Universidad de
Las Palmas de Gran Canaria

Facultad de Informática

Evaluación y Prueba de Algoritmos de Construcción de Mapas para un Robot Móvil

Proyecto Final de Carrera de:
Samuel Artiles Artiles

Tutorizado por:
D. Antonio Carlos Domínguez Brito
D. José Daniel Hernández Sosa

Enero de 2010

A mis padres.

Agradecimientos

Quisiera agradecer de una forma sincera y especial a mis padres, Juan y Antonia, por el apoyo que me han brindado siempre, y por la insistencia continua para que estudiase y terminara por fin mis estudios. A mi hermana y mi cuñado, Saray y Aridane, por su colaboración permanente en las cosas más diversas que hacen un poco más fácil el camino. A mi novia, Isabel, por su compañía en esas largas noches dedicadas al proyecto final de carrera y ayudarme a coger fuerzas en tantos momentos en los que no veía el final. A mi sobrino Juan Antonio y a Juan Pablo, por esos momentos de distracción y alegría en los que volvía a disfrutar de la inocencia y simplicidad de la infancia.

No quisiera quitar mérito alguno a mis tutores, D. Antonio Carlos Domínguez Brito y D. José Daniel Hernández Sosa, por su permanente ayuda a lo largo del proceso de realización del proyecto, y por ello les agradezco inmensamente su apoyo.

También me gustaría dar las gracias a todos mis compañeros, que han hecho que estos años de carrera se hayan grabado a fuego en mis recuerdos, dejando un agradable sentimiento. Dar las gracias a mi amigo y compañero de fatigas, D. Gregorio Mena Rodríguez, porque su compañía a lo largo de estos años ha hecho que el camino sea más fácil, y haya contribuido en cierta forma a la culminación de este proyecto.

Por último, y no menos importante, quisiera manifestar mi agradecimiento a todas aquellas personas que han estado en algún momento a mi lado durante todos estos años, dándome aliento y ánimo en los momentos difíciles.

Gracias a todos/as.

Índice general

Índice de figuras

Capítulo 1

Introducción

Los robots son máquinas en las que se integran componentes mecánicos, eléctricos, electrónicos y de comunicaciones, y además, están dotadas de un sistema informático para su control en tiempo real, percepción del entorno y programación [Ne00].

A partir de la década de los ochenta y noventa del siglo pasado, los aspectos relacionados con la robótica denominada móvil, han ido cobrando una importancia creciente. El desarrollo de robots móviles responde a la necesidad de extender el campo de aplicación de la robótica, restringido inicialmente al alcance de una estructura mecánica anclada en uno de sus extremos [Ol01]. Hoy día, los robots pueden desplazarse por el terreno, por el agua o incluso volar libremente.

La robótica móvil estudia a los robots que realizan tareas desplazándose autónomamente en ambientes dinámicos y complejos, es decir, desde espacios interiores como oficinas y casas, hasta exteriores, como espacios terrestres, submarinos y sustentados en el aire. Estas habilidades llevan implícita la capacidad de razonar sobre una representación interna del mundo.

Uno de los problemas más abordados en la literatura de la robótica móvil son los robots que interactúan en espacios interiores, pensados para funcionar en ambientes dinámicos, haciendo tareas de servicio para los humanos: entrega de mensajes o paquetería, guía turístico, recolector de basura, vigilante, etcétera. También en trabajos demasiado peligrosos que pongan en riesgo la vida humana, en ambientes inestables como basureros de desechos tóxicos, o de acceso difícil. La principal desventaja que presentan dichos robots es que la autonomía de movimiento, en entornos semiestructurados, es bastante limitada; entornos que no han sido especialmente preparados para que el

robot pueda operar en ellos, donde se pueden producir eventos inesperados, fluctuaciones en los sensores y situaciones de incertidumbre.

En este contexto, los problemas típicos de la robótica móvil son los relacionados con la navegación del robot en un entorno, tratando de incrementar su autonomía, y limitando en todo lo posible la intervención humana. La autonomía de un robot móvil se basa en el sistema de navegación automática, donde se incluyen tareas de planificación, percepción y control [Ol01]. Se trata de que el robot tenga la suficiente inteligencia como para reaccionar, tomar decisiones y poder realizar un planteamiento inteligente de sus movimientos, basándose en observaciones de su entorno, sin suponer que este entorno es perfectamente conocido.

1.1. La Navegación de Vehículos Autónomos

Una capacidad fundamental requerida por un robot móvil o AGV[1] es la capacidad de conocer su posición. Todas las acciones que un AGV probablemente tome, dependen de ello [Cs97].

Durante bastante tiempo se ha reconocido la importancia de la navegación de un AGV, y en la última década se ha realizado un gran esfuerzo de investigación para alcanzar una navegación plenamente autónoma; la navegación se puede definir como la combinación de tres capacidades fundamentales [Ne00]:

- La localización propia (Problema de la localización)
- Planificación de un camino
- Construcción de un mapa y su interpretación (Problema de la construcción de mapas o Mapping)

El mapa, en este contexto, denota cualquier conocimiento del mundo (entorno) sobre una representación de datos interna. Esta representación no necesariamente tiene que parecerse a un mapa, tal como lo conocemos habitualmente. De hecho, en la práctica puede adoptar múltiples y variados formatos.

La localización denota la capacidad del agente para establecer su propia posición dentro de un marco de referencia, y la planificación de un camino se considera una extensión de la localización, ya que requiere la posición actual

[1] *Autonomous Guided Vehicle*, Vehículo Guiado Autónomo

del agente y la posición donde se encuentra localizado el objetivo, ambos dentro del mismo marco o sistema de referencia.

La construcción del mapa no sólo se basa en un mapa, como se ha descrito antes, sino también en tener la capacidad de localizarse e identificar localizaciones en un determinado marco de referencia, dentro del mismo. El mapa muestra el territorio explorado dentro de dicho marco. Finalmente, para usar el mapa, es preciso disponer de la capacidad para interpretarlo.

1.1.1. Principios de los Sistemas de Localización

En aplicaciones industriales, los robots que se desplazan por un entorno mediante el seguimiento de líneas, son el origen de los sistemas de navegación de los AGV. El seguimiento de líneas es el proceso de dirigir un AGV a lo largo de una ruta inalterable, usando un camino que el AGV puede identificar y seguir. Éste es un método sencillo de navegación, indicado por una línea, el cual requiere unos recursos sensoriales y de procesamiento muy simples.

Los enfoques que dependen del seguimiento de líneas poseen desventajas importantes, las cuales hacen que las capacidades de navegación de un AGV sean sumamente limitadas. La mayor desventaja es que el AGV sólo puede seguir una línea, y ese funcionamiento, por lo tanto, es severamente limitado. Una desventaja adicional es el desconocimiento por parte del AGV de su posición actual en el camino y por tanto su incapacidad para localizarse.

Un problema adicional de las estrategias de seguimiento de líneas es el costo y el tiempo necesario para colocar caminos para el AGV. En algunos casos, una estrategia de seguimiento de líneas o rutas puede ser imposible de implementar, debido a las condiciones del terreno. En otros casos, algunas rutas del AGV pueden necesitar atravesar otras, confundiendo al sensor de seguimiento.

Un método de navegación alternativo, que no dependiera de los problemas descritos anteriormente, implicaría la localización del AGV. La localización difiere del seguimiento de una línea en que éste trata de especificar la posición y orientación del vehículo dentro del entorno, más que hacer pequeñas correcciones en los movimientos para poder seguir la línea. De esta forma, el AGV no está limitado para moverse en una ruta inalterable, al contrario, cualquier posición dentro del entorno está permitida.

Un desarrollo limitado de la localización es posible utilizando odometría o

un INS[2]. Ambos permiten al vehículo localizar, por deducción, su posición a través de sus movimientos.

La odometría[3] es uno de los métodos más ampliamente usados para estimar la posición de un robot. Proporciona una buena precisión a corto plazo y es barata de implantar, sin embargo, la idea fundamental de la odometría es la integración de información incremental del movimiento a lo largo del tiempo, lo cual conlleva una inevitable acumulación de errores. En concreto, la acumulación de errores de orientación, causa grandes errores en la estimación de la posición, los cuales van aumentando proporcionalmente con la distancia recorrida por el robot. A pesar de estas limitaciones, muchos investigadores están de acuerdo en que la odometría es una parte importante del sistema de navegación de un robot, y que debe usarse con medidas del posicionamiento absolutas para proporcionar una estimación de la posición más fiable.

Un INS se basa en la magnitud de aceleración del robot. Mide la fuerza experimentada durante las aceleraciones, convirtiéndola en una posición precisa del robot mediante la integración de las medidas. Al igual que sucede con la odometría, un INS padece de la acumulación de errores durante sus operaciones, y por lo tanto, se generan errores de localización.

1.1.2. Localización basada en Balizas

El primer sistema de localización exitoso empleó balizas artificiales o hitos para determinar la posición del vehículo. Los sistemas de localización basados en balizas, o marcas, dependen de un conjunto de balizas localizadas en posiciones conocidas en el entorno; en este caso, el AGV está equipado con un sensor que puede observar estos hitos o balizas, y el sistema de navegación usa esas observaciones y el conocimiento de las posiciones de los hitos para localizar la posición del vehículo.

A diferencia de la odometría e INS, la exactitud de la localización no se deteriora con el paso del tiempo. Esto es debido a que el sensor en el AGV proporciona observaciones de posición, a diferencia de las observaciones de movimiento que proporcionan el INS y la odometría, y por lo tanto, no

[2] *Inertial Navigation System*, Sistema de Navegación Inercial

[3] Se conoce como *odometría* a las técnicas de posicionamiento que emplean información de sensores propioceptivos (aquéllos que adquieren datos del propio sistema. La propriocepción se refiere a la percepción del estado interno del robot; por ejemplo, medidas de carga de baterías, posición del robot, etc. [Al09]), para obtener una aproximación de la posición real en la que se encuentra un sistema móvil, en un determinado instante, respecto a un sistema de referencia inicial.

está condicionado a la acumulación de errores, debido a la integración de las medidas.

Los sistemas de localización basados en marcas se pueden clasificar acorde al tipo de señal del sensor. Esto es debido a que, según el tipo de señal y las características que presentan los hitos frente a dicha señales, se utilizarán un tipo de balizas u otros, haciendo que éstas sean lo más adecuadas posibles para detectarlas en el entorno de manera sencilla. Además, se pueden clasificar los sistemas acorde al tipo de procesamiento de la información empleado por el sistema, o por el elemento a percibir en el entorno que corresponda.

Los principales tipos de señales usados por los sistemas de localización basados en marcas son: láser infrarrojo, ultrasonidos y radar de ondas milimétricas, siendo los sistemas basados en láser los más extendidos. En los últimos años se han desarrollado además esquemas de localización basados en visión. En ellos se utilizan cámaras para realizar la identificación de las balizas.

Las dos principales categorías de sistemas de localización basados en hitos o marcas pueden ser identificadas según el tipo de procesamiento de la información empleado por el sistema. Estas categorías son las basadas en triangulación y las basadas en estimación.

La triangulación implica combinar distintas líneas de observación para deducir la posición del AGV, suponiendo que la posición y la orientación del vehículo no cambian entre las observaciones. Las observaciones proporcionan limitaciones a la posición del vehículo. El número de limitaciones necesarias para determinar exclusivamente la posición del vehículo depende del número de grados de libertad de la posición. Para la ubicación de un vehículo, definido en términos de la posición (x,y) y una orientación ϕ, son necesarias tres observaciones θ_1, θ_2 y θ_3 para definirla unívocamente. La triangulación es el método más usado para la ubicación de un AGV.

Un enfoque alternativo a la triangulación es realizar una estimación de la localización del vehículo. La estimación depende de un modelo del vehículo y del sensor. La estimación se actualiza cada vez que se hace una nueva observación, ubicándose el vehículo de observación en observación, usando una predicción de la localización del vehículo, basada en las entradas de control del vehículo. Se emplean filtros para integrar las medidas y manejar la incertidumbre, como por ejemplo el filtro de Kalman.

El filtro de Kalman es usado ampliamente para realizar una estimación

probabilística de la localización del AGV, acorde a la Teoría de Estimación Bayesiana, en términos de una estimación media y la covarianza de la estimación; se puede realizar también una estimación no probabilística de la localización del vehículo, por ejemplo, expresando la estimación en términos de regiones limitadas.

Un problema común de los sistemas de localización basados en balizas es que dichos indicadores tienen que ser instalados. Esto puede ser caro y también puede incrementar el tiempo de despliegue del AGV. Un problema añadido del uso de indicadores es que éstos pueden ser dañados u ocultados, comprometiendo el funcionamiento del AGV.

1.1.3. Localización con un Mapa "a priori"

Una mejora significativa del coste de eficacia y flexibilidad de los métodos de localización basados en balizas, se consigue cuando se utilizan hitos naturales como indicadores. Usando balizas naturales del entorno, eliminamos el coste de la instalación de balizas artificiales, y tampoco hay coste de mantenimiento; para ello, es necesario identificar, a través de los sensores, las balizas naturales en las señales recibidas. Partes de una estructura existente en el entorno pueden servir también como un hito natural. La identificación, referida a la obtención de características, ha sido tema de considerable investigación.

Las características pueden a veces proporcionar más información que una alineación y una dirección (orientación). Baeg [Ba95] mide la orientación de características y usa la medida para mejorar la localización del AGV; varios investigadores han examinado también cómo seleccionar hitos para incrementar la exactitud de la localización. Sutherland y Thompson [Su94] argumentan que la ruta del vehículo debería seguir caminos donde es posible la mejor localización, debido a hitos más favorables.

El método ICP[4] considera el problema de la localización como un problema de correspondencia. La observación hecha por los sensores es comparada con el mapa "a priori", con cada iteración se produce una comparación mejor. Zhang identifica dos métodos ICP.

1. "Aproximación basada en primitivas": obtiene un conjunto de primitivas de la observación, tal como líneas y esquinas, y entonces compara la posición de las primitivas obtenidas, con la posición de las primitivas en el mapa "a priori".

[4] *Iterative Closest Position*, Posición Iterativa más Cercana

2. "Aproximación basada en superficie": considera la observación en su totalidad, y trata de compararla con el mapa "a priori", de acuerdo a una medida de correlación.

La estimación de la posición del vehículo usando el filtro de Kalman, ha sido usado también en aproximaciones basadas en hitos naturales. Si los hitos pueden ser identificados, se pueden tratar como indicadores. Los hitos son seleccionados e identificados normalmente a través de primitivas geométricas. Deben ser hitos o balizas que se encuentran en el mapa.

Un tercer enfoque para localizar un AGV usando un mapa "a priori" supone argumentos cualitativos. Los mapas usados son representaciones topológicas del entorno. Bertin [Be93] argumenta que las aproximaciones cualitativas son generalmente más robustas que las aproximaciones cuantitativas, mediante la utilización de mapas topológicos, y no geométricos.

Todos los ejemplos de localización tienen un elemento común importante: el entorno es conocido para el AGV, en términos del mapa "a priori". Como consecuencia, el AGV está limitado a operar dentro del entorno conocido. Añadida a esta restricción está la desventaja de que el entorno ha sido examinado o mapeado antes de que el AGV pueda operar. Si se desea extender el entorno de operación, se deben mapear nuevas áreas. Si el área de operación del AGV es fija, aún podría ser necesario realizar nuevas inspecciones puesto que pueden haber cambiado características en el entorno por el paso del tiempo. Para vencer esos problemas, se ha investigado la posibilidad de que el AGV construya automáticamente un mapa del entorno y lo use para localizarse simultáneamente.

1.1.4. Construir Nuestro Propio Mapa

El siguiente paso surge de forma natural, y consiste en plantear que sea el propio sistema robótico el que construya el mapa de manera automática. Resolviendo este problema podríamos permitir que un AGV pudiese desenvolverse fácilmente en un entorno, sin la necesidad de una preparación inicial ante dicho entorno. El AGV podría también ser flexible, capaz de hacer frente a modificaciones en el entorno, en suma, se convertiría en mucho más autónomo.

La paradoja que surge con este enfoque es que para construir un buen mapa es necesario estar localizado con precisión, lo que sólo es posible si se dispone de un mapa preciso. Este problema es conocido como Localización

y construcción de mapas simultáneos SLAM[5] o, alternativamente, como Construcción de Mapas y Localización Concurrente CML[6] [Th05]. Los problemas SLAM surgen pues cuando el robot no tiene acceso a un mapa del entorno, ni conoce su propia ubicación; se parte simplemente de las medidas generadas por los sensores y los controles aplicados al vehículo. El SLAM es más complejo que la localización en el mapa, cuando es desconocido, y ha de ser estimado a lo largo del trayecto. También es más complejo que la construción de mapas con ubicaciones conocidas, ya que las ubicaciones son desconocidas y han de ser estimadas a lo largo de la ruta.

1.1.5. Revisión de técnicas

Elfes [El90] describe una aproximación de la construcción de mapas basado en rejillas de ocupación, que dividen el entorno en celdas individuales. Inicialmente cada celda es etiquetada como "desconocido". Cuando el AGV explora el entorno, asigna nuevas etiquetas ("vacío" u "ocupado") a las celdas, a medidas que las va "encontrando" (percibiendo). Sin embargo, Kriegman [Kr90] argumenta que la aproximación de rejillas de ocupación no pueden hacer frente a la inherente incertidumbre de la información disponible por el AGV. Un problema adicional de las rejillas de ocupación es que cuanto más grande es el entorno, más rejillas se necesitan, lo que crea un importante demanda de memoria para almacenar la rejilla.

Primitivas geométricas se han usado también para modelar el entorno, por ejemplo por Ayache [Ay89], donde los detalles del mismo son obtenidos de una imagen de vídeo, y usados para modelar el entorno. Ayache usa el filtro de Kalman para estimar la variación de las primitivas, basadas en la variación de las observaciones y la variación de los desplazamientos, entre los orígenes de las observaciones.

Los métodos cualitativos también se han usado para construir mapas. Kuipers [Ku91] describe un algoritmo que define lugares y rutas interconectadas distintivas, y construye un mapa topológico. Sin embargo, no se examina ninguno de los efectos del SLAM. Tabién se puede construir un mapa topológico del entorno, usando redes neuronales. Esto se examina en Morellas [Mo95], pero se ignoran otra vez los efectos del SLAM.

[5] *Simultaneous Localization and Map Building*, Localización y Construcción de Mapas Simultáneos

[6] *Concurrent Mapping and Localization*, Construcción de Mapas y Localización Concurrente

La primera solución coherente del problema del SLAM fue publicada por Smith, Self y Cheeseman [Sm90]. Aunque en ese trabajo no se examinan los efectos del SLAM, su solución proporciona un tratamiento consistente de dichos efectos. Emplean la estimación y el filtro de Kalman, y consideran el problema de la construcción de mapas una extensión del trabajo de localización del AGV. El enfoque es importante porque aplica métodos de estimación rigurosos al problema del SLAM.

El efecto fundamental del SLAM es la introducción de correlaciones entre el error en el vehículo y las estimaciones de las características. Esas correlaciones constituyen la esencia del problema del SLAM.

1.2. Objetivos del Proyecto

El objetivo final de este proyecto es el desarrollo de un entorno Matlab para la prueba y análisis de algoritmos de SLAM con el objetivo de poder realizar un análisis exhaustivo de los mismos tanto cualitativamente como cuantitativamente, con vistas a su implementación en plataformas robóticas móviles.

Este objetivo se plantea a partir de una serie de etapas a cubrir, y son:

- Etapa 1. Estudio y análisis del problema del SLAM.
- Etapa 2. Selección de algoritmos.
- Etapa 3. Realización de pruebas.
- Etapa 4. Análisis de resultados.

1.3. Organización del Documento

Los capítulos que se desarrollan en el resto del documento están organizados de la siguiente manera:

En el capítulo 2 se introduce la formulación matemática del problema de la localización; el problema de determinar la pose de un vehículo dentro de un entorno basado en observaciones de dicho entorno. El capítulo justifica y desarrolla la aproximación teórica de la estimación para el problema,

con la pose del vehículo identificada como un estado que necesita ser estimado.

En el capítulo 3 se examina el problema del SLAM en detalle. Se desarrolla la aproximación de la estimación para el problema y se utiliza para exponer el porqué tantos intentos previos para solucionar el problema no hubiesen salido bien. Se expone que los errores de las estimaciones del estado están correlacionadas, y se examina mejor que las correlaciones son fundamentales para el problema del SLAM.

En el capítulo 4 se estudia la aplicación del filtros estimadores al problema de la localización de la posición de un vehículo y la construcción del mapa del entorno de forma simultánea. Además, se describen de forma teórica los algoritmos que se utilizarán posteriormente en los siguientes capítulos.

El capítulo 5 está destinado a la comparativa de los algoritmos. En dicho capítulo se realizan las simulaciones que permitan una sintonización de los parámetros propios de cada algoritmo, teniendo en cuenta la caracterización de los mapas. Para ello, también se presentan los diferentes mapas a utilizar en las ejecuciones, y se describen a groso modo, las funciones más importantes de los algoritmos empleados.

En el capítulo 6, destinado a las pruebas y evaluación de los algoritmos, podremos comparar los diversos algoritmos de construcción de mapas en entornos diferentes. Se realizará el estudio de las ejecuciones de los algoritmos en los diferentes mapas y el estudio de la eficiencia de cada uno de los algoritmos, utilizando para ello la notación asintótica O.

En el capítulo 7 se detallarán los resultados de las diferentes comparaciones realizadas en los capítulos anteriores, obteniendo al final la decisión del algoritmo que mejor se adapta para los diferentes entornos.

Capítulo 2

El Problema del SLAM

2.1. Introducción

Uno de los principales obstáculos de la fabricación de robots inteligentes útiles y confiables, es la incertidumbre de posicionamiento del robot. Después de todo, los robots móviles no pueden funcionar de manera eficaz si no conocen dónde están [Id06]. El conocimiento preciso del posicionamiento es necesario para crear mapas de alta resolución y lograr una ruta fiable de forma reiterada, para poder conseguir a continuación un comportamiento deliberativo de alto nivel, como pudiera ser la búsqueda sistemática o la vigilancia de un área.

En exteriores, el empleo de GPS[1] ha proporcionado a los robots móviles un apoyo que hace mucho más fácil saber dónde se encuentran en cada momento. En interiores, sin embargo, no se dispone de este recurso. En este caso se desarrollan métodos de localización que usan el muestreo del rango de lecturas de una exploración de los sensores láser y de ultrasonidos para decidir de forma probabilística dónde se encuentra el robot dentro de su modelo interno del mundo.

El problema de la localización de un robot se puede dividir en dos subtareas: estimación de la posición global y seguimiento de la posición local. La estimación de la posición global es la capacidad para determinar la posición del robot en un mapa a priori o un mapa que ha sido previamente aprendido, considerando la información que el robot percibe en algún sitio de la región representada por el mapa. Una vez que la posición del robot ha sido encontrada en el mapa, el seguimiento local es el problema de mantener la pista

[1] *Global Positioning System*, Sistema de Posicionamiento Global

de esa posición en el tiempo. Mientras que las aproximaciones existentes para el seguimiento de la posición pueden estimar la posición de un robot de manera eficiente y con exactitud, típicamente fallan en la localización global de un robot desde el principio o para recuperarse de un fallo de localización. Los métodos de localización globales son menos exactos y a menudo requieren considerablemente mayor rendimiento computacional.

2.2. Definición del problema

La solución al problema de la localización y la construcción del mapa simultáneos (SLAM) ha sido, y aún es, en muchos sentidos, el "máximo exponente" de la comunidad de investigación de vehículos autónomos. La capacidad de colocar un vehículo autónomo en una posición desconocida en un entorno desconocido y además hacer que construya un mapa, usando sólo las observaciones relativas del entorno, y luego usar este mapa de forma simultánea para navegar, haría al robot verdaderamente "autónomo". Así, la principal ventaja del SLAM consiste en que elimina la necesidad de infraestructuras artificiales, o el conocimiento topológico, *a priori*, del ambiente. Una solución al problema del SLAM sería de valor inestimable en un rango de aplicaciones donde la posición absoluta o la información exacta del mapa son imposibles de conseguir, incluyendo, entre otros, la exploración autónoma planetaria, vehículos autónomos submarinos, vehículos autónomos aerotransportados, y vehículos todoterreno autónomos en tareas como la minería y la construcción.

En un proceso típico de construcción del mapa de un entorno [Ro04], se parte de un mapa vacío o desconocido en el que el robot parte de una posición local conocida, típicamente el origen de coordenadas. Comienza el proceso, y el robot es capaz de construir la parte del mapa cercano a su posición, a partir de la información que le proporcionan sus sensores. Como es evidente, debido al limitado rango de los sensores, el robot debe moverse hacia las zonas que quieran ser incluidas en el mapa. Según se va desplazando hacia esas nuevas áreas, observa nuevos objetos presentes en ellas y va actualizando el mapa o modelo espacial [Ja05] del entorno. El ruido de las medidas hace que el mapa no sea exacto, pero a su vez el robot se mueve con desplazamiento de los que sólo se tiene una estimación, con lo que su posición tampoco es exacta. Cuando se realizan observaciones de objetos nuevos, la estimación de la posición de esos objetos depende tanto de la incertidumbre de la posición del robot como de la incertidumbre de las medidas realizadas.

Esta tendencia a que el robot genere sus propios mapas, en lugar de que

el humano le provea uno, viene de la idea de que la percepción humana y la del robot son diferentes, y de este modo, un mapa generado a través de los limitados sensores del robot será mucho más útil que un mapa generado por un individuo con sentidos distintos. Además, si se persigue una verdadera autonomía en el robot, es mejor que éste pueda generar una representación de su ambiente sin ayuda exterior.

2.3. Revisión de soluciones adoptadas

El estudio de soluciones de estimaciones teóricas para el problema del SLAM, dentro de la comunidad robótica, tiene una historia interesante. El trabajo inicial de Smith [Ch86] y Durrant-Whyte [Du88] estableció una base estadística para describir relaciones entre señales y la manipulación de la incertidumbre geométrica. Un elemento clave de este trabajo fue mostrar que debe haber un alto grado de correlación entre las estimaciones de la localización de marcas diferentes en un mapa, y que de hecho estas correlaciones llegarían a la unidad siguiendo observaciones sucesivas.

Al mismo tiempo Ayache y Faugeras [Ay89], y Chatila y Laumond [Ch85] emprendían un primer trabajo en la navegación visual de robots móviles que usan tipos de algoritmos como el filtro de Kalman. La conclusión principal de este trabajo fue doble. Primero, la suposición de correlaciones entre señales en un mapa es importante, si la consistencia del filtro puede ser mantenida. Segundo, que una solución del SLAM completo requiere que un vector de estado conste de todos los estados del modelo del vehículo, y todos los estados de cada marca en el mapa necesita ser mantenida y actualizada siguiendo cada observación, si se requiere una solución completa para el problema del SLAM. La consecuencia de esto en cualquier aplicación real, es que el filtro de Kalman tiene que emplear un vector de estado enorme (del orden del número de marcas mantenidas en el mapa) y esto en general, no es posible computacionalmente.

Del trabajo original de la estimación teórica de Smith, Self y Cheeseman [Sm90], se puede obtener lo siguiente:

- La estructura completa del problema del SLAM depende críticamente de mantener el conocimiento completo de la correlación cruzada entre las estimaciones de los hitos. Minimizar o ignorar las correlaciones cruzadas es precisamente lo contrario a la estructura del problema.

- Mientras que el vehículo progresa en el entorno, los errores en las estimaciones de cualquier par de marcas se vuelven más y más correlacionadas, y de hecho nunca vuelven a correlacionarse menos.
- En el límite, los errores en las estimaciones de cualquier par de marcas se vuelven completamente correlacionadas. Esto significa que dado la localización exacta de cualquier marca, la localización de cualquier otra marca en el mapa se puede determinar también con absoluta certeza.
- Mientras que el vehículo se mueve a través del entorno tomando observaciones de hitos individuales, el error en las estimaciones de la localización relativa entre marcas distintas reduce de forma monótona al punto donde el mapa de localizaciones relativas se conoce con precisión absoluta.
- Mientras que el mapa converge de la manera descrita antes, el error en la localización absoluta de cada hito (y de este modo, del mapa entero) alcanza un límite inferior determinado solamente por el error que existió cuando se hizo la primera observación.

Así, existe una solución al problema general del SLAM, y es de hecho posible construir un mapa perfectamente exacto y simultáneamente calcular las estimaciones de la posición del vehículo sin ningún conocimiento anterior del vehículo o de las localizaciones de las marcas.

En la figura 2.1 [Ja05] se observa un esquema que refleja, de manera más comprensible, las operaciones involucradas en el problema del SLAM, mientras que en la tabla 2.1 se describen las variables utilizadas en el esquema. Para el problema incremental del SLAM, al robot que se encuentra en la posición $x\,(t-1)$ se le aplica un vector de control $u\,(t)$ en el instante *t-1*. El instante t se considera discreto $(1,2,\cdots,n)$. En la nueva posición $x\,(t)$, el robot, utilizando sus sensores de distancia, toma mediciones $z_i\,(t-1)$ a los objetos (m_i) que componen el entorno visible para el robot.

La construcción incremental de mapas impone tres problemas fundamentales:

1. Localizar al robot, utilizando únicamente los sensores de odometría y de distancia.
2. Construir un mapa global dados los puntos de observación.
3. Fusionar la información de los sensores de distancia para distinguir cada objeto que forma parte del entorno.

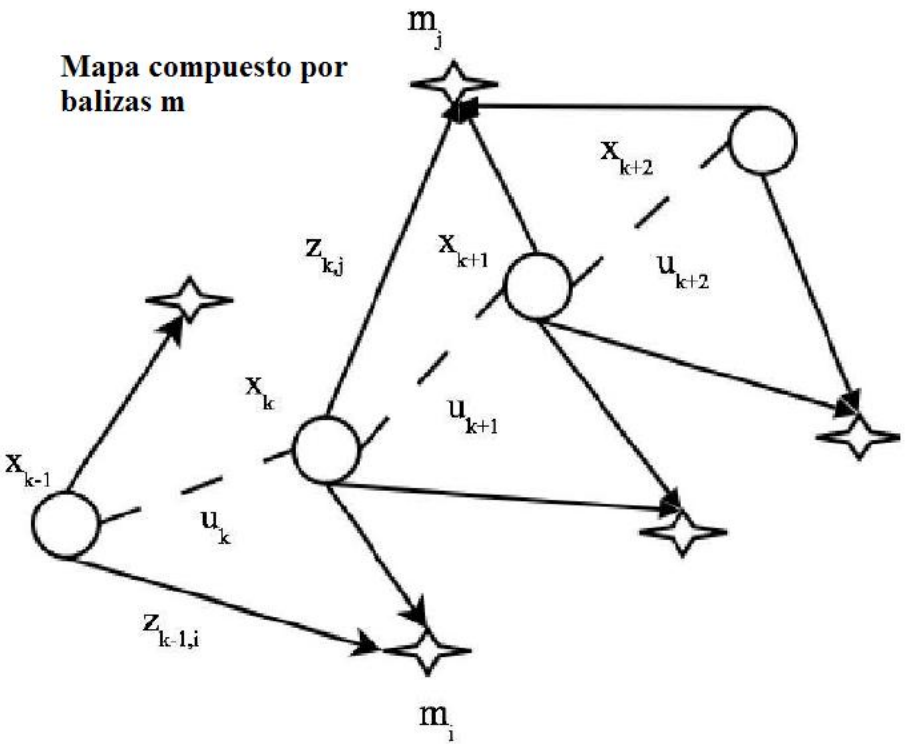
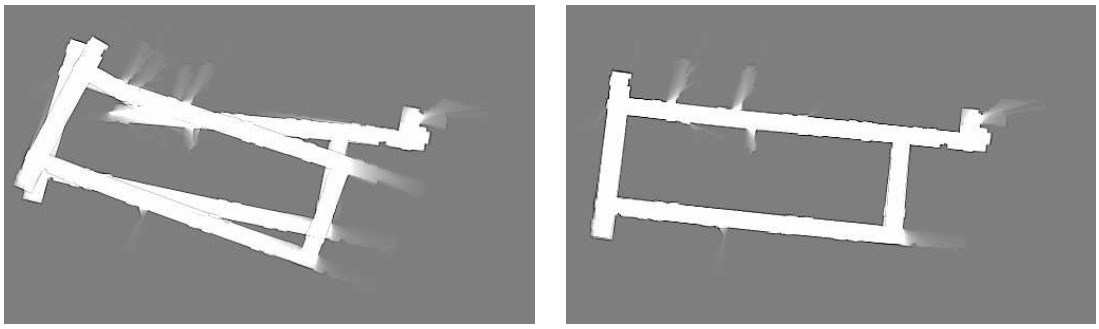


Figura 2.1: Descripción del Problema del SLAM.

Variable	Descripción
$x\,(t)$	Posición del robot en el instante t.
$u\,(t)$	Vector de control aplicado al instante *t-1* para mover al robot desde $x\,(t-1)$ a $x\,(t)$ en el instante t.
m_i	Posición del hito (elemento del mapa) m_i.
$z_i\,(t)$	Observación (medición) del hito m_i desde la posición $x\,(t)$ en el instante t.

Tabla 2.1: Notación General en la Descripción del SLAM.



(a) Sólo Odometría (b) Con Localización

Figura 2.2: La figura (a) es un mapa extraído utilizando únicamente la información odométrica del robot. La figura (b) es un mapa aprendido utilizando la localización del robot en el entorno.

Los sensores de movimiento propios del robot obtienen información importante para la construcción del mapa, al ofrecer la posición desde el punto de vista de observación; pero el movimiento del robot también está sujeto a errores, por lo que la simple información odométrica es insuficiente para determinar la posición y orientación del robot. Es por esta razón que el trabajo de la construcción del mapa exige una constante localización del robot.

En la figura 2.2(a) se observa un mapa generado sin el trabajo de localización, utilizando únicamente los datos arrojados por el sensor odométrico, los cuales, como ya se ha dicho, son susceptibles a error. Para empeorar la situación, el error odométrico es estadísticamente dependiente, es decir, es acumulativo: mientras más se desplaza el robot, el error registrado es mayor. En cambio, en la figura 2.2(b) se aprecia un mapa donde se calcula la localización del robot.

Básicamente, el problema de la construcción del mapa [Ca99] consiste en:

1. Captar el entorno en un instante t usando los sensores exteroceptivos[2] colocados en el robot móvil.

2. Representación de la información sensorial en el marco de referencia del robot móvil para obtener un mapa local en el instante t.

3. Asociación de los sensores para obtener una representación aumentada del mapa local actual, mediante la combinación de sus datos.

[2]Sensores que proporcionan percepción de aspectos externos al robot; por ejemplo, temperatura, presión, localización de objetos. [Al09]

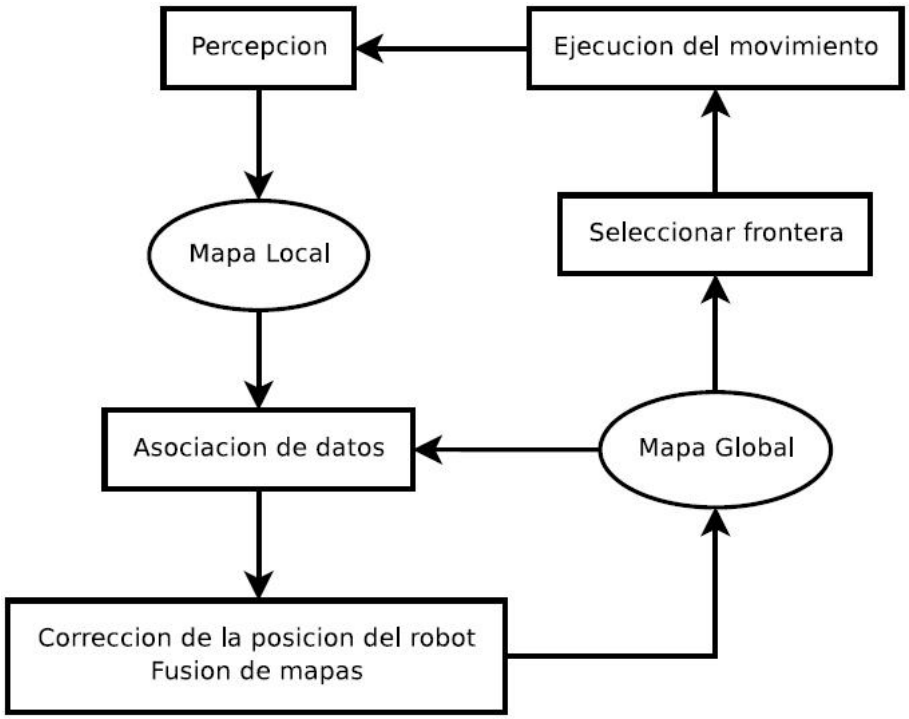


Figura 2.3: Procedimiento General del SLAM Incremental.

4. Integración del mapa local en el mapa global previo, para obtener un nuevo mapa global, y actualizar la localización estimada del robot.

5. Desplazamiento del robot móvil, el cual es estimado mediante *dead-reckoning*[3], entre dos puntos intermedios a lo largo de la trayectoria, para iterar el proceso.

El problema de la construcción de mapas incremental [Ja05] se puede esquematizar como se describe en la figura 2.3. La percepción de los sensores de distancia generan un mapa local en el instante de observación. En la asociación de datos, se tratan de identificar los puntos que ya se habían visto previamente para correlaccionarlos en el mapa global. Asimismo, se tiene que hacer un trabajo de corrección en la posición del robot, ya que, como se ha mencionado anteriormente, el uso único de información odométrica es

[3]Según la literatura, la definición de dead-reckoning, recogida en [Ge03], es un método de localización que implica el uso de un procedimiento matemático simple para estimar la localización actual de un vehículo, basada en una localización anterior conocida y una información incluida en la medida y la velocidad a través del tiempo del recorrido. Un ejemplo típico es la odometría, que deduce la posición del vehículo supervisando la rotación de las ruedas.

insuficiente. Posteriormente se selecciona un nuevo punto de observación, y se controla el robot para que se dirija a esa posición; y el ciclo se repite.

El problema de la construcción de mapas no se detiene con los problemas generados por la naturaleza ruidosa de los sensores, ya que existen también otras cuestiones difíciles de resolver.

- *Dimensionalidad del espacio.* Describir un espacio requiere mucha información, ya que hay que definir cada elemento dentro del ambiente, y mientras más detalle se le quiera agregar para tener una mejor estimación de movimientos, la dimensionalidad crece.

- *Asociación de correspondencias.* Este es el problema de mayor dificultad, y trata de determinar si diferentes mediciones, hechas en tiempos distintos, corresponden a un mismo objeto físico.

- *Ambientes dinámicos.* Una de las restricciones aplicadas al problema de la exploración es que el ambiente debe ser estacionario en el momento de realizar esta tarea. Esto se debe a que la eliminación de ruido en los sensores se complica, ya que un objeto trasladándose dentro del lugar, es un error de medición a eliminar dentro del mapa.

- *Exploración.* Determinar los movimientos exactos para la exploración en un tiempo y desplazamientos mínimos.

Muchos trabajos de la literatura de robótica, concerniente al problema de la construcción de mapas, imponen restricciones geométricas en la localización relativa entre observaciones y características globales para actualizar sus localizaciones estimadas [Ca99]. La precisión de sus soluciones han sido altamente influenciadas por la exactitud del sistema de estimación del robot móvil, cuyas estimaciones de localización cambian con el tiempo.

En un principio, el problema del SLAM es considerado si se asume la independencia entre la localización del robot, y las localizaciones características del mapa. De esta forma, se considera el problema en dos pasos diferentes. Primero, la localización del robot móvil se estima mediante el emparejamiento de observaciones actuales con el conocimiento previo del entorno. Entonces, se estiman las localizaciones características del mapa, considerando el mismo conjunto de emparejamientos y la reciente localización estimada del robot. Cuando tratamos con modelos probabilísticos, ésta aproximación no genera soluciones rigurosas para el problema, porque el

mismo conjunto de observaciones es usado dos veces sin considerar correlaciones. De este modo, se obtienen estimaciones poco realistas de incertidumbre.

Aproximaciones matemáticas exactas, teniendo en cuenta el problema de correlaciones, fueron dirigidas originalmente por Smith [Sm90], el cual introdujo el concepto de mapa estocástico, una representación de las relaciones espaciales, sus incertidumbres y las interdependencias de las estimaciones. Se representaba cada relación espacial incierta mediante el vector de sus variables espaciales con una distribución de probabilidad determinada. De este modo, el mapa estocástico se formaba mediante un vector de todas las variables espaciales y una matriz de covarianza que reflejaba la interdependencia entre ellas. Más tarde, esta aproximación se llamó *"aproximación directa"*, porque el vector de estado se consideraba como una entidad inseparable.

Mountarlier [Mo89] observó que la aproximación directa era propensa a inestabilidades en las etapas tempranas del proceso. Para solucionar el problema, modificó el esquema previo y propuso la aproximación de *fusión de reubicación*, donde el vector de estado era actualizado en dos etapas. Del emparejamiento entre las observaciones locales y las características del mapa, estimaba la localización del robot, su matriz de covarianza y la correlación entre la localización del robot y todas las características del mapa. Entonces, el mismo conjunto de emparejamientos se usaba para actualizar las características del mapa, sus covarianzas y sus interdependencias, usando la localización del robot actualizada previamente y sus correlaciones con esas características del mapa. En estos trabajos se experimentó con la aproximación de fusión de reubicación y se llegó a la conclusión de que se producía un algoritmo menos preciso pero con mejores propiedades de estabilidad. Observó mayor calidad del mapa, en comparación con la aproximación propuesta por Smith [Sm90].

Más tarde, Leonard y Durrant-Whyte ([Le91],[Le92a] y [Le92b]) propusieron un esquema con menor optimalidad, donde las correlaciones eran descartadas. Mediante la evaluación de una medida determinada, se decidía si actualizar la localización del robot o la localización de las características del mapa. Cuando la medida era muy ruidosa, se ignoraba.

Luego, Hébert ([He95] y [He96]) propuso un nuevo esquema, llamado el esquema de *fusión de reubicación con agrupamiento*. En estos trabajos se construyó el mapa mediante las relaciones espaciales observadas entre las características geométricas. Esas relaciones eran independientes de la información odométrica y estaban contenidas en los mapas locales. Sólo se consideró el modelo odométrico para orientar el reconocimiento y para

representar los mapas locales en un marco de referencia global cuando no podían ser relacionados por la observación del sensor.

Posteriormente, Chong [Ch97], trabajando con un robot móvil equipado con un conjunto de sensores sonar, abordó el problema del SLAM usando la aproximación de fusión de reubicación combinada con el filtro de Kalman. Con esto, en palabras del autor, se mejoraba la exactitud de la propagación de la covarianza a través de ecuaciones no lineales y eliminaba la necesidad de derivar las matrices Jacobianas.

2.4. Formas de SLAM: en línea y completa

Desde una perspectiva probabilística, hay dos formas fundamentales del problema del SLAM, las cuales son de igual importancia práctica. La primera es conocida como el *Problema del SLAM en línea*, y que implica la estimación de la ubicación actual sobre el paso anterior, junto con el mapa.

$$p\left(x\left(t\right), m | z_{1:t}, u_{1:t}\right)$$

Donde $x\left(t\right)$ es la ubicación en el instante t, m es el mapa, y $z_{1:t}$ y $u_{1:t}$ son las medidas y los controles, respectivamente.

Este problema es llamado el *problema del SLAM en línea* puesto que sólo supone la estimación de las variables que persisten en el instante t. Muchos algoritmos para el problema del SLAM en línea son incrementales, de manera que descartan medidas y controles anteriores, una vez que hayan sido procesados. El modelo gráfico del SLAM en línea es representado en la figura 2.4.

El segundo problema tipo es llamado el *problema del SLAM completo*. En SLAM completo, procuramos calcular la ruta completa $x_{1:t}$ sobre pasos anteriores, en vez de con la posición actual $x\left(t\right)$ solamente. El modelo gráfico del SLAM completo es representado en la figura 2.5.

$$p\left(x_{1:t}, m | z_{1:t}, u_{1:t}\right)$$

Esa sutil diferencia entre SLAM en línea y SLAM completo tiene implicaciones en el tipo de algoritmos que pueden llegar a ser soportados. En particular, el problema del SLAM en línea es el resultado de integrar hacia afuera ubicaciones anteriores del problema del SLAM completo.

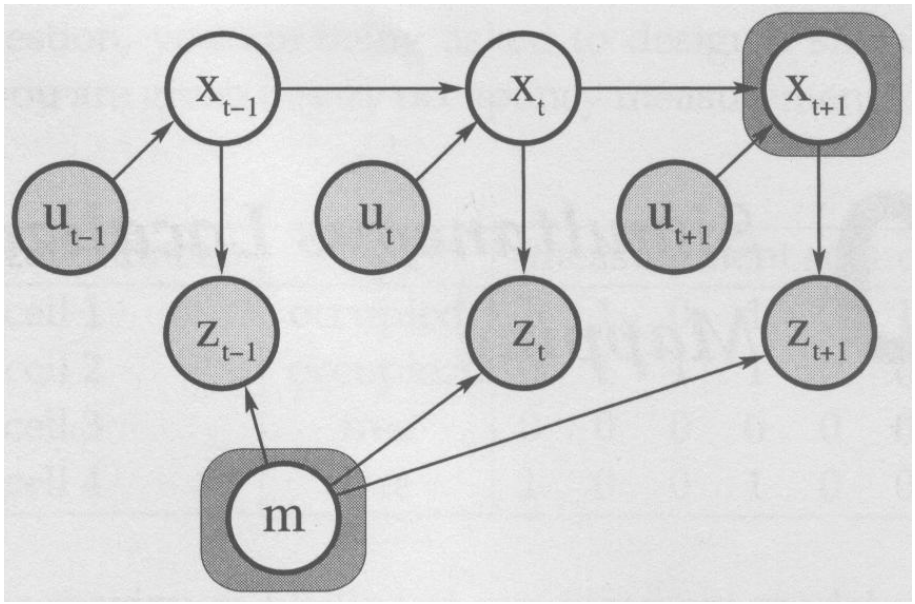


Figura 2.4: Imagen del SLAM en línea.

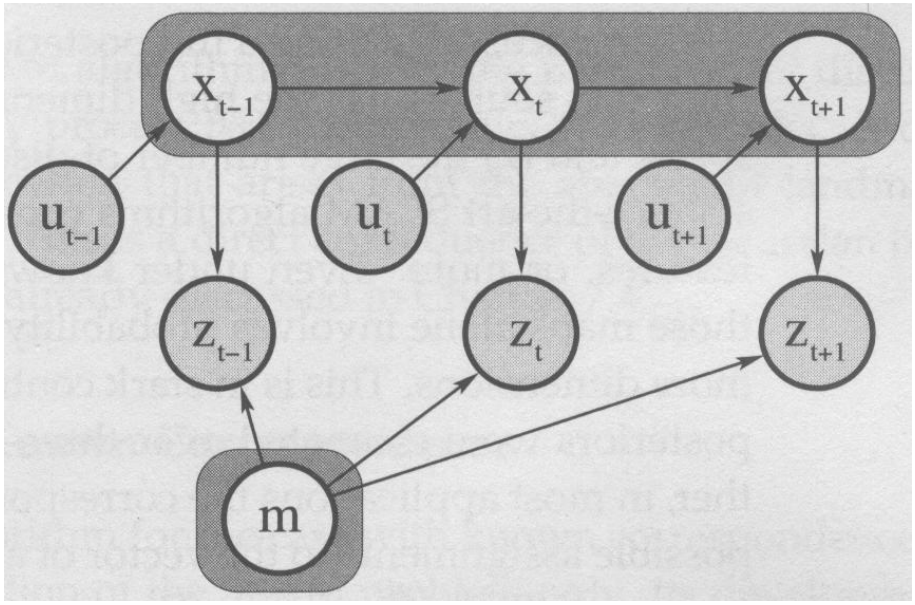


Figura 2.5: Imagen del SLAM completo.

$$p\left(x\left(t\right), m|z_{1:t}, u_{1:t}\right) = \int\int\cdots\int p\left(x_{1:t}, m|z_{1:t}, u_{1:t}\right) dx_1 dx_2 \ldots dx_{t-1}$$

En SLAM en línea, esas integraciones son típicamente representadas una por una, lo que determina cambios interesantes en las estructuras de dependencia en SLAM.

Una segunda característica clave del problema del SLAM tiene que ver con la naturaleza del problema de la estimación. Los problemas de SLAM poseen un componente continuo y un componente discreto. El problema de la estimación continua está relacionado con la localización de los objetos en el mapa y las variables de posición del propio robot. Los objetos pueden ser marcas en la representación, a base de características, o podrían ser áreas de un objeto detectado por buscadores de rango. La natraleza discreta tiene que ver con la correspondencia. Cuando se detecta un objeto, un algoritmo SLAM debe razonar sobre la relación de ese objeto con los objetos previamente detectados. Este razonamiento es típicamente discreto: el objetivo es uno de los detectados previamente, o no.

2.5. Filtros de Kalman

El filtro de Kalman se puede definir como un algoritmo recursivo óptimo de procesamiento de datos [Ma79] o como un estimador recursivo [Wi06]. Un aspecto de esa optimalidad es que el filtro de Kalman incluye toda la información que se le puede proporcionar, lo que significa que sólo se necesita el estado estimado en el paso previo y la medición actual para calcular la estimación del estado actual. En contraste con las técnicas de estimación por lotes [Ma79], no se requiere ningún histórico de las observaciones y/o las estimaciones, ya que se procesan todas las medidas disponibles, sin reparar en sus precisiones, para estimar el valor actual de las variables de interés, con el uso de:

1. El conocimiento del dinamismo del sistema y del dispositivo de medida.
2. La descripción estadística de los ruidos del sistema, errores de medida e incertidumbre en los modelos dinámicos.
3. Cualquier información disponible sobre las condiciones iniciales de las variables de interés.

Con respecto a la recursividad, significa que, a diferencia de ciertos conceptos de procesamiento de datos, el filtro de Kalman no necesita que

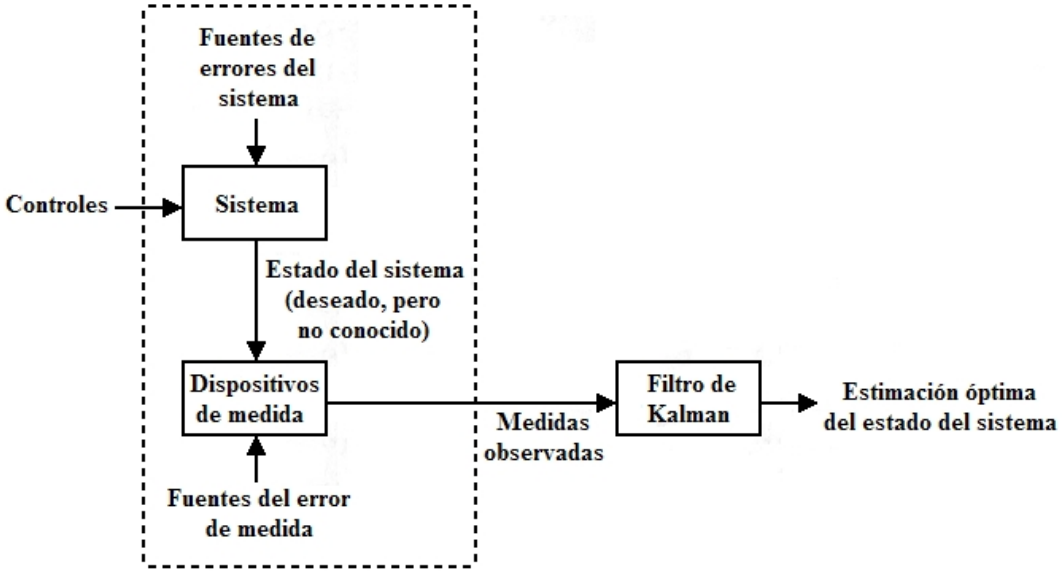


Figura 2.6: Aplicación Típica del Filtro de Kalman.

todos los datos previos sean almacenados y vueltos a procesar cada vez que se obtiene una nueva medida. Esto es de vital importancia para que la implementación del filtro sea factible.

El estado del filtro de Kalman se representa mediante dos variables [Wi06]:

- $\hat{x}\,(t|t)$, la estimación del estado en el instante t;
- $P\,(t|t)$, la matriz de covarianza del error.

El filtro de Kalman tiene dos fases distintas: *Predicción* y *Actualización*. La fase de Predicción usa la estimación del paso anterior para generar una estimación del estado actual. En la fase de Actualización, la información medida en el instante actual se usa para perfecionar la predicción, para lograr una nueva estimación más precisa.

La figura 2.6 describe una situación típica donde un filtro de Kalman puede ser usado con una gran ventaja. Un sistema de algún tipo es comandado por algunos controles conocidos, y dispositivos de medida proporcionan el valor de ciertas magnitudes. El conocimiento de esas entradas y salidas del sistema es todo lo que está explícitamente disponible del sistema físico para los objetivos estimados.

A menudo las variables de interés, algún número finito de cantidades para describir el estado del sistema, no pueden ser medidas directamente, y se han de generar los valores de esos datos deseados realizando las suposiciones por deducción. Esa deducción es complicada, por el hecho de que el sistema es controlado típicamente por más entradas que nuestros propios controles conocidos, y que la relación entre varias variables de estado y las salidas medidas, son conocidas sólo con unos grados de incertidumbre. Además, cualquier medida estará influenciada por algún grado de ruido e inexactitudes del dispositivo, y así también debe proporcionar el medio de extraer la información valiosa de una señal ruidosa. También puede haber un número de dispositivos de medida diferentes, cada uno con sus características particulares de dinamismo y error, que proporciona alguna información sobre alguna variable particular, y sería deseable combinar esas salidas de una manera sistemática y óptima. Un filtro de Kalman combina todos los datos medidos disponibles, conocimiento anterior favorable sobre el sistema y los dispositivos de medida, para producir una estimación de las variables deseadas de tal manera que el error es minimizado estadísticamente.

Conceptualmente, lo que cualquier tipo de filtro trata de hacer es obtener una estimación *óptima* de las cantidades disponibles procedentes de los datos obtenidos de un entorno ruidoso, entendiéndose por *óptimo* el que minimiza errores en algún sentido.

Hay varias maneras de lograr este objetivo. Si adoptamos un punto de vista bayesiano, queremos el filtro para propagar la densidad de probabilidad condicional de las cantidades disponibles, condicionado en el conocimiento de la llegada del dato real de los dispositivos de medida.

Para entender este concepto, la figura 2.7 muestra una descripción de una densidad de probabilidad condicionada del valor de una cantidad escalar x en el instante de tiempo i $(x\,(i))$, condicionado en el conocimiento de que el vector de medida $z\,(1)$ en el instante de tiempo 1 asume el valor z_1 $(z\,(1) = z_1)$, y de forma similar para los instantes 2 hasta i, dibujada como una función de $x\,(i)$ valores posibles. Esto se denota como $f_{x(i)|z(1),z(2),\cdots,z(i)}\,(x|z_1, z_2, \cdots, z_i)$. Por ejemplo, sea $x\,(i)$ la posición unidimensional de un vehículo en el instante 1, y sea $z\,(i)$ un vector bidimensional que describe las medidas de posición en el instante j por dos radares separados. Tal densidad de probabilidad condicionada contiene toda la información disponible sobre $x\,(i)$: Indica, para el valor dado de todas las medidas obtenidas en el instante i, cual puede ser la probabilidad de que $x\,(i)$ asuma cualquier valor concreto, o rango de valores.

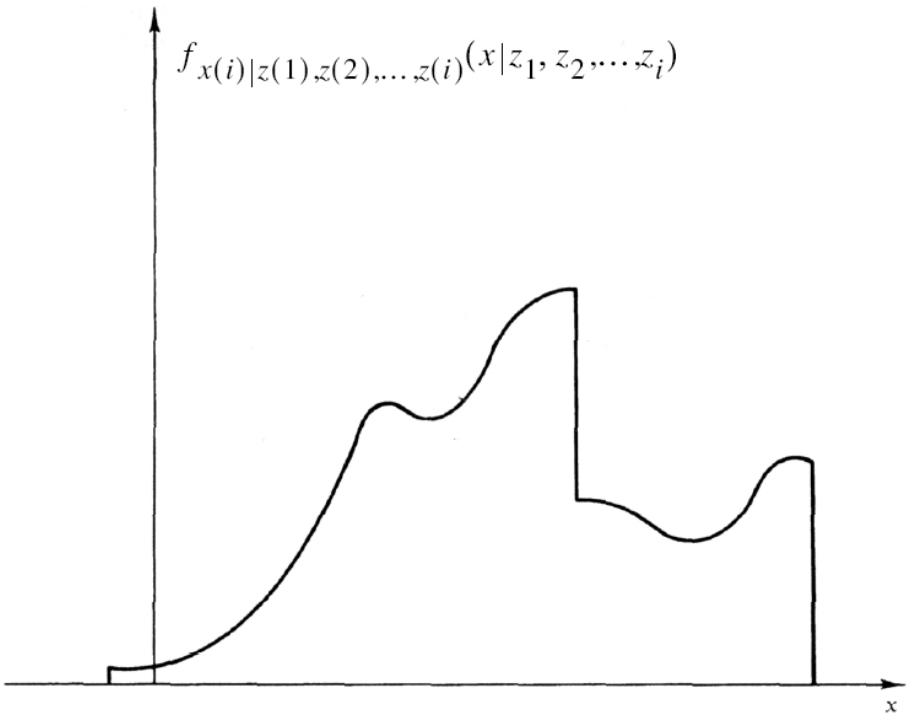


Figura 2.7: Densidad de Probabilidad Condicionada.

Se califica como densidad de probabilidad condicionada porque su estado y localización en el eje x es dependiente de los valores de las medidas tomadas. Su estado trasmite el valor de certeza que tiene al conocer el valor de x. Si la gráfica de densidad tiene un pico estrecho, entonces el mayor peso de la probabilidad estará concentrado en una franja reducida de valores de x. Por el contrario, si la gráfica tiene una forma suave, el peso de la probabilidad se encuentra repartido en un amplio rango de x, indicando menos seguridad en su valor.

Una vez que la función de densidad de probabilidad condicionada es propagada, se puede definir la estimación óptima. Se incluyen varias posibilidades:

1. La media: La estimación del centro de la concentración de la probabilidad.

2. La moda: El valor de x que tiene la probabilidad mayor, localizando el pico de la densidad.

3. La mediana: El valor de x, tal que la mitad del peso de la probabilidad se encuentra repartida a su izquierda, y la otra mitad a su derecha.

Un filtro de Kalman realiza esta propagación de densidad de probabilidad condicional para problemas en donde el sistema puede ser descrito por un modelo lineal y en donde el sistema y los ruidos de medida son blancos y gaussianos. En estas condiciones, la media, la moda, la mediana y prácticamente cualquier alternativa razonable para una estimación *"optima"* coinciden, de manera que, de hecho, hay una única mejor estimación del valor de x. Conforme a esas tres restricciones, el filtro de Kalman puede verse como el mejor filtro posible. Algunas restricciones pueden ser suavizadas, flexibilizando un filtro óptimo cualificado. Por ejemplo, si eliminamos la suposición gaussiana, el filtro de Kalman puede verse como el mejor filtro (varianza de error mínimo) fuera de la clase de filtros lineales imparciales. Sin embargo, estas tres suposiciones pueden estar justificadas para muchos usos potenciales.

2.6. Filtros de Partículas

El filtro de partículas es un método empleado para estimar el estado de un sistema que cambia a lo largo del tiempo [Wi06]. Más concretamente, es un método Monte Carlo (secuencial) que conforma la base para la mayoría de los filtros Monte Carlo desarrollados [Ja05]. Dicho método es usado comúnmente en visión artificial para el seguimiento de objetos, en secuencias de imágenes. [Wi06]

Este método fue propuesto en 1993 por N. Gordon, D. Salmond y A. Smith [Go93] como filtro bootstrap para implementar filtros bayesianos recursivos. Básicamente, el filtro de partículas se compone de un conjunto de muestras (las partículas) y unos valores, o pesos, asociados a cada una de esas muestras. Las partículas son estados posibles del proceso, que se pueden representar como puntos en el espacio de estados de dicho proceso.

A los filtros de partículas también se les conoce como secuencias Monte Carlo, muestreo secuencial de importancia (SIS[4]), bootstrap filter, algoritmo de condensación, supervivencia por adaptación, etc [Ja05]. Se considera una implementación alternativa no paramétrica del filtro de Bayes [Th05].

Al igual que los filtros de histograma, el filtro de partículas aproxima el estado siguiente mediante un número finito de parámetros. Sin embargo, se diferencian en el modo en que estos parámetros se generan, y en el modo que pueblan el espacio de estados. La idea fundamental es representar la función de distribución posterior, buscada con un conjunto de muestras aleatorias con

[4] *Sequencial Importance Sampling*, Muestreo Secuencial de Importancia

pesos asociados, y calcular estimaciones basadas en estas muestras y pesos [Ja05]. Al aumentar el número de muestras, esta caracterización Monte Carlo se vuelve una representación equivalente de la descripción funcional usual de la función de distribución posterior, y el filtro SIS se acerca al estimador Bayesiano óptimo.

En el filtro de partículas [Th05], las muestras de una distribución del siguiente estado se llaman *partículas*, y se denotan

$$X(t) = x_t^{[1]}, x_t^{[2]}, \cdots, x_t^{[M]} \tag{2.1}$$

Cada partícula $x_t^{[m]}$, con $1 \leq m \leq M$, es una instancia concreta del estado en el instante t. Puesto de manera diferente, una partícula es una hipótesis, en cuanto a en qué estado global verdadero se puede estar en el instante t. M denota el número de partículas en el conjunto de partículas $X(t)$.

La intuición detrás de los filtros de partículas es aproximar la confianza mediante un conjunto de partículas $X(t)$. Idealmente, la probabilidad para una hipótesis de estado x_t para ser incluida en el conjunto de partículas $X(t)$ será proporcional a su posterior filtro de Bayes:

$$x_t^{[m]} \sim p(x_t | z_{1:t}, u_{1:t}) \tag{2.2}$$

Como consecuencia de la ecuación 2.2, en una subregión más densa del espacio de estados poblada de muestras, lo más probable es que el verdadero estado se encuentre dentro de esa región. Para M finita, las partículas son dibujadas por una distribución ligeramente diferente. En la práctica, esta diferencia es insignificante mientras el número de partículas no sea demasiado pequeño.

Al igual que el resto de los algoritmos del filtro de Bayes, el algoritmo del filtro de partículas contruye, recurrentemente, la confianza del instante t mediante la confianza del instante $t-1$ en un paso anterior. Puesto que la confianza se representa por un conjunto de partículas, esto significa que los filtros de partículas construyen el conjunto de partículas $X(t)$ recursivamente a partir del conjunto $X(t-1)$.

El filtro de partículas posee cuatro etapas principales [Wi06]:

- Inicialización.
- Actualización.

- Estimación.
- Predicción.

Para realizar el seguimiento de un objeto sobre una secuencia de imágenes, por ejemplo, el filtro de partículas "lanza" al azar un conjunto de puntos sobre la imagen (etapa de inicialización, se crea un conjunto de partículas con un estado aleatorio), realizando cálculos, se le asignará un valor, o valores, a cada uno de esos puntos (etapa de actualización). A partir de estos valores, se creará un nuevo conjunto de puntos que reemplazará al anterior. Esta elección también será al azar, pero los valores que se han adjudicado a cada uno de los puntos provocarán que sea más probable de elegir aquellos puntos que hayan capturado al objeto sobre el que se quiere realizar el seguimiento (etapa de estimación). Una vez que se crea el nuevo conjunto de puntos, se realiza una leve modificación al estado (posición) de cada uno de ellos, con el fin de estimar el estado del objeto en el instante siguiente (etapa de predicción).

Al terminar la etapa de predicción, se obtiene un nuevo conjunto de puntos al que se le vuelve a aplicar la etapa de actualización, repitiéndose este bucle hasta que termine la secuencia o desaparezca el objeto, caso en el cual se volvería a la etapa de inicialización.

2.7. Otros métodos

2.7.1. Estimación Coherente de la Ubicación

El método de la Estimación Coherente de la Ubicación (CPE[5]) [Id06] es un método para realizar la Construcción de Mapas y Localización Simultánea (problema SLAM), desarrollado por Lu y Milios [Lu97]. Ambos demostraron que la información de los codificadores del robot y los sensores láser podrían ser representados como una red de restricciones de probabilidad, uniendo las posiciones sucesivas (ubicaciones) del robot mediante la estimación; las exploraciones láser se emparejan para dar nuevas restricciones entre ubicaciones del robot, incluyendo restricciones para cuando un robot vuelve a un área visitada anteriormente. Las restricciones entre ubicaciones son encontradas mediante la correspondencia de características comunes. Dicho método proporciona mapas métricos de alta calidad.

El método CPE tiene en cuenta la incorporación eficiente de nueva información de exploración láser en un mapa creciente. Dentro de este marco,

[5] *Consistent Pose Estimation*, Estimación Coherente de la Ubicación

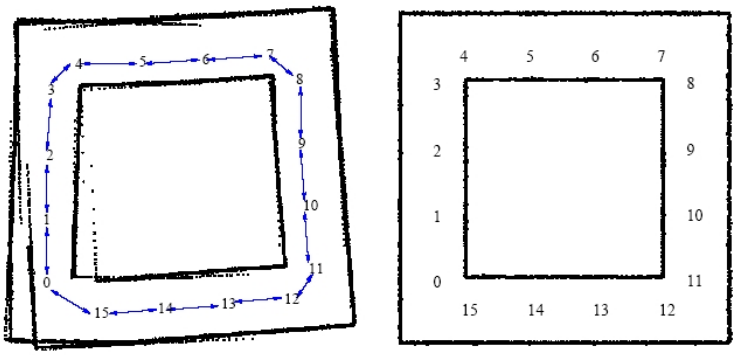


Figura 2.8: Estimación coherente de la ubicación: Conjunto de ubicaciones del robot basadas en odometría con restricciones entre las ubicaciones (izquierda) y el mapa obtenido de la estimación coherente de la ubicación (derecha).

se ha encontrado una solución para el conocido problema del cierre del lazo de control: cómo detectar de forma óptima la información láser cuando el robot regresa a un área explorada con anterioridad (y "reconocer" que estuvo allí antes). El mapa se determina minimizando el error de las restricciones locales [Ko05]. Con CPE, es posible crear mapas de alta resolución y ejecutar repetidamente la ruta precisa necesaria para el comportamiento deliberativo de alto nivel.

2.7.2. Registro Local y Correlación Global

Es un nuevo método para la estimación del método CPE. En el método LRGC[6] [Gu99], con respecto al método CPE, se introducen dos técnicas que permiten agregar eficientemente la nueva información al mapa actual, y la determinación de las relaciones topológicamente correctas entre las ubicaciones, sobre todo después de ciclos largos.

El registro local trabaja bien si el robot explora constantemente nuevas áreas. Mientras que termina un ciclo largo, el problema de la corrección topológica llega a ser importante, porque las nuevas ubicaciones se deben relacionar con las anteriores. Llegados a este punto, es crítico hacer identificaciones topológicas de forma fiable, porque un error puede causar que el mapa sea mal alineado. Las exploraciones individuales no tienen generalmente suficiente información para producir un buen rechazo de falso positivo, especialmente si el entorno es relativamente uniforme en una dirección, como sucede a menudo

[6] *Local Registration and Global Correlation* , Registro Local y Correlación Global

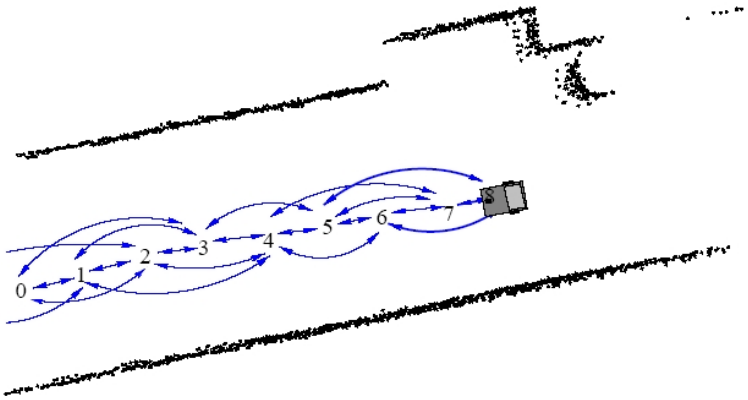


Figura 2.9: Efecto de añadir una nueva ubicación de un entorno no explorado al mapa.

a lo largo de un pasillo. En su lugar, se integra un conjunto de exploraciones locales en una parcela del mapa, y utilizamos esta plantilla menos limitada para encontrar coincidencias en el mapa antiguo. La técnica de la parcela del mapa es obviamente más fiable que las exploraciones individuales en el rechazo de falso positivo, si bien complica ligeramente la fase de comparación, que debe utilizar ahora mecanismos de correlación.

El algoritmo LRGC demuestra cómo las técnicas de correspondencia de exploración, de registro, y de correlación se pueden unir en un sistema práctico de tiempo real para la construcción del mapa usando una densa información de rango. El LRGC depende de tres técnicas: Correspondencia de exploración, estimación coherente de la ubicación, y correlación del mapa.

Capítulo 3

Filtros de Kalman

3.1. Introducción

Como se ha visto en el capítulo anterior, existen multitud de aproximaciones que se han desarrollado para abordar el problema del SLAM. Era preciso, por tanto, enfocar el desarrollo de este proyecto de fin de carrera hacia un área concreta, con vistas a permitir cumplir con los objetivos en un plazo razonable.

La decisión final adoptada fue la de centrar el estudio en las técnicas basadas en filtros y estimadores. Por ello, se va a dedicar el presente capítulo al análisis de los filtros de Kalman y sus derivados.

3.2. Definición

Cuando se controla un robot móvil, con frecuencia se debe combinar información de múltiples fuentes [Du00]. Mientras hay muchos enfoques de aplicaciones específicas, para estimar un estado desconocido, de un conjunto de medidas procesadas, muchos de esos métodos no toman en consideración, intrínsecamente, la naturaleza ruidosa de las medidas [We01]. Ese ruido es típicamente de carácter estadístico (o puede ser modelado como tal de manera eficaz), el cual nos conducirá a métodos estocásticos para tratar el problema.

Los modelos espaciales de estado son básicamente un convenio notacional para problemas de estimación y de control, desarrollado para hacer manejable lo que de otra manera sería un análisis de notación intratable. Consideremos un proceso dinámico descrito por una ecuación en diferencias de orden n (similar a una ecuación diferencial) de la forma

$$y\left(t+1\right) = a_{0,t}z_t + \cdots + a_{n-1,t}z_{t-n+1} + u\left(t\right)$$

la variable independiente t puede tomar los valores $t_0 \leq t_1 \leq \cdots \leq t_n$, donde los t_i no están necesariamente equidistantes [So85]. El término $u(t)$ ([We01]) es un proceso "ruidoso" aleatorio blanco de media cero con autocorrelación

$$E(u(t), u(j)) = R(u) = Q(t)\,\delta_{k,j},$$

donde $\delta_{k,j}$ representa la delta de Kronecker, y los valores iniciales $z_0, z_{-1}, \cdots, z_{-n+1}$ son variables aleatorias de media cero con una matriz de covarianza nxn conocida

$$P(0) = E(z(-j), z(-t)),\ j, t \in \{0, n-1\}.$$

También se supone que

$$E(u(t), z(t)) = 0\ para\ -n+1 \leq j \leq 0\ y\ t \geq 0$$

que asegura que

$$E(u(t), z(t)) = 0,\ t \geq j \geq 0.$$

En otras palabras, el ruido es estadísticamente independiente del proceso a ser estimado. Bajo algunas otras condiciones básicas, esta ecuación en diferencias puede ser reescrita como

$$\hat{x}(t+1) \equiv \begin{bmatrix} z_{t+1} \\ z_t \\ z_{t-1} \\ \vdots \\ z_{t-n+2} \end{bmatrix} = \underbrace{\begin{bmatrix} a_0 & a_1 & \cdots & a_{n-2} & a_{n-1} \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}}_{A(t)} \underbrace{\begin{bmatrix} z_t \\ z_{t-1} \\ z_{t-2} \\ \vdots \\ z_{t-n+1} \end{bmatrix}}_{\hat{x}(t)} + \underbrace{\begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}}_{G(t)} u(t)$$

que conduce al modelo espacial de estado

$$\hat{x}(t+1) = A(t)\,\hat{x}(t) + G(t)\,u(t) \tag{3.1}$$

$$\hat{z}(t) = H(t)\,\hat{x}(t) \tag{3.2}$$

La ecuación 3.1 representa la forma de modelar un nuevo estado $\hat{x}(t+1)$ como una combinación lineal del estado anterior $\hat{x}(t)$ y del ruido de proceso $u(t)$. La ecuación 3.2 describe la forma de obtener las medidas de proceso u observaciones $\hat{z}(t)$ del estado interno $\hat{x}(t)$. Estas dos ecuaciones son a menudo referidas como el *modelo de proceso* y el *modelo de medida*, respectivamente, y sirven como base para prácticamente todos los métodos de valoración lineales,

como el filtro de Kalman.

Hay un problema general relacionado en el área de la teoría de sistemas lineales, generalmente llamado *el problema del diseño del observador*. El problema básico es determinar (estimar) los estados internos de un sistema lineal, dado que sólo se tiene acceso a las salidas del sistema. Esto es parecido al problema de la "caja negra", donde se tiene acceso a algunas señales provenientes de la caja (las salidas) pero no se puede observar directamente qué ocurre en el interior.

Muchos enfoques para este problema, por regla general, están basados en los modelos espaciales de estado. Hay generalmente un modelo de proceso que modela la transformación del estado de proceso. Usualmente puede ser representado como una ecuación en diferencias lineal estocástica, similar a la ecuación 3.1:

$$x\left(t+1\right) = A\left(t\right)x\left(t\right) + B\left(t\right)u\left(t\right) + v\left(t\right) \tag{3.3}$$

Existen varias formas del *modelo de medida* que describe la relación entre el estado de proceso y las medidas. Normalmente se puede representar con una expresión lineal similar a la ecuación 3.2:

$$z\left(t\right) = H\left(t\right)x\left(t\right) + w\left(t\right) \tag{3.4}$$

Los términos $v\left(t\right)$ y $w\left(t\right)$ son variables aleatorias que representan el ruido de proceso y el ruido de la medida, respectivamente. El fundamento debe reforzar la idea de que las medidas no tienen que ser de elementos del estado específicamente, sino que puede ser cualquier combinación lineal de los elementos del estado.

El filtro de Kalman es esencialmente un conjunto de ecuaciones matemáticas que implementa un estimador de tipo predictor-corrector, óptimo en el sentido que minimiza la covarianza del error estimado, cuando algunas condiciones supuestas son satisfechas. Se considera una herramienta estándar de estimación usada extensamente en la construcción de sistemas de navegación [Cs97]. Desde el instante de su presentación, en el año 1960, el filtro de Kalman ha sido el argumento de importantes investigaciones y aplicaciones, particularmente en el área de la navegación autónoma o navegación asistida, donde se utiliza para mantener una estimación en curso de la posición del vehículo y la orientación, o de los parámetros, describiendo objetos de interés en el entorno [We01].

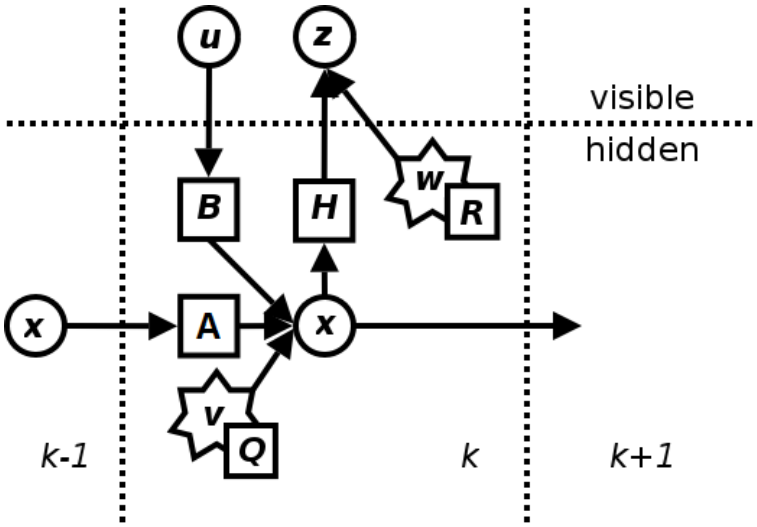


Figura 3.1: Modelo base del filtro de Kalman.

Según el uso del filtro de Kalman ([Wi06]), para estimar el estado interno de un proceso, dado sólo una secuencia de observaciones ruidosas, se debe modelar el proceso de acuerdo con la estructura del filtro de Kalman. De esta manera, especificando las matrices $A(t), H(t), Q(t), R(t)$ y algunas veces $B(t)$ para cada instante de tiempo t, se puede obtener una representación como la mostrada en la figura 3.1. En dicha figura, los círculos son vectores, los cuadrados son matrices, y las estrellas representan el ruido Gaussiano, con la matriz de covariancia asociada en el lado inferior derecho.

El filtro de Kalman permite una estimación actual de la posición del robot, por ejemplo, para combinarla con información posicional de uno o más sensores [Du00]. Una propiedad crítica del filtrado de Kalman es que permite y requiere la estimación en curso no solo de una variedad de parámetros, sino además la seguridad en esas estimaciones en forma de matriz de covarianza. Bajo circunstancias seguras, el filtrado de Kalman logra esa actualización de una manera óptima, tal que minimiza el error esperado en la estimación.

Las suposiciones hechas por el filtro de Kalman son:

- Ruido de sistema de media cero $E[v_i] = 0$ donde $E[\ ]$ es el valor esperado.
- Ruido independiente: $E[v_i, v_j] = 0\ \forall i \neq j$. En otras circunstancias, la covarianza del ruido del sistema es dada por $E[v_i, v_i] = \sigma_i(t)$.
- Un modelo lineal de la evolución del sistema activo.

- Una relación lineal entre el estado del sistema (posicionamiento) y las medidas acabadas de hacer.

Desafortunadamente, es raro el caso de una aplicación práctica que emplee todos esos requisitos. Si no se consideran las suposiciones anteriores, el filtro de Kalman se puede usar todavía, pero la garantía de una actualización óptima podría no ser válida. Si las suposiciones se incumplen de manera importante, el filtro conducirá a resultados de muy mala calidad.

En el caso de datos cuantitativos, y en particular, en los datos estimados del estado, una manera natural de describir nuestra confianza en una medida es usando una matriz de covarianza. Si se asume que los errores están distribuidos normalmente, esto permite que describamos la distribución de probabilidad para los errores explícitamente sobre la media. El filtro de Kalman es un mecanismo para combinar la información con el fin de que la información fiable posea mayor peso. Los pasos claves implican el uso de la ganancia de Kalman para ponderar la contribución relativa de nuevas medidas a nuestras expectativas anteriores. La ganancia de Kalman varía en proporción directa con la matriz de covariaza del estado, e inversamente, como la matriz de covarianza de la medida.

3.3. Formulación

En el filtrado de Kalman se definen tres modelos (Modelo del sistema, Modelo del sensor o de medición y Modelo a priori), al conjunto de los cuales se le suele llamar modelo del proceso, y dos fases (Fase de Propagación y Fase de Actualización) que constituyen el filtrado de Kalman propiamente dicho [Va96]. En la figura 3.2 se muestra un esquema ilustrativo del funcionamiento de un filtro de Kalman.

- *Modelo del sistema.* Describe la evolución en el tiempo de la cantidad que se desea estimar, expresada mediante un vector de estado $x(t)$. La transición entre estados $x(t) \rightarrow x(t+1)$ se caracteriza por la matriz de transición $A(t)$ y la adición de un ruido gaussiano $v(t)$ de media cero y con matriz de covarianza Q.

$$x(t+1) = A(t)\,x(t) + B(t)\,u(t) + v(t)\,, v(t) \rightarrow N(0, Q)$$

- *Modelo del sensor o de medición.* Relaciona el vector de medida $z(t)$ con el estado del sistema $x(t)$ a través de la matriz de medición $H(t)$ y la adición de un ruido gaussiano $w(t)$ con matriz de covarianza R.

$$z(t) = H(t)\,x(t) + w(t)\,, w(t) \rightarrow N(0, R)$$

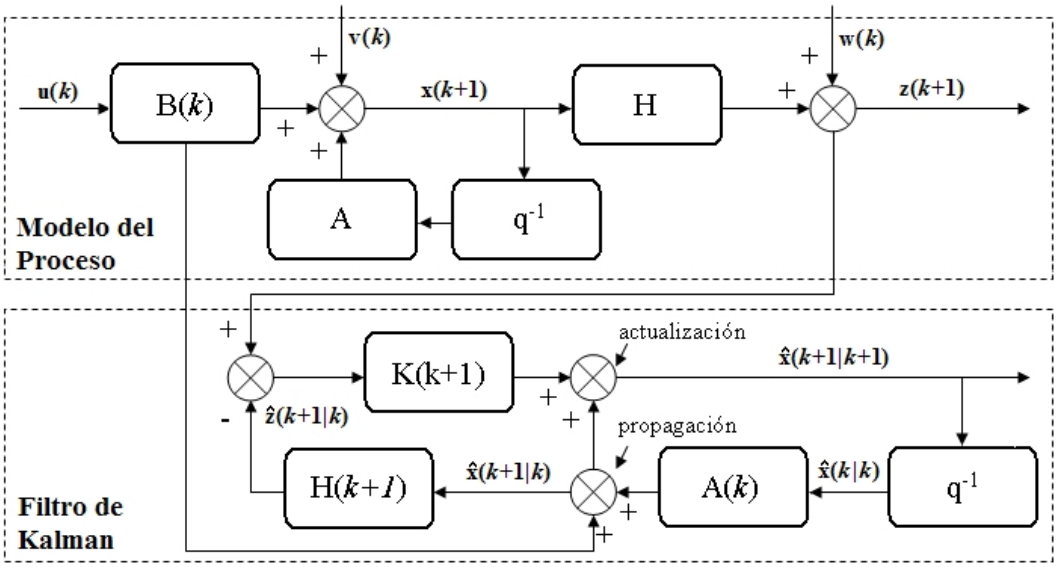


Figura 3.2: Filtrado de Kalman.

- *Modelo a priori.* Describe el conocimiento previo sobre el vector de estado en el instante inicial $x(0)$ en cuanto a valor esperado y matriz de covarianza $P(0)$. Se suponen incorrelados los ruidos correspondientes al proceso y al sensor.

$$E[x(0)] = \hat{x}(0), cov[x(0)] = P(0)$$
$$E\left[w_i v_j^T\right] = 0$$

- *Fase de Propagación.* En esta fase, se pretende predecir cuál va a ser el nuevo valor de la cantidad que se desea estimar. Para ello, la estimación del estado anterior $\hat{x}(t|t)$ y su matriz de covarianza $P(t|t)$ son extrapoladas para formar el vector de estado predicho $\hat{x}(t+1|t)$ y su matriz de covarianza $P(t+1|t)$.

$$\hat{x}(t+1|t) = A(t)\,\hat{x}(t|t) + B(t)\,u(t)$$

$$P(t+1|t) = A(t)\,P(t|t)\,A^T(t) + Q(t)$$

- *Fase de Actualización.* En esta fase, se calcula el nuevo vector de estado $\hat{x}(t+1|t+1)$ y su matriz de covarianza $P(t+1|t+1)$. Para ello, se utiliza la covarianza predicha para calcular la ganancia de Kalman $K(t)$, escalando ésta por el valor del residuo de medición $z(t+1) -$

$H\,(t+1)\,\hat{x}\,(t+1|t)$, o sea por la estimación del error cometido en la predicción, y sumándose al vector de estado predicho $\hat{x}\,(t+1|t)$ para calcular el nuevo vector de estado $\hat{x}\,(t+1|t+1)$.

$$K\,(t+1) = P\,(t+1|t)\,H^T\,(t+1)\left[H\,(t+1)\,P\,(t+1|t)\,H^T\,(t+1) + R\right]^{-1}$$

$$\hat{x}\,(t+1|t+1) = \hat{x}\,(t+1|t) + K\,(t+1)\left[z\,(t+1) - H\,(t+1)\,\hat{x}\,(t+1|t)\right]$$

$$P\,(t+1|t+1) = \left[1 - K\,(t+1)\,H\,(t+1)\right]P\,(t+1|t)$$

Según [We01], las ecuaciones para el filtro de Kalman se pueden dividir en dos grupos: las ecuaciones de actualización de tiempo y las ecuaciones de actualización de la medida. Las ecuaciones de actualización de tiempo son responsables de proyectar adelante el estado actual y las estimaciones de la covarianza del error para obtener las estimaciones a priori del siguiente paso. Las ecuaciones de actualización de la medida son responsables de la realimentación, por ejemplo para incorporar una nueva medida en la estimación a priori para obtener una estimación a posteriori mejorada; Las ecuaciones de actualización de tiempo se pueden pensar como *ecuaciones de predicción*, mientras las ecuaciones de actualización de medida se pueden pensar como *ecuaciones de corrección o actualización*.

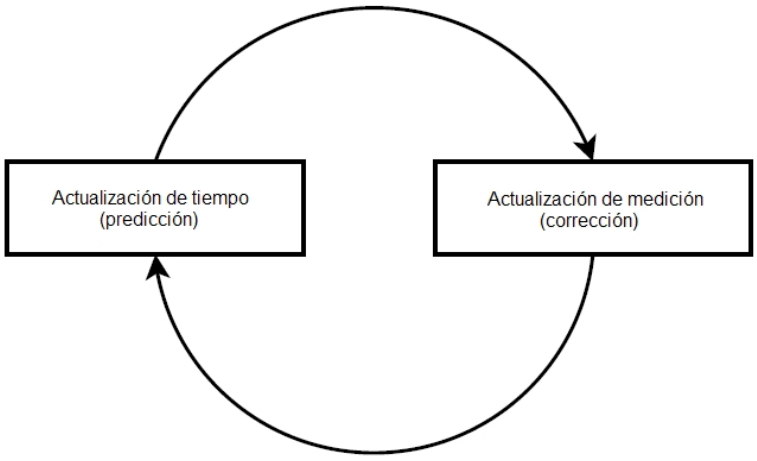


Figura 3.3: El ciclo del filtro de Kalman en desarrollo.

En la figura 3.3, la actualización del tiempo proyecta la estimación del estado actual hacia adelante en el tiempo. La actualización de la medida

Predicción

Predicción del estado

$$\hat{x}(t|t-1) = A(t)\,\hat{x}(t-1|t-1) + B(t)\,u(t)$$

Covarianza de la predicción estimada

$$P(t|t-1) = A(t)\,P(t-1|t-1)\,A^T(t) + Q(t)$$

Actualización

Innovación o medida residual

$$\tilde{y}(t) = z(t) - H(t)\,\hat{x}(t|t-1)$$

Covarianza de innovación (o residual)

$$S(t) = H(t)\,P(t|t-1)\,H^T(t) + R(t)$$

Ganancia de Kalman óptima

$$K(t) = P(t|t-1)\,H^T(t)\,S^{-1}(t)$$

Estimación de la actualización del estado

$$\hat{x}(t|t) = \hat{x}(t|t-1) + K(t)\,\tilde{y}(t)$$

Covarianza de la actualización estimada

$$P(t|t) = (I - K(t)\,H(t))\,P(t|t-1)$$

Tabla 3.1: Formulación de las fases de predicción y actualización del filtro de Kalman.

ajusta la estimación proyectada por una medida real, en este instante de tiempo.

En la tabla 3.1 se resume la formulación de las dos fases del filtro de Kalman: las ecuaciones de predicción y las ecuaciones de actualización [Wi06].

La fórmula para la covarianza de la actualización estimada es válida solamente para la ganancia óptima de Kalman. El uso de otros valores de ganancia requiere una fórmula más compleja.

Si el modelo es preciso, y los valores para $\hat{x}(0|0)$ y $P(0|0)$ reflejan exactamente la distribución de los valores del estado inicial, entonces se mantienen los siguientes invariantes: todas las estimaciones tienen error de media cero.

- $E[x(t) - \hat{x}(t|t)] = E[x(t) - \hat{x}(t|t-1)] = 0$.
- $E[\tilde{y}(t)] = 0$.

y las matrices de covarianza reflejan exactamente la covarianza de las estimaciones.

- $P(t|t) = cov(x(t) - \hat{x}(t|t))$.

- $P(t|t-1) = cov(x(t) - \hat{x}(t|t-1))$.
- $S(t) = cov(\tilde{y}(t))$.

3.4. El Filtro de Kalman Discreto

El filtro de Kalman aborda el problema general de intentar estimar el estado $x \in \Re^n$ de un proceso controlado en tiempo discreto [We01]. La probabilidad de transición de estado $p(x(t+1)|u(t), x(t))$ debe ser una función lineal de sus argumentos, con ruido gaussiano añadido [Th05]. Dicha estimación es dirigida por la ecuación de diferencia lineal estocástica ([We01]):

$$x(t+1) = A(t)x(t) + B(t)u(t) + v(t) \tag{3.5}$$

con una medición $z \in \Re^m$. La probabilidad de medición $p(z(t)|x(t))$ debe ser también lineal según sus argumentos, con ruido Gaussiano añadido [Th05]. La medición se representa como ([We01]):

$$z(t) = H(t)x(t) + w(t) \tag{3.6}$$

$x(t+1)$ y $x(t)$ son vectores de estado, y $u(t)$ es el vector de control en el instante t [Th05]. Ambos vectores son vectores verticales, de la forma:

$$x(t) = \begin{pmatrix} x_{1,t} \\ x_{2,t} \\ \vdots \\ x_{n,t} \end{pmatrix} \quad \text{y} \quad u(t) = \begin{pmatrix} u_{1,t} \\ u_{2,t} \\ \vdots \\ u_{m,t} \end{pmatrix}$$

La matriz cuadrada A $(A(t))$ de tamaño nxn, donde n es la dimensión del vector de estado $x(t)$, relaciona el estado en el paso previo *t-1* con el estado del paso actual t en la ecuación de diferencia 3.5, en ausencia de una función dirigida o en ausencia del ruido de proceso (modelo de transición de estado) [We01]. La matriz B $(B(t))$ de tamaño nxm, siendo m la dimensión del vector de control $u(t)$, relaciona la entrada de control opcional $u \in \Re$ con el estado x (modelo de las entradas de control). Multiplicando el vector de estado y el vector de control con las matrices $A(t)$ y $B(t)$, respectivamente, la función de transición de estado se convierte en lineal según sus argumentos. De este modo, el filtro de Kalman supone lineal la dinámica del sistema [Th05].

La matriz H $(H(t))$ de tamaño lxn, donde l es la dimensión del vector de medidas $z(t)$, en la ecuación de medida 3.6, relaciona el estado con la medida

$z(t)$ (modelo de observación) [We01].

La variable aleatoria $v(t)$ en la ecuación 3.5 representan el ruido de proceso [Th05]. Dicha variable es un vector aleatorio gaussiano que modela la incertidumbre introducida por la transición del estado. Posee la misma dimensión que el vector de estado. Su media es cero y su covarianza se denota $Q(t)$.

$$\begin{aligned} E[v(k)] &= 0 \qquad \forall k \\ E\left[v(k)\, v^T(j)\right] &= Q(k)\, \delta_{k,j} \end{aligned}$$

donde $\delta_{k,j}$ es la delta de Kronecker [So85]. La matriz $Q(k)$ se asume como definida no negativa, así es posible que $v(k) = 0$.

El vector $w(t)$ es una secuencia aleatoria aditiva, que describe el ruido de medida. La distribución de $w(t)$ es Gaussiana multivariada con media cero y covarianza $R(t)$.

$$\begin{aligned} E[w(k)] &= 0 \qquad \forall k \\ E\left[w(k)\, w^T(j)\right] &= R(k)\, \delta_{k,j} \end{aligned}$$

La matriz $R(k)$ se asume como definida no negativa a menos que se fije de otra manera.

Ambas variables se asumen como independientes, blancas, y con distribuciones de probabilidad normales (distribución normal de media cero) [We01].

$$p(v) \sim N(0, Q) \tag{3.7}$$

$$p(w) \sim N(0, R) \tag{3.8}$$

Además, se supone que las variables aleatorios $w(k)$ y $v(k)$ no están correlacionadas [So85]. Estas variables serán llamadas *secuencias de ruido blanco.*

$$\begin{aligned} E\left[w(k)\, v^T(j)\right] &= 0 \quad \forall k, j \\ E\left[w(k)\, x^T(0)\right] &= 0 \quad \forall k, j \end{aligned}$$

La transición de estados se describe por una matriz de transición $A(t)$, y las entradas de control son mapeadas en el espacio de estados por la matriz $B(t)$ [We01]. La observación es descrita por la matriz de observación $H(t)$ [Cs97].

En la práctica, la matriz de *covarianza del ruido de proceso* Q y la matriz de *covarianza del ruido de medida* R podrían cambiar con cada intervalo de

tiempo o medida.

Definimos $\hat{x}(t) \in \Re^n$ como el estado estimado en el instante t dada la medida $z(t)$. Una estimación en el instante *t+1* puede ser realizada mediante la obtención de los valores esperados de la ecuación 3.5, condicionado sobre las primeras t observaciones.

$$\hat{x}(t+1|t) = A(t)\,\hat{x}(t|t) + B(t)\,u(t) \tag{3.9}$$

La estimación debe ser "mejor"([So85]), en el sentido de que el valor esperado de la suma del cuadrado del error en la estimación es mínima. Esto es, $\hat{x}(k)$ se elige de manera que

$$E\left[(\hat{x}(k) - x(k))^T (\hat{x}(k) - x(k))\right] \quad = \quad \text{mínimo}$$

Podemos definir el error estimado del vector de estado del vehículo ([We01]) como

$$e(t) \equiv \tilde{x}(t) = x(t) - \hat{x}(t)$$

y la covarianza del error estimado como

$$P(t) = E\left[e(t)\,e^T(t)\right] \equiv E\left[\tilde{x}(t)\,\tilde{x}^T(t)\right] \tag{3.10}$$

Ya que el modelo de proceso es lineal ([Cs97]), la predicción de la matriz de covarianza puede ser expresada como:

$$\begin{aligned}
\tilde{x}(t+1|t) &= x(t+1) - \hat{x}(t+1|t) \\
&= A(t)\,x(t) + B(t)\,u(t) + v(t) - [A(t)\,\hat{x}(t|t) + B(t)\,u(t)] \\
&= A(t)\,\tilde{x}(t|t) + v(t)
\end{aligned}$$

$$\begin{aligned}
P(t+1|t) &= E\left[\tilde{x}(t+1|t)\,\tilde{x}^T(t+1|t)\right] \\
&= E\left[A(t)\,\tilde{x}(t|t)\,\tilde{x}^T(t|t)\,A^T(t)\right] + E\left[v(t)\,v^T(t)\right] \\
&\quad + E\left[A(t)\,\tilde{x}(t|t)\,v^T(t)\right] + E\left[v(t)\,\tilde{x}^T(t|t)\,A^T(t)\right] \\
&= A(t)\,P(t|t)\,A^T(t) + Q(t)
\end{aligned}$$

donde $\tilde{x}(t|t)$ y $v(t)$ se asumen no correlacionadas. El estado predicho permite que la observación en el instante *t+1* sea predicha. El vector de innovación $y(t+1)$ se define como la diferencia entre la observación real en el instante *t+1* y la observación predicha.

$$\begin{aligned}\hat{z}\,(t+1|t) &= E\,[z\,(t+1)\,|Z^t] \\ &= E\,[H\,(t+1)\,x\,(t+1)\,|Z^t] \\ &= H\,(t+1)\,\hat{x}\,(t+1|t)\end{aligned}$$

$$\begin{aligned}y\,(t+1) &= z\,(t+1) - \hat{z}\,(t+1|t) \\ &= z\,(t+1) - H\,(t+1)\,\hat{x}\,(t+1|t)\end{aligned}$$

Si tenemos en cuenta que ([We01]):

$$E\,[x\,(t)] = \hat{x}\,(t)$$

$$E\left[e\,(t)\,e^T\,(t)\right] = E\left[(x\,(t) - \hat{x}\,(t))\,(x\,(t) - \hat{x}\,(t))^T\right] = P\,(t)$$

Podemos deducir que la ecuación de la covarianza de la estimación del error (ecuación 3.10) refleja la varianza de la distribución del estado.

$$p\,(x\,(t)\,|z\,(t)) \sim N\left(E\,[x\,(t)]\,, E\left[(x\,(t) - \hat{x}\,(t))\,(x\,(t) - \hat{x}\,(t))^T\right]\right) = N\,(\hat{x}\,(t)\,, P\,(t))$$

El filtro de Kalman estima un proceso usando la forma de control retroactivo: el filtro estima el estado de proceso en algún instante de tiempo, y luego obtiene la realimentación en forma de medidas.

La realimentación de la estimación es igual a la suma ponderada de la observación y la predicción

$$\hat{x}\,(t+1|t+1) = \hat{x}\,(t+1|t) + K\,(t+1)\,y\,(t+1) \tag{3.11}$$

donde $K\,(t)$ es la matriz de ganancia de Kalman que determina la influencia de la innovación sobre la estimación de realimentación [Cs97]. La matriz $K\,(t)$ se determinará de manera que $E\left[(\hat{x}\,(t) - x\,(t))^T\,(\hat{x}\,(t) - x\,(t))\right]$ sea minimizado [So85].

El error en la realimentación de la estimación ([Cs97]) es por lo tanto

$$\begin{aligned}
\tilde{x}\left(t+1|t+1\right) &= x\left(t+1\right)-\hat{x}\left(t+1|t+1\right)\\
&= x\left(t+1\right)-\left[\hat{x}\left(t+1|t\right)+K\left(t+1\right)y\left(t+1\right)\right]\\
&= \tilde{x}\left(t+1|t\right)-K\left(t+1\right)y\left(t+1\right)
\end{aligned}$$

$$\begin{aligned}
P\left(t+1|t+1\right) &= E\left[\tilde{x}\left(t+1|t+1\right)\tilde{x}^{T}\left(t+1|t+1\right)\right]\\
&= E\left[\left[\tilde{x}\left(t+1|t\right)-K\left(t+1\right)y\left(t+1\right)\right]\left[\tilde{x}\left(t+1|t\right)-K\left(t+1\right)y\left(t+1\right)\right]^{T}\right]\\
&= E\left[\tilde{x}\left(t+1|t\right)\tilde{x}^{T}\left(t+1|t\right)\right]\\
&\quad +E\left[K\left(t+1\right)y\left(t+1\right)y^{T}\left(t+1\right)K^{T}\left(t+1\right)\right]\\
&\quad -E\left[\tilde{x}\left(t+1|t\right)y^{T}\left(t+1\right)K^{T}\left(t+1\right)\right]\\
&\quad -E\left[K\left(t+1\right)y\left(t+1\right)\tilde{x}^{T}\left(t+1|t\right)\right]\\
&= P\left(t+1|t\right)+K\left(t+1\right)S_{yy}\left(t+1|t\right)K^{T}\left(t+1\right)\\
&\quad -S_{xy}\left(t+1|t\right)K^{T}\left(t+1\right)-K\left(t+1\right)S_{yx}\left(t+1|t\right)
\end{aligned} \tag{3.12}$$

donde

$$\begin{aligned}
S_{yy}\left(t+1|t\right) &= E\left[y\left(t+1\right)y^{T}\left(t+1\right)\right]\\
S_{xy}\left(t+1|t\right) &= E\left[\tilde{x}\left(t+1|t\right)y^{T}\left(t+1\right)\right]\\
&= E\left[y\left(t+1\right)\tilde{x}^{T}\left(t+1|t\right)\right]^{T}\\
&= S_{xy}^{T}\left(t+1|t\right)
\end{aligned}$$

$S_{yy}\left(t+1|t\right)$ es la covarianza de la innovación, y $S_{xy}\left(t+1|t\right)$ es la covarianza del error de predicción e innovación. La matriz de ganancia se elige para reducir al mínimo el error cuadrático medio esperado, $E\left[\tilde{x}^{T}\left(t+1|t+1\right)\tilde{x}\left(t+1|t+1\right)\right]$, el cual es equivalente a la traza de $P\left(t+1|t+1\right)$. La solución para $K\left(t+1\right)$ es

$$K\left(t+1\right) = S_{xy}\left(t+1|t\right)S_{yy}^{-1}\left(t+1|t\right) \tag{3.13}$$

El filtro de Kalman calcula la realimentación óptima, en el sentido menos equitativo, para una estimación del estado basada en la covarianza del error de la estimación del estado y de la innovación, y la covarianza de la innovación.

La ecuación de la realimentación de Kalman para la estimación del estado y la covarianza de la estimación son por lo tanto

$$\hat{x}(t+1|t+1) = \hat{x}(t+1|t) + K(t+1)\, y(t+1) \quad (3.14)$$
$$P(t+1|t+1) = P(t+1|t) - K(t+1)\, S_{yy}(t+1|t)\, K^T(t+1) \quad (3.15)$$

Las ecuaciones 3.11-3.15 indican que el efecto del modelo de la observación en el filtro de Kalman se deben a la innovación y las covarianzas de la innovación, más que por la observación en sí misma o la manera en que se forman.

3.5. El Filtro de Kalman Extendido

Las suposiciones de que las observaciones son funciones lineales del estado y que el siguiente estado es una función lineal del estado previo, son cruciales para la exactitud del filtro de Kalman [Th05]. La observación de que cualquier transformación lineal de una variable aleatoria Gaussiana, ocasiona otra variable aleatoria Gaussiana juega un papel importante en la derivación del algoritmo del filtro de Kalman.

Desafortunadamente, las transiciones de estado y las medidas son, en la práctica, rara vez lineales [Th05]. Por ejemplo, un robot que se mueve con velocidad de traslación constante y con velocidad de rotación constante, por regla general se mueve en una trayectoria circular, la cual no puede ser descrita por transiciones de estado lineales.

Para estimar el estado de un sistema no lineal [Cs97], se usa ampliamente el *filtro de Kalman extendido*, o EKF[1]. El funcionamiento básico del EKF es el mismo que el del filtro de Kalman discreto lineal [We01]; En el caso no lineal, asumimos que nuestro proceso posee de nuevo un vector de estado $x \in \Re^n$, el sistema es descrito por modelos de proceso y observación, guiados por la ecuación de diferencia no lineal estocástica, de la forma:

$$x(t+1) = f(x(t), u(t), v(t)). \quad (3.16)$$

con una medición $z \in \Re^m$ tal que

$$z(t) = h(x(t), w(t)). \quad (3.17)$$

Este modelo generaliza estrictamente el modelo gaussiano lineal, que es la base del filtro de Kalman, como se ha supuesto en las ecuaciones 3.5 y

[1] *Extended Kalman Filter*, Filtro de Kalman Extendido

3.6 [Th05]. La función no lineal *f*, descrita en la ecuación 3.16, reemplaza las matrices $A(t)$ y $B(t)$ en 3.5, relacionando el estado del paso actual t con el estado del paso siguiente *t+1* [We01]. Incluye como parámetros cualquier función de control $u(t)$ y el ruido de proceso de media cero $v(t)$ con momentos preescritos como ([So85]):

$$\begin{aligned} E[v(t)] &= 0 & \forall t \\ E\left[v(t)\, v^T(\tau)\right] &= Q(t)\,\delta(t-\tau) & \forall t, \tau \end{aligned}$$

donde $Q(t)$ es una matriz simétrica definida no negativa, y $\delta(t-\tau)$ representa la función delta de Dirac.

La función no lineal *h*, en la ecuación 3.17, reemplaza la matriz $H(t)$ en 3.6, relacionando el estado $x(t)$ con la medición $z(t)$ [We01]. Además, dicha función incluye como parámetro el ruido de proceso de media cero $w(t)$ con momentos preescritos como ([So85]):

$$\begin{aligned} E[w(t)] &= 0 & \forall t \\ E\left[w(t)\, w^T(\tau)\right] &= R(t)\,\delta(t-\tau) & \forall t, \tau \end{aligned}$$

siendo $R(t)$ una matriz simétrica definida no negativa, y $\delta(t-\tau)$, al igual que antes, representa la función delta de Dirac.

Las variables aleatorias $v(t)$ y $w(t)$ representan de nuevo el ruido de proceso y el ruido de medida, respectivamente, como en la ecuación 3.5 y la ecuación 3.6 [Th05].

El EKF asume que los modelos de proceso y observación son localmente lineales, $v(t)$ y $w(t)$ son pequeños, y que $\hat{x}(t|t) \approx E[x(t)|Z^t]$ [We01].

En la práctica, desde luego, no conocemos los valores individuales del ruido $v(t)$ y $w(t)$ en cada instante. Sin embargo, podemos aproximar el vector de estado y el vector de medida sin ellos, como

$$\tilde{x}(t+1) = f(\hat{x}(t), u(t), 0). \tag{3.18}$$

con una medición $z \in \Re^m$ tal que

$$\tilde{z}(t) = h(\tilde{x}(t), 0). \tag{3.19}$$

donde $\hat{x}(t)$ es alguna estimación del estado en el instante t.

Es importante comentar que un defecto fundamental del EKF es que las distribuciones de varias variables aleatorias no son más normales después de sus respectivas transformaciones no lineales. El EKF es simplemente un estimador de estado en sí mismo, que sólo aproxima la optimalidad de la regla de Bayes mediante la linealización.

EKF se asemeja en algo a las series de Taylor, pues, podemos linealizar la estimación alrededor de la estimación actual usando las derivadas parciales de las funciones de proceso y las funciones de medida para calcular las estimaciones que se encuentran aún ante una relación no lineal. El modelo de proceso ([Cs97]) es linealizado como una extensión de la serie de Taylor sobre $\tilde{x}(t|t)$,

$$x(t+1) = f[\hat{x}(t|t), u(t), 0] + \nabla f_x \tilde{x}(t|t) + \nabla f_v v(t) \tag{3.20}$$

donde ∇f_x es el jacobiano de f con respecto a x, evaluado en $\tilde{x}(t|t)$, y ∇f_v es el jacobiano de f con respecto a v evaluado en $v(t)$. Usando las suposiciones de que los términos de mayor orden en la extensión de la serie de Taylor para $x(t+1)$ son insignificantes, y que $\tilde{x}(t|t)$ y $v(t)$ son variables aleatorias pequeñas de media cero, conduce a las ecuaciones de predicción siguientes:

$$\begin{aligned}
\hat{x}(t+1|t) &= E[x(t+1)|Z_t] \\
&= E[f[\tilde{x}(t|t), u(t), 0] + \nabla f_x \tilde{x}(t|t) + \nabla f_v v(t)] \\
&\approx f[\tilde{x}(t|t), u(t), 0] \\
\\
\tilde{x}(t+1|t) &= x(t+1) - \hat{x}(t+1) \\
&\approx \nabla f_x \tilde{x}(t|t) + \nabla f_v v(t) \\
\\
P(t+1|t) &= E\left[\tilde{x}(t+1|t)\,\tilde{x}^T(t+1|t)\right] \\
&\approx \nabla f_x P(t|t) \nabla f_x^T + \nabla f_v Q(t) \nabla f_v^T
\end{aligned}$$

Las ecuaciones de realimentación se forman mediante la linealización del modelo de observación. El modelo de observación se expresa como una extensión de la serie de Taylor sobre $\tilde{x}(t+1|t)$,

$$z(t+1) = h[\hat{x}(t+1|t), 0] + \nabla h_x \tilde{x}(t+1|t) + \nabla h_w w(t+1) \tag{3.21}$$

donde ∇h_x es el jacobiano de h con respecto a x, evaluado en $\tilde{x}(t+1|t)$, y ∇h_w es el jacobiano de h con respecto a w evaluado en $w(t+1)$. Usando la suposición

$$\hat{x}(t+1|t) \approx E[x(t+1)|Z_t],$$

y las suposiciones de que $\hat{x}(t+1|t)$ y $w(t+1)$ son variables aleatorias pequeñas y de media cero, y que los términos de mayor orden en la serie de Taylor son insignificantes, conduce a las siguientes expresiones,

$$\begin{aligned}
\hat{z}(t+1|t) &= E\left[h\left[\hat{x}(t+1|t),0\right]+\nabla h_x\tilde{x}(t+1|t)+\nabla h_w w(t+1)\right] \\
&= h\left[\hat{x}(t+1|t),0\right]
\end{aligned}$$

$$\begin{aligned}
y(t+1) &= z(t+1)-\hat{z}(t+1|t) \\
&\approx \nabla h_x\tilde{x}(t+1|t)+\nabla h_w w(t+1) \qquad (3.22)
\end{aligned}$$

$$\begin{aligned}
S_{yy}(t+1|t) &= E\left[y(t+1)\,y^T(t+1)\right] \\
&\approx \nabla h_x P(t+1|t)\,\nabla h_x^T+\nabla h_w R(t+1)\,\nabla h_w^T \qquad (3.23)
\end{aligned}$$

$$\begin{aligned}
S_{xy}(t+1|t) &= E\left[\tilde{x}(t+1|t)\,y^T(t+1)\right] \\
&\approx E\left[\tilde{x}(t+1|t)\left[\nabla h_x\tilde{x}(t+1|t)+\nabla h_w w(t+1)\right]^T\right] \\
&= P(t+1|t)\,\nabla h_x^T, \qquad (3.24)
\end{aligned}$$

donde $\tilde{x}(t+1|t)$ y $w(t+1)$ se asumen otra vez no correlacionados.

Las ecuaciones 3.11-3.15 aún aplicadas en el caso no lineal, indican que el efecto del modelo de observación en el EKF es, como en el caso lineal, debido a la innovación y las covarianzas de la innovación.

3.6. El Filtro UKF

En el trabajo de Julie y Uhlmann ([Ju97]) se plantea una solución mejorada del EKF, denominada UKF ("Unscented Kalman Filter"). En líneas generales, permite conseguir una mejor aproximación de los modelos no lineales implicados en el estimador, que los obtenidos mediante el uso de una serie de Taylor que se emplean en el EKF.

Cuando el estado de transición y los modelos de observación (funciones de predicción y actualización) son altamente no lineales, el filtro de Kalman extendido (EKF) puede dar un resultado particularmente pobre. Esto se debe a que la media y la covarianza se propagan a través de linealización del modelo no lineal subyacente. El UKF, en lugar de "linealizar" el problema, utiliza una técnica de muestreo determinista del estado actual conocida

como la transformada “sin pistas”, para recoger un conjunto mínimo de puntos de muestreo (llamados puntos sigma) alrededor de la media. Estos puntos sigma se propagan entonces a través de las funciones no lineales, de los cuales la media y la covarianza de la estimación se restablecen. El resultado es un filtro que captura con más precisión la verdadera media y covarianza. Además, esta técnica elimina el requisito de calcular jacobianos explícitamente, los cuales para las funciones complejas puede ser una tarea difícil en sí mismo (requieren complicadas derivadas si se hace de forma analítica, o son computacionalmente costosos si se hace numéricamente).

Se puede considerar el filtro UKF como el resultado de incorporar la UT al filtro EKF, para mejorar las aproximaciones que se hacen de los dos primeros momentos de una variable aleatoria, que resulta de propagar otra variable aleatoria (supuesta gaussiana) a través de una transformación no lineal.

Las etapas principales del filtro UKF son las mismas que en el algoritmo Ekf-slam.

- Predicción: Como en el EKF, la predicción UKF puede ser utilizada independientemente de la actualización UKF, en combinación con una actualización lineal (o incluso EKF), o viceversa. El estado estimado y la covarianza son aumentadas con la media y la covarianza del ruido de proceso.
- Actualización: El estado predicho y la covarianza se aumentan como antes, sólo que ahora con la media y la covarianza del ruido de medición.

Capítulo 4

SLAM basado en Filtros

4.1. Introducción

Como ya se ha indicado, en este proyecto se ha limitado el análisis de las técnicas de SLAM para definir un alcance razonable para el mismo.

Este capítulo estudia la aplicación del filtros estimadores al problema de la localización de la posición de un vehículo y la construcción del mapa del entorno de forma simultánea [Cs97].

Dentro de este ámbito, se han seleccionado tres algoritmos de SLAM para su análisis y comparación: el EKF-Slam, el UKF-Slam y el FastSlam.

4.2. El Estado Aumentado

Para modelar totalmente el problema del SLAM el vector de estado del vehículo se debe aumentar con los vectores de estado de todas las características trazadas [Cs97]. Esto es necesario para seguir la pista a todas las correlaciones entre el vehículo y los errores de la estimación de la característica. Estas correlaciones son esenciales para la solución del problema del SLAM.

En el instante k, el estado del vehículo se denota por $x_v(k)$ y el estado de la característica p_i se denota por $p_i(k)$.

El mapa del estado aumentado del vehículo, $x_{mav}(k)$, describiendo el estado del vehículo y todas las características en el entorno en el instante k se define como:

$$x_{mav}(k) \cong \left[x_v^T(k)\ p_1^T(k)\ \dots\ p_N^T(k)\right]^T \tag{4.1}$$

La estimación $\hat{x}_{mav}(k|k)$ del estado aumentado y su correspondiente covarianza $P_{mav}(k|k)$ son:

$$\begin{aligned}
\hat{x}_{mav}(k|k) &= \left[\hat{x}_v^T(k|k)\ \hat{p}_1^T(k|k)\ \dots\ \hat{p}_N^T(k|k)\right]^T \\
P_{mav}(k|k) &= E\left[\tilde{x}_{mav}(k|k)\,\tilde{x}_{mav}^T(k|k)\right] \\
&= E\left[\begin{bmatrix}\tilde{x}_v(k|k)\\ \tilde{p}_1(k|k)\\ \vdots \\ \tilde{p}_N(k|k)\end{bmatrix}\begin{bmatrix}\tilde{x}_v^T(k|k)\\ \tilde{p}_1^T(k|k)\\ \vdots \\ \tilde{p}_N^T(k|k)\end{bmatrix}^T\right] \\
&= \begin{bmatrix} P_{vv}(k|k) & P_{v1}(k|k) & \dots & P_{vN}(k|k) \\ P_{1v}(k|k) & P_{11}(k|k) & \dots & P_{1N}(k|k) \\ \vdots & \vdots & \ddots & \vdots \\ P_{Nv}(k|k) & P_{N1}(k|k) & \dots & P_{NN}(k|k)\end{bmatrix}
\end{aligned} \tag{4.2}$$

donde

$$\begin{aligned}
P_{vv}(k|k) &= E\left[\tilde{x}_v(k|k)\,\tilde{x}_v^T(k|k)\right] \\
P_{ii}(k|k) &= E\left[\tilde{p}_i(k|k)\,\tilde{p}_i^T(k|k)\right] \\
P_{vi}(k|k) &= E\left[\tilde{x}_v(k|k)\,\tilde{p}_i^T(k|k)\right] \\
P_{ij}(k|k) &= E\left[\tilde{p}_i(k|k)\,\tilde{p}_j^T(k|k)\right]
\end{aligned}$$

Los términos $P_{vv}(k|k)$ y $P_{ii}(k|k)$ para $i = 1, \dots, N$ son las matrices de covarianzas para el vehículo y las N características estimadas, respectivamente. Los términos $P_{vi}(k|k) = P_{iv}^T(k|k)$ miden la correlación entre el error en el vehículo y la característica estimada, para cada característica p_i, y los términos $P_{ij}(k|k) = P_{ji}^T(k|k)$ mide la correlación entre los errores en las estimaciones de dos características p_i y p_j.

La solución particular del filtro de Kalman para el problema del SLAM desarrollado a partir de la formulación descrita en el apartado A.1 del anexo A, es lo que se conoce como filtro de Kalman del Mapa Aumentado (MAK[1]).

[1] *Map Augmented Kalman*, Kalman del Mapa Aumentado

4.2.1. Importancia de las Correlaciones

El mantenimiento de todos los términos de la matriz de covarianza de la ecuación 4.2 es fundamental para la solución del problema del SLAM. El estado aumentado permite al filtro MAK, de forma explícita, mantener correlaciones entre errores en el vehículo y la característica, y entre la característica y las estimaciones de la característica. El significado de esas correlaciones se pueden examinar considerando una actualización obtenida de una observación de la característica p_i.

El estado aumentado de la característica del vehículo, la matriz de covarianza y el modelo de observación se dan a continuación con los índices de tiempo, que se encuentran de nuevo dentro de una expresión escrita, como índices de tiempo externos, para mejorar la claridad:

$$
\begin{aligned}
\hat{x}_{mav}(k+1|k) &= \begin{bmatrix} \hat{x}_v^T & \dots & \hat{p}_i^T & \dots & \hat{p}_j^T & \dots & \hat{p}_N^T \end{bmatrix}^T (k|k) \\
P_{mav}(k|k) &= \begin{bmatrix} P_{vv} & \dots & P_{vi} & \dots & P_{vj} & \dots & P_{vN} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ P_{iv} & \dots & P_{ii} & \dots & P_{ij} & \dots & P_{iN} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ P_{jv} & \dots & P_{ji} & \dots & P_{jj} & \dots & P_{jN} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ P_{Nv} & \dots & P_{Ni} & \dots & P_{Nj} & \dots & P_{NN} \end{bmatrix} (k|k) \\
H_i(k+1) &= \begin{bmatrix} H_v & 0 & \dots & 0 & H_p & 0 & \dots \end{bmatrix} (k+1).
\end{aligned}
$$

A partir de la ecuación 3.23, la varianza de la innovación es

$$
\begin{aligned}
S_i\left(k+1|k\right) &= H_i\left(k+1\right)P_{mav}\left(k+1|k\right)H_i^T\left(k+1\right) \\
&\quad +H_w\left(k+1\right)R\left(k+1\right)H_w^T\left(k+1\right) \\
&= H_v\left(k+1\right)P_{vv}\left(k+1|k\right)H_v^T\left(k+1\right) \\
&\quad +H_p\left(k+1\right)P_{ii}\left(k+1|k\right)H_p^T\left(k+1\right) \\
&\quad +H_v\left(k+1\right)P_{vi}\left(k+1|k\right)H_p^T\left(k+1\right) \\
&\quad +H_p\left(k+1\right)P_{iv}\left(k+1|k\right)H_v^T\left(k+1\right) \\
&\quad +H_w\left(k+1\right)R\left(k+1\right)H_w^T\left(k+1\right) \\
&= \nabla\left(h_{x_v,p}\right)_{x_v}P_{vv}\left(k+1|k\right)\nabla\left(h_{x_v,p}\right)_{x_v}^T \\
&\quad +\nabla\left(h_{x_v,p}\right)_{p_i}P_{ii}\left(k+1|k\right)\nabla\left(h_{x_v,p}\right)_{p_i}^T \\
&\quad +\nabla\left(h_{x_v,p}\right)_{x_v}P_{vi}\left(k+1|k\right)\nabla\left(h_{x_v,p}\right)_{p_i}^T \\
&\quad +\nabla\left(h_{x_v,p}\right)_{p_i}P_{iv}\left(k+1|k\right)\nabla\left(h_{x_v,p}\right)_{x_v}^T \\
&\quad +\nabla\left(h_{x_v,p}\right)_{w}R\left(k+1\right)\nabla\left(h_{x_v,p}\right)_{w}^T,
\end{aligned}
$$

la cuál se reconoce por estar en la forma de $S_{yy}\left(k+1|k\right)$. Por lo tanto, la solución del filtro MAK para el problema del SLAM calcula correctamente la covarianza de la innovación. Ya que cualquiera de las características p_i, $i = 1, 2, \ldots, N$, pueden ser observadas, todos los términos de la covarianza $P_{vi}\left(k+1|k\right)$, $i = 1, 2, \ldots, N$, deben ser conocidos, de modo que la covarianza de la innovación siempre pueda ser calculada. Una matriz de covarianza que mantiene los términos $P_{vi}\left(k+1|k\right)$, $i = 1, 2, \ldots, N$, deben mantener también los términos $P_{ij}\left(k+1|k\right)$, $i = 1, 2, \ldots, N$ y $j = 1, 2, \ldots, N$, para completar la matriz de covarianza, de lo contrario, la condición $P\left(k+1|k\right) = E\left[\tilde{x}\left(k+1|k\right)\tilde{x}^T\left(k+1|k\right)\right]$ no se satisface.

La formulación detallada puede consultarse en el apartado A.2 del anexo A.

4.3. El algoritmo general Ekf-slam

El primer algoritmo SLAM, y quizás el más influyente, se basa históricamente en el filtro de Kalman extendido, o EKF. En pocas palabras, el algoritmo Ekf-slam aplica el EKF para el SLAM en línea, usando la asociación de datos de probabilidad máxima [Th05]. De hecho, el Ekf-slam está sometido

a un número de aproximaciones y suposiciones limitadas.

Los mapas, en Ekf-slam, están *basados en características*, y están compuestos de puntos de referencia. Por motivo computacional, el número de puntos de referencia es generalmente pequeño (menor de 1000). Además, la aproximación del EKF suele trabajar bien con los puntos de referencia menos ambiguos. Por esta razón, el Ekf-slam necesita considerable ingeniería de detectores de características, a veces utilizando hitos artificiales como características.

Como cualquier algoritmo EKF, el Ekf-slam *supone* ruido Gaussiano para el movimiento y percepción del robot. La cantidad de incertidumbre en el siguiente estado debe ser relativamente pequeño, ya que de otra manera la linealización en EKF tiende a introducir errores intolerables.

El algoritmo Ekf-slam puede procesar sólo observaciones *positivas* de los puntos de referencia. No puede procesar información negativa que surja, debido a la falta de puntos de referencia en las medidas del sensor.

Posiblemente, el problema de mayor dificultad se trate de la asociación de correspondencias [Ro04]. Este problema trata de determinar si diferentes mediciones realizadas en diferentes instantes, corresponden a un mismo objeto físico. Para asociar las correspondencias, existen dos formas: correspondencia conocida y correspondencia no conocida. En la primera, se conoce en todo momento la correspondencia unívoca entre los hitos referenciales observados en el entorno y los presentes en el mapa que se está construyendo. De este modo se podrá determinar la observación de un nuevo hito o bien la observación de uno ya tratado anteriormente. Cuando se afronta el problema del SLAM en situaciones reales, es muy raro conocer la correspondencia entre las observaciones, los elementos del mapa y el número total de elementos que integran el entorno, utilizando para ello la correspondencia no conocida. En este tipo de correspondencia, cada vez que el robot realiza una observación, esa lectura debe asociarse a un elemento del mapa, o bien considerar que proviene de un elemento no visto anteriormente.

4.3.1. Ekf-slam con Correspondencia Conocida

El algoritmo SLAM, para el caso con correspondencia conocida, direcciona sólo la parte continua del problema del SLAM [Th05]. Su desarrollo es en muchos sentidos paralelo a la derivación del algoritmo de localización EKF, pero con una diferencia clave: además de la estimación de la ubicación del

robot $x(t)$, el algoritmo del Ekf-slam también estima las coordenadas de todos los puntos de referencia encontrados a lo largo del camino. Esto hace necesario incluir las coordenadas de los puntos de referencia en el vector de estado.

Por conveniencia, se permite llamar al vector de estado, que consta de la ubicación del robot y el mapa, el *vector de estado combinado*, y se denota a éste como $y(t)$. El vector combinado es dado como:

$$y(t) = \begin{pmatrix} x(t) \\ m \end{pmatrix} \tag{4.3}$$

$$= (x\ y\ \theta\ m_{1,x}\ m_{1,y}\ s_1\ \cdots\ m_{N,x}\ m_{N,y}\ s_N)^T \tag{4.4}$$

x, y y θ denotan las coordenadas del robot en el instante t (no se deben confundir con las variables de estado $x(t)$ e $y(t)$), $m_{i,x}$, $m_{i,y}$ son las coordenadas del punto de referencia i-ésimo, para $i = 1, \ldots, N$, y s_i es su característica. La dimensión de este vector de estado es $3N + 3$, donde N denota el número de puntos de referencia en el mapa. El Ekf-slam calcula el siguiente estado en línea $p(y(t) | z(1:t)\, u(1:t))$.

1. **Algoritmo Ekf-slam con Correspondencia Conocida**$(\mu(t-1), \sum(t-1), u(t), z(t), c(t))$:
2. $F(x) = \begin{pmatrix} 1 & 0 & 0 & 0 \cdots 0 \\ 0 & 1 & 0 & 0 \cdots 0 \\ 0 & 0 & 1 & \underbrace{0 \cdots 0}_{3N} \end{pmatrix}$
3. $\overline{\mu}(t) = \mu(t-1)$
   $+ F(x)^T \begin{pmatrix} -\frac{v(t)}{w(t)} sin\mu(t-1,\theta) + \frac{v(t)}{w(t)} sin(\mu(t-1,\theta) + w(t)\Delta(t)) \\ \frac{v(t)}{w(t)} cos\mu(t-1,\theta) - \frac{v(t)}{w(t)} cos(\mu(t-1,\theta) + w(t)\Delta(t)) \\ w(t)\Delta(t) \end{pmatrix}$
4. $G(t) = I$
   $+F(x)^T \begin{pmatrix} 0 & 0 & \frac{v(t)}{w(t)} cos\mu(t-1,\theta) - \frac{v(t)}{w(t)} cos(\mu(t-1,\theta) + w(t)\Delta(t)) \\ 0 & 0 & \frac{v(t)}{w(t)} sin\mu(t-1,\theta) - \frac{v(t)}{w(t)} sin(\mu(t-1,\theta) + w(t)\Delta(t)) \\ 0 & 0 & 0 \end{pmatrix} F(x)$
5. $\overline{\sum}(t) = G(t) \sum(t-1) G(t)^T + F(x)^T R(t) F(x)$

6. $Q(t) = \begin{pmatrix} \sigma_r & 0 & 0 \\ 0 & \sigma_\phi & 0 \\ 0 & 0 & \sigma_s \end{pmatrix}$

7. para todas las características observadas $z(t)^i = \begin{pmatrix} r(t)^i & \phi(t)^i & s(t)^i \end{pmatrix}$ hacer:

8. j $= c(t)^i$

9. si el punto de referencia j no ha sido visto antes

10. $$\begin{pmatrix} \overline{\mu}(j,x) \\ \overline{\mu}(j,y) \\ \overline{\mu}(j,s) \end{pmatrix} = \begin{pmatrix} \overline{\mu}(t,x) \\ \overline{\mu}(t,y) \\ s(t)^i \end{pmatrix} + r(t)^i \begin{pmatrix} cos\left(\phi(t)^i + \overline{\mu}(t,\theta)\right) \\ sin\left(\phi(t)^i + \overline{\mu}(t,\theta)\right) \\ 0 \end{pmatrix}$$

11. fin si

12. $$\delta = \begin{pmatrix} \delta(x) \\ \delta(y) \end{pmatrix} = \begin{pmatrix} \overline{\mu}(j,x) - \overline{\mu}(t,x) \\ \overline{\mu}(j,y) - \overline{\mu}(t,y) \end{pmatrix}$$

13. $q = \delta^T \delta$

14. $$\hat{z}(t)^i = \begin{pmatrix} \sqrt{q} \\ atan2(\delta(y), \delta(x)) - \overline{\mu}(t,\theta) \\ \overline{\mu}(j,s) \end{pmatrix}$$

15. $$F(x,j) = \begin{pmatrix} 1 & 0 & 0 & 0 \cdots 0 & 0 & 0 & 0 & 0 \cdots 0 \\ 0 & 1 & 0 & 0 \cdots 0 & 0 & 0 & 0 & 0 \cdots 0 \\ 0 & 0 & 1 & 0 \cdots 0 & 0 & 0 & 0 & 0 \cdots 0 \\ 0 & 0 & 0 & 0 \cdots 0 & 1 & 0 & 0 & 0 \cdots 0 \\ 0 & 0 & 0 & 0 \cdots 0 & 0 & 1 & 0 & 0 \cdots 0 \\ 0 & 0 & 0 & \underbrace{0 \cdots 0}_{3j-3} & 0 & 0 & 1 & \underbrace{0 \cdots 0}_{3N-3j} \end{pmatrix}$$

16. $$H(t)^i = \frac{1}{q} \begin{pmatrix} \sqrt{q}\delta(x) & -\sqrt{q}\delta(y) & 0 & -\sqrt{q}\delta(x) & \sqrt{q}\delta(y) & 0 \\ \delta(y) & \delta(x) & -1 & -\delta(y) & -\delta(x) & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} F(x,j)$$

17. $$K(t)^i = \overline{\sum}(t) H(t)^{iT} \left(H(t)^i \overline{\sum}(t) H(t)^{iT} + Q(t) \right)^{-1}$$

18. $$\overline{\mu}(t) = \overline{\mu}(t) + K(t)^i \left(z(t)^i - \hat{z}(t)^i \right)$$

19. $$\overline{\sum}(t) = \left(I - K(t)^i H(t)^i \right) \overline{\sum}(t)$$

20. fin para
21. $\mu(t) = \overline{\mu}(t)$
22. $\sum(t) = \overline{\sum}(t)$
23. devolver $\mu(t), \sum(t)$

En el algoritmo Ekf-slam, las líneas 2 a 5 aplican la actualización de movimiento, mientras que las líneas 6 a 20 incorporan el vector de medidas. Las líneas 3 y 5 manipulan la media y la covarianza de la confianza de acuerdo al modelo del movimiento. Las líneas 7 a 20 iteran a través de todas las medidas. La comprobación en la línea 9 devuelve verdadero sólo para puntos de referencia para los cuales no se tiene ninguna estimación de la posición inicial. Para estos, la línea 10 inicializa la posición de tal punto de referencia por la posición proyectada obtenida del rango correspondiente y la medida de relación. Este paso es importante para la linealización en los EKF; no sería necesario en los filtros de Kalman lineales. Para cada medida, se calcula una medida "esperada" en la línea 14, y la correspondiente *ganancia de Kalman* se calcula en la línea 17. Nótese que la ganancia de Kalman es una matriz de tamaño 3 por 3N + 3. Esta matriz no es por lo general dispersa. La información es propagada por la estimación entera del estado. La actualización del filtro se produce en la línea 18 y 19, donde se utilizan las medidas de innovación para reevaluar la incertidumbre.

El hecho de que la ganancia de Kalman esté totalmente poblada por todas las variables de estado (y no solamente el punto de referencia observado y la ubicación del robot) es importante. En el SLAM, observando un punto de referencia, mejora la estimación de ubicación del robot, y por consiguiente, elimina un poco la incertidumbre de puntos de referencia vistos antes por el mismo robot. El efecto extraordinario aquí es que no tenemos que modelar ubicaciones anteriores explícitamente, que nos pondría dentro del problema del SLAM completo y no haría del EKF un algoritmo en tiempo real. En cambio, esta dependencia es capturada en el estado Gaussiano siguiente, más expresamente, en los elementos de covarianza no diagonales de la matriz $\sum(t)$.

La figura 4.1 ilustra el algoritmo EKF aplicado al problema del SLAM en línea, para un ejemplo de forma artificial. El camino del robot es una línea de puntos, y las estimaciones de su propia posición son elipses sombreadas. Ocho señales distinguibles de localización desconocida se muestran como puntos pequeños, y sus estimaciones de localización se muestran como elipses no sombreadas. Entre 4.1a y 4.1c la incertidumbre posicional del robot aumenta,

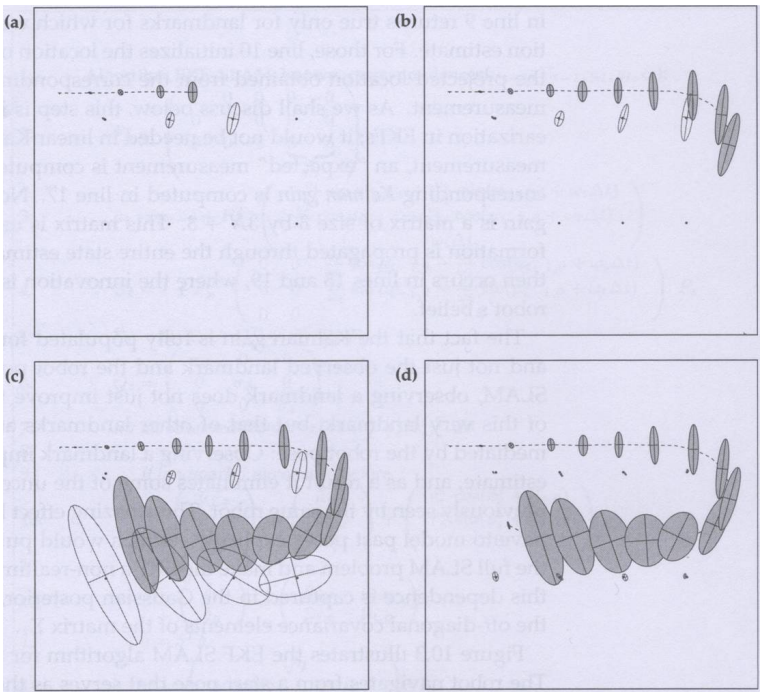


Figura 4.1: EKF aplicado al problema del SLAM en línea.

al igual que su incertidumbre sobre las marcas que encuentra. En 4.1d el robot percibe la primera marca otra vez, y la incertidumbre de todas las marcas disminuye, al igual que la incertidumbre de su ubicación actual. El robot navega a partir de una ubicación inicial, que sirve como el origen de su sistema de coordenadas. Como se mueve, la incertidumbre de su propia ubicación crece, como se indica mediante las elipses de incertidumbre de diámetro creciente. También detecta las señales próximas y las asocia con una incertidumbre que combina la incertidumbre de la medida establecida con la incertidumbre creciente de la ubicación. Como resultado, la incertidumbre en las localizaciones de los hitos crece con el tiempo.

En la figura 4.1d ocurre una interesante transición, cuando el robot observa la marca que vio al principio de la construcción del mapa, y cuya posición es relativamente conocida. Con esta observación, el error de la ubicación del robot se reduce, como se indica en la figura 4.1d, ocurriendo lo mismo con la incertidumbre de otros indicadores en el mapa. Este fenómeno proviene de una correlación expresada en la matriz de covarianza del estado gaussiano siguiente.

4.3.2. Ekf-slam sin Correspondencia Conocida

En la mayoría de las situaciones reales, los sensores no son capaces de asociar directamente sus observaciones a objetos del entorno unívocamente. Se hace necesario entonces incorporar mecanismos de indentificación como paso previo a la utilización de los datos en el algoritmo de SLAM. A continuación se describe el algoritmo Ekf-slam sin Correspondencia Conocida.

1. **Algoritmo Ekf-slam sin Correspondencia Conocida**$(\mu(t-1), \sum(t-1), u(t), z(t), N(t-1))$:
2. $N(t) = N(t-1)$
3. $F(x) = \begin{pmatrix} 1 & 0 & 0 & 0 \cdots 0 \\ 0 & 1 & 0 & 0 \cdots 0 \\ 0 & 0 & 1 & 0 \cdots 0 \end{pmatrix}$
4. $\overline{\mu}(t) = \mu(t-1)$
$+ F(x)^T \begin{pmatrix} -\frac{v(t)}{w(t)} sin\mu(t-1,\theta) + \frac{v(t)}{w(t)} sin(\mu(t-1,\theta) + w(t)\Delta(t)) \\ \frac{v(t)}{w(t)} cos\mu(t-1,\theta) - \frac{v(t)}{w(t)} cos(\mu(t-1,\theta) + w(t)\Delta(t)) \\ w(t)\Delta(t) \end{pmatrix}$
5. $G(t) = I$
$+F(x)^T \begin{pmatrix} 0 & 0 & \frac{v(t)}{w(t)} cos\mu(t-1,\theta) - \frac{v(t)}{w(t)} cos(\mu(t-1,\theta) + w(t)\Delta(t)) \\ 0 & 0 & \frac{v(t)}{w(t)} sin\mu(t-1,\theta) - \frac{v(t)}{w(t)} sin(\mu(t-1,\theta) + w(t)\Delta(t)) \\ 0 & 0 & 0 \end{pmatrix} F(x)$
6. $\overline{\sum}(t) = G(t) \sum(t-1) G(t)^T + F(x)^T R(t) F(x)$
7. $Q(t) = \begin{pmatrix} \sigma_r & 0 & 0 \\ 0 & \sigma_\phi & 0 \\ 0 & 0 & \sigma_s \end{pmatrix}$
8. para todas las características observadas $z(t)^i = \begin{pmatrix} r(t)^i & \phi(t)^i & s(t)^i \end{pmatrix}$ hacer:
9. $\begin{pmatrix} \overline{\mu}(N(t+1), x) \\ \overline{\mu}(N(t+1), y) \\ \overline{\mu}(N(t+1), s) \end{pmatrix} = \begin{pmatrix} \overline{\mu}(t, x) \\ \overline{\mu}(t, y) \\ s(t)^i \end{pmatrix} + r(t)^i \begin{pmatrix} cos\left(\phi(t)^i + \overline{\mu}(t, \theta)\right) \\ sin\left(\phi(t)^i + \overline{\mu}(t, \theta)\right) \\ 0 \end{pmatrix}$
10. para k desde 1 hasta $N(t)+1$ hacer:

11. $\delta(k) = \begin{pmatrix} \delta(k,x) \\ \delta(k,y) \end{pmatrix} = \begin{pmatrix} \overline{\mu}(k,x) - \overline{\mu}(t,x) \\ \overline{\mu}(k,y) - \overline{\mu}(t,y) \end{pmatrix}$

12. $q(k) = \delta(k)^T \delta(k)$

13. $\hat{z}(t)^k = \begin{pmatrix} \sqrt{q(t)} \\ atan2(\delta(k,y), \delta(k,x)) - \overline{\mu}(t,\theta) \\ \overline{\mu}(k,s) \end{pmatrix}$

14. $F(x,k) = \begin{pmatrix} 1 & 0 & 0 & 0\cdots0 & 0 & 0 & 0 & 0\cdots0 \\ 0 & 1 & 0 & 0\cdots0 & 0 & 0 & 0 & 0\cdots0 \\ 0 & 0 & 1 & 0\cdots0 & 0 & 0 & 0 & 0\cdots0 \\ 0 & 0 & 0 & 0\cdots0 & 1 & 0 & 0 & 0\cdots0 \\ 0 & 0 & 0 & 0\cdots0 & 0 & 1 & 0 & 0\cdots0 \\ 0 & 0 & 0 & 0\cdots0 & 0 & 0 & 1 & 0\cdots0 \end{pmatrix}$

15. $M_{aux} =$

$$\begin{pmatrix} \sqrt{q(k)}\delta(k,x) & -\sqrt{q(k)}\delta(k,y) & 0 & -\sqrt{q(k)}\delta(k,x) & \sqrt{q(k)}\delta(k,y) & 0 \\ \delta(k,y) & \delta(k,x) & -1 & -\delta(k,y) & -\delta(k,x) & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

16. $H(t)^k = \frac{1}{q(k)} M_{aux} F(x,k)$

17. $\Psi(k) = H(t)^k \overline{\sum}(t) \left[H(t)^k\right]^T + Q(t)$

18. $\pi(k) = \left(z(t)^i - \hat{z}(t)^k\right)^T \Psi(k)^{-1} \left(z(t)^i - \hat{z}(t)^k\right)$

19. fin para

20. $\pi(N(t)+1) = \alpha$

21. $j(i) = argmin_k \pi(k)$

22. $N(t) = max\{N(t), j(i)\}$

23. $K(t)^i = \overline{\sum}(t) \left[H(t)^{j(i)}\right]^T \Psi(j(i))^{-1}$

24. $\overline{\mu}(t) = \overline{\mu}(t) + K(t)^i \left(z(t)^i - \hat{z}(t)^{j(i)}\right)$

25. $\overline{\sum}(t) = \left(I - K(t)^i H(t)^{j(i)}\right) \overline{\sum}(t)$

26. fin para

27. $\mu(t) = \overline{\mu}(t)$
28. $\sum(t) = \overline{\sum}(t)$
29. devolver $\mu(t), \sum(t)$

Puesto que la correspondencia no es conocida, la entrada del algoritmo Ekf-slam carece de una variable de correspondencia $c(t)$. En su lugar, se incluye el tamaño momentáneo del mapa, $N(t-1)$. La actualización de movimiento entre las líneas 3 y 6 es equivalente al del algoritmo Ekf-slam con Correspondencia Conocida. El bucle de actualización de la medida, sin embargo, es diferente. Comenzando en la línea 8, primero se crea la hipótesis de una nueva marca con el índice $N(t) + 1$; este índice es mayor que las señales en el mapa en este momento. La posición de la nueva marca se inicializa en la línea 9, calculando su localización esperada, dada la estimación de la ubicación del robot, la distancia y la relación en la medida. La línea 9 también asigna el valor singular observado a esta "nueva" marca. Luego, varias cantidades de realimentación se calculan entre las líneas 10 y 19 para todas las marcas $N_t + 1$ posibles, incluyendo la "nueva" marca. La línea 20 fija el umbral para la creación de una nueva marca, de modo que se crea una nueva marca si la distancia de Mahalanobis a todas las marcas existentes en el mapa excede el valor α. La correspondencia de máxima verosimilitud ML[2] se selecciona entonces en la línea 21. Si la medida se asocia a una marca no vista previamente, el contador de marcas se incrementa en la línea 22, y varios vectores y matrices se amplian en consecuencia (este paso algo tedioso no se hace explícito en el algoritmo). Finalmente, la actualización del EKF ocurre en las líneas 24 y 25. El algoritmo Ekf-slam devuelve el nuevo número de señales $N(t)$ junto con la media $\mu(t)$ y la covarianza $\sum(t)$.

La derivación de este algoritmo Ekf-slam se deduce directamente de derivaciones anteriores. En particular, la inicialización en la línea 9 es idéntica a la que está en la línea 10 en el algoritmo Ekf-slam con Correspondencia Conocida. Desde las líneas 10 hasta la 19 son semejantes a las líneas comprendidas entre la 12 y la 17 en el algoritmo Ekf-slam con Correspondencia Conocida, con la variable añadida $\pi(k)$, necesaria para calcular la correspondencia ML. La selección de la correspondencia ML en la línea 21, y la definición de la distancia de Mahalanobis en la línea 18, es análoga a la descrita en el algoritmo Ekf-slam del apartado 3.5. Las actualizaciones de las medidas en las líneas 24 y 25 son también análogas a las líneas 18 y 19 en el algoritmo EKF con Correspondencia Conocida, asumiendo que la participación de los vectores y matrices son de la dimensión apropiada en caso de que el mapa haya sido ampliado.

[2] *Maximum Likelihood*, Probabilidad Máxima

4.3.3. Selección de Características y Manejo del Mapa

La realización robusta del Ekf-slam requiere en la práctica, a menudo, las técnicas adicionales para el *manejo del mapa*. Muchas de ellas están relacionadas con el hecho de que la suposición del ruido Gaussiano sea poco realista, y muchas medidas falsas ocurren en el tramo final de la distribución ruidosa. Tales medidas falsas pueden causar la creación de señales falsas en el mapa que, sucesivamente, afectan negativamente a la localización del robot.

Muchas técnicas avanzadas poseen mecanismos para resolver apariciones en el espacio de la medida. Tales apariciones se definen como observaciones de hitos falsos fuera del rango de incertidumbre de cualquier indicador en el mapa. La técnica más simple para rechazar tales apariciones es mantener una *lista provisional de marcas*. En vez de aumentar el mapa con una marca nueva a la vez que una medida indica la existencia de dicha marca, la marca nueva se agrega primero a una lista provisional de marcas. Esta lista es similar al mapa, pero las marcas en esta lista *no* se utilizan para modificar la ubicación del robot (los gradientes correspondientes en las ecuaciones de medida se fijan a cero). Una vez que un hito se haya observado coherentemente y su elipse de incertidumbre se ha contraído, transiciona en el mapa regular.

En aplicaciones prácticas, este mecanismo tiende a reducir el número de indicadores en el mapa a un factor significativo, mientras que todavía conserva todas las marcas físicas con alta probabilidad. Otro paso, comúnmente encontrado también en aplicaciones avanzadas, es mantener una *probabilidad de existencia de marca*. Una probabilidad posterior así se puede poner en práctica como la relación de transformación de las probabilidades de dicho registro, y se denota o_j para la marca j-ésima en el mapa. Siempre que se observe la marca j-ésima, o_j es incrementado mediante un valor fijo. No observar m_j, cuando estuviera en el rango perceptivo de los sensores del robot, conduce a un decremento de o_j. Puesto que nunca se puede conocer con certeza si una señal se encuentra dentro del rango perceptivo de un robot, el decremento puede tener en cuenta la probabilidad de tal acontecimiento. Las señales se eliminan del mapa cuando el valor o_j se encuentra por debajo de un umbral. Tales técnicas conducen a mapas mucho más pobres ante el ruido de medida no gaussiano.

Al inicializar la estimación para una nueva marca, que comienza con una covarianza con elementos muy grandes, puede inducir *inestabilidades numéricas*. Esto es porque el primer paso de la actualización de la covarianza cambiará este valor a varias órdenes de magnitud, muchos quizás para generar una matriz que siga siendo definida positiva. Una mejor estrategia implica

un paso de inicialización explícito para cualquier característica que no se haya observado antes. En particular, tal paso inicializaría la covarianza $\sum(t)$ directamente con la incertidumbre real de la marca.

Algunas cuestiones adicionales:

- **Orden espacial.** Las marcas más separadas tienen las posibilidades más pequeñas de confundirlas involuntariamente. Es por lo tanto común la práctica de elegir hitos que están suficientemente lejanos unos de otros de modo que la probabilidad de confundir uno con otro sea pequeña. Esto introduce una compensación interesante: un número grande de marcas aumenta el peligro de confundirlas. Muy pocas marcas hacen más difícil la localización del robot, que sucesivamente también aumenta las posibilidades de confundir las marcas. Poco se sabe actualmente sobre la densidad óptima de señales, y los investigadores utilizan a menudo la intuición al seleccionar las marcas específicas.

- **Singularidades.** Cuando seleccionamos las marcas apropiadas, es esencial maximizar las peculiaridades perceptuales de los hitos. Por ejemplo, las puertas podrían tener colores diferentes, o los pasillos podrían tener anchuras diferentes. Las singularidades resultantes son esenciales para un SLAM eficaz.

4.4. El algoritmo general Ukf-slam

El algoritmo UKF, desde que fué propuesto por Julier y Uhlman [Ju97], ha demostrado ser una buena alternativa a la del filtro de Kalman extendido (EKF). A diferencia del filtro de Kalman ampliado (EKF), que se basa en la linealización de la función no lineal mediante el uso de desarrollos de la serie de Taylor, UKF utiliza los verdaderos modelos no lineales y se aproxima a una distribución aleatoria de la variable de estado [Qi08].

Considere el sistema general no lineal discreto:

$$\begin{cases} x_{k+1} &= f(x_k, u_k) + w_k \\ y_k &= h(x_k) + v_k \end{cases}$$

donde $x_k \in \Re^n$ es el vector de estado, $u_k \in \Re^r$ es el vector conocido de entrada, $y_k \in \Re^m$ es el vector de salida en el instante k; w_k y v_k son los vectores de perturbación y ruido del sensor, respectivamente, los cuales se asumen con

ruido blanco gaussiano y media cero.

La estimación UKF se puede expresar de la siguiente forma:

Inicialización.

$$\begin{cases} \overline{x}_0 &= E\left[x_0\right] \\ P_0 &= E\left[\left(x_0-\overline{x}_0\right)\left(x_0-\overline{x}_0\right)^T\right] \end{cases}$$

Puntos de cálculo Sigma e instante de actualización.

$$\begin{cases} \chi_{k-1} &= \left[\overline{x}_{k-1}, \overline{x}_{k-1} \pm \sqrt{\left(n+\lambda\right)P_{k-1}}\right] \\ \chi^*_{k|k-1} &= f\left(\chi_{k-1}\right) \\ \overline{x}_{k|k-1} &= \sum_{i=0}^{2n} w_i^m \chi^*_{i,k|k-1} \\ P_{k|k-1} &= \sum_{i=0}^{2n} w_i^c \left(\chi^*_{i,k|k-1}-\overline{x}_{k|k-1}\right)\cdot\left(\chi^*_{i,k|k-1}-\overline{x}_{k|k-1}\right)^T + Q \\ \chi_{k|k-1} &= \left[\overline{x}_{k|k-1}, \overline{x}_{k|k-1} \pm \sqrt{\left(n+\lambda\right)P_{k|k-1}}\right] \\ \gamma_{k|k-1} &= h\left(\chi_{k|k-1}\right) \\ \overline{y}_{k|k-1} &= \sum_{i=0}^{2n} w_i^m \gamma_{i,k|k-1} \end{cases}$$

donde

$$\begin{cases} w_0^m &= \frac{\lambda}{n+\lambda} \\ w_0^c &= \frac{\lambda}{n+\lambda} + \left(n-\alpha^2+\beta\right) \\ w_i^m &= w_i^c = \frac{1}{2(n+\lambda)} \qquad\qquad i=1,\ldots,2n \\ \lambda &= n\left(\alpha^2-1\right) \end{cases}$$

Actualización de medición.

$$\begin{cases} P_{\overline{y}_k\overline{y}_k} &= \sum_{i=0}^{2n} w_i^c \left(\gamma_{i,k|k-1} - \overline{y}_{k|k-1}\right) \times \left(\gamma_{i,k|k-1} - \overline{y}_{k|k-1}\right)^T + R \\ P_{\overline{x}_k\overline{y}_k} &= \sum_{i=0}^{2n} w_i^c \left(\chi_{i,k|k-1} - \overline{x}_{k|k-1}\right) \times \left(\gamma_{i,k|k-1} - \overline{y}_{k|k-1}\right)^T \\ K_k &= P_{\overline{x}_k\overline{y}_k} P_{\overline{y}_k\overline{y}_k}^{-1} \\ P_k &= P_{k|k-1} - K_k P_{\overline{y}_k\overline{y}_k} K_k^T \\ \overline{x}_k &= \overline{x}_{k|k-1} + K_k \left(y_k - \overline{y}_{k|k-1}\right) \end{cases}$$

donde las variables se definen de la siguiente forma: $\{w_i\}$ es un conjunto de pesos escalares, y n es la dimensión del estado; el parámetro α determina el reparto de los puntos sigma alrededor de $\overline{x}$, y generalmente toma valores en el rango 1e-4$\leq \alpha \leq$1. La constante β se utiliza para incorporar parte del conocimiento previo de la distribución de x, y para la distribución gaussiana, β=2 es óptima. Q y R son las covarianzas de perturbación y ruido del sensor, respectivamente. La operación de álgebra lineal para agregar una columna a una matriz, por ejemplo $A \pm z$, se define como la suma del vector para cada columna de la matriz.

4.5. El algoritmo general Fastslam

El algoritmo del Fastslam, introducido por M. Montemerlo y otros [Mo02], marca un cambio conceptual fundamental en el diseño del cálculo del SLAM con métodos probabilísticos recursivos. Esfuerzos previos se centraban en mejorar la implementación del Ekf-slam, mientras mantenían la presunción inicial de la gaussiana lineal. FastSLAM, que está basada en el muestreo recursivo de Montemerlo, o filtro de partículas, fue el primero en representar directamente el modelo de proceso no lineal y distribución no gaussiana de la posición [Ve08].

Los algoritmos Fastslam son una propuesta para abordar el problema del SLAM en tiempo real [De07]. En ellos se descompone el problema del SLAM en un problema de localización del robot, y en una colección de problemas de estimación de los hitos que condicionan la estimación de la ubicación del robot [Mo02]. FastSLAM utiliza un filtro de partículas modificado para estimar las siguientes rutas del robot. Cada partícula posee K filtros de Kalman, los cuales estiman la localización de las K balizas que a su vez condicionan la

estimación de la ruta.

Una explicación informal del algoritmo podría ser la siguiente. Se sabe que es relativamente simple localizar las características (elementos relevantes, como esquinas o paredes) de un mapa, una vez conocido el camino que ha seguido el robot cartografiador [De07]. Por este motivo el algoritmo propone evaluar la distribución probabilística del camino seguido por el robot mediante un filtro de partículas, es decir, realizando apuestas sobre la pose (x,y,θ). Cada partícula considera su pose apostada como verdadera y realiza un proceso de localización de características, mediante filtros de Kalman extendidos (Extended Kalman Filters o EKF), empleando un filtro por cada característica. Una vez evaluada la bondad de la apuesta, se selecciona un conjunto de partículas "buenas", a partir de las cuales se vuelve a apostar la pose.

En Fastslam, al igual que en Ekf-slam, se adoptan las ubicaciones que se comportan de acuerdo a una ley de probabilidad llamada modelo de movimiento, con una densidad subyacente [Ca06]

$$p\left(s_t|s_{t-1}\right)$$

Asimismo, las mediciones se rigen por el modelo de medición (probabilístico) $p\left(z_t|s_t,\theta,n_t\right)$ con z_t medidas, $\theta = \{\theta_1,\ldots,\theta_K\}$ el conjunto de hitos, y $n_t \in \{1,\ldots,K\}$ el índice del hito observado en el instante t. El objetivo final es estimar el posterior $p\left(s^t,\theta|z^t\right)$.

La representación exacta empleada por Fastslam se interpreta como:

$$\begin{aligned} p\left(s^t,\theta|z^t,n^t\right) &= p\left(s^t|z^t,n^t\right)p\left(\theta|z^t,n^t\right) && (4.5)\\ &= p\left(s^t|z^t,n^t\right)\prod_{1\le k\le K} p\left(\theta_k|s^t,z^t,n^t\right). && (4.6)\end{aligned}$$

Obsérvese la dependencia condicional de n^t. Los $K+1$ factores en la ecuación 4.6 se obtienen de la siguiente forma.

- Estimación de la ruta $p\left(s^t,\theta|z^t,n^t\right)$
  Esto se logra mediante un filtro de partículas: Fastslam mantiene un conjunto de M partículas, $\{s^{t,[m]}\} = \{s_1^{[m]},\ldots,s_t^{[m]}\}$, donde $[m]$ se refiere al número de partículas en el conjunto. Para crear el conjunto de partículas en el instante t, $s^{t,[m]}$, primeras toma muestras de las M partículas del

modelo de movimiento en un conjunto temporal; $s_t^{[m]} \approx p\left(s_t|s_{t-1}^{[m]}\right)$. Entonces, se calculan los factores de peso importantes $w_t^{[m]}$, que determinan la probabilidad de que una partícula determinada del conjunto temporal pase al conjunto final.

- Estimación de las localizaciones de los hitos $p\left(\theta_k|s^t, z^t, n^t\right)$
  La actualización posterior depende de si el hito k-ésimo se ha observado o no,

$$p\left(\theta_k|s^t, z^t, n^t\right) \propto \begin{cases} p\left(z_t|\theta_k, s_t, n_t\right) \cdot p\left(\theta_k|s^{t-1}, z^{t-1}, n^{t-1}\right) & si\ k = n_t \\ p\left(\theta_k|s^{t-1}, z^{t-1}, n^{t-1}\right) & si\ k \neq n_t \end{cases}$$

  Fastslam implementa la actualización de la ecuación anterior por medio de un EKF.

- Problema de asociación de datos.
  Fastslam (como la mayoría de las soluciones existentes basadas en Ekf-slam) resuelve el problema de asociación de datos a través de la estimación de máxima probabilidad.

Capítulo 5

Preparación de la Comparativa

5.1. Introducción

Abordar un problema como la prueba y evaluación de diversos algoritmos de SLAM no es una tarea sencilla. Desde un principio, quedó claro que el esfuerzo que requeriría la implementación propia de los diferentes algoritmos no era abordable en el marco de este proyecto. De hecho, puede considerarse un proyecto fin de carrera en sí mismo la programación de cualquiera de ellos. El objetivo, pues, se centró en seleccionar alguna alternativa dentro de los paquetes y librerías software existentes, de modo que se redujera considerablemente el tiempo de puesta en funcionamiento del entorno experimental.

Conviene aclarar que, al tratarse de un campo de investigación activo, los recursos disponibles se incrementan regularmente. Esto implica que una evaluación realizada algunos meses atrás, puede no ser la más adecuada transcurrido el tiempo. En el caso concreto de este proyecto, dicho análisis tuvo lugar hace más de un año y medio.

Las principales alternativas que se consideraron fueron las basadas en los lenguajes de programación C/C++ y Matlab. Por cuestiones relativas a la curva de aprendizaje, facilidad de prototipado y por la mayor cantidad de recursos disponibles, la selección se centró en la segunda de las opciones.

El software matemático MATLAB (MATrix LABoratory) ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio. Trabajar con Matlab tiene ventajas y desventajas [Ar04]. Las ventajas son:

- Matlab es de amplia difusión, estandarizado y con un entorno de programación que posee un gran número de funcionalidades incorporadas y añadidas.

- Programar en Matlab es fácil.
- Matlab corre en muchas plataformas y sistemas operativos.
- Cualquier toolbox no requiere ninguna instalación o compilación (cuando Matlab está instalado).

En cambio, las desventajas son:

- Matlab es comercial y costoso.
- Aunque haya funciones rápidas incorporadas, Matlab puede ser lento.

A su vez, dentro de Matlab existen diferentes toolboxes que se han desarrollado en el ámbito descrito en este proyecto, como son "CAS Robot Navigation Toolbox" [Op07], "Kalmtool" [Dk09], "Kalman filter" [Mu04], "ReBEL" [Re06], etc.

Finalmente, se optó por basar la fase de análisis y comparativa de algoritmos del proyecto en los recursos software desarrollados por Tim Bailey, y que están disponibles desde su página web http://www-personal.acfr.usyd.edu.au/tbailey/. En primera instancia, como se ha señalado previamente, se realizarán simulaciones de los algoritmos EKF-Slam, FastSLAM 1.0 y UKF-Slam [Ba06] en diferentes mapas.

5.1.1. Criterios de calificación

En toda comparativa de métodos parametrizables surge siempre el problema de cómo definir un criterio de calificación adecuado. De nuevo, y considerado en toda su extensión, este problema excede el alcance del proyecto. Concretamente, el mejor algoritmo de SLAM es aquél que ofrece un mejor comportamiento frente a sus competidores en la mayoría de los posibles escenarios de comparación planteables; y simplemente la definición de dichos escenarios ya es un problema en sí.

Por todo lo expuesto, se adoptó por abordar el problema de forma parcial, definiendo una serie de experimentos para obtener más conclusiones cualitativas que cuantitativas. De esta forma se evaluaron aspectos como:

- Robustez.
- Calidad del resultado.

- Coste y complejidad computacional.

El objetivo del análisis de robustez fue determinar la sensibilidad de los diferentes algoritmos ante variaciones de sus parámetros. El mejor algoritmo será aquél que presente una menor variación en su salida frente a la misma perturbación en dichos parámetros.

El análisis de calidad se centra en obtener medidas de la calidad de los mapas obtenidos por cada método. El mejor algoritmo es el que, de forma global sobre un conjunto variado de casos, proporcione un error menor en el resultado.

Finalmente, el análisis del coste y la complejidad tratan de averiguar qué algoritmo es el que ofrece un mejor comportamiento en términos de recursos computacionales demandados.

Todos los aspectos que se tratan en el presente capítulo, se definen detalladamente a continuación. En la sección 5.2 se detallan los diferentes mapas a utilizar a partir de este momento, para las ejecuciones de los diferentes algoritmos En la sección 5.3 se definen los algoritmos a utilizar en el análisis, y posteriores capítulos. En la sección 5.5 se realiza el estudio de la sintonización de los parámetros propios de los algoritmos, según la caracterización de los diferentes mapas. Y en la última sección de este capítulo, sección **??**, se detallarán los resultados obtenidos del estudio de la sintonización de los parámetros propios de los algoritmos, obteniendo la configuración más idónea de dichos parámetros para las pruebas a ejecutar en el capítulo siguiente.

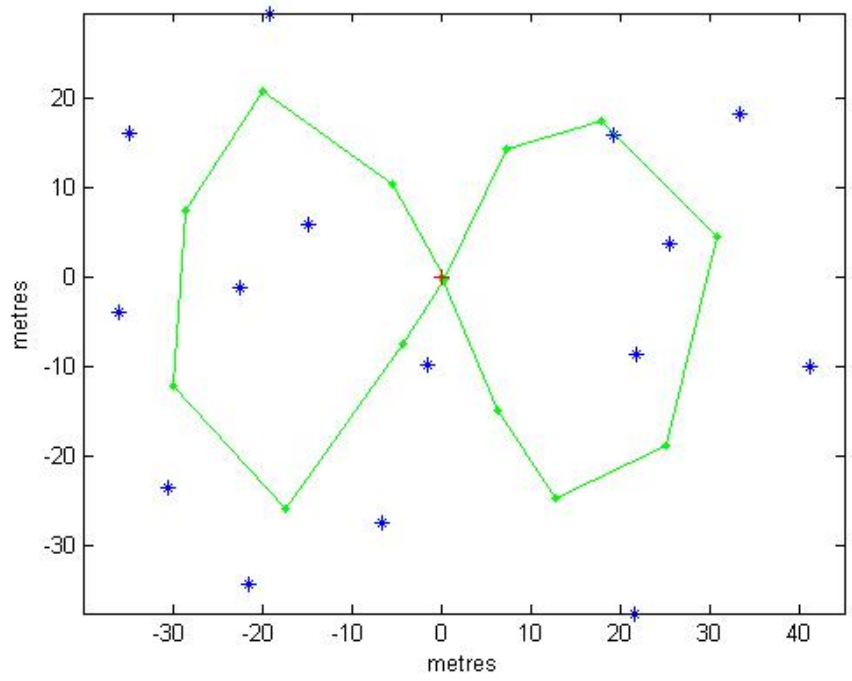
5.2. Los mapas

Las ejecuciones de los algoritmos se realizarán utilizando cuatro mapas diferentes. Cada uno de ellos posee características especiales que permitirán ensayar el comportamiento de los distintos algoritmos sobre escenarios variados.

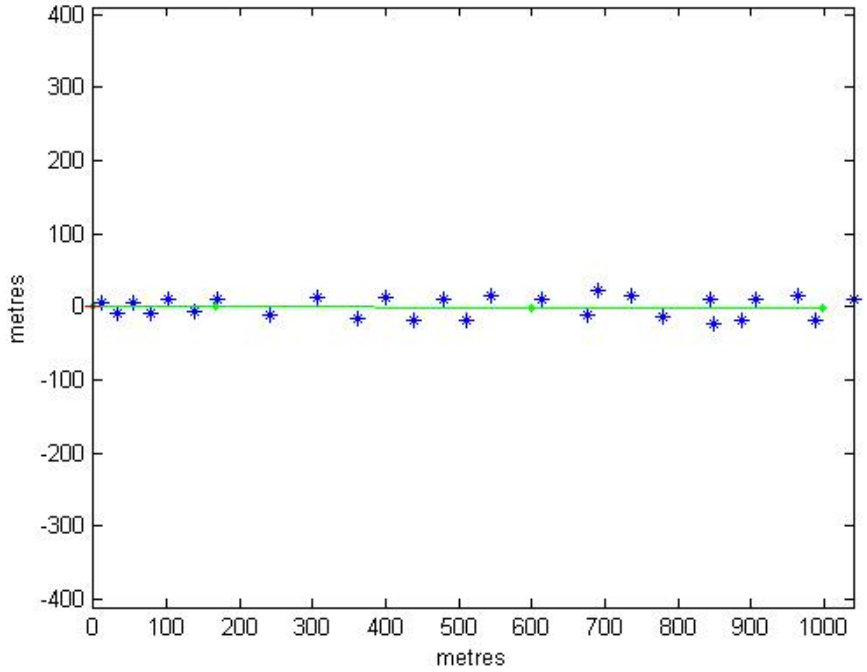
Los mapas considerados han sido los siguientes:

1. Mapa *"Smallmap"*. Este mapa (figura 5.1a) corresponde a un circuito de unos 40 metros de ancho por 60 metros de largo, en el que el robot debe realizar un bucle. Existe una intersección en el centro del mismo.

2. Mapa *"Linemap"*. Este mapa (figura 5.1b) corresponde a un recorrido a lo largo de una línea recta de 1000 metros de longitud.

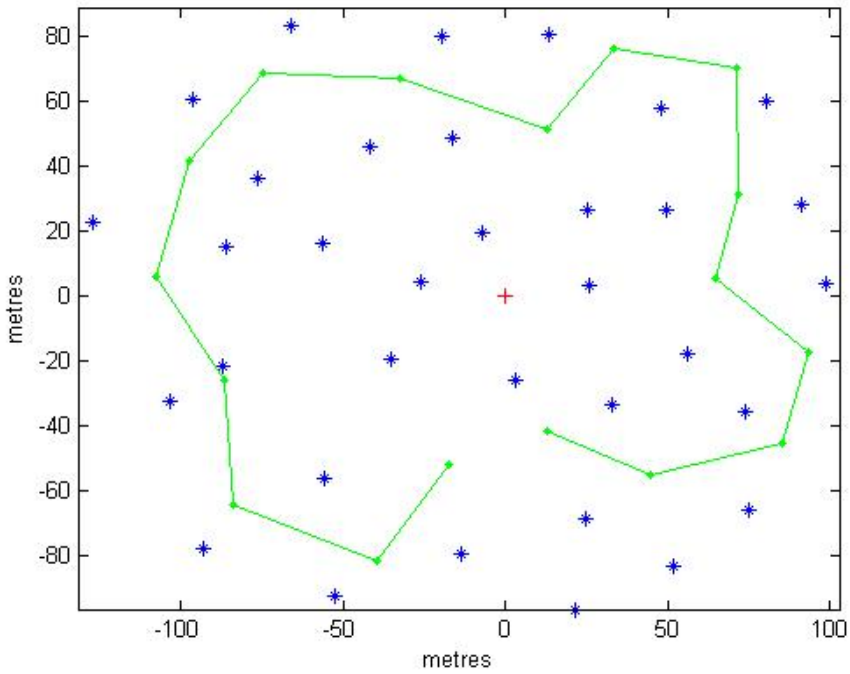


a) Mapa “Smallmap” para la ejecución de los algoritmos.

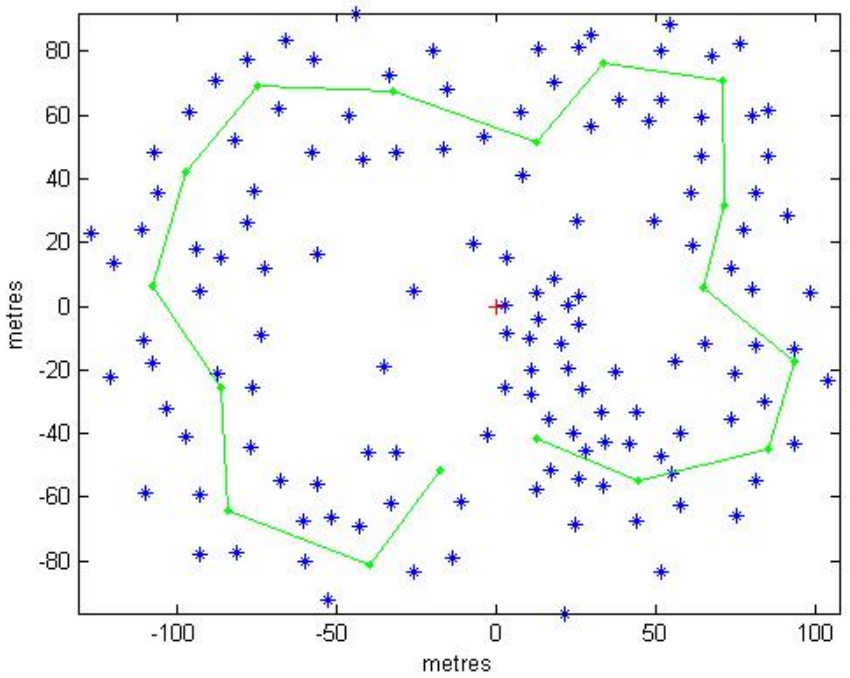


b) Mapa “Linemap” para la ejecución de los algoritmos.

Figura 5.1: Mapas para la ejecución de los algoritmos. (I)



a) Mapa “Webmap” para la ejecución de los algoritmos.



b) Mapa “Densemap” para la ejecución de los algoritmos.

Figura 5.2: Mapas para la ejecución de los algoritmos. (II)

3. Mapa *"Webmap"*. Este mapa (figura 5.2a) corresponde a un circuito de grandes dimensiones, en torno a 30000 metros cuadrados, en el que se combinan tramos rectos con curvas.

4. Mapa *"Densemap"*. Este mapa (figura 5.2b) posee un recorrido similar al anterior, salvo porque éste posee más hitos a lo largo del circuito.

Hay que comentar que el mapa "Linemap" difiere en su naturaleza del resto. No se trata de un circuito cerrado, sino de un simple recorrido en línea recta, por lo que ya de entrada es posible anticipar que los resultados de localización van a ser necesariamente pobres, e inferiores al resto. En este caso más que el error en sí, convendrá analizar la rapidez de crecimiento del mismo.

Los mapas definidos en esta sección pertenecen a Tim Bailey[1], y se pueden obtener de su página web [Ba06]. Se trata de mapas pensados para aplicaciones de SLAM con vehículos grandes en exteriores.

5.2.1. Caracterización de los mapas

Se ha considerado conveniente caracterizar los mapas a partir de un conjunto de medidas que sirvan como base tanto para la selección de los parámetros como para la interpretación de los resultados.

Está claro que la parametrización de los diferentes algoritmos debería tener en cuenta la naturaleza del escenario en que se va a ejecutar la simulación. Así, por ejemplo, hay una gran diferencia entre un SLAM pensado para operar sobre un mapa de grandes dimensiones con poca densidad de balizas, a hacerlo en uno pequeño con un número elevado de hitos de referencia.

Los factores fundamentales que se han tenido en cuenta son la distancia recorrida, la densidad de balizas y el tipo de trayectoria. Para cada factor, se han programado una serie de funciones en Matlab que extraen de los mapas diferentes medidas relacionadas con los mismos. Dichas medidas son las siguientes:

- Longitud de ciclo del mapa (LCM): Corresponde a la suma de las distancias euclídeas[2] entre cada dos puntos de referencia. Esta medida co-

[1] Tim Bailey es personal docente de la Universidad de Sydney, y colabora en el Centro Australiano de Robótica Móvil (ACFR).

[2] La distancia expresa la proximidad o lejanía entre dos objetos, o el intervalo de tiempo que transcurre entre dos sucesos [Cu89]. En matemática, la distancia entre dos puntos del

rresponde al recorrido (en metros) que tiene que realizar el robot para completar un ciclo.

- Velocidad de rotacion promedio (VRP): Ángulo medio (en radianes) por segundos que gira un robot al realizar el recorrido de un mapa determinado.
- Distancia promedio 1 (DP1): Media de las distancias euclídeas entre el hito más cercano a cada punto de referencia, y el hito más cercano a este otro hito.
- Distancia promedio 2 (DP2): Media de la distancia euclídea del segundo hito más cercano a cada punto de referencia.
- Distancia promedio 3 (DP3): Media de las distancias euclídeas mínimas de cada hito con respecto a los demás hitos del conjunto.
- Angulo promedio 1 (AP1): Separación angular media entre balizas desde cada punto de referencia.

Tabla 5.1: Resultado de las medidas relacionadas con los mapas.

	Smallmap	**Linemap**	**Webmap**	**Densemap**
LCM	226.400636 m	1997.168483 m	634.857415 m	634.857415 m
VRP	0.210349 rad/sg	0.007891 rad/sg	0.093675 rad/sg	0.093675 rad/sg
DP1	16.251294 m	43.094647 m	30.053661 m	11.854885 m
DP2	15.917890 m	39.823266 m	24.894574 m	12.132816 m
DP3	16.034785 m	42.320701 m	28.861573 m	11.430397 m
AP1	1.032118 rad	0.000000 rad	2.146901 rad	0.626513 rad

La tabla 5.1 muestra los valores obtenidos para cada uno de los mapas.

espacio euclídeo equivale a la longitud del segmento de recta que los une, expresado numéricamente. Se denomina distancia euclídea entre dos puntos A(x1,y1) y B(x2,y2) del plano, a la longitud del segmento de recta que tiene por extremos A y B. Dicha distancia puede calcularse de la siguiente forma: $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$.

5.3. Los algoritmos

Los algoritmos que se van a analizar en la comparativa son el EKF-Slam, el UKF-Slam y el FastSlam. En esta sección se analizan y describen en detalle, incluyendo la definición de sus parámetros, su estructura y las funciones principales que los integran.

Cada uno de los algoritmos parte del software de simulación desarrollado por Tim Bailey, al igual que los mapas, y se pueden obtener de su página web [Ba06]. Posteriormente se han introducido una serie de modificaciones en los mismos.

Para ejecutar el simulador de cada algoritmo, primero se carga un mapa, para posteriormente poder ejecutar el simulador. La estructura de datos que devuelve la ejecución, contiene valores almacenados para el subsiguiente procesamiento fuera de línea.

5.3.1. Descripción previa de parámetros propios

En este apartado se describen los parámetros propios que poseen los algoritmos que serán objeto de análisis.

- **Parámetro "Gate_Reject".**
  El parámetro propio "Gate_Reject" determina la distancia máxima para la asociación de hitos. Los valores más comunes para dicho parámetro son 1.0, que representa 1-sigma, 4.0 (2-sigma), 9.0 (3-sigma) y 16.0 (4-sigma). Dependiendo del valor de sigma que se le asigne, posee una probabilidad porcentual diferente de eficiencia: 1-sigma posee un porcentaje de 40 %, 2-sigma posee un porcentaje de 86 %, 3-sigma posee un porcentaje de 99 %, y 4-sigma posee un porcentaje de 99.9 %.

- **Parámetro "Gate_Augment".**
  El parámetro propio "Gate_Augment" determina la distancia mínima para la utilización de una nueva característica. Para la utilización de dicho parámetro, se ha de tener en cuenta no asignar valores menores al que posea en ese mismo instante el parámetro "Gate_Reject".

- **Parámetro "Nparticles".**
  Parámetro propio que indica el número de partículas que se utilizarán para estimar la trayectoria del robot, de forma que dada una partícula, las marcas son independientes.

- **Parámetro "Neffective".**
  El parámetro propio "Neffective" indica el número mínimo de partículas efectivas (en porcentaje) antes del nuevo muestreo. El algoritmo, según el porcentaje indicado en este parámetro, se quedará con un número determinado de partículas, las cuales se consideran las mejores en ese instante, eliminando el resto de las partículas, para la generación de otras en el siguiente instante.

5.3.2. EKF-Slam

El software utilizado incluye una implementación simple del algoritmo EKF-Slam estándar [Ba06]. Corresponde a la versión más antigua del simulador de EKF-Slam, pero probablemente es la más fácil de entender, pues evita usar variables globales.

Permite la configuración de forma sencilla, mediante el fichero ‘‘`configfile.m`’’, el cual permite modificar la configuración de los diferentes parámetros de la simulación por medio de un fichero auxiliar, incluyendo número de ciclos, nivel de ruido, etc... para realizar el SLAM. También, están disponibles varios switches entre los que se permiten elegir la asociación de datos conocidos, salida gráfica, etc.

Además de animaciones en línea, el simulador devuelve una estructura de datos de la información de estado registrada para el procesamiento fuera de línea.

5.3.2.1. Definición de funciones principales

A continuación se definen las funciones principales del algoritmo EKF-Slam:

- `function [G,iwp] = compute_steering(x,wp,iwp,minD,G,rateG, maxG,dt)`: Determina si se alcanzó el punto de referencia actual y calcula el cambio en el ángulo de dirección actual (G) para señalar hacia el punto de referencia actual.

- `function [V,G] = add_control_noise(V,G,Q,addnoise)`: Añade ruido de forma aleatoria a los valores nominales del control.

- `function [xn,Pn] = predict (x,P,v,g,Q,WB,dt)`: Predice el estado y la covarianza.

- `function [x,P] = observe_heading(x,P,phi,useheading)`: Representa la actualización del estado para una medida dada como parámetro, phi, con ruido de medida fijo, sigmaPhi.

- `function [z,idf] = get_observations(x,lm,idf,rmax)`: Selecciona un conjunto de hitos, los cuales son visibles dentro del campo de visión semicircular del robot.

- `function z = add_observation_noise(z,R,addnoise)`: Añade ruido de medida de forma aleatoria. Se asume que R es diagonal.

- `function [zf,idf,zn,table] = data_associate_known(x,z,idz, table)`: Esta función mantiene una tabla de consulta de característica/observación. Devuelve la tabla actualizada, el conjunto de observaciones asociadas y el conjunto de observaciones para las nuevas características.

- `function [zf,idf,zn] = data_associate(x,P,z,R,gate1,gate2)`: Obtiene de forma sencilla la asociación de datos del vecino más cercano.

- `function [x,P] = update(x,P,z,R,idf,batch)`: Actualiza el estado y la covarianza.

- `function [x,P] = augment(x,P,z,R)`: Añade las nuevas características al estado.

5.3.2.2. Estructura del algoritmo

```
function data= ekfslam_sim(lm,wp)
...
while iwp ≠ 0

    [G,iwp] = compute_steering(xtrue,wp,iwp,AT_WAYPOINT,G,RATEG,
    MAXG,dt);

    ...

    [Vn,Gn] = add_control_noise(V,G,Q,SWITCH_CONTROL_NOISE);

    [x,P] = predict (x,P,Vn,Gn,QE,WHEELBASE,dt);

    [x,P] = observe_heading(x,P,xtrue(3),

    SWITCH_HEADING_KNOWN);
```

```
    ...
    if dtsum ≥ DT_OBSERVE
        ...
        [z,ftag_visible] = get_observations(xtrue,lm,ftag,
        MAX_RANGE);
        z = add_observation_noise(z,R,SWITCH_SENSOR_NOISE);
        if SWITCH_ASSOCIATION_KNOWN == 1
            [zf,idf,zn, da_table] = data_associate_known(x,z,
            ftag_visible,da_table);
        else
            [zf,idf, zn] = data_associate(x,P,z,RE,GATE_REJECT,
            GATE_AUGMENT);
        end
        [x,P] = update(x,P,zf,RE,idf,SWITCH_BATCH_UPDATE);
        [x,P] = augment(x,P,zn,RE);
    end
    ...
end
...
```

5.3.3. FastSlam

Se incluyen implementaciones de los algoritmos de FastSlam 1.0 y 2.0. En este proyecto se ha utilizado principalmente la versión 1.0. Este simulador realiza una demostración simple de la implementación del algoritmo FastSlam [Ba06]. Dicha implementación posee una ejecución lenta en Matlab, debido a los gastos indirectos de las construcciones de colocación etc.

El simulador de FastSlam permite la parametrización por medio de un fichero de configuración, e incorpora inicialmente sólo asociación de datos conocida.

El tiempo de ejecución de este simulador es muy lento debido, en parte, a los gráficos de las animaciones del simulador, y también al uso de bucles ‘‘`for`’’. Los bucles en Matlab son muy ineficientes, pero es difícil evitarlos si se desea realizar un código que no sea de difícil comprensión. Los algoritmos básicos, sin embargo, están bastante claros.

5.3.3.1. Definición de funciones principales

A continuación se definen las funciones principales del algoritmo FastSlam:

- `function [G,iwp] = compute_steering(x,wp,iwp,minD,G,rateG, maxG,dt)`: Determina si se alcanzó el punto de referencia actual y calcula el cambio en el ángulo de dirección actual (G) para señalar hacia el punto de referencia actual.
- `function xv = predict_true(xv,V,G,WB,dt)`: Predice la ubicación del vehículo en cada momento.
- `function [V,G] = add_control_noise(V,G,Q,addnoise)`: Añade ruido aleatorio a los valores de control nominal.
- `function particle = predict(particle,V,G,Q,WB,dt,addrandom)`: Predice el estado del vehículo.
- `function [z,idf] = get_observations(x,lm,idf,rmax)`: Selecciona un conjunto de hitos, los cuales son visibles dentro del campo de visión semicircular del robot.
- `function z = add_observation_noise(z,R,addnoise)`: Añade ruido de medición de forma aleatoria.
- `function [zf,idf,zn,table] = data_associate_known(z,idz, table,Nf)`: Asocia las nuevas características a las ya existentes.
- `function [zf,idf,zn] = data_associate_f1(x,P,z,R,gate1,gate2)`: Obtiene de forma sencilla la asociación de datos del vecino más cercano. Esta función fue desarrollada en este proyecto y añadida posteriormente al software.
- `function particle = feature_update(particle,z,idf,R)`: Seleccionando una nueva ubicación de la distribución propuesta, se asume perfecta esta ubicación, y cada actualización de las características se puede computar independientemente y sin incertidumbre de la ubicación.
- `function particle = add_feature(particle,z,R)`: Añade nuevas características.
- `function particles = resample_particles(particles,Nmin, doresample)`: Vuelve a tomar muestras de las partículas si su varianza del peso es tal que las N partículas logradas es menor que el número mínimo.

5.3.3.2. Estructura del algoritmo

```
function data= fastslam1_sim(lm,wp)
...
while iwp ≠ 0

    [G,iwp] = compute_steering(xtrue,wp,iwp,AT_WAYPOINT,G,RATEG,
    MAXG,dt);

    ...

    xtrue = predict_true(xtrue,V,G,WHEELBASE,dt);

    [Vn,Gn] = add_control_noise(V,G,Q,SWITCH_CONTROL_NOISE);

    for i=1:NPARTICLES

        particles(i) = predict(particles(i),Vn,Gn,Qe,WHEELBASE,
        dt,SWITCH_PREDICT_NOISE);
        ...

    end

    ...

    if dtsum ≥ DT_OBSERVE

        ...

        [z,ftag_visible] = get_observations(xtrue,lm,ftag,
        MAX_RANGE);

        z = add_observation_noise(z,R,SWITCH_SENSOR_NOISE);

        ...

        if SWITCH_ASSOCIATION_KNOWN == 1

            ...

            [zf,idf,zn,da_table] = data_associate_known(z,
            ftag_visible,da_table,Nf);

            for i=1:NPARTICLES

                if not isempty(zf)

                  ...

                  particles(i) = feature_update(particles(i),zf,
                  idf,R);

                end
```

```
            if not isempty(zn)
              particles(i) = add_feature(particles(i),zn,R);
            end
        end
        ...
        particles = resample_particles(particles,NEFFECTIVE,
        SWITCH_RESAMPLE);
    else
        ...
        for i=1:NPARTICLES
            [zfi,idfi,zni] = data_associate_f1(particles(i).xv,
            particles(i).xf,particles(i).Pf,z,Re,GATE_REJECT,
            GATE_AUGMENT);
            if not isempty(zfi)
              ...
              particles(i) = feature_update(particles(i),zf{i},
              idf{i},R);
            end
            if not isempty(zni)
              particles(i) = add_feature(particles(i),zn{i},R);
            end
        end
        ...
        particles = resample_particles(particles,NEFFECTIVE,
        SWITCH_RESAMPLE);

end

...

end
...
```

5.3.4. UKF-Slam

Este conjunto de funciones de Matlab realiza una simulación de la localización y construcción del mapa de forma simultánea (SLAM), usando el filtro de Kalman sin correspondencia conocida (UKF) [Ba06]. Este simulador es una adaptación directa del código de EKF-Slam, pero sustituye el filtro de Kalman

extendido (EKF) por un filtro de Kalman sin correspondencia conocida (UKF).

Para utilizar este simulador, se necesita primero descargar el paquete ‘‘Matlab Utilities’’ y añadirlo a la ruta de Matlab. Estas utilidades desempeñan toda clase de operaciones de filtrado de forma rápida.

Para configurar el simulador, se emplea un fichero auxiliar, en el que se pueden modificar, entre otros parámetros, los niveles de ruido en el proceso y la observación, número de ciclos, flags de visualización, etc. Inicialmente sólo se incluye la asociación de datos conocida.

5.3.4.1. Definición de funciones principales

- `function [G,iwp]= compute_steering(x,wp,iwp,minD,G,rateG, maxG,dt)`: Determina si se alcanzó el punto de referencia actual y calcula el cambio en el ángulo de dirección actual (G) para señalar hacia el punto de referencia actual.
- `function [V,G]= add_control_noise(V,G,Q,addnoise)`: Añade ruido de forma aleatoria a los valores nominales del control.
- `function predict(v,g,Q,WB,dt)`: Predice el estado del vehículo.
- `function observe_heading(phi,useheading)`: Realiza la actualización del estado para una medida dada como parámetro, phi, con ruido de medida fijo, sigmaPhi.
- `function [z,idf] = get_observations(x,lm,idf,rmax)`: Selecciona un conjunto de hitos, los cuales son visibles dentro del campo de visión semicircular del robot.
- `function z = add_observation_noise(z,R,addnoise)`: Añade ruido de medida aleatorio.
- `function [zf,idf,zn,table] = data_associate_known(x,z,idz, table)`: Esta función mantiene una tabla de consulta de característica/observación. Devuelve la tabla actualizada, el conjunto de observaciones asociadas y el conjunto de observaciones para las nuevas características.
- `function [zf,idf,zn] = data_associate(x,P,z,R,gate1,gate2)`: Obtiene de forma sencilla la asociación de datos del vecino más cercano.
- `function update(z,R,idf)`: Actualiza el estado y la covarianza.
- `function augment(z,R)`: Añade las nuevas características al estado.

5.3.4.2. Estructura del algoritmo

```
function data= ukfslam_sim(lm,wp)
...
while iwp ≠ 0

    [G,iwp] = compute_steering(xtrue,wp,iwp,AT_WAYPOINT,G,RATEG,
    MAXG,dt);

    ...

    [Vn,Gn] = add_control_noise(V,G,Q,SWITCH_CONTROL_NOISE);

    predict(Vn,Gn,QE,WHEELBASE,dt);

    observe_heading(xtrue(3),SWITCH_HEADING_KNOWN);

    ...

    if dtsum ≥ DT_OBSERVE

        ...

        [z,ftag_visible] = get_observations(xtrue,lm,ftag,
        MAX_RANGE);

        z = add_observation_noise(z,R,SWITCH_SENSOR_NOISE);

        if SWITCH_ASSOCIATION_KNOWN == 1

            [zf,idf,zn,da_table] = data_associate_known(XX,z,
            ftag_visible,da_table);

        else

            [zf,idf,zn] = data_associate(XX,PX,z,RE,GATE_REJECT,
            GATE_AUGMENT);

        end

        update(zf,RE,idf);

        augment(zn,RE);

    end

    ...

end
...
```

5.4. Modificaciones de los algoritmos

Para llevar a cabo las evaluaciones expuestas con anterioridad, hay que destacar las modificaciones realizadas sobre el software original, haciendo posible con ello una mejor comparativa. Dichas modificaciones consistieron en:

- Adaptación de entradas: permitir la modificación de los parámetros en tiempo de ejecución, y no a partir de los ficheros de configuración.
- Adaptación de salidas: modificar el código para suministrar datos de error, además de la salida gráfica.
- Implementación de la asociación de datos no conocida: permitir la ejecución de aquellos algoritmos que no la implementaban.

De todos estos cambios fue sin duda la incorporación de la asociación de datos no conocida la que mayor esfuerzo supuso. No tanto en el caso del UKF-Slam, donde pudo realizarse una traslación bastante directa del código utilizado en el EKF-Slam, como en el caso del FastSLAM. Para este último fue necesario introducir modificaciones en numerosas funciones del código, especialmente en lo relativo al manejo de las matrices de covarianza. Estos cambios eran necesarios para permitir la comparativa de todos los algoritmos en ambas situaciones, de asociación conocida y no conocida de datos.

En este proyecto se ha desarrollado asimismo un entorno que permite el lanzamiento de las simulaciones en modo tanda y la visualización de los resultados. Fue necesario adaptar el código para poder permitir la ejecución remota de las simulaciones en diferentes computadores y acortar así el tiempo requerido para la realización de las diferentes pruebas.

5.5. Sintonización de los parámetros propios

Como se ha indicado anteriormente, para realizar la comparativa es necesario primero analizar la parametrización de cada uno de los algoritmos. De este forma, se intenta partir de una configuración adecuada que no penalice de entrada el rendimiento que pueda ofrecer un determinado algoritmo.

En esta sección se realizarán diversas ejecuciones, en diferentes mapas, para los parámetros propios de los distintos algoritmos. Posteriormente se analizarán los resultados obtenidos de esas ejecuciones para determinar los valores de dichos parámetros que mejor comportamiento general hayan producido en los algoritmos. Hay que tener en cuenta que el elevado costo

computacional y el tiempo requerido para las simulaciones han dificultado la realización de las pruebas, que no han podido realizarse con toda la extensión que se hubiera deseado.

Para la ejecución de cada mapa con los diferentes algoritmos, primero se debe cargar en el sistema el conjunto de hitos y el conjunto de puntos de referencia que se utilizarán para la ejecución del mismo.

Hay que indicar que en las simulaciones se realiza un único recorrido a lo largo de la lista de puntos de referencia, de forma que en la parte final se realiza el cierre del bucle. Lo ideal sería mostrar varios ciclos, pero nuevamente las exigencias computacionales y tiempos de simulación no lo hicieron viable. Sin embargo, se realizaron pruebas puntuales para cada uno de los ensayos descritos en esta memoria, y se comprobó que en todos los casos, los ciclos posteriores al primero no son más que una repetición de la tendencia mostrada. Es decir, si en el primer ciclo se observa un patrón estable para el error, en los siguientes ciclos se repite ligeramente atenuado. Si el primer ciclo no converge, en los siguientes simplemente se observa un aumento de la divergencia. Por ello, se considera que un único ciclo es suficiente para extraer conclusiones sobre la evolución de los algoritmos.

5.5.1. Parámetros propios del algoritmo EKF-Slam

El algoritmo EKF-Slam presenta dos parámetros propios, el parámetro "Gate_Reject" y el parámetro "Gate_Augment", descritos en el apartado 5.3.1, los cuales constituyen las entradas de innovación para la asociación de datos. Dichos parámetros están directamente relacionados con la distancia de Mahalanobis.

5.5.1.1. Búsqueda de los parámetros propios óptimos

Para llevar a cabo la consecución de la búsqueda de los parámetros propios óptimos, se ha realizado una serie de ejecuciones, variando en cada momento el valor de alguno de dichos parámetros, y manteniendo invariables el resto de posibles parámetros que se encuentran en el fichero de configuración.

Hay que destacar, que las ejecuciones a tener en cuenta para la obtención de estos parámetros, serán aquellas en la que se ha utilizado asociación de datos no conocidos, pues son en estas ejecuciones donde estos parámetros influyen.

Los valores utilizados para cada uno de los parámetros han sido los siguientes:

- Gate_Reject: 1.0, 4.0, 9.0, 16.0.
- Gate_Augment: 10.0, 25.0, 50.0.

Nota: la combinación Gate_Reject = 16.0 y Gate_Augment = 10.0 no se tiene en cuenta, ajustándose a las definiciones de dichos parámetros.

El resultado final que se obtiene para cada combinación de parámetros es el resultado del promedio de diez ejecuciones, para cada una de las combinaciones, y para cada uno de los mapas definidos en la sección 5.2.

Los resultados finales obtenidos se visualizan en las gráficas de las figuras 5.3, 5.4, 5.5 y 5.6. En dichas gráficas se muestra, en el eje de abcisas el tiempo transcurrido durante la ejecución del algoritmo en el mapa (en segundos), y en el eje de ordenadas el error medio cometido por el algoritmo (en metros). El error se calcula como la diferencia en valor absoluto entre las posiciones reales del robot en cada instante de tiempo y el estado estimado por cada algoritmo.

En algunos casos se muestra la trayectoria real seguida por el robot en el mapa y su estimación.

Una vez realizadas todas y cada una de las ejecuciones, en búsqueda de los parámetros propios óptimos para el algoritmo EKF-Slam, se eligen los parámetros que mejor comportamiento global presentan para el conjunto de todos los mapas. De esta forma, se pueden fijar los parámetros propios con los valores elegidos en esta sección, y centrar las pruebas en otros parámetros.

Ejecutadas las diferentes combinaciones de parámetros propios del algoritmo EKF-Slam, se obtiene como resultado que la mejor combinación global es la formada por Gate_Reject = 4.0 y Gate_Augment = 25.0. En las figuras 5.7 y 5.8 se representan los resultados del promedio de las ejecuciones del algoritmo EKF-Slam para cada mapa, con los valores de los parámetros propios indicados anteriormente.

Aunque de forma individual para cada mapa, la ejecución del algoritmo EKF-Slam pueda obtener un menor error con parámetros propios diferentes a los seleccionados, se ha decidido elegir aquellos parámetros que presenten un comportamiento global moderado en todos los mapas, aunque ello suponga

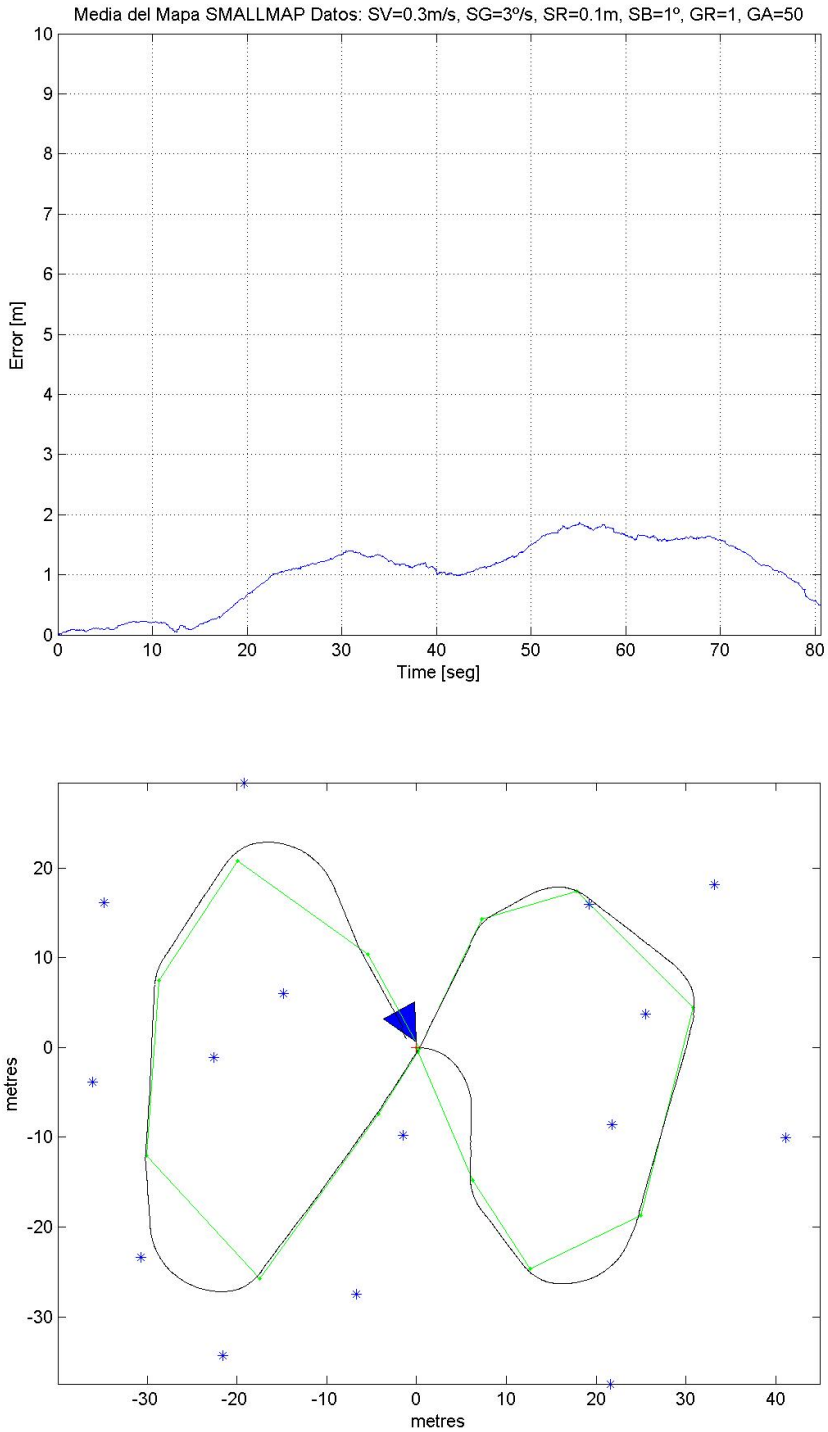


Figura 5.3: Ejemplo de resultados del promedio de las ejecuciones del algoritmo EKF-Slam con el mapa SMALLMAP y simulación de una ejecución con Gate_Reject = 1.0 y Gate_Augment = 50.0.

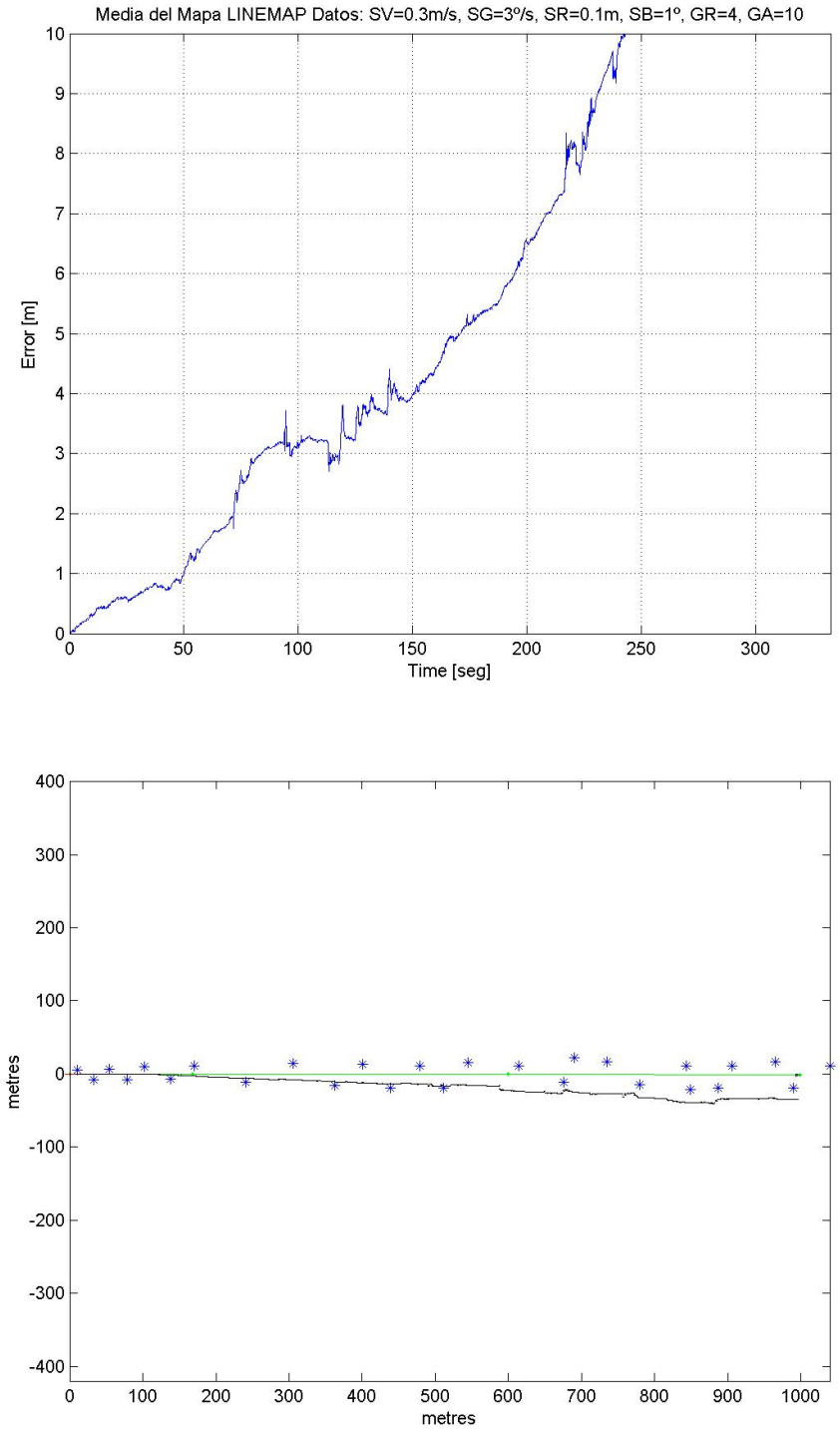


Figura 5.4: Ejemplo de resultados del promedio de las ejecuciones del algoritmo EKF-Slam con el mapa LINEMAP y simulación de una ejecución con Gate_Reject = 4.0 y Gate_Augment = 10.0.

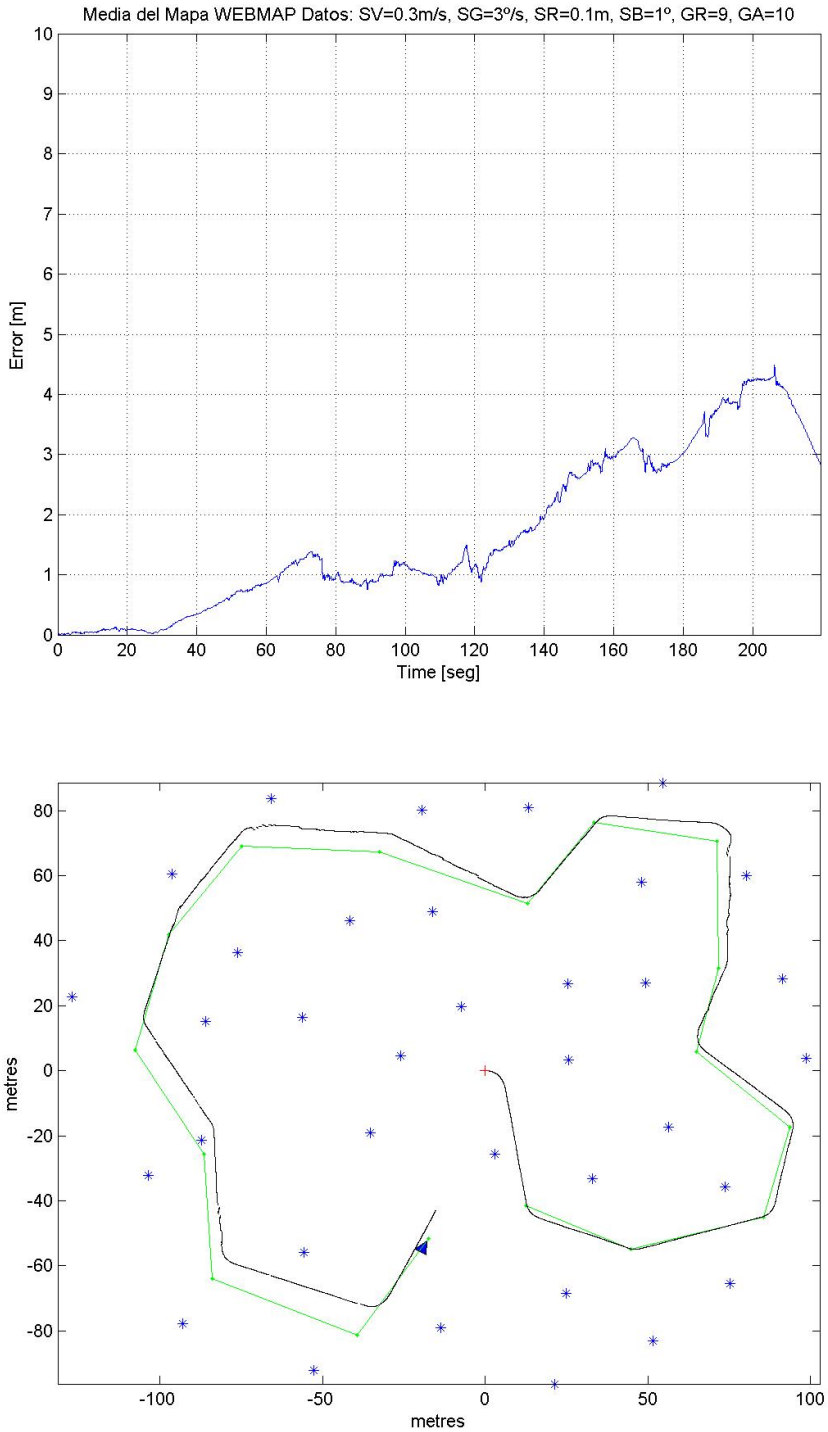


Figura 5.5: Ejemplo de resultados del promedio de las ejecuciones del algoritmo EKF-Slam con el mapa LINEMAP y simulación de una ejecución con Gate_Reject = 9.0 y Gate_Augment = 10.0.

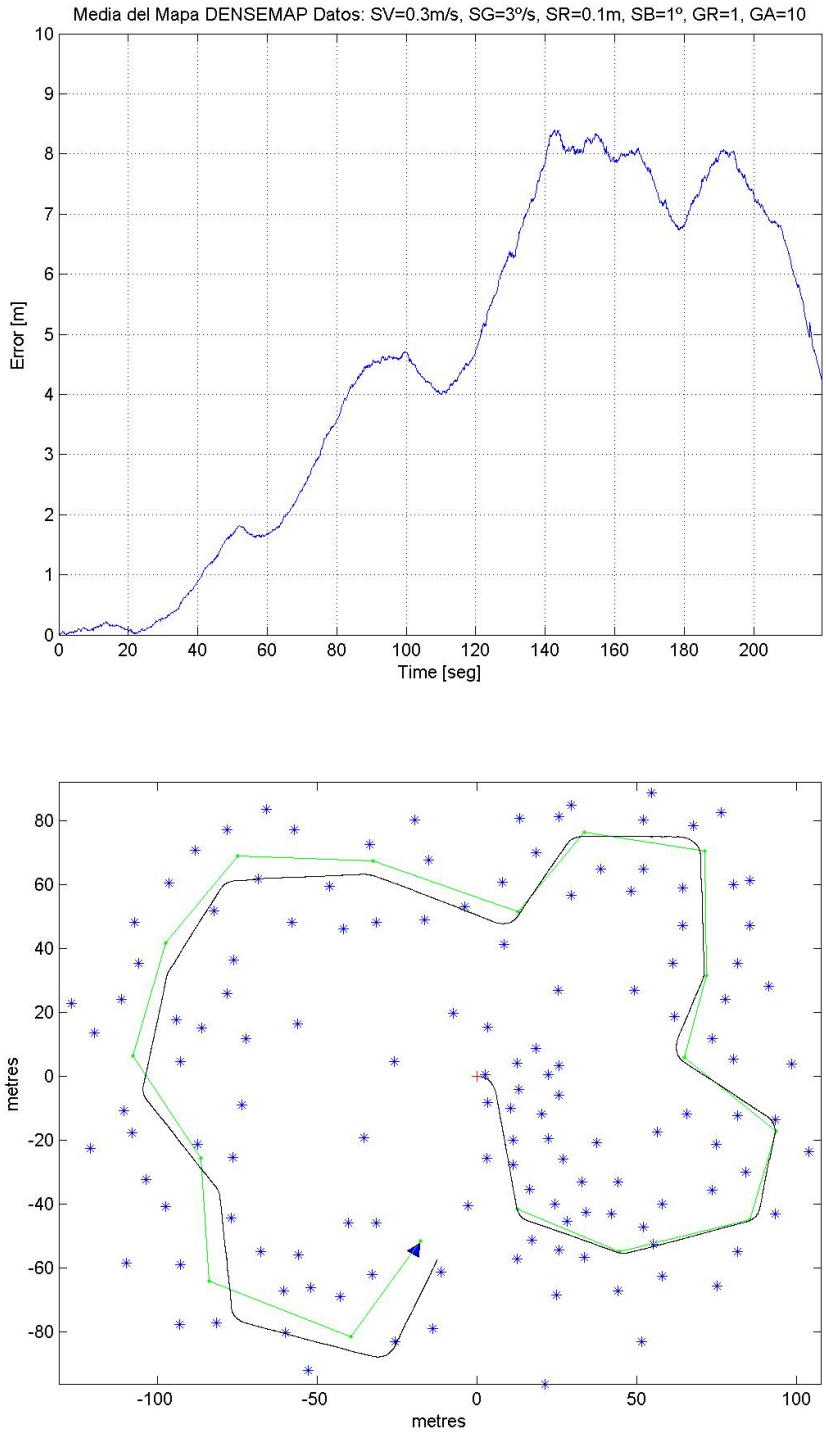


Figura 5.6: Ejemplo de resultados del promedio de las ejecuciones del algoritmo EKF-Slam con el mapa LINEMAP y simulación de una ejecución con Gate_Reject = 1.0 y Gate_Augment = 10.0.

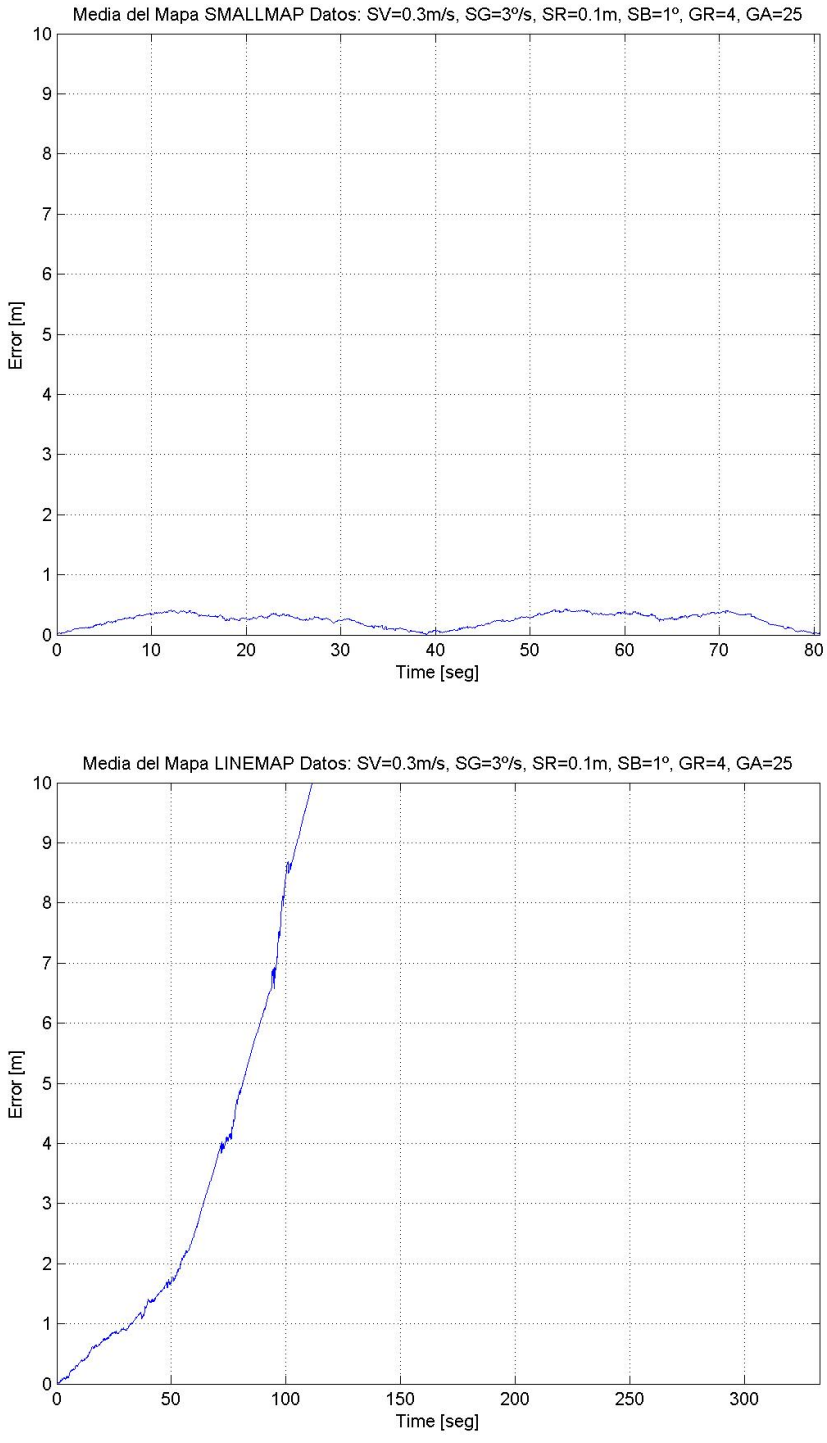


Figura 5.7: Resultado del promedio de las ejecuciones del algoritmo EKF-Slam con Gate_Reject = 4.0 y Gate_Augment = 25.0 para los mapas "SMALLMAP" y "LINEMAP".

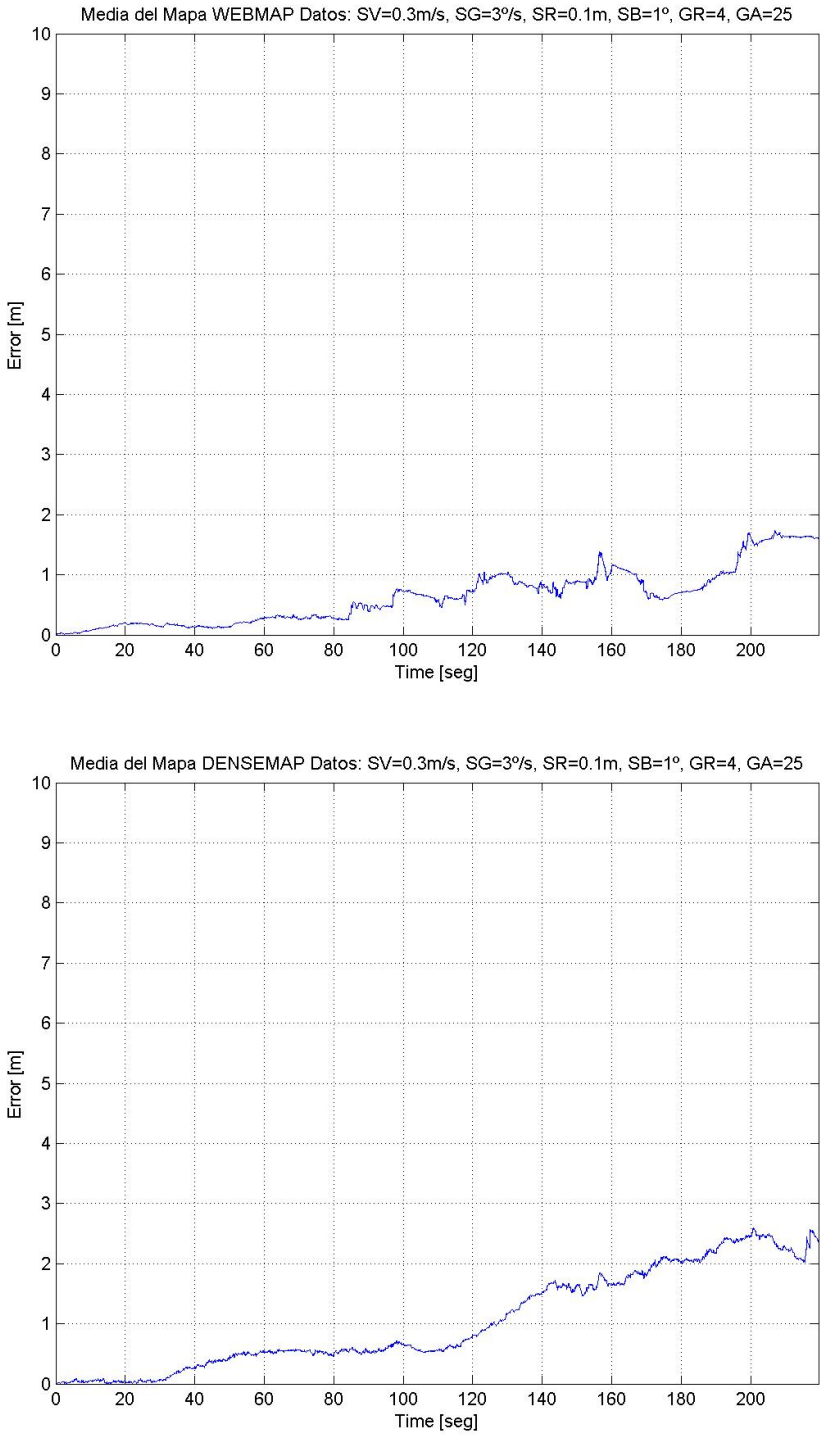


Figura 5.8: Resultado del promedio de las ejecuciones del algoritmo EKF-Slam con Gate_Reject = 4.0 y Gate_Augment = 25.0 para los mapas "WEBMAP" y "DENSEMAP".

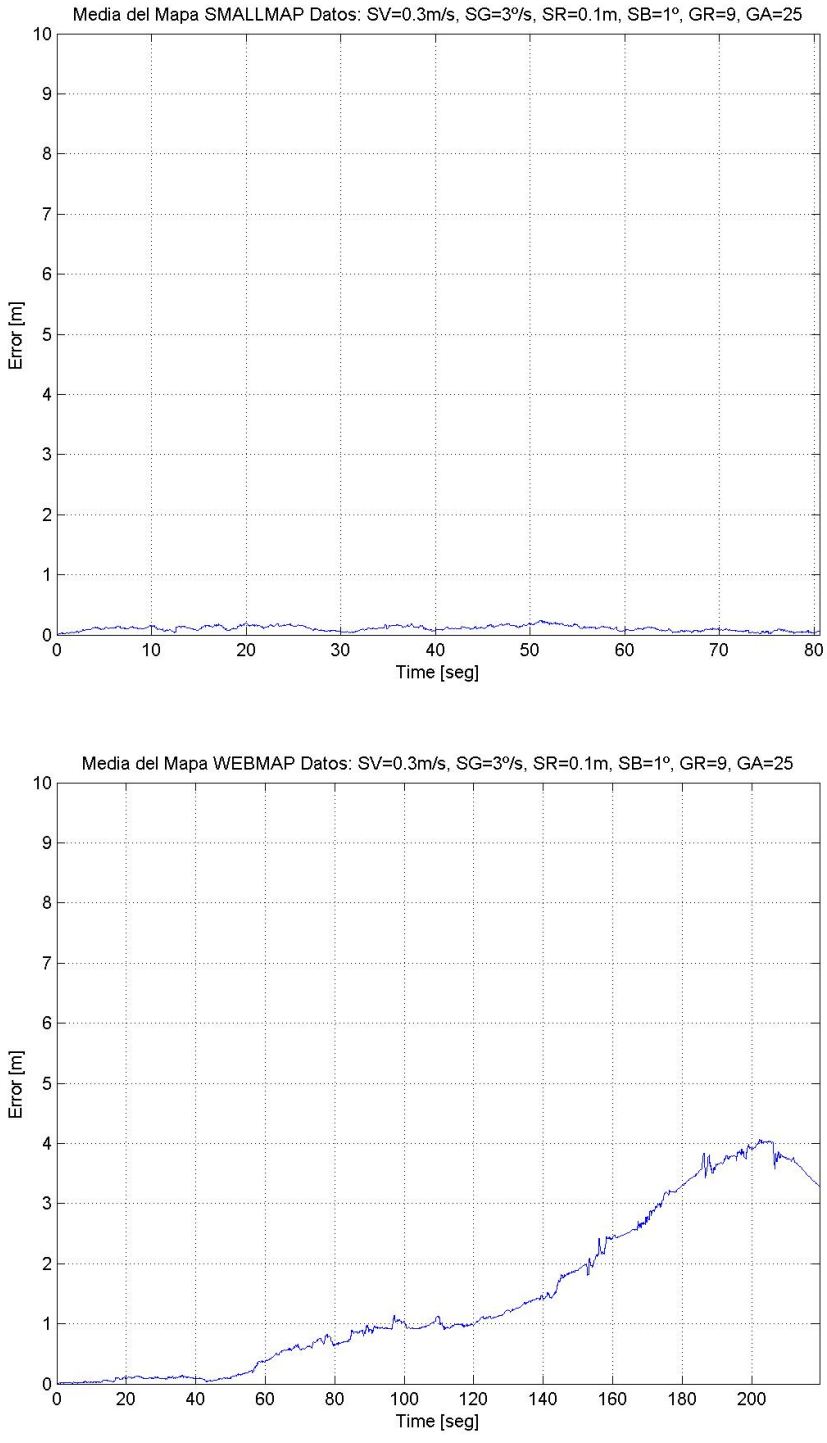


Figura 5.9: Ejemplos de ejecuciones del algoritmo EKF-Slam, donde los mejores parámetros propios para el primero, perjudican al segundo. (I)

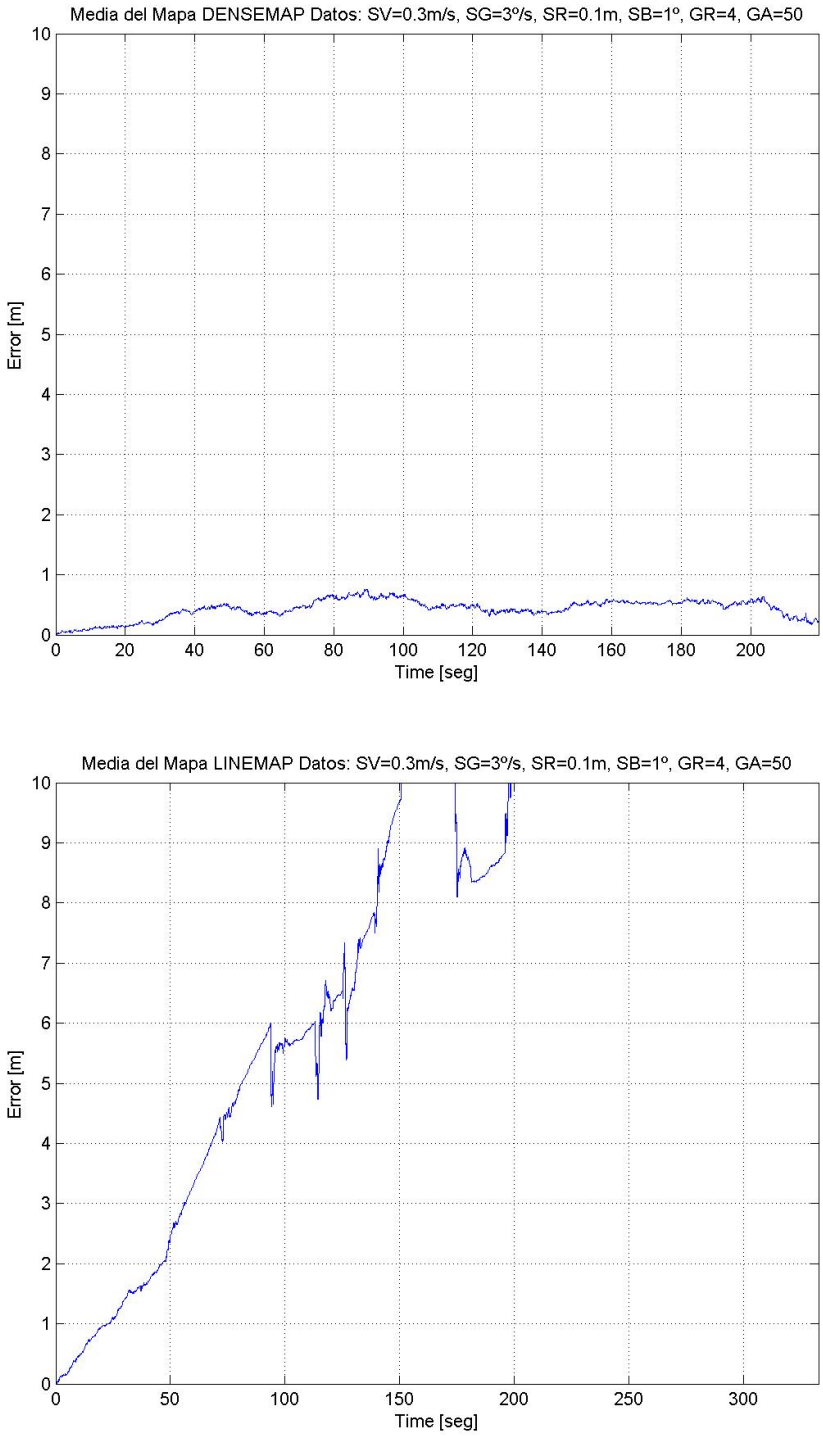


Figura 5.10: Ejemplos de ejecuciones del algoritmo EKF-Slam, donde los mejores parámetros propios para el primero, perjudican al segundo. (II)

obtener un error superior individualmente, dentro de un margen permisible, con respecto a la elección de los mejores parámetros.

Como ejemplo a lo mencionado anteriormente, podemos visualizar la figura 5.9 y la figura 5.10, en donde la primera gráfica de ambas representa la mejor combinación de los parámetros Gate_Reject y Gate_Augment, y sin embargo, en las segundas gráficas, dicha combinación elegida presenta un comportamiento poco deseado. Es por ello que, la elección de los parámetros Gate_Reject = 4.0 y Gate_Augment = 25.0, mencionada al principio de esta sección, es la combinación que presenta un mejor comportamiento global para todos los mapas, aunque con ello se perjudique ligeramente las caracteristicas e individualidades de cada uno de los mapas ante la ejecución del algoritmo EKF-Slam. A partir de este momento, esos valores serán los fijados para estos parámetros cuando se quiera utilizar la asociación de datos no conocidos.

5.5.2. Parámetros propios del algoritmo UKF-Slam

El algoritmo UKF-Slam presenta, al igual que el algoritmo EKF-Slam, los parámetros propios relativos a la asociación de datos no conocida ("Gate_Reject" y "Gate_Augment"), descritos en el apartado 5.3.1.

Los resultados obtenidos en las ejecuciones con este algoritmo son similares a los obtenidos para el caso del algoritmo EKF-Slam, debido a su similitud con éste en el desarrollo de la asociación de datos no conocidos.

5.5.3. Parámetros propios del algoritmo FastSlam

El algoritmo FastSlam presenta cuatro parámetros propios: los relativos a la asociación de datos no conocida y los asociados al número de partículas y el remuestreo, todos ellos descritos en el apartado 5.3.1.

5.5.3.1. Búsqueda de los parámetros propios óptimos

Al igual que se hizo en la búsqueda de los parámetros propios del algoritmo EKF-Slam, para realizar la búsqueda de los parámetros propios óptimos del algoritmo FastSlam, se ha realizado una serie de ejecuciones, variando en cada momento el valor de alguno de dichos parámetros, y manteniendo invariables el resto de posibles parámetros que se encuentran en el fichero de configuración. Se ha considerado el caso tanto de la asociación conocida como no conocida.

Aunque no se incluyen en la memoria por razones de espacio y por considerar que no aportan nada nuevo, se realizaron simulaciones con diferentes valores de los parámetros tipo Gate, llegándose a la conclusión de que los mismos valores que se seleccionaron como adecuados para el EKF-Slam (Gate_Reject = 4.0 y Gate_Augment = 25.0) resultan aquí también los que mejor resultado promedio arrojan.

Los valores utilizados para cada uno de los parámetros relacionados con las partículas han sido los siguientes:

- Nparticles: 10.0, 50.0, 100.0.
- Neffective: 0.25, 0.50, 0.75.

De la misma forma que con el algoritmo EKF-Slam, el resultado final que se obtiene para cada combinación de parámetros es el resultado del promedio de diez ejecuciones, para cada una de las combinaciones, y para cada uno de los mapas definidos en la sección 5.2.

Los resultados finales obtenidos se visualizan también en unas gráficas, como las representadas en las figuras 5.11, 5.12, 5.13 y 5.14, si hacemos uso de la asociación de datos no conocidos, o en las figuras 5.15, 5.16, 5.17 y 5.18, si hacemos uso de la asociación de datos conocidos. En el eje de abcisas se representa el tiempo transcurrido durante la ejecución del algoritmo en el mapa (en segundos), y en el eje de ordenadas el error medio cometido por el algoritmo (en metros), tal como sucediera con el algoritmo EKF-Slam. Además, se incluye en cada figura una simulación de algoritmo con cada mapa en cuestión, y con la combinación de parámetros propios elegidos en ese momento.

En algunos casos se muestra también la evolución de las diferentes partículas asociadas al mapa durante la ejecución.

Hay que aclarar que las representaciones del algoritmo FastSlam presentan unas gráficas escalonadas en los resultados del promedio. Esto es debido a los saltos que el algoritmo realiza en cada instante, en la búsqueda de la partícula que mejor se ajusta en cada momento. Otra alternativa hubiera sido la de seleccionar al final la mejor partícula global y representar sólo su trayectoria, lo que hubiera producido una gráfica más limpia. Sin embargo, las necesidades de almacenamiento de este último caso no lo hicieron viable.

Ejecutadas las diferentes combinaciones de parámetros propios del algoritmo FastSlam, se obtiene como resultado que la mejor combinación global es la

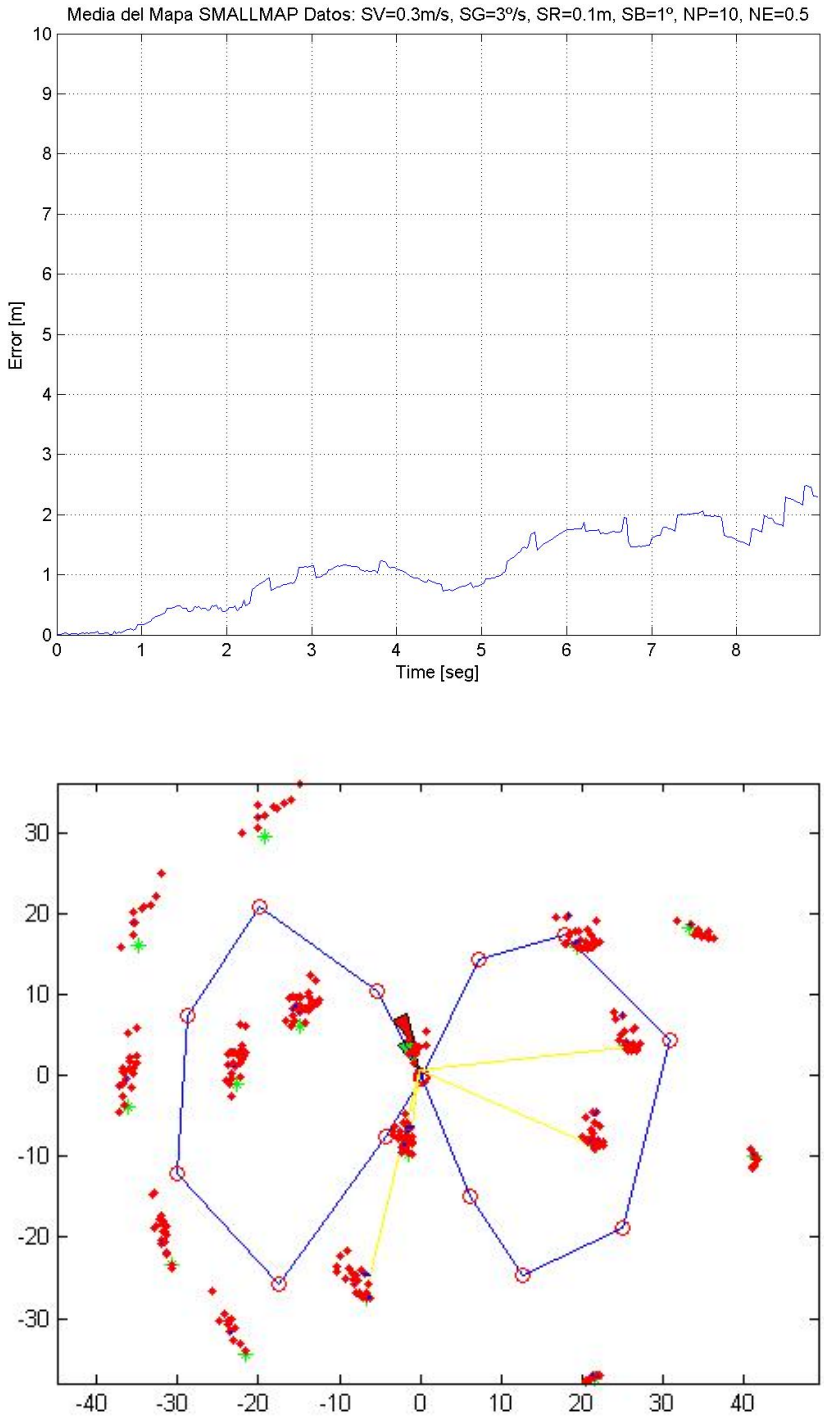


Figura 5.11: Ejemplo de resultados del promedio de las ejecuciones del algoritmo FastSlam con el mapa SMALLMAP y simulación de una ejecución con Nparticles = 10.0 y Neffective = 0.50, con asocación de datos no conocidos.

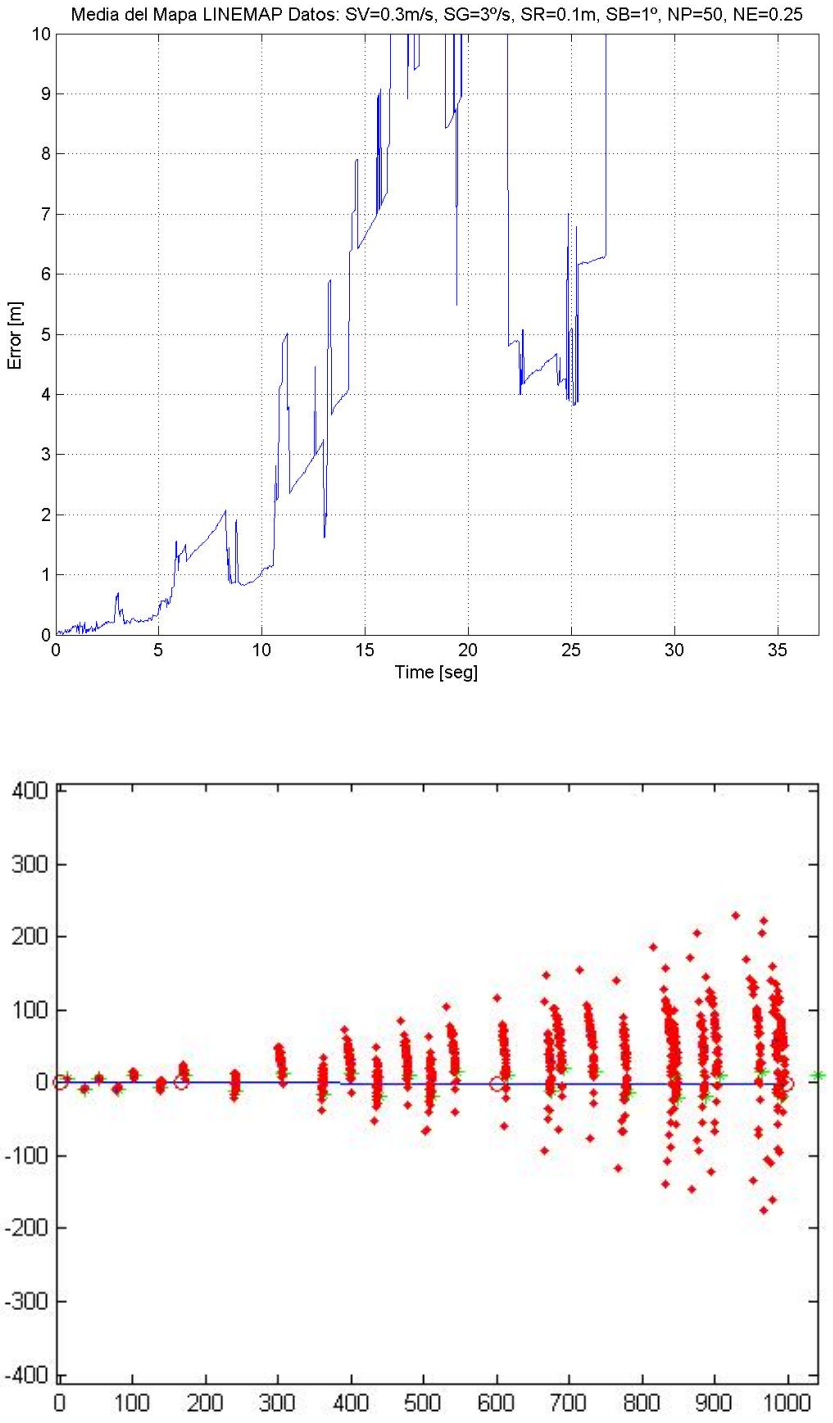


Figura 5.12: Ejemplo de resultados del promedio de las ejecuciones del algoritmo FastSlam con el mapa LINEMAP y simulación de una ejecución con Nparticles = 50.0 y Neffective = 0.25, con asocación de datos no conocidos.

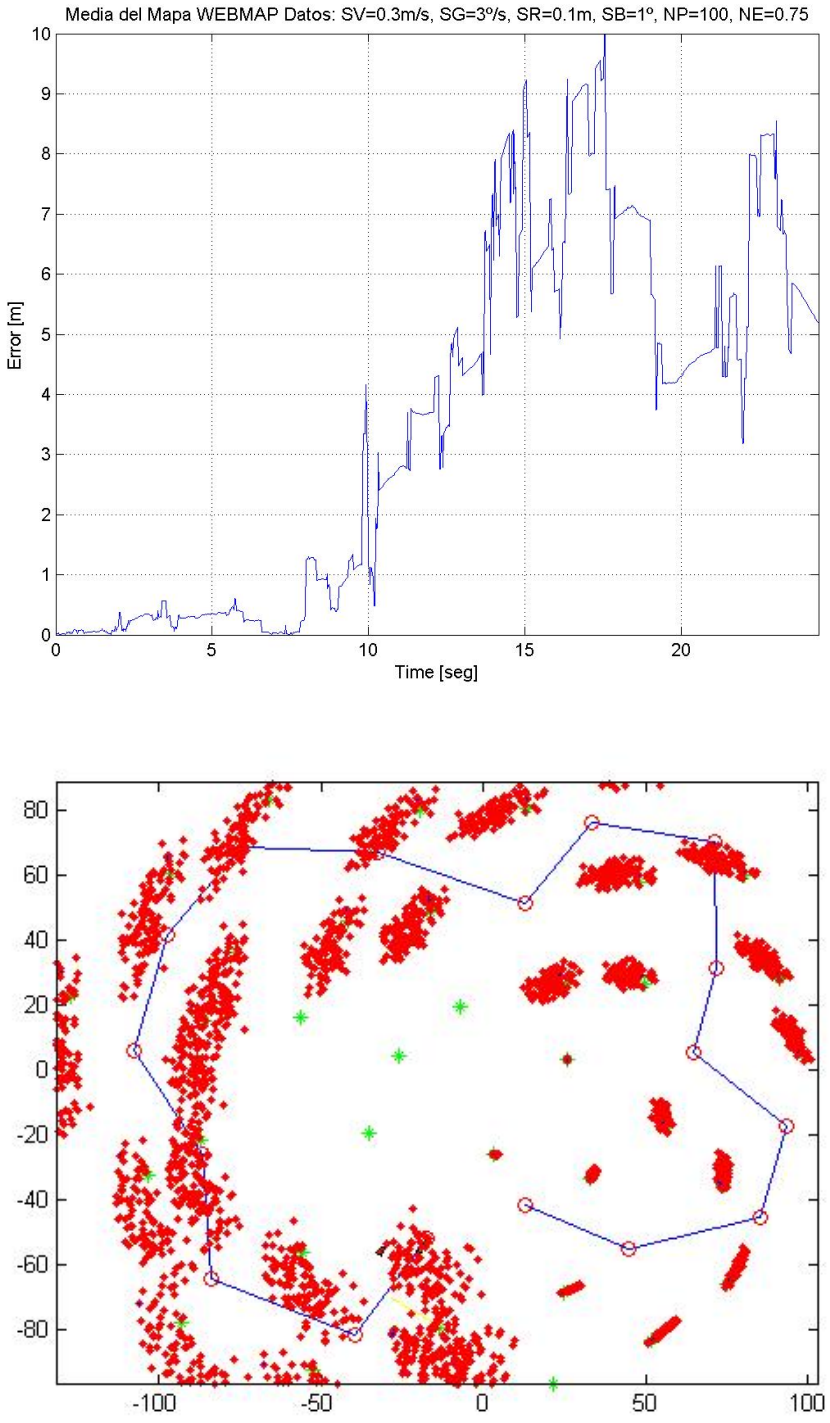


Figura 5.13: Ejemplo de resultados del promedio de las ejecuciones del algoritmo FastSlam con el mapa WEBMAP y simulación de una ejecución con Nparticles = 100.0 y Neffective = 0.75, con asocación de datos no conocidos.

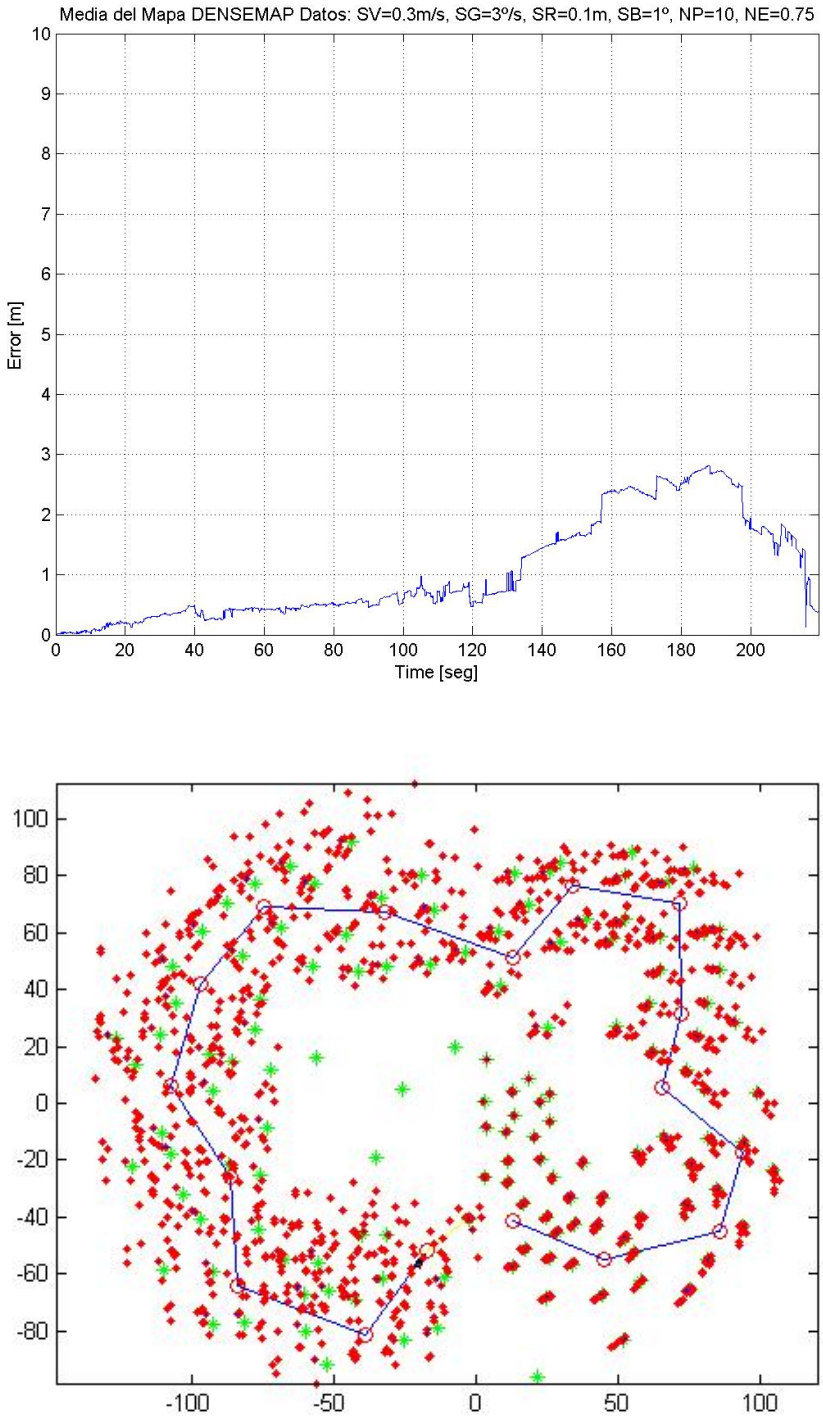


Figura 5.14: Ejemplo de resultados del promedio de las ejecuciones del algoritmo FastSlam con el mapa DENSEMAP y simulación de una ejecución con Nparticles = 10.0 y Neffective = 0.75, con asocación de datos no conocidos.

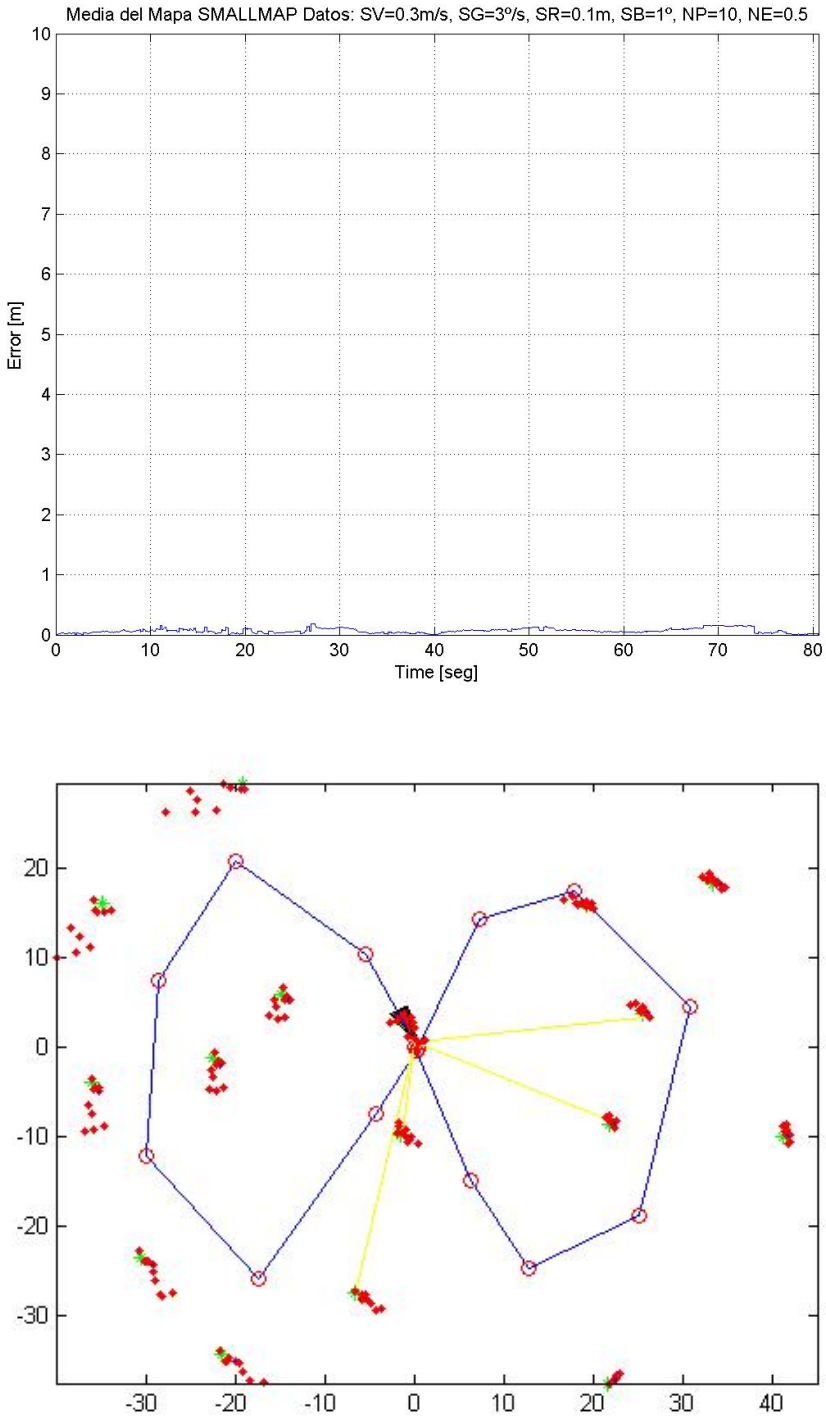


Figura 5.15: Ejemplo de resultados del promedio de las ejecuciones del algoritmo FastSlam con el mapa SMALLMAP y simulación de una ejecución con Nparticles = 10.0 y Neffective = 0.50, con asocación de datos conocidos.

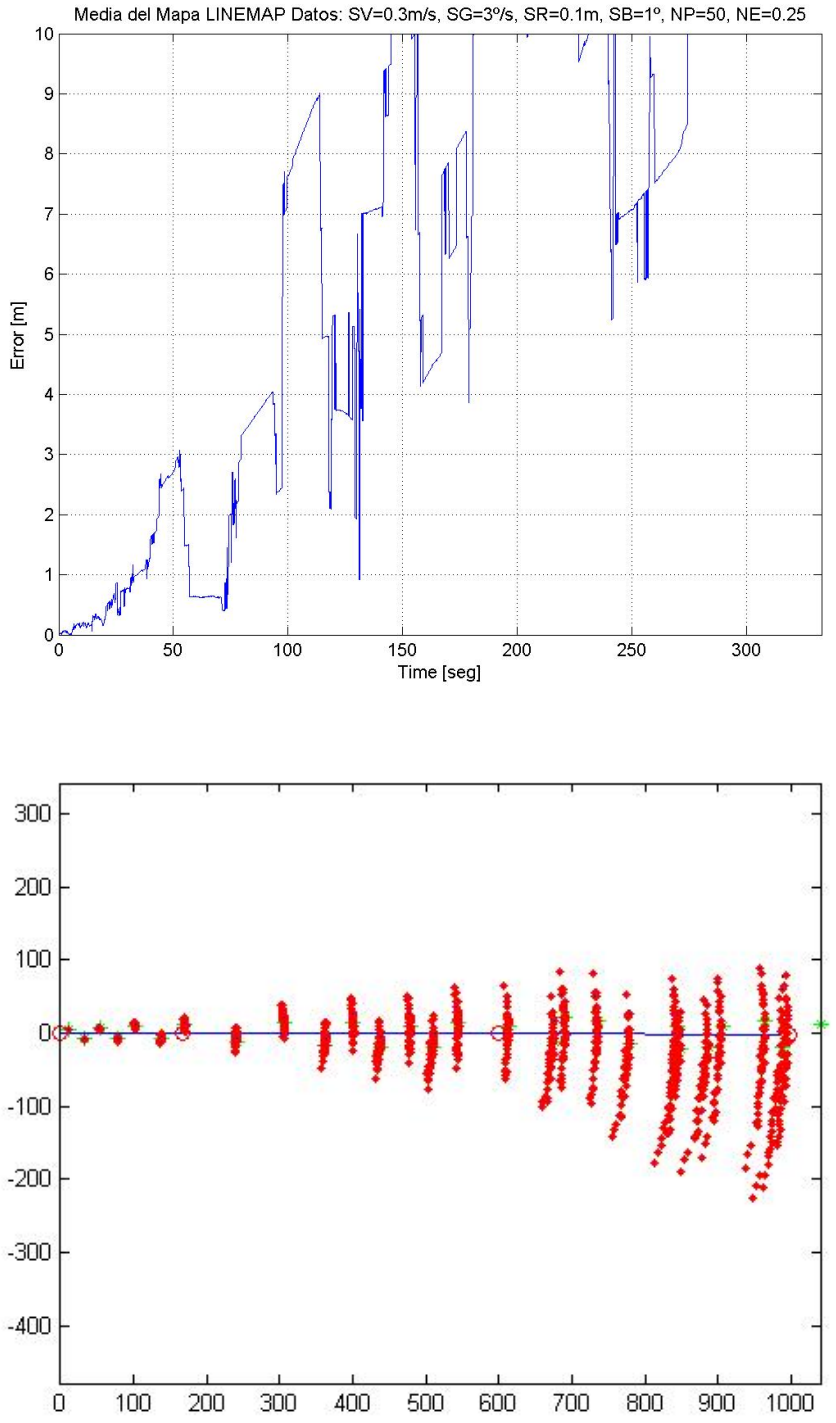


Figura 5.16: Ejemplo de resultados del promedio de las ejecuciones del algoritmo FastSlam con el mapa LINEMAP y simulación de una ejecución con Nparticles = 50.0 y Neffective = 0.25, con asocación de datos conocidos.

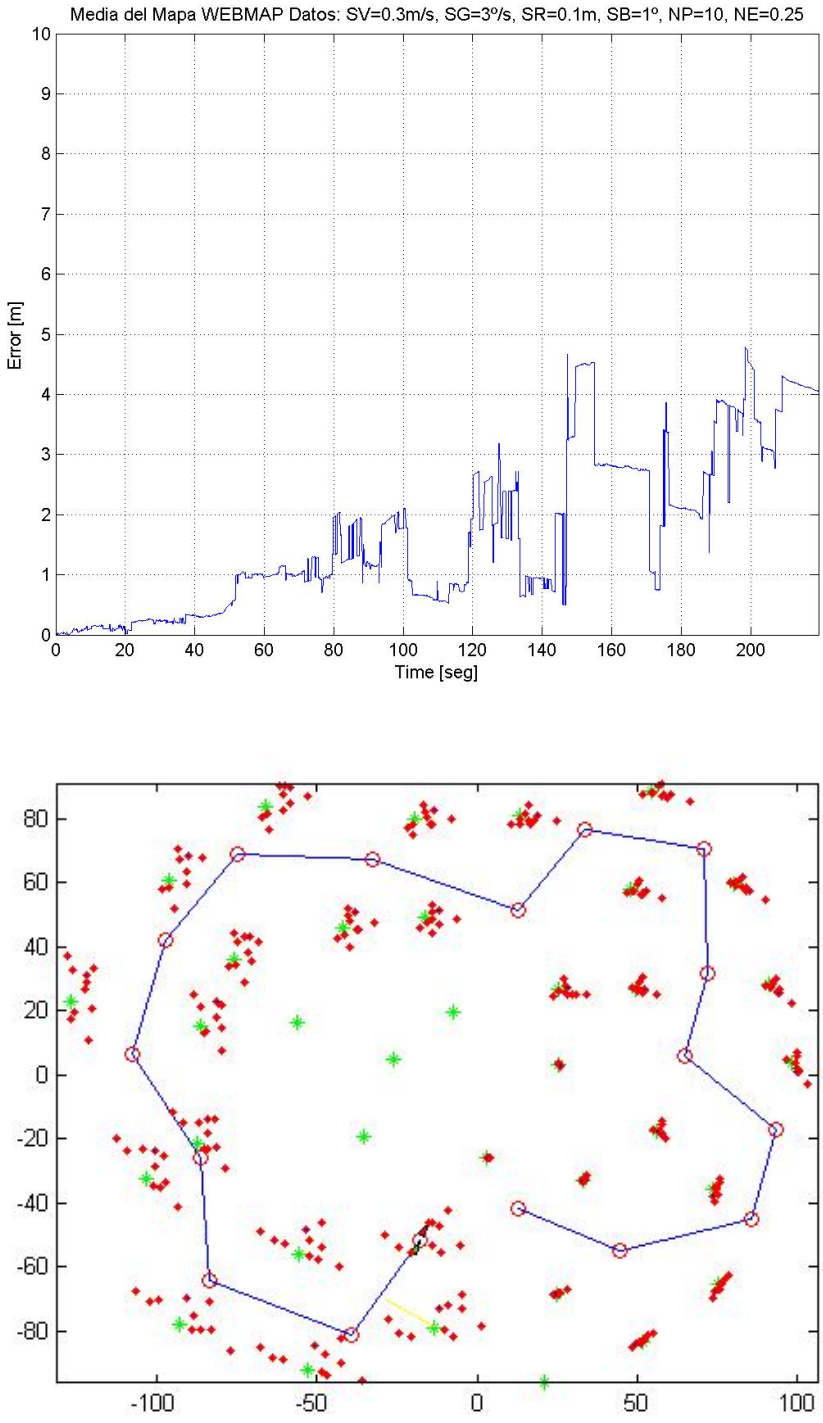


Figura 5.17: Ejemplo de resultados del promedio de las ejecuciones del algoritmo FastSlam con el mapa WEBMAP y simulación de una ejecución con Nparticles = 10.0 y Neffective = 0.25, con asocación de datos conocidos.

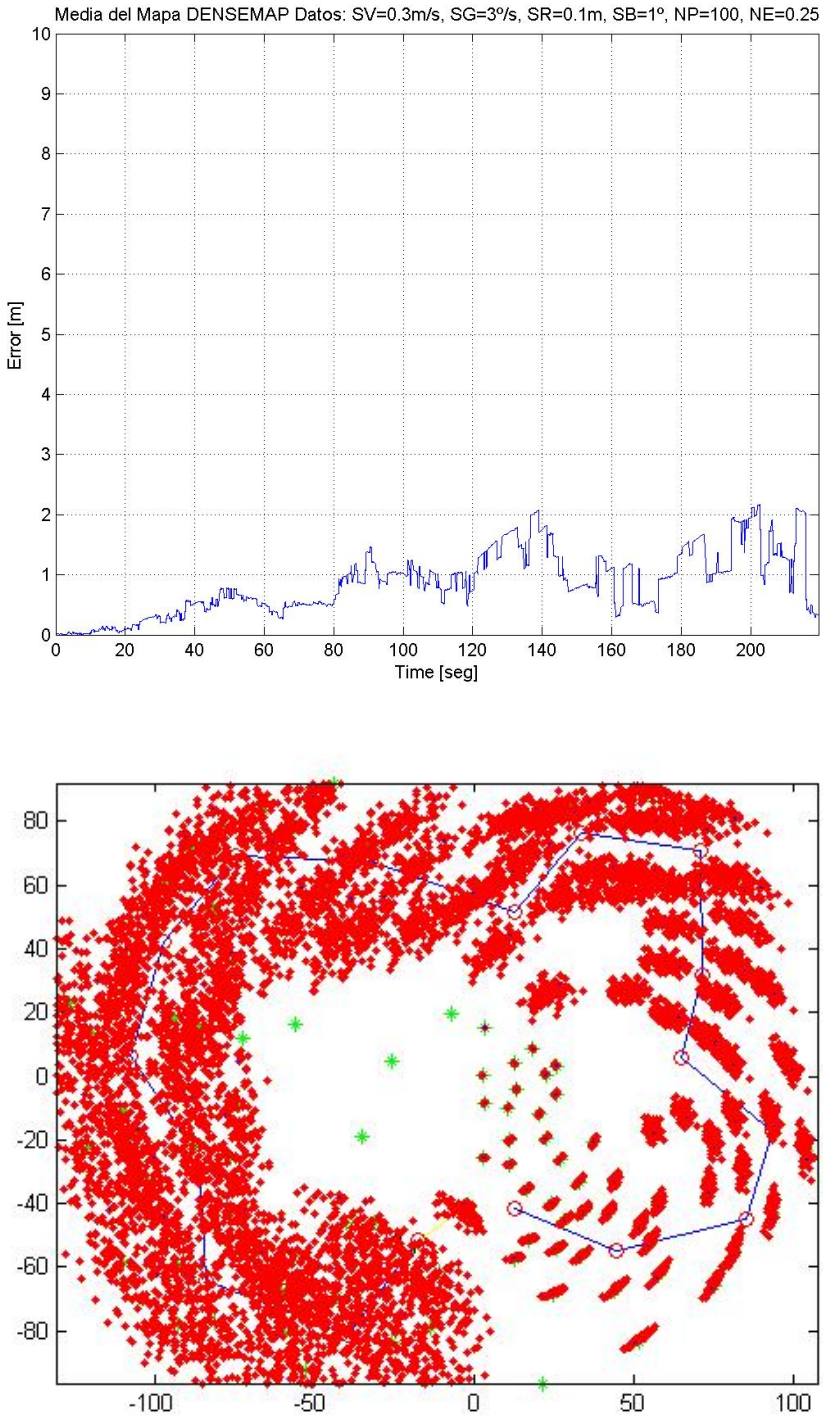


Figura 5.18: Ejemplo de resultados del promedio de las ejecuciones del algoritmo FastSlam con el mapa DENSEMAP y simulación de una ejecución con Nparticles = 100.0 y Neffective = 0.25, con asocación de datos conocidos.

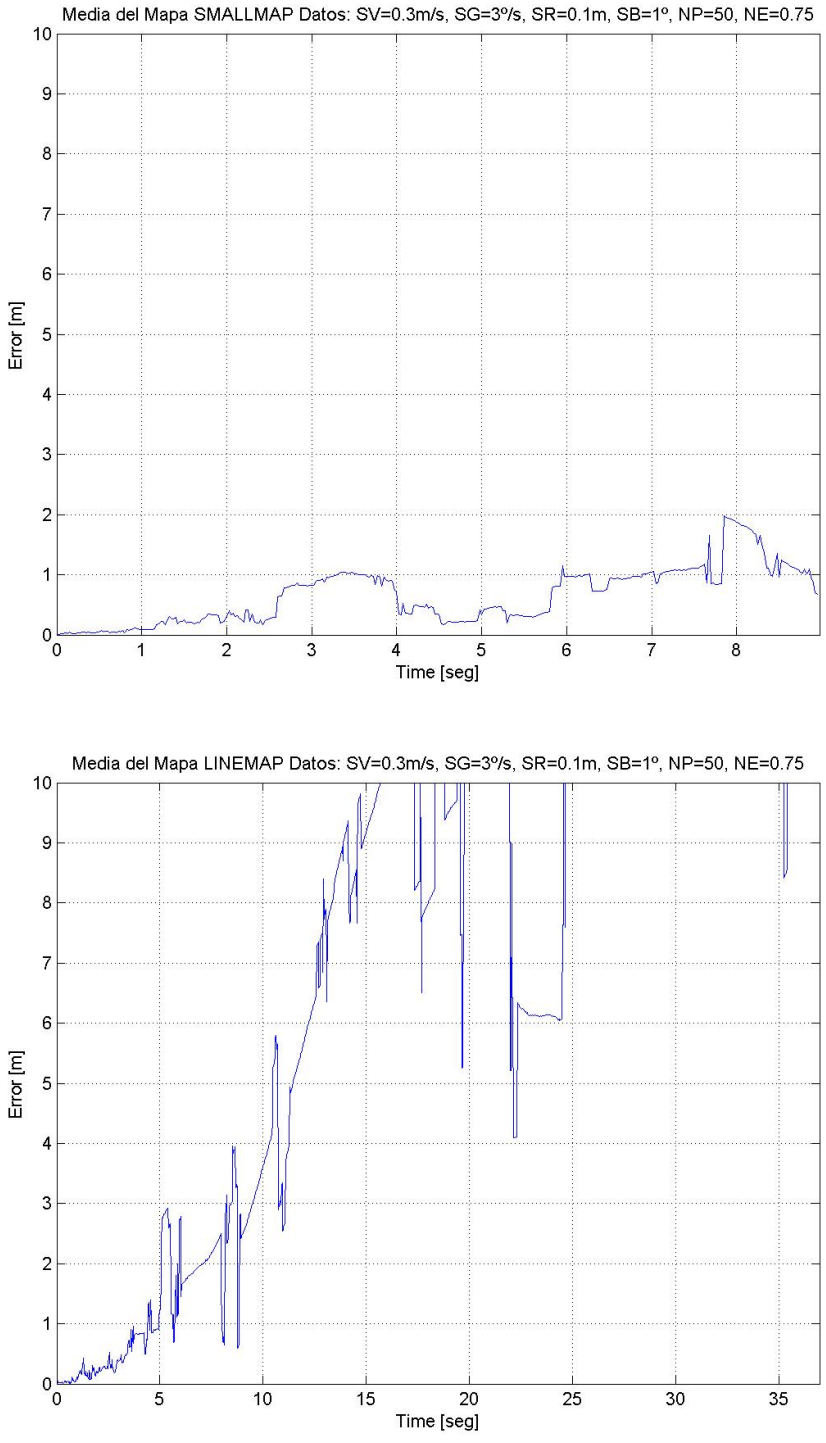


Figura 5.19: Resultado del promedio de las ejecuciones del algoritmo FastSlam con Nparticles = 50.0 y Neffective = 0.75 para los mapas "SMALLMAP" y "LINEMAP" con asociación de datos no conocidos.

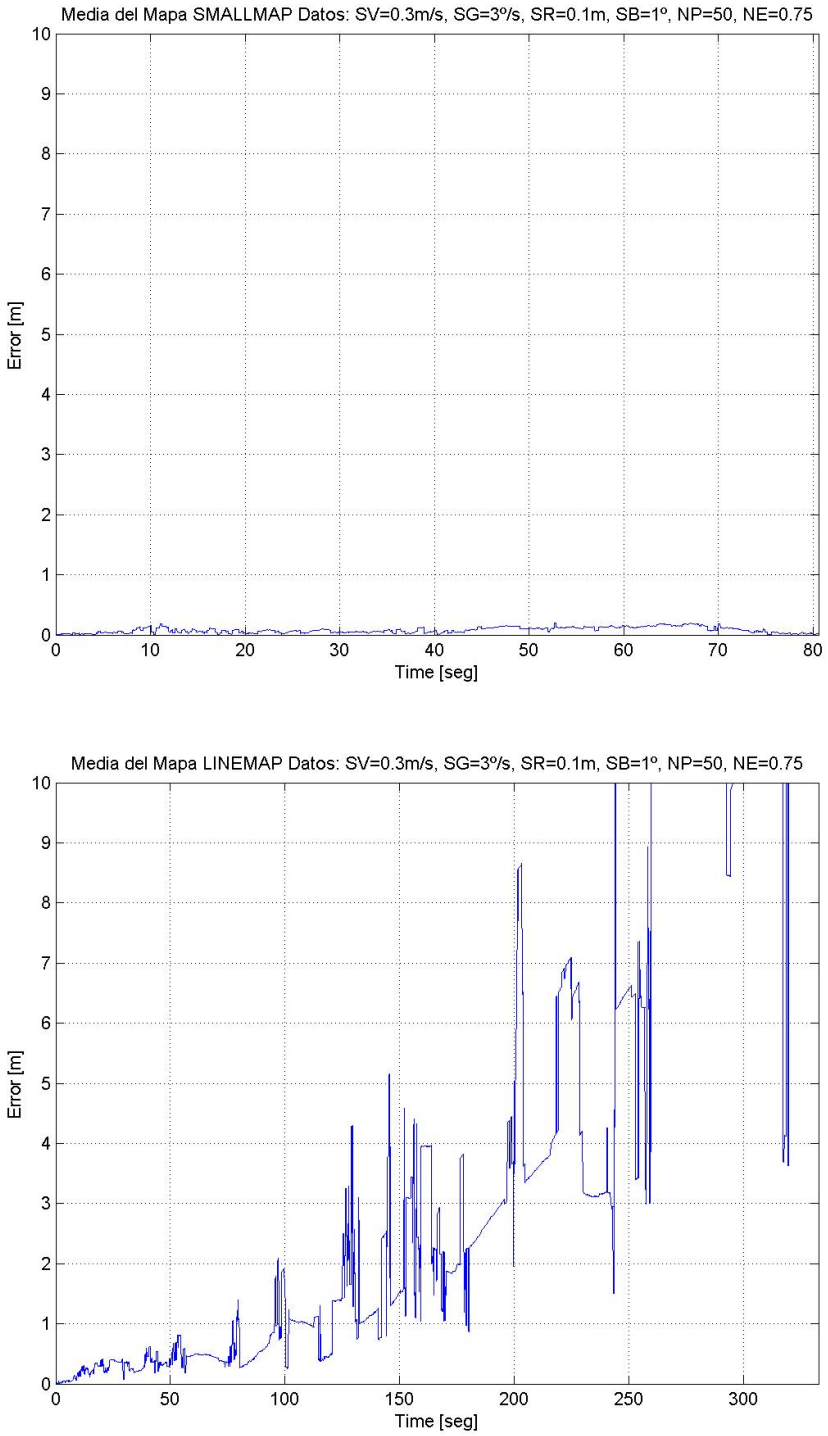


Figura 5.20: Resultado del promedio de las ejecuciones del algoritmo FastSlam con Nparticles = 50.0 y Neffective = 0.75 para los mapas "SMALLMAP" y "LINEMAP" con asociación de datos conocidos.

formada por Nparticles = 50.0 y Neffective = 0.75. En las figuras 5.19 y 5.20 se representan los resultados del promedio de las ejecuciones del algoritmo FastSlam, con los valores de los parámetros propios indicados anteriormente, para los mapas "SMALLMAP" y "LINEMAP", con asociación de datos no conocidos y con asociación de datos conocidos, respectivamente. Y en las figuras 5.21 y 5.22 se representan los resultados del promedio de las ejecuciones del algoritmo FastSlam, también con los valores de los parámetros propios indicados anteriormente, para los mapas "WEBMAP" y "DENSEMAP", con asociación de datos no conocidos y con asociación de datos conocidos, respectivamente.

En las figuras 5.23, 5.24, 5.25 y 5.26 se representan varias simulaciones del algoritmo FastSlam en varios mapas, utilizando asociación de datos conocidos y asociación de datos no conocidos. Todas estas simulaciones están realizadas con los parámetos propios que se han indicado anteriormente.

De la misma forma que se indicara anteriormente con el algoritmo EKF-Slam, puede suceder que de forma individual, para cada mapa, la ejecución del algoritmo FastSlam pueda obtener un menor error con parámetros propios diferentes a los seleccionados. A pesar de todo, se ha decidido elegir aquellos parámetros que presenten un comportamiento global moderado en todos los mapas, aunque ello suponga obtener un error superior individualmente, con respecto a la elección de los mejores parámetros.

Como ejemplo de lo mencionado anteriormente, con respecto a ejecuciones con un menor error de forma individual para un mapa determinado, podemos visualizar las figuras 5.27, 5.28, 5.29 y 5.30, en donde la primera gráfica de todas ellas representa la mejor combinación de los parámetros Nparticles y Neffective para ese mapa, y sin embargo, en las segundas gráficas, la combinación elegida presenta un comportamiento no deseado para el mapa tratado. Debido a ello, la combinación que presenta un mejor comportamiento global para todos los mapas, es la que presenta los parámetros Nparticles = 50.0 y Neffective = 0.75, aunque dicha combinación pueda perjudicar ligeramente las caracteristicas e individualidades de cada uno de los mapas ante la ejecución del algoritmo FastSlam.

Conviene aclarar que se han detectado ciertas anomalías en el comportamiento de los algoritmos de FastSlam. En teoría, un aumento de partículas siempre tendría que traducirse en un mejor resultado final. Sorprende, sin embargo, que pueden obtenerse resultados con un número muy reducido de partículas (5 por ejemplo) que en ocasiones superan a los obtenidos con

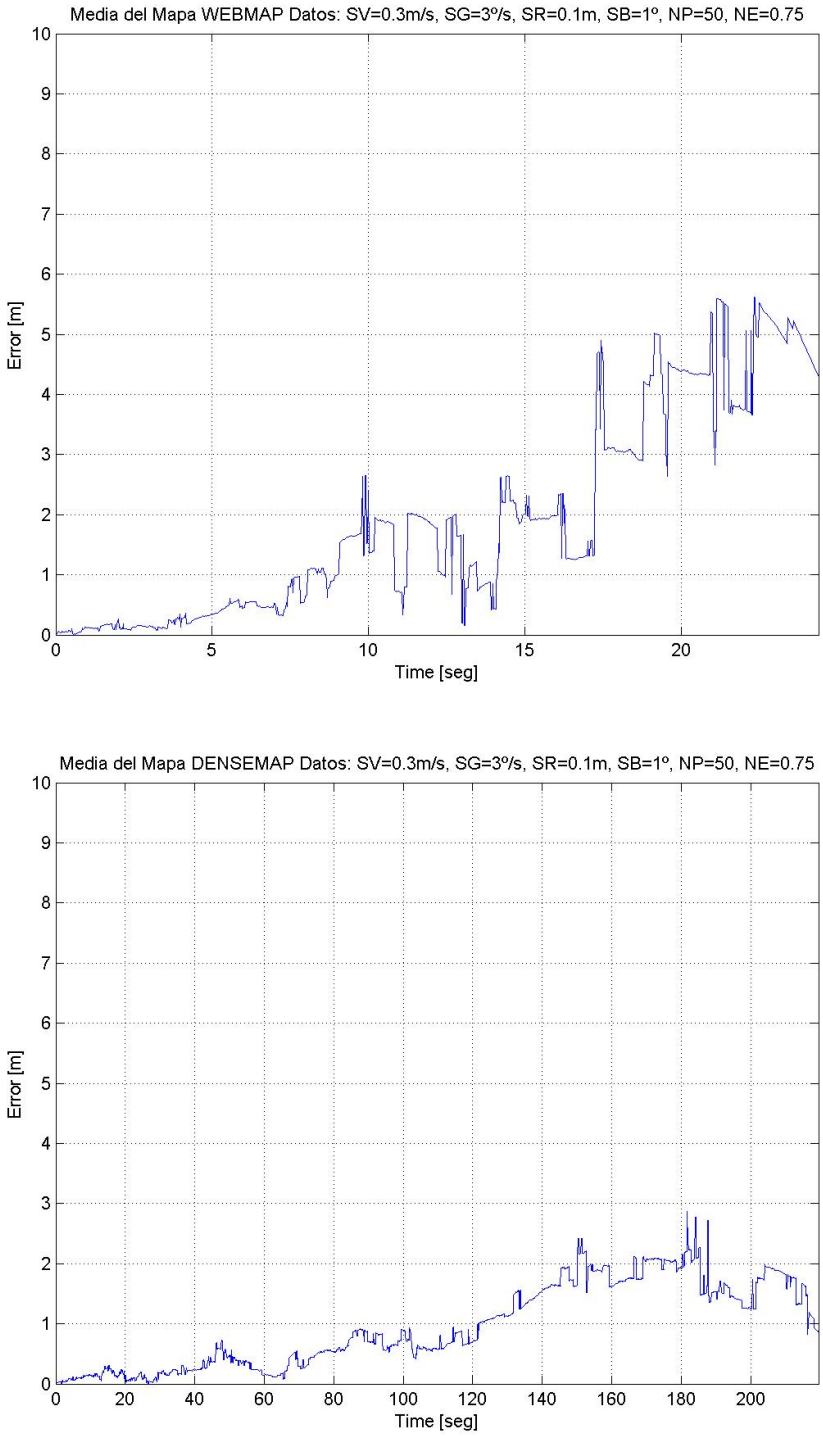


Figura 5.21: Resultado del promedio de las ejecuciones del algoritmo FastSlam con Nparticles = 50.0 y Neffective = 0.75 para los mapas "WEBMAP" y "DENSEMAP" con asociación de datos no conocidos.

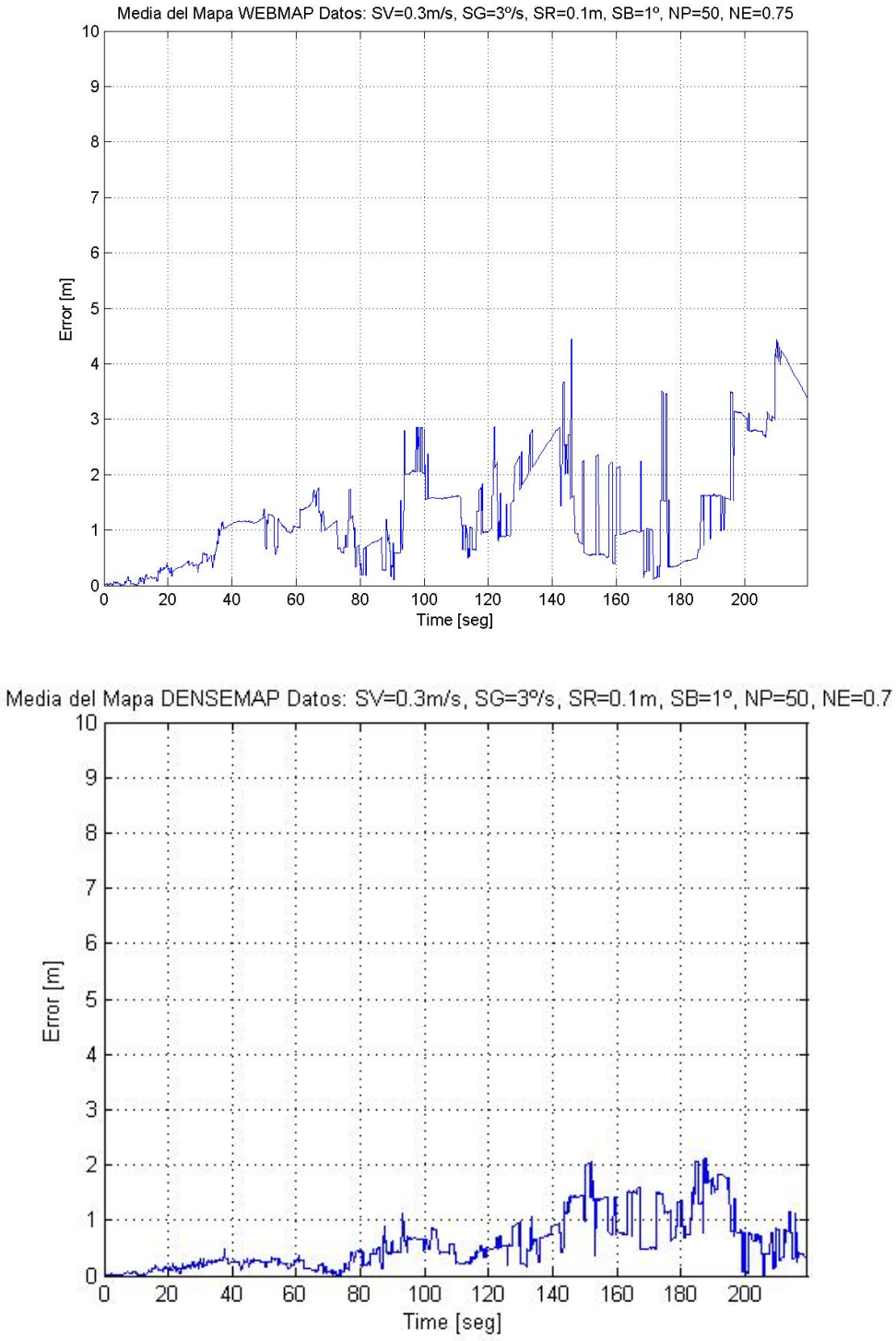


Figura 5.22: Resultado del promedio de las ejecuciones del algoritmo FastSlam con Nparticles = 50.0 y Neffective = 0.75 para los mapas "WEBMAP" y "DENSEMAP" con asociación de datos conocidos.

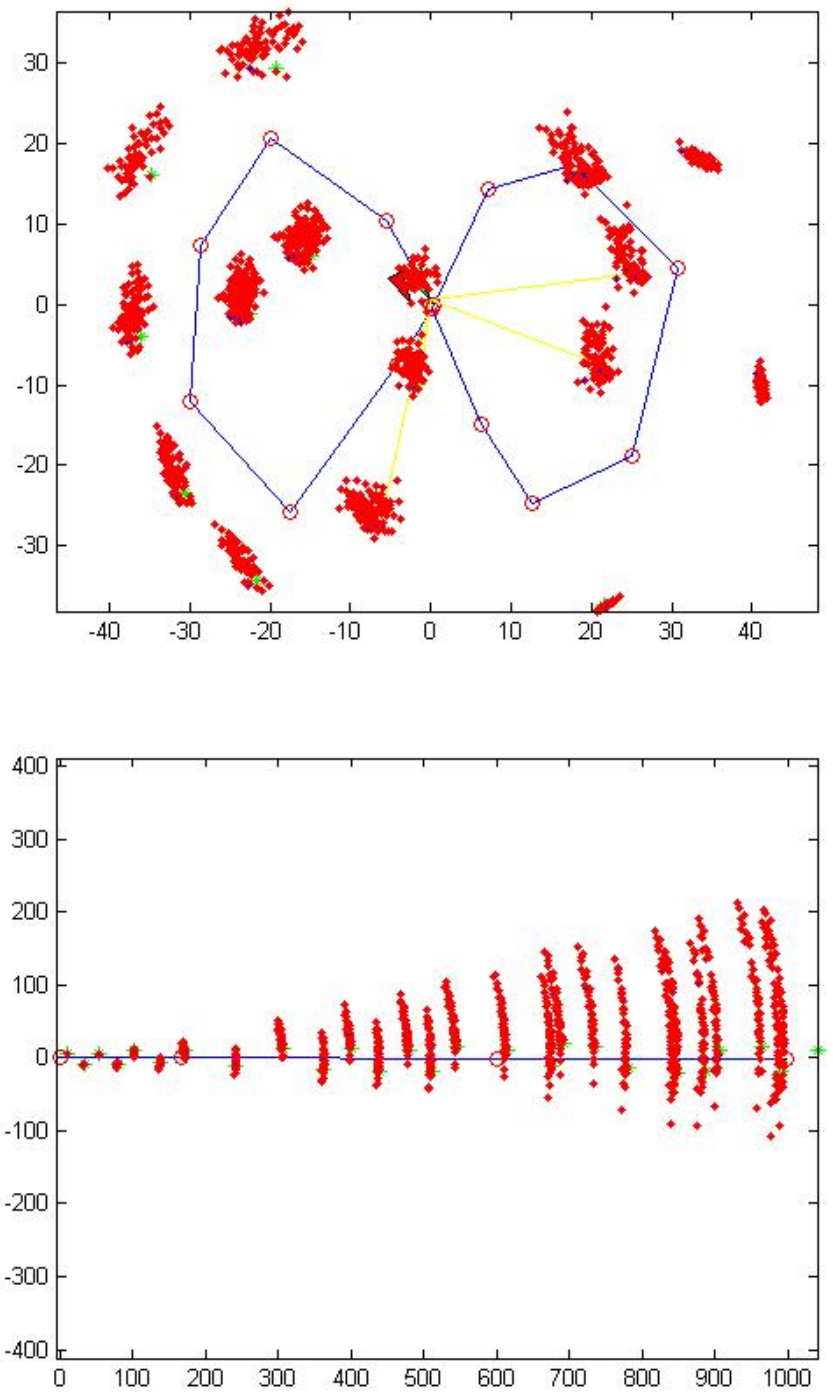


Figura 5.23: Ejemplos de simulaciones en diferentes mapas del algoritmo FastSlam, con los parámetros propios seleccionados y asociación de datos no conocidos. (I)

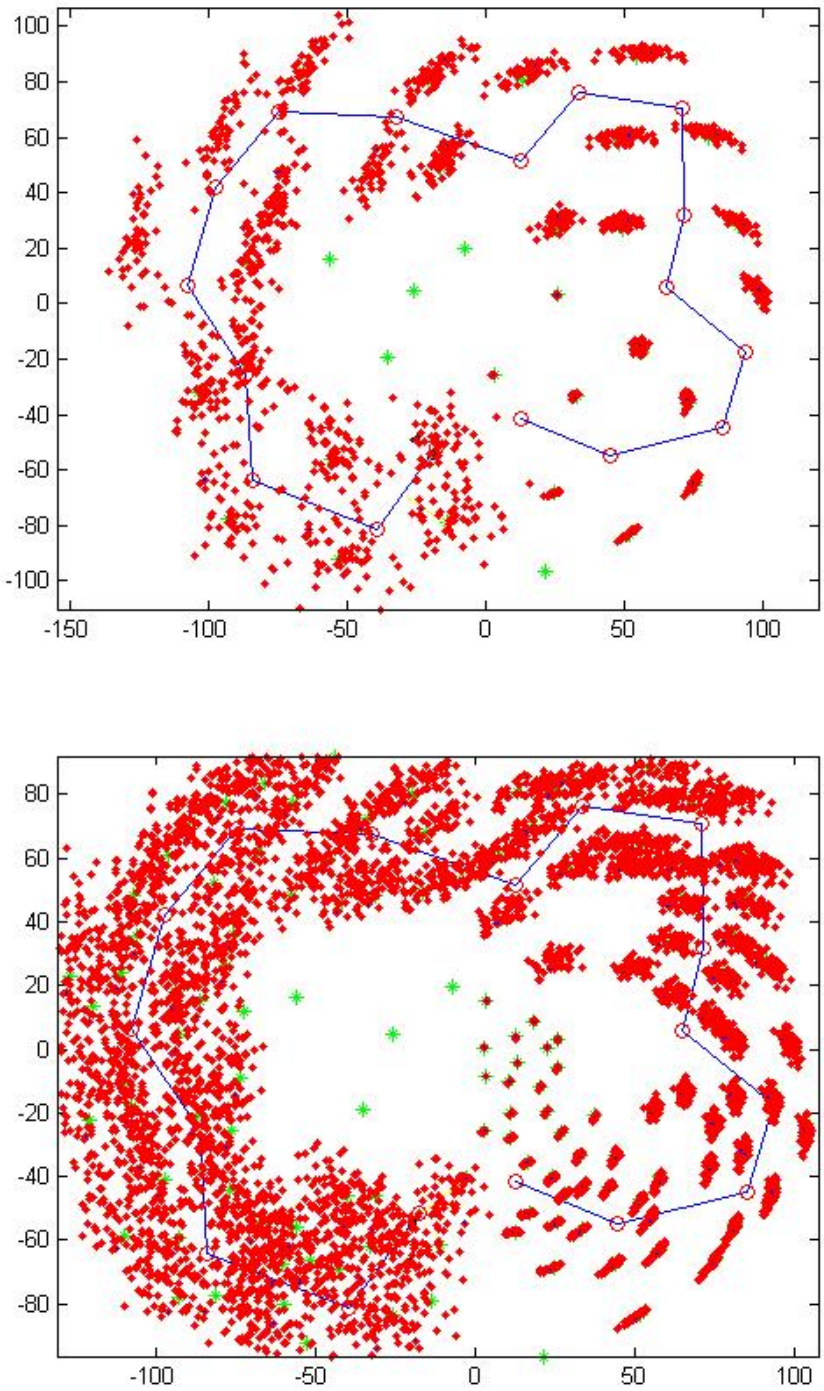


Figura 5.24: Ejemplos de simulaciones en diferentes mapas del algoritmo FastSlam, con los parámetros propios seleccionados y asociación de datos no conocidos. (II)

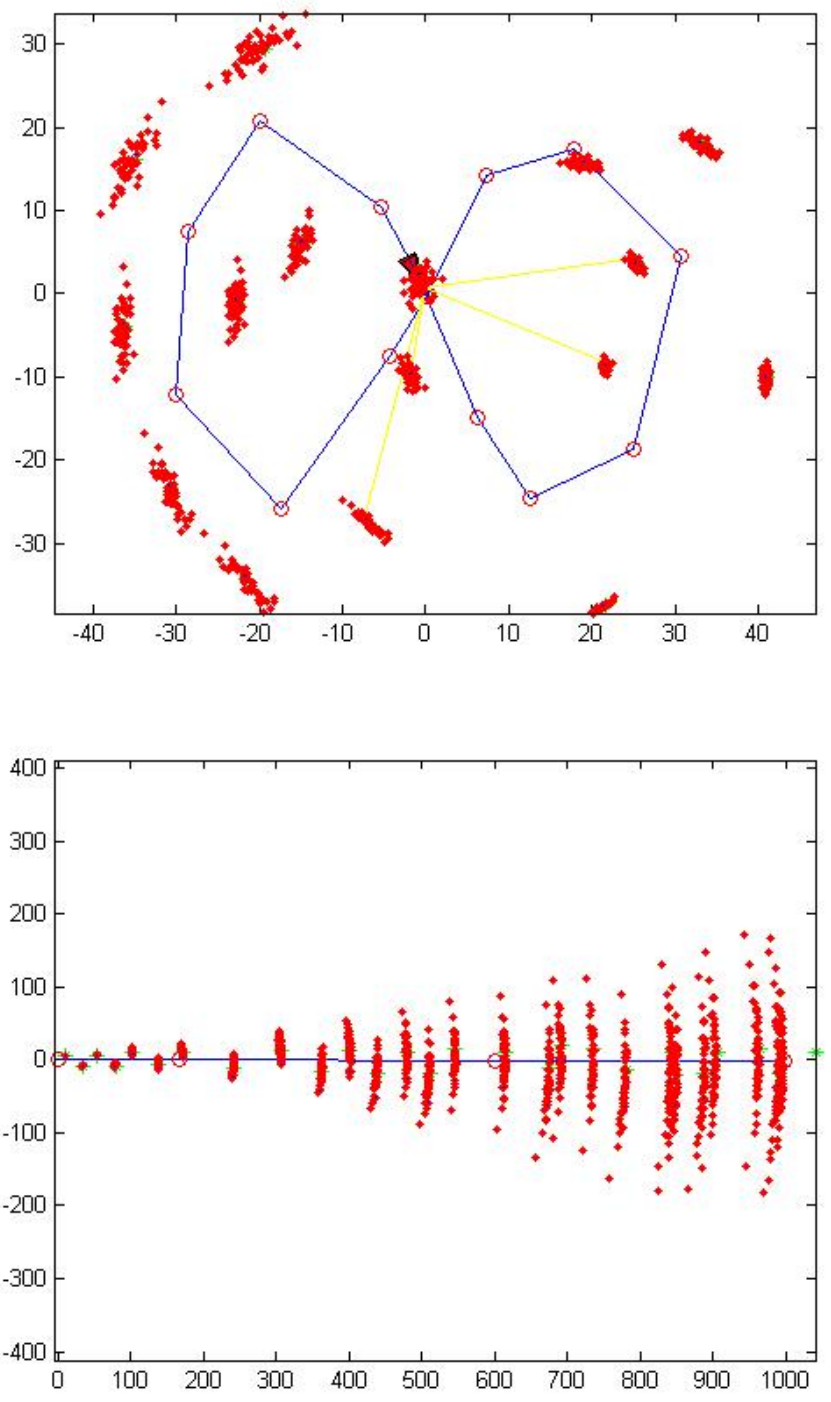


Figura 5.25: Ejemplos de simulaciones en diferentes mapas del algoritmo FastSlam, con los parámetros propios seleccionados y asociación de datos no conocidos. (III)

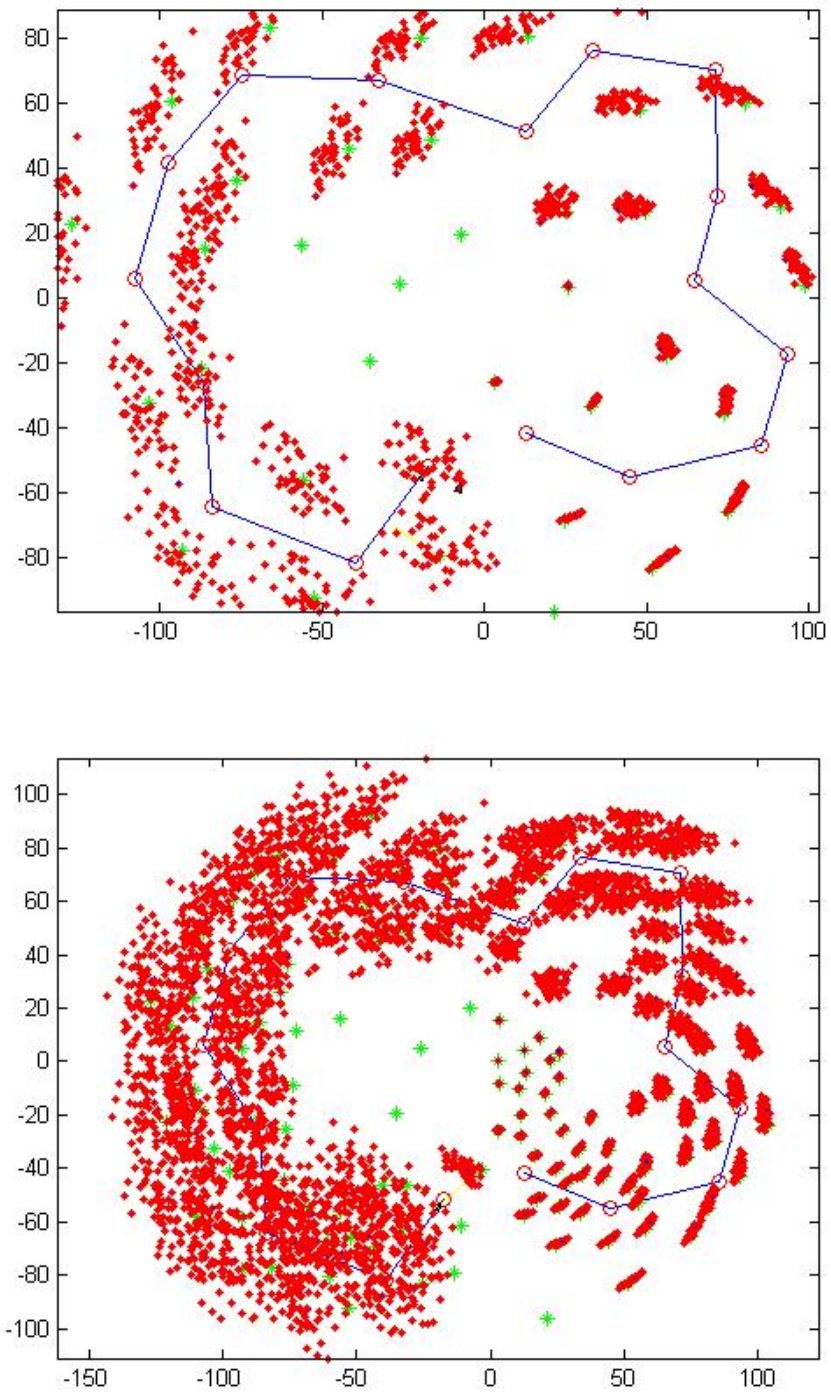


Figura 5.26: Ejemplos de simulaciones en diferentes mapas del algoritmo FastSlam, con los parámetros propios seleccionados y asociación de datos no conocidos. (IV)

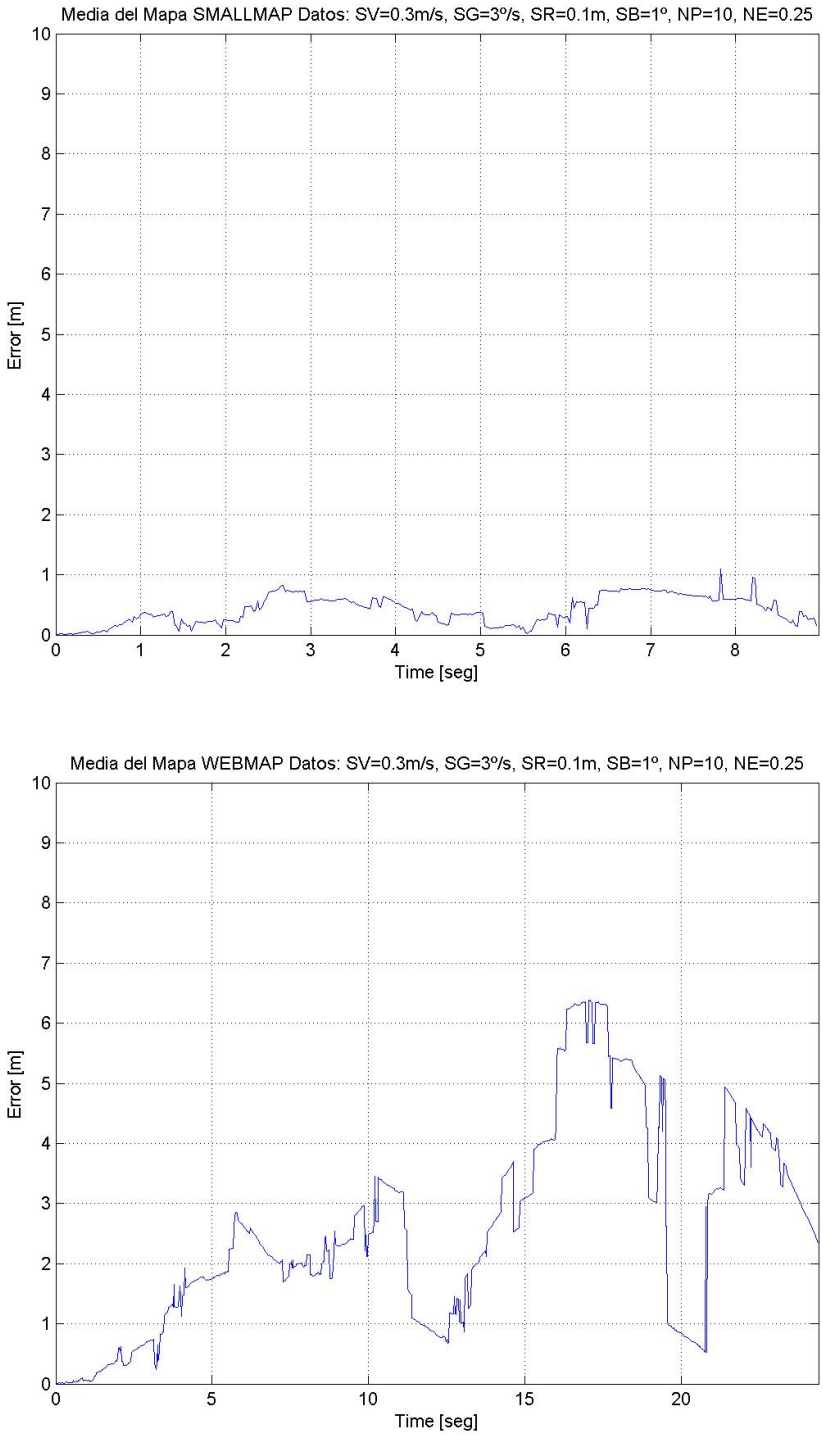


Figura 5.27: Ejemplos de ejecuciones del algoritmo FastSlam con asociación de datos no conocidos, donde los mejores parámetros propios para el primer mapa, perjudican al segundo. (I)

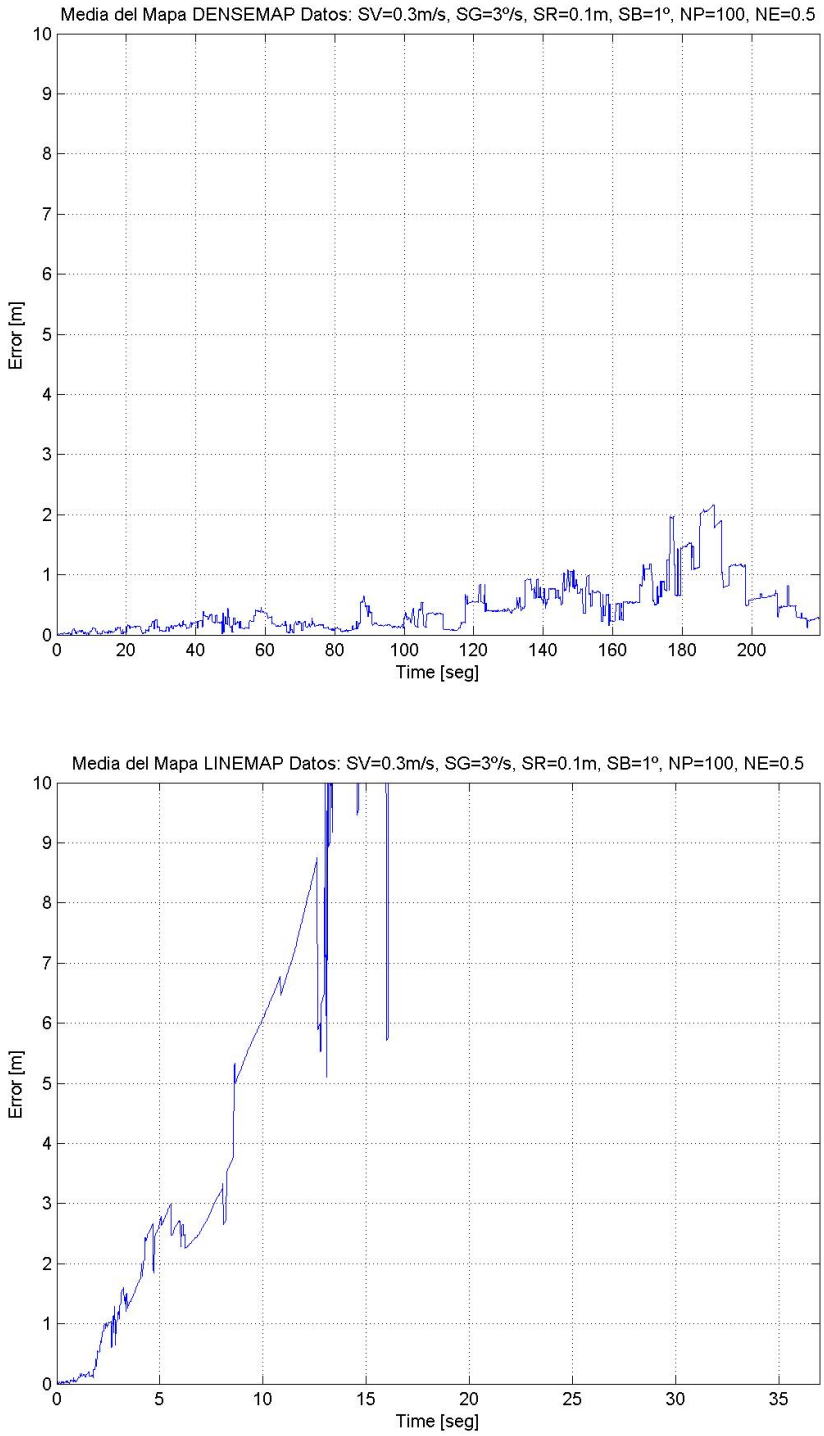


Figura 5.28: Ejemplos de ejecuciones del algoritmo FastSlam con asociación de datos no conocidos, donde los mejores parámetros propios para el primer mapa, perjudican al segundo. (II)

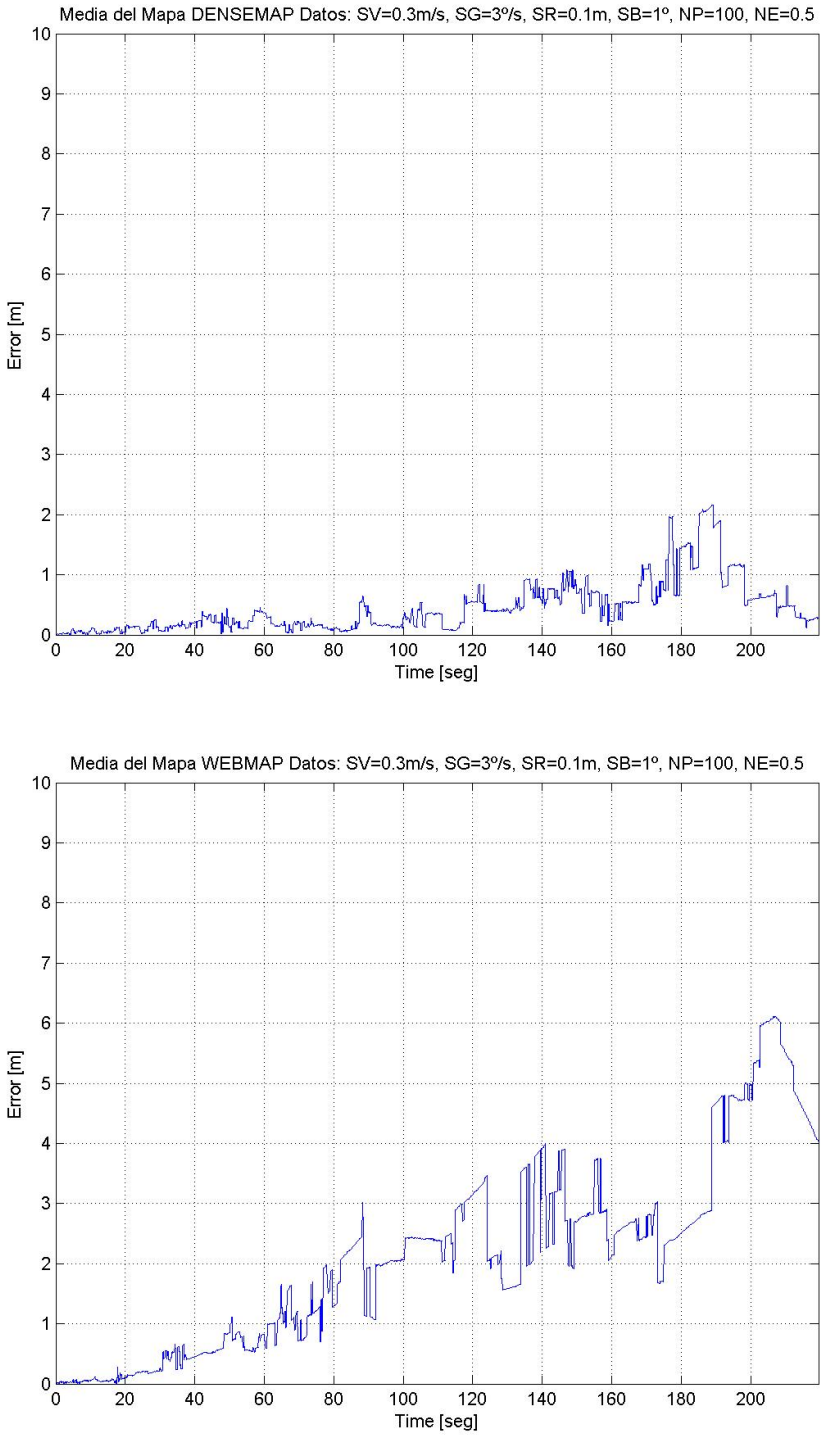


Figura 5.29: Ejemplos de ejecuciones del algoritmo FastSlam con asociación de datos conocidos, donde los mejores parámetros propios para el primer mapa, perjudican al segundo. (III)

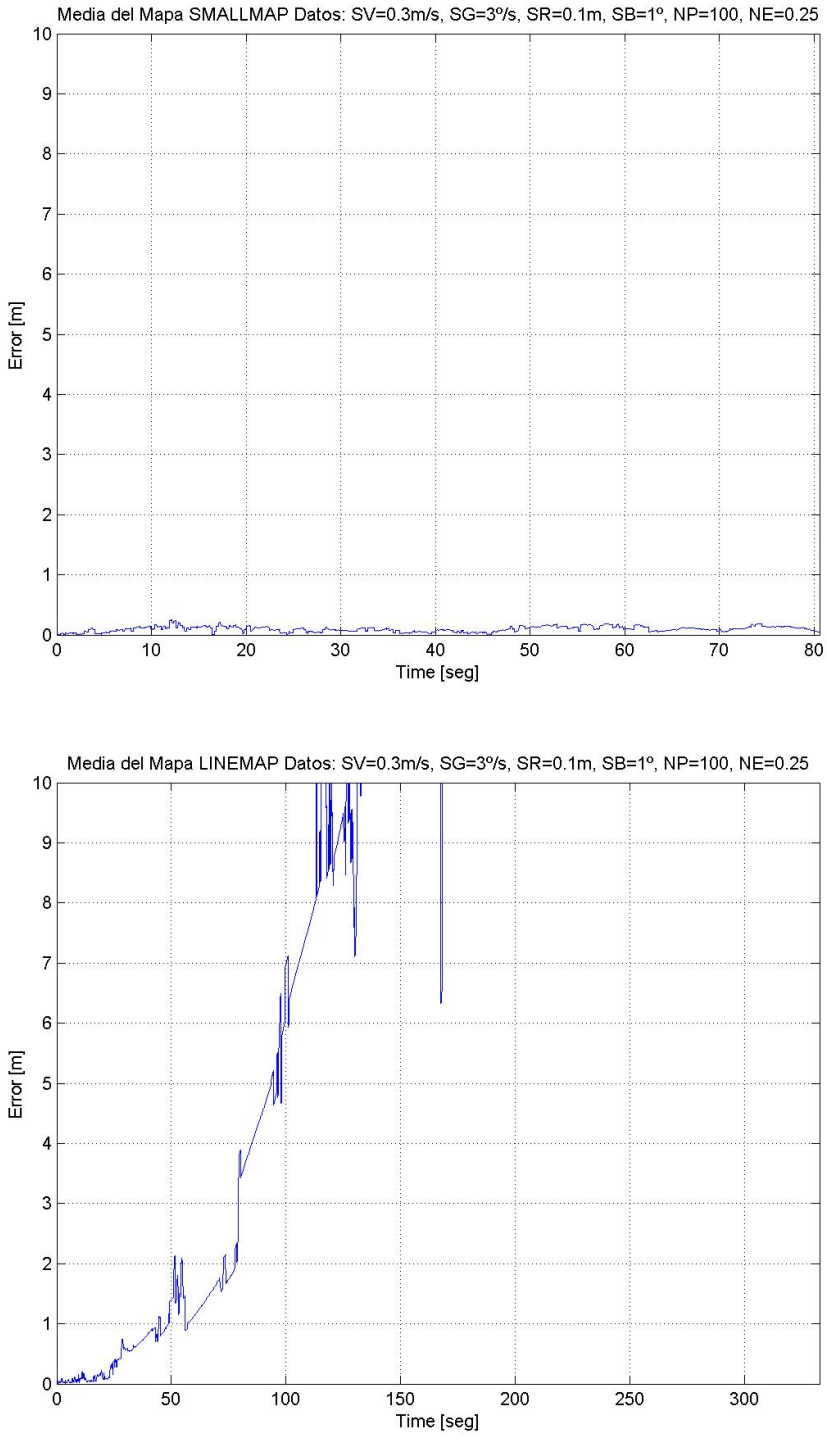


Figura 5.30: Ejemplos de ejecuciones del algoritmo FastSlam con asociación de datos conocidos, donde los mejores parámetros propios para el primer mapa, perjudican al segundo. (IV)

un número mucho mayor (100). Nuestra primera explicación fue pensar en que las modificaciones introducidas en el código estaban provocando ese efecto, pero posteriormente se verificó que también estaba presente usando el código original. Por lo tanto, debe de tratarse de algún tipo de problema de estabilidad numérica que hace que se compense negativamente la mayor capacidad de muestreo que se tiene con un número elevado de partículas.

En FastSlam 2.0 este efecto se acentúa, pero esto ya no es tan extraño al tratarse precisamente de una de las características del algoritmo, esto es, el funcionar adecuadamente con un bajo número de partículas.

5.5.4. Análisis de sensibilidad

Desde el punto de vista de la sensibilidad a los parámetros propios, es difícil obtener una evaluación precisa entre algoritmos. En líneas generales, sin embargo, la impresión que se desprende es que el algoritmo de FastSlam presenta una mayor variabilidad. Esto resulta en parte lógico, puesto que se trata del algoritmo con mayor número de parámetros ajustables, lo que puede traducirse en la aparición de efectos acoplados.

Capítulo 6

Prueba y Evaluación

6.1. Introducción

En primera instancia, se realizarán simulaciones de los algoritmos EKF-Slam, FastSLAM 1.0 y UKF-Slam [Ba06] en diferentes mapas, utilizando los valores de los parámetros propios seleccionados en el capítulo anterior. El propósito de estas simulaciones es el de permitir la comparación de los diversos algoritmos de construcción de mapas en entornos diferentes.

En la sección 6.2 se realiza el estudio de las ejecuciones de los algoritmos en los diferentes mapas, y en la sección 6.3 se estudiará la eficiencia de cada uno de los algoritmos, utilizando para ello la notación asintótica O.

6.2. Ejecución y comparación entre algoritmos

En esta sección se realizarán diversas ejecuciones para cada algoritmo en diferentes mapas, variando aquellos parámetros que determinan el ruido de control y el ruido de observación (parámetros que son comunes a todos los algoritmos), y posteriormente, se realizarán las comparaciones pertinentes para determinar qué algoritmo tiene mejor comportamiento en cada mapa. Dichas ejecuciones persiguen el objetivo de verificar el funcionamiento en diferentes circunstancias.

6.2.1. Parámetros comunes

Los parámetros encargados de generar el ruido de control son dos: "SigmaV" y "SigmaG", que integran la matriz de covarianza del modelo. El primer parámetro es el encargado de añadir el factor de error en la velocidad

del robot (en m/s), y el segundo parámetro, añade el factor de error en los grados de giro del robot (medido en º/s).

Los parámetros encargados de generar el ruido de observación, al igual que los anteriores, son dos: "SigmaR" y "SigmaB", que integran la matriz de covarianza de la medida. El primer parámetro es el encargado de añadir el factor de error en la distancia de observación del robot (en metros), y el segundo parámetro, añade el factor de error en los grados de observación del robot (en grados).

Al igual que se hiciera con la sintonización de los parámetros propios, en la sección 5.5 del capítulo anterior, se utilizará el software matemático MATLAB (MATrix LABoratory) para la realización de las ejecuciones; y los resultados obtenidos durante esta sección, serán el resultado del promedio de diez ejecuciones para cada combinación que se quiera comprobar. De nuevo conviene recordar que los resultados se muestran sólo para el primer ciclo por cuestiones de tiempo de ejecución de las simulaciones.

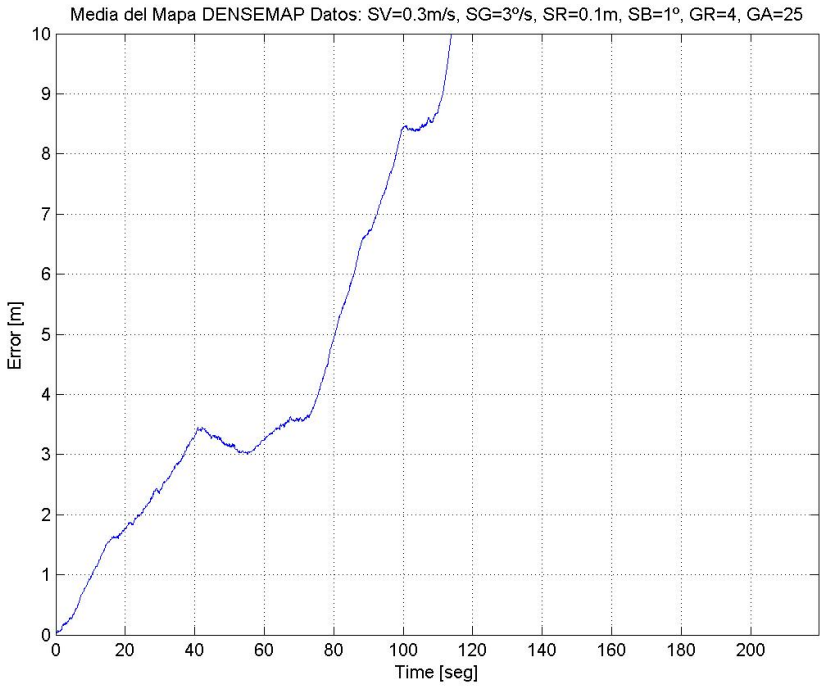
6.2.2. Asociación de datos

Una primera verificación que se realizó fue el análisis del comportamiento de los algoritmos con y sin asociación conocida de datos. Tal y como resulta esperable, el error cometido por los algoritmos es mayor en el caso de la asociación no conocida, cuyo límite superior de rendimiento se sitúa precisamente en el caso de la asociación conocida.

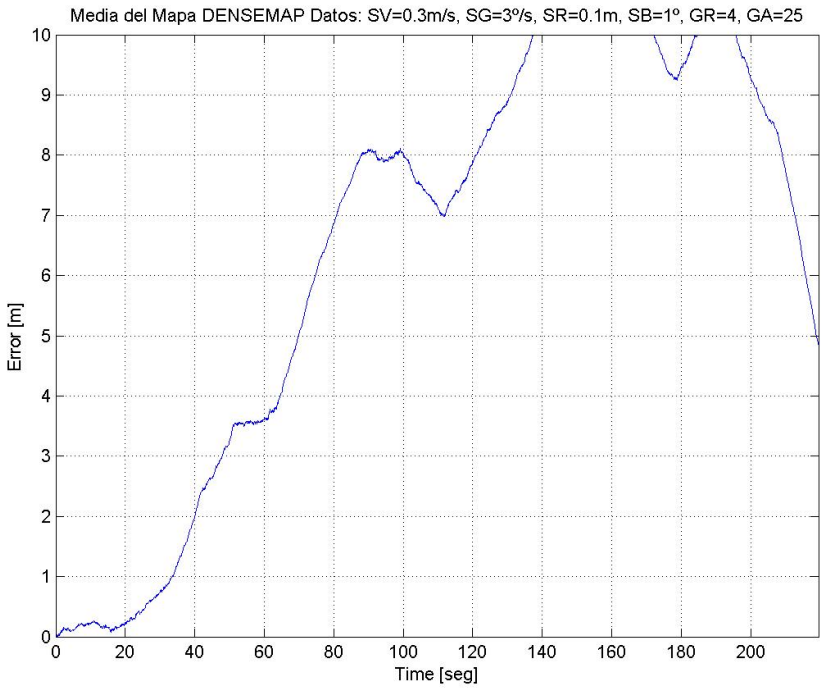
En las figuras 6.1 y 6.2 se pueden observar varios ejemplos de ejecuciones del algoritmo UKF-Slam, en varios mapas, dónde se varía el parámetro "SWITCH_ASSOCIATION_KNOWN", que indica si se utiliza o no la asociación de datos conocidos. De la misma forma, en las figuras 6.3 y 6.4 se pueden observar varios ejemplos de ejecuciones del algoritmo FastSlam, donde se varía nuevamente dicho parámetro.

6.2.3. Variaciones de parámetros

Para el análisis de la sensibilidad de parámetros se realizarán diversas ejecuciones en diferentes mapas, para cada uno de los algoritmos, variando en cada momento el valor de alguno de los parámetros que determinan el error de control y/o el error de observación, y manteniendo invariables el resto de posibles parámetros que se encuentran en el fichero de configuración. Posteriormente se analizarán los resultados obtenidos de esas ejecuciones,

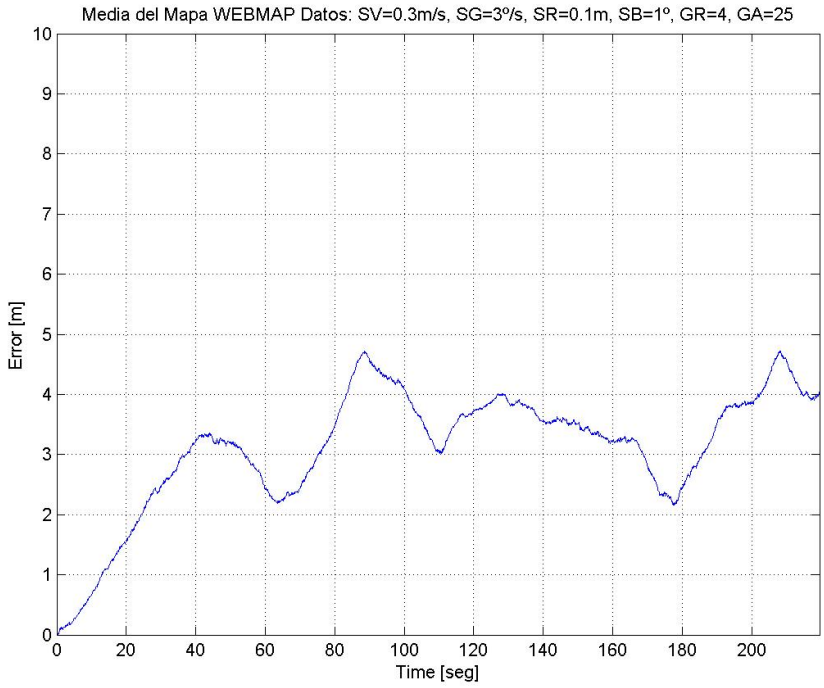


Ejecución con asociación de datos conocidos.

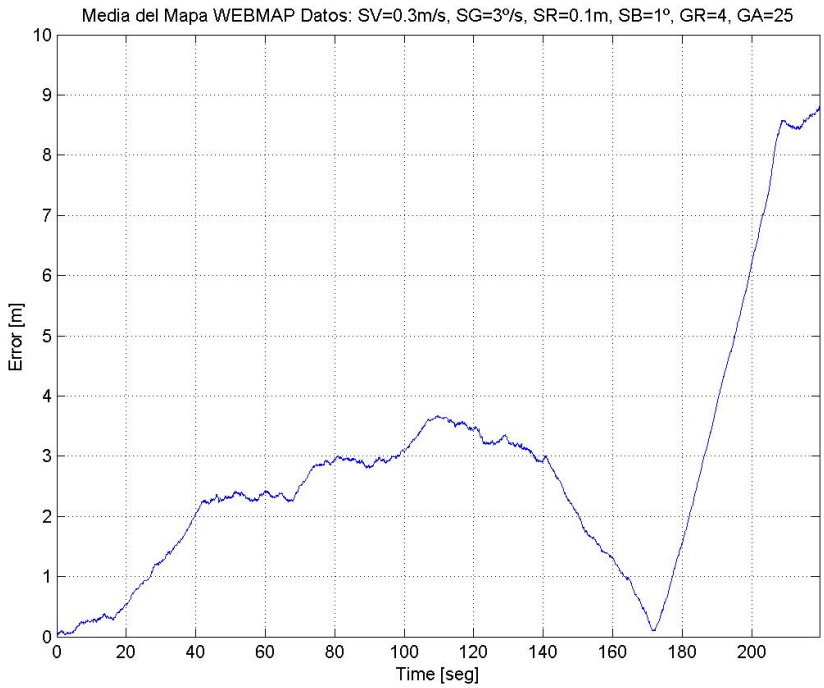


Ejecución sin asociación de datos conocidos.

Figura 6.1: Ejemplos de ejecuciones del algoritmo UKF-Slam, con el mapa "DENSEMAP", utilizando el parámetro de asociación de datos conocidos.

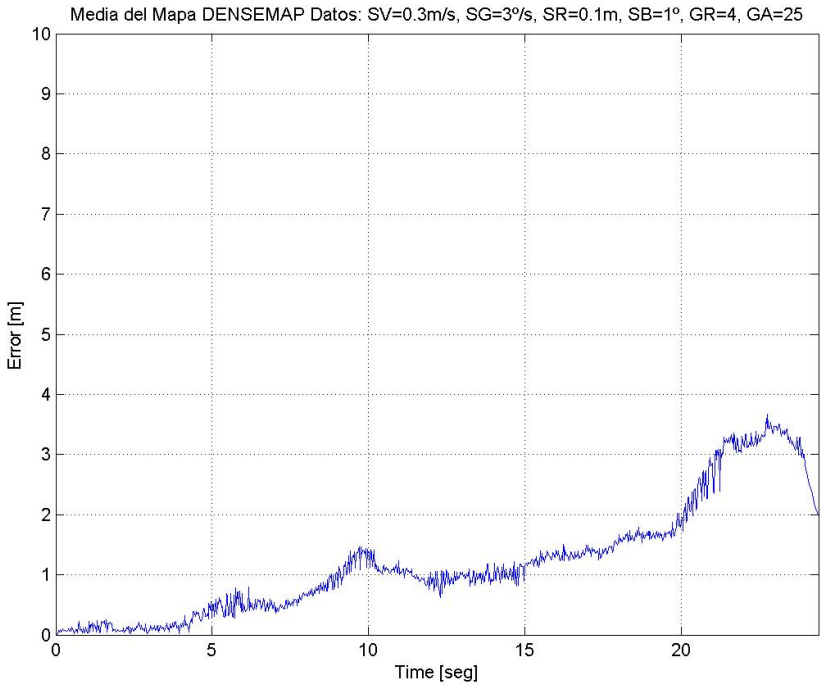


Ejecución con asociación de datos conocidos.

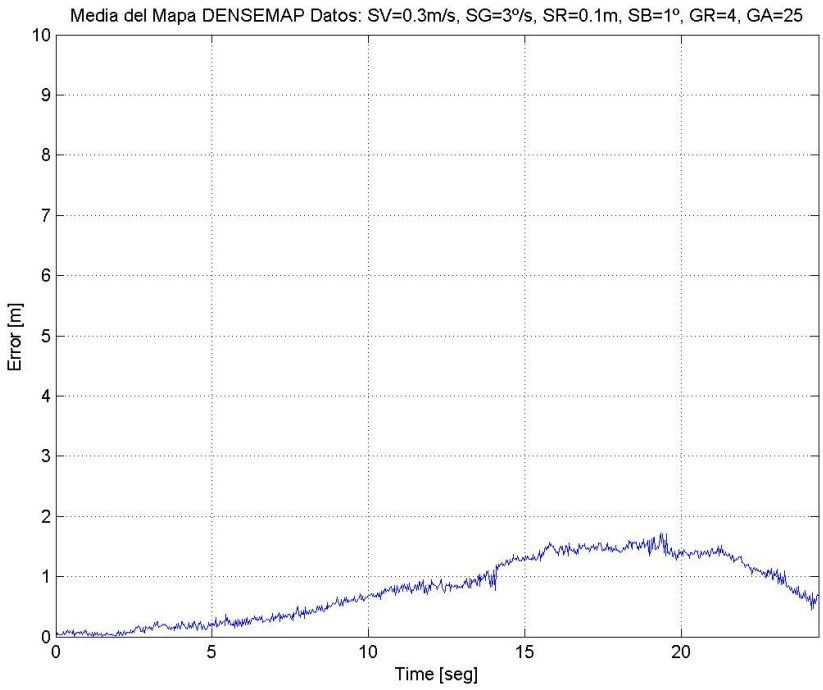


Ejecución sin asociación de datos conocidos.

Figura 6.2: Ejemplos de ejecuciones del algoritmo UKF-Slam, con el mapa "WEBMAP", utilizando el parámetro de asociación de datos conocidos.

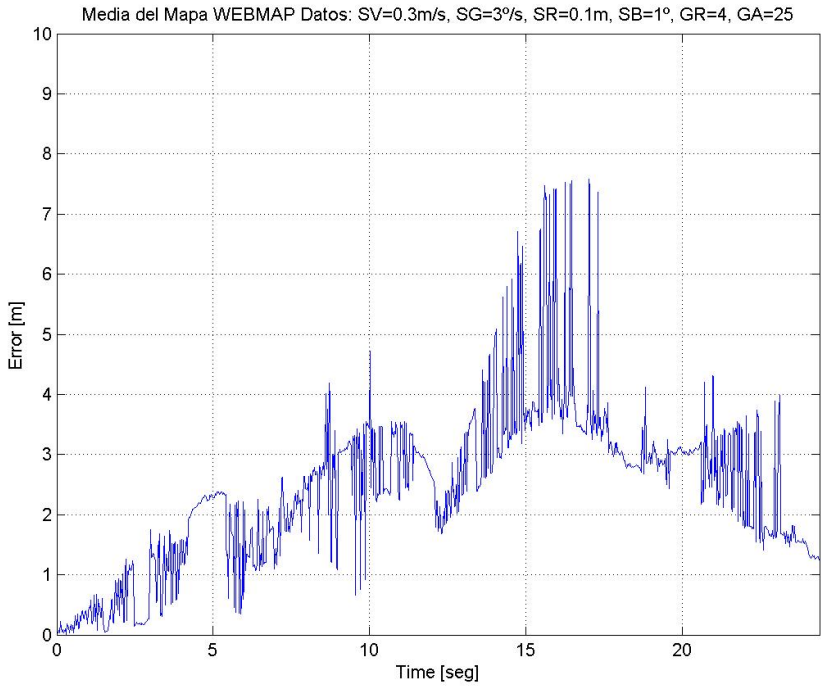


Ejecución con asociación de datos conocidos.

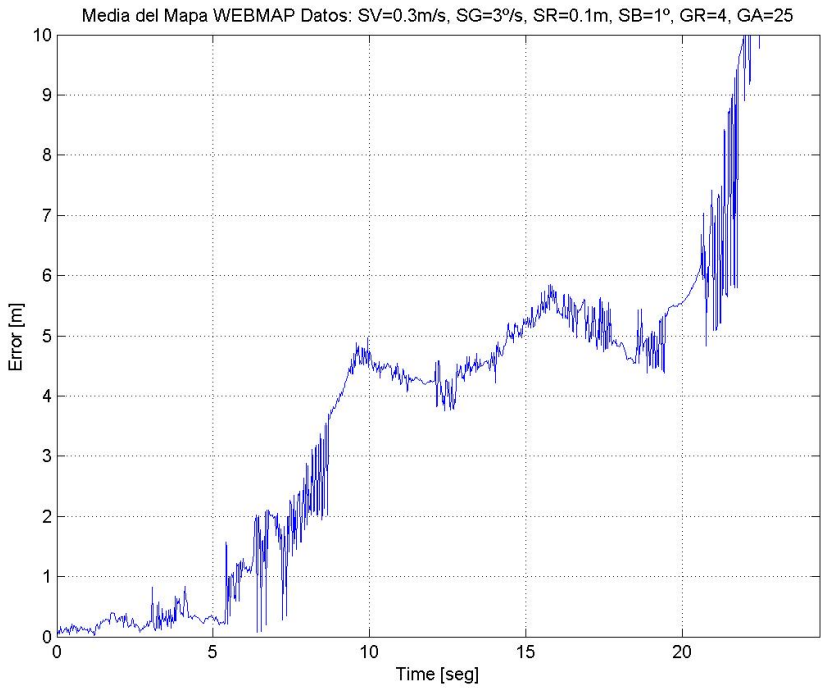


Ejecución sin asociación de datos conocidos.

Figura 6.3: Ejemplos de ejecuciones del algoritmo FastSlam, con el mapa "DENSEMAP", utilizando el parámetro de asociación de datos conocidos.



Ejecución con asociación de datos conocidos.



Ejecución sin asociación de datos conocidos.

Figura 6.4: Ejemplos de ejecuciones del algoritmo FastSlam, con el mapa "WEBMAP", utilizando el parámetro de asociación de datos conocidos.

para determinar los valores de los parámetros que mejor comportamiento global hayan generado en los algoritmos.

Los valores utilizados para cada uno de los parámetros que conforman el error de control, han sido seleccionados de la siguiente forma: Para el parámetro "SigmaV" se han utilizado valores obtenidos tras operar con los parámetros "V" y "DT_OBSERVE" del fichero de configuración. El parámetro "V" determina la velocidad máxima del robot (en m/s), y el parámetro "DT_OBSERVE" corresponde al intervalo de tiempo entre observaciones (en segundos). Como dichos parámetros poseen valores fijos en el fichero de configuración, el valor base para los posibles parámetros se obtendrá de la forma *ValorBaseSV* = V * DT_OBSERVE = 3 * 0.2 = 0.6 m/s. Obtenido el valor base de "SigmaV", los valores que tomará el parámetro en las siguientes ejecuciones será *ValorBaseSV*/2, *ValorBaseSV* y 2**ValorBaseSV*.

Haciendo mención ahora al valor del parámetro "SigmaG", en este caso se mantendrá el factor multiplicativo inicial de diez, asignado en el fichero de configuración, con respecto a "SigmaV". Los valores que tomará el parámetro "SigmaG" en las siguientes ejecuciones será (*ValorBaseSV*/2)*10, *ValorBaseSV**10 y (2**ValorBaseSV*)*10.

Como resumen, los valores que utilizarán los parámetros "SigmaV" y "SigmaG" serán los siguientes:

- SigmaV: 0.3, 0.6, 1.2.
- SigmaG: 3.0, 6.0, 12.0.

Los valores que se utilizarán para los parámetros que conforman el error de observación, estarán basados en valores obtenidos tras realizar unos cálculos con el conjunto de hitos (lm) y con el conjunto de puntos de referencia (wp), descritos en el apartado 5.2 del capítulo anterior. La elección del valor base para el parámetro "SigmaR" será la media de las distancias euclídeas mínimas de cada hito con respecto a los demás hitos del conjunto. Expresado de otra forma, para cada hito, se calcularán las distancias euclídeas de éste con respecto a los demás, y de esas distancias obtenidas, se elegirá la distancia mínima. Una vez se han obtenido las distancias euclídeas mínimas para cada hito, se calcula la media de esas distancias. El resultado que se utilizará como valor base para dar valores al parámetro "SigmaR" será el redondeo de la media indicada anteriormente. Obtenido el valor base de "SigmaR", los valores que tomará el parámetro en las siguientes ejecuciones, al igual que se hizo con el parámetro "SigmaV", será *ValorBaseSR*/2, *ValorBaseSR* y 2**ValorBaseSR*.

Para la elección del valor base del parámetro "SigmaB", se utilizará la separación angular media entre balizas desde cada punto de referencia (Ángulo promedio 1 (AP1)), descrito en el apartado 5.2.1.

Esa media global calculada en el último paso, será el valor base que se utilizará para dar valor al parámetro "SigmaB". Los valores que tomará dicho parámetro en las siguientes ejecuciones, al igual que se hizo con los demás parámetros, será *ValorBaseSB*/2, *ValorBaseSB* y 2**ValorBaseSB*.

Hay que tener en cuenta que los valores de "SigmaR" y "SigmaB" serán diferentes, dependiendo del mapa que se utilice para cada ejecución, debido a las características que presenta cada uno de ellos con respecto a puntos de referencia y conjunto de hitos. Para los mapas utilizados en las ejecuciones, tenemos que los valores selecionados para dichos parámetros serán:

- Mapa "LINEMAP".
  - SigmaR: 21.0 42.0 84.0.
  - SigmaB: 0.313 0.626 1.252.
- Mapa "SMALLMAP".
  - SigmaR: 8.0 16.0 32.0.
  - SigmaB: 0.516 1.032 2.064.
- Mapa "WEBMAP".
  - SigmaR: 14.5 29.0 58.5.
  - SigmaB: 1.073 2.146 4.292.
- Mapa "DENSEMAP".
  - SigmaR: 5.5 11.0 22.0.
  - SigmaB: 0.313 0.626 1.252.

Nota: La separación angular media entre balizas desde cada punto de referencia (SigmaB) del mapa "Linemap" es igual a cero, debido a que el mapa es una recta, y no existe ángulo de giro. Debido a ello, y para poder introducir error con dicho parámetro, se le asigna el valor menor obtenido para "SigmaB", que corresponde al que se obtiene en el mapa "Densemap".

El resultado final que se obtiene para cada combinación de parámetros es el resultado del promedio de diez ejecuciones, para cada una de las combinaciones, y para cada uno de los mapas definidos en la sección 5.2.

6.2.3.1. Ejecuciones del algoritmo EKF-Slam

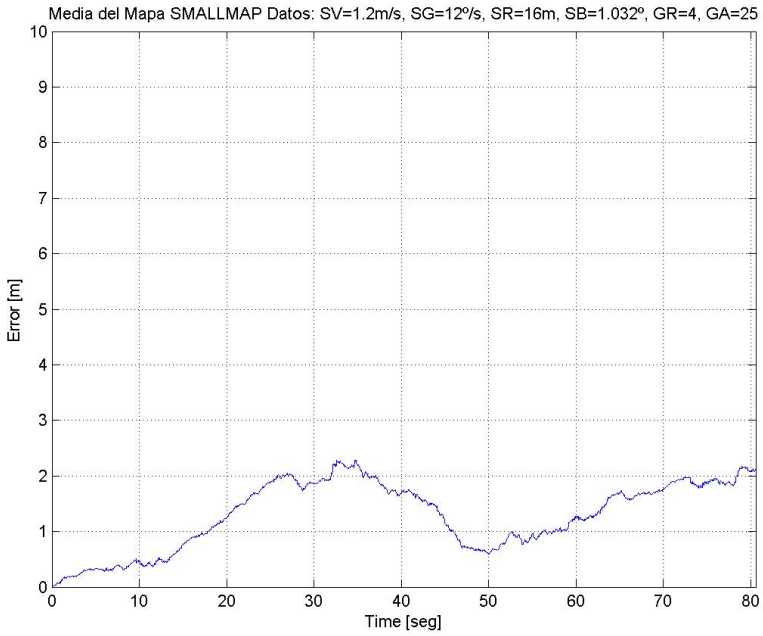
Para llevar a cabo es estudio de la sensibilidad de parámetros para dicho algoritmo, se ha realizado una serie de ejecuciones, variando en cada momento el valor de alguno de dichos parámetros, y manteniendo invariables el resto de posibles parámetros que se encuentran en el fichero de configuración.

El resultado final que se obtiene para cada combinación de parámetros es el resultado del promedio de diez ejecuciones, para cada una de las combinaciones, y para cada uno de los mapas definidos en la sección 5.2.

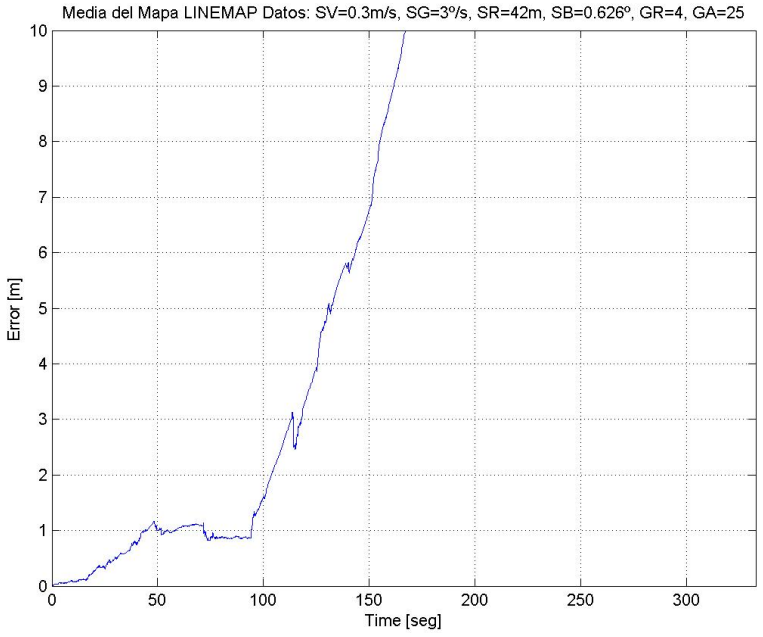
Los resultados finales obtenidos se visualizan en unas gráficas, como las representadas en las figuras 6.5 y 6.6 si hacemos uso de la asociación de datos no conocidos, o en las figuras 6.7 y 6.8 si hacemos uso de la asociación de datos conocidos. Cada una de ellas representan en el eje de abcisas el tiempo transcurrido durante la ejecución del algoritmo en el mapa (en segundos), y en el eje de ordenadas representan el error medio cometido por el algoritmo (en metros). El error se calcula como la diferencia en valor absoluto entre las posiciones reales del robot en cada instante de tiempo y el estado estimado por cada algoritmo.

Una vez realizadas todas y cada una de las ejecuciones, con el propósito de analizar la sensibilidad de los parámetros para el algoritmo EKF-Slam, se eligen los parámetros que mejor comportamiento global presentan para el conjunto de todos los mapas.

Ejecutadas las diferentes combinaciones de parámetros a analizar para el algoritmo EKF-Slam, se obtiene como resultado que la mejor combinación global es la formada por SigmaV = 0.3, SigmaG = 3.0, SigmaR = $ValorBaseSR/2$ y SigmaB = $ValorBaseSB/2$. En las figuras 6.9 y 6.10 se representan los resultados del promedio de las ejecuciones del algoritmo EKF-Slam para cada mapa, con los valores de los parámetros indicados anteriormente si hacemos uso de la asociación de datos no conocidos, o en las figuras 6.11 y 6.12, si hacemos uso de la asociación de datos conocidos.

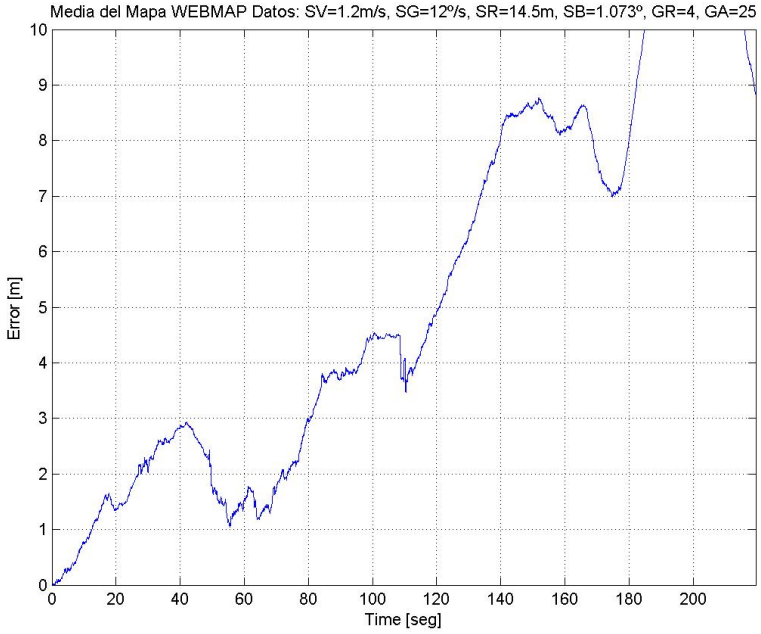


a) Mapa "SMALLMAP" con SigmaV = 1.2, SigmaG = 12.0, SigmaR = 16.0 y SigmaB = 1.032

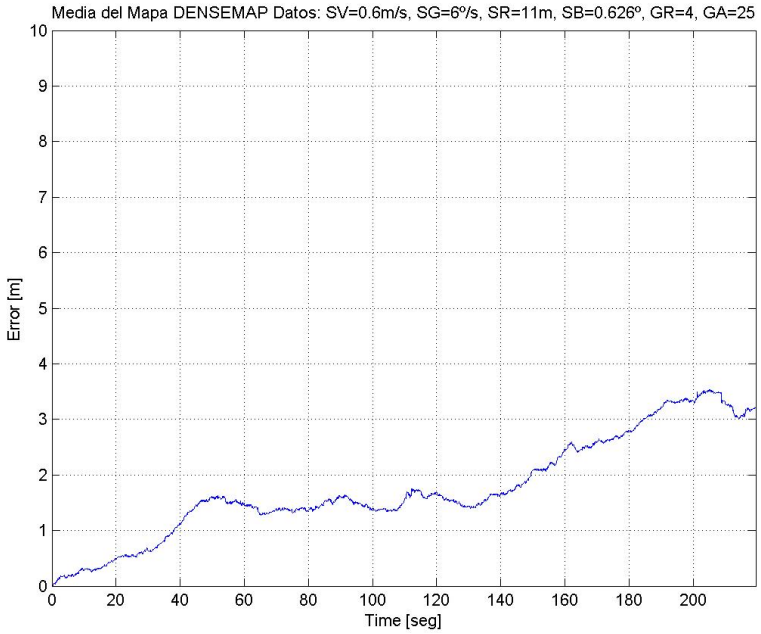


b) Mapa "LINEMAP" con SigmaV = 0.3, SigmaG = 3.0, SigmaR = 42.0 y SigmaB = 0.626

Figura 6.5: Ejemplo de resultados del promedio de las ejecuciones del algoritmo EKF-Slam con el mapa "SMALLMAP" y "LINEMAP", ambos con asociación de datos no conocidos.

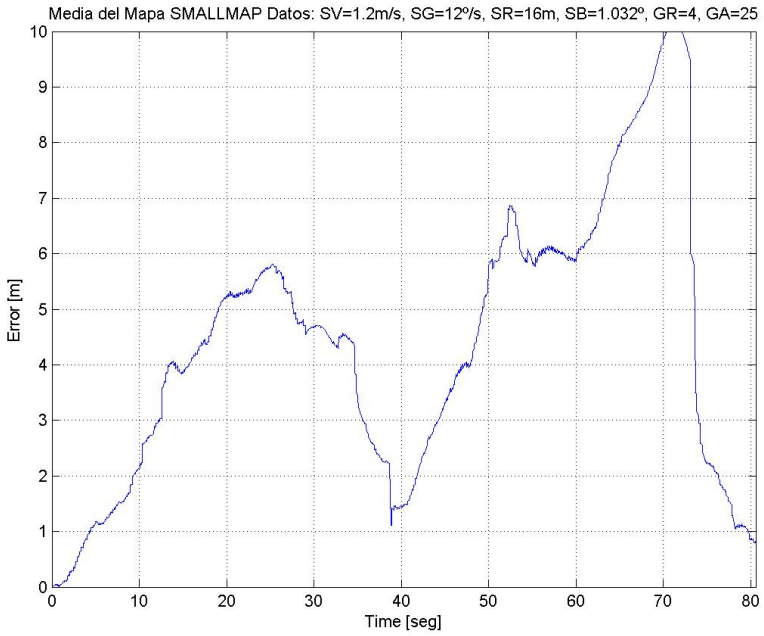


a) Mapa "WEBMAP" con SigmaV = 1.2, SigmaG = 12.0, SigmaR = 14.5 y SigmaB = 1.073

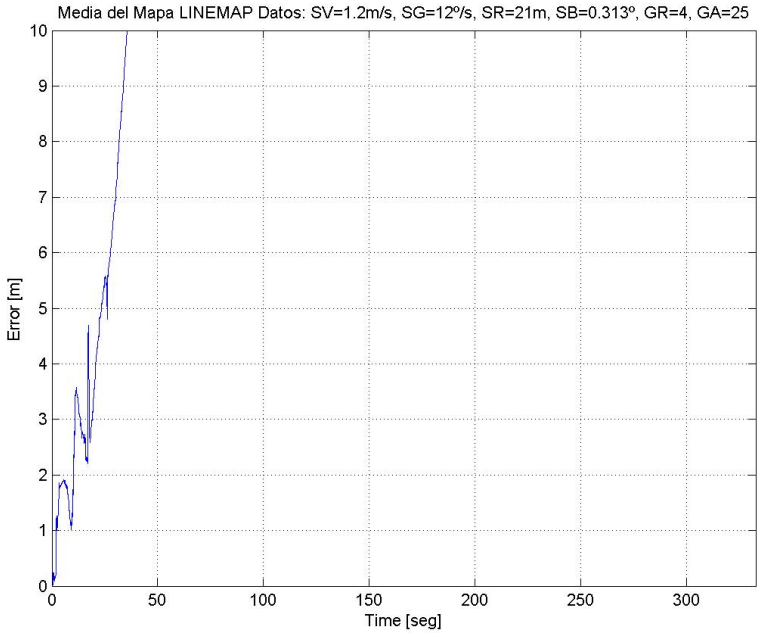


b) Mapa "DENSEMAP" con SigmaV = 0.6, SigmaG = 6.0, SigmaR = 5.5 y SigmaB = 0.313

Figura 6.6: Ejemplo de resultados del promedio de las ejecuciones del algoritmo EKF-Slam con el mapa "WEBMAP" y "DENSEMAP", ambos con asociación de datos no conocidos.

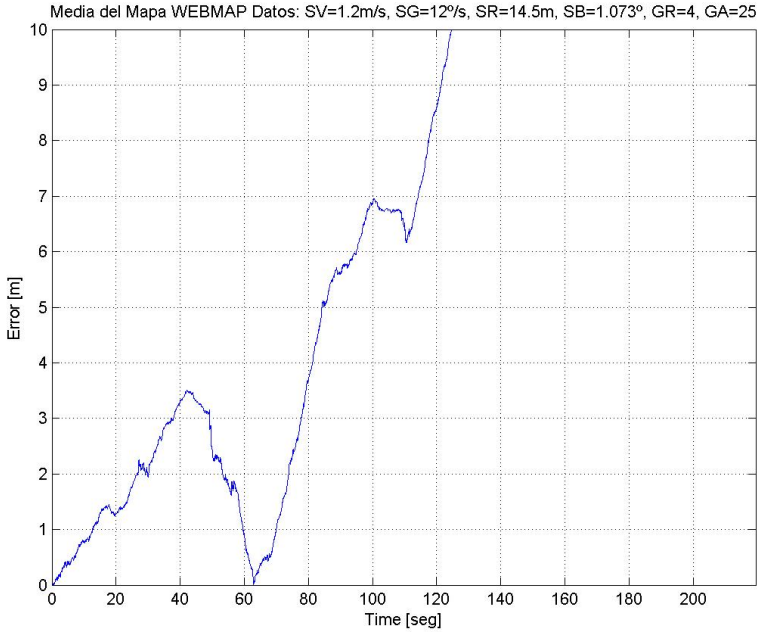


a) Mapa "SMALLMAP" con SigmaV = 1.2, SigmaG = 12.0, SigmaR = 16.0 y SigmaB = 1.032

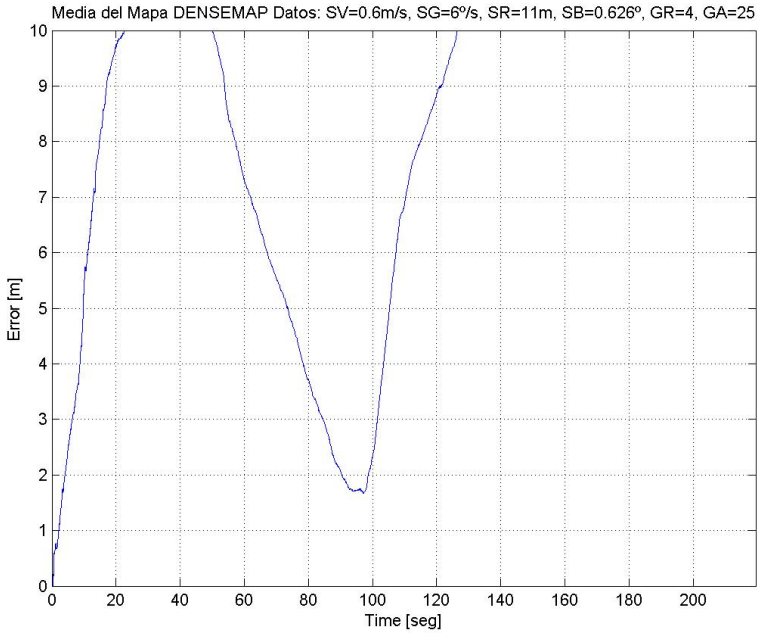


b) Mapa "LINEMAP" con SigmaV = 1.2, SigmaG = 12.0, SigmaR = 21.0 y SigmaB = 0.313

Figura 6.7: Ejemplo de resultados del promedio de las ejecuciones del algoritmo EKF-Slam con el mapa "SMALLMAP" y "LINEMAP", ambos con asociación de datos conocidos.

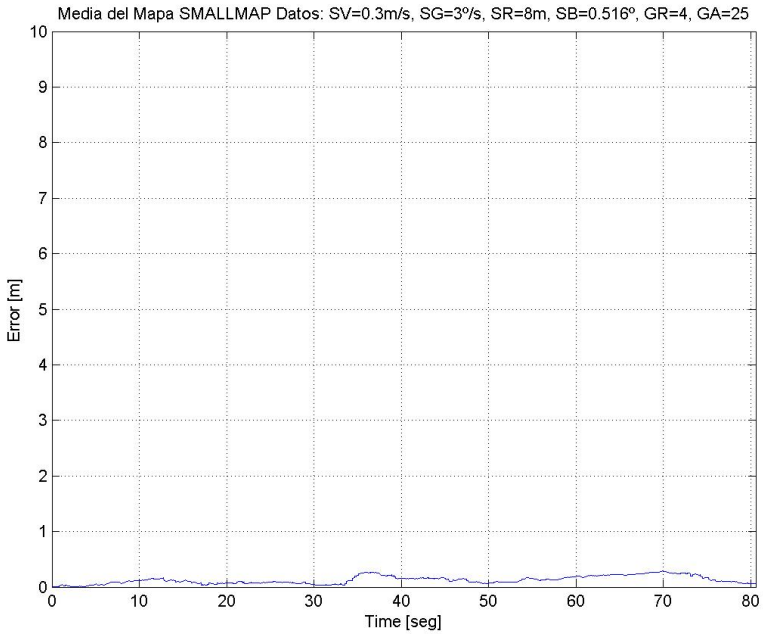


a) Mapa "WEBMAP" con SigmaV = 1.2, SigmaG = 12.0, SigmaR = 14.5 y SigmaB = 1.073

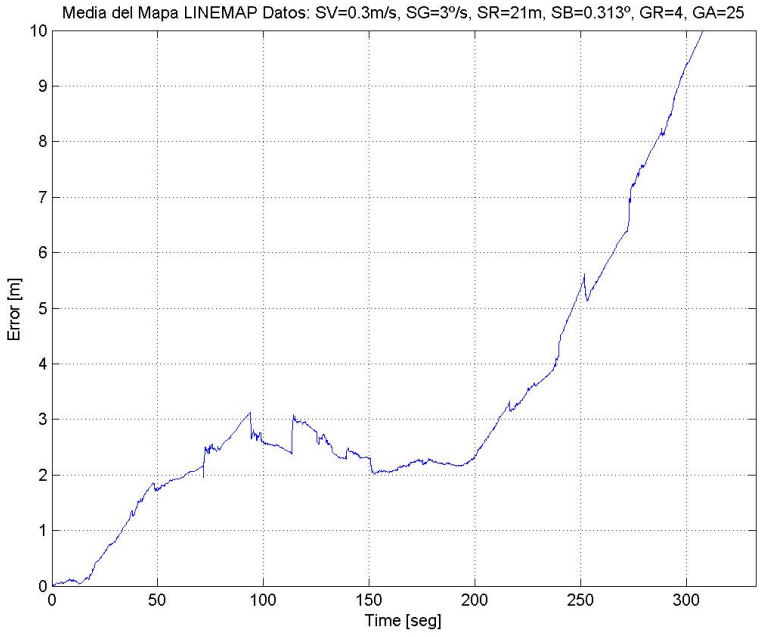


b) Mapa "DENSEMAP" con SigmaV = 0.6, SigmaG = 6.0, SigmaR = 5.5 y SigmaB = 0.313

Figura 6.8: Ejemplo de resultados del promedio de las ejecuciones del algoritmo EKF-Slam con el mapa "WEBMAP" y "DENSEMAP", ambos con asociación de datos conocidos.

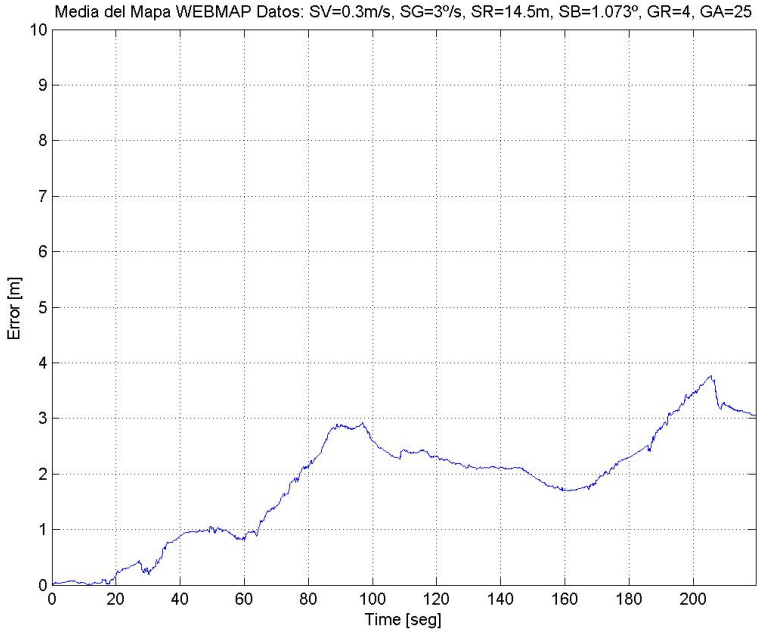


a) Mapa "SMALLMAP" con SigmaV = 0.3, SigmaG = 3.0, SigmaR = 8.0 y SigmaB = 0.516

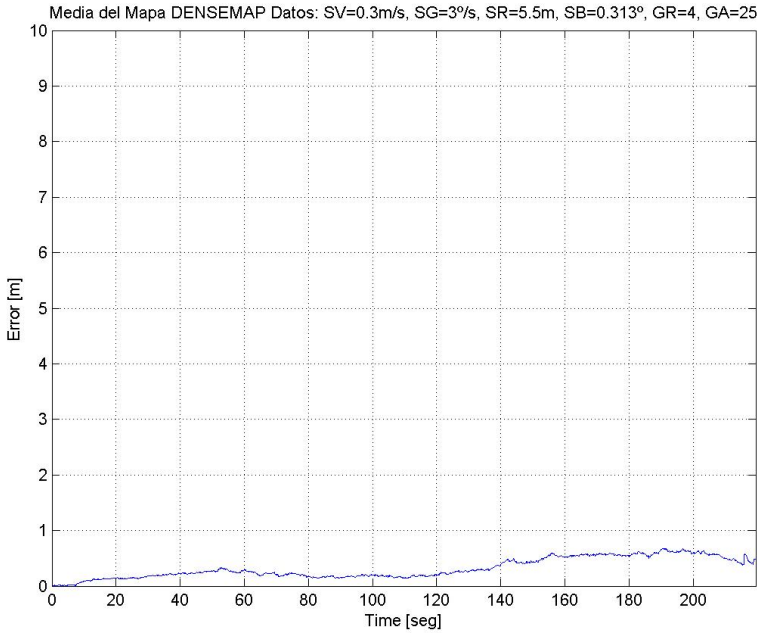


b) Mapa "LINEMAP" con SigmaV = 0.3, SigmaG = 3.0, SigmaR = 21.0 y SigmaB = 0.313

Figura 6.9: Resultados del promedio de las ejecuciones del algoritmo EKF-Slam con el mapa "SMALLMAP" y "LINEMAP", ambos con asociación de datos no conocidos.

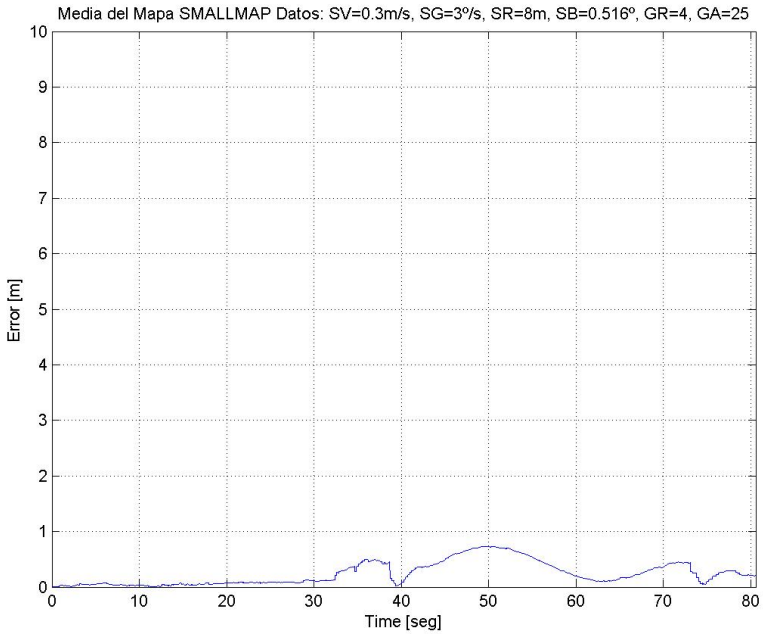


a) Mapa "WEBMAP" con SigmaV = 0.3, SigmaG = 3.0, SigmaR = 14.5 y SigmaB = 1.073

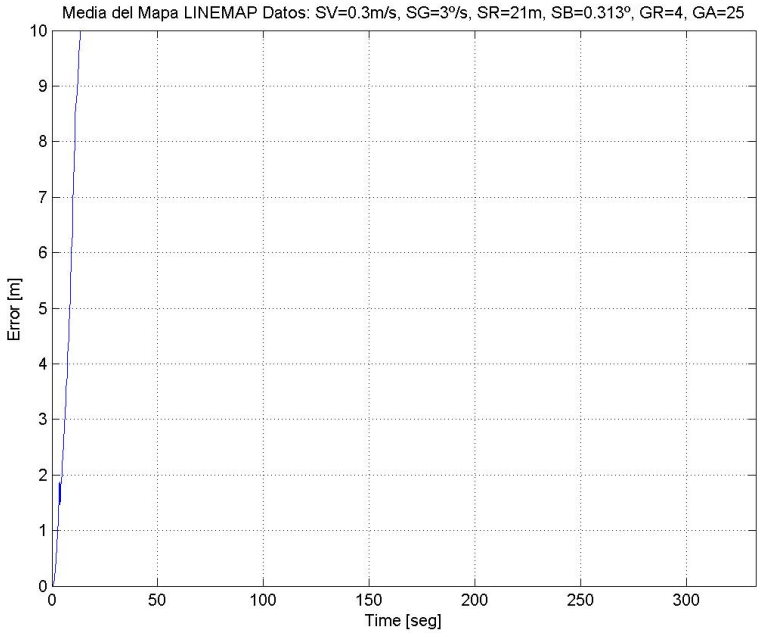


b) Mapa "DENSEMAP" con SigmaV = 0.3, SigmaG = 3.0, SigmaR = 5.5 y SigmaB = 0.313

Figura 6.10: Resultados del promedio de las ejecuciones del algoritmo EKF-Slam con el mapa "WEBMAP" y "DENSEMAP", ambos con asociación de datos no conocidos.

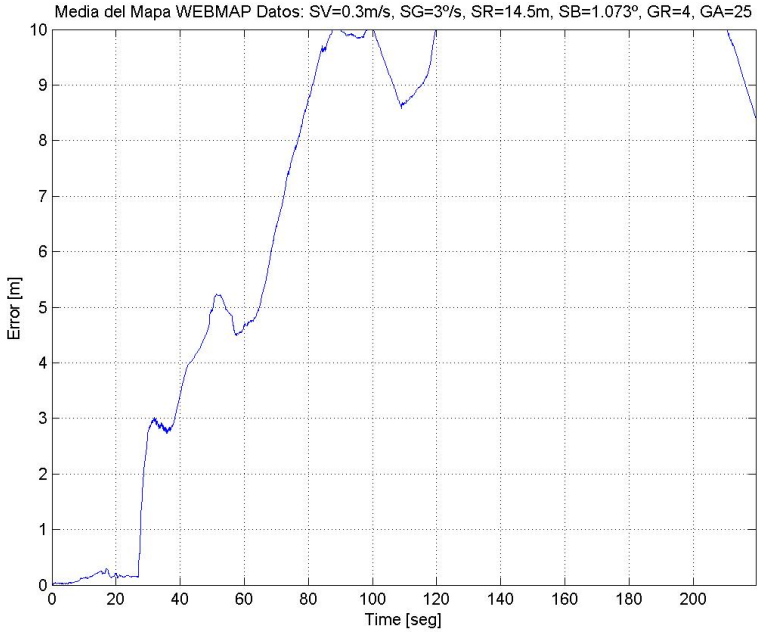


a) Mapa "SMALLMAP" con SigmaV = 0.3, SigmaG = 3.0, SigmaR = 8.0 y SigmaB = 0.516

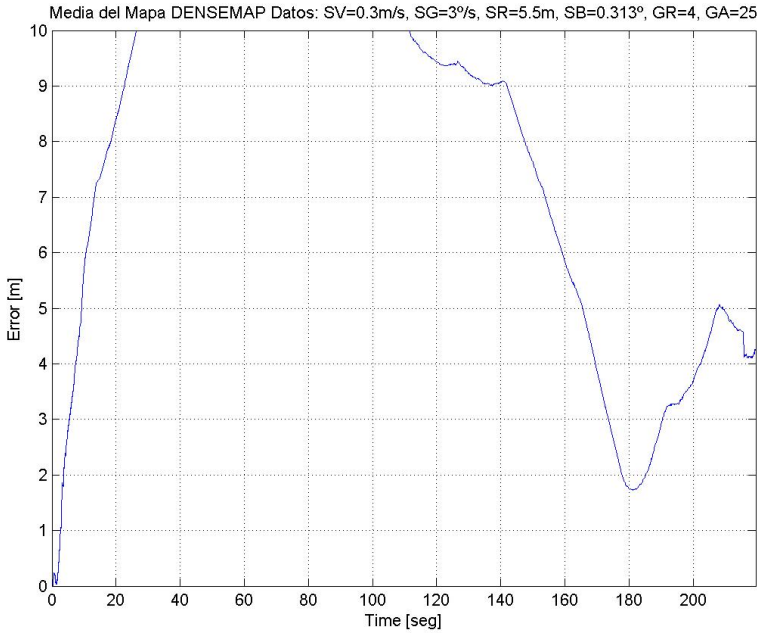


b) Mapa "LINEMAP" con SigmaV = 0.3, SigmaG = 3.0, SigmaR = 21.0 y SigmaB = 0.313

Figura 6.11: Resultados del promedio de las ejecuciones del algoritmo EKF-Slam con el mapa "SMALLMAP" y "LINEMAP", ambos con asociación de datos conocidos.



a) Mapa "WEBMAP" con SigmaV = 0.3, SigmaG = 3.0, SigmaR = 14.5 y SigmaB = 1.073



b) Mapa "DENSEMAP" con SigmaV = 0.3, SigmaG = 3.0, SigmaR = 5.5 y SigmaB = 0.313

Figura 6.12: Resultados del promedio de las ejecuciones del algoritmo EKF-Slam con el mapa "WEBMAP" y "DENSEMAP", ambos con asociación de datos conocidos.

6.2.3.2. Ejecuciones del algoritmo FastSlam

Al igual que realizó en el apartado anterior con EKF-Slam, se ha realizado una serie de ejecuciones variando en cada momento el valor de alguno de los parámetros, con el propósito de estudiar la sensibilidad de parámetros para dicho algoritmo, manteniendo invariables el resto de posibles parámetros que se encuentran en el fichero de configuración.

El resultado final que se obtiene para cada combinación de parámetros es el resultado del promedio de diez ejecuciones, para cada una de las combinaciones, y para cada uno de los mapas definidos en la sección 5.2.

Los resultados finales obtenidos se visualizan en unas gráficas, como las representadas en las figuras 6.13 y 6.14 si hacemos uso de la asociación de datos no conocidos, o en las figuras 6.15 y 6.16 si hacemos uso de la asociación de datos conocidos. En el eje de abcisas representan el tiempo transcurrido durante la ejecución del algoritmo en el mapa (en segundos), y en el eje de ordenadas representan el error medio cometido por el algoritmo (en metros).

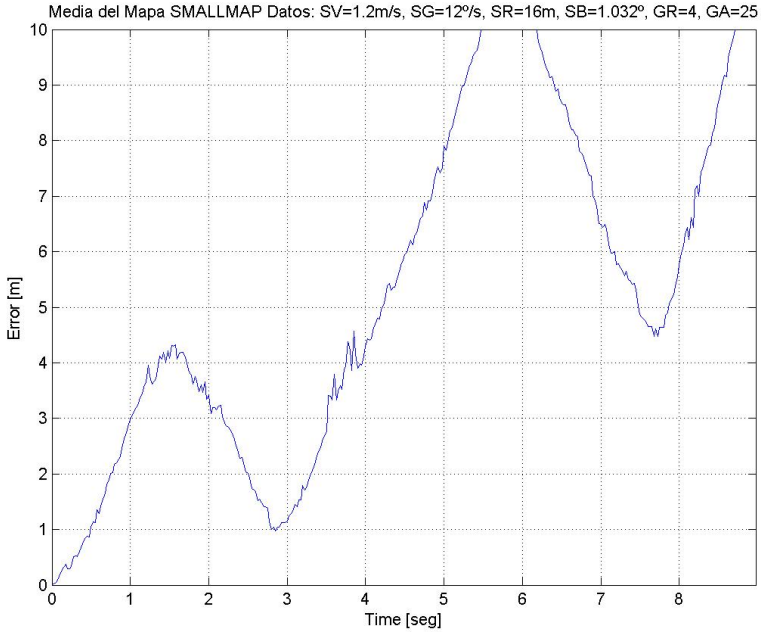
Una vez realizadas todas y cada una de las ejecuciones, con el propósito de analizar la sensibilidad de los parámetros para el algoritmo FastSlam, se eligen los parámetros que mejor comportamiento global presentan para el conjunto de todos los mapas.

Ejecutadas las diferentes combinaciones de parámetros a analizar para el algoritmo EKF-Slam, se obtiene como resultado que la mejor combinación global es la formada por SigmaV = 0.3, SigmaG = 3.0, SigmaR = $ValorBaseSR/2$ y SigmaB = $ValorBaseSB/2$. En las figuras 6.17 y 6.18 se representan los resultados del promedio de las ejecuciones del algoritmo EKF-Slam para cada mapa, con los valores de los parámetros indicados anteriormente si hacemos uso de la asociación de datos no conocidos, o en las figuras 6.19 y 6.19, si hacemos uso de la asociación de datos conocidos.

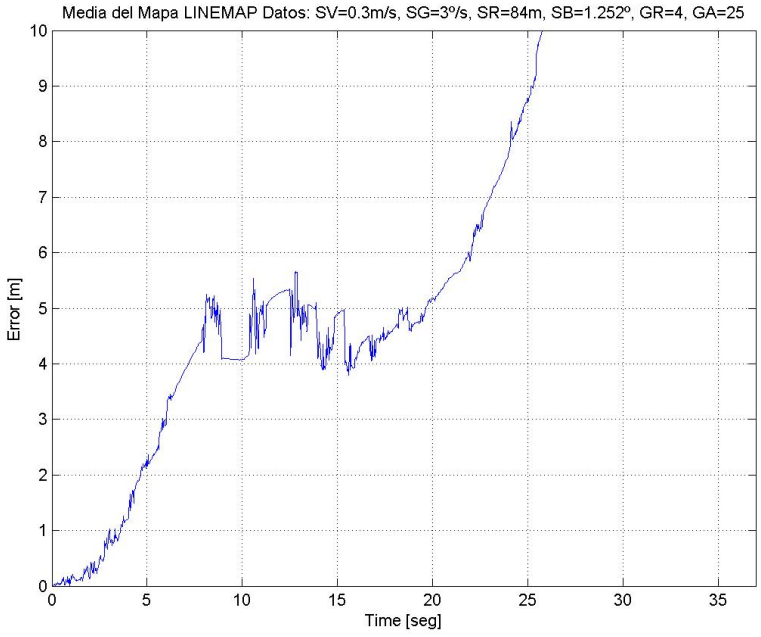
6.2.3.3. Ejecuciones del algoritmo UKF-Slam

De la misma forma que en los dos apartados anteriores, para llevar a cabo es estudio de la sensibilidad de parámetros para dicho algoritmo, se ha realizado una serie de ejecuciones, variando en cada momento el valor de alguno de dichos parámetros, y manteniendo invariables el resto de posibles parámetros que se encuentran en el fichero de configuración.

El resultado final que se obtiene para cada combinación de parámetros

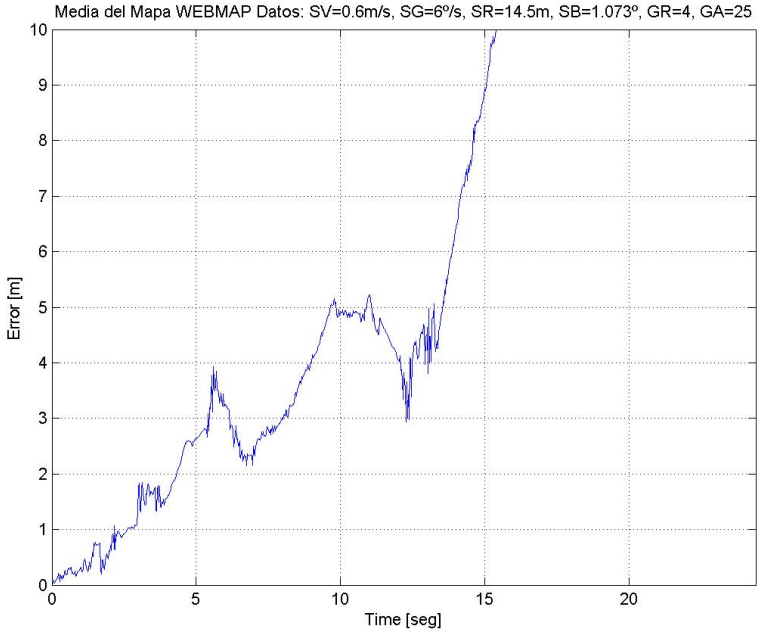


a) Mapa "SMALLMAP" con SigmaV = 1.2, SigmaG = 12.0, SigmaR = 16.0 y SigmaB = 1.032

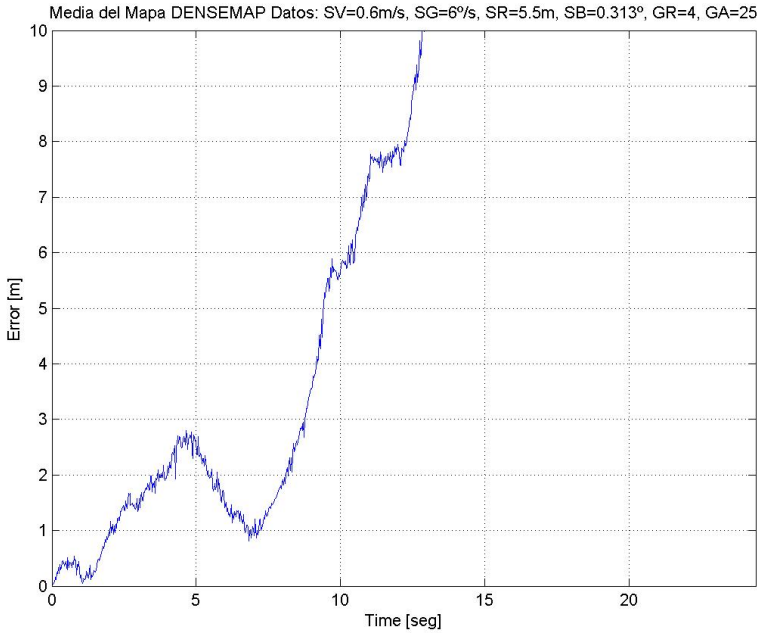


b) Mapa "LINEMAP" con SigmaV = 0.3, SigmaG = 3.0, SigmaR = 84.0 y SigmaB = 1.252

Figura 6.13: Ejemplo de resultados del promedio de las ejecuciones del algoritmo FastSlam con el mapa "SMALLMAP" y "LINEMAP", ambos con asociación de datos no conocidos.

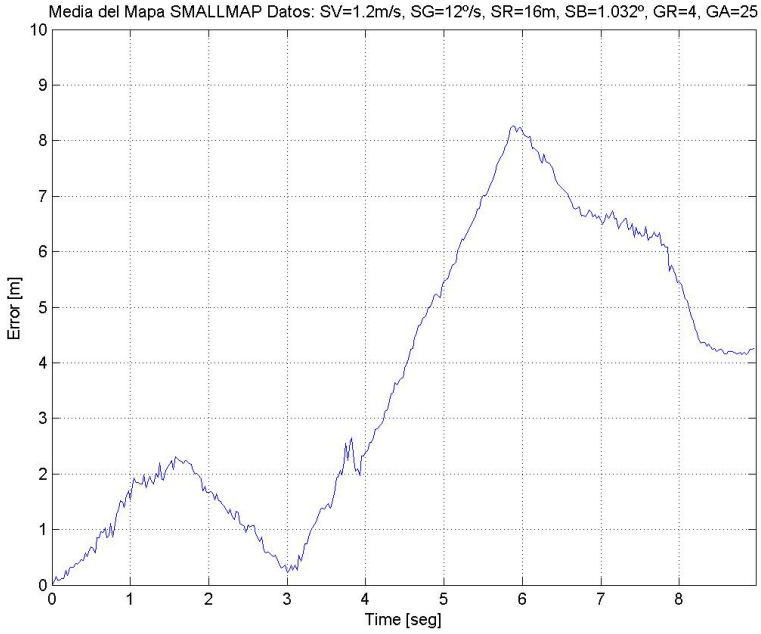


a) Mapa "WEBMAP" con SigmaV = 1.2, SigmaG = 12.0, SigmaR = 14.5 y SigmaB = 1.073

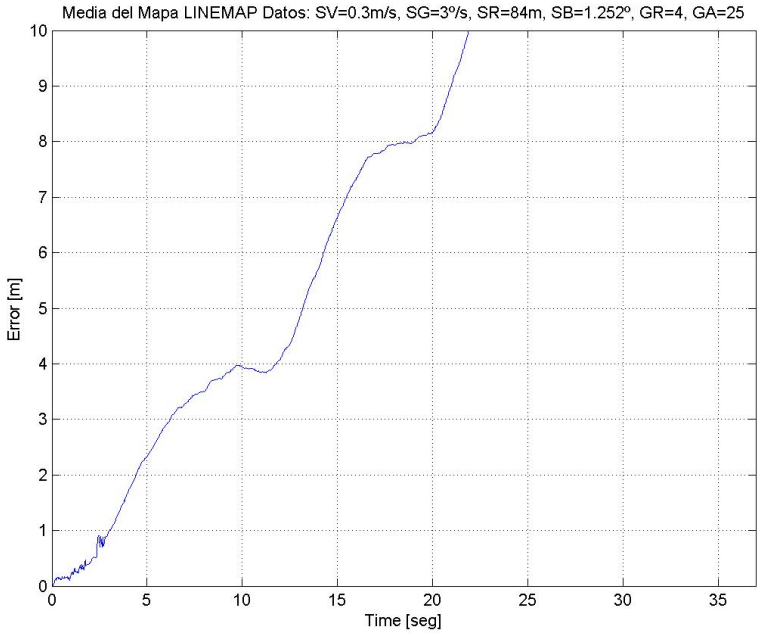


b) Mapa "DENSEMAP" con SigmaV = 0.6, SigmaG = 6.0, SigmaR = 5.5 y SigmaB = 0.313

Figura 6.14: Ejemplo de resultados del promedio de las ejecuciones del algoritmo FastSlam con el mapa "WEBMAP" y "DENSEMAP", ambos con asociación de datos no conocidos.

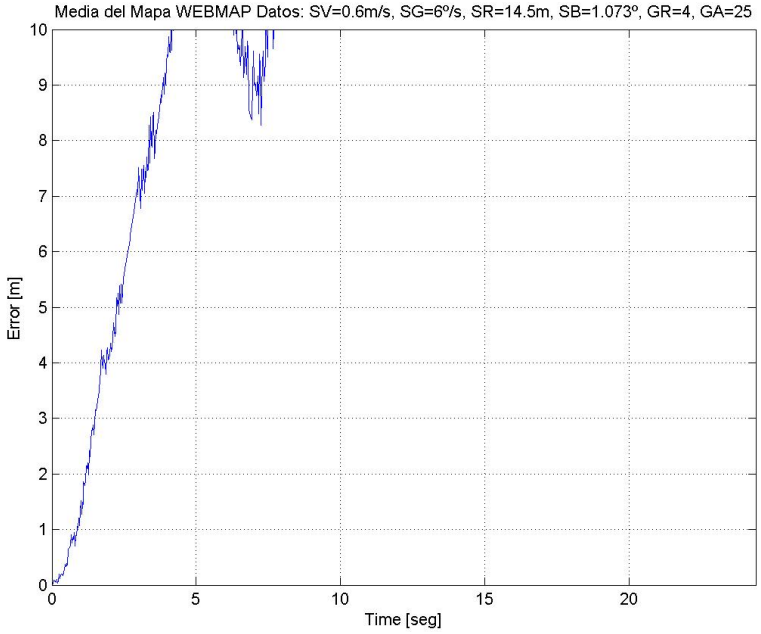


a) Mapa "SMALLMAP" con SigmaV = 1.2, SigmaG = 12.0, SigmaR = 16.0 y SigmaB = 1.032

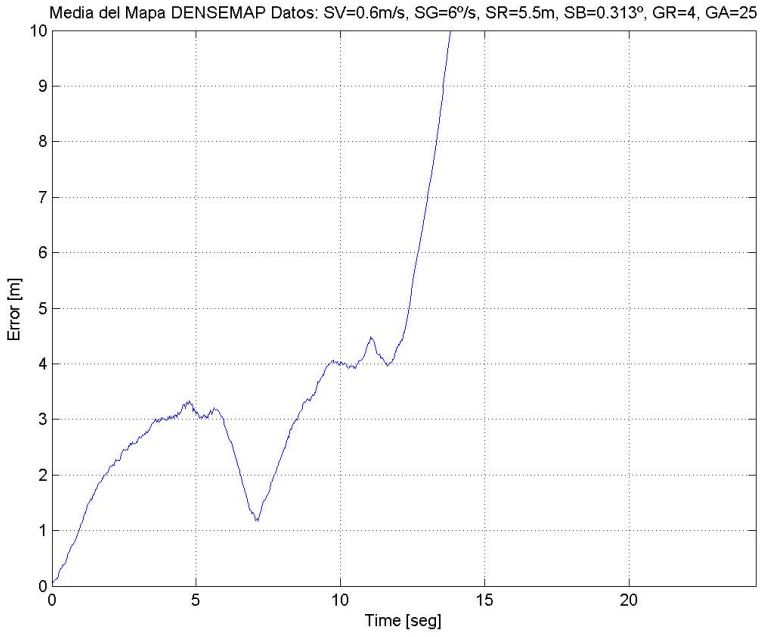


b) Mapa "LINEMAP" con SigmaV = 0.3, SigmaG = 3.0, SigmaR = 84.0 y SigmaB = 1.252

Figura 6.15: Ejemplo de resultados del promedio de las ejecuciones del algoritmo FastSlam con el mapa "SMALLMAP" y "LINEMAP", ambos con asociación de datos conocidos.

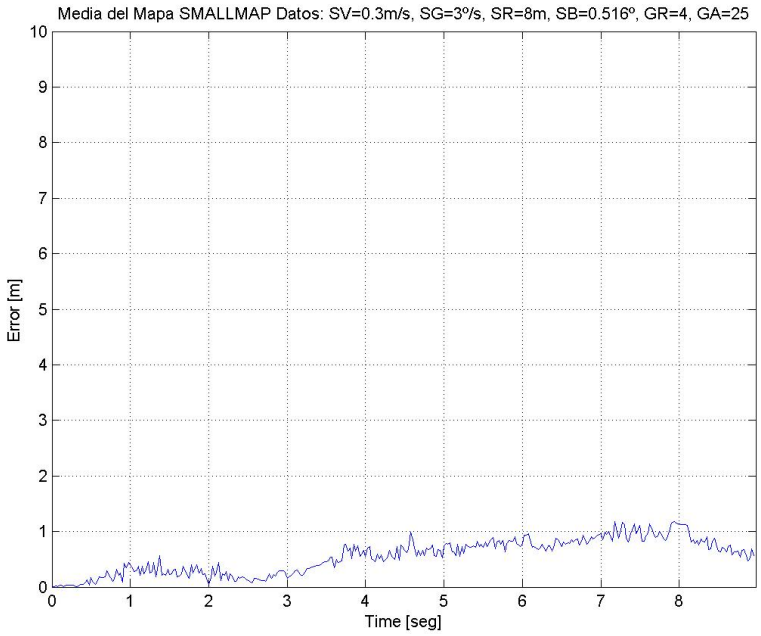


a) Mapa "WEBMAP" con SigmaV = 1.2, SigmaG = 12.0, SigmaR = 14.5 y SigmaB = 1.073

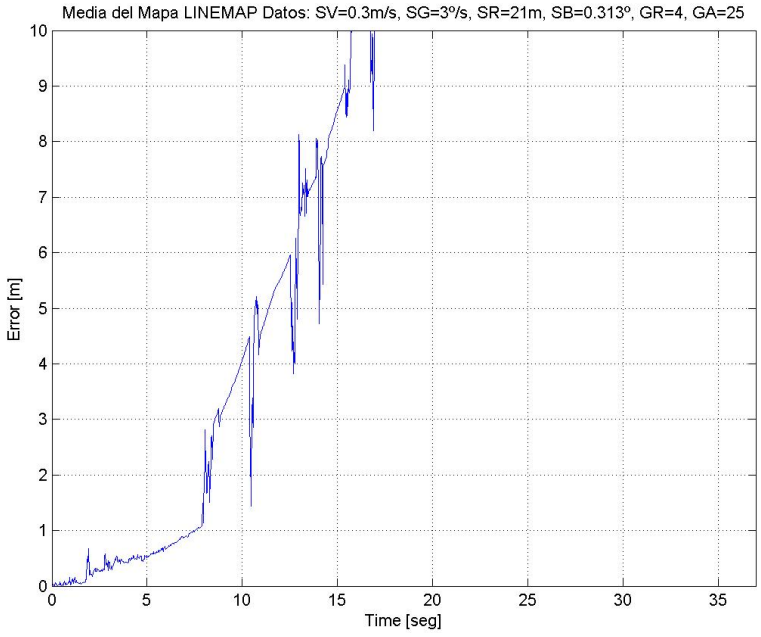


b) Mapa "DENSEMAP" con SigmaV = 0.6, SigmaG = 6.0, SigmaR = 5.5 y SigmaB = 0.313

Figura 6.16: Ejemplo de resultados del promedio de las ejecuciones del algoritmo FastSlam con el mapa "WEBMAP" y "DENSEMAP", ambos con asociación de datos conocidos.

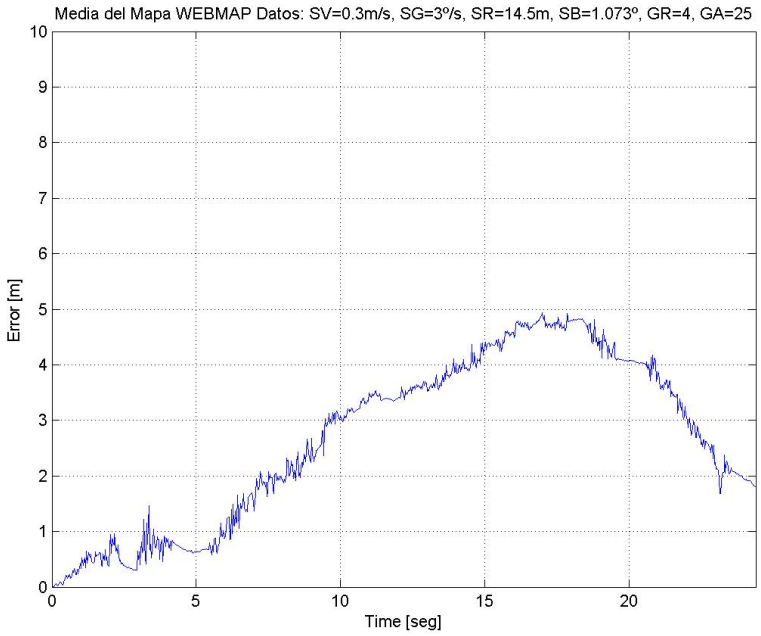


a) Mapa "SMALLMAP" con SigmaV = 0.3, SigmaG = 3.0, SigmaR = 8.0 y SigmaB = 0.516

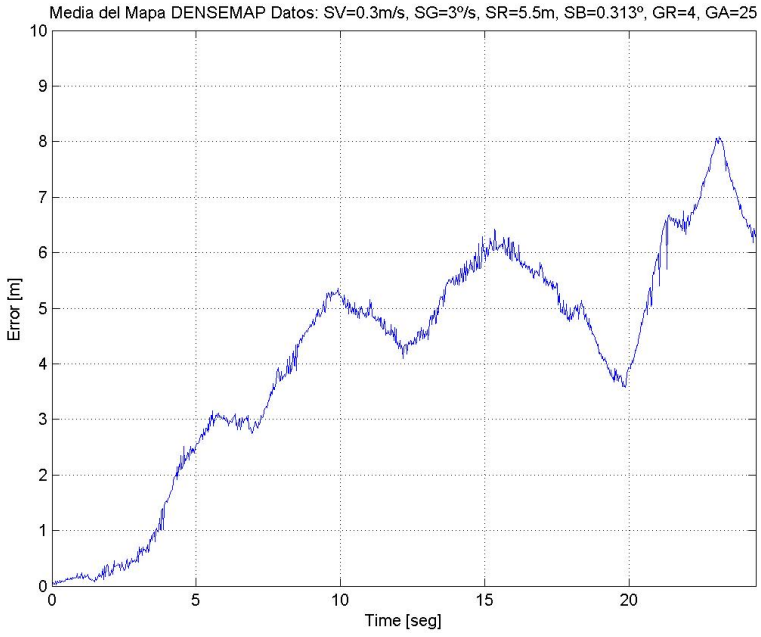


b) Mapa "LINEMAP" con SigmaV = 0.3, SigmaG = 3.0, SigmaR = 21.0 y SigmaB = 0.313

Figura 6.17: Resultados del promedio de las ejecuciones del algoritmo FastSlam con el mapa "SMALLMAP" y "LINEMAP", ambos con asociación de datos no conocidos.

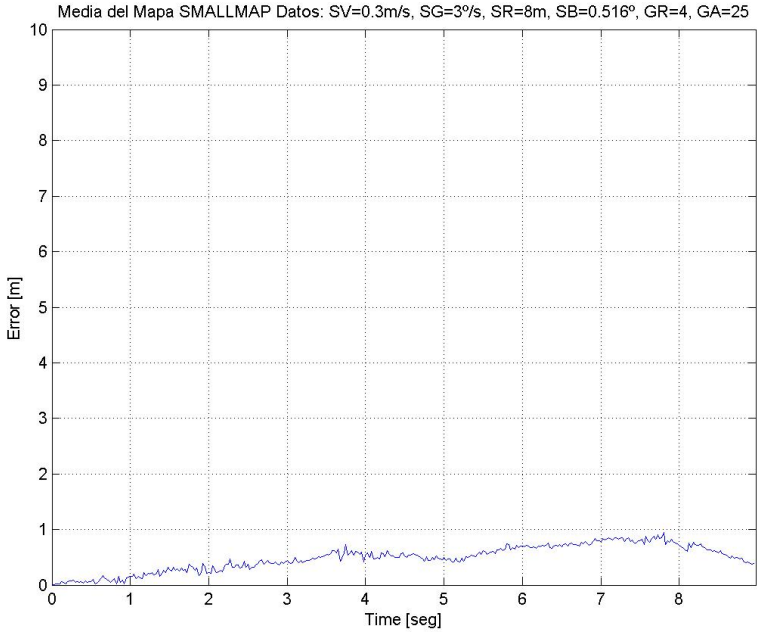


a) Mapa "WEBMAP" con SigmaV = 0.3, SigmaG = 3.0, SigmaR = 14.5 y SigmaB = 1.073

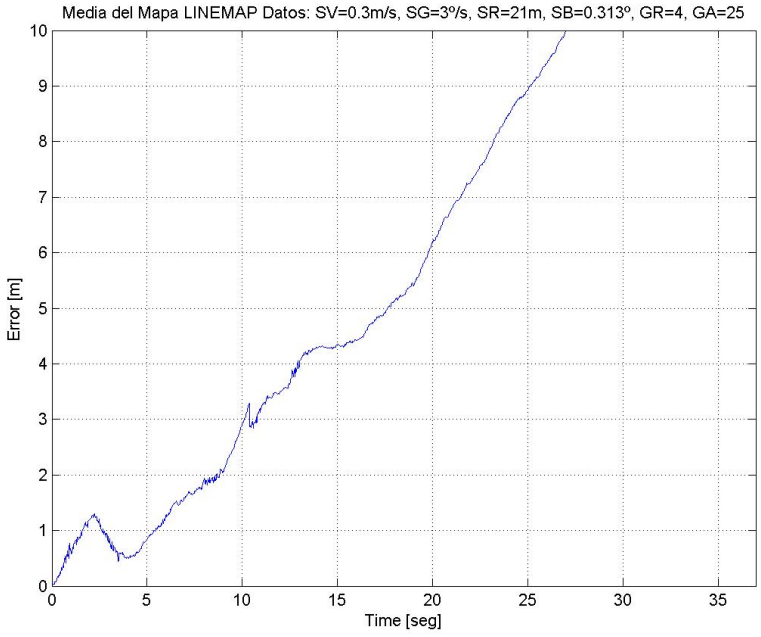


b) Mapa "DENSEMAP" con SigmaV = 0.3, SigmaG = 3.0, SigmaR = 5.5 y SigmaB = 0.313

Figura 6.18: Resultados del promedio de las ejecuciones del algoritmo FastSlam con el mapa "WEBMAP" y "DENSEMAP", ambos con asociación de datos no conocidos.

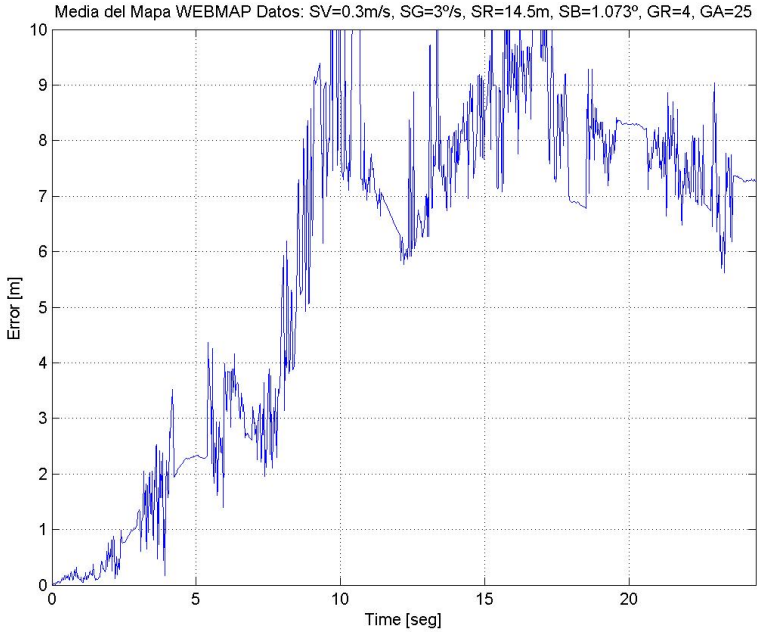


a) Mapa "SMALLMAP" con SigmaV = 0.3, SigmaG = 3.0, SigmaR = 8.0 y SigmaB = 0.516

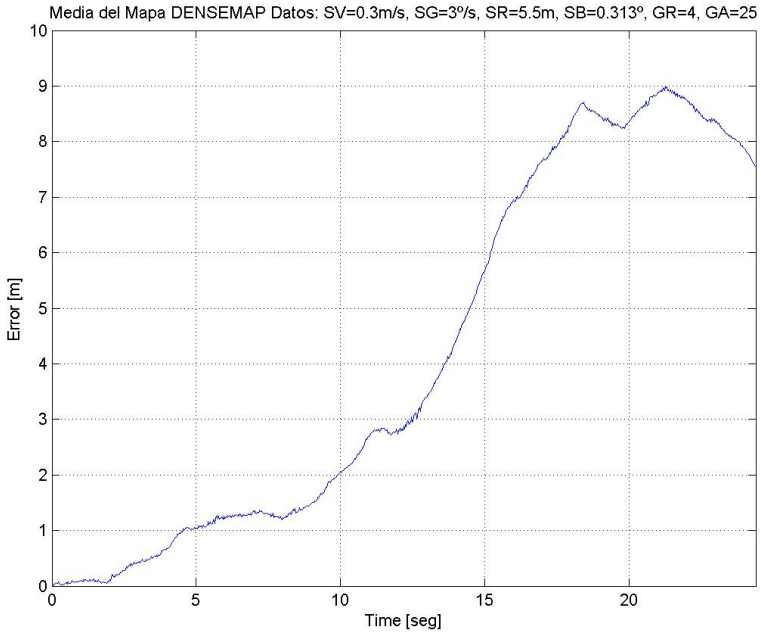


b) Mapa "LINEMAP" con SigmaV = 0.3, SigmaG = 3.0, SigmaR = 21.0 y SigmaB = 0.313

Figura 6.19: Resultados del promedio de las ejecuciones del algoritmo FastSlam con el mapa "SMALLMAP" y "LINEMAP", ambos con asociación de datos conocidos.



a) Mapa "WEBMAP" con SigmaV = 0.3, SigmaG = 3.0, SigmaR = 14.5 y SigmaB = 1.073



b) Mapa "DENSEMAP" con SigmaV = 0.3, SigmaG = 3.0, SigmaR = 5.5 y SigmaB = 0.313

Figura 6.20: Resultados del promedio de las ejecuciones del algoritmo FastSlam con el mapa "WEBMAP" y "DENSEMAP", ambos con asociación de datos conocidos.

es el resultado del promedio de diez ejecuciones, para cada una de las combinaciones, y para cada uno de los mapas definidos en la sección 5.2.

Los resultados finales obtenidos se visualizan en unas gráficas, como las representadas en las figuras 6.21 y 6.22 si hacemos uso de la asociación de datos no conocidos, o en las figuras 6.23 y 6.24 si hacemos uso de la asociación de datos conocidos. En el eje de abcisas representan el tiempo transcurrido durante la ejecución del algoritmo en el mapa (en segundos), y en el eje de ordenadas representan el error medio cometido por el algoritmo (en metros).

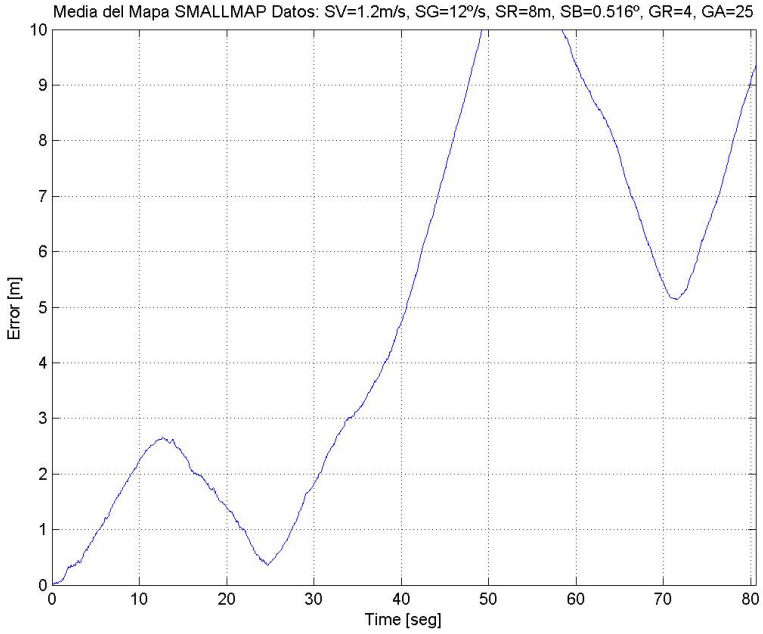
Una vez realizadas todas y cada una de las ejecuciones, con el propósito de analizar la sensibilidad de los parámetros para el algoritmo UKF-Slam, se eligen los parámetros que mejor comportamiento global presentan para el conjunto de todos los mapas.

Ejecutadas las diferentes combinaciones de parámetros a analizar para el algoritmo EKF-Slam, se obtiene como resultado que la mejor combinación global es la formada por SigmaV = 0.3, SigmaG = 3.0, SigmaR = *ValorBaseSR**2 y SigmaB = *ValorBaseSB**2. En las figuras 6.25 y 6.26 se representan los resultados del promedio de las ejecuciones del algoritmo UKF-Slam para cada mapa, con los valores de los parámetros indicados anteriormente si hacemos uso de la asociación de datos no conocidos, o en las figuras 6.27 y 6.28, si hacemos uso de la asociación de datos conocidos.

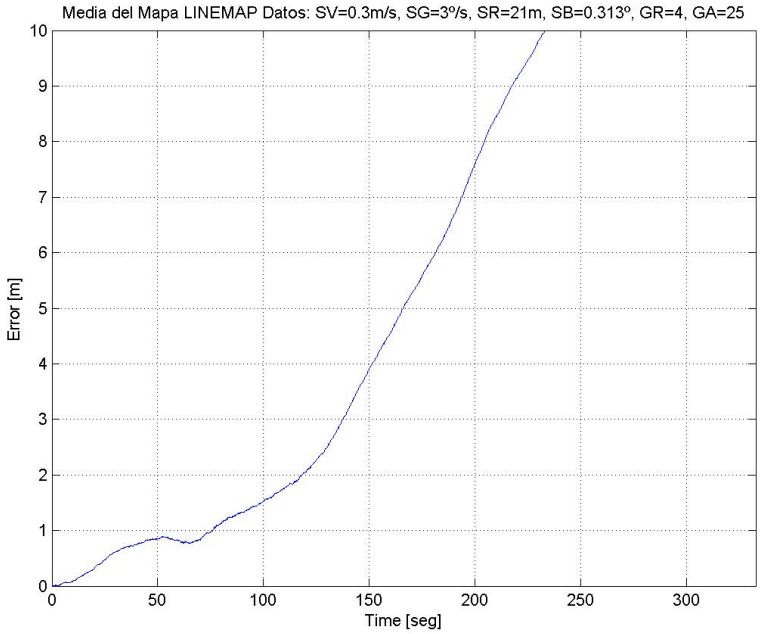
6.2.4. Resultados

Una vez realizadas las ejecuciones y comparaciones entre algoritmos, se puede indicar que el mejor algoritmo en el conjunto global es el UKF-Slam. Esto es debido a su mejor comportamiento genérico de ejecución frente a los diferentes mapas, un error de ejecución general menor con respecto a los obtenidos con los demás algoritmos, un buen tiempo de ejecución para la realización de simulaciones con respecto a otros algoritmos, etc. Todo ello se debe en parte a que el UKF-Slam se encuentra libre de derivadas, por lo que mejora los resultados obtenidos respecto al FastSlam y al EKF-Slam, representando una complejidad computacional similar a este último, al no requerir una evaluación de las matrices jacobianas.

Desde el punto de vista de la sensibilidad, el mejor algoritmo es también el UKF-Slam. En su código original, el cual solo tiene en cuenta la asociación de datos conocidos, las ejecuciones con los diferentes mapas presenta un resultado más estable que los demás algoritmos. En la versión modificada, en

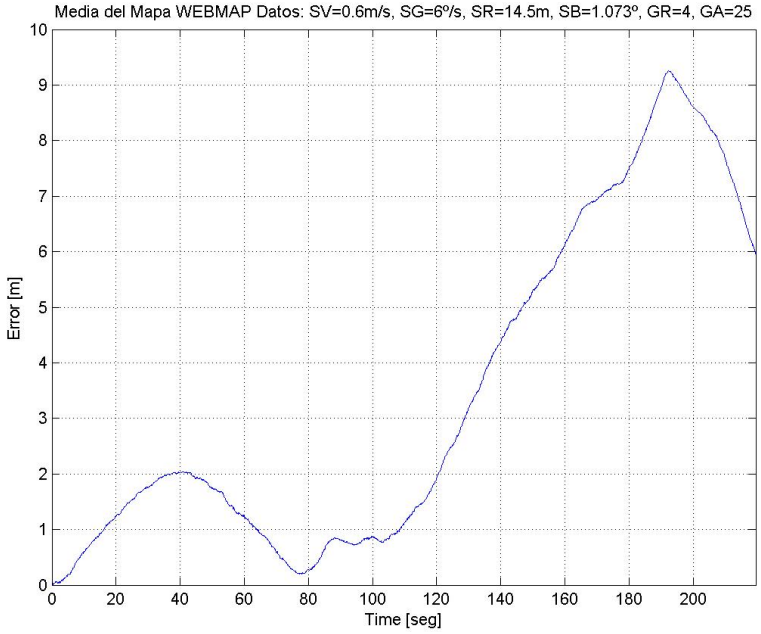


a) Mapa "SMALLMAP" con SigmaV = 1.2, SigmaG = 12.0, SigmaR = 8.0 y SigmaB = 0.516

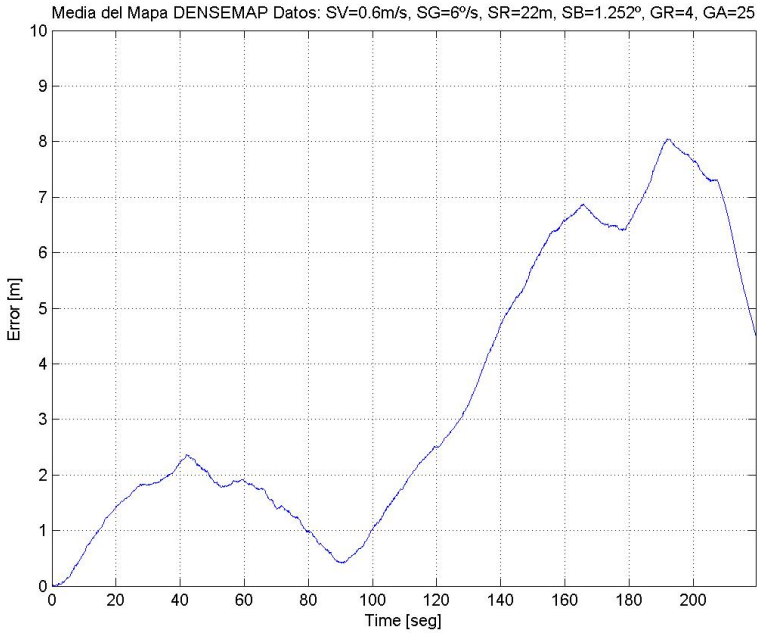


b) Mapa "LINEMAP" con SigmaV = 0.3, SigmaG = 3.0, SigmaR = 42.0 y SigmaB = 0.626

Figura 6.21: Ejemplo de resultados del promedio de las ejecuciones del algoritmo UKF-Slam con el mapa "SMALLMAP" y "LINEMAP", ambos con asociación de datos no conocidos.

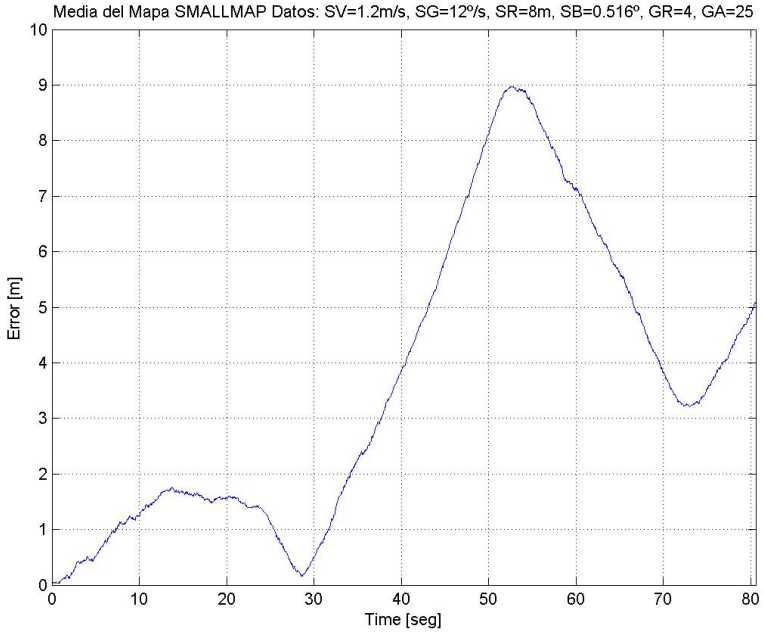


a) Mapa "WEBMAP" con SigmaV = 0.6, SigmaG = 6.0, SigmaR = 14.5 y SigmaB = 1.073

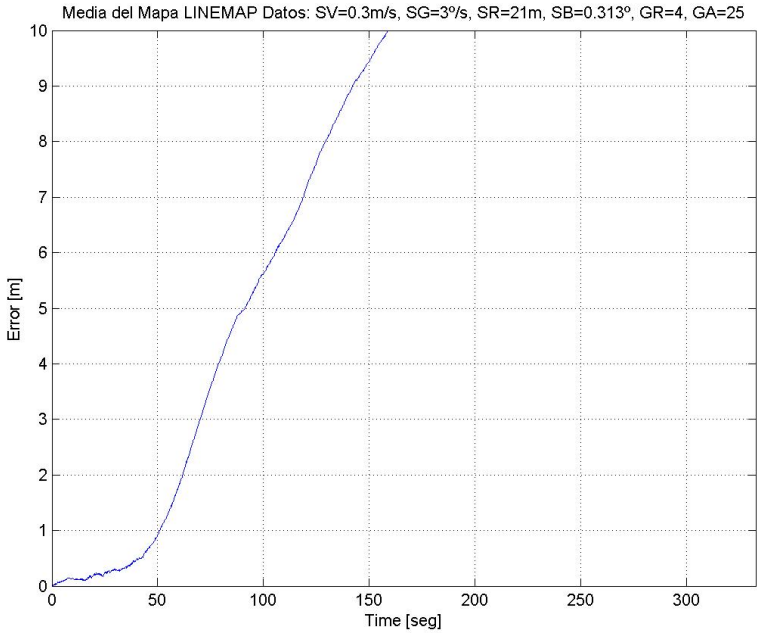


b) Mapa "DENSEMAP" con SigmaV = 0.6, SigmaG = 6.0, SigmaR = 22.0 y SigmaB = 1.252

Figura 6.22: Ejemplo de resultados del promedio de las ejecuciones del algoritmo UKF-Slam con el mapa "WEBMAP" y "DENSEMAP", ambos con asociación de datos no conocidos.

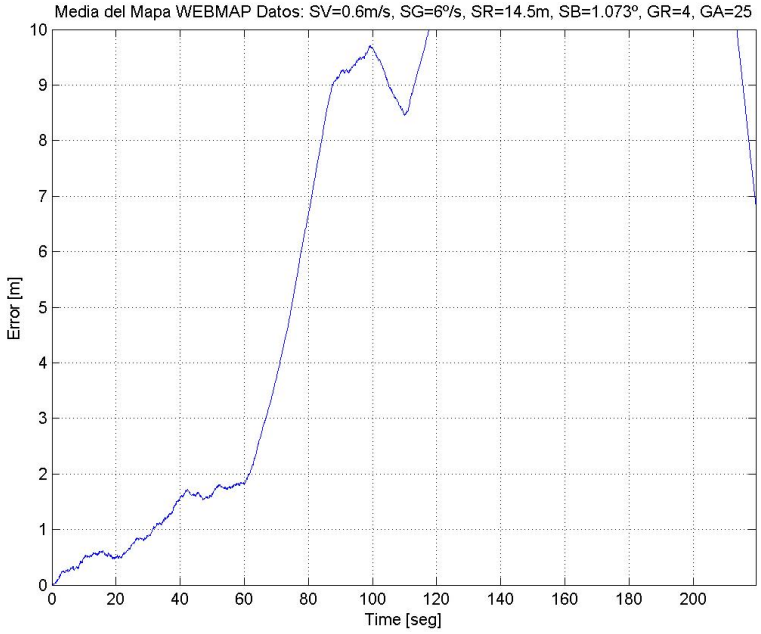


a) Mapa "SMALLMAP" con SigmaV = 1.2, SigmaG = 12.0, SigmaR = 8.0 y SigmaB = 0.516

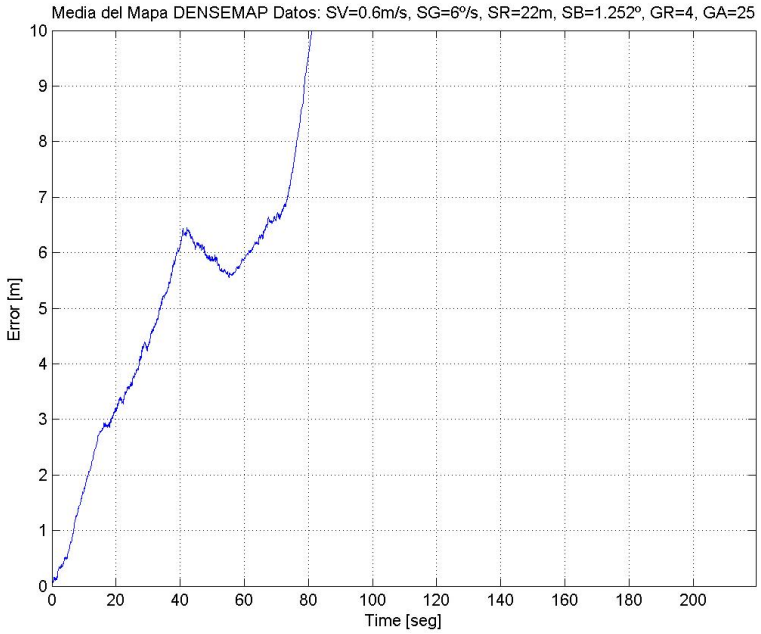


b) Mapa "LINEMAP" con SigmaV = 0.3, SigmaG = 3.0, SigmaR = 42.0 y SigmaB = 0.626

Figura 6.23: Ejemplo de resultados del promedio de las ejecuciones del algoritmo UKF-Slam con el mapa "SMALLMAP" y "LINEMAP", ambos con asociación de datos conocidos.

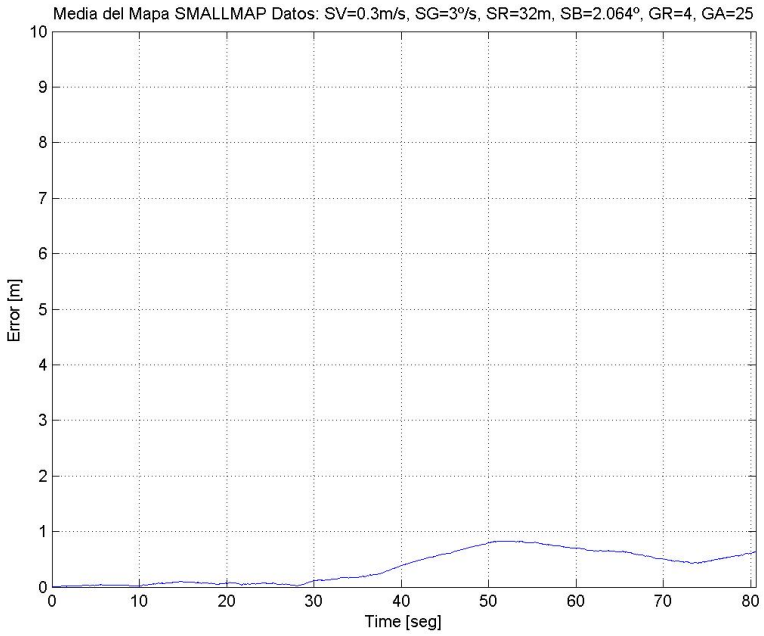


a) Mapa "WEBMAP" con SigmaV = 0.6, SigmaG = 6.0, SigmaR = 14.5 y SigmaB = 1.073

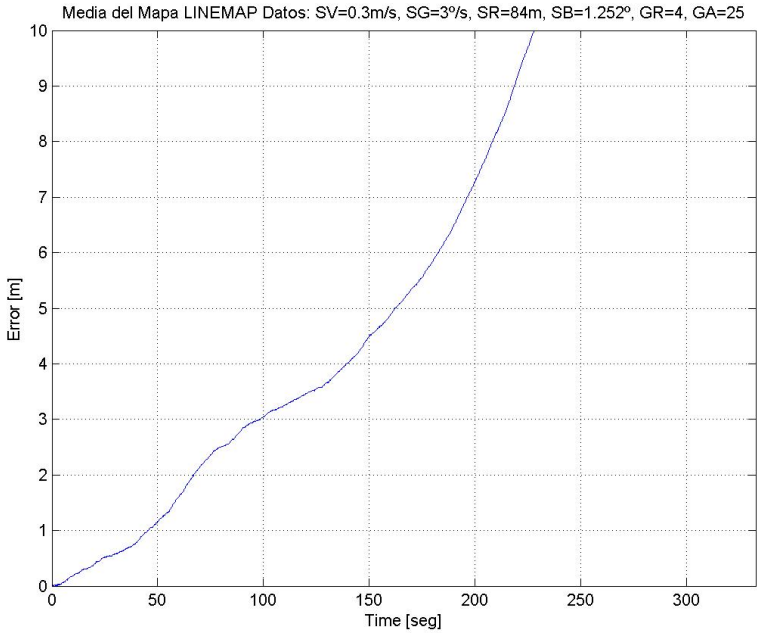


b) Mapa "DENSEMAP" con SigmaV = 0.6, SigmaG = 6.0, SigmaR = 22.0 y SigmaB = 1.252

Figura 6.24: Ejemplo de resultados del promedio de las ejecuciones del algoritmo UKF-Slam con el mapa "WEBMAP" y "DENSEMAP", ambos con asociación de datos conocidos.

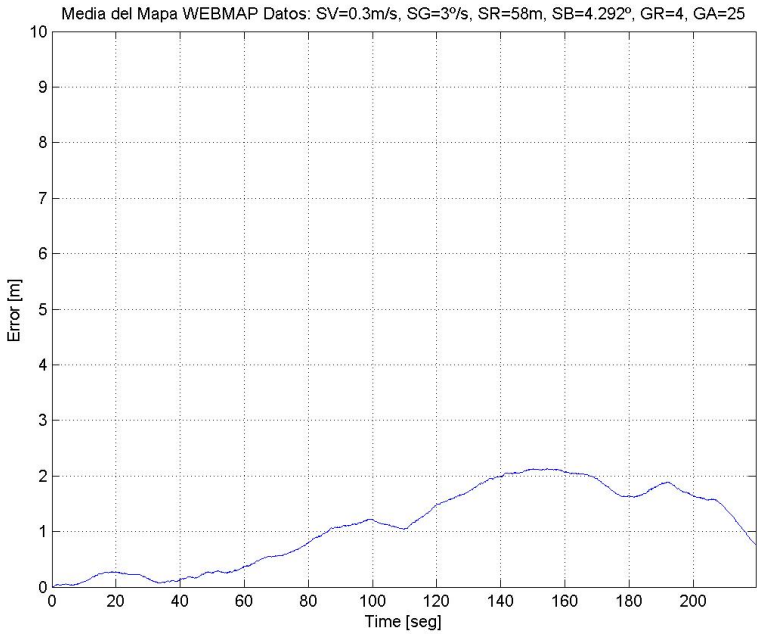


a) Mapa "SMALLMAP" con SigmaV = 0.3, SigmaG = 3.0, SigmaR = 32.0 y SigmaB = 2.064

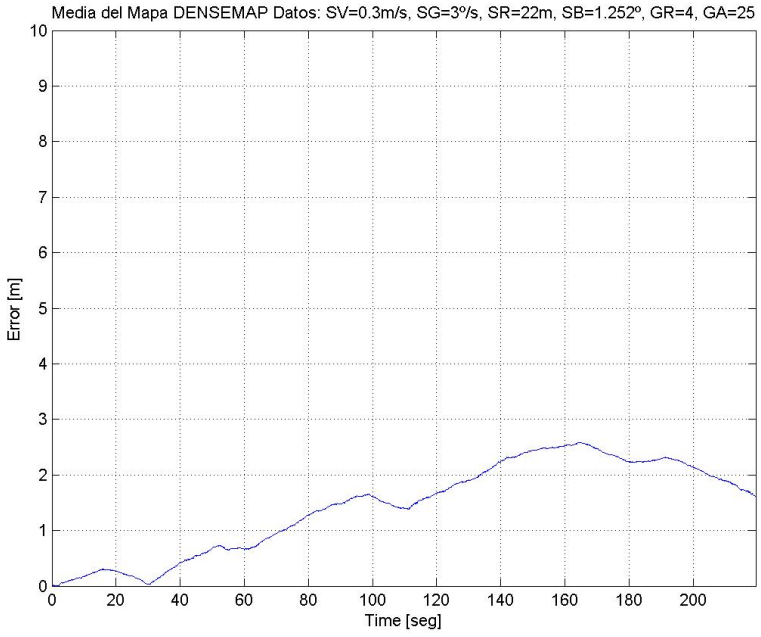


b) Mapa "LINEMAP" con SigmaV = 0.3, SigmaG = 3.0, SigmaR = 84.0 y SigmaB = 1.252

Figura 6.25: Resultados del promedio de las ejecuciones del algoritmo UKF-Slam con el mapa "SMALLMAP" y "LINEMAP", ambos con asociación de datos no conocidos.

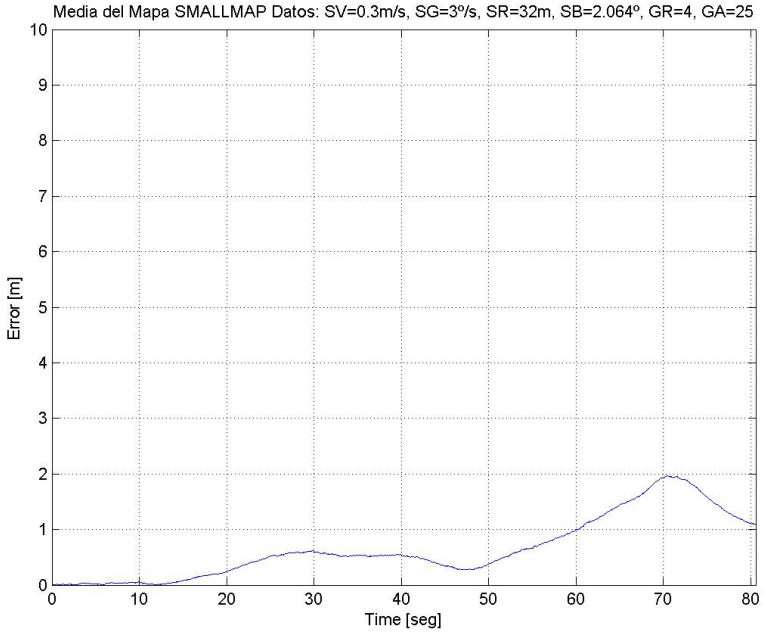


a) Mapa "WEBMAP" con SigmaV = 0.3, SigmaG = 3.0, SigmaR = 58.5 y SigmaB = 4.292

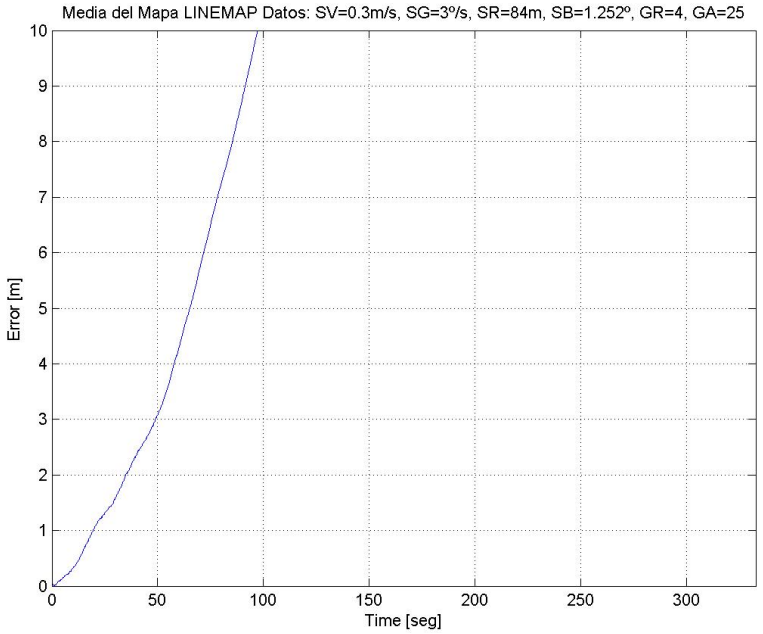


b) Mapa "DENSEMAP" con SigmaV = 0.3, SigmaG = 3.0, SigmaR = 22.0 y SigmaB = 1.252

Figura 6.26: Resultados del promedio de las ejecuciones del algoritmo UKF-Slam con el mapa "WEBMAP" y "DENSEMAP", ambos con asociación de datos no conocidos.

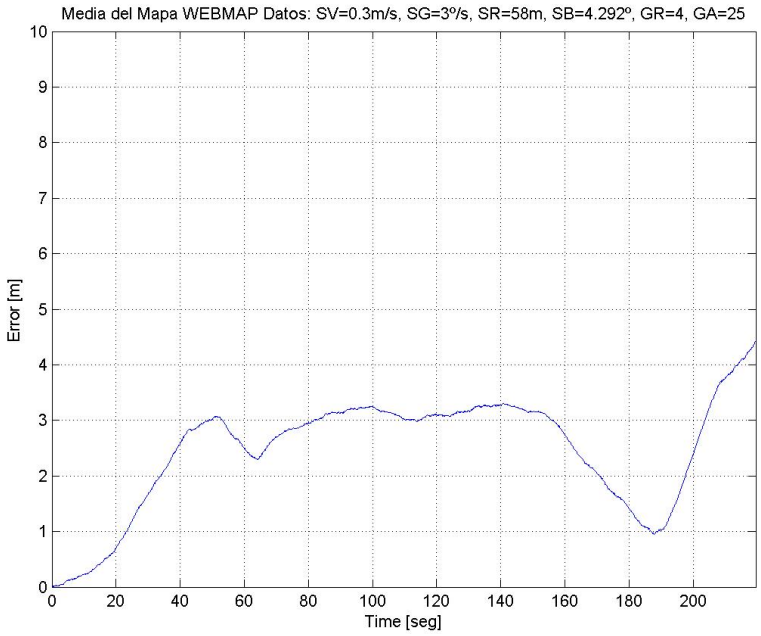


a) Mapa "SMALLMAP" con SigmaV = 0.3, SigmaG = 3.0, SigmaR = 32.0 y SigmaB = 2.064

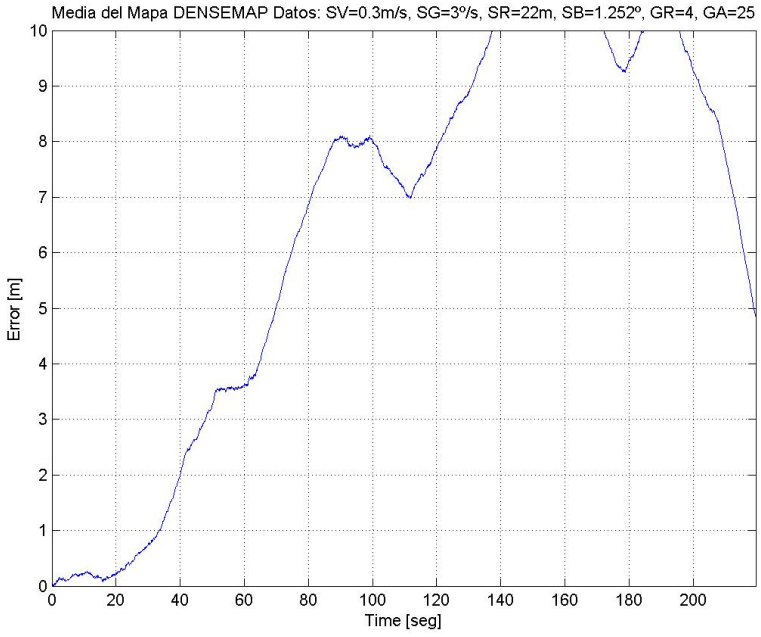


b) Mapa "LINEMAP" con SigmaV = 0.3, SigmaG = 3.0, SigmaR = 84.0 y SigmaB = 1.252

Figura 6.27: Resultados del promedio de las ejecuciones del algoritmo UKF-Slam con el mapa "SMALLMAP" y "LINEMAP", ambos con asociación de datos conocidos.



a) Mapa "WEBMAP" con SigmaV = 0.3, SigmaG = 3.0, SigmaR = 58.5 y SigmaB = 4.292



b) Mapa "DENSEMAP" con SigmaV = 0.3, SigmaG = 3.0, SigmaR = 22.0 y SigmaB = 1.252

Figura 6.28: Resultados del promedio de las ejecuciones del algoritmo UKF-Slam con el mapa "WEBMAP" y "DENSEMAP", ambos con asociación de datos conocidos.

la que se añade la posibilidad de ejecutar el algoritmo con asociación de datos no conocidos, también presenta un comportamiento más estable en las ejecuciones, con un menor error en las medidas, con respecto a los demás algoritmos.

Teniendo en cuenta los resultados de las ejecuciones, clasificados por mapas, el mejor algoritmo para el mapa "SMALLMAP" es el algoritmo EKF-Slam; para el mapa "LINEMAP" es el algoritmo UKF-Slam; el algoritmo EKF-Slam es el mejor para el mapa "DENSEMAP"; y por último, para el mapa "WEBMAP", el mejor algoritmo es el UKF-Slam. Estas conclusiones derivan de los comportamientos globales de estos algoritmos con los mapas utilizados, teniendo en cuenta no solo los mejores valores de los parámetros comunes, sino el comportamiento de los algoritmos ante los cambios de estos parámetros, además de los comportamientos en las ejecuciones con asociación de datos conocidos o no conocidos.

Puede resultar llamativo la no presencia del algoritmo FastSlam dentro de los mejores. La causa creemos que está relacionada con los aspectos ya comentados de la mayor dificultad de parametrización del algoritmo (posee cuatro parámetros frente a los demás que poseen sólo dos) y a la existencia de problemas de estabilidad numérica del algoritmo cuando el número de partículas es elevado.

6.3. Eficiencia de los algoritmos

El estudio de la eficiencia nos permite medir de alguna forma el coste (en tiempo y recursos) que consume un algoritmo para encontrar la solución, y nos ofrece la posibilidad de comparar distintos algoritmos que resuelven un mismo problema [Gu00], o sea, permite seleccionar el algoritmo más eficiente para resolver un problema, cuando existen varios correctos que compiten entre sí [Di01].

Las ventajas que presenta el análisis de la eficiencia de un algoritmo son [Hu05]:

- Mejor comprensión de los algoritmos.
- Diseñar algoritmos mejores.
- Determinar la escalabilidad.

Para medir la eficiencia de un algoritmo existe un análisis empírico y otro teórico [Hu05]. El primero ofrece una medida real (a posteriori), consistente en

medir el tiempo de ejecución del algoritmo para unos valores de entrada dados y en un ordenador concreto [Gu00]. El segundo proporciona una medida teórica (a priori), que consiste en obtener una función que acote (por arriba o por abajo) el tiempo de ejecución del algoritmo para unos valores de entrada dados.

Ambas medidas son importantes puesto que, si bien la primera representa las medidas reales del comportamiento del algoritmo, la segunda nos ofrece estimaciones del comportamiento de los algoritmos de forma independiente del ordenador en donde serán implementados y sin necesidad de ejecutarlos. Estas medidas son funciones temporales de los datos de entrada.

Nos centraremos en el análisis teórico, el cual realiza una comparación independiente de la implementación, sin necesidad de implementar el algoritmo ni de establecer baterías de pruebas [Di01]. Si se desea aplicar dicho análisis teórico, es necesario analizar la cantidad de recursos de tiempo y espacio que consumirá su ejecución, independientemente de la máquina en que se ejecute.

Se dice que un algoritmo para algún problema requiere un orden de *t(n)* para una función dada *t*, si existe una constante positiva y una implementación del algoritmo capaz de resolver todos los casos de tamaño n en un tiempo que no sea superior a *ct(n)*. Según el principio de invarianza, si cualquier implementación del algoritmo tiene la propiedad requerida, entonces también la tienen todas las demás, aunque la constante pueda cambiar de unas implementaciones a otras.

Es necesario utilizar la notación asintótica para expresar el tiempo requerido por el algoritmo en función del tamaño del ejemplar del problema, el número de operaciones elementales que requiere el algoritmo determina esta función [Di01]. La notación asintótica proporciona el comportamiento del algoritmo para valores del tamaño del ejemplar lo suficientemente grandes sin necesidad de ejecutarlo, o sea, estudia el comportamiento del algoritmo cuando el tamaño de las entradas, n, es lo suficientemente grande, sin tener en cuenta lo que ocurre para entradas pequeñas y obviando factores constantes [Hu05].

Para facilitar la evaluación del tiempo de ejecución de un algoritmo que se obtiene por agregación de varios previamente analizados, se introducen las operaciones con notaciones asintóticas.

O(*f(n)*), que ha de leerse *del orden de f(n)*, es el conjunto de todas las funciones *t(n)* no negativas, acotadas superiormente por un múltiplo real positivo de *f(n)* para valores de n suficientemente grandes, es decir, a partir

de un cierto umbral n_0 en adelante.

Esta notación 0 posee varias reglas a tener en cuenta [Be04]:

- Asignaciones y expresiones simples. x = x+1; t(n) = O(1)
- Secuencias de instrucciones. I1; I2; t(n) = t1(n) + t2(n) = max O(t1(n)), O(t2(n))
- Estructuras de control condicionales (if-then-else). t(n) = O (t_condición(n)) + max O(t_then(n)), O(t_else(n))
- Estructuras de control iterativas. Producto del número de iteraciones por la complejidad de cada iteración. Si la complejidad varía en función de la iteración, obtenemos una sumatoria. Si no conocemos con exactitud el número de iteraciones (bucles while y do-while), se estima este número para el peor caso.

Por último, se ha de tener en cuenta como regla general para cálculo de la eficiencia, que se ha de empezar por la parte más interna del algoritmo y se avanza (mediante sucesivas aplicaciones de la regla de la suma y/o el producto) hacia las partes más externas [Hu05]. Para ello, tener en cuenta en cada momento lo siguiente:

- Sentencia Simple.
  - Su tiempo de ejecución está acotado superiormente por una constante, O(1).
- Secuencia de sentencias.
  - Se aplica la regla de la suma, que nos indica que el orden de todo el bloque es el máximo de los ordenes de eficiencia de dicho bloque.
- Sentencia condicional.
  - Si (condición) Entonces Accion_Si Sino Accion_No. Pertenece al orden del max O(cond),O(Acc_SI),O(Acc_NO) .
- Bucle.
  - Se aplica la regla del producto. El tiempo asociado se obtiene como producto del orden del numero de iteraciones que realiza el bucle por el orden del tiempo de ejecución del conjunto de sentencias que forman el cuerpo del bucle.

- Llamadas a funciones.
    - Se analiza primero el orden de eficiencia de las funciones a las que se llama y posteriormente, considerando las reglas anteriores, se pasa a calcular el tiempo de ejecución de la función que las llama.

6.3.1. Algoritmo EKF-Slam

A continuación se detalla el cálculo de la eficiencia para el algoritmo EKF-Slam, a través de las principales estructuras del mismo, sin hacer inciso en aquellas sentencias que no influyan en el cálculo de dicha eficiencia.

```
function data= ekfslam_sim(lm,wp) //t(n)=O(N^3)
...
h = setup_animations; //O(1)
...
data = initialise_store(x,P,x); //O(1)
...
while iwp ≠ 0 //∑_{i=1}^{n} O(i^2) = O(N^3)
    [G,iwp] = compute_steering(xtrue,wp,iwp,AT_WAYPOINT,G,
    RATEG,MAXG,dt); //O(1)
    ...
    xtrue = vehicle_model(xtrue,V,G,WHEELBASE,dt); //O(1)
    [Vn,Gn] = add_control_noise(V,G,Q,SWITCH_CONTROL_NOISE);
    //O(1)
    [x,P] = predict (x,P,Vn,Gn,QE,WHEELBASE,dt); //O(1)
    [x,P] = observe_heading(x,P,xtrue(3),SWITCH_HEADING_KNOWN);
    //O(1)
    ...
    if dtsum ≥ DT_OBSERVE //max{O(1), O(N^2)} = O(N^2)
        ...
        [z,ftag_visible] = get_observations(xtrue,lm,ftag,
        MAX_RANGE); //O(N)
        z = add_observation_noise(z,R,SWITCH_SENSOR_NOISE);
        //O(1)
```

```
    if SWITCH_ASSOCIATION_KNOWN == 1
    //max{O(1), O(N), O(N^2)} = O(N^2)
        [zf,idf,zn, da_table] = data_associate_known(x,z,
        ftag_visible,da_table); //O(N)
    else
        [zf,idf, zn] = data_associate(x,P,z,RE,GATE_REJECT,
        GATE_AUGMENT); //O(N^2)
    end
    if SWITCH_USE_IEKF == 1 //max{O(1), O(N^2), O(N)} = O(N^2)
        [x,P] = update_iekf(x,P,zf,RE,idf,5); //O(N^2)
    else
        [x,P] = update(x,P,zf,RE,idf,SWITCH_BATCH_UPDATE);
        //O(N)
    end
    [x,P] = augment(x,P,zn,RE); //O(N)
end

data = store_data(data,x,P,xtrue); //O(1)

xt = transformtoglobal(veh,xtrue); //O(1)

xv = transformtoglobal(veh,x(1:3)); //O(1)

...

ptmp = make_covariance_ellipses(x(1:3),P(1:3,1:3)); //O(1)

...

if dtsum==0 //max{O(1), O(N)} = O(N)
    ...
    if not isempty(z) //max{O(1), O(N)} = O(N)
        plines = make_laser_lines (z,x(1:3)); //O(1)
        ...
        pcov = make_covariance_ellipses(x,P); //O(N)
    end
end
```

```
    ...

end
data = finalise_data(data); //O(1)
...
```

En resumen, la eficiencia del algoritmo EKF-Slam es del orden $O(N^3)$.

6.3.2. Algoritmo FastSlam

En esta subsección se detalla el cálculo de la eficiencia para el algoritmo FastSlam, a través de las principales estructuras del mismo, al igual que en el algoritmo anterior, sin hacer incapié en aquellas sentencias que no influyan en el cálculo de dicha eficiencia.

```
function data= fastslam1_sim(lm,wp) //t(n)=O(N^4)
...
h = setup_animations(lm,wp); //O(1)
...
particles = initialise_particles(NPARTICLES); //O(N)
...
while iwp ≠ 0 //∑_{i=1}^{n} O(i^3) = O(N^4)

    [G,iwp] = compute_steering(xtrue,wp,iwp,AT_WAYPOINT,G,RATEG,
    MAXG,dt); //O(1)

    ...

    xtrue = predict_true(xtrue,V,G,WHEELBASE,dt); //O(1)

    [Vn,Gn] = add_control_noise(V,G,Q,SWITCH_CONTROL_NOISE);
    //O(1)

    for i=1:NPARTICLES //∑_{i=1}^{n} O(1) = O(N)

        particles(i) = predict(particles(i),Vn,Gn,Qe,WHEELBASE,
        dt,SWITCH_PREDICT_NOISE); //O(1)

        ...

    end

    ...

    if dtsum ≥ DT_OBSERVE //max{O(1), O(N^3)} = O(N^3)
```

```
...
[z,ftag_visible] = get_observations(xtrue,lm,ftag,
MAX_RANGE); //O(N)
z = add_observation_noise(z,R,SWITCH_SENSOR_NOISE);
//O(1)
if not isempty(z) //max{O(1),O(1)} = O(1)
    plines = make_laser_lines(z,xtrue); //O(1)
end
if SWITCH_ASSOCIATION_KNOWN == 1
//max{O(1),O(N^2),O(N^3)} = O(N^3)
    ...
    [zf,idf,zn,da_table] = data_associate_known(z,
    ftag_visible,da_table,Nf); //O(N)
    for i=1:NPARTICLES //sum_{i=1}^{n} O(i) = O(N^2)
        if not isempty(zf) //max{O(1),O(N)} = O(N)
          w = compute_weight(particles(i),zf,idf,R);
          //O(N)
          ...
          particles(i) = feature_update(particles(i),zf,
          idf,R); //O(N)
        end
        if not isempty(zn) //max{O(1),O(N)} = O(N)
          particles(i) = add_feature(particles(i),zn,R);
          //O(N)
        end
    end
    ...
    particles = resample_particles(particles,NEFFECTIVE,
    SWITCH_RESAMPLE); //O(N)
else
    ...
    for i=1:NPARTICLES //sum_{i=1}^{n} O(i^2) = O(N^3)
        [zfi,idfi,zni] = data_associate_f1
        (particles(i).xv,particles(i).xf,particles(i).Pf,
        z,Re,GATE_REJECT,GATE_AUGMENT); //O(N^2)
        if not isempty(zfi) //max{O(1),O(N)} = O(N)
```

```
                w = compute_weight(particles(i),zf{i},idf{i},
                R); //O(N)
                ...
                particles(i) = feature_update(particles(i),
                zf{i},idf{i},R); //O(N)
              end
              if not isempty(zni) //max{O(1),O(N)} = O(N)
                particles(i) = add_feature(particles(i),zn{i},
                R); //O(N)
              end
            end
            ...
            particles = resample_particles(particles,NEFFECTIVE,
            SWITCH_RESAMPLE); //O(N)

    end

    do_plot(h,particles,xtrue,plines,veh) //O(N)

end
...
```

Globalmente, la eficiencia del algoritmo FastSlam es del orden $O(N^4)$.

6.3.3. Algoritmo UKF-Slam

Por último, se detalla el cálculo de la eficiencia para el algoritmo UKF-Slam, a través de las principales estructuras del mismo, que al igual que en los algoritmos anteriores, no se tienen en cuenta aquellas sentencias que no influyen en el cálculo de dicha eficiencia.

```
function data= ukfslam_sim(lm,wp) //t(n)=O(N^3)
...
h = setup_animations; //O(1)
...
DATA = initialise_store(XX,PX,XX); //O(1)
...
while iwp ≠ 0 //∑_{i=1}^{n} O(i^2) = O(N^3)

    [G,iwp] = compute_steering(xtrue,wp,iwp,AT_WAYPOINT,G,RATEG,
    MAXG,dt); //O(1)
```

```
...

xtrue = vehicle_model(xtrue,V,G,WHEELBASE,dt); //O(1)

[Vn,Gn] = add_control_noise(V,G,Q,SWITCH_CONTROL_NOISE);
//O(1)

predict(Vn,Gn,QE,WHEELBASE,dt); //O(1)

observe_heading(xtrue(3),SWITCH_HEADING_KNOWN); //O(1)

...

if dtsum ≥ DT_OBSERVE //max{O(1),O(N^2)} = O(N^2)

    ...

    [z,ftag_visible] = get_observations(xtrue,lm,ftag,
    MAX_RANGE); //O(N)

    z = add_observation_noise(z,R,SWITCH_SENSOR_NOISE);
    //O(1)

    if SWITCH_ASSOCIATION_KNOWN == 1
    //max{O(1),O(N),O(N^2)} = O(N^2)

        [zf,idf,zn,da_table] = data_associate_known(XX,z,
        ftag_visible,da_table); //O(N)

    else

        [zf,idf,zn] = data_associate(XX,PX,z,RE,GATE_REJECT,
        GATE_AUGMENT); //O(N^2)

    end

    update(zf,RE,idf); //O(N)

    augment(zn,RE); //O(N)

end

store_data(XX, PX, xtrue); //O(1)

xt = transform_to_global(veh,xtrue); //O(1)

...

if SWITCH_GRAPHICS //max{O(1),O(N)} = O(N)

    xv = transform_to_global(veh,XX(1:3)); //O(1)
```

```
        pvcov = make_vehicle_covariance_ellipse(XX,PX); //O(1)
        ...
        if dtsum==0 and not isempty(z) //max{O(1),O(N)} = O(N)
            ...
            plines = make_laser_lines(z,XX(1:3)); //O(1)
            ...
            pfcov = make_feature_covariance_ellipses(XX,PX);
            //O(N)
            ...
        end
    end
    ...
end
...
data = finalise_data(DATA); //O(1)
...
```

Por tanto, la eficiencia del algoritmo UKF-Slam es del orden $O(N^3)$.

6.3.4. Conclusiones

En el análisis respecto a la eficiencia de los algoritmos, se puede observar en las secciones 6.3.1, 6.3.2 y 6.3.3 cómo la eficiencia del algoritmo EKF-Slam posee un resultado t(n)=$O(N^3)$, el algoritmo FastSlam posee un resultado t(n)=$O(N^4)$ y el algoritmo UKF-Slam posee un resultado t(n)=$O(N^3)$, respectivamente. Con respecto a dichos resultados, se puede concluir que tanto el algoritmo EKF-Slam como el algoritmo UKF-Slam poseen ambos la mejor eficiencia.

Tabla 6.1: Resultado del análisis de la eficiencia de los algoritmos.

EKF-Slam	**FastSlam**	**UKF-Slam**
$O(N^3)$	$O(N^4)$	$O(N^3)$

La tabla 6.1 muestra los valores obtenidos para cada uno de los algoritmos.

Capítulo 7

Conclusión

Este proyecto de fin de carrera abordaba el análisis y comparativa de diferentes algoritmos de SLAM. Creemos que los objetivos propuestos se han alcanzado, destacando las siguientes aportaciones:

- Se ha realizado una extensa revisión bibliográfica, tanto de los antecedentes como del estado actual del tema. En ella se puede comprender el alcance que los algoritmos de SLAM tienen hoy día, los esfuerzos que se llevan a cabo por mejorar dichos algoritmos y poder dotar cada vez más a los robots de una mayor autonomía, intentando solventar problemas odométricos, de ubicación en un entorno no conocido, etc ...

- Se han analizado diferentes entornos de desarrollo, seleccionándose el más adecuado para cubrir con las metas. Elegir el entorno de desarrollo Matlab, suponía una ventaja en la comparación entre diferentes algoritmos, al existir inmensas referencias y permitir una simulación fácil y cómoda frente a otros entornos. Además, ante los cambios realizados en los algoritmos para poder realizar las diferentes comparaciones, resultaba mucho más inmediato la elección de Matlab, debido a la experiencia anterior en el entorno.

- Se ha realizado una comparativa, que si bien no ha sido todo lo extensa que se hubiera deseado por limitaciones de tiempo y recursos computacionales, sí ha servido para poder evaluar diferentes algoritmos SLAM que existen actualmente frente a situaciones características que se pueden encontrar en entornos reales. Ha servido para determinar de forma cualitativa el algoritmo que se consideraba mejor, teniendo en cuenta los errores que se pudieran producir en las ejecuciones, posibles variaciones en los parámetros que los configuran y evaluando la eficiencia de cada uno de ellos.

7.1. Trabajo futuro

Como trabajo futuro se ha pensado en las siguientes líneas:

- Ampliar el alcance del análisis, cubriendo una mayor variedad de algoritmos y escenarios.
- Utilizar recursos computacionales de mayor capacidad para permitir la realizacion de pruebas más exhaustivas.
- Desarrollar una interfáz gráfica para facilitar la realización de las diferentes comparativas.
- Implementación optimizada de los algoritmos para su ejecución sobre robots reales.
- Ensayo sobre escenarios reales.

Apéndice A

Anexo I: Formulación

A.1. Formulación del Estado Aumentado

El mapa de la estructura del estado aumentado del vehículo permite al filtro de Kalman mantener una medida de todas las correlaciones que surgen entre los errores en sus estimaciones.

El mapa del modelo aumentado del vehículo $f_{x_{mav}}$ se deriva a partir del modelo del vehículo f_{x_v}, en la ecuación 3.20, y de la suposición de que las características están fijadas en un marco de referencia global,

$$\begin{aligned}
x_v(k+1) &= f_{x_v}[x_v(k), u(k), v(k), k] \\
&\approx f_{x_v}[\hat{x}_v(k|k), u(k), 0, k] + \nabla(f_{x_v})_{x_v} \tilde{x}_v(k|k) + \nabla(f_{x_v})_v v(k) \\
p_i(k+1) &= p_i(k) \quad i = 1, 2, \ldots, N
\end{aligned}$$

$$\begin{aligned}
x_{mav}(k+1) &= f_{x_{mav}}[x_{mav}(k), u(k), v(k), k] \\
&= \left[x_v^T(k+1) \;\; p_1^T(k+1) \;\; \ldots \;\; p_N^T(k+1)\right]^T \\
&= \begin{bmatrix} f_{x_v}[x_v(k), u(k), v(k), k] \\ p_1(k) \\ \vdots \\ p_N(k) \end{bmatrix} \\
&\approx \begin{bmatrix} f_{x_v}[\hat{x}_v(k|k), u(k), 0, k] + \nabla(f_{x_v})_{x_v} \tilde{x}_v(k|k) + \nabla(f_{x_v})_v v(k) \\ p_1(k) \\ \vdots \\ p_N(k) \end{bmatrix}
\end{aligned} \tag{A.1}$$

La predicción del mapa de la estimación del estado aumentado del vehículo $\hat{x}_{mav}(k+1|k)$ se deriva como sigue,

$$\begin{aligned}
\hat{x}_{mav}(k+1|k) &= E\left[x_{mav}(k+1)|Z^k\right] \\
&\approx \begin{bmatrix} f_{x_v}\left[\hat{x}_v(k|k), u(k), 0, k\right] \\ \hat{p}_1(k|k) \\ \vdots \\ \hat{p}_N(k|k) \end{bmatrix} \\
&= f_{x_{mav}}\left[\hat{x}_{mav}(k|k), u(k), 0, k\right]
\end{aligned} \tag{A.2}$$

$$\begin{aligned}
\tilde{x}_{mav}(k+1|k) &= x_{mav}(k+1) - \hat{x}_{mav}(k+1|k) \\
&= \begin{bmatrix} \nabla\left(f_{x_v}\right)_{x_v} \tilde{x}_v(k|k) + \nabla\left(f_{x_v}\right)_v v(k) \\ \tilde{p}_1(k) \\ \vdots \\ \tilde{p}_N(k) \end{bmatrix},
\end{aligned}$$

del cual sigue que

$$\begin{aligned}
&P_{mav}(k|k+1) = E\left[\tilde{x}_{mav}(k+1|k)\,\tilde{x}^T_{mav}(k+1|k)\right] \\
&= \begin{bmatrix} \nabla\left(f_{x_v}\right)_{x_v} P_{vv}(k|k) \nabla\left(f_{x_v}\right)^T_{x_v} & \nabla\left(f_{x_v}\right)_{x_v} P_{v1}(k|k) & \dots & \nabla\left(f_{x_v}\right)_{x_v} P_{vN}(k|k) \\ P_{1v}(k|k) \nabla\left(f_{x_v}\right)^T_{x_v} & P_{11}(k|k) & \dots & P_{1N}(k|k) \\ \vdots & \vdots & \ddots & \vdots \\ P_{Nv}(k|k) \nabla\left(f_{x_v}\right)^T_{x_v} & P_{N1}(k|k) & \dots & P_{NN}(k|k) \end{bmatrix}, \\
&\quad + \begin{bmatrix} \nabla\left(f_{x_v}\right)_v Q(k) \nabla\left(f_{x_v}\right)^T_v & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix} \\
&= \nabla\left(f_{x_{mav}}\right)_{x_{mav}} P_{mav}(k|k) \nabla\left(f_{x_{mav}}\right)^T_{x_{mav}} + \nabla\left(f_{x_{mav}}\right)_v Q(k) \nabla\left(f_{x_{mav}}\right)_v,
\end{aligned} \tag{A.3}$$

donde

$$\begin{aligned}
\nabla\left(f_{x_{mav}}\right)_{x_{mav}} &= \begin{bmatrix} \nabla\left(f_{x_v}\right)_{x_v} & 0 & 0 & \dots \\ 0 & I & 0 & \dots \\ 0 & 0 & I & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \\
\nabla\left(f_{x_{mav}}\right)_v &= \begin{bmatrix} \nabla\left(f_{x_v}\right)^T_v & 0 & \dots \end{bmatrix}^T.
\end{aligned}$$

Por lo tanto, se puede obtener una etapa de predicción coherente para el mapa del estado aumentado del vehículo.

El modelo de observación también debe justificar el aumento del vector de estado, y por lo tanto, se define un nuevo modelo. Sin embargo, el nuevo modelo de observación sólo representa un cambio de notación, en tanto que $h_{x_{mav},i}$ debe describir la observación de una sola característica como se describe mediante $h_{x_v,p}$, pero se debe expresar en términos del mapa del estado aumentado del vehículo x_{mav}. Lo siguiente es que

$$\begin{aligned} z_{x_{mav},i}(k) &= z_{x_v,p_i}(k) \\ h_{x_{mav},i}[x_{mav}(k), i, w(k), k] &= h_{x_v,p}[x_v(k), p_i(k), w(k), k], \end{aligned} \tag{A.4}$$

y también

$$\begin{aligned} \nabla(h_{x_{mav},i})_{x_v} &= \nabla(h_{x_v,p})_{x_v} \\ \nabla(h_{x_{mav},i})_{p_i} &= \nabla(h_{x_v,p})_{p_i} \\ \nabla(h_{x_{mav},i})_{p_j} &= 0;\ para\ j \neq i \\ \nabla(h_{x_{mav},i})_{w} &= \nabla(h_{x_v,p})_{w} \end{aligned} \tag{A.5}$$

Usando las identidades en las ecuaciones A.5, el Jacobiano del modelo aumentado de observaciones, con respecto al estado aumentado característico del vehículo es:

$$\begin{aligned} \nabla(h_{x_{mav},i})_{x_{mav}} &= \left[\nabla(h_{x_{mav},i})_{x_v} \quad \nabla(h_{x_{mav},i})_{p_1} \quad \dots \quad \nabla(h_{x_{mav},i})_{p_N} \right] \\ &= \left[\nabla(h_{x_v,p})_{x_v} \quad 0 \quad \dots \quad 0 \quad \nabla(h_{x_v,p})_{p_i} \quad 0 \quad \dots \right] \end{aligned} \tag{A.6}$$

Una simplificación de la notación hace más fácil el posterior análisis, de la siguiente forma:

$$\begin{aligned} H_i &\cong \nabla(h_{x_{mav},i})_{x_{mav}} \\ H_v &\cong \nabla(h_{x_{mav},i})_{x_v} \\ H_p &\cong \nabla(h_{x_{mav},i})_{p_i} \\ H_w &\cong \nabla(h_{x_{mav},i})_{w}. \end{aligned}$$

Por ejemplo, el Jacobiano $\nabla(h_{x_{mav},i})_{x_{mav}}$ de las observaciones de las características p_1, p_2 y p_3 son, respectivamente[1],

[1]Si el modelo de observación es lineal, la estructura de $H_i(k)$ se preserva aún. Sin embargo, los términos $H_v(k)$ y $H_p(k)$ se obtienen directamente mediante el modelo de observación.

$$
\begin{aligned}
H_1(k) &= \begin{bmatrix} H_v(k) & H_p(k) & 0 & 0 & 0 & \dots \end{bmatrix} \\
H_2(k) &= \begin{bmatrix} H_v(k) & 0 & H_p(k) & 0 & 0 & \dots \end{bmatrix} \\
H_3(k) &= \begin{bmatrix} H_v(k) & 0 & 0 & H_p(k) & 0 & \dots \end{bmatrix}
\end{aligned}
$$

A.2. Formulación de las Correlaciones

A.2.1. Importancia de las Correlaciones

El filtro MAK asegura que la covarianza de la innovación es calculada correctamente, manteniendo la covarianza entre el vehículo y todas las estimaciones de característica, y también manteniendo la covarianza entre todos los pares de estimaciones de las características.

En la formulación del filtro MAK, la matriz ponderada de Kalman se expresa como

$$
\begin{aligned}
W_i(k+1) &= P_{mav}(k+1|k)\, H_i^T(k+1)\, S_i^{-1}(k+1|k) \\
&= \begin{bmatrix} P_{vv}H_v^T + P_{vi}H_p^T \\ \vdots \\ P_{iv}H_v^T + P_{ii}H_p^T \\ \vdots \\ P_{jv}H_v^T + P_{ji}H_p^T \\ \vdots \\ P_{Nv}H_v^T + P_{Ni} \end{bmatrix} (k+1|k)\, S_i^{-1}(k+1|k) \\
&= \begin{bmatrix} P_{vv}(k+1|k)\,\nabla\,(h_{x_v,p})_{x_v}^T + P_{vi}(k+1|k)\,\nabla\,(h_{x_v,p})_{p_i}^T \\ \vdots \\ P_{iv}(k+1|k)\,\nabla\,(h_{x_v,p})_{x_v}^T + P_{ii}(k+1|k)\,\nabla\,(h_{x_v,p})_{p_i}^T \\ \vdots \\ P_{jv}(k+1|k)\,\nabla\,(h_{x_v,p})_{x_v}^T + P_{ji}(k+1|k)\,\nabla\,(h_{x_v,p})_{p_i}^T \\ \vdots \\ P_{Nv}(k+1|k)\,\nabla\,(h_{x_v,p})_{x_v}^T + P_{Ni}(k+1|k)\,\nabla\,(h_{x_v,p})_{p_i}^T \end{bmatrix} S_{yy}^{-1}(k+1|k)\,.
\end{aligned}
\tag{A.7}
$$

De la expresión para la innovación dada en ecuación 2.32 y la utilización de la suposición

$$E\left[\tilde{p}_j\left(k+1|k\right)w^T\left(k+1\right)\right]=0,$$

se deduce que

$$\begin{aligned}
S_{p_jy}\left(k+1|k\right) &= E\left[\tilde{p}_j\left(k+1|k\right)y^T\left(k+1\right)\right] \\
&= E\left[\tilde{p}_j\left(k+1|k\right)\nabla\left(h_{x_v,p}\right)_{x_v}\tilde{x}_v\left(k+1|k\right)\right] \\
&\quad +E\left[\tilde{p}_j\left(k+1|k\right)\nabla\left(h_{x_v,p}\right)_{p_i}\tilde{p}_i\left(k+1|k\right)\right] \\
&\quad +E\left[\tilde{p}_j\left(k+1|k\right)\nabla\left(h_{x_v,p}\right)_{w}w\left(k+1\right)\right] \\
&= P_{jv}\left(k+1|k\right)\nabla\left(h_{x_v,p}\right)_{x_v}^T+P_{ji}\left(k+1|k\right)\nabla\left(h_{x_v,p}\right)_{p_i}^T.
\end{aligned}$$

Por lo tanto, combinando con las ecuaciones 2.34 y 2.35, obtenemos $W_i\left(k+1\right)$ como:

$$W_i\left(k+1\right) = \begin{bmatrix} S_{x_vy}\left(k+1|k\right)S_{yy}^{-1}\left(k+1|k\right) \\ \vdots \\ S_{p_iy}\left(k+1|k\right)S_{yy}^{-1}\left(k+1|k\right) \\ \vdots \\ S_{p_jy}\left(k+1|k\right)S_{yy}^{-1}\left(k+1|k\right) \\ \vdots \\ S_{p_Ny}\left(k+1|k\right)S_{yy}^{-1}\left(k+1|k\right) \end{bmatrix}. \tag{A.8}$$

Las submatrices $S_{x_vy}\left(k+1|k\right)S_{yy}^{-1}\left(k+1|k\right)$ y $S_{p_iy}\left(k+1|k\right)S_{yy}^{-1}\left(k+1|k\right)$ de $W_i\left(k+1\right)$ son reconocidas como las matrices ponderadas de Kalman para la actualización del vehículo y la estimación de la característica observada.

La submatriz $S_{p_jy}\left(k+1|k\right)S_{yy}^{-1}\left(k+1|k\right)$ de $W_i\left(k+1\right)$ demuestra que el filtro MAK puede además, actualizar la estimación de las características no observadas, debido a la correlación entre la innovación y el error en la estimación de la característica no observada p_j, representada mediante el término $S_{p_jy}\left(k+1|k\right)$. La submatriz $S_{p_jy}\left(k+1|k\right)S_{yy}^{-1}\left(k+1|k\right)$, que consiste en el producto de $S_{p_jy}\left(k+1|k\right)$ (la covarianza de la estimación y la innovación) y la inversa de $S_{yy}\left(k+1|k\right)$ (la covarianza de la innovación), se encuentra en la forma de la matriz ponderada de Kalman, confirmando además la optimalidad del filtro MAK.

De la ecuación A.8,

$$
\begin{aligned}
W_i\left(k+1\right) &= \begin{bmatrix} E\left[\tilde{x}_v\left(k+1|k\right)y^T\left(k+1\right)\right] \\ \vdots \\ E\left[\tilde{p}_i\left(k+1|k\right)y^T\left(k+1\right)\right] \\ \vdots \\ E\left[\tilde{p}_j\left(k+1|k\right)y^T\left(k+1\right)\right] \\ \vdots \\ E\left[\tilde{p}_N\left(k+1|k\right)y^T\left(k+1\right)\right] \end{bmatrix} S_{yy}^{-1}\left(k+1|k\right) \\
&= E\left[\tilde{x}_{mav}\left(k+1|k\right)y^T\left(k+1\right)\right] S_{yy}^{-1}\left(k+1|k\right) \\
&= S_{x_{mav}y}\left(k+1|k\right) S_{yy}^{-1}\left(k+1|k\right) \qquad \text{(A.9)}
\end{aligned}
$$

Por lo tanto, $W_i\left(k+1\right)$ satisface la condición para una actualización coherente de $\hat{x}_{mav}\left(k+1|k\right)$ y $P_{mav}\left(k+1|k\right)$ según las ecuaciones A.10 y A.11,

$$
\begin{aligned}
\hat{x}_{mav}\left(k+1|k+1\right) &= \hat{x}_{mav}\left(k+1|k\right) + W_i\left(k+1\right)y\left(k+1\right) \qquad \text{(A.10)} \\
P_{mav}\left(k+1|k+1\right) &= P_{mav}\left(k+1|k\right) \\
&\quad -W_i\left(k+1\right)S_i^T\left(k+1|k\right)W_i^T\left(k+1\right). \qquad \text{(A.11)}
\end{aligned}
$$

Los resultados en esta sección se pueden interpretar de la siguiente forma. La matriz ponderada se elije para minimizar el error cuadrático medio esperado. Esto conduce a una actualización coherente y óptima de la estimación del estado $\hat{x}\left(k+1|k\right)$. Sin embargo, la estimación del estado $\hat{x}_{mav}\left(k+1|k\right)$ en el filtro MAK incluye todas las estimaciones del estado característico $\hat{p}_i\left(k+1|k\right)$, $i = 1, 2, \ldots, N$, y la matriz de covarianza $P_{mav}\left(k+1|k\right)$ se asume para satisfacer $P_{mav}\left(k+1|k\right) = E\left[\tilde{x}_{mav}\left(k+1|k\right)\tilde{x}_{mav}^T\left(k+1|k\right)\right]$. Por lo tanto, una actualización coherente y óptima del vehículo, y cada estimación de la característica, se computa mediante el filtro de Kalman, de acuerdo a las ecuaciones A.10 y A.11, si $W_i\left(k+1\right)$ satisface la ecuación A.9.

El filtro MAK asegura que la estimación del vehículo y las estimaciones de las características son actualizadas coherentemente.

El filtro MAK conduce con cada observación, a la actualización óptima del vehículo y todas las estimaciones de características.

A.2.2. Procedencia de las Correlaciones

Esta sección se examina cómo las correlaciones surgen entre los errores de estimación en el filtro MAK. Complementa la sección anterior (sección A.2.1)

mostrando que los términos en P_{mav}, identificado en el apartado 4.2.1 como esencial para la solución del problema del SLAM, se deben asumir como no nulos.

El filtro MAK supone al principio el estado estimado $\hat{x}_{mav}$ y la matriz de covarianza P_{mav} como:

$$\hat{x}_{mav}(0|0) = \begin{bmatrix} \hat{x}_v \\ \hat{p}_1 \\ \hat{p}_2 \end{bmatrix}(0|0) \tag{A.12}$$

$$P_{mav}(0|0) = \begin{bmatrix} P_{vv} & 0 & 0 \\ 0 & P_{11} & 0 \\ 0 & 0 & P_{22} \end{bmatrix}(0|0), \tag{A.13}$$

por lo tanto, las covarianzas entre todas las estimaciones se asumen inicialmente nulas. Se consideran dos realimentaciones consecutivas. La primera realimentación sigue una observación de p_1 y la segunda realimentación sigue una observación de p_2. Por lo tanto,

$$H_1(1) \cong \begin{bmatrix} H_v & H_p & 0 \end{bmatrix}(1) \tag{A.14}$$

$$H_1(2) \cong \begin{bmatrix} H_v & 0 & H_p \end{bmatrix}(2) \tag{A.15}$$

Examinando la ecuación A.3, se puede observar que la predicción $P_{mav}(1|0)$ no cambia la estructura de $P_{mav}(0|0)$, vista en la ecuación A.13, por lo tanto $P_{mav}(1|0)$ podría redefinirse como $P_{mav}(1|0) = P_{mav}(0|0)$.

La primera realimentación se calcula usando

$$\begin{aligned} S_1(1|0) &= H_1(1) P_{mav}(1|0) H_1^T(1) + H_w(1) R(1) H_w^T(1) \\ W_1(1) &= P_{mav}(1|0) H_1^T(1) S_1^{-1}(1|0) \\ \hat{x}_{mav}(1|1) &= \hat{x}_{mav}(1|0) + W_1(1) y(1) \\ P_{mav}(1|1) &= P_{mav}(1|0) - W_1(1) S_1(1|0) W_1^T(1), \end{aligned}$$

donde $y(1)$ es la innovación en el instante $k = 1$. La matriz ponderada $W_1(1)$ del filtro MAK es:

$$
\begin{aligned}
W_1(1) &= \begin{bmatrix} S_{x_v y} S_{yy}^{-1} \\ S_{p_1 y} S_{yy}^{-1} \\ S_{p_2 y} S_{yy}^{-1} \end{bmatrix} (1|0) \\
&= \begin{bmatrix} P_{vv} H_v^T + P_{v1} H_p^T \\ P_{1v} H_v^T + P_{11} H_p^T \\ P_{2v} H_v^T + P_{21} H_p^T \end{bmatrix} (1|0) S_1^{-1} (1|0) \\
&= \begin{bmatrix} P_{vv} H_v^T \\ P_{11} H_p^T \\ 0 \end{bmatrix} (1|0) S_1^{-1} (1|0), \qquad \text{(A.16)}
\end{aligned}
$$

y la actualización de la estimación del estado se escribe de forma explícita como

$$
\begin{aligned}
\hat{x}_v(1|1) &= \hat{x}_v(1|0) + P_{vv}(1|0) H_v^T(1) y(1) \qquad \text{(A.17)} \\
\hat{p}_1(1|1) &= \hat{p}_1(1|0) + P_{11}(1|0) H_p^T(1) y(1) \qquad \text{(A.18)} \\
\hat{p}_2(1|1) &= \hat{p}_2(1|0) \qquad \text{(A.19)}
\end{aligned}
$$

Puesto que $H_1(1)$ representa una observación relativa de la característica p_1 con respecto al vehículo, $H_v(1)$ y $H_p(1)$ son no nulos. Esto correlaciona la innovación $y(1)$ con el error $\tilde{x}_v(1|0)$ en $\hat{x}_v(1|0)$, y también con el error $\tilde{p}_1(1|0)$ en $\hat{p}_1(1|0)$. El término no nulo $P_{vv}(1|0) H_v^T(1) = S_{x_v y}(1|0)$ indica la correlación entre $\tilde{x}_v(1|0)$ e $y(1)$, y el término no nulo $P_{11}(1|0) H_p^T(1) = S_{p_1 y}(1|0)$ indica la correlación entre $\tilde{p}_1(1|0)$ e $y(1)$. Estas correlaciones permiten al filtro MAK actualizar la estimación $\hat{x}_v(1|0)$ del estado del vehículo, ecuación A.17, y la estimación $\hat{p}_1(1|0)$ del estado de la característica p_1, ecuación A.18.

Si $H_1(1)$ consta sólo de $H_p(1)$, realmente haciendo $H_v(1)$ nulo y correspondiente a una observación directa de la característica p_1, la innovación sólo estaría correlacionada con el error $\tilde{p}_1(1|0)$, como ocurre en $P_{vv}(1|0) H_v^T(1) = S_{x_v y}(1|0) = 0$. Por lo tanto, sólo $\hat{p}_1(1|0)$ sería actualizado y $\hat{x}_v(1|1) = \hat{x}_v(1|0)$. Del mismo modo, si $H_1(1)$ consta sólo de $H_v(1)$, realmente haciendo $H_p(1)$ nulo y correspondiente a una observación directa del vehículo, la innovación sólo estaría correlacionada con el error $\tilde{x}_v(1|0)$, como ocurre en $P_{11}(1|0) H_p^T(1) = S_{p_1 y}(1|0) = 0$. Por lo tanto, sólo $\hat{x}_v(1|0)$ sería actualizado y $\hat{p}_1(1|1) = \hat{p}_1(1|0)$.

Escribiendo la actualización de la covarianza explícitamente como:

$$\begin{aligned} P_{vv}(1|1) &= P_{vv}(1|0) - \left[W_{x_v} S_1^{-1} W_{x_v}\right](1|0) \\ &= P_{vv}(1|0) - \left[P_{vv} H_v^T S_1^{-1} H_v P_{vv}\right](1|0) \end{aligned} \tag{A.20}$$

$$\begin{aligned} P_{v1}(1|1) &= P_{v1}(1|0) - \left[W_{x_v} S_1^{-1} W_{p_1}\right](1|0) \\ &= -\left[P_{vv} H_v^T S_1^{-1} H_p P_{11}\right](1|0) \end{aligned} \tag{A.21}$$

$$\begin{aligned} P_{v2}(1|1) &= P_{v2}(1|0) - \left[W_{x_v} S_1^{-1} W_{p_2}\right](1|0) \\ &= 0 \end{aligned} \tag{A.22}$$

$$\begin{aligned} P_{11}(1|1) &= P_{11}(1|0) - \left[W_{p_1} S_1^{-1} W_{p_1}\right](1|0) \\ &= P_{11}(1|0) - \left[P_{11} H_p^T S_1^{-1} H_p P_{11}\right](1|0) \end{aligned} \tag{A.23}$$

$$\begin{aligned} P_{12}(1|1) &= P_{12}(1|0) - \left[W_{p_1} S_1^{-1} W_{p_2}\right](1|0) \\ &= 0 \end{aligned} \tag{A.24}$$

$$\begin{aligned} P_{22}(1|1) &= P_{22}(1|0) - \left[W_{p_2} S_1^{-1} W_{p_2}\right](1|0) \\ &= P_{22}(1|0) \end{aligned} \tag{A.25}$$

conduce a la siguiente estructura en $P_{mav}(1|1)$,

$$P_{mav}(1|1) = \begin{bmatrix} P_{vv} & P_{v1} & 0 \\ P_{1v} & P_{11} & 0 \\ 0 & 0 & P_{22} \end{bmatrix}(1|1). \tag{A.26}$$

El error en la estimación del estado del vehículo $\hat{x}_v(1|1)$ y el error en la estimación de la característica $\hat{p}_1(1|1)$ no se correlacionan ahora, como se mostró con el término no nulo $P_{v1}(1|1)$. Esta correlación surge debido a que se utiliza una observación relativa de p_1 en la actualización. Los términos $H_v(1)$ y $H_p(1)$ aparecen en la realimentación de $P_{v1}(1|0)$, ecuación A.21, y sólo si ambos no son nulos, lo cual es una consecuencia directa de una observación relativa, siendo $P_{v1}(1|1)$ no nulo. Este resultado se puede interpretar de la siguiente forma. Si $H_v(1)$ y $H_p(1)$ son ambos no nulos, entonces, de acuerdo a las ecuaciones A.17 y A.18, la estimación del estado del vehículo $\hat{x}_v(1|0)$ y la característica p_1 de la estimación del estado $\hat{p}_1(1|0)$ son actualizados utilizando la misma observación. Esto correlaciona los errores en $\hat{x}_v(1|1)$ y

$\hat{p}_1(1|1)$, conduciendo al término no nulo $P_{v1}(1|1)$.

Si $H_1(1)$ consta sólo de $H_p(1)$, se actualizaría solamente $P_{11}(1|0)$, de acuerdo a la ecuación A.23. Todos los demás términos en $P_{mav}(1|0)$ permanecerían sin alterarse. De manera similar, si $H_1(1)$ consta sólo de $H_v(1)$, sólo $P_{vv}(1|0)$ se podría actualizar, de acuerdo a la ecuación A.20, y todos los demás términos en $P_{mav}(1|0)$, permanecerían inalterados. Esto es compatible con la realimentación de la estimación, deducida para estos dos casos.

Una realimentación de una observación relativa, correlaciona el error en la estimación del vehículo con el error en la estimación de la característica observada, debido a que la misma observación se usa para realimentar la característica observada y la estimación del vehículo.

La ecuación A.3 se usa de nuevo para concluir que la predicción $P_{mav}(2|1)$ no cambia la estructura de $P_{mav}(1|1)$, vista en la ecuación A.26. Por lo tanto, $P_{mav}(2|1)$ podría redefinirse como $P_{mav}(2|1) = P_{mav}(1|1)$.

La segunda realimentación sigue la observación de la característica p_2, y por lo tanto, se calcula usando

$$\begin{aligned}
S_2(2|1) &= H_2(2) P_{mav}(2|1) H_2^T(2) + H_w R(2) H_w^T \\
W_2(2) &= P_{mav}(2|1) H_2^T(2) S_2^{-1}(2|1) \\
\hat{x}_{mav}(2|2) &= \hat{x}_{mav}(2|1) + W_2(2) y(2) \\
P_{mav}(2|2) &= P_{mav}(2|1) - W_2(2) S_2(2|1) W_2^T(2),
\end{aligned}$$

donde $y(2)$ es la innovación en el instante $k = 2$. La matriz ponderada $W_2(2)$ del filtro MAK, para la segunda realimentación es:

$$\begin{aligned}
W_2(2) &= \begin{bmatrix} S_{x_v y} S_{yy}^{-1} \\ S_{p_1 y} S_{yy}^{-1} \\ S_{p_2 y} S_{yy}^{-1} \end{bmatrix} (2|1) \\
&= \begin{bmatrix} P_{vv} H_v^T + P_{v2} H_p^T \\ P_{1v} H_v^T + P_{12} H_p^T \\ P_{2v} H_v^T + P_{22} H_p^T \end{bmatrix} (2|1) S_2^{-1}(2|1) \\
&= \begin{bmatrix} P_{vv} H_v^T \\ P_{1v} H_v^T \\ P_{22} H_p^T \end{bmatrix} (2|1) S_2^{-1}(2|1). \qquad \text{(A.27)}
\end{aligned}$$

La actualización del segundo estado estimado se escribe explícitamente como:

$$\hat{x}_v(2|2) = \hat{x}_v(2|1) + P_{vv}(2|1) H_v^T(2) y(2) \quad \text{(A.28)}$$
$$\hat{p}_1(2|2) = \hat{p}_1(2|1) + P_{1v}(2|1) H_v^T(2) y(2) \quad \text{(A.29)}$$
$$\hat{p}_2(2|2) = \hat{p}_2(2|1) + P_{22}(2|1) H_p^T(2) y(2). \quad \text{(A.30)}$$

Como en la primera realimentación, la observación relativa correlaciona la innovación $y(2)$ con el error en la estimación del vehículo $\tilde{x}_v(2|1)$, y también con el error en la estimación de la característica observada $\tilde{p}_2(2|1)$. Sin embargo, esta vez la innovación también esta correlacionada con el error $\tilde{p}_1(2|1)$ en la estimación de la característica no observada, ya que $\tilde{p}_1(2|1)$ está correlacionado con $\tilde{x}_v(2|1)$. Por lo tanto, $S_{x_v y}(2|1)$, $S_{p_1 y}(2|1)$ y $S_{p_2 y}(2|1)$ son no nulos, y se actualizan $\hat{x}_v(2|1)$, $\hat{p}_1(2|1)$ y $\hat{p}_2(2|1)$. Si $H_2(2)$ consta sólo de $H_p(2)$, correspondiendo a una observación directa de la característica p_2, la innovación sólo estaría correlacionada con $\tilde{p}_2(2|1)$, por lo tanto, sólo $\hat{p}_2(2|1)$ podría actualizarse. Sin embargo, si $H_2(2)$ consta sólo de $H_v(2)$, correspondiente a una observación directa del vehículo, la innovación estaría correlacionada con $\tilde{x}_v(2|1)$ y también con $\tilde{p}_1(2|1)$, puesto que $\tilde{p}_1(2|1)$ está correlacionado con $\tilde{x}_v(2|1)$ debido a la primera realimentación. Por lo tanto, $\hat{x}_v(2|1)$ y $\hat{p}_1(2|1)$ podrían ser actualizados.

Escribiendo la actualización de la segunda covarianza de forma explícita,

$$
\begin{aligned}
P_{vv}\left(2|2\right) &= P_{vv}\left(2|1\right) - \left[W_{x_v} S_2^{-1} W_{x_v}\right]\left(2|1\right) \\
&= P_{vv}\left(2|1\right) - \left[P_{vv} H_v^T\right] S_2^{-1} \left[H_v P_{vv}\right]\left(2|1\right) \qquad \text{(A.31)}
\end{aligned}
$$

$$
\begin{aligned}
P_{v1}\left(2|2\right) &= P_{v1}\left(2|1\right) - \left[W_{x_v} S_1^{-1} W_{p_1}\right]\left(2|1\right) \\
&= P_{v1}\left(2|1\right) - \left[P_{vv} H_v^T\right] S_2^{-1} \left[H_v P_{v1}\right]\left(2|1\right) \qquad \text{(A.32)}
\end{aligned}
$$

$$
\begin{aligned}
P_{v2}\left(2|2\right) &= P_{v2}\left(2|1\right) - \left[W_{x_v} S_2^{-1} W_{p_2}\right]\left(2|1\right) \\
&= -\left[P_{vv} H_v^T\right] S_2^{-1} \left[H_p P_{22}\right]\left(2|1\right) \qquad \text{(A.33)}
\end{aligned}
$$

$$
\begin{aligned}
P_{11}\left(2|2\right) &= P_{11}\left(2|1\right) - \left[W_{p_1} S_2^{-1} W_{p_1}\right]\left(2|1\right) \\
&= P_{11}\left(2|1\right) - \left[P_{1v} H_v^T\right] S_1^{-1} \left[H_v P_{v1}\right]\left(2|1\right) \qquad \text{(A.34)}
\end{aligned}
$$

$$
\begin{aligned}
P_{12}\left(2|2\right) &= P_{12}\left(2|1\right) - \left[W_{p_1} S_2^{-1} W_{p_2}\right]\left(2|1\right) \\
&= -\left[P_{1v} H_v^T\right] S_2^{-1} \left[H_p P_{22}\right]\left(2|1\right) \qquad \text{(A.35)}
\end{aligned}
$$

$$
\begin{aligned}
P_{22}\left(2|2\right) &= P_{22}\left(2|1\right) - \left[W_{p_2} S_2^{-1} W_{p_2}\right]\left(2|1\right) \\
&= P_{22}\left(2|1\right) - \left[P_{22} H_p^T\right] S_2^{-1} \left[H_p P_{22}\right]\left(2|1\right), \qquad \text{(A.36)} \\
& \qquad \text{(A.37)}
\end{aligned}
$$

conduce a la siguiente estructura en $P_{mav}\left(2|2\right)$,

$$
P_{mav}\left(2|2\right) = \begin{bmatrix} P_{vv} & P_{v1} & P_{v2} \\ P_{1v} & P_{11} & P_{12} \\ P_{2v} & P_{21} & P_{22} \end{bmatrix}\left(2|2\right). \qquad \text{(A.38)}
$$

Cada término en $P_{mav}\left(2|2\right)$ es no nulo, indicando que todos los errores de estimación están ahora correlacionados. Como en la primera realimentación, la observación relativa correlaciona el error en la estimación del vehículo y la estimación de la característica observada, mostrado por el término no nulo $P_{v2}\left(2|2\right)$. De nuevo, $H_v\left(2\right)$ y $H_p\left(2\right)$ tienen que ser no nulos en la realimentación de $P_{v2}\left(2|1\right)$, ecuación A.33, para que $P_{v2}\left(2|2\right)$ no sea nulo.

Si $H_2\left(2\right)$ consta sólo de $H_p\left(2\right)$, sólo se actualizaría $P_{22}\left(2|1\right)$, de acuerdo a la ecuación A.36. Todos los demás términos en $P_{mav}\left(2|1\right)$ permanecerían inalterados. Sin embargo, Si $H_2\left(2\right)$ consta sólo de $H_v\left(2\right)$, la actualización de la covarianza no estaría limitada a $P_{vv}\left(2|1\right)$. La realimentación se extendería

a los términos $P_{11}(2|1)$, ecuación A.34, y también a $P_{v1}(2|1)$, ecuación A.32. Esto es debido a la correlación entre el error en $\hat{p}_1(2|1)$ y $\hat{x}_v(2|1)$, producido por la primera realimentación y representado por el término no nulo $P_{v1}(2|1)$, y refleja el hecho de que tanto $\hat{x}_v(2|1)$ como $\hat{p}_1(2|1)$ sean actualizados si $H_v(2)$ aparece en $H_2(2)$.

Finalmente, la observación relativa permite también un término no nulo $P_{12}(2|2)$. La realimentación de $P_{12}(2|1)$, ecuación A.35, implica los términos $H_p(2)$ y $H_v(2)$. Sólo si ambos son no nulos, correspondiente a una observación relativa, $P_{12}(2|2)$ será no nulo. Esto se puede interpretar de la siguiente manera. Cuando $H_p(2)$ y $H_v(2)$ son no nulos, la observación se usa para actualizar $\hat{p}_1(2|1)$, debido al término no nulo $P_{v1}(2|1)$, y también se usa para actualizar $\hat{p}_2(2|1)$. Debido a que la misma observación se usa para actualizar $\hat{p}_1(2|1)$ y $\hat{p}_2(2|1)$, los errores en $\hat{p}_1(2|1)$ y $\hat{p}_2(2|1)$ se correlacionarán, conduciendo al término no nulo $P_{12}(2|2)$.

La innovación de una observación relativa de una característica p_i, se correlaciona con el error de la estimación del vehículo y con el error de estimación de la característica p_i. Si el error de la estimación del vehículo se correlaciona al principio con el error de estimación de la característica p_j, entonces la innovación también será correlacionada con el error de estimación de la característica p_j. Por lo tanto, una actualización de la característica p_i con la innovación, correlacionará el error de la estimación de la característica p_i con el error de la estimación de la característica p_j. Por lo tanto, estimando el vehículo y los estados de las características, usando observaciones relativas, correlaciona necesariamente todos los errores de la estimación.

En el apartado 4.2.1 se pudo observar que cada correlación entre el vehículo y una característica, y entre dos características, se debe mantener mediante la matriz de covarianza MAK, para alcanzar una actualización constante. Puesto que estas correlaciones se han identificado en este apartado como inevitables, se llega a una importante conclusión.

No es posible solucionar el problema del SLAM para N características como N problemas de filtración de una sola característica independiente. Para solucionar el problema del SLAM, la estimación del estado del vehículo $\hat{x}_v$ tiene que ser aumentada con cada estimación del estado de la característica $\hat{p}_i$, donde $i = 1, 2, \ldots, N$. Asociado con el estado aumentado, denotado x_{mav},

está la matriz de covariancia aumentada, denotada como P_{mav}. Durante una actualización, cada término en x_{mav} y cada término en P_{mav} deben ser actualizados, si todas las estimaciones siguen siendo compatibles con sus covarianzas.

La dimensión del mapa del estado aumentado del vehículo x_{mav}, crece linealmente con el número de características. Por lo tanto, la dimensión de la matriz de covarianza correspondiente P_{mav} es del orden N^2, para un entorno de N características. Durante una actualización, cada término de P_{mav} debe ser actualizado. Esto conduce a la siguiente conclusión.

Para un entorno de N características, la solución del problema del SLAM requiere almacenamiento del orden de N^2, y cómputo de, al menos, el mismo orden.

A.3. Formulación de la Evolución del Mapa

En el apartado anterior A.2.2 se estableció que la estimación del estado del vehículo se debe aumentar con cada estimación del estado de la característica, para solucionar el problema del SLAM de una manera coherente. Una vez que se construye el vector del estado del vehículo del mapa aumentado, el filtro de Kalman se utiliza para estimar el estado de las características y del vehículo, simultáneamente.

El valor de la estructura SLAM, obtenida en el apartado 4.2, se demostró considerando la evolución del mapa. El análisis de la evolución del mapa requiere que algunas observaciones sean hechas sobre las propiedades de las matrices que se presentan en el filtro de Kalman.

Se establecen las siguientes propiedades:

$$\begin{array}{rcll} P(k|k) & = & E\left[\tilde{x}(k|k)\,\tilde{x}(k|k)^T\right] & \Rightarrow \;\; P(k|k) \;\; es\ psd. \\ Q(k) & = & E\left[v(k)\,v^T(k)\right] & \Rightarrow \;\; Q(k) \;\; es\ psd. \\ R(k) & = & E\left[w(k)\,w^T(k)\right] & \Rightarrow \;\; R(k) \;\; es\ psd. \end{array}$$

donde *psd* significa *positiva semidefinida*. También se cumple

$$\begin{aligned} P(k+1|k) &= \nabla f_x P_{mav}(k|k) \nabla f_x^T + \nabla f_v Q(k) \nabla f_v^T \\ &\Rightarrow P(k+1|k) \ es\ psd. \end{aligned}$$

$$\begin{aligned} S(k+1|k) &= \nabla h_x P(k+1|k) \nabla h_x^T + \nabla h_w R(k+1) \nabla h_w^T \\ &\Rightarrow S(k+1|k) \ es\ psd. \end{aligned}$$

$$W(k+1) S(k+1|k) W^T(k+1) \ es\ psd.$$

Por lo tanto, las matrices $P(k|k)$ y $P(k+1|k)$, y el producto $W(k+1) S(k+1|k) W^T(k+1)$ son todos positivos semidefinidos. Esto es un resultado general para el filtro de Kalman, y por lo tanto se cumple también que $P_{mav}(k|k)$ y $P_{mav}(k+1|k)$, y $W_i(k+1) S_i(k+1) W_i^T(k+1)$ son todos positivos semidefinidos.

A.3.1. Evolución de las Estimaciones de la Característica

Se considera la evolución de las secciones del mapa, los grupos de características, y en particular, la evolución de estimaciones de características individuales. La ecuación de actualización de la matriz de covarianza del filtro general de Kalman es

$$\begin{aligned} P(k+1|k+1) &= P(k+1|k) \\ &\quad -W(k+1) S(k+1|k) W^T(k+1) \end{aligned} \tag{A.39}$$

Teniendo en cuenta que el determinante de una matriz positiva semidefinida no puede disminuir cuando otra matriz positiva semidefinida se le añade, la ecuación A.39, y el hecho que $P(k+1|k+1)$, $P(k+1|k)$ y $W(k+1) S(k+1|k) W^T(k+1)$ son todos positivos semidefinidos, conduce a

$$|P(k+1|k+1)| = \left|P(k+1|k) - W(k+1) S(k+1|k) W^T(k+1)\right|$$

$$\begin{aligned} |P(k+1|k+1)| \leq &\left|P(k+1|k) - W(k+1) S(k+1|k) W^T(k+1)\right. \\ &\left.+W(k+1) S(k+1|k) W^T(k+1)\right| \end{aligned}$$

y por lo tanto

$$|P(k+1|k+1)| \leq |P(k+1|k)| \tag{A.40}$$

Este resultado expresa que el determinante de la matriz de covariancia P no puede aumentar durante una actualización.

Un resultado similar se puede obtener para ciertas submatrices de P. Ya que la suma y la resta de matrices lleva a cabo un elemento acertado, si la misma submatriz se forma en $P(k+1|k+1)$, $P(k+1|k)$ y $W(k+1)S(k+1|k)W^T(k+1)$, entonces la ecuación A.39 sigue siendo válida para todas las submatrices. Las submatrices particulares, que se considerarán como submatrices principales, se construyen mediante la supresión de cualquier número de columnas y las correspondientes filas. En general, si $[M]_p$ denota una submatriz principal de una matriz M, entonces

$$\begin{aligned}[P(k+1|k+1)]_p &= [P(k+1|k)]_p \\ &\quad - \left[W(k+1)S(k+1|k)W^T(k+1)\right]_p \end{aligned} \tag{A.41}$$

Cualquier submatriz principal de una matriz positiva semidefinida será también positiva semidefinida, por lo tanto, una ecuación similar a la ecuación A.40 se obtiene a partir de las submatrices principales de $P(k+1|k+1)$ y $P(k+1|k)$,

$$\left|[P(k+1|k+1)]_p\right| = \left|[P(k+1|k)]_p - \left[W(k+1)S(k+1|k)W^T(k+1)\right]_p\right|$$

$$\begin{aligned}\left|[P(k+1|k+1)]_p\right| \leq \Big|&[P(k+1|k)]_p - \left[W(k+1)S(k+1|k)W^T(k+1)\right]_p \\ &+ [P(k+1|k)]_p + \left[W(k+1)S(k+1|k)W^T(k+1)\right]_p\Big|,\end{aligned}$$

y por lo tanto

$$\left|[P(k+1|k+1)]_p\right| \leq \left|[P(k+1|k)]_p\right|.$$

Este resultado expresa que el determinante de cualquier submatriz principal de la matriz de covariancia P, y por lo tanto P_{mav}, no puede aumentar durante una actualización. En particular, las covarianzas de las características P_{ii} son las submatrices principales de P_{mav}, como la covarianza del vehículo P_{vv}, de manera que se deducen los dos importantes resultados siguientes,

$$\begin{aligned} |P_{vv}(k+1|k+1)| &\leq |P_{vv}(k+1|k)| \\ |P_{ii}(k+1|k+1)| &\leq |P_{ii}(k+1|k)|. \end{aligned}$$

El determinante de la matriz de covarianza de cualquier característica y el determinante de la matriz de covarianza del vehículo no aumentan durante una actualización. Formando otras matrices principales, se puede demostrar que el determinante de la matriz de covarianza de cualquier grupo de estimaciones no aumenta durante una actualización. Los grupos pueden incluir el vehículo, o pueden consistir sólo en estimaciones de características. Por ejemplo, cualquier grupo de dos estimaciones tomadas del vector de estado del vehículo del mapa aumentado, satisface

$$\left| \begin{bmatrix} P_{nn} & P_{nm} \\ P_{mn} & P_{mm} \end{bmatrix}_{cualquiera} (k+1|k+1) \right| \leq \left| \begin{bmatrix} P_{nn} & P_{nm} \\ P_{mn} & P_{mm} \end{bmatrix}_{cualquiera} (k+1|k) \right|,$$

donde P_{nm} denota la covarianza entre el bloque n y m de P_{mav}, y n y m pueden referirse al vehículo. Los índices de tiempo comunes se toman fuera de la submatriz para mejorar la transparencia.

Como la etapa de predicción no afecta a las estimaciones de las características, ecuación A.3, se cumple para las submatrices de la covarianza de la estimación de características, que

$$\begin{aligned} |P_{ii}(k+1|k)| &= |P_{ii}(k|k)| \\ \left| \begin{bmatrix} P_{nn} & P_{nm} \\ P_{mn} & P_{mm} \end{bmatrix}_{mapa} (k+1|k) \right| &= \left| \begin{bmatrix} P_{nn} & P_{nm} \\ P_{mn} & P_{mm} \end{bmatrix}_{mapa} (k|k) \right|, \end{aligned}$$

y por lo tanto

$$\begin{aligned} |P_{ii}(k+1|k+1)| &\leq |P_{ii}(k|k)| \\ \left| \begin{bmatrix} P_{nn} & P_{nm} \\ P_{mn} & P_{mm} \end{bmatrix}_{mapa} (k+1|k+1) \right| &\leq \left| \begin{bmatrix} P_{nn} & P_{nm} \\ P_{mn} & P_{mm} \end{bmatrix}_{mapa} (k|k) \right|, \end{aligned}$$

donde P_{nm} denota de nuevo la covarianza entre el bloque n y m de P_{mav}, pero n y m sólo se refieren a características. Esto conduce a otra conclusión importante.

El determinante de la matriz de covarianza de todas las estimaciones de características, y el determinante de la matriz de covarianza de cualquier grupo de características, son ambos funciones monótonas no crecientes en el tiempo. En particular, el determinante de la matriz de covarianza de cada estimación de una característica individual, es una función no creciente en el tiempo.

Las dimensiones individuales del vehículo y las estimaciones de las características se pueden examinar mediante la consideración de las entradas diagonales de P_{mav}. La entrada diagonal q-ésima de P_{mav} es la varianza de la estimación de la dimensión q-ésima del vector de estado aumentado de las características. Se puede ver en la ecuación A.39 que la actualización de la entrada diagonal q-ésima de P_{mav} se forma mediante la resta de la entrada diagonal q-ésima de $W_i S_i W_i^T$, que no debe ser negativa. La varianza en cada dimensión de la estimación de la característica del estado del vehículo aumentado, no puede aumentar durante una actualización,

$$\sigma^2_{cualquiera}\left(k+1|k+1\right) \;\leq\; \sigma^2_{cualquiera}\left(k+1|k\right),$$

donde $\sigma^2_{cualquiera}$ es la varianza en alguna dimensión de la estimación del vehículo o una característica.

Además, puesto que la etapa de predicción no afecta a las estimaciones de características, la varianza de la estimación en cada dimensión de cada característica es una función monótona no creciente del instante de tiempo,

$$\sigma^2_{mapa}\left(k+1|k+1\right) \;\leq\; \sigma^2_{mapa}\left(k|k\right),$$

donde σ^2_{mapa} es la varianza en alguna dimensión de la estimación de la característica.

El mapa producido por el filtro de Kalman, en la forma del vector de estado del vehículo del mapa aumentado, consiste en estimaciones de características con varianzas no crecientes en cada dimensión.

A.3.2. Evolución de Distancias Relativas

Habiendo establecido el comportamiento de las estimaciones de características individuales y grupos de estimaciones de características, se estudia ahora

la evolución de la estimación de distancias relativas entre cualquiera dos características en el mapa. Seleccionando dos características aleatorias p_i y p_j del mapa, la distancia relativa se puede expresar como un vector Δp_{ij},

$$\begin{aligned} \Delta p_{ij} &= \begin{bmatrix} p_{j_1} - p_{i_1} \\ \vdots \\ p_{j_n} - p_{i_n} \end{bmatrix} \\ &= \begin{bmatrix} -I & I \end{bmatrix} \begin{bmatrix} p_i \\ p_j \end{bmatrix} \end{aligned}$$

donde p_{i_j} es la dimensión j-ésima de la característica p_i. Se puede asociar una matriz de covarianza con este vector, definida del modo habitual,

$$\begin{aligned} \Delta P_{ij} &\cong E\left[(\Delta\tilde{p}_{ij})\,(\Delta\tilde{p}_{ij})^T\right] \\ &= E\left[\begin{bmatrix} (\hat{p}_{j_1} - \hat{p}_{i_1}) - (p_{j_1} - p_{i_1}) \\ \vdots \\ (\hat{p}_{j_n} - \hat{p}_{i_n}) - (p_{j_n} - p_{i_n}) \end{bmatrix} \begin{bmatrix} (\hat{p}_{j_1} - \hat{p}_{i_1}) - (p_{j_1} - p_{i_1}) \\ \vdots \\ (\hat{p}_{j_n} - \hat{p}_{i_n}) - (p_{j_n} - p_{i_n}) \end{bmatrix}^T\right] \\ &= E\left[\begin{bmatrix} (\hat{p}_{j_1} - p_{j_1}) - (\hat{p}_{i_1} - p_{i_1}) \\ \vdots \\ (\hat{p}_{j_n} - p_{j_n}) - (\hat{p}_{i_n} - p_{i_n}) \end{bmatrix} \begin{bmatrix} (\hat{p}_{j_1} - p_{j_1}) - (\hat{p}_{i_1} - p_{i_1}) \\ \vdots \\ (\hat{p}_{j_n} - p_{j_n}) - (\hat{p}_{i_n} - p_{i_n}) \end{bmatrix}^T\right] \\ &= E\left[\begin{bmatrix} \tilde{p}_{j_1} - \tilde{p}_{i_1} \\ \vdots \\ \tilde{p}_{j_n} - \tilde{p}_{i_n} \end{bmatrix} \begin{bmatrix} \tilde{p}_{j_1} - \tilde{p}_{i_1} \\ \vdots \\ \tilde{p}_{j_n} - \tilde{p}_{i_n} \end{bmatrix}^T\right] \\ &= E\left[\begin{bmatrix} -I & I \end{bmatrix} \begin{bmatrix} \tilde{p}_i \\ \tilde{p}_j \end{bmatrix} \begin{bmatrix} \tilde{p}_i \\ \tilde{p}_j \end{bmatrix}^T \begin{bmatrix} -I \\ I \end{bmatrix}\right] \\ &= \begin{bmatrix} -I & I \end{bmatrix} E\left[\begin{bmatrix} \tilde{p}_i \\ \tilde{p}_j \end{bmatrix} \begin{bmatrix} \tilde{p}_i \\ \tilde{p}_j \end{bmatrix}^T\right] \begin{bmatrix} -I \\ I \end{bmatrix} \\ &= \begin{bmatrix} -I & I \end{bmatrix} \begin{bmatrix} P_{ii} & P_{ij} \\ P_{ji} & P_{jj} \end{bmatrix} \begin{bmatrix} -I \\ I \end{bmatrix}, \end{aligned} \tag{A.42}$$

y por lo tanto

$$|\Delta P_{ij}| = \left|\begin{bmatrix} -I & I \end{bmatrix} \begin{bmatrix} P_{ii} & P_{ij} \\ P_{ji} & P_{jj} \end{bmatrix} \begin{bmatrix} -I \\ I \end{bmatrix}\right|.$$

La elección de la submatriz principal en la ecuación A.41 para incluir sólo las covarianzas de las características p_i y p_j, y denotando la matriz positiva semidefinida $\left[W_d\left(k+1\right)S_d\left(k+1|k\right)W_d^T\left(k+1\right)\right]_p$, obtenida de la observación de una característica aleatoria p_d, según M_{psd} conduce a

$$\begin{aligned}
\left[P_{mav}\right]_p &= \begin{bmatrix} P_{ii} & P_{ij} \\ P_{ji} & P_{jj} \end{bmatrix} \\
\left[P_{mav}\left(k+1|k\right)\right]_p &= \left[P_{mav}\left(k+1|k+1\right)\right]_p + M_{psd} \\
\left[P_{mav}\left(k|k\right)\right]_p &= \left[P_{mav}\left(k+1|k+1\right)\right]_p + M_{psd}.
\end{aligned} \tag{A.43}$$

La expresión final cumple además, que la predicción no afecta las estimaciones de la característica. Así

$$\begin{aligned}
\begin{bmatrix} -I & I \end{bmatrix}\left[P_{mav}\left(k|k\right)\right]_p\begin{bmatrix} -I \\ I \end{bmatrix} &= \begin{bmatrix} -I & I \end{bmatrix}\left[P_{mav}\left(k+1|k+1\right)\right]_p\begin{bmatrix} -I \\ I \end{bmatrix} \\
&\quad + \begin{bmatrix} -I & I \end{bmatrix}M_{psd}\begin{bmatrix} -I \\ I \end{bmatrix}.
\end{aligned}$$

Combinando con la ecuación A.42, se establece el resultado siguiente, donde M'_{psd} es positiva semidefinida,

$$\begin{aligned}
\left|\Delta P_{ij}\left(k|k\right)\right| &= \left|\Delta P_{ij}\left(k+1|k+1\right) + M'_{psd}\right| \\
&\geq \left|\Delta P_{ij}\left(k+1|k+1\right)\right| \\
\Rightarrow \left|\Delta P_{ij}\left(k+1|k+1\right)\right| &\leq \left|\Delta P_{ij}\left(k|k\right)\right|
\end{aligned} \tag{A.44}$$

Por lo tanto, se alcanza la siguiente conclusión.

Los determinantes de las matrices de covarianza de los vectores de distancia relativa entre todos los pares posibles de características, son funciones monótonas no crecientes en el tiempo.

De manera similar al estudio de dimensiones individuales descritas en el apartado A.3.1, es posible analizar las dimensiones individuales de una distancia relativa. La distancia escalar relativa, en una dimensión particular m, entre las características p_i y p_j se denotará $\Delta_m p_{ij}$,

$$\begin{aligned}\Delta_m p_{ij} &= p_{j_m} - p_{i_m} \\ &= \begin{bmatrix} \dots & 0 & -1 & 0 & \dots & 0 & 1 & 0 & \dots \end{bmatrix} \begin{bmatrix} pi \\ p_j \end{bmatrix} \\ &= u_m^T \begin{bmatrix} pi \\ p_j \end{bmatrix}.\end{aligned}$$

Indicando la varianza de $\Delta_m p_{ij}$ mediante $\Delta_m P_{ij}$ y siguiendo la derivación de la ecuación A.42, conduce a

$$\Delta_m P_{ij} = u_m^T \begin{bmatrix} P_{ii} & P_{ij} \\ P_{ji} & P_{jj} \end{bmatrix} u_m. \tag{A.45}$$

Combinando las ecuaciones A.43 y A.45 se demuestra que

$$\begin{aligned} u_m^T \left[P_{mav}\left(k|k\right)\right]_p u_m &= u_m^T \left[P_{mav}\left(k+1|k+1\right)\right]_p u_m + u_m^T M_{psd} u_m \\ \Delta_m P_{ij}\left(k|k\right) &= \Delta_m P_{ij}\left(k+1|k+1\right) + u_m^T M_{psd} u_m \end{aligned}$$

Puesto que $u_m^T M_{psd} u_m \geq 0$, obtenemos que

$$\Delta_m P_{ij}\left(k+1|k+1\right) \leq \Delta_m P_{ij}\left(k|k\right)$$

La varianza en cada dimensión de la estimación del vector de distancias relativas entre cualquier par de características, es una función monótona no creciente en el tiempo.

Los resultados obtenidos para distancias relativas, también se aplican a todas las otras combinaciones lineales, como la posición media de un número aleatorio de estimaciones de posiciones de características.

Apéndice B

Anexo II: Definición de las funciones

B.1. Ekf-slam

Aquí se detallan las funciones que integran el algoritmo Ekf-slam. En las figuras B.1 y B.2 se pueden observar las dependencias existentes entre ellas.

- `function varargout = frontend(varargin)`

Interfaz gráfica de usuario (GUI) para la realización del entorno del Ekf-slam.

Este programa permite la creación y manipulación gráfica de la indicación de hitos en un entorno, y la especificación de puntos de referencia de la trayectoria del vehículo dentro de el.

USO: Escribir "frontend" para comenzar.

1. Pulsar en la operación deseada: <enter> (insertar), <move> (mover), o <delete> (eliminar).
2. Pulsar en el tipo: <waypoint> (punto de referencia) o <landmark> (hito) para comenzar la operación.
3. Si se incorporan nuevos hitos o puntos de referencia, pulsar con el botón izquierdo del ratón para añadir nuevos puntos. Pulsar el botón derecho del ratón, o pulsar <enter> para terminar.
4. Para mover o suprimir un punto, basta con pulsar cerca del punto deseado.

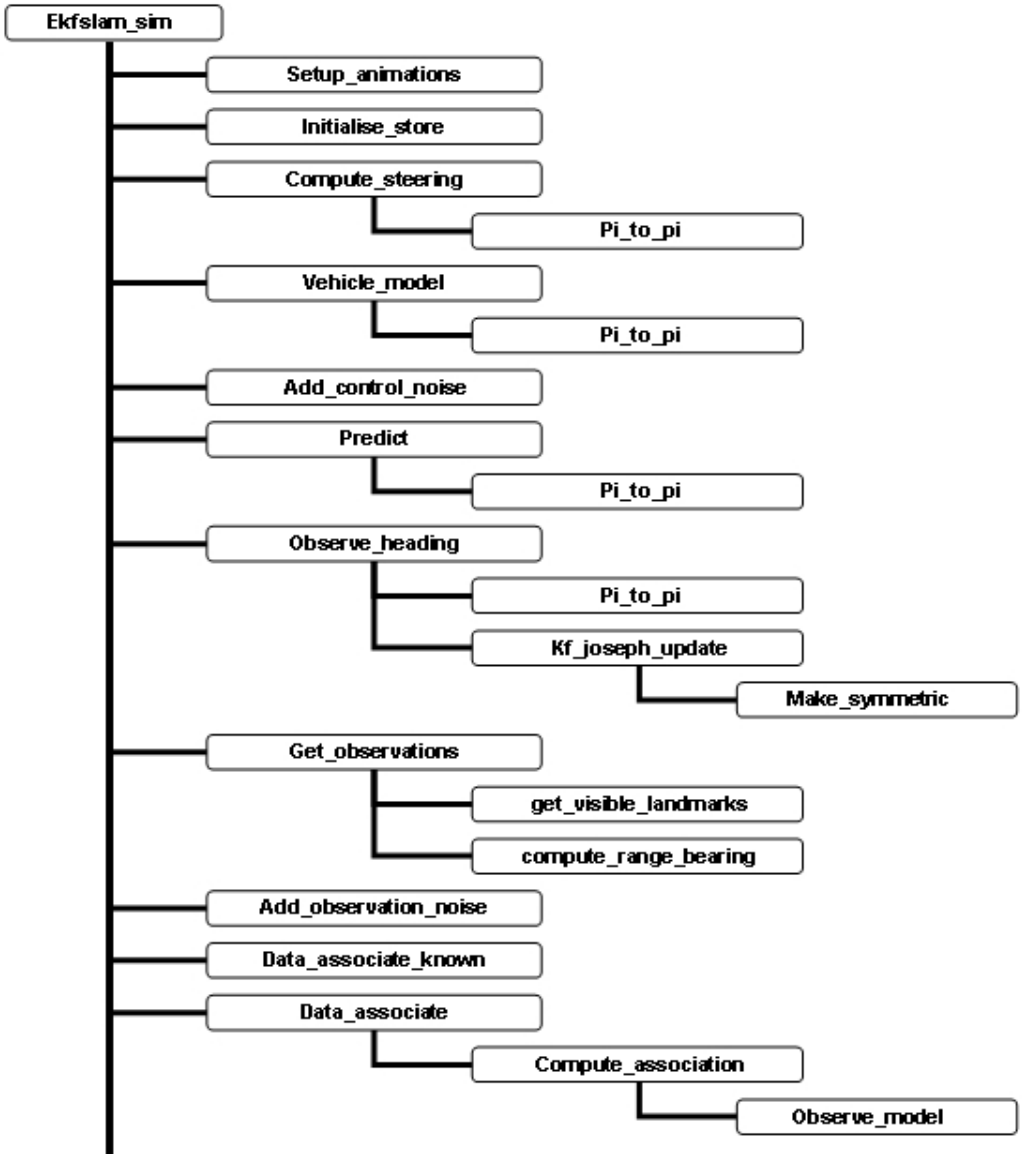


Figura B.1: Dependencia de funciones para el algoritmo Ekf-slam.

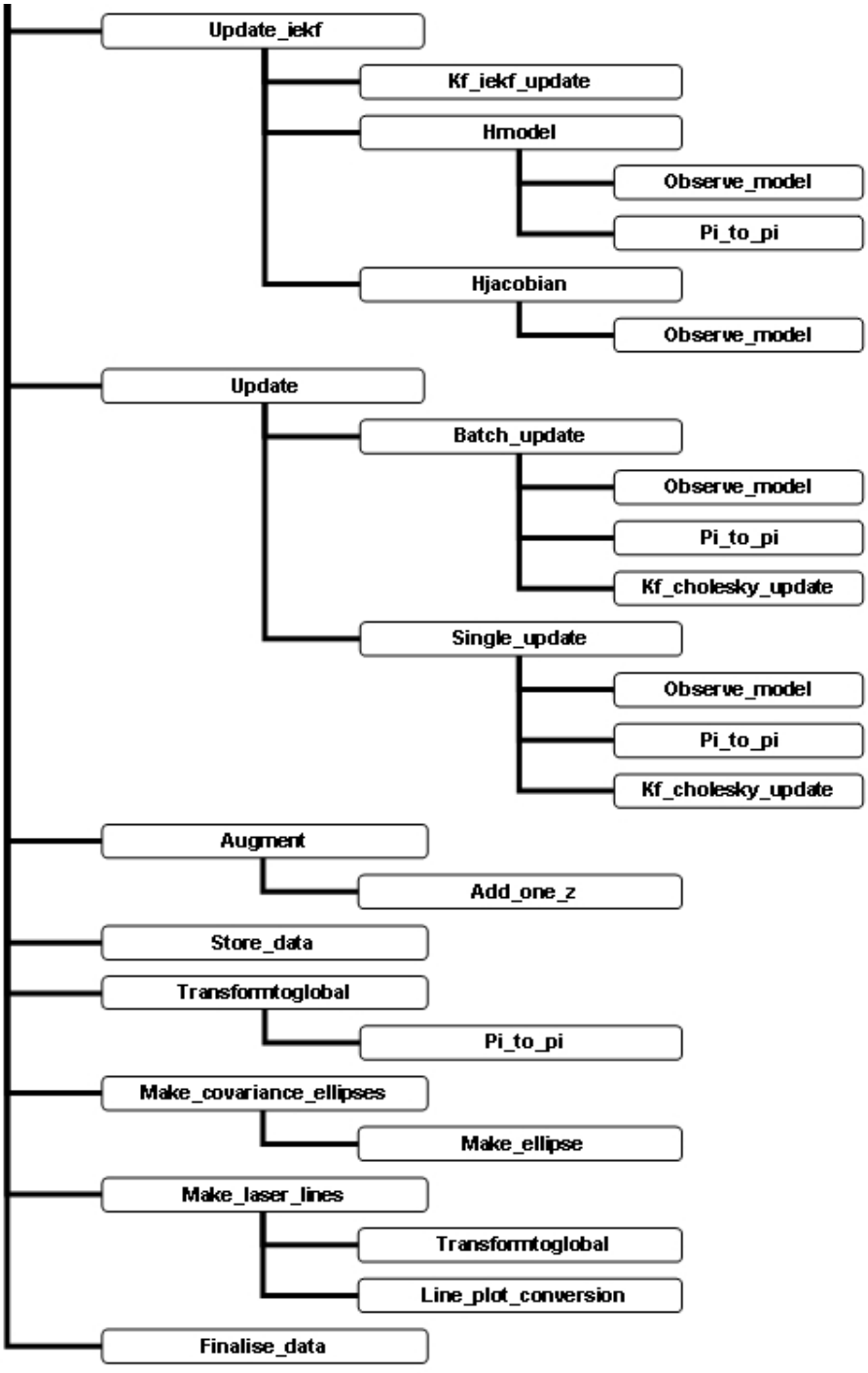


Figura B.2: Continuación de la dependencia de funciones para el algoritmo Ekf-slam.

5. Guardar mapas y cargar mapas anteriores se realiza mediante los botones <save> (guardar) y <load> (cargar), respectivamente.

- `function data = ekfslam_sim(lm,wp)`

ENTRADAS:

lm - conjunto de hitos.

wp - conjunto de puntos de referencia.

SALIDAS:

data - estructura de datos que contiene:

- data.true: la trayectoria "verdadera" del vehículo (por ejemplo, a donde fue "realmente" el vehículo).
- data.path: la estimación de la trayectoria del vehículo (por ejemplo, donde estima el SLAM que fue el vehículo).
- data.state(k).x: el vector de estado del SLAM en el instante k.
- data.state(k).P: las diagonales de la matriz de covarianza del SLAM en el instante k.

NOTAS: Este programa es un simulador del SLAM. Para utilizarlo, crear un conjunto de hitos y puntos de referencia del vehículo (por ejemplo, puntos de referencia para la trayectoria deseada del vehículo). El programa "frontend.m" se puede utilizar para crear este ambiente simulado.

La configuración del simulador es manejada por el fichero de script "configfile.m". Para cambiar los parámetros del vehículo, de los sensores, etc..., hay que modificar este fichero. También hay varios switches que controlan ciertas opciones.

- `function h = setup_animations()`

Configura la forma en que se van a dibujar todos los posibles objetos de la simulación.

- `function data = initialise_store(x,P,xtrue)`

Inicializa la estructura *data*, almacenando en ella los parámetros de forma offline.

- `function [x,P] = KF_joseph_update(x,P,v,R,H)`

Este módulo es idéntico al KF_simple_update() salvo que utiliza la actualización de la covarianza de la forma Joseph[1].

- `function P= make_symmetric(P)`

Calcula la matriz simétrica de aquella dada como parámetro.

- `function [lm,idf] = get_visible_landmarks(x,lm,idf,rmax)`

Selecciona un conjunto de hitos que son visibles dentro del campo de visión semicircular del vehículo.

- `function z = compute_range_bearing(x,lm)`

Calcula la observación exacta.

- `function [V,G] = add_control_noise(V,G,Q,addnoise)`

Añade ruido de forma aleatoria a los valores nominales del control. Se asume que Q es diagonal.

[1]La actualización de la covarianza de la forma Joseph es la primera expresión de la ecuación de P_k^+ [Si06].

$$\begin{aligned} P_k^+ &= (I - K_k H_k)\, P_k^- \,(I - K_k H_k)^T + K_k R_k K_k^T \\ &= \left[\left(P_k^-\right)^{-1} + H_k^T R_k^{-1} H_k\right]^{-1} \\ &= (I - K_k H_k)\, P_k^- \end{aligned}$$

Dicha expresión garantiza que P_k^+ será siempre simétrica definida positiva, siempre que P_k^- sea simétrica definida positiva.

La primera expresión es más estable y robusta que la tercera expresión para P_k^+. La tercera expresión es más simple computacionalmente que la primera expresión, pero esta forma no garantiza simetría o definición positiva para P_k^+.

- `function z = add_observation_noise(z,R,addnoise)`

Añade ruido de medida de forma aleatoria. Se asume que R es diagonal.

- `function [x,P] = augment(x,P,z,R)`

ENTRADAS:

x,P - estado del SLAM y covarianza.

z,R - medidas alineación-dirección y covarianzas, cada uno de una nueva característica.

SALIDA:

x,P - estado aumentado y covarianza.

NOTAS:

Asumimos que el número de estados de la ubicación del vehículo es tres.

Solamente se utiliza un valor para R, aunque todas las medidas se asumen que tienen las mismas propiedades de ruido.

Añade las nuevas características al estado.

- `function [x,P,table] = augment_associate_known(x,P,z,R,idz,table)`

Añade los nuevos índices de las características a la tabla de consulta de la asociación de datos y además añade las características al estado.

- `function p = TransformToGlobal(p,b)`

Transforma una lista de ubicaciones [x;y;phi] de modo que sean global con respecto a la ubicación base.

- `function [xn,Pn] = predict (x,P,v,g,Q,WB,dt)`

ENTRADAS:

x,P - estado del SLAM y covarianza.

v,g - entradas de control: velocidad y gamma (ángulo de dirección).

Q - matriz de covarianza para la velocidad y gamma.

WB - distancia entre ejes del vehículo.

dt - instante de tiempo.

SALIDAS:

xn,Pn - estado predicho y covarianza.

Predice el estado y la covarianza.

- `function plot_feature_loci(data)`

A partir de los datos almacenados fuera de línea del simulador, traza trayectorias de las estimaciones de la característica en un cierto tiempo.

- `function angle = pi_to_pi(angle)`

ENTRADA:

angle - vector de ángulos.

- `function [z,H] = observe_model(x,idf)`

ENTRADAS:

x - vector de estado.

idf - índice de la característica ordenado en el estado.

SALIDAS:

z - observación prevista.

H - Jacobiano de la observación.

Dado un índice de la característica (por ejemplo, la ordenación de la característica en el vector de estado), predice la observación alineación-dirección prevista de esta característica y de su Jacobiano.

- `function [x,P] = observe_heading(x,P,phi,useheading)`

Representa la actualización del estado para una medida dada como parámetro, phi, con ruido de medida fijo, sigmaPhi.

- `function [nis,nd] = compute_association(x,P,z,R,idf)`

Devuelve el cuadrado de la innovación normalizada (es decir, la distancia de Mahalanobis) y la distancia normalizada.

- `function p = line_plot_conversion(lne)`

ENTRADA:

lne - lista de líneas [x1; y1; x2; y2].

SALIDA:

p - lista de puntos [x; y].

Convierte una lista de líneas de modo que sean trazadas como un conjunto de líneas no relacionadas, pero requiere solamente un manejador para hacerlo de esta manera. Esto se realiza convirtiendo las líneas en un conjunto de puntos, donde un punto NaN se inserta entre cada par de puntos:

$$l = \begin{matrix}[x1a & x1b & x1c; \\ y1a & y1b & y1c; \\ y2a & y2b & y2c];\end{matrix}$$

se convierte

$$p = \begin{matrix}[x1a & x2a & NaN & x1b & x2b & NaN & x1c & x2c; \\ y1a & y2a & NaN & y1b & y2b & NaN & y1c & y2c]\end{matrix}$$

- `function [x,P] = KF_simple_update(x,P,v,R,H)`

Calcula la actualización del KF (o EKF) dado el estado previo [x, P], la innovación [v, R] y el modelo de observación (linearizado) H. El resultado se calcula usando una inversión simple de S, y es menos estable numéricamente que la factorización de Cholesky[2] basada en la actualización.

[2]La factorización de Cholesky indica que cualquier matriz cuadrada A con pivotes no nulos puede ser escrita como el producto de una matriz triangular inferior L y una matriz triangular superior U; esto se llama la factorización LU [Wi06]. Sin embargo, si A es simétrica y definida positiva, se pueden escoger los factores tales que U es la traspuesta de

- `function [x,P] = KF_IEKF_update(x,P,z,R,hfun,hjac,N)`

ENTRADAS:

x - $x(k|k-1)$ - estado previsto.

P - $P(k|k-1)$ - covarianza prevista.

z - observación.

R - incertidumbre de la observación.

hfun - función para computar la innovación, dado el modelo de observación no lineal: $v = hfun(x, z)$;

hjac - función para computar el modelo de observación jacobiano: $H = hjac(x)$;

N - número de iteraciones de la actualización de IEKF.

L, y esto se llama la descomposición o factorización de Cholesky.

En general, si A es Hermitiana (ver definición a continuación) y definida positiva, entonces A puede ser descompuesta como $A = LL^*$, donde L es una matriz triangular inferior con entradas diagonales estrictamente positivas, y L^* representa la conjugada traspuesta de L. Esta es la descomposición de Cholesky.

La descomposición de Cholesky es única: dada una matriz Hermitiana positiva definida A, hay una única matriz triangular inferior L con entradas diagonales estrictamente positivas tales que $A = LL^*$. El recíproco se tiene trivialmente: si A se puede escribir como LL^* para alguna matriz invertible L, triangular inferior o no, entonces A es Hermitiana y definida positiva.

El requerimento de que L tenga entradas diagonales estrictamente positivas puede extenderse para el caso de la descomposición en el caso de ser semidefinida positiva. La proposición se lee ahora: una matriz cuadrada A tiene una descomposición de Cholesky si y sólo si A es Hermitiana y semidefinida positiva. Las factorizaciones de Cholesky para matrices semidefinidas positivas no son únicas en general.

Una matriz hermitiana (o hermítica) es una matriz cuadrada de elementos complejos que tiene la característica de ser igual a su propia traspuesta conjugada [Wi06]. Es decir, el elemento en la i-ésima fila y j-ésima columna es igual al conjugado del elemento en la j-ésima fila e i-ésima columna, para todos los índices i y j:

$$a_{ij} = \overline{a_{ji}}$$

o, escrita con la traspuesta conjugada A^*:

$$A = A^*$$

SALIDAS:

x - $x(k|k)$ - estado siguiente.

P - $P(k|k)$ - covarianza siguiente.

Esta implementación es bastante ineficaz para el SLAM, pues implica la inversa de P (por ejemplo, la actualización es $O(N^3)$ para n hitos).

- `function [x,P] = add_one_z(x,P,z,R)`

Añade una observación al estado previsto del vehículo y su covarianza.

- `function [x,P] = KF_cholesky_update(x,P,v,R,H)`

Calcula la actualización del KF (o EKF) dado el estado previo [x,P], la innovación [v,R] y el modelo de observación (linearizado) H. El resultado se calcula usando la factorización de Cholesky, la cual es numéricamente más estable que una implementación sencilla.

- `function [z,idf] = get_observations(x,lm,idf,rmax)`

ENTRADAS:

x - ubicación del vehículo [x;y;phi].

lm - conjunto de todos los hitos.

idf - etiquetas del índice para cada hito.

rmax - rango máximo del sensor de rango de dirección.

SALIDAS:

z - conjunto de observaciones de rango de dirección.

idf - etiqueta del índice del hito para cada observación.

Selecciona un conjunto de hitos, los cuales son visibles dentro del campo de visión semicircular del robot.

- `function [zf,idf,zn,table] = data_associate_known(x,z,idz,table)`

Para las simulaciones con asociaciones de datos conocidas, esta función mantiene una tabla de consulta de característica/observación. Devuelve la tabla actualizada, el conjunto de observaciones asociadas y el conjunto de observaciones para las nuevas características.

- `function [zf,idf,zn] = data_associate(x,P,z,R,gate1,gate2)`

Obtiene de forma sencilla la asociación de datos del vecino más cercano.

- `function data= store_data(data,x,P,xtrue)`

Añade los datos actuales en la estructura "data" para el almacenamiento offline.

- `function p = make_covariance_ellipses(x,P)`

Calcula las elipses para dibujar las covarianzas del estado.

- `function p = make_ellipse(x,P,s,phi)`

Obtiene una sola elipse en 2D de s-sigmas sobre intervalos de ángulo phi.

- `function p= make_laser_lines(rb,xv)`

Calcula un conjunto de segmentos de línea para las mediciones de rango láser relacionadas.

- `function data = finalise_data(data)`

Finaliza el almacenamiento offline.

- `function [G,iwp] = compute_steering(x,wp,iwp,minD,G,rateG,maxG,dt)`

ENTRADAS:

x - posición verdadera.

wp - puntos de referencia.

iwp - índice al punto de referencia actual.

minD - distancia mínima al punto de referencia actual antes de cambiar al siguiente.

G - ángulo de dirección actual.

rateG - ratio de dirección máximo (rad/s).

maxG - ángulo de dirección máximo (rad).

dt - instante de tiempo.

SALIDAS:

G - nuevo ángulo de dirección actual.

iwp - nuevo punto de referencia actual.

Determina si se alcanzó el punto de referencia actual y calcula el cambio en el ángulo de dirección actual (G) para señalar hacia el punto de referencia actual.

- `function xv = vehicle_model(xv,V,G,WB,dt)`

ENTRADAS:

xv - ubicación del vehículo [x; y; phi].

V - velocidad.

G - ángulo de dirección (gamma).

WB - distancia entre ejes.

dt - instante de tiempo.

SALIDA:

xv - nueva ubicación del vehículo.

- `function [x,P] = update_iekf(x,P,z,R,idf,N)`

ENTRADAS:

x,P - estado del SLAM y covarianza.

z,R - medidas alineación-dirección y covarianzas.

idf - índice de la característica para cada z.

N - número de iteraciones del IEKF.

SALIDA:

x,P - estado actualizado y covarianza.

Actualiza el estado y la covarianza.

- `function [x,P] = update(x,P,z,R,idf,batch)`

ENTRADAS:

x,P - estado del SLAM y covarianza.

z,R - medidas alineación-dirección y covarianzas.

idf - índice de la característica para cada z.

batch - switch para especificar si se procesan las medidas juntas o secuencialmente.

SALIDA:

x,P - estado actualizado y covarianza.

Actualiza el estado y la covarianza.

B.2. Fastslam

Aquí se detallan las funciones que integran el algoritmo FastSLAM. En las figuras B.3 y B.4 se pueden observar las dependencias existentes entre ellas.

- `function varargout = frontend(varargin)`

Interfaz gráfica de usuario (GUI) para la realización del entorno del Fastslam.

Este programa permite la creación y manipulación gráfica de la indicación de hitos en un entorno, y la especificación de puntos de referencia de la trayectoria del vehículo dentro de el.

USO: Escribir “frontend” para comenzar.

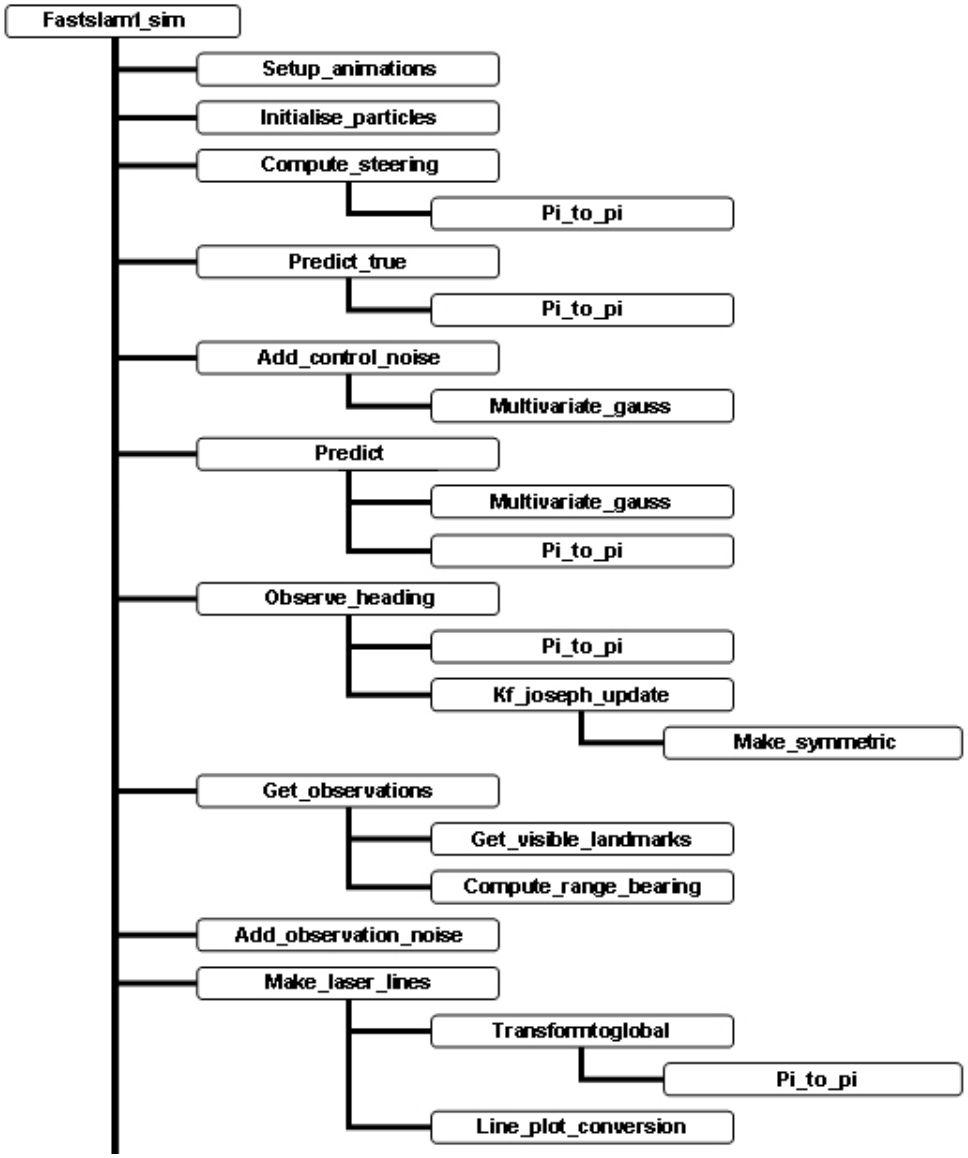


Figura B.3: Dependencia de funciones para el algoritmo Fastslam1.

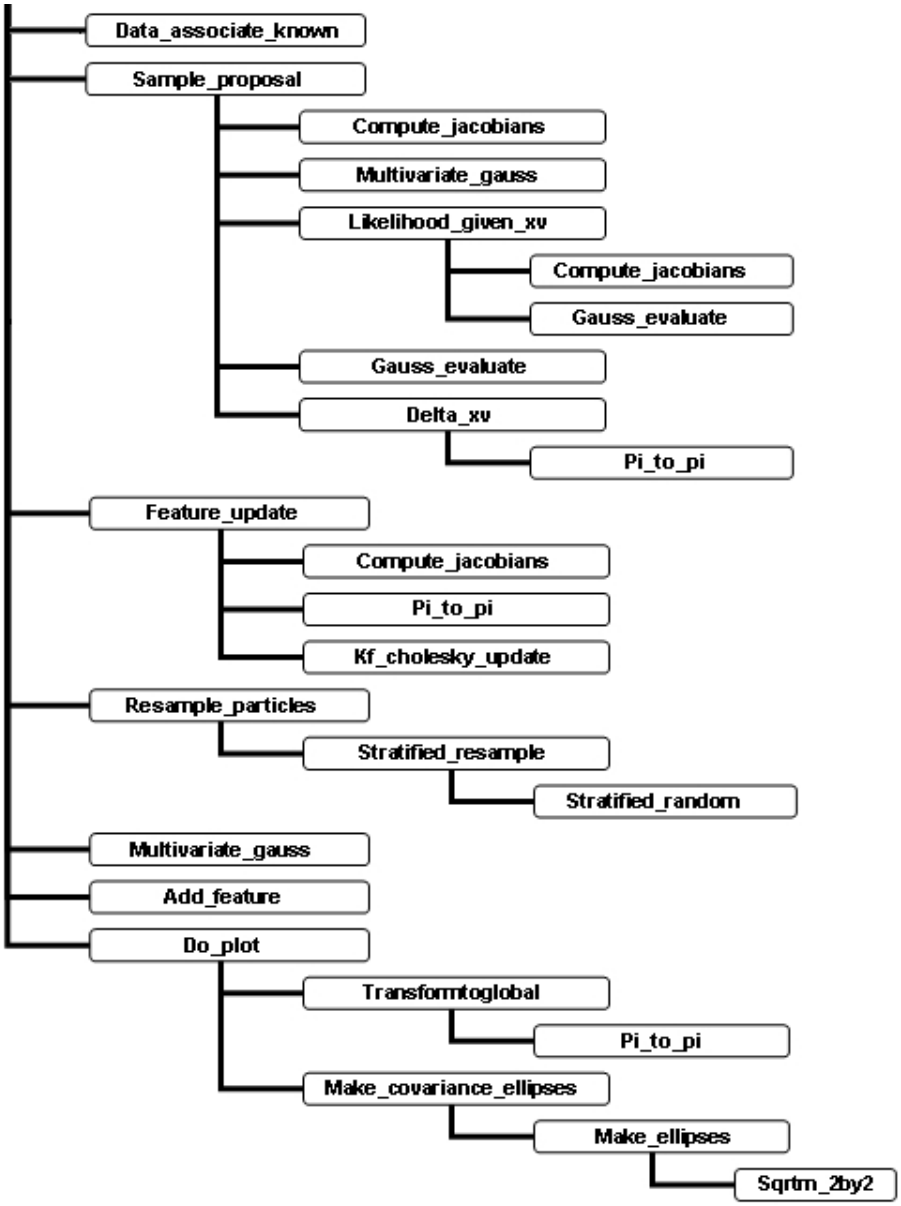


Figura B.4: Continuación de la dependencia de funciones para el algoritmo Fastslam1.

1. Pulsar en la operación deseada: <enter> (insertar), <move> (mover), o <delete> (eliminar).

2. Pulsar en el tipo: <waypoint> (punto de referencia) o <landmark> (hito) para comenzar la operación.

3. Si se incorporan nuevos hitos o puntos de referencia, pulsar con el botón izquierdo del ratón para añadir nuevos puntos. Pulsar el botón derecho del ratón, o pulsar <enter> para terminar.

4. Para mover o suprimir un punto, basta con pulsar cerca del punto deseado.

5. Guardar mapas y cargar mapas anteriores se realiza mediante los botones <save> (guardar) y <load> (cargar), respectivamente.

- `function data = fastslam1_sim(lm,wp)`

ENTRADAS:

lm - conjunto de hitos.

wp - conjunto de puntos de referencia.

SALIDA:

data - conjunto de partículas que representan en estado final.

NOTAS: Este programa es un simulador del Fastslam 1.0. Para utilizarlo, crear un conjunto de hitos y puntos de referencia del vehículo (por ejemplo, puntos de referencia para la trayectoria deseada del vehículo). El programa "frontend.m" se puede utilizar para crear este ambiente simulado.

La configuración del simulador es manejada por el fichero de script "configfile.m". Para cambiar los parámetros del vehículo, de los sensores, etc..., hay que modificar este fichero. También hay varios switches que controlan ciertas opciones.

- `function h = setup_animations()`

Configura la forma en que se van a dibujar todos los posibles objetos de la simulación.

- `function p = initialise_particles(np)`

Inicializa las propiedades de las partículas.

- `function angle = pi_to_pi(angle)`

ENTRADA:

angle - vector de ángulos.

- `function xv = predict_true(xv,V,G,WB,dt)`

Predice la ubicación del vehículo en cada momento.

- `function P= make_symmetric(P)`

Calcula la matriz simétrica de aquella dada como parámetro.

- `function [V,G] = add_control_noise(V,G,Q,addnoise)`

Añade ruido aleatorio a los valores de control nominal.

- `function s = multivariate_gauss(x,P,n)`

Muestra aleatoria de la distribución gaussiana multivariada.

- `function particle = observe_heading(particle,phi,useheading)`

(DESCRIPCION).

- `function p = TransformToGlobal(p,b)`

Transforma una lista de ubicaciones [x;y;phi] de modo que sean global con respecto a la ubicación base.

- `function [keep,Neff] = stratified_resample(w)`

ENTRADA:

w - conjunto de N pesos [w1, w2, ..., wN].

SALIDAS:

keep - N índices de partículas para guardar.

Neff - número de partículas eficaces (medida de la varianza del peso).

- `function s = stratified_random(N)`

Genera N números aleatoriamente uniformes estratificados dentro del intervalo (0,1). El conjunto de muestras, s, está en orden creciente.

- `function X = sqrtm_2by2(A)`

Raíz cuadrada de la matriz.

La función devuelve la raíz cuadrada principal de la matriz A, es decir X*X = A.

X es la raíz cuadrada única para la cual cada valor propio tiene una parte real no negativa. Si A tiene cualesquiera valores propios con parte real negativa, entonces se produce un resultado complejo. Si A es singular, entonces A puede no tener una raíz cuadrada.

- `function particles = resample_particles(particles,Nmin,doresample)`

Vuelve a tomar muestras de las partículas si su varianza del peso es tal que las N particulas logradas es menor que Nmin.

- `function s = multivariate_gauss(x,P,n)`

ENTRADA:

(x,P) - vector principal y matriz de covarianza de las que se obtienen las n muestras.

SALIDA:

s - conjunto de muestra.

Muestra escogida al azar de la distribución gausiana multivariante.

- `function p= line_plot_conversion(lne)`

ENTRADA:

lne - lista de líneas [x1; y1; x2; y2].

SALIDA:

p - lista de los puntos [x; y].

Convierte una lista de líneas de modo que sean trazadas como un conjunto de líneas no relacionadas, pero requieren solamente un solo manejador para hacerlo de esa manera. Esto se realiza convirtiendo las líneas a un conjunto de puntos, donde un punto NaN se inserta entre cada par de puntos:

$$l = \begin{bmatrix} x1a & x1b & x1c; \\ y1a & y1b & y1c; \\ y2a & y2b & y2c \end{bmatrix};$$

se convierte

$$p = \begin{bmatrix} x1a & x2a & NaN & x1b & x2b & NaN & x1c & x2c; \\ y1a & y2a & NaN & y1b & y2b & NaN & y1c & y2c \end{bmatrix}$$

- `function [x,P]= KF_joseph_update(x,P,v,R,H)`

Este módulo es idéntico al KF_simple_update() salvo que utiliza la actualización de la covarianza de la forma Joseph[3].

- `function [x,P]= KF_cholesky_update(x,P,v,R,H)`

[3]La actualización de la covarianza de la forma Joseph es la primera expresión de la ecuación de P_k^+ [Si06].

$$\begin{aligned} P_k^+ &= (I - K_k H_k) P_k^- (I - K_k H_k)^T + K_k R_k K_k^T \\ &= \left[\left(P_k^-\right)^{-1} + H_k^T R_k^{-1} H_k \right]^{-1} \\ &= (I - K_k H_k) P_k^- \end{aligned}$$

Dicha expresión garantiza que P_k^+ será siempre simétrica definida positiva, siempre que P_k^- sea simétrica definida positiva.

La primera expresión es más estable y robusta que la tercera expresión para P_k^+. La tercera expresión es más simple computacionalmente que la primera expresión, pero esta forma no garantiza simetría o definición positiva para P_k^+.

Calcula la actualización del KF (o EKF) dado el estado anterior [x,P], la innovación [v,R] y el modelo de observación (linearizado) H.

El resultado se calcula usando la factorización de Cholesky, que es más estable numéricamente que una implementación sencilla.

- `function [z,idf] = get_observations(x,lm,idf,rmax)`

ENTRADAS:

x - ubicación del vehículo [x;y;phi].

lm - conjunto de todos los hitos.

idf - etiquetas del índice para cada hito.

rmax - rango máximo del sensor de rango de dirección.

SALIDAS:

z - conjunto de observaciones de rango de dirección.

idf - etiqueta del índice del hito para cada observación.

Selecciona un conjunto de hitos, los cuales son visibles dentro del campo de visión semicircular del robot.

- `function particle = feature_update(particle,z,idf,R)`

Seleccionando una nueva ubicación de la distribución propuesta, se asume perfecta esta ubicación, y cada actualización de las características se puede computar independientemente y sin incertidumbre de la ubicación.

- `function [x,P]= fast_to_ekf_diag(particles)`

Las partículas(i) tienen xv, Pv, xf, Pf.

Para que este código trabaje, las partículas deben ser de igual peso. También se asumen que todas las partículas poseen las mismas características (en el mismo orden).

- `function [G,iwp] = compute_steering(x,wp,iwp,minD,G,rateG,maxG,dt)`

ENTRADAS:

x - posición verdadera.

wp - puntos de referencia.

iwp - índice al punto de referencia actual.

minD - distancia mínima al punto de referencia actual antes de cambiar al siguiente.

G - ángulo de dirección actual.

rateG - ratio de dirección máximo (rad/s).

maxG - ángulo de dirección máximo (rad).

dt - instante de tiempo.

SALIDAS:

G - nuevo ángulo de dirección actual.

iwp - nuevo punto de referencia actual.

Determina si se alcanzó el punto de referencia actual y calcula el cambio en el ángulo de dirección actual (G) para señalar hacia el punto de referencia actual.

- `function w = gauss_evaluate(v,S,logflag)`

ENTRADAS:

v - un conjunto de vectores de la innovación.

S - matriz de covarianza para las innovaciones.

logflag (opcional) - si se indica "1" calcula la probabilidad logarítmica, si no, calcula la probabilidad.

SALIDA:

w - conjunto de probabilidades gausianas o probabilidades logarítmicas para cada v(:,i).

Esta implementación utiliza el factor de Cholesky de S para calcular las probabilidades y de esa forma se consigue mayor estabilidad numérica que una forma sencilla de la covarianza completa. Esta función es idéntica a gauss_likelihood().

- `function particle = sample_proposal(particle,z,idf,R)`

Calcula la distribución propuesta, después toma muestras a partir de ella, y calcula el nuevo peso de la partícula.

- `function particle = predict(particle,V,G,Q,WB,dt,addrandom)`

ENTRADAS:

particle.xv - muestra de la ubicación del vehículo.

particle.Pv - Covarianza de la predicción de la ubicación del vehículo.

Predice el estado del vehículo.

NOTA: Pv se debe poner a cero después de cada observación. Acumula la incertidumbre de la ubicación del vehículo entre medidas.

- `function [lm,idf] = get_visible_landmarks(x,lm,idf,rmax)`

Selecciona un conjunto de hitos que son visibles dentro del campo de visión semicircular del vehículo.

- `function z = compute_range_bearing(x,lm)`

Calcula la observación exacta.

- `function z = add_observation_noise(z,R,addnoise)`

Añade ruido de medición de forma aleatoria. Se asume que R es diagonal.

- `function [zf,idf,zn,table] = data_associate_known(z,idz,table,Nf)`

Asocia las nuevas características a las ya existentes.

- `function [zf,idf, zn]= data_associate_f1(xv,xf,Pf,z,R, gate1, gate2)`

Obtiene de forma sencilla la asociación de datos del vecino más cercano.

- `function [nis,nd] = compute_association(x,P,z,R,idf)`

Devuelve el cuadrado de la innovación normalizada (es decir, la distancia de Mahalanobis) y la distancia normalizada.

- `function [z,H] = observe_model(x,idf)`

ENTRADAS:

x - vector de estado.

idf - índice de la característica ordenado en el estado.

SALIDAS:

z - observación prevista.

H - Jacobiano de la observación.

Dado un índice de la característica (por ejemplo, la ordenación de la característica en el vector de estado), predice la observación alineación-dirección prevista de esta característica y de su Jacobiano.

- `function [zp,Hv,Hf,Sf]= compute_jacobians(particle,idf,R)`

Calcula los jacobianos de las propiedades de cada partícula.

- `function dx = delta_xv(xv1,xv2)`

Calcula la innovación entre dos estados estimados, normalizando el componente de cabecera.

- `function particle = add_feature(particle,z,R)`

Añade nuevas características.

- `function do_plot(h,particles,xtrue,plines,veh)`

Realiza el trazado de los datos en el mapa.

- `function p = make_covariance_ellipses(x,P)`

Calcula las elipses para dibujar las covarianzas del estado.

- `function p = make_ellipse(x,P,s,phi)`

Obtiene una sola elipse en 2D de s-sigmas sobre intervalos de ángulo phi.

B.3. Ukf-slam

Aquí se detallan las funciones que integran el algoritmo Ukf-slam. En las figuras B.5 y B.6 se pueden observar las dependencias existentes entre ellas.

- `function varargout = frontend(varargin)`

Interfaz gráfica de usuario (GUI) para la realización del entorno del Ukf-slam.

Este programa permite la creación y manipulación gráfica de la indicación de hitos en un entorno, y la especificación de puntos de referencia de la trayectoria del vehículo dentro de el.

USO: Escribir “frontend” para comenzar.

1. Pulsar en la operación deseada: <enter> (insertar), <move> (mover), o <delete> (eliminar).

2. Pulsar en el tipo: <waypoint> (punto de referencia) o <landmark> (hito) para comenzar la operación.

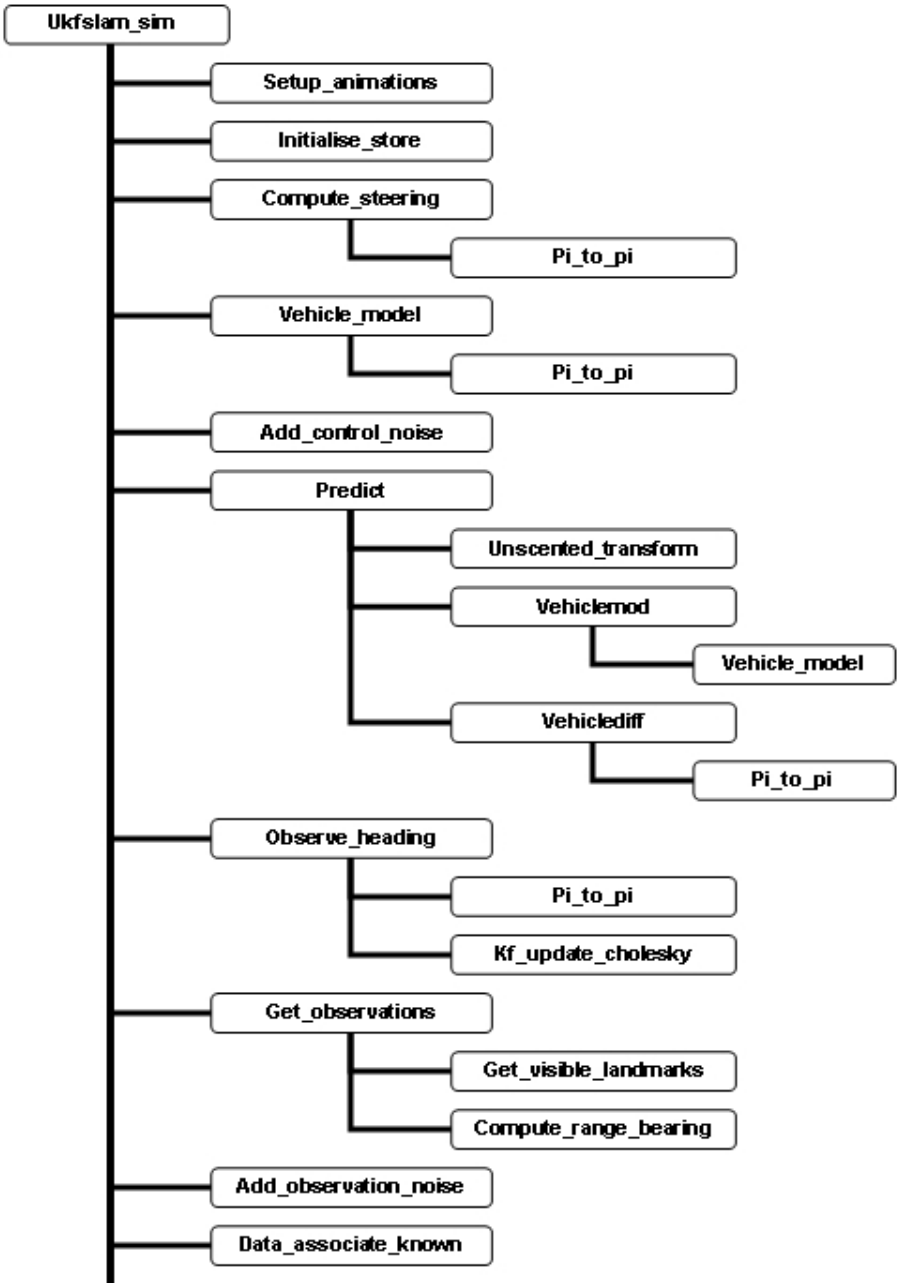


Figura B.5: Dependencia de funciones para el algoritmo Ukf-slam.

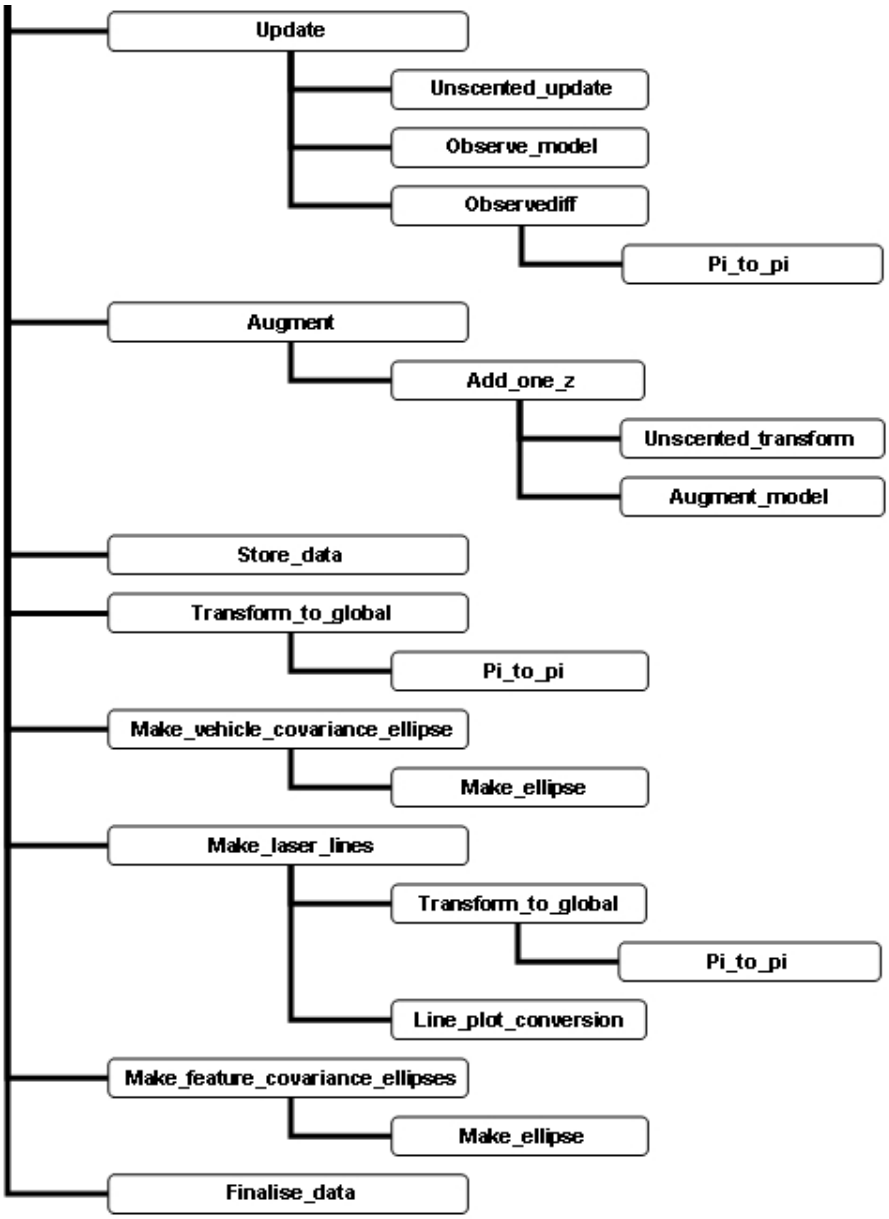


Figura B.6: Continuación de la dependencia de funciones para el algoritmo Ukf-slam.

3. Si se incorporan nuevos hitos o puntos de referencia, pulsar con el botón izquierdo del ratón para añadir nuevos puntos. Pulsar el botón derecho del ratón, o pulsar <enter> para terminar.

4. Para mover o suprimir un punto, basta con pulsar cerca del punto deseado.

5. Guardar mapas y cargar mapas anteriores se realiza mediante los botones <save> (guardar) y <load> (cargar), respectivamente.

- `function data = ukfslam_sim(lm,wp)`

ENTRADAS:

lm - conjunto de hitos.

wp - conjunto de puntos de referencia.

SALIDAS:

data - estructura de datos que contiene:

- data.true: la trayectoria "verdadera" del vehículo (por ejemplo, a donde fue "realmente" el vehículo).
- data.path: la estimación de la trayectoria del vehículo (por ejemplo, donde estima el SLAM que fue el vehículo).
- data.state(k).x: el vector de estado del SLAM en el instante k.
- data.state(k).P: las diagonales de la matriz de covarianza del SLAM en el instante k.

NOTAS: Este programa es un simulador del SLAM. Para utilizarlo, crear un conjunto de hitos y puntos de referencia del vehículo (por ejemplo, puntos de referencia para la trayectoria deseada del vehículo). El programa "frontend.m" se puede utilizar para crear este ambiente simulado.

La configuración del simulador es manejada por el fichero de script "configfile.m". Para cambiar los parámetros del vehículo, de los sensores, etc..., hay que modificar este fichero. También hay varios switches que controlan ciertas opciones.

- `function h = setup_animations()`

Configura la forma en que se van a dibujar todos los posibles objetos de la simulación.

- `function data = initialise_store(x,P,xtrue)`

Configura la forma en que se van a dibujar todos los posibles objetos de la simulación.

Inicialización del almacenamiento de forma offline.

- `function angle = pi_to_pi(angle)`

ENTRADA:

angle - vector de ángulos.

- `function observe_heading(phi,useheading)`

Realiza la actualización del estado para una medida dada como parámetro, phi, con ruido de medida fijo, sigmaPhi.

- `function [z,idf] = get_observations(x,lm,idf,rmax)`

ENTRADAS:

x - ubicación del vehículo [x;y;phi].

lm - conjunto de todos los hitos.

idf - etiquetas del índice para cada hito.

rmax - rango máximo del sensor de rango de dirección

SALIDAS:

z - conjunto de observaciones de rango de dirección.

idf - etiqueta del índice del hito para cada observación.

Selecciona un conjunto de hitos, los cuales son visibles dentro del campo de visión semicircular del robot.

- `function [zf,idf,zn,table] = data_associate_known(x,z,idz,table)`

Para las simulaciones con asociaciones de datos conocidas, esta función mantiene una tabla de consulta de característica/observación. Devuelve la tabla actualizada, el conjunto de observaciones asociadas y el conjunto de observaciones para las nuevas características.

- `function [zf,idf,zn] = data_associate(x,P,z,R,gate1,gate2)`

Obtiene de forma sencilla la asociación de datos del vecino más cercano.

- `function [nis,nd] = compute_association(x,P,z,R,idf)`

Devuelve el cuadrado de la innovación normalizada (es decir, la distancia de Mahalanobis) y la distancia normalizada.

- `function [z,H] = observe_model(x,idf)`

ENTRADAS:

x - vector de estado.

idf - índice de la característica ordenado en el estado.

SALIDAS:

z - observación prevista.

H - Jacobiano de la observación.

Dado un índice de la característica (por ejemplo, la ordenación de la característica en el vector de estado), predice la observación alineación-dirección prevista de esta característica y de su Jacobiano.

- `function [G,iwp]= compute_steering(x,wp,iwp,minD,G,rateG,maxG,dt)`

ENTRADAS:

x - posición verdadera.

wp - puntos de referencia.

iwp - índice al punto de referencia actual.

minD - distancia mínima al punto de referencia actual antes de cambiar al siguiente.

G - ángulo de dirección actual.

rateG - ratio de dirección máximo (rad/s).

maxG - ángulo de dirección máximo (rad).

dt - instante de tiempo.

SALIDAS:

G - nuevo ángulo de dirección actual.

iwp - nuevo punto de referencia actual.

Determina si se alcanzó el punto de referencia actual y calcula el cambio en el ángulo de dirección actual (G) para señalar hacia el punto de referencia actual.

- `function z = add_observation_noise(z,R,addnoise)`

Añade ruido de medida aleatorio. Asumimos que R es diagonal.

- `function [V,G]= add_control_noise(V,G,Q,addnoise)`

Añade ruido de forma aleatoria a los valores nominales del control. Se asume que Q es diagonal.

- `function xv = vehicle_model(xv,V,G,WB,dt)`

ENTRADAS:

xv - ubicación del vehículo [x; y; phi].

V - velocidad.

G - ángulo de dirección (gamma).

WB - distancia entre ejes.

dt - instante de tiempo.

SALIDA:

xv - nueva ubicación del vehículo.

- `function predict(v,g,Q,WB,dt)`

Predice el estado del vehículo.

- `function [lm,idf] = get_visible_landmarks(x,lm,idf,rmax)`

Selecciona un conjunto de hitos que son visibles dentro del campo de visión semicircular del vehículo.

- `function z = compute_range_bearing(x,lm)`

Calcula la observación exacta.

- `function p = make_laser_lines(rb,xv)`

Calcula un conjunto de segmentos de línea para las mediciones de rango láser relacionadas.

- `function p = make_feature_covariance_ellipses(x,P)`

Calcula elipses para trazar las covarianzas de las características.

- `function p = make_ellipse(x,P,circ)`

Obtiene una elipse en 2D.

- `function p = make_vehicle_covariance_ellipse(x,P)`

Calcula elipses para trazar las covarianzas del vehículo.

- `function data = finalise_data(data)`

Finaliza el almacenamiento offline.

- `function [x,P] = KF_update_cholesky(x,P,v,R,H)`

Calcula la actualización del KF (o EKF) dado el estado previo [x,P], la innovación [v,R] y el modelo de observación (linearizado) H. El resultado se calcula usando la factorización de Cholesky, la cual es numéricamente más estable que una implementación sencilla.

- `function augment(z,R)`

Aumenta el mapa a traves de las observaciones y la covarianza.

- `function data= store_data(x,P,xtrue)`

Añade los datos actuales en el almacenamiento offline.

Apéndice C

Glosario

C.1. Definiciones

AGV	Vehículo Guiado Autónomo. (*Autonomous Guided Vehicle.*)
CML	Construcción de Mapas y Localización Concurrente. (*Concurrent Mapping and Localization.*)
CPE	Estimación Coherente de la Ubicación. (*Consistent Pose Estimation*)
EKF	Filtro de Kalman Extendido. (*Extended Kalman Filter.*)
GPS	Sistema de Posicionamiento Global. (*Global Positioning System.*)
ICP	Posición Iterativa más Cercana. (*Iterative Closest Position.*)
IMU	Unidad de Medida Inercial. (*Inertial Measurement Unit.*)
INS	Sistema de Navegación Inercial. (*Inertial Navigation System.*)
LRGC	Registro Local y Correlación Global. (*Local Registration and Global Correlation.*)
MAK	Kalman del Mapa Aumentado. (*Map Augmented Kalman.*)
ML	Probabilidad Máxima. (*Maximum Likelihood.*)
MMWR	Radar de Ondas Milimétricas. (*Millimetrical Wide Radar.*)
PSD	Semidefinida Positiva.

	(*Positive Semidefinite.*)
SIS	Muestreo Secuencial de Importancia. (*Sequencial Importance Sampling.*)
$SLAM$	Localización y Construcción de Mapas Simultáneo. (*Simultaneous Localization and Map Building.*)
$NNSF$	Filtro Estándar del Vecino más Cercano. (*Nearest Neighbor Standard Filter.*)

Bibliografía

[Al09] Álvaro Sánchez Miralles. *Sensores y actuadores*. Escuela Técnica Superior de Ingeniería. Universidad Pontificia Comillas. Madrid, 2009.

[Ar04] Kai Oliver Arras. *The CAS robot navigation toolbox. Users guide and reference*. The Centre for Autonomous Systems. Kungliga Tekniska Högskolan. Stockholm, 2004.

[Ay89] N. Ayache y O.D. Faugeras. *Maintaining representations of the environment of a mobile robot*. IEEE Transactions on Robotics and Automation, 1989.

[Ba06] Página web de Tim Bailey. *http://www-personal.acfr.usyd.edu.au/tbailey/*. Australian Centre for Field Robotics (ACFR), University of Sydney, 2006.

[Ba95] M. Baeg, H. Hashimoto, F. Harashima y J.B. Moore. *Pose estimation of quadratic surface using surface fitting technique*. Proceedings of the IEEE Conference on Intelligent Robots and Systems, 1995.

[Be04] Fernando Berzal Galiano. *Análisis de algoritmos*. Departamento de Ciencias de la Computación e Inteligencia Artificial. Universidad de Granada, 2004.

[Be93] R.S. Bertin y T.W. Pendelton. *Using qualitative maps to direct reactive robots*. Proceedings of SPIE - The International Society for Optical Engineering, 1993.

[Ca06] Michael Calonder. *EKF SLAM vs. FastSLAM: A Comparison*. Computer Vision Group. Swiss Federal Institute of Technology. Lausanne, 2006.

[Ca99] José A. Castellanos y Juan D. Tardós. *Mobile robot localization and map building. A multisensor fusion approach*. Kluwer Academic Publishers, 1999.

[Ch85] R Chatila y Laumond J-P. *Referencing and consistant world modelling for mobile robots*. IEEE International Conference on Robotics and Automation, 1985.

[Ch86] R. Cheeseman y P. Smith. *The representation and estimation of spatial uncertainty*. International Journal of Robotics Research, 1986.

[Ch97] K.S. Chong y L. Kleeman. *Sonar based map building for a mobile robot*. IEEE International Conference on Robotics and Automation. Albuquerque, New Mexico. 1997.

[Cs97] Michael Csorba. *Simultaneous localisation and map building*. Robotics Research Group. Department of Engineering Science. University of Oxford, 1997.

[Cu89] Carles M. Cuadras. *Distancias estadísticas*. Departament d'Estadística. Universitat de Barcelona, 1989.

[De07] Julio Delgado Mangas. *Evaluación y prueba del algoritmo FastSLAM de construcción de mapas para un robot móvil*. Facultad de Informática. Universidad de Las Palmas de Gran Canaria, 2007.

[Di01] Margarita Díaz Roca y Juan Carlos Rodríguez del Pino. *Metodología y tecnología de la programación II*. Facultad de Informática. Universidad de Las Palmas de Gran Canaria, 2001.

[Dk09] Página web del Departamento de Ingeniería Eléctrica de la Universidad Técnica de Dinamarca. *http://www.elektro.dtu.dk*. Department of Electrical Engineering. Technical University of Denmark, 2009.

[Du00] Gregory Dudek y Michael Jenkin. *Computational principles of mobile robotics*. Cambridge University Press, 2000.

[Du88] H.F. Durrant-Whyte. *Uncertain geometry in robotics*. IEEE Transactions on Robotics and Automation, 1988.

[El90] A. Elfes. *Sonar-based real-world mapping and navigation*. Springer-Verlag, 1990.

[Ga01] M. W. M. Gamini Dissanayake, Paul Newman, Steven Clark, Hugh F. Durrant-Whyte y Michael Csorba. *A solution to the simultaneous localization and map building (SLAM) problem*. IEEE Transactions on Robotics and Automation, 2001.

[Ge03] Atanas Georgiev. *Design, implementation and localization of a mobile robot for urban site modeling*. Columbia University, 2003.

[Go93] N. Gordon, D. Salmond and A. Smith. *Novel approach to nonlinear/non-Gaussian Bayesian state estimation*. IEEE International Conference on Robotics and Automation, 1993.

[Gu00] Rosa Guerequeta y Antonio Vallecillo. *Técnicas de Diseño de Algoritmos*. Departamento de Lenguajes y Ciencias de la Computación. Universidad de Málaga, 2000.

[Gu99] Jens-Steffen Gutmann y Kurt Konolige. *Incremental mapping of large cyclic environments*. CIRA 99. Monterey. California, 1999.

[He95] P. Hébert, S. Betgé-Brezetz y R. Chatila. *Probabilistic map learning: Necessity and difficulties*. International Workshop Reasoning with Uncertainty in Robotics. Amsterdam, The Netherlands. 1995.

[He96] P. Hébert, S. Betgé-Brezetz y R. Chatila. *Decoupling odometry and exteroceptive perception in building a global world map of a mobile robot: The use of local maps*. IEEE International Conference on Robotics and Automation. Minneapolis, Minnesota. 1996.

[Hu05] Juan F. Huete Guadix. *Análisis de eficiencia de los algoritmos*. Departamento de Ciencias de la Computación e Inteligencia Artificial. Universidad de Granada, 2005.

[Id06] Página web del Laboratorio Nacional Idaho. *http://www.inl.gov*. Idaho National Laboratory. U.S. Department of Energy's Office of Nuclear Energy, Science and Technology. 2006.

[Ja05] Víctor Manuel Jáquez Leal. *Construcción de mapas y localización simultánea con robots móviles*. Departamento de Computación. Instituto Tecnológico y de Estudios Superiores de Monterrey, 2005.

[Ju97] Simon J. Julier y Jeffrey K. Uhlmann. *A New Extension of the Kalman Filter to Nonlinear Systems*. The Robotics Research Group, Department of Engineering Science. University of Oxford, 1997.

[Ko05] Kurt Konolige. *SLAM via variable reduction from constraint maps*. IEEE International Conference on Robotics and Automation. Barcelona. España, 2005.

[Kr90] D.J. Kriegman, E. Triendl y T.O. Binford. *A mobile robot: Sensing, planning and locomotion*. Springer-Verlag. 1990.

[Ku91] B.J. Kuipers y Y.T. Byun. *A robust, qualitative approach to a spatial learning mobile robot*. IEEE Computer Science Press. 1991.

[Le91] John J. Leonard y Hugh F. Durrant-Whyte. *Simultaneous map building and localization for an autonomous mobile robot*. IEEE/RSJ International Workshop on Intelligent Robots and Systems. Osaka, Japan. 1991.

[Le92a] John J. Leonard y Hugh F. Durrant-Whyte. *Directed sonar sensing for mobile robot navigation*. Kluwer Academic Publishers, 1992.

[Le92b] John J. Leonard, Hugh F. Durrant-Whyte y I.J. Cox. *Dynamic map building for an autonomous mobile robot*. International Journal of Robotics Research, 1992.

[Lu97] F. Lu y E. Milios. *Robot pose estimation in unknown environments by matching 2d range scans*. Journal of Intelligent and Robotic Systems, 1997.

[Ma79] Peter S. Maybeck. *Stochastic models, estimation, and control*. Academic Press, 1979.

[Mo02] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. *Fast-SLAM: A factored solution to the simultaneous localization and mapping problem*. Proceedings of the AAAI National Conference on Artificial Intelligence. Edmonton, Canada. 2002.

[Mo89] P. Mountarlier y R. Chatila. *Stochastic multisensory data fusion for mobile robot location and environment modeling*. 5th International Symposium on Robotics Research. Tokio, Japan. 1989.

[Mo95] V. Morellas, J. Minners y M. Donath. *Implementation of real time spatial mapping in robot systems through self-organizing neural networks*. IEEE International Conference on Intelligent Robots and Systems. Pittsburgh, 1995.

[Mu04] Página web de Kevin Patrick Murphy. *http://people.cs.ubc.ca/ murphyk/*. Department of Computer Science. University of British Columbia, 2004.

[Ne00] Ulrich Nehmzow. *Mobile robotics: A practical introduction*. Springer-Verlag London Limited, 2000.

[Ol01] Aníbal Ollero Baturone. *Robótica: manipuladores y robots móviles*. Marcombo, 2001.

[Op07] Página web OpenSLAM. *http://www.openslam.org*. OpenSLAM.org, 2007.

[Pi01] Isabel Pita Andreu. *Tiempo de ejecución de un algoritmo*. Departamento de Sistemas Informáticos y Programación. Universidad Complutense de Madrid, 2001.

[Qi08] Song Qi y Han Jian-Da. *An adaptive UKF algorithm for the state and parameter estimations of a mobile robot*. Robotics Laboratory, Shenyang Institute of Automation. Chinese Academy of Sciences, 2008.

[Re06] Página web del Toolbox ReBEL. *http://choosh.csee.ogi.edu/rebel*. Department of Science and Engineering, Oregon Health and Science University. 2006.

[Ro04] Diego Rodríguez-Losada González. *SLAM geométrico en tiempo real para robots móviles en interiores basado en EKF*. Departamento de Automática, Ingeniería Electrónica e Informática Industrial. Escuela Técnica Superior de Ingenieros Industriales. Universidad Politécnica de Madrid, 2004.

[Si06] Dan Simon. *Optimal state estimation: Kalman, H-infinity, and nonlinear approaches*. John Wiley and Sons, Inc. 2006.

[Sm90] R. Smith, M. Self y P. Cheeseman. *Estimating uncertain spatial relationships in robotics*. Springer-Verlag, 1990.

[So85] Harold W. Sorenson. *Kalman filtering: Theory and application*. IEEE Press, 1985.

[Su94] K.T. Sutherland y W.B. Thompson. *Localizing in unstructured environments: Dealing with the errors*. IEEE Transactions on Robotics and Automation, 1994.

[Th05] Sebastian Thrun, Wolfram Burgard y Dieter Fox. *Probabilistic robotics*. MIT Press, 2005.

[Va96] Fernando Vázquez Núñez. *Segmentación de imágenes en grafos de contorno. Aplicación a la estimación de la profundidad y el movimiento relativo para un robot móvil autónomo*. Departamento de Ingeniería de Sistemas y Automática. Universidad de Vigo, 1996.

[Ve08] Julia Verdejo Domínguez. *Implementación del filtro de Kalman extendido para SLAM*. Escuela Técnica Superior de Ingenieros. Universidad de Sevilla, 2008.

[We01] Greg Welch y Gary Bishop. *An introduction to the kalman filter*. AMC, Inc. 2001.

[Wi06] Página web española de Wikipedia. *http://es.wikipedia.org*. Wikimedia Foundation, Inc. 2006.