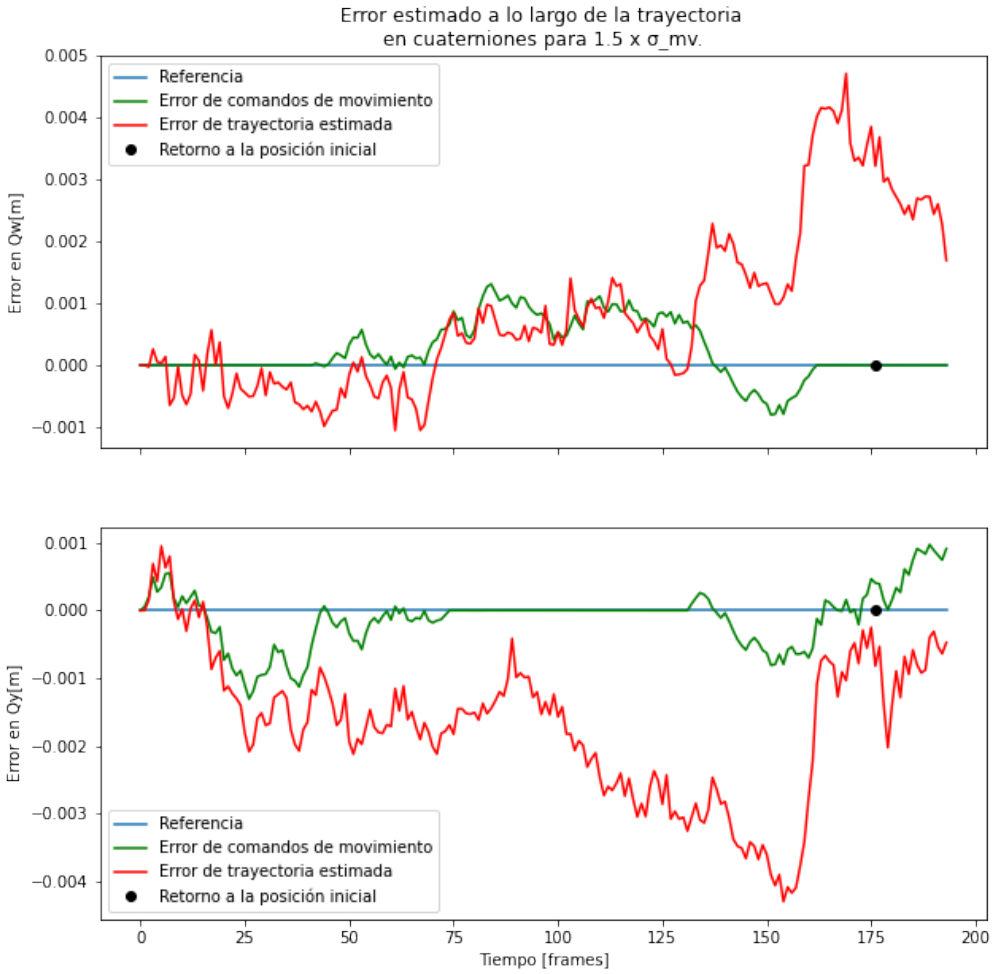

Figura 4.15: Errores a lo largo de la trayectoria para Q_w y Q_y al aumentar en 1.5 veces la covarianza descrita en la Tabla 3.1. Se observa una diferencia moderada de la trayectoria estimada respecto a los comandos de control.

4.2.1.3. Mapa generado

Las Figuras 4.16, 4.17 y 4.18 ilustran el mapa estimado mediante RFS-SLAM:

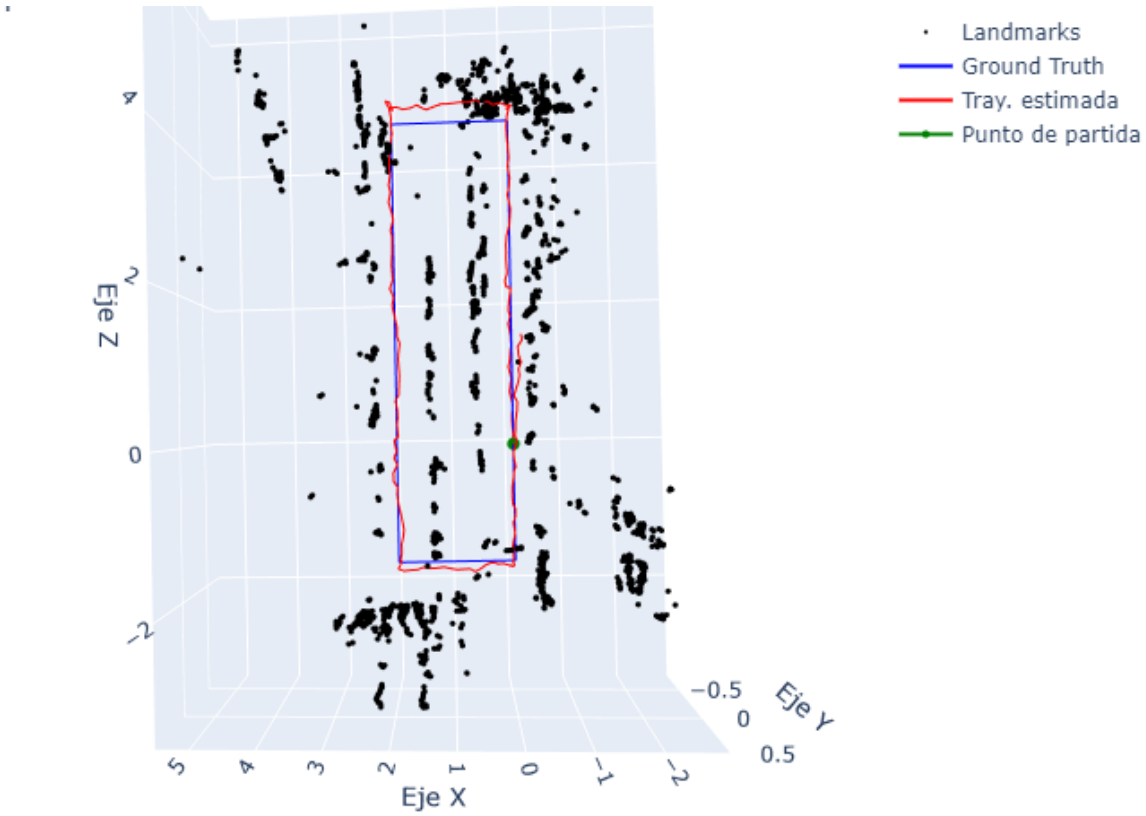

Figura 4.16: Gráfica del Mapa estimado por RFS-SLAM, en donde los *landmarks* son representados por puntos de color negro, *Ground Truth* en azul, la trayectoria estimada por RFS-SLAM en rojo, y el Punto de partida de la cámara (definido a la vez como el origen global del mapa) en verde.

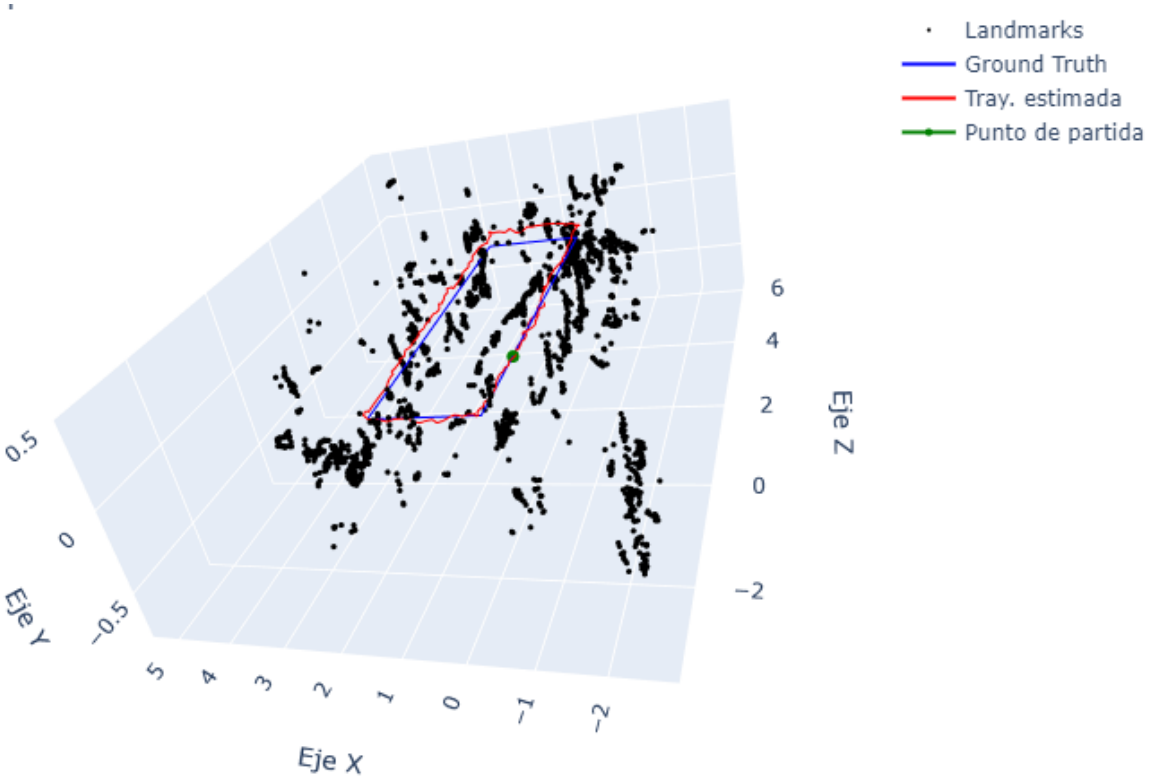

Figura 4.17: Gráfica del Mapa estimado por RFS-SLAM, en donde los *landmarks* son representados por puntos de color negro, *Ground Truth* en azul, la trayectoria estimada por RFS-SLAM en rojo, y el Punto de partida de la cámara (definido a la vez como el origen global del mapa) en verde.

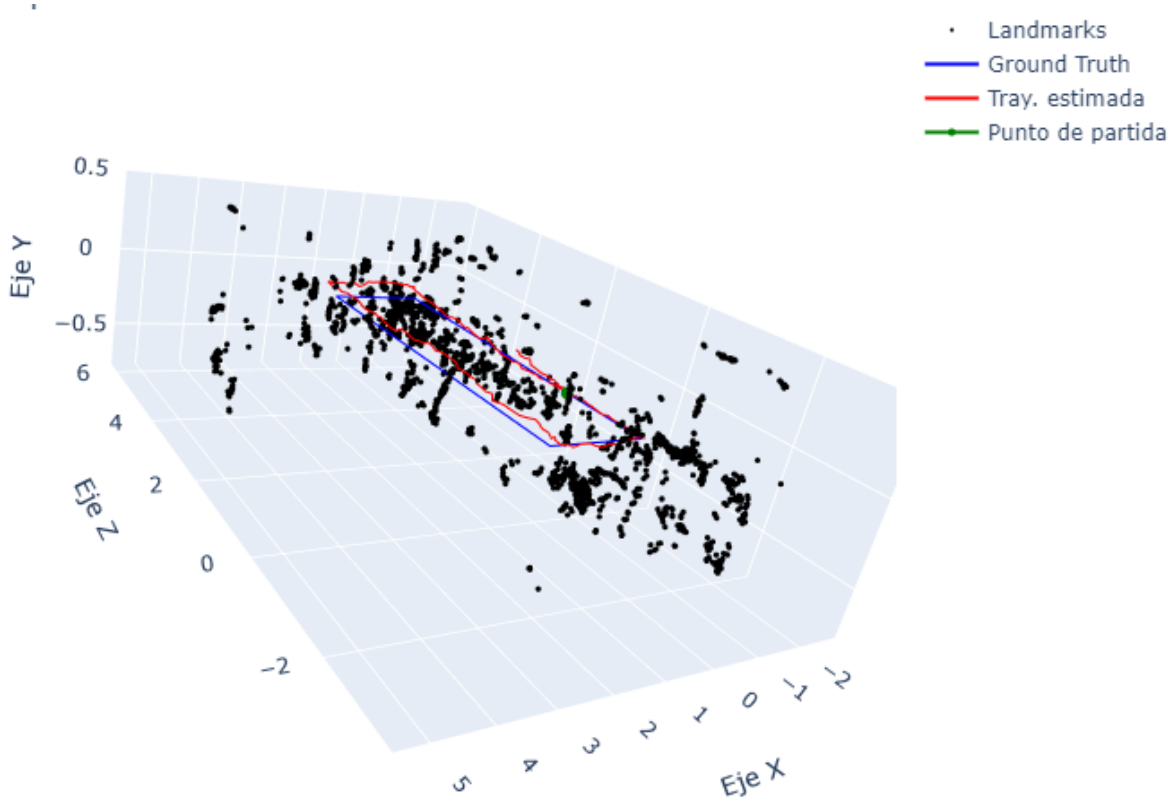

Figura 4.18: Gráfica del Mapa estimado por RFS-SLAM, en donde los *landmarks* son representados por puntos de color negro, *Ground Truth* en azul, la trayectoria estimada por RFS-SLAM en rojo, y el Punto de partida de la cámara (definido a la vez como el origen global del mapa) en verde.

Se registró un total de 14639 *landmarks*.

4.2.2. 2 veces Covarianzas de movimiento

4.2.2.1. Trayectorias

La Figura 4.19 ilustra las tres trayectorias principales (*Ground Truth*, comandos de control y trayectoria estimada por RFS-SLAM), al amplificar la covarianza de movimiento por un factor de 2. A su vez, las Figuras 4.20, 4.21 y 4.22 representan de manera gráfica los valores de dichas trayectorias en los ejes X, Y y Z respectivamente.

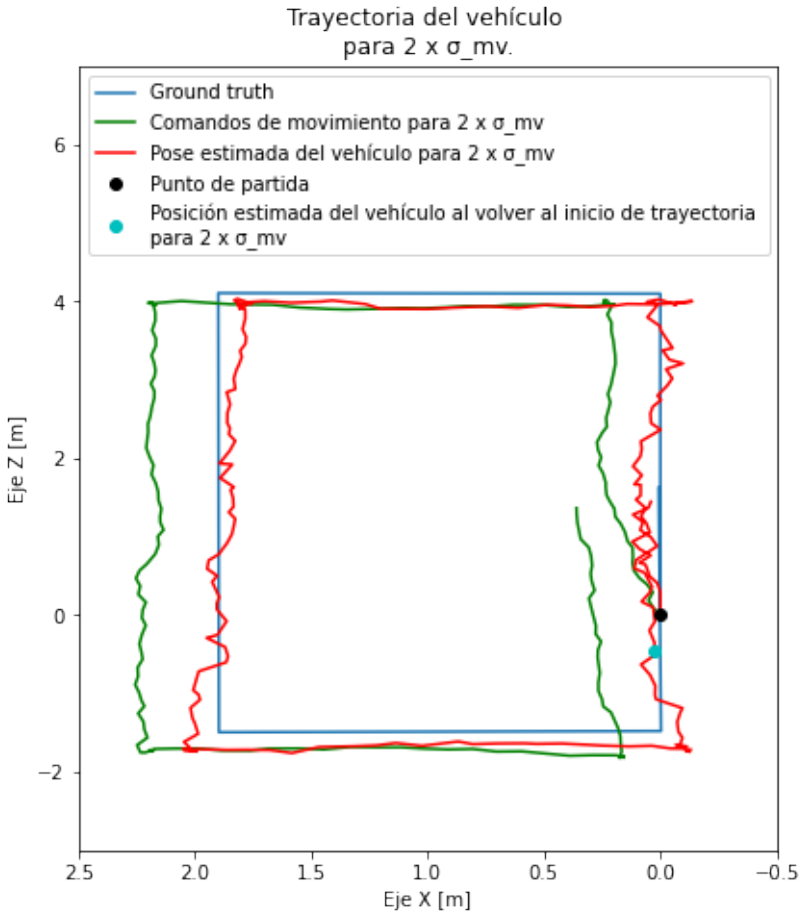

Figura 4.19: Comparación de *Ground Truth* (azul), comandos de movimiento (verde), y trayectoria estimada (rojo), a lo largo de los ejes X y Z al amplificar la covarianza de movimiento por un factor de 2. Se muestra además en negro el punto de partida de la cámara, y en celeste la posición estimada de la cámara al retornar a dicho punto.

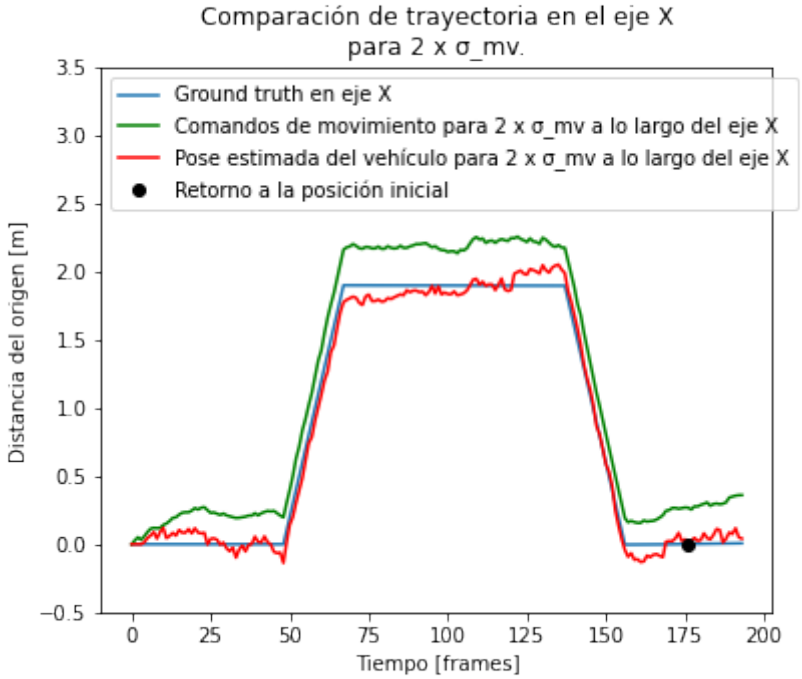

Figura 4.20: Comparación de *Ground Truth* (azul), comandos de movimiento (verde), y trayectoria estimada (rojo), a lo largo del eje X al amplificar la covarianza de movimiento por un factor de 2. Se muestra además en negro la posición ideal de la cámara en dicho eje al momento de retornar al punto de origen.

Figura 4.21: Comparación de *Ground Truth* (azul), comandos de movimiento (verde), y trayectoria estimada (rojo), a lo largo del eje Y al amplificar la covarianza de movimiento por un factor de 2. Se muestra además en negro la posición ideal de la cámara en dicho eje al momento de retornar al punto de origen.

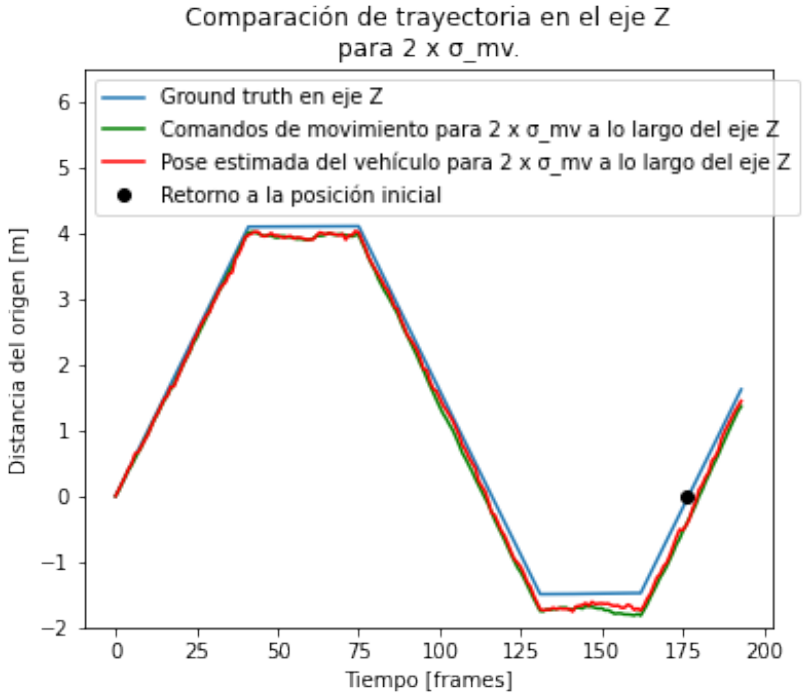

Figura 4.22: Comparación de *Ground Truth* (azul), comandos de movimiento (verde), y trayectoria estimada (rojo), a lo largo del eje Z al amplificar la covarianza de movimiento por un factor de 2. Se muestra además en negro la posición ideal de la cámara en dicho eje al momento de retornar al punto de origen.

4.2.2.2. Estimaciones de error

Se tiene a continuación en la Figura 4.23 la estimación de error en distancia para cada uno de los ejes tanto del *dead reckoning* como de la trayectoria estimada:

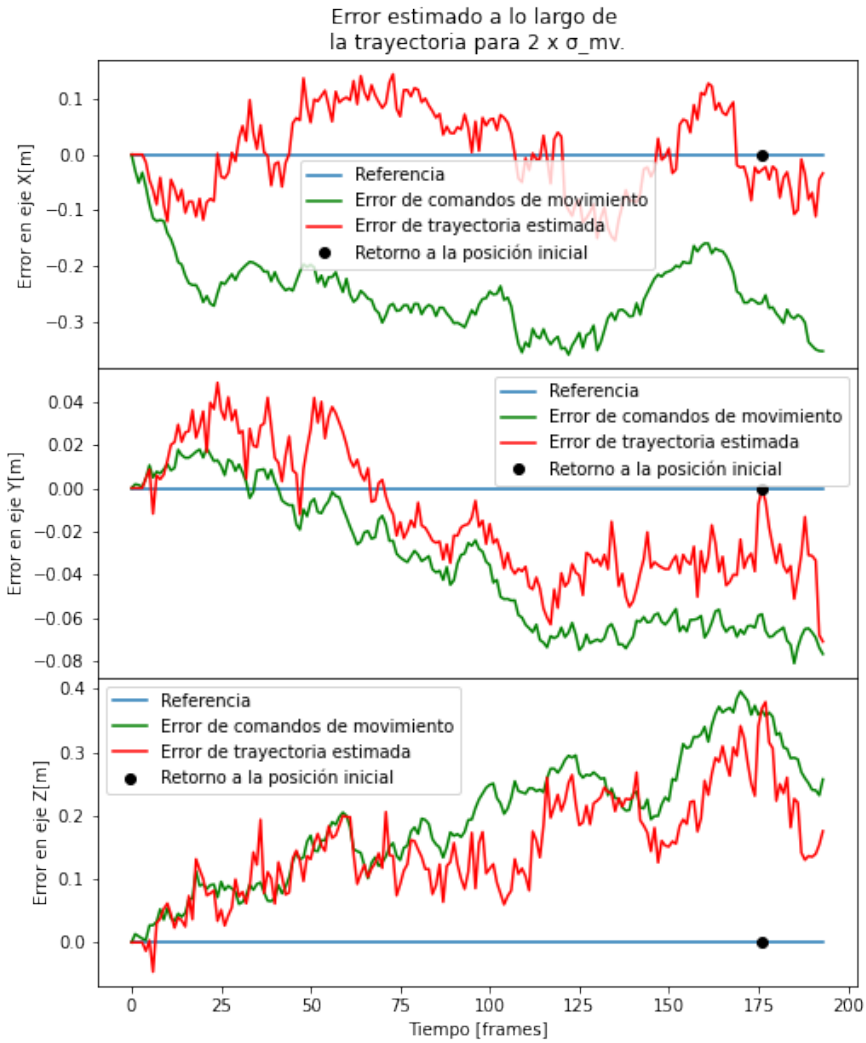

Figura 4.23: Errores a lo largo de la trayectoria en cada uno de los ejes cartesianos al amplificar la covarianza de movimiento por un factor de 2. Se observa que la tendencia de error de la trayectoria estimada se encuentra por debajo de los comandos de control de la cámara de manera predominante con la excepción del eje Y.

Por otra parte, en la Figura 4.24 se presenta la estimación de error de los cuaterniones:

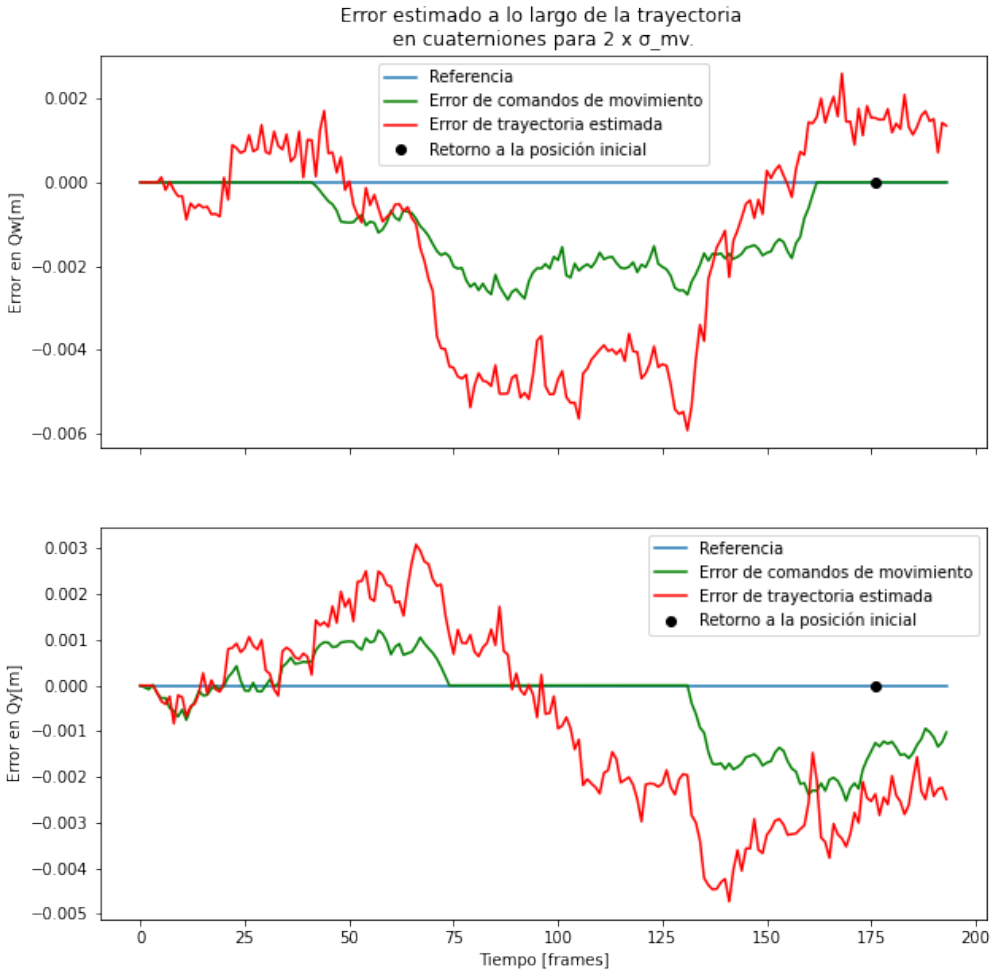

Figura 4.24: Errores a lo largo de la trayectoria para Q_w y Q_y al amplificar la covarianza de movimiento por un factor de 2. Se observa una diferencia moderada de la trayectoria estimada respecto a los comandos de control.

4.2.2.3. Mapa generado

Las Figuras 4.25, 4.26 y 4.27 ilustran el mapa estimado mediante RFS-SLAM:

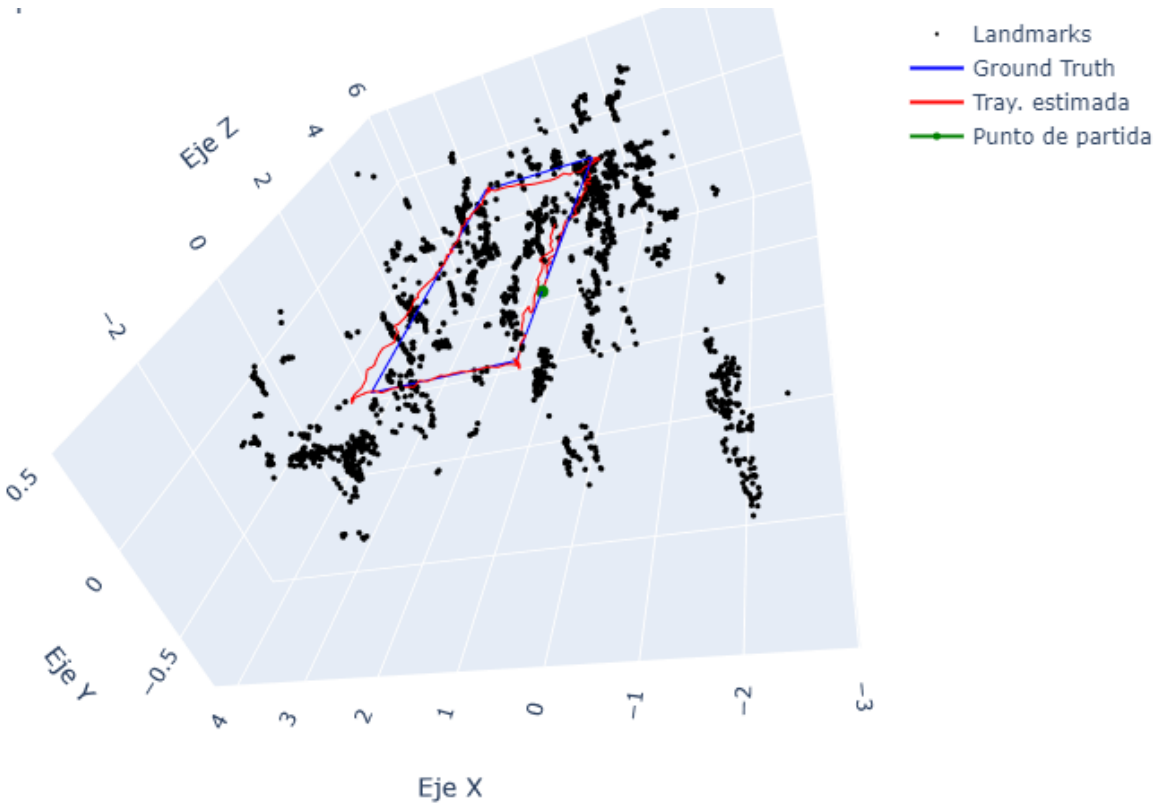

Figura 4.25: Gráfica del Mapa estimado por RFS-SLAM, en donde los *landmarks* son representados por puntos de color negro, *Ground Truth* en azul, la trayectoria estimada por RFS-SLAM en rojo, y el Punto de partida de la cámara (definido a la vez como el origen global del mapa) en verde.

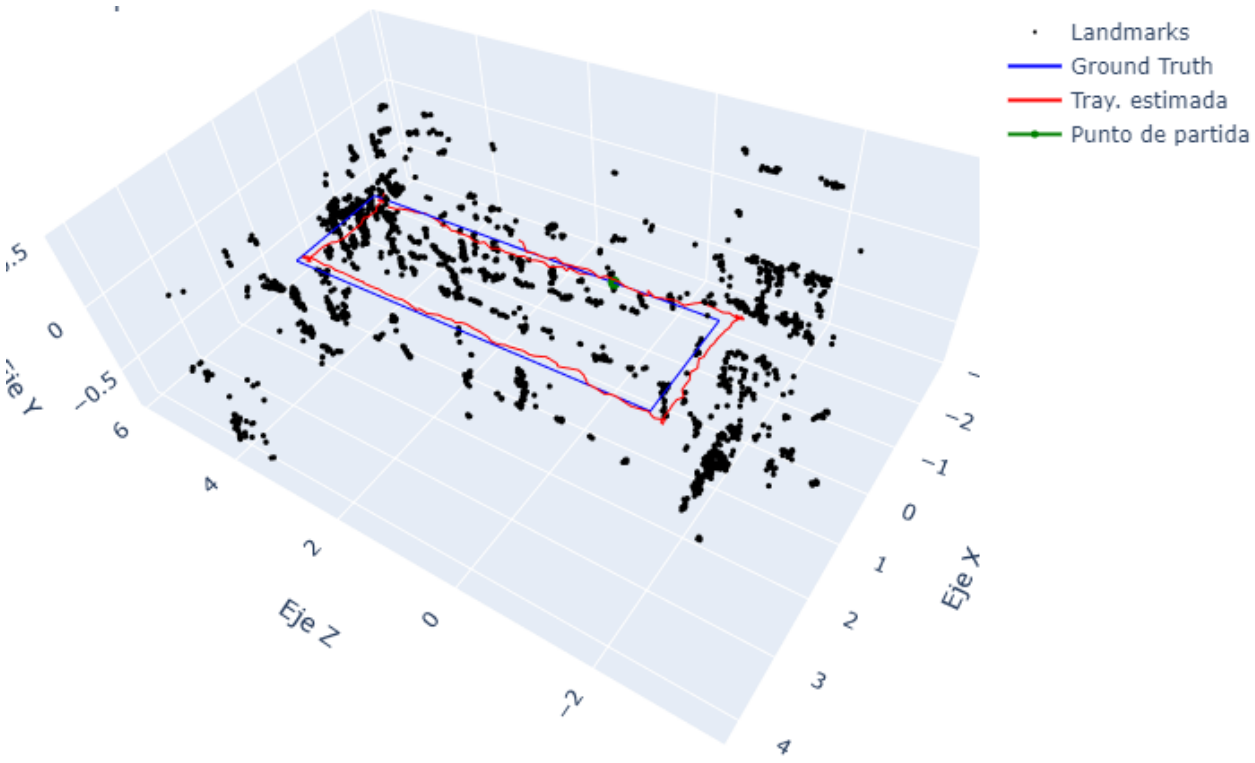

Figura 4.26: Gráfica del Mapa estimado por RFS-SLAM, en donde los *landmarks* son representados por puntos de color negro, *Ground Truth* en azul, la trayectoria estimada por RFS-SLAM en rojo, y el Punto de partida de la cámara (definido a la vez como el origen global del mapa) en verde.

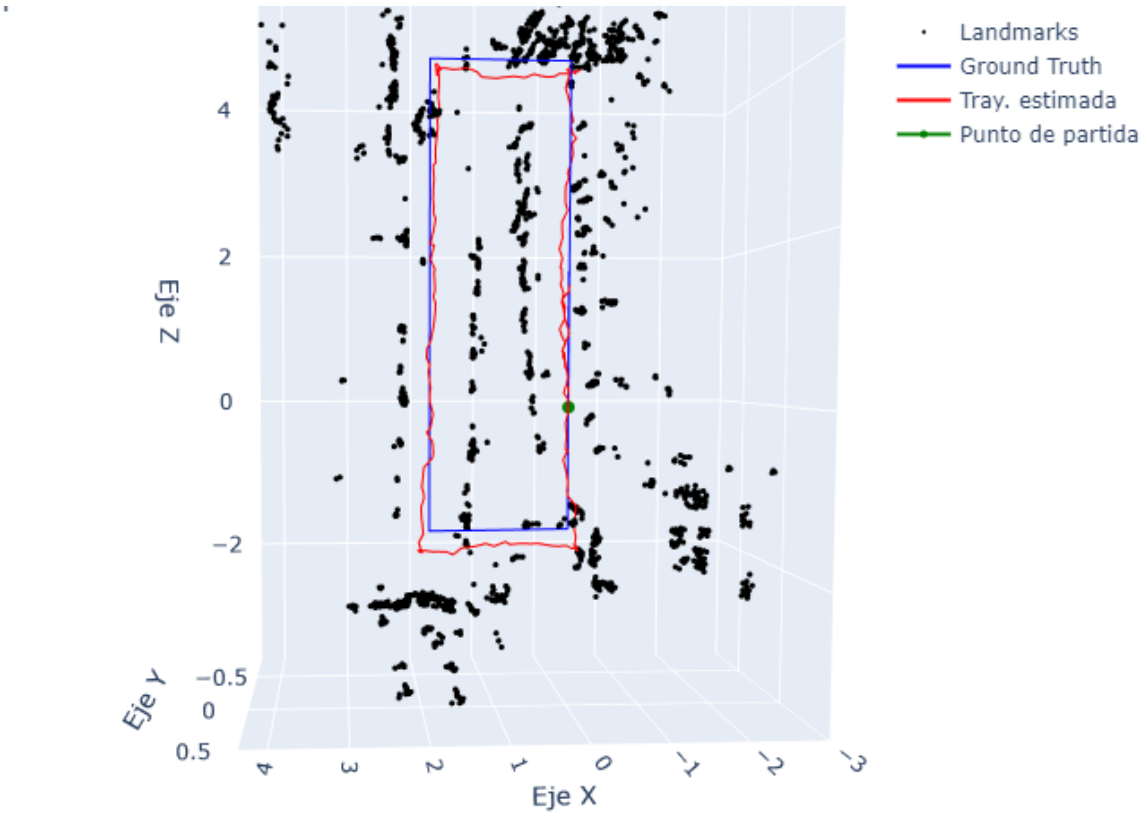

Figura 4.27: Gráfica del Mapa estimado por RFS-SLAM, en donde los *landmarks* son representados por puntos de color negro, *Ground Truth* en azul, la trayectoria estimada por RFS-SLAM en rojo, y el Punto de partida de la cámara (definido a la vez como el origen global del mapa) en verde.

Se registró un total de 14798 *landmarks*.

4.3. Resultados para diferente cantidad de partículas

Por último, al variar la cantidad de partículas, se presentan los siguientes gráficos:

4.3.1. 200 partículas

4.3.1.1. Trayectorias

La Figura 4.28 ilustra las tres trayectorias principales (*Ground Truth*, comandos de control y trayectoria estimada por RFS-SLAM), al ocupar 200 partículas. A su vez, las Figuras 4.29, 4.30 y 4.31 representan de manera gráfica los valores de dichas trayectorias en los ejes X, Y y Z respectivamente.

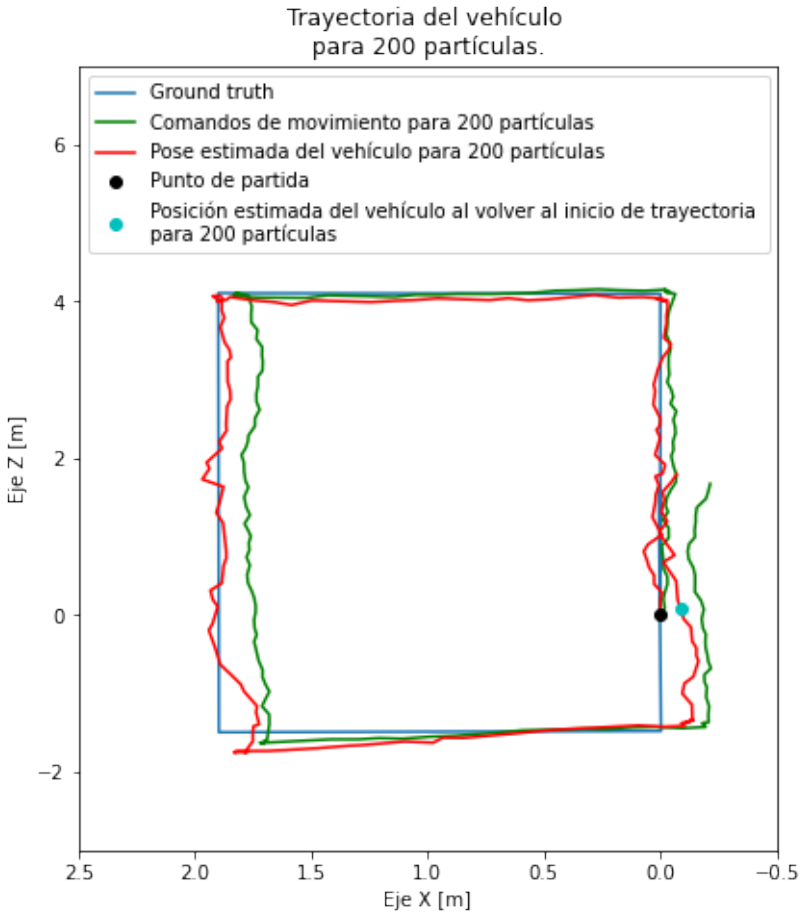

Figura 4.28: Comparación de *Ground Truth* (azul), comandos de movimiento (verde), y trayectoria estimada (rojo), a lo largo de los ejes X y Z al ocupar 200 partículas. Se muestra además en negro el punto de partida de la cámara, y en celeste la posición estimada de la cámara al retornar a dicho punto.

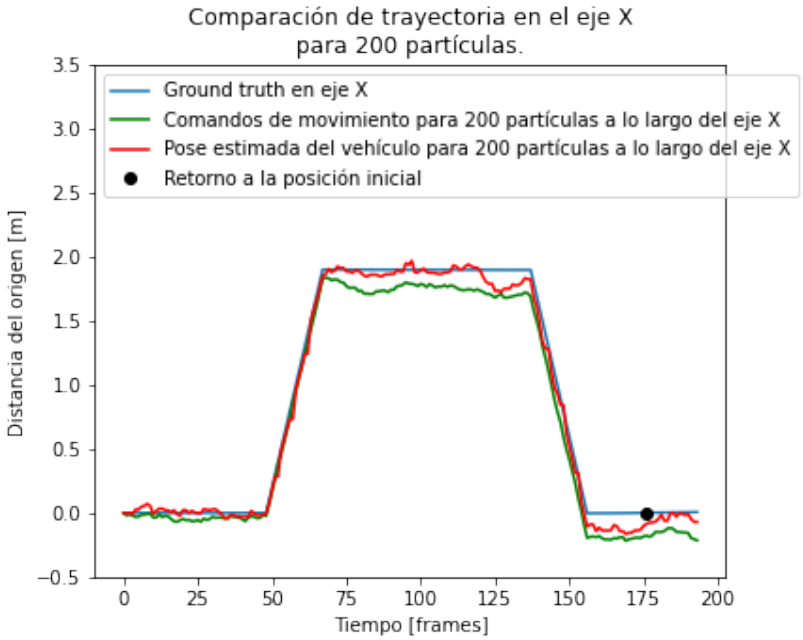

Figura 4.29: Comparación de *Ground Truth* (azul), comandos de movimiento (verde), y trayectoria estimada (rojo), a lo largo del eje X al ocupar 200 partículas. Se muestra además en negro la posición ideal de la cámara en dicho eje al momento de retornar al punto de origen.

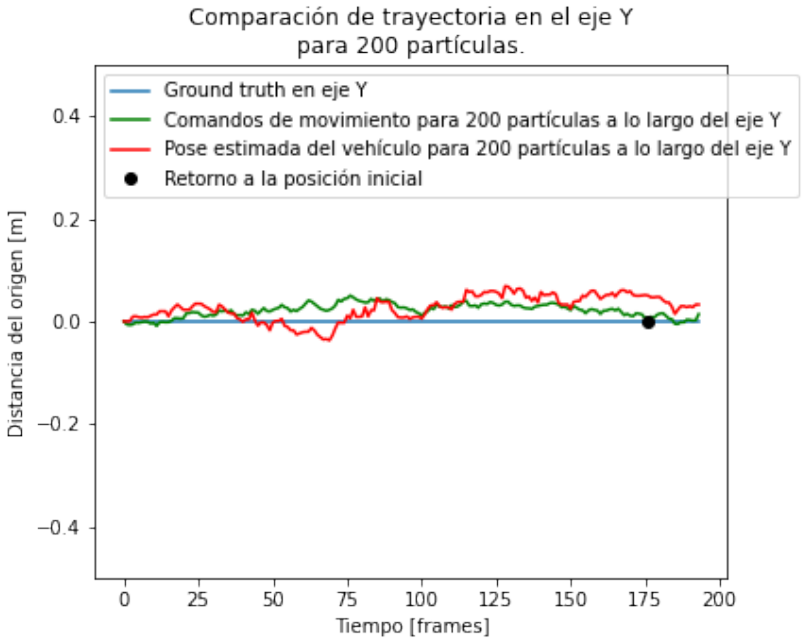

Figura 4.30: Comparación de *Ground Truth* (azul), comandos de movimiento (verde), y trayectoria estimada (rojo), a lo largo del eje Y al ocupar 200 partículas. Se muestra además en negro la posición ideal de la cámara en dicho eje al momento de retornar al punto de origen.

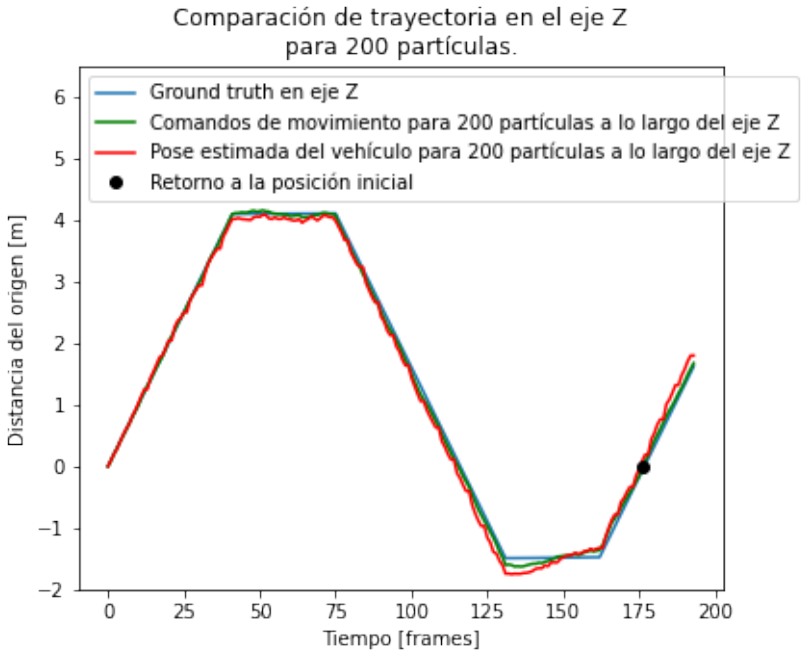

Figura 4.31: Comparación de *Ground Truth* (azul), comandos de movimiento (verde), y trayectoria estimada (rojo), a lo largo del eje Z al ocupar 200 partículas. Se muestra además en negro la posición ideal de la cámara en dicho eje al momento de retornar al punto de origen.

4.3.1.2. Estimaciones de error

Se tiene a continuación en la Figura 4.32 la estimación de error en distancia para cada uno de los ejes tanto del *dead reckoning* como de la trayectoria estimada:

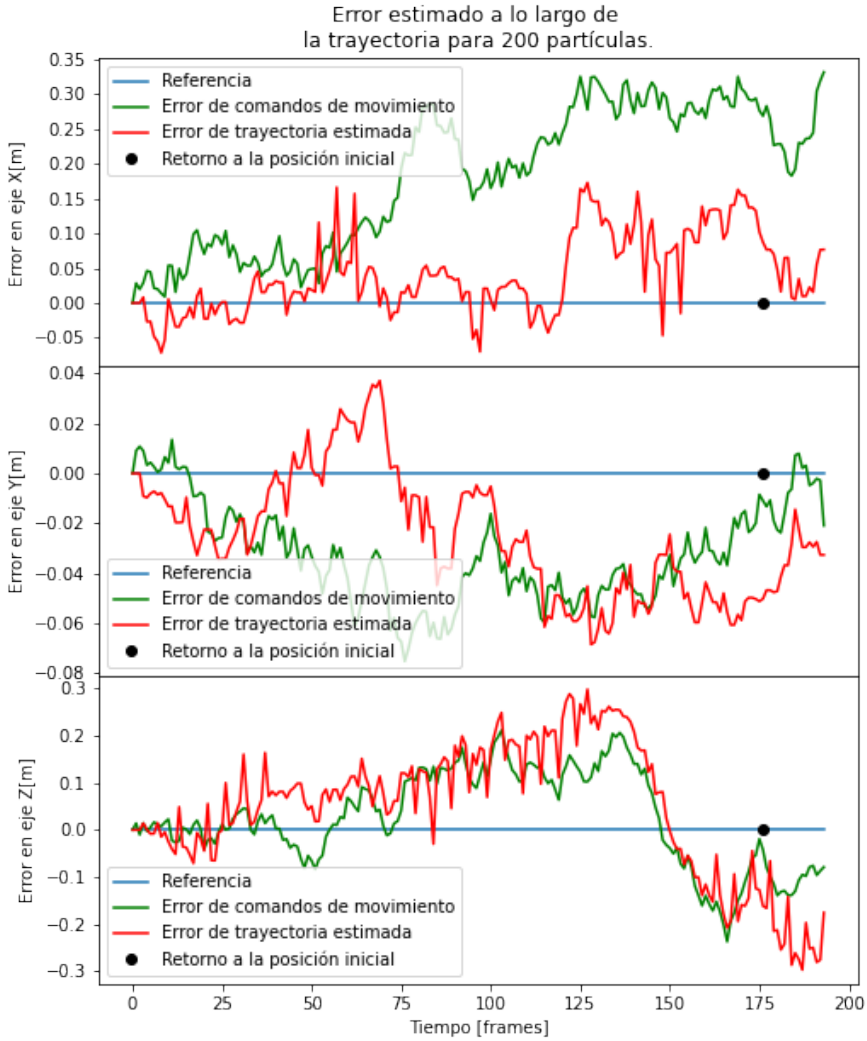

Figura 4.32: Errores a lo largo de la trayectoria en cada uno de los ejes cartesianos al ocupar 200 partículas. Se observa que la tendencia de error de la trayectoria estimada se encuentra por debajo de los comandos de control de la cámara de manera predominante con la excepción del eje Z.

Por otra parte, en la Figura 4.33 se presenta la estimación de error de los cuaterniones:

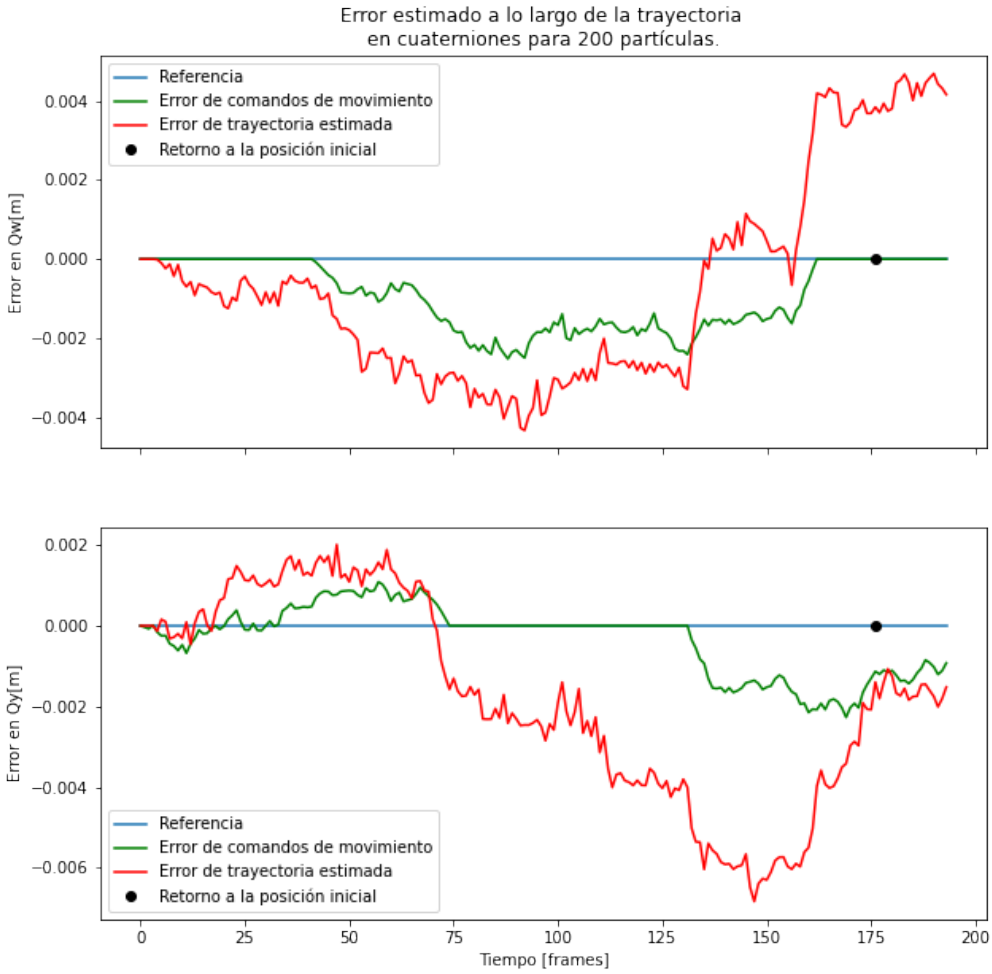

Figura 4.33: Errores a lo largo de la trayectoria para Q_w y Q_y al ocupar 200 partículas. Se observa una diferencia moderada de la trayectoria estimada respecto a los comandos de control.

4.3.2. 20 partículas

4.3.2.1. Trayectorias

La Figura 4.34 ilustra las tres trayectorias principales (*Ground Truth*, comandos de control y trayectoria estimada por RFS-SLAM), al ocupar 20 partículas. A su vez, las Figuras 4.35, 4.36 y 4.37 representan de manera gráfica los valores de dichas trayectorias en los ejes X, Y y Z respectivamente.

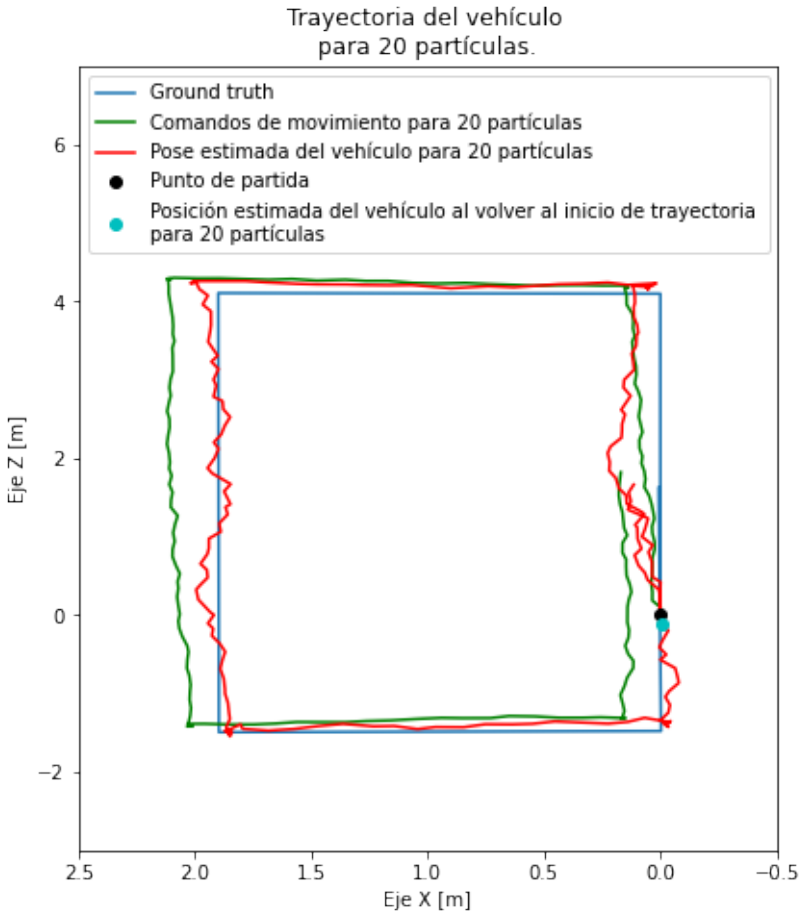

Figura 4.34: Comparación de *Ground Truth* (azul), comandos de movimiento (verde), y trayectoria estimada (rojo), a lo largo de los ejes X y Z al ocupar 20 partículas. Se muestra además en negro el punto de partida de la cámara, y en celeste la posición estimada de la cámara al retornar a dicho punto.

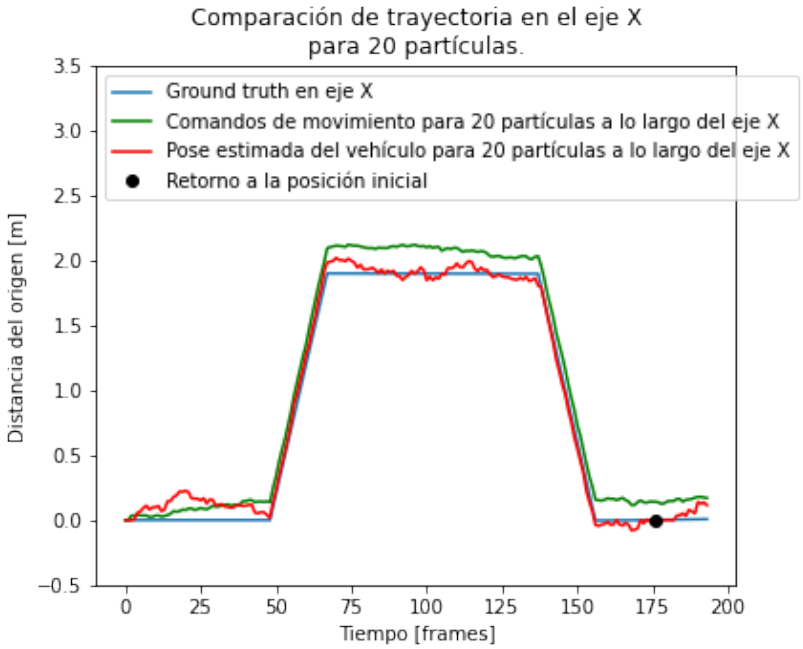

Figura 4.35: Comparación de *Ground Truth* (azul), comandos de movimiento (verde), y trayectoria estimada (rojo), a lo largo del eje X al ocupar 20 partículas. Se muestra además en negro la posición ideal de la cámara en dicho eje al momento de retornar al punto de origen.

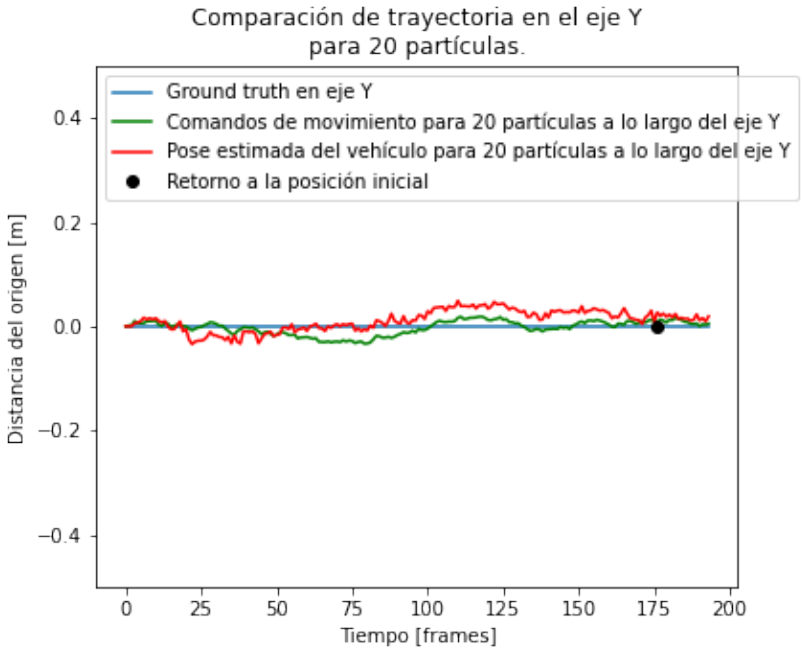

Figura 4.36: Comparación de *Ground Truth* (azul), comandos de movimiento (verde), y trayectoria estimada (rojo), a lo largo del eje Y al ocupar 20 partículas. Se muestra además en negro la posición ideal de la cámara en dicho eje al momento de retornar al punto de origen.

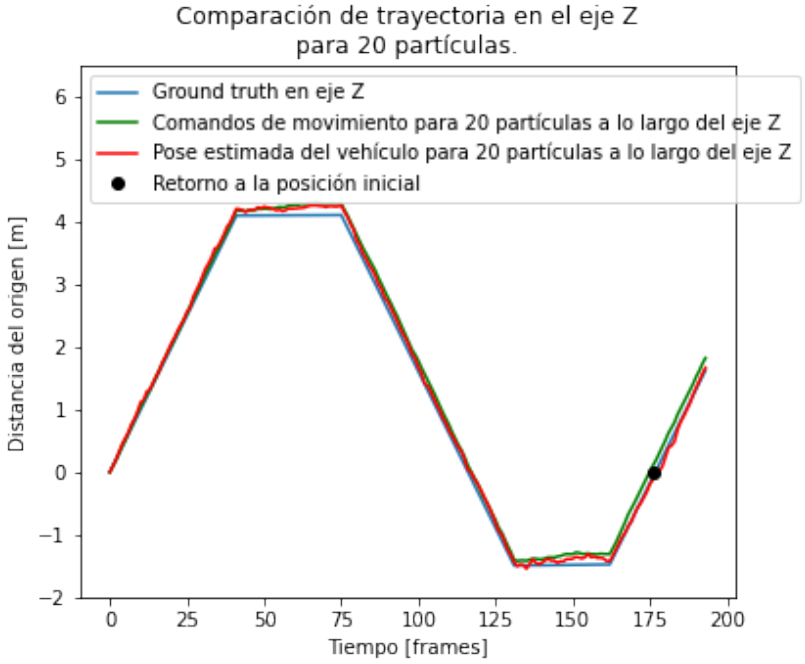

Figura 4.37: Comparación de *Ground Truth* (azul), comandos de movimiento (verde), y trayectoria estimada (rojo), a lo largo del eje Z al ocupar 20 partículas. Se muestra además en negro la posición ideal de la cámara en dicho eje al momento de retornar al punto de origen.

4.3.2.2. Estimaciones de error

Se tiene a continuación en la Figura 4.23 la estimación de error en distancia para cada uno de los ejes tanto del *dead reckoning* como de la trayectoria estimada:

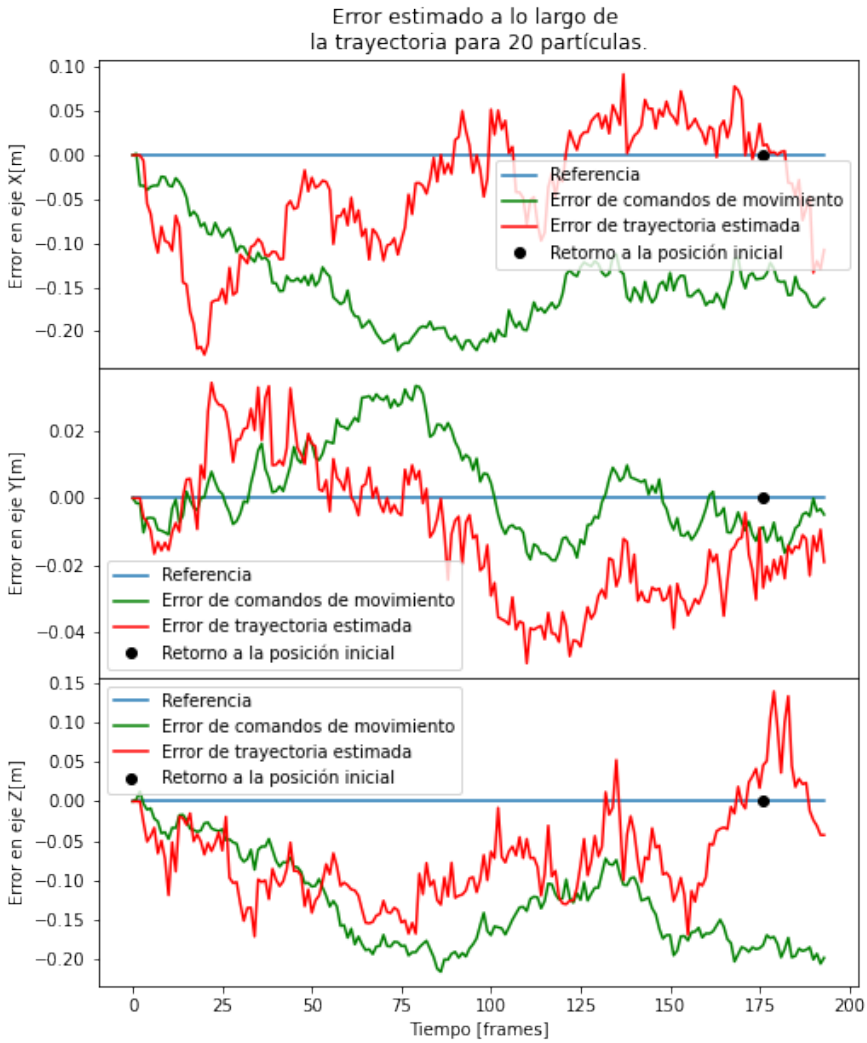

Figura 4.38: Errores a lo largo de la trayectoria en cada uno de los ejes cartesianos al ocupar 20 partículas. Se observa que la tendencia de error de la trayectoria estimada se encuentra por debajo de los comandos de control de la cámara de manera predominante con la excepción del eje Y.

Por otra parte, en la Figura 4.24 se presenta la estimación de error de los cuaterniones:

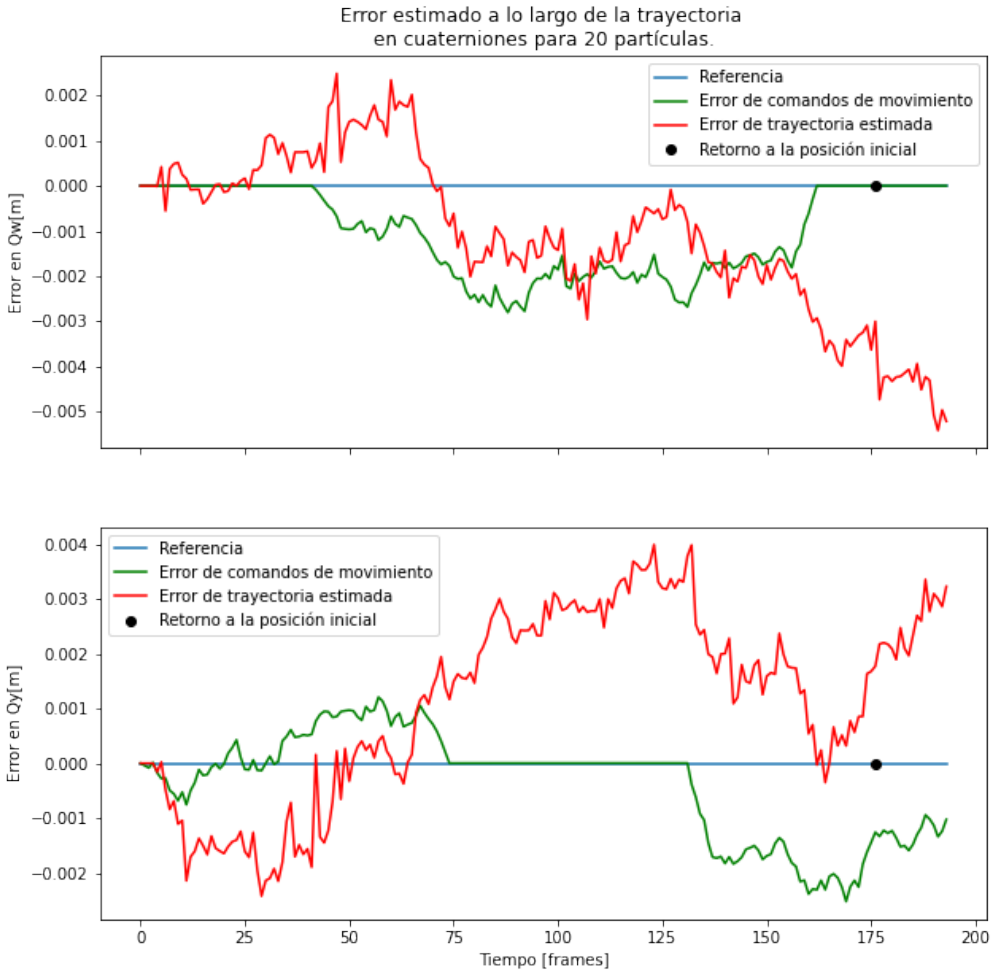

Figura 4.39: Errores a lo largo de la trayectoria para Q_w y Q_y al ocupar 20 partículas. Se observa una diferencia moderada de la trayectoria estimada respecto a los comandos de control.

Capítulo 5

Discusión

En la Figura 4.1, se aprecia que el algoritmo realiza una estimación razonable de la ruta ideal en el eje X y Z. Esto se observa en mayor detalle en las Figuras 4.2, 4.4 y 4.5, donde se distingue que la diferencia entre *Ground Truth* y la estimación de RFS-SLAM es bastante baja, además que dicho error es inferior al de los comandos de movimiento en prácticamente la totalidad del recorrido para dichos ejes.

No obstante, lo anterior no se cumple en el eje Y, donde se aprecia en la Figura 4.3 que presenta una desviación notoria respecto al movimiento ideal. Esto se refuerza en la Figura 4.5, donde se observa que, al contrario de los ejes X y Z, el error en el eje Y resulta ser incluso mayor que los comandos de movimiento; además, la Figura 4.6 permite observar que en los cuaterniones se presenta una mayor diferencia de los valores en la trayectoria estimada con respecto a los comandos de movimiento.

Al momento de aumentar en 1.5 veces la covarianza de movimiento, es posible examinar en las Figuras 4.10, 4.11, 4.12, 4.13, 4.14, y 4.15 un comportamiento ligeramente distinto al caso inicial en los ejes Y y Z: el error de estimación de trayectoria en el eje Y se encuentra por debajo del error de los comandos de movimiento, mientras que para el eje Z se presenta una notoria desviación en la primera curva, sobreestimando la distancia, y resultando así un mayor error en dicho eje; no obstante, este se atenúa una vez se vuelve a desplazar a lo largo de dicho eje. Al igual que en el caso base, hay una mejor estimación de trayectoria en eje X respecto a los comandos de movimiento de movimiento, y resultados levemente peores en cuaterniones.

Posteriormente, en los experimentos con 2 veces la covarianza de movimiento original, se observa que el desempeño de la trayectoria estimada es, de manera general, superior a los comandos de movimiento en los ejes X, Y y Z, tal como se observa en las Figuras 4.19, 4.20, 4.21, 4.22, y 4.23; no obstante, la Figura 4.24 refleja que no se presenta un mejor rendimiento en los cuaterniones.

Observando los resultados al variar la cantidad de partículas, se puede concluir que, a medida que se disminuye el número de partículas, el error acumulado en la trayectoria aumenta ligeramente, siendo esto consistente con lo esperado teóricamente. Esto se observa en las Figuras 4.32 y 4.38. A pesar de lo anterior, en ambos casos ($p = 200$ y $p = 20$) el desempeño de la estimación en los ejes X, Y y Z resulta ser iguales o mejor que los coman-

dos de control; no obstante, se continúa la tendencia de un error mayor para los cuaterniones.

A pesar de los resultados obtenidos en ciertos ejes y en los cuaterniones en los casos presentados, es posible concluir que **el algoritmo estima de manera adecuada la trayectoria del caso de estudio**, estando dicha estimación más cercana a la ruta ideal que los comandos de movimiento incluso al aumentar la incertidumbre de movimiento. Además, se distingue que, al disminuir la cantidad de partículas el error acumulado aumenta, permitiendo esto inferir que al aumentar la cantidad de partículas se presentará un mejor desempeño a lo largo de la trayectoria. Se concluye así que los resultados obtenidos son consistentes con lo esperado teóricamente.

Respecto al mapa, se observa en las Figuras 4.7, 4.8, 4.9, 4.16, 4.17, 4.18, 4.25, 4.26 y 4.27 que se genera con éxito un mapa de acuerdo a la trayectoria, siendo ejemplos la existencia de ciertos objetos sobre la mesa (el conjunto de puntos dentro del rectángulo dado por la trayectoria), y las paredes momentos antes de realizar un giro en 90°. Por otra parte, si bien se contaba con las métricas OSPA y COLA en la librería de RFS-SLAM, estas recibian *landmarks* en 2D, por lo que se requería adaptar estas al uso en 3D. Esto no se logró realizar, por lo que no fue posible analizar dichos mapas mediante esas métricas.

Inicialmente, se presentaron problemas al programar el FoV, lo que generaba que los *landmarks* estimados desaparecieran del mapa una vez la cámara dejaba de tomar mediciones de ellos; así, se desencadenaba una mayor imprecisión a lo largo de la trayectoria y empeoraba el desempeño del algoritmo; no obstante, se logró corregir eso, mejorando notoriamente el desempeño al poder contar con la información del mapa para la estimación de la ruta.

Otro elemento importante a destacar es la base de datos: se observó que la poca cantidad de imágenes (195) ocasionaba algunos problemas al momento de propagar la información a lo largo de la trayectoria. Esto se presenta en mayor medida en las curvas, dado que ahí la cantidad de fotogramas no era la suficiente para poder interpolar información entre fotogramas. Además, el entorno donde se grabó cuenta con alta reflectividad, generando así algunos errores de medición en ciertos casos; en adición, la altura a la que se posicionó la cámara es tal que, en ciertas ocasiones, la mesa limita la visión de la cámara, por lo que no es capaz de detectar *landmarks* vistos con anterioridad que están a rango del FoV, empeorando el desempeño del algoritmo; más aún, debido a que se tiene una vista parcial, resulta difícil realizar un seguimiento de *landmarks* vistos a distancia si estos se encuentran en la periferia del FoV.

Lo anterior limita el alcance de los resultados del proyecto, ya que, si bien se logró obtener estimaciones razonables de la trayectoria, la base de datos es poco flexible y no cuenta con las óptimas condiciones para poder estimar de manera precisa la trayectoria; además, como la base de datos se realizó en un circuito rectangular y estrecho, sus resultados quedan acotados a entornos similares, ya que no es posible determinar si el algoritmo logrará un desempeño similar tanto en circuitos de mayor complejidad, como en ambientes más abiertos. Con lo anterior en mente, se postula que, al rehacer la base de datos con mayor cantidad de fotogramas, o al procesar el algoritmo con una base de datos existente cuya calidad sea mayor a la utilizada en este trabajo, será posible contar con un desempeño de RFS-SLAM que entregue una estimación aún más certera de la *Ground Truth*.

Capítulo 6

Conclusiones

Se cumplieron los objetivos planteados respecto a la **captura y procesamiento de imágenes**: Se calibró adecuadamente la cámara RGB-D a utilizar, y se logró recopilar una base de datos pertinente, además de extraer características visuales y sus mediciones correspondientes respecto a objetos de interés. Se integró posteriormente estas mediciones al algoritmo RFS-SLAM en conjunto con las direcciones de movimiento para cada instante de tiempo, ejecutando así el algoritmo.

La trayectoria estimada es similar a la trayectoria definida para la base de datos, además de presentar una menor desviación que los comandos de movimiento; a su vez, esta tendencia se mantiene al aumentar tanto la covarianza de movimiento como la cantidad de partículas. En adición, el diseño y posterior ejecución del FoV contribuyeron a que se lograra conformar adecuadamente el mapa. De esta manera, se considera la implementación de RFS-SLAM utilizando mediciones obtenidas a partir de una cámara RGB-D como lograda.

A pesar de haberse ejecutado el algoritmo RFS-SLAM de acuerdo a lo planteado en los objetivos, se diagnosticó que la base de datos posee ciertas falencias que frenan el desempeño del algoritmo, acotando su alcance respecto a la variedad de escenarios en los que se puede aplicar. Se espera que, una vez se disponga de una base de datos de mayor complejidad, el desempeño del algoritmo entregue resultados aún más consistentes.

A pesar de las limitaciones observadas, los resultados permiten concluir que RFS-SLAM resulta ser más robusto que los comandos de control, incluso al aumentar la incertidumbre de movimiento, por lo que la hipótesis de estudio se logró comprobar para el caso de estudio.

Bibliografía

[1] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte & M. Csorba, "A solution to the simultaneous localization and map building (SLAM) problem", en *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 229-241, June 2001, doi: 10.1109/70.938381. [En línea]. Disponible en https://ieeexplore.ieee.org/document/938381.

[2] M. D. Adams. "EL7031: Part II - Sensors in Engineering Applications". Diapositivas preparadas para el curso EL7031 - Robotics, Sensing & Autonomous Systems. Universidad de Chile. Otoño 2021.

[3] M. D. Adams. "EL7031: Part IV - SLAM & the Extended Kalman Filter". Diapositivas preparadas para el curso EL7031 - Robotics, Sensing & Autonomous Systems. Universidad de Chile. Otoño 2021.

[4] R. Mur-Artal, J. M. M. Montiel & J. D. Tardós, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System", en *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147-1163, Oct. 2015, doi: 10.1109/TRO.2015.2463671. [En línea]. Disponible en https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7219438.

[5] M. Labbé & F. Michaud, "RTAB-Map as an Open-Source Lidar and Visual SLAM Library for Large-Scale and Long-Term Online Operation," Journal of Field Robotics, 2019. [En línea]. Disponible en https://introlab.3it.usherbrooke.ca/mediawiki-introlab/index.php/RTAB-Map#Publications.

[6] J. Mullane, B. Vo, M. D. Adams & B. Vo, "A Random-Finite-Set Approach to Bayesian SLAM", en *IEEE Transactions on Robotics*, vol. 27, no. 2, pp. 268-282, April 2011, doi: 10.1109/TRO.2010.2101370. [En línea]. Disponible en https://ieeexplore.ieee.org/document/5710428.

[7] A. Falchetti P., "Random Finite Sets in Visual SLAM". [En línea]. Disponible en https://repositorio.uchile.cl/handle/2250/144603.

[8] B. Hampton, A. Al-Hourani, B. Ristic & B. Moran, "RFS-SLAM robot: An experimental platform for RFS based occupancy-grid SLAM," 2017 20th International Conference on Information Fusion (Fusion). [En línea]. Disponible en https://ieeexplore.ieee.org/abstract/document/8009744.

[9] B. Siciliano, & O. Khatib (Eds.), "Springer Handbook of Robotics", pp. 1153-1175.

[10] S. Thrun; W. Burgard. & D. Fox, "Probabilistic robotics" , MIT Press, Cambridge.

[11] M. I. Ribeiro. "Kalman and Extended Kalman Filters: Concept, Derivation and Properties". Institute for Systems and Robotics. [En línea]. Disponible en http://users.isr.ist.utl.pt/~mir/pub/kalman.pdf.

[12] M. Montemerlo, S. Thrun, D. Koller, & B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem", en *Proc. AAAI National Conf. Artificial Intelligence*, 2004, pp. 593–598.

[13] N. Mohamad Yatim, & N. Buniyamin, "Particle Filter in Simultaneous Localization and Mapping (SLAM) using differential drive mobile robot". Jurnal Teknologi, 77(20).

[14] G. Grisetti, R. Kümmerle, C. Stachniss & W. Burgard, "A Tutorial on Graph-Based SLAM," en IEEE Intelligent Transportation Systems Magazine, vol. 2, no. 4, pp. 31-43. [En línea]. Disponible en https://ieeexplore.ieee.org/document/5681215.

[15] M. Budge & S. German, "Basic Radar Analysis", Second Edition, Artech, 2020.

[16] C. Wolff, "Radar Tutorial" [Sitio Web]. https://www.radartutorial.eu/

[17] L. Rosenthaler, F. Heitger, O. Kübler, R. von der Heydt, "Detection of general edges and keypoints", en Sandini G. (eds) *Computer Vision — ECCV'92*. ECCV 1992. Lecture Notes in Computer Science, vol 588. Springer, Berlin, Heidelberg. [En línea]. Disponible en https://link.springer.com/chapter/10.1007/3-540-55426-2_10.

[18] M. Calonder, V. Lepetit, C. Strecha, P. Fua, "BRIEF: Binary Robust Independent Elementary Features", en Daniilidis K., Maragos P., Paragios N. (eds) Computer Vision – ECCV 2010. ECCV 2010. Lecture Notes in Computer Science, vol 6314. Springer, Berlin, Heidelberg. [En línea]. Disponible en https://www.cs.ubc.ca/~lowe/525/papers/calonder_eccv10.pdf

[19] E. Rosten & T. Drummond, "Machine learning for high speed corner detection", en *9th Euproean Conference on Computer Vision*, vol. 1, 2006, pp. 430–443. [En línea]. Disponible en https://link.springer.com/chapter/10.1007/11744023_34

[20] E. Rublee, V. Rabaud, K. Konolige & G. Bradski. "ORB: an efficient alternative to SIFT or SURF". 2011 International Conference on Computer Vision, 2011, pp. 2564-2571, doi: 10.1109/ICCV.2011.6126544. [En línea]. Disponible en https://ieeexplore.ieee.org/document/6126544.

[21] C. Harris & M. Stephens, "A Combined Corner and Edge Detector", *Alvey Vision Conference*. Vol. 15, 1988.

[22] P. Rosin, "Measuring Corner Properties", *Computer Vision and Image Understanding*, 1999.

[23] H. Wei, L. Wu, H. Song & Y. Wei, "RBRIEF: a robust descriptor based on random binary comparisons". IET Comput. Vis. 7, 2013, pp 29-35.

[24] A. Falchetti & M.Adams, "Probability Hypothesis Density Filter Visual Simultaneous Localization and Mapping". *2021 International Conference on Control, Automation and Information Sciences (ICCAIS)*, 2021, pp. 879-886.

[25] I. R. Goodman, Ronald P. S. Mahle, Hung T. Nguyen. "Finite-Set Statistics", en *Mathematics of Data Fusion. Theory and Decision Library (Series B: Mathematical and Statistical Methods)*, vol 37. Springer, Dordrecht. [En línea]. Disponible en https://doi.org/10.1007/978-94-015-8929-1_5.

[26] M. Adams, B. Vo, R. Mahler & J. Mullane, "SLAM Gets a PHD: New Concepts in Map Estimation", en *IEEE Robotics & Automation Magazine*, vol. 21, no. 2, pp. 26-37, June

2014, doi: 10.1109/MRA.2014.2304111. [En línea]. Disponible en https://ieeexplore.ieee.org/document/6814323.

[27] K. Leung, F. Inostroza & M. Adams, "Multifeature-based importance weighting for the PHD SLAM filter," en *IEEE Transactions on Aerospace and Electronic Systems*, vol. 52, no. 6, pp. 2697-2714, Diciembre 2016.

[28] K. Leung, F. Inostroza. "RFS-SLAM: A C++ Library for Simultaneous Localization and Mapping using Random Finite Sets". [En línea]. Disponible en https://github.com/kykleung/RFS-SLAM.

[29] OpenCV. [Sitio web]. https://opencv.org/

[30] Y. Choubik & A. Mahmoudi, "Machine learning for real time poses classification using Kinect skeleton data", en *2016 13th International Conference on Computer Graphics, Imaging and Visualization (CGiV)*. [En línea]. Disponible en https://ieeexplore.ieee.org/document/7467728

[31] O. Bailo, R. Rameau et al. "Efficient adaptive non-maximal suppression algorithms for homogeneous spatial keypoint distribution", in *Pattern Recognition Letters, Volume 106, 2018*. [En línea]. Disponible en https://doi.org/10.1016/j.patrec.2018.02.020

[32] J. Montenegro, "Desarrollo de una metodología computacional para la captura automática de imágenes 4D del rostro usando un escáner 3D". [En línea]. Disponible en https://www.tamps.cinvestav.mx/~wgomez/documentos/JMJ2013.pdf

[33] Z. Cai, J. Han, L. Liu, et al., "RGB-D datasets using microsoft kinect or similar sensors: a survey", Multimed Tools Appl 76, 4313–4355, 2017. [En línea]. Disponible en https://doi.org/10.1007/s11042-016-3374-6

[34] D. Schumacher, B.T. Vo, & B.N. Vo. "A consistent metric for performance evaluation of multi-object filters". *IEEE Transactions on Signal Processing, 86(8):3447–3457, 2008*. [En línea]. Disponible en https://ieeexplore.ieee.org/document/4567674

[35] P. Barrios, G. Naqvi, M. D. Adams, K. Y. Leung, & F. inostroza, "The Cardinalized Optimal Linear Assignment (COLA) metric for multi-object error evaluation". 2015. 18th International Conference on Information Fusion (Fusion), 271-279. [En Línea]. Disponible en https://www.semanticscholar.org/paper/The-Cardinalized-Optimal-Linear-Assignment-(COLA)-Barrios-Naqvi/130d1482c271b35796312e2e52799bd7b461f6f7