CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS
DEL INSTITUTO POLITÉCNICO NACIONAL

UNIDAD ZACATENCO

DEPARTAMENTO DE CONTROL AUTOMÁTICO

“Localización y mapeo simultáneo robusto aplicado a la planeación de trayectorias en robot móviles”

TESIS
Que presenta

M.C. Salvador Ortiz Santos

Para obtener el grado de

Doctor en Ciencias

En la especialidad de
Control Automático

Directores de la Tesis: Dr.Wen Yu Liu
Dr. Erik Zamora Gómez

Ciudad de México Marzo 2019

CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL
INSTITUTO POLITÉCNICO NACIONAL

UNIDAD ZACATENCO

DEPARTAMENTO DE CONTROL AUTOMÁTICO

Resumen

Localización y mapeo simultáneo robusto aplicado a la planeación de trayectorias en robot móviles

Por M.C. Salvador Ortiz Santos

En este trabajo de tesis se presenta la localización y mapeo simultáneo robusto aplicado a la planeación de trayectorias, en otras palabras, se presenta una técnica para el problema de navegación autónoma de sistemas no lineales. Una navegación autónoma tiene como objetivo en explorar entornos desconocidos o en alcanzar objetivos establecidos sin instrucciones externas evitando colisiones con los obstáculos, donde los sistemas son equipados para obtener la percepción parcial del entorno así como las velocidades traslacionales y angulares. Para lograr la navegación autónoma se resuelven tres procesos. El primer proceso consiste en localizar el sistema en el entorno y construir un mapa, es decir, estimar la posición y orientación del sistema así como la posición de los obstáculos dentro del entorno. Este proceso se vuelve complejo cuando se requiere de un mapa para estimar la localización, de igual manera, se requiere conocer la localización para construir un mapa, este problema se le conoce como localización y mapeo simultáneo. El método más utilizado para la solución de dicho problema es el Filtro de Kalman Extendido donde una de las restricciones de este observador de estados consiste en tener ruidos distribuidos de forma gaussiana con media de cero. Para relajar dicha restricción se propone modificar el Filtro de Kalma Extendido utilizando la teoría de modos deslizantes de primer orden que tiene la característica de ser robustos ante perturbaciones acotadas e incertidumbres en los modelos matemáticos. Por otra parte, para probar el algoritmo propuesto se utilizan dos sistemas: un robot móvil y un cuadricóptero. Posteriormente, se requiere el proceso de planeación de trayectorias para generar una ruta óptima libre de obstáculos. El planificador recibe la información de la posición actual del sistema, el mapa del entorno y la posición objetivo, para generar dicha trayectoria se plantea dos algoritmos; un algoritmo utilizando la técnica del histograma polar y el algoritmo genético con Roadmap. Al generar la trayectoria se requiere que el sistema realice la navegación sobre la ruta. Para ello, hay que diseñar un controlador de seguimientos de trayectorias capaz de controlar las dinámicas del sistema para seguir la trayectoria planeada. En este trabajo, se utiliza un controlador por modos deslizantes de orden superior también conocido como super-twisting generalizado. Este es el último proceso en la navegación autónoma.

CENTER FOR RESEARCH AND ADVANCED STUDIES OF THE
NATIONAL POLYTECHNIC INSTITUTE

CAMPUS ZACATENCO

DEPARTMENT OF AUTOMATIC CONTROL

Abstract

Localización y mapeo simultáneo robusto aplicado a la planeación de trayectorias en robot móviles

By M.C. Salvador Ortiz Santos

In this thesis we describe a robust simultaneous location and mapping applied to the path planning, in other words, the problem of autonomous navigation of non-linear systems is solved. Autonomous navigation aims to explore unknown environments or achieve established goals without external instructions avoiding collisions with obstacles, where systems are equipped to obtain partial perception of the environment as well as translational and angular velocities. To achieve autonomous navigation, three processes are solved. The first process is to locate the system in the environment and build a map, that is, estimate the position and orientation of the system as well as the position of the obstacles within the environment. This process becomes complex when a map is required to estimate the location, in the same way, it is required to know the location to build a map, this problem is known as simultaneous localization and mapping. The most widely used method to solve this problem is the Extended Kalman Filter, where one of the restrictions of this state observer is to have Gaussian distributed noises with mean-zero. In order to relax this restriction, it is proposed to modify the Extended Kalma Filter using the first-order sliding mode theory, which has the characteristic of being robust to bounded perturbations and uncertainties in mathematical models. On the other hand, two systems are used to test the proposed algorithm: a mobile robot and a quadcopter. Subsequently, the process of trajectory planning is required to generate an optimal route free of obstacles. The planner receives the information of the current position of the system, the map of the environment and the target position. To generate this trajectory, two algorithms are proposed; an algorithm using the polar histogram technique and the genetic algorithm with Roadmap. When generating the trajectory, the system is required to navigate the route. For this purpose, a trajectory tracking controller capable of controlling the dynamics of the system must be designed to follow the planned trajectory. In this work, the controller is used by higher order sliding modes, also known as super-twisting. This is the last process in autonomous navigation.

Agradecimientos

Gracias al departamento de control automático del CINVESTAV por darme el espacio en su programa de doctorado y a mis profesores por su conocimiento y experiencia.

Estoy muy agradecido con mi asesor el Dr. Wen Yu por su orientación continua durante todo el programa, por su paciencia y por aceptarme como su alumno. De igual manera, a mi coasesor el Dr. Erik Zamora por sus observaciones, sus consejos profesionales, su entusiasmo y sus motivaciones. Agradezco los comentarios y sugerencias aportados por el Dr. Jorge Torres, Dr. Rafael Martínez, Dr. Sergio Salazar y Dr. Alejandro Malo.

A mis compañeros Jesús Guerrero y Luis Juárez gracias por sus consejos y hacer divertido la estancia en el departamento. También quiero agradecer a mis amigos, con ellos, se ha notado menos lo complicado que fue lograr esta meta en mi vida.

El camino recorrido hasta ahora no ha sido sencillo, pero con su apoyo, con su inmensa bondad, su enorme amor, todo ha sido posible, sólo me queda manifestar mi gran afecto y agradecimiento hacia ustedes, mi hermosa familia.

Finalmente, agradezco al Consejo Nacional de Ciencia y Tecnología (CONACYT) por el apoyo brindado durante el programa de doctorado.

Dedicado a mis padres:

Marina y Ricardo

A mis hermanos:

Hilda, Imeldo y Adrian

Índice general

Índice de figuras

Capítulo 1

Introducción

Las investigaciones relacionadas con robots móviles, vehículos submarinos autónomos, AUV (por sus siglas en inglés: Autonomous Underwater Vehicle), vehículos aéreos no tripulados, UAV (del inglés: Unmanned Aerial Vehicle) o robots humanoides son algunos de los sistemas de interés que se aplican para resolver problemas reales o para facilitar tareas del hombre. Son comúnmente utilizados para la exploración donde el acceso del hombre es imposible o de alto riego. También, estos sistemas son de gran interés para probar nuevas técnicas de control o algoritmos donde son considerados como sistemas no lineales por no cumplir con la ley de superposición, esto significa que la respuesta de salida no es proporcional a la suma de las entradas.

Estos robots tienen un problema en común cuando su tarea es explorar zonas desconocidas. El problema principal es conocer su ubicación del robot para no perderse en la exploración. Por ello, este problema es de gran interés para la comunidad científica ya que la solución a este problema no es trivial. De manera específica, este problema se conoce como: localización y mapeo simultáneo, SLAM (del inglés: Simultaneous Localization And Mapping), es decir, se desea explorar el entorno (crear un mapa con las características de interés) sin perder la ubicación del robot.

Por otra parte, en el problema SLAM las incertidumbres dificultan la solución, incertidumbres tales como: los ruidos presentes en los sensores, los inevitables errores cometidos en la aproximación del modelo matemático en los sistemas (que es una aproximación a la realidad), y la dificultad de representar el entorno a medida que la navegación autónoma incrementa producen que la tarea de resolver el mencionado problema sea ardua. Por lo tanto, no es inusual que las soluciones más exitosas hasta el momento hayan estado basadas en la utilización de técnicas probabilísticas [19], [5], [43].

Algunas técnicas probabilísticas que se conocen para resolver el problema de SLAM: las técnicas basadas por las reglas de Bayes, mapas de ocupación de rejillas o filtros de partículas (método Montercalo). Existen diversos trabajos que tienen como base estas técnicas. Sin embargo, el Filtro de Kalman Extendido, EKF (del inglés: Extended Kalman Filter) es una técnica que tiene el principio básico, la regla de Bayes [1]. El EKF es considerado como una de las mejores técnicas para la solución del problema SLAM. El EKF presenta resultados satisfactorios en la práctica y fue desarrollado por Randall Smith, Matthew Self y Peter Cheeseman a finales de la década de 1980. El EKF es un Filtro de Kalman con la característica de ser extendido a la solución de sistemas no lineales, es decir, es un estimador de estados no lineales que están bajo incertidumbres.

Los vehículos autónomos están tomando un giro importante en el mundo automotriz para imitar las capacidades humanas de manejo y control, existen diversas empresas de automóviles interesadas en realizar autos totalmente autónomos para reducir los accidentes provocados por el hombre. Hace pocas décadas esto era imposible de imaginar en vehículos que puedan llevarte al trabajo, sin embargo, en los recientes años las empresas han desarrollado diversos vehículos que presentan una autonomía precedente (figura 1.1). Por otra lado, los vehículos autónomos para la exploración requieren de algoritmos sofisticados para realizar tareas con gran éxito. De tal manera, que las investigaciones de crear vehículos autónomos es un trabajo que tiene futuro.

Cabe mencionar, los vehículos autónomos no sólo se utilizan para el transporte también se utilizan para la exploración de planetas, en trabajos en el campo, entre otras. Además, con la tecnología 4.0 que permite que las máquinas trabajen en conexión unas con otras y los procesos puedan ser automatizados, los vehículos autónomos están tomando un fuerte papel en las industrias 4.0, donde aportan en la eficiencia de las industrias. En consecuencia, con la revolución industrial 4.0 los vehículos o robots autónomos son de gran interés para tener industrias autónomas y con mayor eficiencia. Un caso de este avance, es la industria de Amazon donde implementa la tecnología 4.0 utilizando vehículos autónomos para desempeñar labores de almacenamiento y la distribución de sus productos, figura (1.2). En consecuencia, las investigaciones para estos vehículos autónomos son de gran interés y existen múltiples problemas que resolver para la inclusión general y convertir las industrias completamente autónomas con tecnologías de información y comunicación. Esto es una perspectiva de los vehículos o robots autónomos en el futuro de las industrias.

El SLAM resuelve el problema de localización y de manera simultánea construye un mapa que permite conocer el entorno para la decisión de las tareas de los robots. Para solucionar este problema podríamos pensar en el uso de dispositivos como el Sistema de

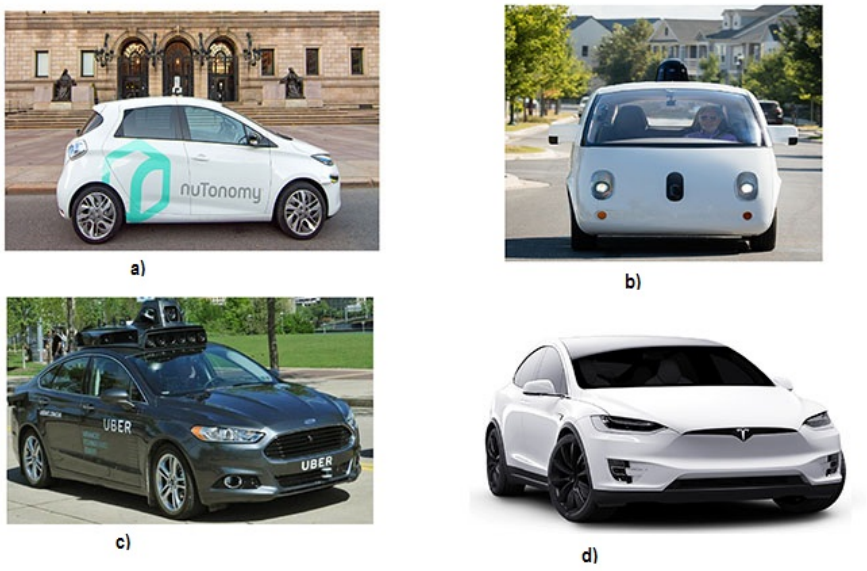

Figura 1.1: Vehículos autónomos: a) Vehículo realizado por MIT, b) Waymo (Google self-driving car project). c) el vehículo autónomo de Uber. d) los autos equipados de TESLA.

Figura 1.2: Vehículos autónomos en industrias 4.0 (Amazon)

Figura 1.3: Robots autónomos aéreos: a) cuadricóptero, b) RPA (del inglés Remotely Piloted Aircraft).

Posicionamiento Global (GPS por sus siglas en inglés: Global Positioning System) para la localización del robot, cabe mencionar que estos dispositivos no pueden ser implementados en los entornos como: interiores, cuevas, océanos, etc. También estos dispositivos son implementados en espacios de 2 dimensiones. Sin embargo, el uso de vehículo aéreo no tripulado, UAV (del inglés unmanned aerial vehicle) o vehículo submarino autónomo, AUV (del inglés autonomous underwater vehicle) se han incrementado considerablemente sus aplicaciones, donde las dimensiones del espacio incrementan a 3 dimensiones. En este trabajo, se presenta un algoritmo SLAM para un sistema UAV con 3 dimensiones, figura (1.3). Dichos vehículos se utilizan para la detección de plagas en cultivos, fallas en estructuras como edificios, puentes o túneles, también se utilizan para la vigilancia de zonas de riesgo, las aplicaciones de estos sistemas son enormes y el interés de estudio en los centros de investigación es muy amplio para probar nuevas técnicas de control o algoritmos debido a que son sistemas no lineales y además son sub-actuadas, es decir, tienen más grados de libertad que los números de actuadores.

1.1. Motivación

Para la exploración dentro de los entornos desconocidos, los vehículos autónomos deben ser capaces de decidir las rutas que deben tomar para llegar al punto objetivo. En estos problemas se requiere de un mapa para realizar la trayectoria del robot móvil, de igual

manera, se requiere de una trayectoria para construir el mapa, por lo tanto, es problema recurrente. Este problema en general se puede dividir en cinco etapas (ver figura 1.4):

1. Entorno: El algoritmo a implementar puede estar ligada directamente al tipo de entorno, por ejemplo: entornos en 2- Dimensiones ó 3-Dimensiones, también en entornos conocidos, parcialmente conocidos o totalmente desconocidos.

2. Percepción: La forma en que se percibe el entorno, es decir, depende del tipo de sensores que se implementan para cuantificar la percepción del entorno dentro de un rango limitado por los sensores.

3. Localización y mapeo: La ubicación del robot móvil está conectado con la construcción del mapa del entorno y viceversa. Por lo tanto, se obtiene un mapa y la posición del robot en todo momento.

4. Cognición y planeación de trayectorias: Reconocer el entorno es fundamental para realizar la planeación de trayectorias, es decir, analizar el mapa para la toma de decisión de la ruta a tomar en el robot móvil.

5. Control de movimiento: Realizar la tarea de seguir con la trayectoria deseada dentro del entorno es trabajo del controlador de movimiento.

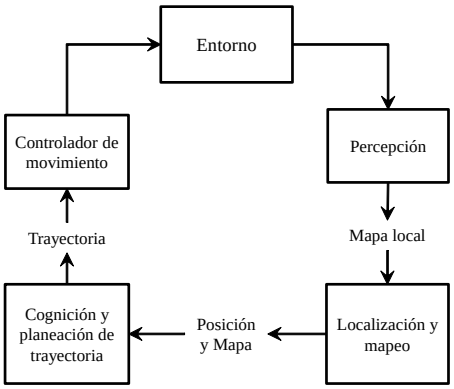

Figura 1.4: Estructura para resolver el problema de navegación autónoma en entornos desconocidos.

En la solución al problema SLAM nos encontramos en la literatura que el algoritmo EKF es el método que se utiliza ampliamente para resolver este problema. Sin embargo, este algoritmo nos proporciona ciertas restricciones que podrían considerarse como desventajas.

Algunas de las desventajas del algoritmo EKF SLAM.

- Las incertidumbres en el sistema son consideradas como distribuciones Gaussianas donde puede no corresponder con la realidad.
- Las linealizaciones introducidas en los modelos harán a lo largo del tiempo que las estimaciones de los estados no correspondan con los estados reales.
- El algoritmo considera los estados anteriores, esto genera un problema de consistencia. Es decir, origina que la exactitud de los resultados obtenidos por el algoritmo sea a menudo impredecible, observándose saltos bruscos en la estimación sin causa aparente alguna.
- El coste computacional crece al cuadrado con el número de objetos contenidos en el mapa. Este hecho limita su aplicación en tiempo real a mapas formados por unos pocos cientos de objetos.
- No siempre es sencillo o inmediato extraer características del entorno. En ocasiones ni siquiera es preciso extraer información que pueda describirse con geometrías simples, tales como: puntos, segmentos, arcos de circunferencia o planos (por ejemplo, en el interior de una mina).
- Es preciso disponer de un método de asociación de datos robusto que permita reducir los ruidos en la estimación de estados.

A pesar de las restricciones, es de los algoritmos más utilizados en la literatura debido a que se puede adaptar a diversos sistemas. Y las restricciones son muy conocidas y estudiadas.

De acuerdo a las restricciones del EKF SLAM, nos motiva a desarrollar una técnica que ayude a mejorar ciertas restricciones del algoritmo EKF. Por lo tanto, en este trabajo consideramos en mejorar el desempeño del algoritmo EKF suavizando la condición de tener incertidumbres Gaussianas, además de atenuar los errores causados por las aproximaciones lineales hechas por el EKF.

En la actualidad hay técnicas que son robustos ante los ruidos del sistema o las incertidumbres. Es el caso de la teoría de modos deslizantes donde es conocido por la robustez antes las incertidumbres acotadas. Por ello, estamos interesados en reducir las restricciones del algoritmo EKF utilizando la técnica de modos deslizantes. Sin embargo, también tenemos que tener en cuenta que utilizar técnicas como los modos deslizantes nos genera el

inconveniente conocida como "Chattering", esto se genera por la función discontinua del algoritmo. Por lo tanto, no podremos eliminar el error del SLAM, pero si atenuarla de manera considerable.

El algoritmo SLAM por sí sólo no convierte el robot en un sistema autónomo, se requiere de un planeador de trayectorias que permita encontrar una ruta desde la posición actual del robot hasta un punto objetivo para realizar la navegación. Por consiguiente, en la planeación de trayectorias se busca la ruta óptima dentro del mapa proporcionada por el SLAM. Existen técnicas como la descomposición de rejillas o las fuerzas de campo potencial que permiten encontrar trayectorias, sin embargo, estas técnicas están sujetos en aplicaciones donde los entornos son totalmente conocidos donde el objetivo principal es encontrar una trayectoria optima y global. Ahora bien, un histograma polar es una técnica sencilla para encontrar soluciones en navegaciones con entornos desconocidos debido a que podemos encontrar zonas que permitan avanzar en la navegación, sin embargo, se encuentra que esta técnica puede caer en soluciones locales cuando los obstáculos no tiene una forma convexa.

El problema de planeación de trayectorias se convierte en un problema de optimización, es decir, se requiere encontrar una trayectoria óptima bajo algún criterio, criterios como: la longitud de la ruta, el consumo de energía, el tiempo de navegación u otras. Los métodos heurísticos son ampliamente utilizados para resolver problemas de optimización, sin embargo, el algoritmo genético (GA del inglés: Genetic Algorithm) es el método más destacado dentro de las técnicas heurísticas, debido a que son capaces de encontrar soluciones globales. En consecuencia, se plantea un planeador de trayectorias utilizando este algoritmo combinado con una con el método Roadmap para modelar el entorno construido por el SLAM.

Finalmente, para obtener una navegación autónoma se requiere de un controlador de seguimiento de trayectorias para lograr seguir la ruta obtenida del planeador de trayectorias. El buen desempeño en la navegación depende del SLAM, la planeación de trayectorias y el controlador de seguimiento de trayectorias. Por ende, para esta última etapa se presenta un controlador por modos deslizantes de orden superior llama Super-Twisting generalizado que tiene la característica de ser robusta ante las perturbaciones o incertidumbres acotadas además es capaz atenuar el fenómeno "chattering" que se manifiesta en los modos deslizantes a causa de las funciones discontinuas,.por ello, se plantea esta teoría para el seguimiento de trayectorias con el objetivo de obtener un desempeño satisfactorio en la navegación autónomo.

1.2. Objetivos

Presentar una nueva tecnica para el problema de localización y mapeo simultáneo de un sistema no lineal aplicando un algoritmo capaz de atenuar los errores de estimación del sistema y con la construcción del mapa realizar la planeación de trayectorias para un robot móvil en entornos desconocidos.

El objetivo principal es dividido en objetivos particulares para lograr una respuesta favorable. Los objetivos particulares buscados al realizar la presente tesis son:

1. Diseñar un algoritmo con la teoría de modos deslizantes para la solución del problema de SLAM, de tal manera, que se atribuya robustez al algoritmo ante las perturbaciones acotadas.

2. Diseñar un algoritmo de planeación de trayectorias para el robot móvil.

3. Implementar un controlador de seguimiento de trayectorias robusto ante perturbaciones acotadas.

1.3. Estructura de la tesis

El capítulo 1, ofrece las motivaciones y objetivos concretos buscados en la presente tesis. En el capítulo 2, presentamos el problema de localización y mapeo simultáneo, en consecuencia, se presentan dos algoritmos de solución: el primer algoritmo se utiliza el método de Filtro de Kalman Extendido y en el segundo algoritmo se utiliza la teoría de modos deslizantes para darle robustez al Filtro de Kalman Extendido y se prueba la estabilidad del nuevo algoritmo llamada modos deslizantes SLAM. También presentamos los resultados obtenidos en la implementación de dos sistemas totalmente diferentes: en un robot móvil y en un sistema UAV. En el capítulo 3, presentamos dos algoritmos de planeación de trayectorias para el robot móvil: planeación de trayectorias con histograma polar y planeación de trayectorias con Roadmap algoritmo genético. En el capítulo 4, se presenta un controlador de seguimiento de trayectorias. En el último capítulo, se realizan las conclusiones obtenidas durante el desarrollo de este trabajo, así como los trabajos futuros a desarrollar.

En la figura (1.5) se resume los capítulos de la presente tesis para lograr con el objetivo de una navegación autónoma en los robots móviles, presentamos dos algoritmos para el

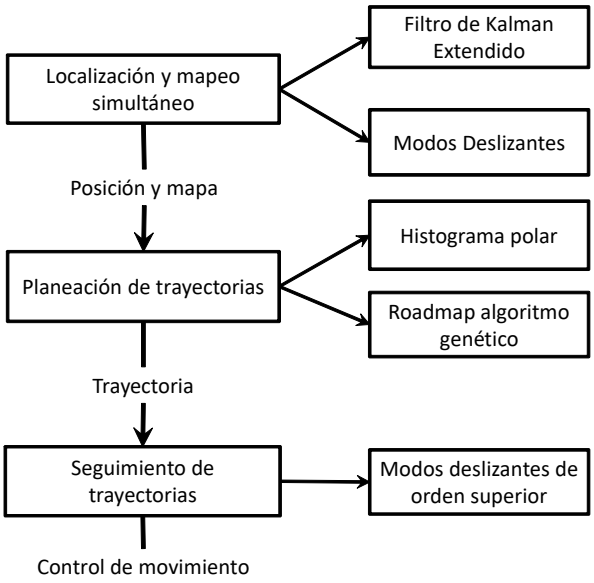

Figura 1.5: Estructura de la tesis.

problema de localización y mapeo simultáneo utilizando un filtro de kalman extendido y la teoría de modos deslizantes, donde obtenemos la posición y orientación del robot. Posteriormente, se requiere de un planeador de trayectorias el cual se proponen dos técnicas: el método del histograma polar y el Roadmap algoritmo genético que permite obtener una trayectoria optima desde la posición del robot hasta un punto objetivo. Finalmente, se diseña un controlador de seguimiento de trayectorias por modos deslizantes de orden superior o Super-Twisting para realizar la navegación.

1.4. Publicaciones

Conferencias

1. Sliding mode SLAM for robust simultaneous localization and mapping, 44th Annual Conference of the IEEE Industrial Electronics Society (IECON18).

2. Sliding mode three-dimension SLAM with application to quadrotor helicopter, 15th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE18).

Revistas

1. Stable sliding mode SLAM for robust simultaneous localization and mapping, *International Journal of Robotics and Automation*, aceptado condicionado.

2. Autonomous navigation in unknown environments using robust SLAM, *Journal of Systems Science*, sometido.

3. Path planning of autonomous robot in unknown environment with roadmap genetic algorithm and SLAM, *Autonomous Robots*, sometido

Capítulo 2

Localización y mapeo simultáneo.

La localización y mapeo simultáneo, SLAM (figura 2.1) consiste en estimar los estados del sistema y de manera simultánea construye un modelo del entorno, esto se logra a través de sensores equipados en el sistema. De manera específica, los estados del sistema son la posición (en caso de un robot móvil los estados son la posición y orientación) y el modelo del entorno consiste en construir un mapa que representa los aspectos de interés (por ejemplo, la posición de las marcas fijas o de los obstáculos) del entorno donde el sistema opera.

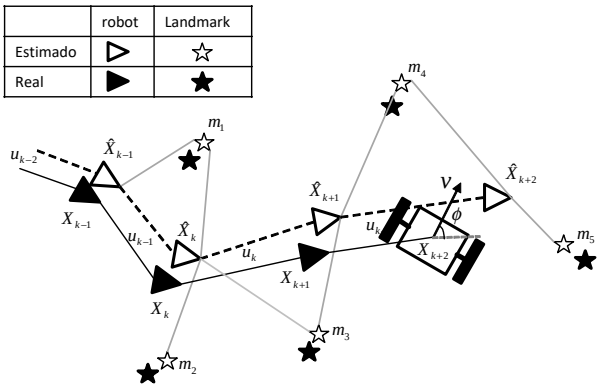

Figura 2.1: Localización y mapeo simultáneo (SLAM).

En el SLAM hay dos importantes razones para usar el mapa del entorno: La primera razón, un mapa es indispensable para realizar otras tareas; por ejemplo, un mapa proporciona información en la planeación de trayectorias o proporciona una visualización intuitiva para un operador humano. La segunda razón, el mapa permite corregir los er-

rores cometidos por la estimación de estados del sistema, es decir, en la ausencia de un mapa los estados del sistema se desvían rápidamente con el tiempo; en caso contrario, utilizando un mapa, por ejemplo, un conjunto de marcas fijas distinguibles, el sistema puede "restablecer" su error de localización utilizando las marcas fijas, el sistema se convierte en lazo cerrado. Por otra parte, la aplicación del SLAM se utiliza en todos los escenarios donde el entorno es desconocido y es necesaria la construcción de un mapa. La implementación del SLAM puede ser en sistemas como robots móviles, aeronaves o submarinos con dos diferentes entornos: en espacios cerrados (dentro de edificios, cuevas, etc.) o abiertos (calles, parques, etc.). Las aplicaciones del algoritmo SLAM están limitados en el tipo de sensores que se utilizan.

Antes de continuar y presentar la técnica para el problema de SLAM, respondemos la siguiente pregunta: ¿Realmente es necesario el SLAM en los robots autónomos? La respuesta depende de la aplicación que se necesita realizar y en el equipo tecnológico disponible. Por ejemplo, se puede contar con un dispositivo de localización como el Sistema de Posicionamiento Global, más conocida por sus siglas en inglés, GPS (Global Positioning System) para entornos 2-D o en superficies que es posible su implementación donde la estimación de la localización del robot no es necesaria, aunque, se utiliza el SLAM para construir un mapa. Por supuesto, el SLAM es ampliamente utilizado donde no es posible utilizar el GPS. También, podríamos pensar en resolver el problema de localización de un robot integrando numéricamente las velocidades para encontrar su posición, esto realmente no es posible debido a los diversos fuentes de incertidumbre, dicho de otra manera, factores que incrementan la dificultad de estimar la localización, factores como: Los sensores presentan usualmente ruidos en los datos que proporcionan, el robot puede tener desplazamientos imprecisos, en otras palabras, tener una velocidad sin ninguna desplazamiento del robot por falta de fricción, también puede haber errores por el tipo de terrenos donde son operados como los terrenos irregulares, por ejemplo, si el robot opera en césped y el robot es pequeño entonces el ruido será mayor que si se trabaja en piso firme. Por ello, el SLAM "resuelve" este problema debido a que es un sistema de lazo cerrado utilizando el mapa de su entorno.

El problema de SLAM es un tema relevante en la comunidad científica debido a su complejidad. En consecuencia, se puede dividir el problema de SLAM en tres partes y así evaluar el desempeño del SLAM conforme a lo que se requiere obtener, por ejemplo, la precisión en la estimación de los estados del sistema. En el SLAM siempre se pierde el desempeño de una de las partes, es decir, si el objetivo es la precisión de los estados

entonces se puede perder el buen desempeño de la velocidad de avance en los sistemas o viceversa. A continuación, presentamos las partes del SLAM [13]:

1. Robot: tipo de movimiento (por ejemplo: la dinámica en el sistema, la velocidad de operación), los sensores disponibles (por ejemplo: la resolución, la velocidad de muestreo) o los recursos computacionales disponibles.

2. Entorno: entornos en 2-D ó 3-D, la presencia de las marcas fijas ya sean naturales o artificiales, entornos irregulares y riesgos de aliasing, por ejemplo: un láser puede escanear el entorno y las señales se tornan indistinguibles entre dos objetos.

3. Requisitos de desempeño: precisión deseada en la estimación de los estados del robot, precisión y tipo de representación del entorno (por ejemplo: marcas fijas o figuras geométricas como las paredes), convergencia de éxito (porcentaje de pruebas en las que se cumple los límites de exactitud), tiempo máximo de operación, tamaño máximo del área mapeada.

En este trabajo nos enfocamos en resolver el problema del SLAM desarrollando un algoritmo capaz de mejorar el desempeño de estimación de los estados del sistema y construir un mapa de marcas fijas. Una gran parte de las soluciones del SLAM plantea las estimaciones de los estados del sistema y el mapa del entorno como distribuciones de probabilidad, tal es el caso del Filtro de Kalman.

El Filtro de Kalman es uno de las herramientas más utilizadas para resolver el problema del SLAM. De forma general, se utiliza el Filtro de Kalman Extendido debido a la implementación a sistemas no lineales además son sencillos de implementar y obtienen resultados favorables. Sin embargo, se continúa realizado diversas modificaciones o nuevas técnicas para resolver el problema de SLAM con el objetivo de mejorar su desempeño, tal es el caso del Filtro de información extendido (EIF), filtros de partículas, Filtro de Bayer, fastSLAM son algunas de las variantes que se encuentra en la literatura. Cabe mencionar, que en el departamento donde se desarrolla este trabajo también se encuentran trabajos que se enfocan en resolver el problema de SLAM o la implementación de este algoritmo. Es el caso del trabajo de tesis de Cariño-Escobar (2015) que implemento un SLAM en un Cuadricóptero utilizando el Filtro de Kalman Extendido.

Resolver el problema de SLAM con el Filtro de Kalman Extendido nos genera diversos limitantes, tal como: los ruidos v_k y w_k en el sistema tienen que tener una distribución Gaussiana, es decir, $E\left\{v_k v_k^T\right\} = I$ y $E\left\{w_k w_k^T\right\} = I$, en muchas aplicaciones esta limitante no se cumple. En el trabajo de Zamora-Gómez [100] presenta este problema y nos

proporciona un algoritmo nuevo llamado SLAM Elipsoide que es una variante del Filtro de Kalman Extendido.

En este trabajo el objetivo es mejorar el desempeño del Filtro de Kalma Extendido utilizando una re-alimentación no lineal que permita ser robusto ante las perturbaciones. Por lo tanto, se considera en presentar el Filtro de Kalman Extendido para el problema del SLAM en la siguiente sección.

2.1. Filtro de Kalman Extendido.

El Filtro de Kalman Extendido asume que el modelo dinámico del sistema y el modelo de observación son establecidos por funciones no lineales f y h, respectivamente:

$$\begin{aligned} X_{k+1}^r = &\quad f(X_k^r, U_k) + w_k \\ Z_k = &\quad h(X_k^r) + v_k \end{aligned} \tag{2.1}$$

Donde X_k^r son los estados del sistema. U_k son los controles de entrada. Z_k son las mediciones de las marcas fijas (Landmarks), con $k \in N_0$ que representa el tiempo discreto. Por último, w_k y v_k representan los ruidos de entrada del sistema (2.1), los ruidos tienen las características que son de tipo Gaussiana y con media cero, es decir, $E\left\{v_k v_k^T\right\} = I$ y $E\left\{w_k w_k^T\right\} = I$. Por consiguiente, describiremos el modelo del robot móvil $f(\cdot,\cdot)$, así como el modelo de observación de las marcas fijas $h(\cdot)$.

Modelo de un robot móvil

1. **Modelo del robot móvil.** Para la descripción del movimiento del robot móvil, utilizamos el modelo cinemático para la trayectoria de un vehículo. esto es:

$$X_{k+1}^r = f(X_k^r, U_k) + w_k = \begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \phi_{k+1} \end{bmatrix} + w_k \tag{2.2}$$

con

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \phi_{k+1} \end{bmatrix} + w_k = \begin{bmatrix} x_k + u_{k,v}\delta t \cos(\phi_k) \\ y_k + u_{k,v}\delta t \sin(\phi_k) \\ \phi_k + f_\phi(u_{k,\gamma}, u_{k,v}, \delta t) \end{bmatrix} + w_k$$

Donde $u_{k,\gamma}$ y $u_{k,v}$ es la velocidad lineal y la velocidad angular (son los controles de entrada[1]). $X_k^r = \begin{bmatrix} x_k & y_k & \phi_k \end{bmatrix}^T$ que representa la posición y la orientación del robot móvil. δt es el periodo de muestreo. La función f_ϕ depende de la estructura del vehículo. A continuación, presentamos dos modelos de vehículos.

Ejemplo 1: Modelo simple de movimiento de un vehículo en tiempo discreto.

$$\begin{aligned} x_{k+1} &= x_k + u_{k,v}\delta t \cos(\phi_k) \\ y_{k+1} &= y_k + u_{k,v}\delta t \sin(\phi_k) \\ \phi_{k+1} &= \phi_k + u_{k,\gamma}\delta t \end{aligned} \tag{2.3}$$

La ecuación (2.3) se puede obtener directamente de la discretización del siguiente modelo dinámico.

$$\begin{aligned} \dot{\phi} &= u_\gamma \\ \dot{x} &= u_v \cos\phi \\ \dot{y} &= u_v \sin\phi \end{aligned} \tag{2.4}$$

Ejemplo 2: Modelo de un vehículo similar a un automóvil.

$$\begin{aligned} x_{k+1} &= x_k + u_{k,v}\delta t \cos(\phi_k) \\ y_{k+1} &= y_k + u_{k,v}\delta t \sin(\phi_k) \\ \phi_{k+1} &= \phi_k + \frac{u_{k,v}\delta t \tan(u_{k,\gamma})}{L} \end{aligned} \tag{2.5}$$

Donde L es la longitud entre las ruedas del vehículo. En este modelo $u_{k,\gamma}$ es la velocidad de la orientación de las ruedas.

2. **Modelo de observación.** En el tiempo k, la observación del $i-$th marca fija, se obtiene mediante los sensores que se encuentran abordo del robot móvil. Esta observación se cuantifica con una distancia r^i, una dirección θ^i y una etiqueta de identificación s^i, esto es:

$$Z_k^i = h(X_k^r) + v_k = \begin{bmatrix} r_k^i \\ \theta_k^i \\ s^i \end{bmatrix} + v_k \tag{2.6}$$

[1]En el apéndice se describe la odometría del robot móvil.

Con

$$\begin{bmatrix} r_k^i \\ \theta_k^i \\ s^i \end{bmatrix} + v_k = \begin{bmatrix} \sqrt{(m_x^i - x_k)^2 + (m_y^i - y_k)^2} + N(0, \sigma_r) \\ \arctan\left(\dfrac{m_y^i - y_k}{m_x^i - x_k}\right) - \phi_k + N(0, \sigma_\theta) \\ m_s^i + N(0, \sigma_s) \end{bmatrix}$$

Donde $\left[m_x^i, m_y^i\right]^T$ son las coordenadas de las marcas fijas observados en el tiempo k, y m_s^i es su etiqueta. Las varianzas del ruido de medición están dadas por σ_r, σ_θ y σ_s.

En resumen, la ecuación (2.1) se describe en los modelos (2.3) ó (2.5) y (2.16) para un robot móvil. Es decir, $X^r = [x, y, \phi]^T$ que son los estados del robot (posición y orientación) y la observación $z^i = [r^i, \theta^i, s^i]^T$ que son la distancia, la dirección y la etiqueta de las marcas fijas.

Los modelos antes presentados cuentan con ciertas restricciones, tales como:

1. El modelo del robot móvil considera que los ejes de las ruedas son perpendiculares al plano de navegación, es decir, que el vehículo siempre navega en una superficie uniforme.

2. Las ruedas del robot móvil no se deslizan en la superficie, es decir, que el desplazamiento lineal del vehículo corresponde al desplazamiento angular de las ruedas.

3. Las marcas fijas son estacionarias, es decir, que las marcas fijas tienen una velocidad cero.

4. No se considera las perturbaciones que son ajenas al sistema.

Aun cumpliendo con estás restricciones no podemos asegurar que la estimación de la posición y orientación del robot móvil por la odometría sean exactos debido a que existen ruidos en los sensores. Por ello, se utiliza el EKF para contrarrestar estos fenómenos y obtener una estimación tolerable. Ahora bien, realizar la tarea del SLAM es nuestro propósito y para esto hacemos uso del EKF. Por lo tanto, planteamos el algoritmo EKF SLAM.

El algoritmo EKF SLAM puede dividir en tres secciones principales: primera etapa de predicción, segunda etapa de medición y observación de las marcas fijas y por último

una etapa de corrección. Por consiguiente, presentaremos el algoritmo del EKF SLAM describiendo las etapas antes mencionados.

Etapas del EKF SLAM.

El EKF SLAM estima la posición y orientación del robot X_k^r y también estima las coordenadas de las marcas fijas X^m. De tal manera, que se incluyen las ubicaciones de las marcas fijas en el vector de estados. La combinación del vector está dado por

$$X_k = [x, y, \phi, m_x^1, y_y^1, s^1, ..., m_x^N, m_y^N, s^N]^T = \begin{bmatrix} X_k^r \\ X^m \end{bmatrix} \tag{2.7}$$

Donde $X^m = [m_x^1, m_y^1, s^1, ..., m_x^N, m_y^N, s^N]^T$ representa el mapa del entorno, note que X^m no depende del tiempo debido a que las marcas fijas son estacionarias. La dimensión del vector de estados X_k tiene una dimensión de $(3N+3)$, donde N corresponde al número de marcas fijas en el mapa.

El modelo dinámico del sistema con las nuevas variables de estado

$$X_{k+1} = \begin{bmatrix} f(X_k^r, U_k) + w_k \\ X^m \end{bmatrix} = F(X_k, U_k) + [w_k, 0]^T \tag{2.8}$$

Donde $[w_k, 0]^T$ es el ruido del sistema que sólo afecta los estados X_k^r.

Al comienzo, el algoritmo SLAM no cuenta con un mapa, por lo tanto, los estados iniciales pueden ser el origen, esta suposición es arbitraria ya que si se conoce las coordenadas y la orientación en las que se encuentra el robot móvil se toma dicha posición y orientación como punto de partida, en este caso, suponemos que iniciamos en el origen, de tal manera que la estimación de los estados del robot móvil y la covarianza inicial se expresan como sigue:

$$\hat{X}_0^+ = [0\ 0\ 0]^T$$

$$P_0^+ = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{2.9}$$

Donde $\hat{X}_0^+$ son los estados iniciales estimados a posteriori y P_0^+ es la matriz de covarianza inicial a posteriori. A continuación, se describen las partes del EKF SLAM anteriormente mencionado:

- **Predicción.** La predicción de la posición y orientación del robot móvil se basa directamente del modelo del robot móvil con la entrada $U_k = [u_{k,v}, u_{k,\gamma}]^T$. Es decir, que la estimación de los estados a priori $\hat{X}_k^-$ se realiza a través del modelo dinámico del sistema con una entrada de excitación U_k. Esto es:

$$\hat{X}_{k+1}^- = \begin{bmatrix} \hat{X}_{k+1}^r \\ X^m \end{bmatrix} = \begin{bmatrix} f(\hat{X}_k^r, U_k) \\ X^m \end{bmatrix} = F(\hat{X}_k^+, U_k) \tag{2.10}$$

Sustituyendo el modelo dinámico del robot (2.3), tenemos que

$$\hat{X}_{k+1}^- = F(\hat{X}_k^+, U_k) = \hat{X}_k^+ + G_x^T \begin{bmatrix} u_{k,v}\delta t \cos(\hat{X}_{k,\phi}^+) \\ u_{k,v}\delta t \sin(\hat{X}_{k,\phi}^+) \\ u_{k,\gamma}\delta t \end{bmatrix} \tag{2.11}$$

Con

$$G_x = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \end{bmatrix} \in R^{(3,3N+3)}$$

Donde $\hat{X}_k^-$ son la estimación de estados a priori, $\hat{X}_k^+$ es la estimación de estados a posteriori y G_x es una matriz de mapeo esto se realiza debido a que la dinámica del robot móvil sólo afecta los tres primeros estados que corresponden al modelo del sistema, las demás variables son ceros y corresponden a las marcas fijas que no son afectados por la dinámica del robot. Este proceso también se conoce como odometría.

El concepto del EKF es aproximar la función no lineal a una función lineal, esta técnica se le llama linealización. Para la linealización se utiliza la serie de Taylor donde aproxima una función F a una función lineal F', en efecto, se determina la pendiente en un punto de operación $\hat{X}_k^+$. Dicho pendiente se obtiene con la derivada parcial.

$$A_k = F'(X_k, U_k) := \frac{\partial F(X_k, U_k)}{\partial X_k} \mid_{X_k = \hat{X}_k^+} \tag{2.12}$$

Esto significa que la ecuación (2.8) se puede expresar como:

$$X_{k+1} = F(\hat{X}_k^+, U_k) + A_k(X_k - \hat{X}_k^+) + O_1\left[(X_k - \hat{X}_k^+)^2\right] + [w_k, 0]^T \tag{2.13}$$

Donde O_1 son las funciones no lineales.

Por otra parte, la matriz de covarianza se define como

$$P_{k+1}^{-} = A_k P_k^{+} A_k^T + G_x^T R_1 G_x \tag{2.14}$$

Con

$$A_k = I + G_x^T \begin{pmatrix} 0 & 0 & -v_k \delta t \sin(\phi_k) \\ 0 & 0 & v_k \delta t \cos(\phi_k \\ 0 & 0 & 0 \end{pmatrix} G_x \tag{2.15}$$

Donde P_k^{+} es la matriz de covarianza a posteriori, R_1 es la matriz de covarianza del ruido del sistema. Además, la matriz identidad I tiene la dimensión $(3N+3) \times (3N+3)$.

Las ecuaciones (2.10) y (2.14) son denominados como la etapa de predicción en el algoritmo EKF SLAM. Dichas ecuaciones representan la estimación de los estados a priori $\hat{X}_{k+1}^{-}$ y la matriz de covarianza a priori P_{k+1}^{-} del Filtro de Kalman Extendido.

- **Medición y observación**. Para la medición de las marcas fijas se realiza a través del modelo de observación, el modelo está dado como

$$Z_k^i = \begin{bmatrix} r_k^i \\ \theta_k^i \\ s_i \end{bmatrix} + v_k = \begin{pmatrix} \sqrt{(m_x^i - x_k)^2 + (m_y^i - y_k)^2} \\ \arctan\left(\dfrac{m_y^i - y_k}{m_x^i - x_k}\right) - \phi_k \\ m_s^i \end{pmatrix} + N(0, R_2) \tag{2.16}$$

con

$$R_2 = \begin{pmatrix} \sigma_r & 0 & 0 \\ 0 & \sigma_\theta & 0 \\ 0 & 0 & \sigma_s \end{pmatrix} \tag{2.17}$$

Donde R_2 es una matriz de covarianza del modelo de observación y de manera simplificada Z_k^i se tiene como:

$$Z_k^i = H(X_k) + v_k \tag{2.18}$$

Al realizada la medición del $i-$th marca fija Z_k^i se obtiene en consecuencia la estimación de la ubicación de la marca fija $[m_x^i, m_y^i, m_s^i]^T$ con respecto a la posición estimada del robot móvil $\hat{X}_k^{-}$. Esto es

$$\begin{pmatrix} m_x^i \\ m_y^i \\ m_s^i \end{pmatrix} = \begin{pmatrix} \hat{X}_{k,x}^{-} \\ \hat{X}_{k,y}^{-} \\ s_k \end{pmatrix} + r_k^i \begin{pmatrix} \cos(\theta_k + \hat{X}_{k,\phi}^{-}) \\ \sin(\theta_k + \hat{X}_{k,\phi}^{-}) \\ 0 \end{pmatrix} \tag{2.19}$$

Si la $i-$th marca fija es observada por primera vez entonces se estima las coordenadas con respeto a la posición del robot móvil, como realiza en la ecuación (2.19). Una observación de estas características corresponde un estado más en el vector de estados. Por consiguiente, el vector de estados incrementa con respecto a las marcas fijas que se observan.

- **Corrección.** En esta etapa se realiza la corrección de los estados a través de las mediciones y observaciones de las marcas fijas. Para esto, se linealiza el modelo de medición no lineal (2.18) utilizando la serie de Taylor, esto es

$$Z_k^i = H(\hat{X}_k^-) + C_k(X_k - \hat{X}_k^-) + O_2\left[(X_k - \hat{X}_k^-)^2\right] + v_k^i \tag{2.20}$$

Donde O_2 son funciones no lineales, C_k es la matriz Jacobiano de la función $H(X_k)$ y evaluada en el punto de operación $\hat{X}_k^-$ para la $i-$th marca fija. Es decir

$$C_k = H'(\hat{X}_k^-, U_k) := \frac{\partial H(X_k, U_k)}{\partial X_k} \mid_{X_k = \hat{X}_k^-} \tag{2.21}$$

De manera específica, la matriz Jacobiano del modelo de observación (2.16), está dada como:

$$h'_k = \begin{bmatrix} \frac{\partial r_k^i}{\partial x_k} & \frac{\partial r_k^i}{\partial y_k} & \frac{\partial r_k^i}{\partial \phi_k} & \frac{\partial r_k^i}{\partial m_x^i} & \frac{\partial r_k^i}{\partial m_y^i} & \frac{\partial r_k^i}{\partial m_s^i} \\ \frac{\partial \theta_k^i}{\partial x_k} & \frac{\partial \theta_k^i}{\partial y_k} & \frac{\partial \theta_k^i}{\partial \phi_k} & \frac{\partial \theta_k^i}{\partial m_x^i} & \frac{\partial \theta_k^i}{\partial m_y^i} & \frac{\partial \theta_k^i}{\partial m_s^i} \\ \frac{\partial S^i}{\partial x_k} & \frac{\partial S^i}{\partial y_k} & \frac{\partial S^i}{\partial \phi_k} & \frac{\partial S^i}{\partial m_x^i} & \frac{\partial S^i}{\partial m_y^i} & \frac{\partial S^i}{\partial m_s^i} \end{bmatrix}_{X_k = \hat{X}_k^-} \tag{2.22}$$

Esto es

$$h'_k = \begin{pmatrix} \frac{m_x^i - \hat{X}_{k,x}^-}{\sqrt{q_k}} & \frac{m_{j,y} - \hat{X}_{k,y}^-}{\sqrt{q_k}} & 0 & \frac{\hat{X}_{k,x}^- - m_x^i}{\sqrt{q_k}} & \frac{\hat{X}_{k,y}^- - m_y^i}{\sqrt{q_k}} & 0 \\ \frac{\hat{X}_{k,y}^- - m_y^i}{q_k} & \frac{\hat{X}_{k,x}^- - m_y^i}{q_k} & -1 & \frac{m_y^i - \hat{X}_{k,y}^-}{q_k} & \frac{m_x^i - \hat{X}_{k,x}^-}{q_k} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.23}$$

Con $q_k = (m_{j,x} - \hat{X}_{k,x}^-)^2 + (m_{j,y} - \hat{X}_{k,y}^-)^2$. Por lo tanto, la matriz Jacobiano del modelo de observación es

$$C_k = h'_k G_z^i \tag{2.24}$$

Con

$$G_z^i = \begin{pmatrix} 1 & 0 & 0 & \underbrace{0\cdots0}_{3i-3} & 0 & 0 & 0 & \underbrace{0\cdots0}_{3N-3i} \\ 0 & 1 & 0 & & 0 & 0 & 0 & \\ 0 & 0 & 1 & & 0 & 0 & 0 & \\ 0 & 0 & 0 & & 1 & 0 & 0 & \\ 0 & 0 & 0 & & 0 & 1 & 0 & \\ 0 & 0 & 0 & & 0 & 0 & 1 & \end{pmatrix} \tag{2.25}$$

Donde G_z^i es una matriz de mapeo con dimensión $(6 \times (3N+3))$.

Antes de realizar la corrección es necesario encontrar la ganancia K_k del EKF en el tiempo k, esto se realiza con la siguiente ecuación

$$K_k = P_k^- C_k (C_k P_k^- C_k + R_2)^{-1} \tag{2.26}$$

La corrección está dada por las ecuaciones (2.27) y (2.28). En otras palabras, se realiza la estimación de los estados a posteriori (2.27) y la matriz de covarianza a posteriori (2.28) con las siguientes ecuaciones.

$$\hat{X}_k^+ = \hat{X}_k^- + K_k(Z_k^i - \hat{Z}_k^i) \tag{2.27}$$

$$P_k^+ = (I - K_k C_k) P_k^- \tag{2.28}$$

Donde $\hat{Z}_k^i$ es la estimación de la medición de la $i-$th marca fija. Esta estimación se realiza de la siguiente manera

$$\hat{Z}_k^i = \begin{pmatrix} \sqrt{(m_x^i - \hat{X}_{k,x}^-)^2 + (m_y^i - \hat{X}_{k,y}^-)^2} \\ \arctan\left(\dfrac{m_y^i - \hat{X}_{k,y}}{m_x^i - \hat{X}_{k,x}^-}\right) - \hat{X}_{k,\phi}^- \\ m_s^i \end{pmatrix} \tag{2.29}$$

En la figura (2.2) tratamos de resumir el algoritmo de localización del sistema $F(\cdot,\cdot)$ donde podemos ver que los estados de salida son $\hat{X}_k^+$. La parte de observación viene dada por $h(\hat{X}_k^-)$ con la medición hecha por Z_k. Por último, el sistema se convierte en lazo cerrado al implementar el ajuste utilizando la ganancia del Filtro de Kalman Extendido. Cabe mencionar, el resumen que se presenta es incompleto para el algoritmo EKF SLAM ya que en ningún momento se construye el mapa, sin embargo, es un apoyo visual del EKF para la localización del vehículo.

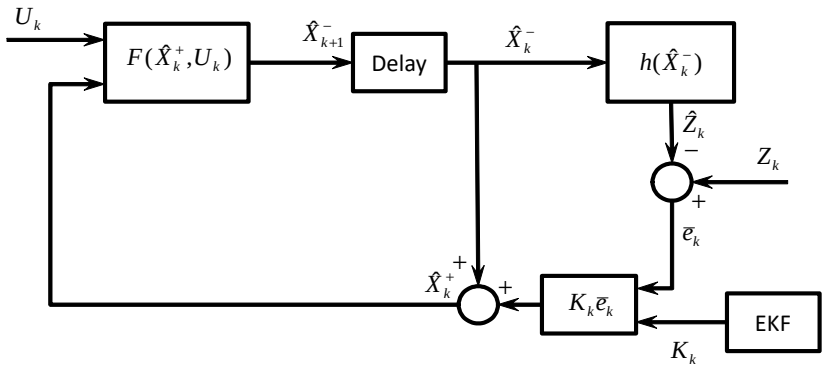

Figura 2.2: Diagrama del algoritmo EKF para la localización.

2.2. Modos deslizantes

El algoritmo EKF se utiliza ampliamente para resolver el problema de SLAM a pesar de conocer las limitantes que presenta el EKF, tal como: los ruidos de entrada deben presentar una distribución Gaussiana (como se puede notar en el algoritmo EKF SLAM presentado anteriormente). Sin embargo, en diversas aplicaciones esta restricción no es posible consumar. Otra limitante del algoritmo EKF SLAM, ocurre en la corrección de los estados de estimación, debido a que la corrección se realiza a través de la linealización del sistema, es decir, que se encuentran las matrices Jacobianas con las cuales se realiza la compensación del algoritmo (evaluando la ganancia del EKF). Por consiguiente, las funciones no lineales O_1 y O_2 de la linealización son despreciados. Estas funciones no lineales de la linealización al no considerarse pueden incrementar el error de la estimación del sistema conforme se incrementa la navegación.

En este trabajo se propone un método para resolver el problema planteada anteriormente. Dicho de otra manera, la restricción de los ruidos no debe pertenecer a una distribución Gaussiana, en cambio, como se verá posteriormente el ruido debe ser acotado, de igual manera, las funciones no lineales de la linealización del sistema se piden que sean acotadas. Ya que se realiza una re-alimentación no lineal del algoritmo EKF para compensar los ruidos y las funciones no lineales que se omiten. De esta manera, el algoritmo EKF es robusto ante las perturbaciones o las no linealidades del sistema.

Para realizar el algoritmo EKF robusto nosotros hacemos uso de la teoría de modos deslizantes [90]. La teoría de modos deslizantes es un tema de interés en la comunidad de control automático. Se han desarrollado una variedad de controladores para diversos sistemas lineales y no lineales, así como estimadores de estados (observadores). Los mo-

dos deslizantes tienen la característica de ser robustos ante perturbaciones o errores de modelado en el sistema. Una de las desventajas que se considera dentro de la teoría de modos deslizantes es el conocido "chattering", este fenómeno se presenta por la discontinuidad que se realiza en esta técnica. Sin embargo, existen técnicas para contrarrestar este problema es el caso de modos deslizantes de orden superior.

Modos deslizantes SLAM

El algoritmo EKF SLAM es un observador no lineal, por ello, se propone modificar para mejorar su desempeño. La modificación consiste en realizar una re-alimentación no lineal del EKF. Dicho de manera específica, se realiza una inyección discontinua a la predicción del algoritmo EKF.

Como se presentó en el sección anterior el algoritmo EKF se puede dividir en tres etapas: Predicción (2.10) y (2.14), medición/observación (2.16) y (2.19) y corrección (2.27 y 2.28). En consecuencia, el algoritmo modos deslizantes SLAM es una variante del algoritmo EKF SLAM donde la predicción robusta está dada como:

- Predicción robusta

$$\hat{X}_{k+1}^{-} = \begin{bmatrix} \hat{X}_{k+1}^{r} \\ X^{m} \end{bmatrix} = F(\hat{X}_{k}^{+}, U_k) - \rho\,[sgn(e_k)] \tag{2.30}$$

  Donde $\rho > 0$ son las ganancias de la inyección discontinua o la ganancia de modos deslizantes. $e_k = \hat{X}_k^- - \hat{X}_k^+$ es el error de estimación del algoritmo SLAM. sgn es la función discontinua dada como:

$$sgn(e_k) = \frac{e_k}{\|e_k\|} = \begin{cases} 1 & \text{if } e_k > 0 \\ -1 & \text{if } e_k < 0 \end{cases} \tag{2.31}$$

  La matriz de covarianza está dada como:

$$P_{k+1}^{-} = A_k P_k^{+} A_k^{T} + G_x^{T} R_1 G_x \tag{2.32}$$

  Donde A_k es la matriz Jacobiano del modelo dinámico del robot evaluado en $\hat{X}_k^+$.

  Hemos mencionado que la predicción robusta es la etapa que se modifica para mejorar el desempeño del algoritmo EKF tradicional. Las etapas de Medición/Observación y la corrección son similares al algoritmo EKF SLAM presentados en el capítulo anterior. Escribiendo estas ecuaciones tenemos:

- Medición y observación.

  El modelo matemático de la $i-$th observación se presenta en la siguiente ecuación

$$Z_k^i = H(X_k) + v_k^i \tag{2.33}$$

  La medición de la $i-$th marca fija se evalúa con respecto a los estados estimados del sistema $\hat{X}_k^-$. Esto es

$$\hat{Z}_k^i = H(\hat{X}_k^-) \tag{2.34}$$

- Corrección

  Las dos siguientes ecuaciones corresponden a la corrección de la estimación de los estados y de la matriz de covarianza con respecto a las observaciones realizadas anteriormente.

$$\hat{X}_k^+ = \hat{X}_k^- + K_k(Z_k^i - \hat{Z}_k^i) \tag{2.35}$$

$$P_k^+ = (I - K_k C_k)P_k^- \tag{2.36}$$

Donde K_k es la ganancia del filtro EKF, dado como

$$K_k = P_k^- C_k (C_k P_k^- C_k + R_2)^{-1} \tag{2.37}$$

En la figura (2.3) se muestra un resumen del el nuevo algoritmo modos deslizantes SLAM, en este diagrama a diferencia del EKF mostramos un bloque que trata de la inyección discontinua (Sliding Mode, SM) al algoritmo EKF para darle robustez. Al igual que el diagrama del EKF este diagrama no representa en su totalidad el algoritmo SLAM debido a que no proporciona un mapa.

Esta nueva técnica nos proporciona robustez al problema de SLAM. A continuación, presentamos esta técnica como un algoritmo de modos deslizantes SLAM aplicado en un robot móvil.

Algoritmo modos deslizantes SLAM

En esta sección presentamos el algoritmo modos deslizantes SLAM aplicado para un robot móvil. Este algoritmo es flexible a diferentes modelos cambiando el modelo dinámico del robot móvil por el sistema en el que se necesita aplicar, de igual manera, se cambia el modelo de observación dependiendo de las características de interés.

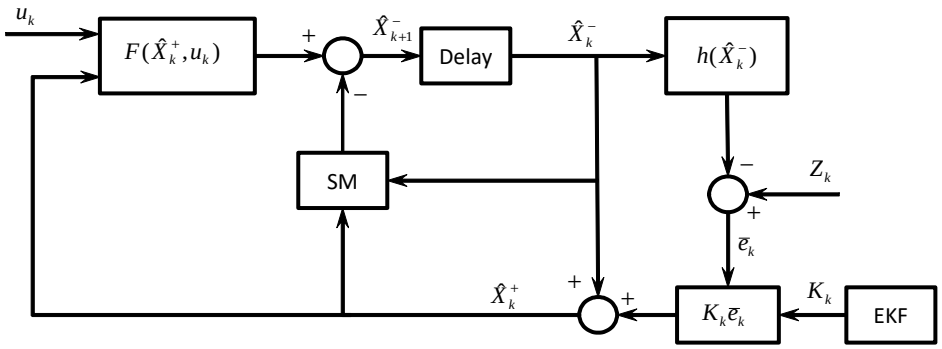

Figura 2.3: Diagrama del algoritmo modificado con la teoría de modos deslizantes.

Algoritmo 2.1 *Modos deslizantes SLAM*$(\hat{X}_k^-, \hat{X}_k^+, P_k^+, U_k, Z_k, \rho, N)$

$$G_x = \begin{pmatrix} 1 & 0 & 0 & \underbrace{0 \cdots 0}_{2N} \\ 0 & 1 & 0 & 0 \cdots 0 \\ 0 & 0 & 1 & 0 \cdots 0 \end{pmatrix}$$

$$e_k = \hat{X}_k^- - \hat{X}_k^+$$

$$\sigma_k = \rho\,[sgn(e_k)]$$

$$\hat{X}_{k+1}^- = \hat{X}_k^+ + G_x^T \begin{bmatrix} U_{k,v}\delta t \cos(\hat{X}_{k,\phi}^+) \\ U_{k,v}\delta t \sin(\hat{X}_{k,\phi}^+) \\ U_{k,\gamma}\delta t \end{bmatrix} - \sigma_k$$

$$A_k = I + G_x^T \begin{pmatrix} 0 & 0 & -u_{k,v}\delta t \sin(\hat{X}_{k,\phi}^+) \\ 0 & 0 & u_{k,v}\delta t \cos(\hat{X}_{k,\phi}^+) \\ 0 & 0 & 0 \end{pmatrix} G_x$$

$$P_{k+1}^- = A_k P_k^+ A_k^T + R_1$$

$$R_2 = \begin{pmatrix} \sigma_r & 0 \\ 0 & \sigma_\theta \end{pmatrix}$$

for *all observed landmark* $z_k^i = \left(r_k^i, \theta_k^i\right)^T$ ***do***

if *lanmark* j *never seen before* ***then***

$$\begin{pmatrix} m_{N+1,x} \\ m_{N+1,y} \end{pmatrix} = \begin{pmatrix} \hat{X}_x^- \\ \hat{X}_y^- \end{pmatrix} + r_k^j \begin{bmatrix} \cos(\theta_k^j + \hat{X}_{k,\phi}^-) \\ \sin(\theta_k^j + \hat{X}_{k,\phi}^-) \end{bmatrix}$$

$$P_{k,N+1}^- = \begin{bmatrix} P_{k+1}^- & 0 \\ 0 & R_2 \end{bmatrix}$$

$$P_{k,N+1}^+ = \nabla T P_{k,N+1}^- \nabla T^T$$

end if

$$G_x^i = \begin{pmatrix} 1 & 0 & 0 & \underbrace{0\cdots0}_{2i-2} & 0 & 0 & \underbrace{0\cdots0}_{2N-2i} \\ 0 & 1 & 0 & & 0 & 0 & \\ 0 & 0 & 1 & & 0 & 0 & \\ 0 & 0 & 0 & & 1 & 0 & \\ 0 & 0 & 0 & & 0 & 1 & \end{pmatrix}$$

$$\Psi = \begin{pmatrix} \Psi_x \\ \Psi_y \end{pmatrix} = \begin{pmatrix} m_x^i - \hat{X}_{k,x}^- \\ m_y^i - \hat{X}_{k,y}^- \end{pmatrix}$$

$$q = \Psi^T \Psi$$

$$\hat{z}_k^i = \begin{pmatrix} \sqrt{q} \\ \arctan\left(\dfrac{\Psi_y}{\Psi_x}\right) - \hat{X}_{k,\phi}^- \end{pmatrix}$$

$$C_k = \frac{1}{q} \begin{pmatrix} \sqrt{q}\Psi_x & \sqrt{q}\Psi_y & 0 & -\sqrt{q}\Psi_x & \sqrt{q}\Psi_y \\ -\Psi_y & -\Psi_y & -q & \Psi_y & \Psi_x \end{pmatrix} G_x^i$$

$$K_k = P_k^- C_k^T (C_k P_k^- C_k^T + R_2)^{-1}$$

$$\hat{X}_k^+ = \hat{X}_k^- + K_k(z_k^i - h(\hat{X}_k^-))$$

$$P_k^+ = (I - K_k C_k) P_k^-$$

end for

return $\hat{X}_k^-, \hat{X}_k^+, P_k^+$

A continuación, presentaremos la prueba de estabilidad del algoritmo modos deslizantes SLAM propuesto en este capítulo.

Prueba de estabilidad del algoritmo modos deslizantes SLAM.

En esta sección presentamos la prueba de estabilidad del algoritmo modos deslizantes SLAM. Donde reescribiendo el sistema no lineal de forma general en tiempo discreto, tenemos la ecuación de estados como:

$$X_{k+1} = \begin{bmatrix} f(X_k^r, U_k) + w_k \\ X^m \end{bmatrix} = F(X_k, U_k) + [w_k, 0]^T \tag{2.38}$$

El modelo de la observación se define de la siguiente manera

$$Z_k^i = H(X_k) + v_k^i \tag{2.39}$$

Ahora bien, si F y H son funciones C^1, entonces estas funciones pueden ser linealizados por la serie de Taylor

$$F(X_k, U_k) - F(\hat{X}_k^+, U_k) = A_k(X_k - \hat{X}_k^+) + O_1\left[(X_k - \hat{X}_k^+)^2\right] \tag{2.40}$$

$$H(X_k) - H(\hat{X}_k^-) = C_k(X_k - \hat{X}_k^-) + O_2\left[(X_k - \hat{X}_k^-)^2\right] \tag{2.41}$$

El algoritmo modos deslizantes SLAM es un observador no lineal especial. Definimos el error de estados como

$$\tilde{X}_k = X_k - \hat{X}_k^- \tag{2.42}$$

Si calculamos el error de estados para el tiempo $k+1$ tenemos

$$\tilde{X}_{k+1} = X_{k+1} - \hat{X}_{k+1}^- \tag{2.43}$$

Substituyendo (2.38) y (2.30) en la ecuación (2.42) y usando la ecuación (2.40) tenemos

$$\begin{aligned}
\tilde{X}_{k+1} &= F(X_k, U_k) + [w_k, 0]^T - F(\hat{X}_k^+, U_k) + \sigma_k \\
&= F(X_k, U_k) - F(\hat{X}_k^+, U_k) + [w_k, 0]^T + \sigma_k \\
&= A_k(X_k - \hat{X}_k^+) + O_1\left[(X_k - \hat{X}_k^+)^2\right] + [w_k, 0]^T + \sigma_k
\end{aligned}$$

Donde $\sigma_k = \rho sgn(e_k)$ es la compensación por modos deslizantes (2.31). Haciendo uso de la ecuación (2.35) con la ecuación (2.39) y posteriormente (2.41), esto es

$$\begin{aligned}
\tilde{X}_{k+1} &= A_k[X_k - \hat{X}_k^- - K_k(Z_k^i - H(\hat{X}_k^-))] + O_1\left[(X_k - \hat{X}_k^+)^2\right] + [w_k, 0]^T + \sigma_k \\
&= A_k(X_k - \hat{X}_k^-) - A_k K_k(Z_k^i - H(\hat{X}_k^-)) + O_1\left[(X_k - \hat{X}_k^+)^2\right] + [w_k, 0]^T + \sigma_k \\
&= A_k(X_k - \hat{X}_k^-) - A_k K_k(H(X_k) - H(\hat{X}_k^-) + v_k^i) + O_1\left[(X_k - \hat{X}_k^+)^2\right] + [w_k, 0]^T + \sigma_k \\
&= A_k(X_k - \hat{X}_k^-) - A_k K_k[C_k(X_k - \hat{X}_k^-) + O_2\left[(X_k - \hat{X}_k^-)^2\right] + v_k^i] + O_1\left[(X_k - \hat{X}_k^+)^2\right] + [w_k, 0]^T + \sigma_k \\
&= A_k(I - K_k C_k)(X_k - \hat{X}_k^-) - A_k K_k O_2\left[(X_k - \hat{X}_k^-)^2\right] + O_1\left[(X_k - \hat{X}_k^+)^2\right] - A_k K_k v_k^i + [w_k, 0]^T + \sigma_k
\end{aligned}$$

Sabemos que $\tilde{X}_k = X_k - \hat{X}_k^-$ entonces

$$\tilde{X}_{k+1} = A_k(I - K_k C_k)\tilde{X}_k + r_k + s_k + \sigma_k \tag{2.44}$$

Donde

$$r_k = O_1\left[(X_k - \hat{X}_k^+)^2\right] - A_k K_k O_2\left[(X, -\hat{X}_k^-)^2\right] \tag{2.45}$$

$$s_k = [w_k, 0]^T - A_k K_k v_k^i \tag{2.46}$$

Teorema 2.1 *Si las ganancias del algoritmo modos deslizantes SLAM*

$$\rho \geq \bar{r} + \bar{s} > 0 \tag{2.47}$$

donde $\bar{r}$ *y* $\bar{s}$ *son las incertidumbres acotadas en (2.44) como* $\|r_k\| \leq \bar{r}$, $\|s_k\| \leq \bar{s}$, *entonces la estimación del error es estable y el error converge a*

$$\left\|\tilde{X}_k\right\|^2 \leq \frac{\kappa}{p_{\text{mín}}\,\alpha} \tag{2.48}$$

donde $p_{\text{mín}} = \lambda_{\text{mín}}\left[\left(P_k^-\right)^{-1}\right]$, $\alpha = \frac{1}{(1+\bar{p}\bar{a}^2/\underline{q})(1+\bar{k}\bar{c})}$, $\underline{p}I \leq P_k^- \leq \bar{p}I$, $\underline{q}I \leq R_1$, $\kappa = \lambda_{\text{máx}}\left[\left(P_{k+1}^-\right)^{-1}\right](\rho^2 + \bar{r}^2 + \bar{s}^2)$, *y*

$$\lim_{n\to\infty}\left\|\tilde{X}_n\right\|^2 \leq \frac{\kappa}{p_{\text{máx}}\alpha} \tag{2.49}$$

con $p_{\text{mín}} = \lim_{n\to\infty}\left\{\lambda_{\text{máx}}\left[\left(P_{n+1}^-\right)\right]\right\}$

Demostración. Se propone una función de Lyapunov como

$$V_k = \tilde{X}_k\left(P_k^-\right)^{-1}\tilde{X}_k \tag{2.50}$$

donde P_k^- es la matriz de covarianza a priori (2.32), y $P_k^- > 0$. Nosotros podemos definir

$$V_{k+1}(\tilde{X}_{k+1}) = \tilde{X}_{k+1}^T\left(P_{k+1}^-\right)^{-1}\tilde{X}_{k+1}$$

Y sustituyendo (2.44) tenemos

$$\begin{aligned}
V_{k+1}(\tilde{X}_{k+1}) =& [A_k(I-K_kC_k)\tilde{X}_k + r_k + s_k + \sigma_k]^T\left(P_{k+1}^-\right)^{-1}[A_k(I-K_kC_k)\tilde{X}_k + r_k + s_k + \sigma_k] \\
=& (A_k(I-K_kC_k)\tilde{X}_k)^T\left(P_{k+1}^-\right)^{-1}(A_k(I-K_kC_k)\tilde{X}_k) \\
&+ s_k^T\left(P_{k+1}^-\right)^{-1}[2A_k(I-K_kC_k)\tilde{X}_k] + r_k^T\left(P_{k+1}^-\right)^{-1}[2A_k(I-K_kC_k)\tilde{X}_k] \\
&+ \sigma_k^T\left(P_{k+1}^-\right)^{-1}[2A_k(I-K_kC_k)\tilde{X}_k] + (s_k+r_k+\sigma_k)^T\left(P_{k+1}^-\right)^{-1}(s_k+r_k+\sigma_k)
\end{aligned}$$

Reduciendo términos

$$\begin{aligned}
&V_{k+1} = \tilde{X}_{k+1}\left(P_{k+1}^-\right)^{-1}\tilde{X}_{k+1} \\
&= \tilde{X}_k^T(I-K_kC_k)^TA_k^T\left(P_{k+1}^-\right)^{-1}A_k(I-K_kC_k)\tilde{X}_k \\
&+2(s_k^T + r_k^T + \sigma_k^T)\left(P_{k+1}^-\right)^{-1}\left[A_k(I-K_kC_k)\tilde{X}_k\right] \\
&+(s_k^T + r_k^T + \sigma_k^T)\left(P_{k+1}^-\right)^{-1}(s_k+r_k+\sigma_k)
\end{aligned} \tag{2.51}$$

Nosotros discutimos el primer término de la ecuación (2.51). De (2.36) nosotros tenemos

$$P_k^+ \geq (I - K_k C_k) P_k^- (I - K_k C_k)^T$$

Si $(I - K_k C_k)$ es invertible, entonces

$$\left(P_k^+\right)^{-1} \leq (I - K_k C_k)_k^{-T} \left(P_k^-\right)^{-1} (I - K_k C_k)^{-1} \tag{2.52}$$

De acuerdo a (2.32),

$$P_{k+1}^- = A_k P_k^+ A_k^T + Q = A_k (P_k^+ + A_k^{-1} R_1 A_k^{-T}) A_k^T$$

Entonces

$$\left(P_{k+1}^-\right)^{-1} = A_k^{-T} (P_k^+ + A_k^{-1} R_1 A_k^{-T})^{-1} A_k^{-1}$$

El siguiente lema de inversión de matrices

$$(\Gamma^{-1} + \Omega)^{-1} = \Gamma - \Gamma(\Gamma + \Omega^{-1})^{-1}\Gamma$$

donde Γ y Ω son dos matrices no singulares, tenemos

$$\left(P_{k+1}^-\right)^{-1} = A_k^{-T}[\left(P_k^+\right)^{-1} - \left(P_k^+\right)^{-1} (\left(P_k^+\right)^{-1} + A_k^T Q^{-1} A_k)^{-1} \left(P_k^+\right)^{-1}] A_k^{-1}$$

Usando (2.52) y definimos $L = (I - K_k C_k)$, obtenemos

$$\left(P_{k+1}^-\right)^{-1} \leq A_k^{-T} L^{-T}[\left(P_k^+\right)^{-1} - \left(P_k^+\right)^{-1} L^{-1}(\left(P_k^+\right)^{-1} + A_k^T R_1^{-1} A_k)^{-1} L^{-T} \left(P_k^+\right)^{-1}] L^{-1} A_k^{-1} \tag{2.53}$$

Multiplicando por izquierda $L^T A_k^T$ y por la derecha $A_k L$ en ambos lados de (2.53)

$$L^T A_k^T \left(P_{k+1}^-\right)^{-1} A_k L \leq (I - \left(P_k^-\right)^{-1} L^{-1} \left(P_k^+\right)^{-1} (I + \left(P_k^+\right)^{-1} A_k^T Q^{-1} A_k)^{-1} L^{-T}) \left(P_k^-\right)^{-1}$$

Ahora

$$\left(P_k^+\right)^{-1} = \left(P_k^-\right)^{-1} (I - K_k C_k)^{-1} = \left(P_k^-\right)^{-1} L^{-1}$$

Por lo tanto

$$L^T A_k^T \left(P_{k+1}^-\right)^{-1} A_k L \leq (I - (I + \left(P_k^+\right)^{-1} A_k^T Q^{-1} A_k)^{-1} L^{-T}) \left(P_k^-\right)^{-1}$$

1) Entonces el primer término del lado derecho de la ecuación (2.51) es

$$\begin{aligned} &(I - K_k C_k)^T A_k^T \left(P_{k+1}^-\right)^{-1} A_k (I - K_k C_k) \\ &\leq (1 - (1 + \bar{p}\bar{a}^2/\underline{q})^{-1}(1 + \bar{k}\bar{c})^{-1}) \left(P_k^-\right)^{-1} \\ &\leq (1 - \alpha) \left(P_k^-\right)^{-1} \end{aligned} \tag{2.54}$$

donde $\|A_k\| = \sqrt{tr(A_k A_k)} \leq \bar{a}$, $\|C_k\| = \sqrt{tr(C_k C_k)} \leq \bar{c}$, $\|K_k\| = \sqrt{tr(K_k K_k)} \leq \bar{k}$, y

$$0 < \alpha = \frac{1}{(1 + \bar{p}\bar{a}^2/\underline{q})(1 + \bar{k}\bar{c})} < 1$$

$\underline{p}I \leq P_k^- \leq \bar{p}I$, $\underline{q}I \leq R_1$.

2) Por (2.31),

$$\sigma_k^T \tilde{X}_k = -\rho sgn\left[\tilde{X}_k\right] \tilde{X}_k = -\rho \left\|\tilde{X}_k\right\|$$

El segundo término del lado derecho de la ecuación (2.51) es

$$\begin{aligned} &2(s_k^T \tilde{X}_k + r_k^T \tilde{X}_k + \sigma_k^T \tilde{X}_k) \left\{ \left(P_{k+1}^-\right)^{-1} \left[A_k(I - K_k C_k)\right] \right\} \\ &\leq 2(\|s_k\| + \|r_k\| - \rho) \left\|\tilde{X}_k\right\| \left\| \left(P_{k+1}^-\right)^{-1} \left[A_k(I - K_k C_k)\tilde{X}_k\right] \right\| \end{aligned} \tag{2.55}$$

De la condición (2.47), el segundo término del lado derecho de la ecuación (2.51) es negativa. Porque $\|\sigma_k\|^2 \leq \rho^2$, $\|s_k\|^2 \leq \bar{r}^2$, $\|r_k\|^2 \leq \bar{s}^2$.

3) El último término del lado derecho de la ecuación (2.51)

$$\begin{aligned} &(s_k^T + r_k^T + \sigma_k^T) \left(P_{k+1}^-\right)^{-1} (s_k + r_k + \sigma_k) \\ &\leq \lambda_{\text{máx}} \left[\left(P_{k+1}^-\right)^{-1}\right] (\rho^2 + \bar{r}^2 + \bar{s}^2) \end{aligned}$$

Donde $\left(P_{k+1}^-\right)^{-1}$ es positiva y decrece, además acotada. Entonces

$$(s_k^T + r_k^T + \sigma_k^T) \left(P_{k+1}^-\right)^{-1} (s_k + r_k + \sigma_k) \leq \kappa \tag{2.56}$$

donde $\kappa = \lambda_{\text{máx}} \left[\left(P_{k+1}^-\right)^{-1}\right] (\rho^2 + \bar{r}^2 + \bar{s}^2)$, $\lambda_{\text{máx}} \left[\left(P_{k+1}^-\right)^{-1}\right]$ es el máximo eigenvalor de $\left(P_{k+1}^-\right)^{-1}$

4) Combinando (2.51), (2.54), (2.55) y (2.56), para $k+1$ tenemos

$$\begin{aligned} V_{k+1} &= \tilde{X}_{k+1} \left(P_{k+1}^-\right)^{-1} \tilde{X}_{k+1} \\ &\leq (1 - \alpha)\tilde{X}_k \left(P_k^-\right)^{-1} \tilde{X}_k + \kappa \\ &= \kappa + (1 - \alpha)V_k \end{aligned} \tag{2.57}$$

Entonces

$$V_{k+1} - V_k \leq -\alpha V_k + \kappa$$

Donde $\alpha V_k \geq \alpha \lambda_{\text{mín}} \left[\left(P_k^-\right)^{-1}\right] \left\|\tilde{X}_k\right\|^2$, si

$$\alpha \lambda_{\text{mín}} \left[\left(P_k^-\right)^{-1}\right] \left\|\tilde{X}_k\right\|^2 \geq \kappa$$

Por lo tanto, $V_{k+1} - V_k \leq 0$, V_k (o $\left\|\tilde{X}_k\right\|$) decrece. Y entonces $\left\|\tilde{X}_k\right\|$ converge a (2.48).

Si queremos saber cómo evoluciono con V_{k+1}, sabemos que para k seria $V_k \leq k + (1-\alpha)V_{k-1}$, entonces para V_{k+1}

$$V_{k+1} = \kappa + (1-\alpha)k + (1-\alpha)^2 V_{k-1}$$

y para $k-1$, $V_{k-1} \leq k + (1-\alpha)V_{k-2}$, resulta

$$V_{k+1} = (1-\alpha)^0\kappa + (1-\alpha)k + (1-\alpha)^2 k + (1-\alpha)^3 V_{k-2}$$

Si esto se aplica de forma recurrente hasta tener $V_{k+1}(V_0)$, es decir, tener todas las anteriores,

$$V_{k+1} \leq \kappa \sum_{i=0}^{k} (1-\alpha)^i + (1-\alpha)^{k+1} V_0$$

En resumen

$$\begin{array}{ll} V_{k+1} = \kappa + (1-\alpha)V_k & V_k = \kappa + (1-\alpha)V_{k-1} \\ V_{k+1} = \kappa + (1-\alpha)k + (1-\alpha)^2 V_{k-1} & V_{k-1} = \kappa + (1-\alpha)V_{k-2} \\ V_{k+1} = (1-\alpha)^0\kappa + (1-\alpha)k + (1-\alpha)^2 k + (1-\alpha)^3 V_{k-2} & V_{k-1} = \kappa + (1-\alpha)V_{k-3} \end{array}$$

$$V_{k+1} \leq \kappa \sum_{i=0}^{k} (1-\alpha)^i + (1-\alpha)^{k+1} V_0$$

Si la función de Lyaponov es acotada

$$\lambda_{\text{mín}}\left[\left(P_{n+1}^{-}\right)^{-1}\right]\left\|\tilde{X}_{n+1}\right\|^2 \leq V_{n+1} \leq \lambda_{\text{mín}}\left[\left(P_{n+1}^{-}\right)^{-1}\right]\left\|\tilde{X}_{n+1}\right\|^2$$

Con $\lambda_{\text{mín}}\left[\left(P_{n+1}^{-}\right)^{-1}\right]$ que es el mínimo eigenvalor de $\left(P_{n+1}^{-}\right)^{-1}$, entonces

$$\lambda_{\text{mín}}\left[\left(P_{n+1}^{-}\right)^{-1}\right]\left\|\tilde{X}_{n+1}\right\|^2 \leq \kappa \sum_{i=0}^{n} (1-\alpha)^i + (1-\alpha)^{n+1} V_0$$

Y si evaluamos que para cuando $n \to \infty$, resulta que

$$\lim_{k\to\infty} V_{k+1} \leq \frac{\kappa}{p_{\text{mín}}\alpha}$$

Es decir

$$\lim_{n\to\infty}\left\|\tilde{X}_{n+1}\right\|^2 \leq \frac{\kappa}{p_{\text{mín}}\alpha}$$

Donde $p_{\text{mín}} = \lim_{n\to\infty}\left\{\lambda_{\text{mín}}\left[\left(P_{n+1}^{-}\right)^{-1}\right]\right\}$. Con $0 < \alpha < 1$ que está dada como como $\alpha = \frac{1}{(1+\bar{p}\bar{a}^2/\underline{q})(1+\bar{k}\bar{c})}$, y $\lambda_{\text{mín}}\left[\left(P_{n+1}^{-}\right)^{-1}\right]$ es una constante donde $n \to \infty$, que es (2.49). Por lo tanto, el error $\tilde{X}_k$ converge. ■

Modos deslizantes SLAM para un UAV

El sistema UAV (Unmanned Aerial Vehicle) es de gran interés para los investigadores en la actualidad. Estos sistemas tienen diversas aplicaciones, tales como: la exploración de campos de cultivo en la detección de plagas, búsqueda de fallas construcciones, edificios o puentes, en vigilancia, entre otras. El sistema UAV más utilizada es el cuadricóptero debido a que pueden ser manipuladas en todas las direcciones, es decir, tienen menos restricciones en el sentido de desplazamientos. El desempeño de la navegación autónoma del sistema cuadricóptero depende de la precisión de la localización y la precisión con la que se construye el entorno (un mapa). Este es un problema conocido como SLAM (localización y mapeo simultáneos) donde el cuadricóptero tiene que estimar un mapa del entorno desconocido mientras que al mismo tiempo se localiza con respecto a este mapa.

Modelo del cuadricóptero para el SLAM

Cuando el cuadricóptero está navegando en el espacio, tenemos la posición absoluta y la altura que se describe como $X^d = [x, y, z]^T$ y los ángulos de Euler $\Theta = [\phi, \theta, \psi]^T$ sistema de coordenadas fijo a tierra con origen O; ubicación de inicio definida. En este trabajo, el algoritmo SLAM se implementa para estimar la posición y orientación del cuadricóptero $X_k^r = [X^d, \Theta]^T$ y construir un mapa X^m a partir de la navegación del cuadricóptero. Por consiguiente, los Landmarks o marcas fijas pertenecen al vector de estados. Dicha combinación de los estados está dada por

$$\begin{aligned} X_k &= [x, y, z, \phi, \theta, \psi, m_x^1, m_y^1, m_z^1, ..., m_x^N, m_y^N, m_z^N]^T \\ &= \begin{bmatrix} X_k^r \\ X^m \end{bmatrix} \end{aligned}$$

Donde $X^m = [m_x^1, m_y^1, m_z^1, ..., m_x^N, m_y^N, m_z^N]^T$ representa el mapa del entorno, tenga en cuenta que X^m no depende del tiempo porque los puntos de referencia son estacionarios. La dimensión del vector de estado X_k tiene una dimensión de $(6N+6)$, aquí, N corresponde al número de puntos de referencia en el mapa. La figura (2.4) muestra las propiedades

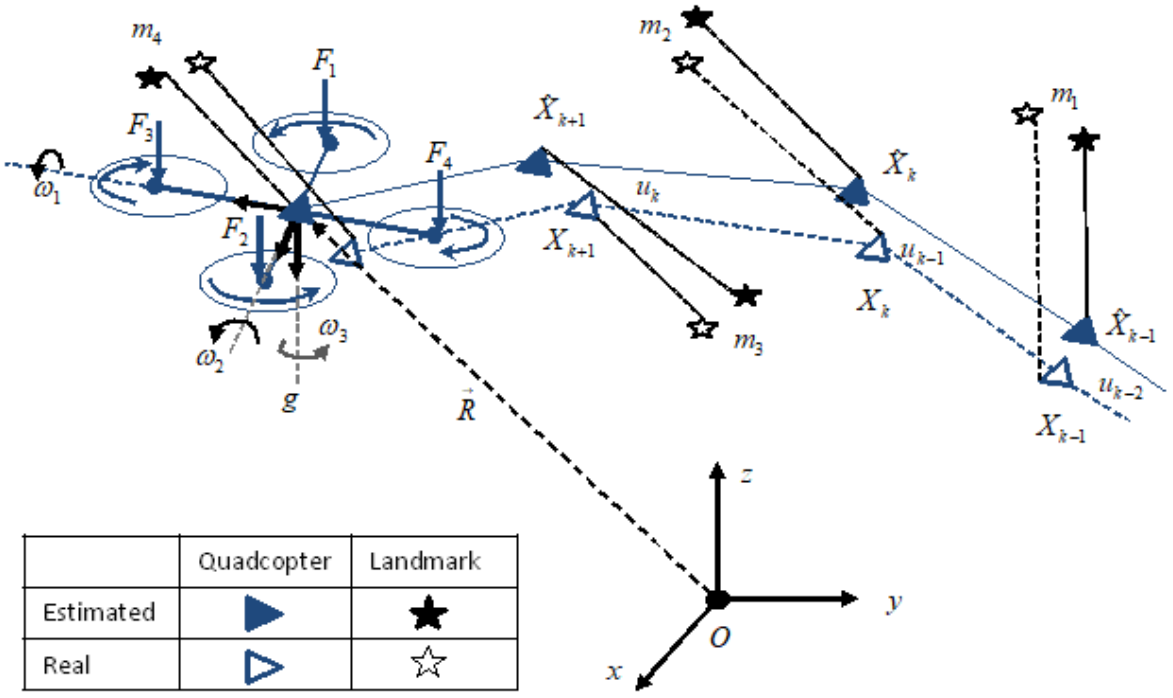

Figura 2.4: Algoritmo SLAM para un cuadricóptero.

del cuadricóptero y la estimación realizada por el SLAM en cada iteración Δt con las entradas U. El SLAM tiene tres etapas: la predicción que consiste en estimar los estados a partir del modelo matemático, la observación donde se obtienen las características de los puntos de referencia (Landmarks) y, finalmente, la actualización que produce un ajuste a los estados estimados a partir de la observación.

El modelo matemático que se utiliza para realizar la SLAM corresponde a la cinemática del sistema. El modelo de navegación del cuadricóptero se presenta en la siguiente ecuación

$$f(X_k^r, U_k) = \begin{bmatrix} X_k^d \\ \Theta_k \end{bmatrix} = \begin{bmatrix} X_{k-1}^d + R(\Theta_{k-1})\beta_k \Delta t \\ \Theta_{k-1} + \vec{M}(\Theta_{k-1})\zeta_k \Delta t \end{bmatrix} \tag{2.58}$$

Donde $\beta_k = [u_k, v_k, w_k]$ y $\zeta_k = [p_k, q_k, r_k]$ son las velocidades lineales y angulares del cuadricóptero en el marco del cuerpo, respectivamente. Con $U_k = [\beta_k, \zeta_k]^T$ y $k \in N_0$ es el tiempo discreto. Además, la matriz de rotación ortogonal $R(\Theta)$ se define como

$$R(\Theta) = \begin{bmatrix} C_\psi C_\theta & C_\psi S_\theta S_\phi - S_\psi C_\phi & C_\psi S_\theta C_\phi + S_\psi S_\phi \\ S_\psi C_\theta & S_\psi S_\theta S_\phi - C_\psi C_\phi & S_\psi S_\theta C_\phi - C_\psi S_\phi \\ -S_\theta & C_\theta S_\phi & C_\psi C_\theta \end{bmatrix} \tag{2.59}$$

Donde $cos(\star) = C_\star$ y $sin(\star) = S_\star$. Además, $\vec{M}(\Theta)$ tiene las siguiente forma:

$$\vec{M}(\Theta) = \begin{bmatrix} 1 & \frac{S_\phi S_\theta}{C_\phi C_\theta^2 + C_\theta S_\phi^2} & -\frac{S_\theta}{S_\phi^2 + C_\phi C_\theta} \\ 0 & \frac{C_\phi}{S_\phi^2 + C_\phi C_\theta} & \frac{-S_\phi}{S_\phi^2 + C_\phi C_\theta} \\ 0 & -\frac{S_\phi}{C_\phi C_\theta^2 + C_\theta S_\phi^2} & \frac{S_\theta}{S_\phi^2 + C_\phi C_\theta} \end{bmatrix} \tag{2.60}$$

Para el diseño del algoritmo SM-SLAM, se deben considerar las siguientes restricciones:

- Restricción 1: $U = [u\ v\ w\ p\ q\ r]$ son las velocidades del cuadricóptero en el marco del cuerpo y pueden medirse mediante un sensor a bordo del sistema o estimarse.
- Restricción 2: Los ángulos tienen un límite de operación: *pitch* $\phi \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$ y *roll* $\theta \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$.
- Restricción 3: Las perturbaciones dentro del sistema deben estar acotadas.
- Restricción 4: $f(\cdot,\cdot)$ y $h(\cdot)$ son funciones diferenciables donde f es el modelo del cuadricóptero y h es el modelo de la observación.

Algoritmo Modos Deslizantes SLAM para el cuadricóptero

En esta sección, describimos el algoritmo SM-SLAM. Los modos deslizantes SLAM se refiere de un algoritmo robusto ante perturbaciones acotadas y con sistema no lineales dados como

$$X_{k+1} = \begin{bmatrix} f(X_k^r, U_k) + w_k \\ X^m \end{bmatrix} = F(X_k, U_k) + [w_k, 0]^T \tag{2.61}$$

Donde w_k son los ruidos del sistema, y $f(\cdot,\cdot)$ es el modelo del cuadricóptero presentado anteriormente (2.58). El algoritmo SM-SLAM se divide en tres procesos (Predicción, Observación y Ajuste) que se explican a continuación.

Proceso de predicción

El algoritmo SM SLAM estima los estados del sistema (2.61). Por lo tanto, proponemos una estimación de los estados $\hat{X}_{k+1}$ de la siguiente manera.

$$\begin{aligned}\hat{X}_{k+1} &= F(\hat{X}_k^+, U_k) - \rho \cdot sgn\,[e_k] \\ &= \hat{X}_k + G_x^T f(\hat{X}_k^r, U_k) - \rho \cdot sgn\,[e_k]\end{aligned} \tag{2.62}$$

Donde $e_k = \hat{X}_k - \hat{X}_k^+$ son los errores de estimación, aquí $\hat{X}_k^+$ son los estados estimados ajustados que se definen en (2.75).$\rho > \rho_1 \cdot \Delta t > 0$ son ganancias de ajuste con tiempo discreto Δt. Y el sgn es un término de inyección, definido por:

$$sgn\,[e_k] = \frac{e_k}{\|e_k\|} = \begin{cases} 1 & \text{if } e_k \geq 0 \\ -1 & \text{if } e_k < 0 \end{cases} \tag{2.63}$$

Además, G_x es una matriz que mapea el vector de estado de 6-dimensiones a la dimensión $6N + 6$. Definida como

$$G_x = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \cdots 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \cdots 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \cdots 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \cdots 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \cdots 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & \underbrace{0 \cdots 0}_{6N} \end{bmatrix}$$

Si el sistema (2.61) se aplica una extrapolación lineal, la ecuación se puede expresar como:

$$X_{k+1} = F(\hat{X}_k^+, U_k) + A_k(X_k - \hat{X}_k^+) + O_1\left[(X_k - \hat{X}_k^+)^2\right] + [w_k, 0]^T \tag{2.64}$$

Donde O_1 son funciones no lineales A_k es la matriz jacobiana con dimensión $n \times n$, donde $n = 6N + 6$ que denota la dimensión de los estados. La dimensión aumenta con respecto al número de Landmarks que se observan en el entorno. Entonces

$$\begin{aligned}A_k = \nabla F(\cdot,\cdot) &:= \frac{\partial F(X_k, U_k)}{\partial X_k} \mid_{X_k = \hat{X}_k^+} \\ &= I_{n\times n} + G_x^T \nabla f(\hat{X}_k^r, U_k) G_x\end{aligned} \tag{2.65}$$

Aquí, $\nabla f(\hat{X}_k^r, U_k) \in R^{6x6}$ es la matriz jacobiana del cuadricóptero (2.58) donde la matriz $G_x \in R^{6\times n}$ se requiere para mapear las dimensiones de la matriz jacobiana A_k. Específicamente para el sistema cuadricóptero; La matriz jacobiana es la siguiente.

$$\nabla f(\hat{X}_k^r, U_k) = \frac{\partial f(X_k^r, U_k)}{\partial X_k^r} \mid_{X_k^r = \hat{X}_k^r} = [O_{6\times 3}, \Delta t M_D^T] \tag{2.66}$$

con

$$M_D = [D_x\beta^T, D_y\beta^T, D_z\beta^T, D_\phi\zeta^T, D_\theta\zeta^T, D_\psi\zeta^T] \tag{2.67}$$

Donde las matrices: $D_x = \frac{\partial f(X_k^d, U_k)}{\partial X_k^x}$, $D_y = \frac{\partial f(X_k^d, U_k)}{\partial X_k^y}$, $D_z = \frac{\partial f(X_k^d, U_k)}{\partial X_k^z}$, $D_\phi = \frac{\partial f(\Theta_k, U_k)}{\partial X_k^\phi}$, $D_\theta = \frac{\partial f(\Theta_k, U_k)}{\partial X_k^\theta}$, y $D_\psi = \frac{\partial f(\Theta_k, U_k)}{\partial X_k^\psi}$ se definen como

$$D_x = \begin{bmatrix} 0 & C_\phi S_\theta C_\psi + S_\phi S_\psi & C_\phi S_\psi - S_\phi S_\theta C_\psi \\ -S_\theta C_\psi & S_\phi C_\theta C_\psi & C_\phi C_\theta C_\psi \\ -C_\theta S_\psi & -S_\phi S_\theta S_\psi - C_\phi C_\psi & S_\phi S_\psi - C_\phi S_\theta S_\psi \end{bmatrix}$$

$$D_y = \begin{bmatrix} 0 & C_\phi S_\theta S_\psi - S_\phi C_\psi & S_\phi S_\theta S_\psi - C_\phi C_\psi \\ -S_\theta S_\phi & S_\phi C_\theta S_\psi & C_\phi C_\theta S_\psi \\ C_\theta C_\psi & S_\phi S_\theta C_\psi - C_\phi S_\psi & S_\phi S_\psi + C_\phi S_\theta C_\psi \end{bmatrix}$$

$$D_z = \begin{bmatrix} 0 & C_\phi C_\theta & -S_\phi C_\theta \\ -C_\theta & -S_\phi S_\theta & -C_\phi S_\theta \\ 0 & 0 & 0 \end{bmatrix}$$

$$D_\phi = \begin{bmatrix} 0 & \frac{S_\theta}{C_\theta(1+C_\theta C_\phi - C_\phi^2)^2} & \frac{S_\theta(C_\theta S_\phi - 2C_\phi S_\phi)}{(S_\phi^2 + C_\theta C_\phi)^2} \\ 0 & \frac{S_\phi(2C_\phi C_\theta - C_\phi^2 - C_\phi C_\theta^3 + 1)}{C_\theta^2(1+C_\theta C_\phi - C_\phi^2)^2} & \frac{C_\phi + C_\theta - C_\phi^2 C_\theta}{(1+C_\theta C_\phi - C_\phi^2)^2} \\ 0 & 0 & 0 \end{bmatrix}$$

$$D_\theta = \begin{bmatrix} 0 & -\frac{S_\phi(2C_\phi^2 + S_\phi^2)}{(S_\phi^2 + C_\phi C_\theta)^2} & -\frac{C_\theta - C_\phi + C_\phi^3}{(1 + C_\theta C_\phi - C_\phi^2)^2} \\ 0 & \frac{C_\phi^2 S_\theta}{(S_\phi^2 + C_\phi C_\theta)^2} & -\frac{C_\phi S_\phi S_\theta}{(S_\phi^2 + C_\phi C_\theta)^2} \\ 0 & 0 & 0 \end{bmatrix}$$

$$D_\psi = \begin{bmatrix} 0 & \frac{C_\theta - C_\phi + C_\phi^3}{C_\theta(1 + C_\theta C_\phi - C_\phi^2)^2} & \frac{2C_\phi S_\phi + C_\theta S_\phi}{(S_\phi^2 + C_\theta C_\phi)^2} \\ 0 & \frac{S_\phi(2C_\phi C_\theta S_\theta + S_\phi^2 S_\theta)}{(C_\phi C_\theta^2 + C_\theta S_\phi^2)^2} & \frac{C_\phi S_\theta}{(S_\phi^2 + C_\theta C_\phi)^2} \\ 0 & 0 & 0 \end{bmatrix}$$

Por otra parte, en la predicción, la matriz P de covarianza se calcula por

$$P_{k+1} = A_k P_k^+ A_k^T + Q \tag{2.68}$$

Donde P_k^+ es la matriz de covarianza ajustada y Q covarianza del ruido en el sistema.

modelo de observación

Se cuantifican las observaciones del cuadricóptero. El modelo matemático de esta observación se da como:

$$Z_k = H(X_k) + v_k \tag{2.69}$$

Donde Z_k es la salida. $H(\cdot)$ función no lineal que representa las características de la observación. v_k son ruidos de observación. Cabe mencionar, el modelo (2.69) representa

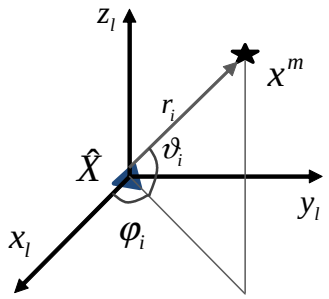

Figura 2.5: Estimación de la observación.

todas las observaciones del sistema hechas en el tiempo k. De una manera específica, para el $i-th$ Landmark observado en el tiempo k, el modelo de observación es el siguiente.

$$\begin{aligned} Z_k^i = h(X_k^r) + v_k^i &= \begin{bmatrix} r_i \\ \varphi_i \\ \vartheta_i \end{bmatrix} + v_k^i \\ &= \begin{bmatrix} \sqrt{\tilde{x}^2 + \tilde{y}^2 + \tilde{z}^2} \\ \tan^{-1}\left(\dfrac{\tilde{y}}{\tilde{x}}\right) \\ \tan^{-1}\left(\dfrac{\tilde{z}}{\sqrt{\tilde{x}^2 + \tilde{y}^2}}\right) \end{bmatrix} + v_k^i \end{aligned} \tag{2.70}$$

Donde $\tilde{x} = x_l - x$, $\tilde{y} = y_l - y$ y $\tilde{z} = z_l - z$ con $x^m = [x_l, y_l, z_l]$ posición de los Landmarks observados dentro del entorno. En consecuencia, r_i es la distancia de la observación con las direcciones de φ_i y ϑ_i obtenidas entre la posición estimada del cuadricóptero y la posición del $i-th$ Landmark observado, como se observa en la figura (2.5).

Si la observación del modelo (2.69) se aplica una extrapolación lineal, la ecuación se puede expresar como:

$$Z_k^i = H(\hat{X}_k) + C_k(X_k - \hat{X}_k) + O_2\left[(X_k - \hat{X}_k)^2\right] + v_k^i \tag{2.71}$$

Donde O_2 son funciones no lineales, C_k es la matriz jacobiana del sistema de observación con dimensión $6 \times n$ y se evalúa en el punto de operación $\hat{X}_k$ para el $i-$th Landmark. Tenemos que

$$\begin{aligned} C_k = \nabla H(\cdot,\cdot) &:= \frac{\partial H(X_k)}{\partial X_k} \mid_{X_k = \hat{X}_k} \\ &= \nabla h(X_k^r) G_y^i \end{aligned} \tag{2.72}$$

Aquí $\nabla h(X_k^r) \in R^{6\times 12}$ es la matriz jacobiana del modelo de observación (2.70). Donde el ajuste se realice con respecto al $i-th$ Landmark observado; por lo tanto, la matriz $G_y \in R^{12\times n}$ se requiere para mapear las dimensiones de la matriz jacobiana C_k en el Landmar i observado, dado como

$$G_y^i = \begin{bmatrix} I_{6\times 6} & \underbrace{0\cdots 0}_{6i-6} & 0_{6\times 6} & \underbrace{0\cdots 0}_{6N-6i} \\ 0_{6\times 6} & & I_{6\times 6} & \end{bmatrix}$$

Entonces el jacobiano del modelo de observación (2.70) para el cuadricóptero se obtiene como

$$\nabla h(X_k^r) = \frac{\partial h(X_k^r)}{\partial X_k^r} \mid_{X_k^r = \hat{X}_k^r} \tag{2.73}$$

Donde

$$\nabla h(X_k^r) = \frac{1}{q}\begin{bmatrix} S_1 & 0_{3\times 3} & S_2 & 0_{3\times 3} \\ 0_{3\times 3} & 0_{3\times 3} & 0_{3\times 3} & 0_{3\times 3} \end{bmatrix} \tag{2.74}$$

Con matrices $S_1 = \dfrac{\partial h(X_k^r)}{\partial X_k^m}$ y $S_2 = \dfrac{\partial h(X_k^r)}{\partial X_k^d}$ definidos por

$$S_1 = \begin{bmatrix} -\tilde{x}\sqrt{q} & -\tilde{y}\sqrt{q} & -\tilde{z}\sqrt{q} \\ q\dfrac{\tilde{y}}{m} & -q\dfrac{\tilde{x}}{m} & 0 \\ \tilde{x}\dfrac{\tilde{z}}{\sqrt{m}} & \tilde{x}\dfrac{\tilde{z}}{\sqrt{m}} & -\sqrt{m} \end{bmatrix}$$

$$S_2 = \begin{bmatrix} \tilde{x}\sqrt{q} & \tilde{y}\sqrt{q} & \tilde{z}\sqrt{q} \\ -q\dfrac{\tilde{y}}{m} & q\dfrac{\tilde{x}}{m} & 0 \\ -\tilde{x}\dfrac{\tilde{z}}{\sqrt{m}} & -\tilde{y}\dfrac{\tilde{z}}{\sqrt{m}} & \sqrt{m} \end{bmatrix}$$

Donde $q = \tilde{x}^2 + \tilde{y}^2 + \tilde{z}^2$ y $m = \tilde{x}^2 + \tilde{y}^2$.

Proceso de ajuste

Asumiendo que las características de la observación z_k^i han sido obtenidas por los sensores, entonces la estimación de los estados $\hat{X}_k$ son ajustados a partir de esta observación, esto es

$$\hat{X}_k^+ = \hat{X}_k + K_k(z_k^i - \hat{z}_k^i) \tag{2.75}$$

Donde $\hat{X}_k^+$ son los estados estimados ajustados, $\hat{z}_k^i$ es la estimación de la observación con respecto a los estados estimados del cuadricóptero $\hat{X}_k$. K_k es una matriz de ganancia de Kalman, dada por

$$K_k = P_k C_k^T (C_k P_k C_k^T + R)^{-1} \tag{2.76}$$

Y R es la covarianza del ruido de observación. De la misma manera, el ajuste de la matriz de covarianza P se define de la siguiente forma:

$$P_k^+ = (I - K_k C_k) P_k \tag{2.77}$$

El proceso del algoritmo SM-SLAM es recurrente en la navegación del cuadricóptero, es decir, las etapas presentadas del SM se realizan de forma iterativa: predicción con las ecuaciones (2.62) y (2.68), observación con la ecuación (2.70) y ajuste con las ecuaciones (2.75) y (2.77).

El algoritmo SLAM de modo deslizante se muestra a continuación.

Algoritmo 2.2 *Sliding Mode -SLAM*($\hat{X}_k, \hat{X}_k^+, U_k, P_k, P_k^+, \rho, z_k$)

$e_k = \hat{X}_k - \hat{X}_k^+$
$\sigma_k = \rho \cdot sgn(e_k)$
$\hat{X}_{k+1} = \hat{X}_k^+ + F(\hat{X}_k^+, U_k) - \sigma_k$
$A_k = \nabla F(\hat{X}_k^+, U_k)$
$P_{k+1} = A_k P_k^+ A_k^T + Q$
for *all observed landmark* $Z_k^i = \left(r_k^i, \varphi_k^i, \vartheta_k^i\right)^T$ ***do***
  if *landmark* j *never seen before* ***then***

$$X_{N+1,k}^m = \begin{pmatrix} \hat{X}_x \\ \hat{X}_y \\ \hat{X}_z \end{pmatrix} + r_k^j \begin{bmatrix} \cos(\vartheta_k^j)\cos(\varphi_k^j) \\ \cos(\vartheta_k^j)\sin(\varphi_k^j) \\ \sin(\vartheta_k^j) \end{bmatrix}$$

  end if
  $\hat{Z}_k^i = H(\hat{X}_k)$
  $C_k = \nabla H(\hat{X}_k)$
  $K_k = P_k C_k^T (C_k P_k^- C_k^T + R)^{-1}$
  $\hat{X}_k^+ = \hat{X}_k + K_k(z_k^i - \hat{z}_k^i)$
  $P_k^+ = (I - K_k C_k) P_k$
end for
return $\hat{X}_k, \hat{X}_k^+, P_k^+$

Los resultados obtenidos de los algoritmos mencionados anteriormente se presentan en la siguiente sección: primero, se presentan resultados del SLAM en un sistema de robot

móvil, el segundo resultado muestran la respuesta del SLAM en un sistema UAV y por último los resultados experimentales.

2.3. Resultados

En esta sección presentamos los resultados obtenidos de simulación utilizando el modelo de un robot móvil y el modelo de un Cuadricóptero. Posteriormente, mostramos resultados experimentales de un robot móvil. Las simulaciones que se presentan a continuación fueron realizadas en Matlab 2015.

Resultados del SLAM para un robot móvil.

Los resultados del algoritmo modos deslizantes SLAM que se presentan a continuación son comparadas con los resultados del EKF SLAM tradicional. De tal manera, poder comparar el desempeño de cada algoritmo.

El sistema (2.5) se utiliza para realizar las simulaciones con una señal de excitación $U_k = [2 - 2\cos(0{,}5k), \cos(4k)]^T$. Con $L = 0{,}5m$. Y las matrices

$$R_1 = \begin{bmatrix} 0{,}5 & 0 & 0 \\ 0 & 0{,}5 & 0 \\ 0 & 0 & 0{,}5 \end{bmatrix}$$

$$R_2 = \begin{bmatrix} 0{,}5 & 0 & 0 \\ 0 & 0{,}5 & 0 \\ 0 & 0 & 0{,}5 \end{bmatrix}$$

Para las simulaciones los estados de la estimación y las matrices de covarianza se inicializan en cero.

En la figura (2.6) se muestra los resultados de la localización del robot móvil, así como la ubicación de las marcas fijas. Los puntos representan los obstáculos y los círculos son las marcas fijas que se detectan durante la navegación. Podemos notar visualmente que la respuesta del algoritmo modos deslizantes SLAM se aproxima al valor real comparada con el resultado del EKF.

En la figura (2.7) se muestran las respuestas de cada estado. En los estados x y ϕ se puede notar que las señales tanto el valor real como las estimaciones hechas por el

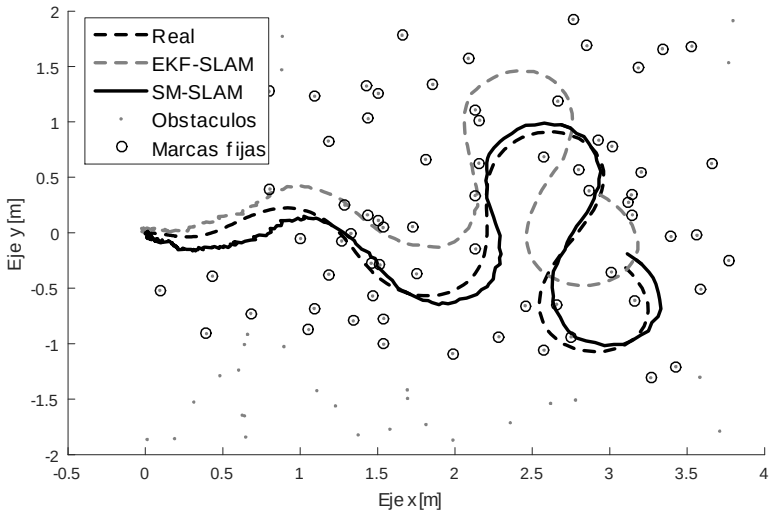

Figura 2.6: Simulación de la navegación de un robot móvil.

algoritmo EKF y con modos deslizantes tienen un desvío pequeño de la señal real, sin embargo en el estado y del EKF su desvio es muy pronunciado con respecto al valor real y del SM-SLAM. Las conclusiones son de manera visual.

Una de las características que se mencionan para el algoritmo EKF es que deben presentar distribuciones Gaussianas, sin embargo, esta característica no se cumple para este problema, por ello, presenta errores en su estimación de los estados. En la figura (2.8) presentamos las distribuciones de los errores en los estados del robot móvil donde se puede observar que estas distribuciones no son de tipo Gaussiana por lo que no cumple con las restricciones del algoritmo EKF, por otra parte, si cumplen con las restricciones del algoritmo SM SLAM donde pide que los errores sean acotadas.

Por otro lado, los ruidos w_k y v_k que son introducidas en las simulaciones para el SLAM se presentan en las figuras (2.9) y (2.10), donde estos ruidos se puede observar que son acotadas, es decir, que no son crecientes.

Ahora bien, los resultados presentados fueron evaluados de manera visual, en consecuencia, es necesario una evaluación numérica del desempeño de los algoritmos planteados, para realizar esta tarea utilizamos dos ecuaciones (2.78) y (2.79): la primera ecuación trata del promedio de la distancia Euclidianas y la segunda ecuación es el promedio del valor absoluto de la orientación del robot móvil.

$$E_d = \frac{1}{N_T} \sum_{k=1}^{N_T} \sqrt{(x_k - x_k^*)^2 + (y_k - y_k^*)^2} \tag{2.78}$$

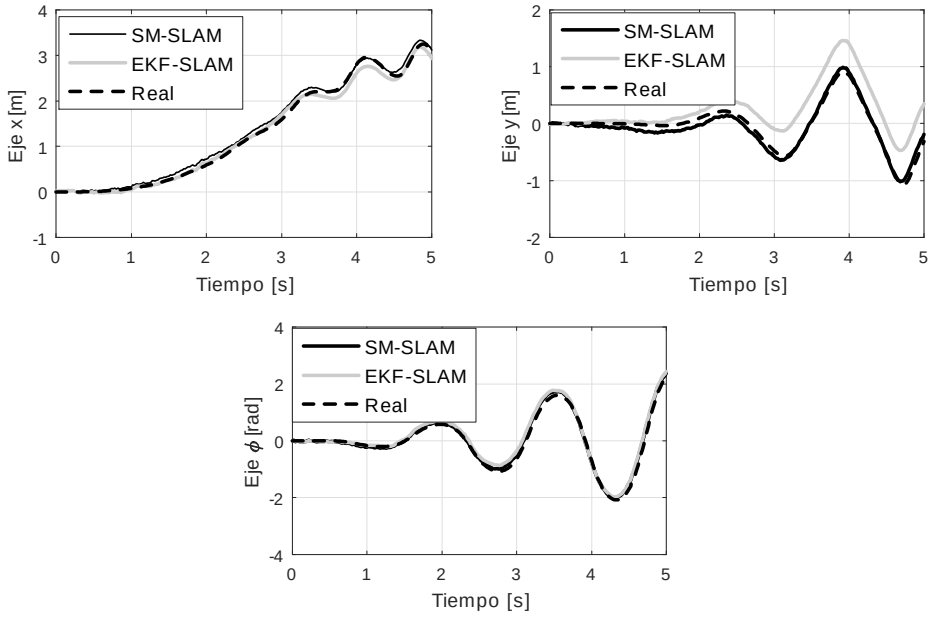

Figura 2.7: Respuesta de los estados x, y y ϕ del robot móvil.

$$E_a = \frac{1}{N_T} \sum_{k=1}^{N_T} |\phi_k - \phi_k^*| \tag{2.79}$$

Donde $[x_k^*, y_k^*, \phi_k^*]$ son la posición y orientación del robot móvil reales. N_T es la cantidad de datos que se adquieren durante la navegación.

El desempeño del algoritmo SM SLAM es comparada con el algoritmo EKF SLAM, los resultados se presentan en la figura (2.11) donde se puede observar que el algoritmo SM SLAM tiene un mejor desempeño con respecto al algoritmo EKF SLAM con distribuciones no Gaussianas.

Estos resultados muestran como el algoritmo SM SLAM tiene un mejor desempeño ante distribuciones no Gaussianas en la navegación de robots móviles. También, estos algoritmo se implementan en un sistema UAV (Unmanned Aerial Vehicle), tal es el caso de un cuadricóptero donde los resultados obtenidos se presentan a continuación.

Resultados del SLAM para un UAV.

En esta sección, presentamos los resultados obtenidos de la navegación del cuadricóptero a partir de las simulaciones realizadas. Los ruidos que se utilizaron en este trabajo tienen

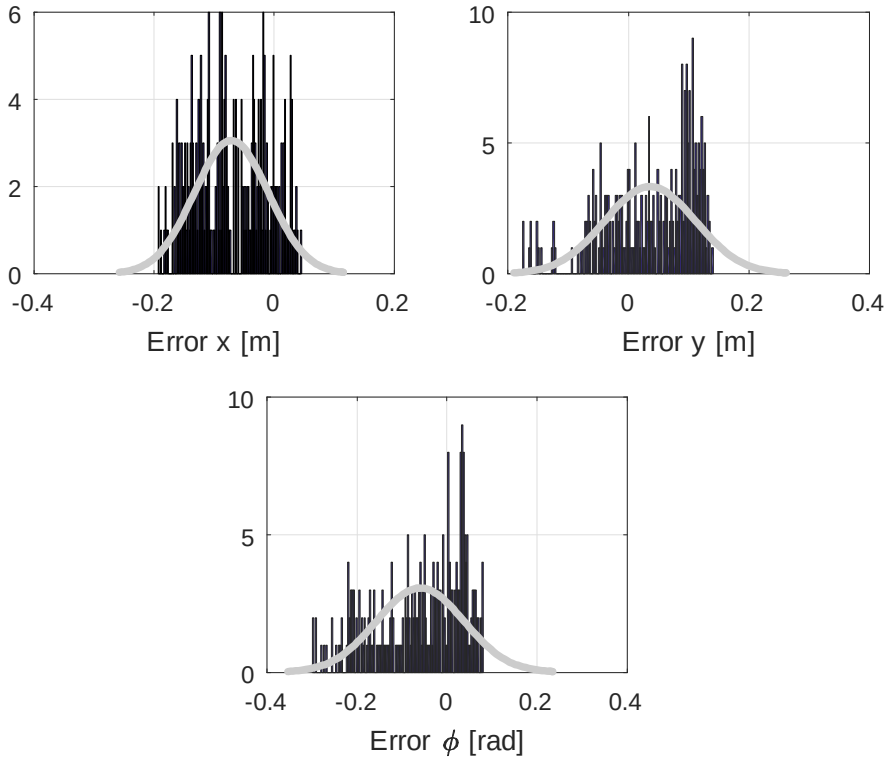

Figura 2.8: Distribución de los errores en los estados del robot móvil.

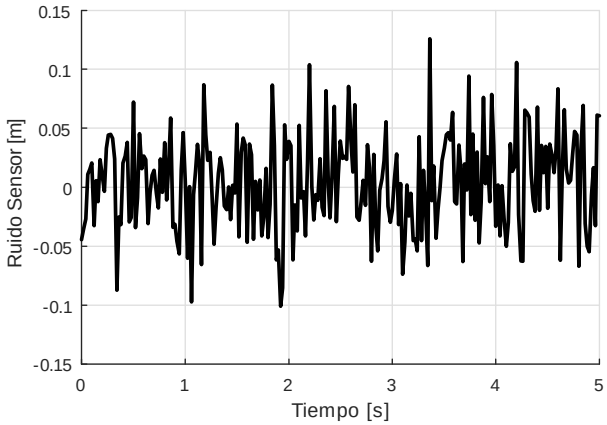

Figura 2.9: Ruido de entrada w_k.

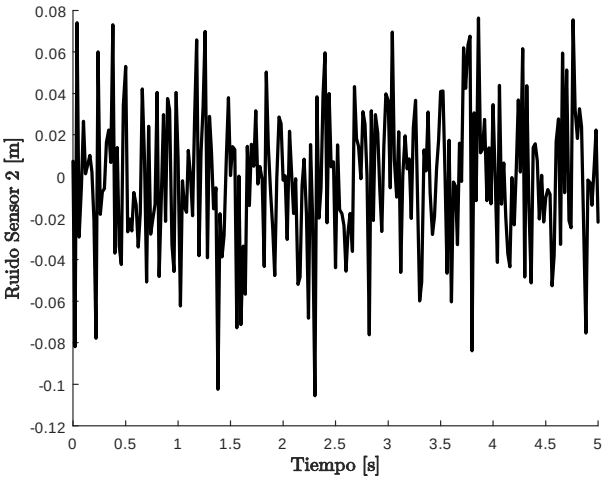

Figura 2.10: Ruido de entrada v_k.

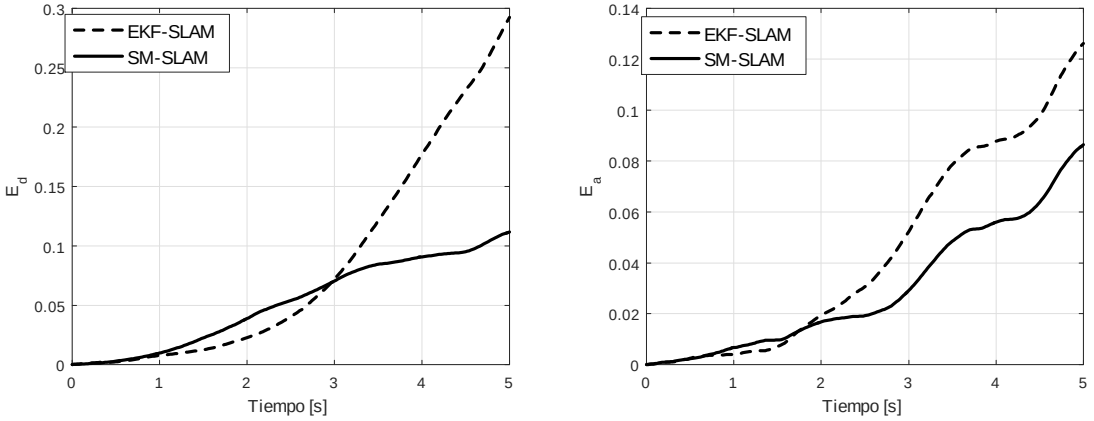

Figura 2.11: Desempeño de los algoritmos SLAMs presentados.

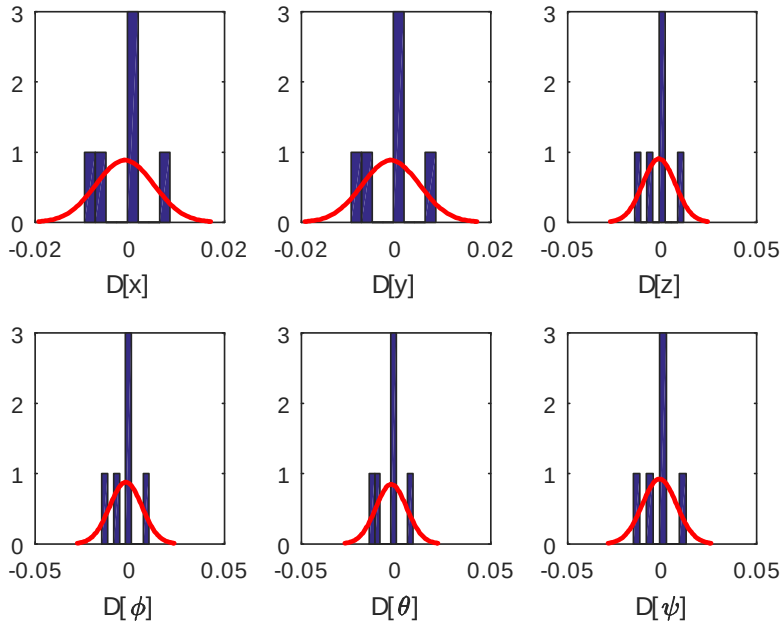

Figura 2.12: Distribución de los errores en los estados del sistema UAV.

la amplitud de $v_k = 0{,}02m$ y $w_k = 0{,}05m$ con el propósito de mostrar la robustez contra las perturbaciones del algoritmo propuesto. En la figura (2.12) muestra la distribución del error de cada estado estimado donde se puede observar que las distribuciones no son del tipo Gaussiano y con una media distinta de cero, que es un requisito esencial para el algoritmo EKF.

Las entradas del algoritmo SLAM son las velocidades del cuadricóptero en el marco del cuerpo: $[u\ v\ w]$ en m/s y $[p\ q\ r]$ en rad/s. Las señales de entrada se muestran en la figura (2.13) que se obtuvieron del trabajo realizado en el capítulo de control de movimiento (capítulo de esta tesis) donde las velocidades corresponden en el marco del cuerpo que permite desarrollar la odometría. Cabe mencionar que dichas señales o entradas se pueden obtener de manera experimental por los sensores equipados en el cuadricóptero.

El algoritmo SM-SLAM tiene un parámetro de ajuste $\rho = [1{,}0, 1{,}0, 1{,}0, 0{,}8, 0{,}8, 0{,}8]$ que son las ganancias de la teoría de los modos deslizantes para los estados estimados del cuadricóptero (etapa de predicción), las ganancias se determinaron de manera heurística. Este parámetro permite ajustar los estados de estimación considerando que los ruidos están acotados (característica de la teoría de los modos deslizantes), lo que es una ventaja sobre el algoritmo EKF-SLAM. Sin embargo, una de las desventajas de la teoría de los

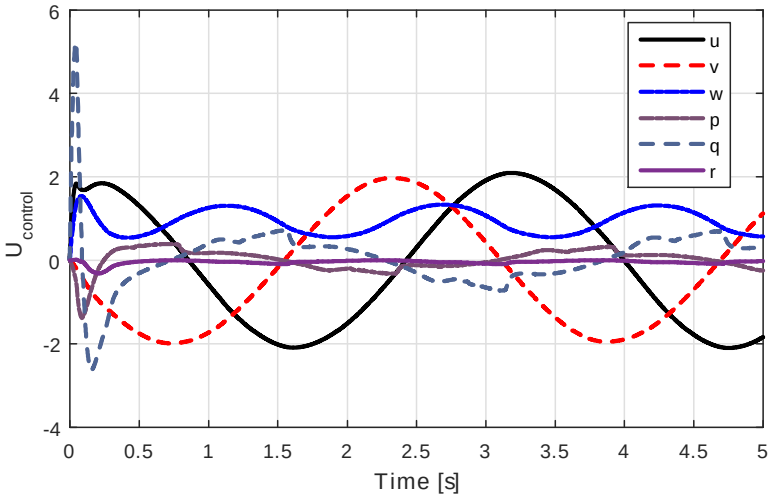

Figura 2.13: Señales de entrada al algoritmo SLAM para el sistema UAV.

modos deslizantes es el efecto de Chaterring que se genera, debido a las vibraciones, la señal muestra imperfecciones. Los resultados obtenidos se muestran en la figura (2.14) donde el algoritmo propuesto muestra una trayectoria estimada (SM-SLAM) cerca de la ruta de navegación (Ref) del cuadricóptero. Además, la navegación se estima con el algoritmo EKF-SLAM que permite comparar el desempeño del algoritmo propuesto. Los obstáculos se agregan aleatoriamente dentro del entorno de navegación del cuadricóptero marcadas con x y los Landmarks que se observan durante la navegación del cuadricóptero son marcadas con un circulo.

La figura (2.15) muestra los resultados obtenidos de los estados de estimación del cuadricóptero utilizando el algoritmo SM-SLAM y es comparada con el algoritmo EKF-SLAM. La estimación de los estados $\hat{X}$ realizada por el algoritmo SM-SLAM propuesto se aproxima a los estados del cuadricóptero (ref). Por el contrario, los estados estimados por el algoritmo EKF muestran mayores desviaciones con respecto a los estados del cuadricóptero. En consecuencia, los errores de estimación del algoritmo EKF-SLAM se atenúan utilizando el algoritmo SM-SLAM. En los estados x, ϕ, θ y ψ la mejora en el desempeño del algoritmo SM SLAM con respecto a la navegación real (Ref) es mejorada comparada con el algoritmo EKF SLAM, conclusión soportada de manera visual. De igual manera, se puede concluir que en los estados y y z no presentan mejoras considerables. Por lo tanto, es de interés conocer la mejora de manera numérica que se presenta a continuación.

Para cuantificar la atenuación del error en los estados estimados, usamos el error de

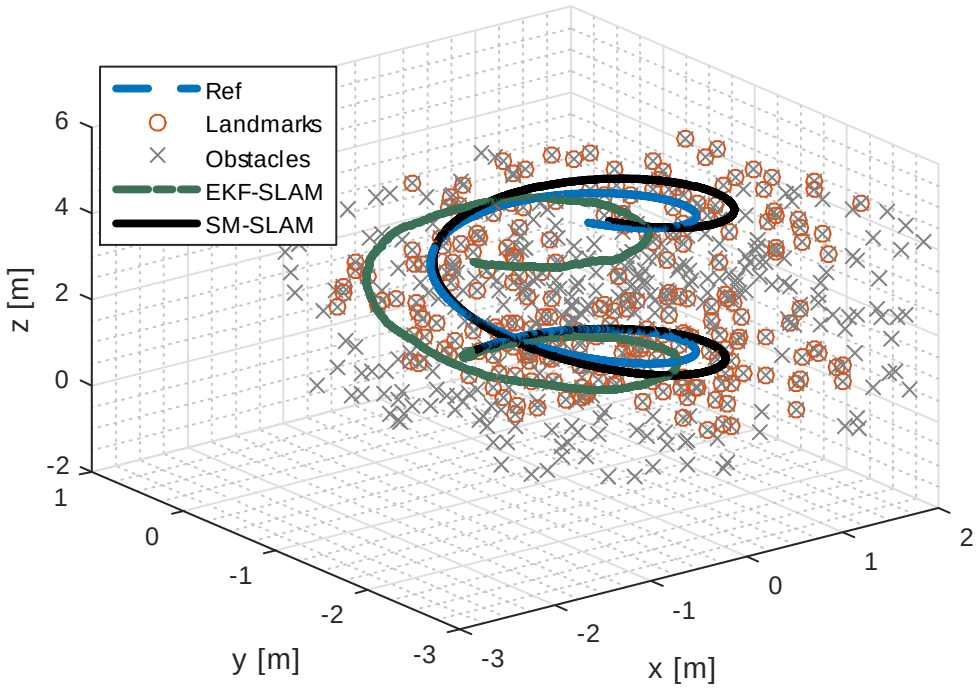

Figura 2.14: Respuesta de los algoritmos SLAM presentados en el sistema UAV.

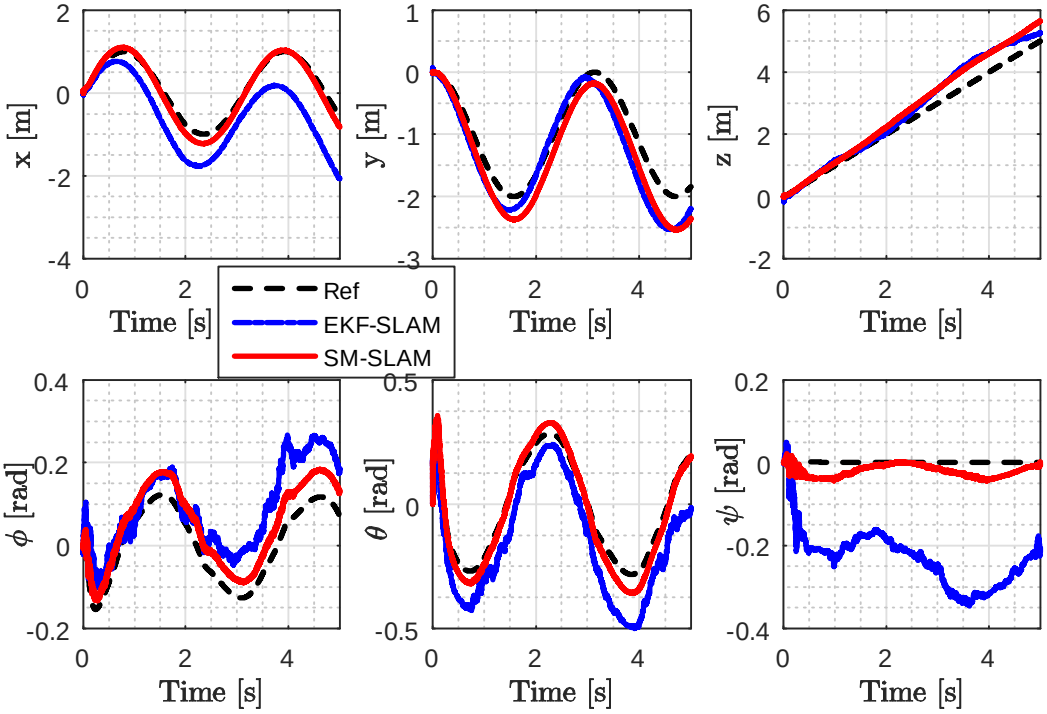

Figura 2.15: Respuesta de estados del sistema UAV.

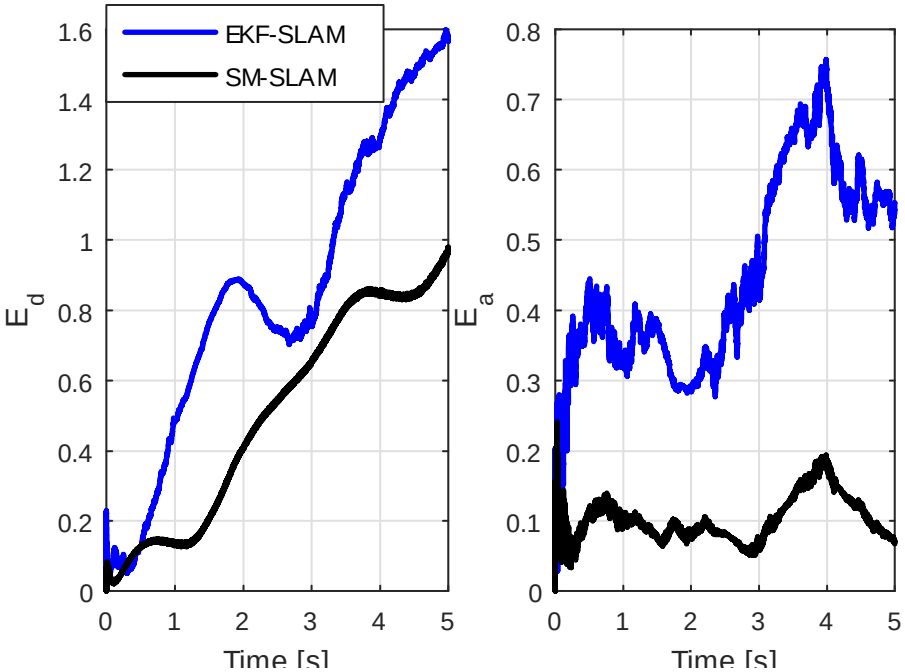

Figura 2.16: Desempeño del algoritmo SM SLAM comparada con el EKF SLAM.

la distancia Euclidiana E_d para el movimiento cartesiano del cuadricóptero y el error absoluto E_a para los ángulos de Euler, definidas como

$$E_d = \frac{1}{N_T} \sum_{k=1}^{N_T} \sqrt{(x_k - x_k^*)^2 + (y_k - y_k^*)^2 + (z_k - y_k^*)^2} \tag{2.80}$$

$$E_a = \frac{1}{N_T} \sum_{k=1}^{N_T} |\phi_k - \phi_k^*| + |\theta_k - \theta_k^*| + |\psi_k - \psi_k^*| \tag{2.81}$$

Donde x^*, y^*, z^*, ϕ_k^*, θ_k^*, ψ_k^* son los estados estimados y N_T el número total de las muestras. Los resultados de la cuantificación del error en los estados estimados se muestran en la figura (2.16). Donde se puede ver que el algoritmo SM-SLAM tiene un error más bajo que el algoritmo EKF-SLAM en presencia de ruidos blancos.

El algoritmo SM SLAM tiene un mejor desempeño comparada con el EKF SLAM en un entorno de 3 dimensiones con sistemas complejos como es el caso del cuadricóptero. Esto muestra que el SM SLAM se puede implementar en diversos sistemas con restricciones como el equipo tecnológico para realizar la percepción del entorno.

Resultados experimentales

Figura 2.17: Entorno de la implementación experimental del SLAM.

En esta sección presentamos los resultados obtenidos de los datos experimentales. La implementación se realiza en un entorno abierto (véase en la figura (2.17)), el robot que se utiliza es un robot koala (K-team Corporation 2013), este robot es un robot comercial que viene equipado con sensores: Encoders para la medición de la edometría y sensores laser Range finder para la medición de distancia de los obstáculos con precisión menor a 0.1m. Datos obtenidos de un trabajo previo [100].

En la figura (2.18) presentamos los resultados al implementar el algoritmo EKF SLAM sin incluir ruido computacional al sistema. En esta prueba no se tiene una trayectoria de referencia para evaluar el desempeño de localización. Para esta prueba se realiza un recorrido por el entorno presentado anteriormente y se regresa al punto de partida. Por lo que se espera que la estimación de los estados del robot móvil también regrese al punto de partida. En este resultado se puede apreciar visualmente que el robot móvil regresa al punto de partida además el mapa que se construye corresponde al entorno de la implementación.

La segunda prueba que sea realiza es incluir ruido externo a la prueba anterior. La respuesta que se obtiene se muestra en la figura (2.19) donde se puede observar que la estimación de los estados no llega al punto de partida como se esperaba y el mapa contiene ruido, es decir, que el mapa de forma visual no corresponde al entrono que se explora.

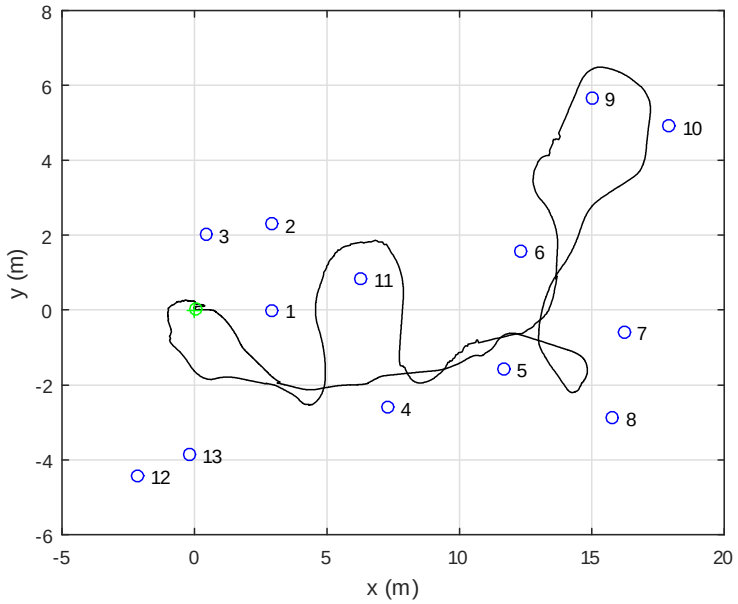

Figura 2.18: Resultados del algoritmo EKF SLAM con datos experimentales.

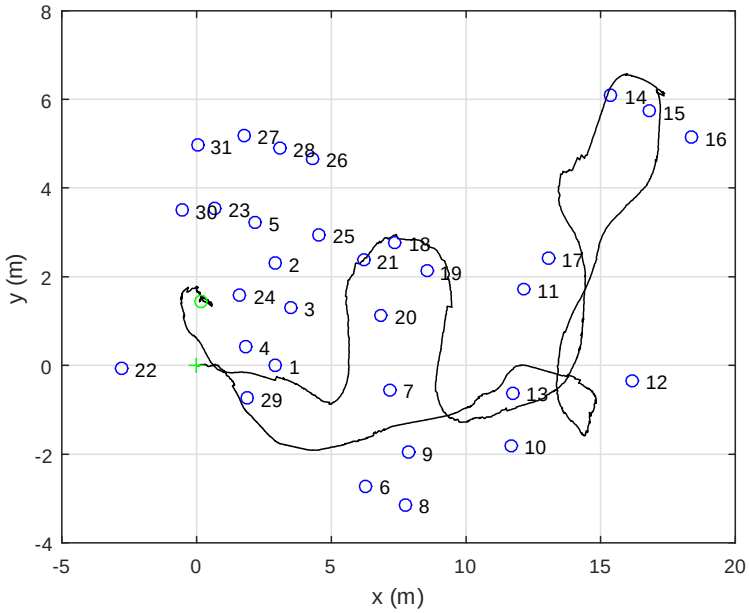

Figura 2.19: Resultados del algoritmo EKF SLAM con ruido en los datos experimentales.

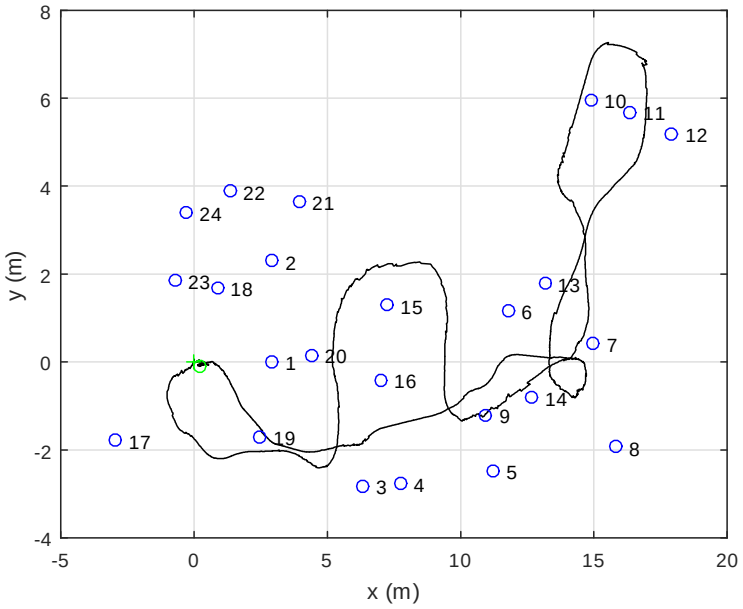

Figura 2.20: Resultados del algoritmo modos deslizantes SLAM y con ruido en los datos experimentales.

Para la última prueba se implementa el algoritmo modos deslizantes SLAM para atenuar los errores que se presentan en la segunda prueba. Los resultados se presentan en la figura 2.20 donde se puede notar que la estimación de los estados termina en el punto de partida y así como una mejora en el mapa, es decir, se logra eliminar algunos errores de estimación en las marcas fijas.

En las figuras presentamos el desempeño del algoritmo modos deslizantes SLAM y comparado con el algoritmo EKF SLAM. Donde podemos observar que el algoritmo propuesto tiene una divergencia menor al del algoritmo EKF. Atenuar los errores de la estimación de los estados nos ayuda a que el robot móvil pueda navegar y explorar distancias largas con resultados satisfactorios. El desempeño se obtiene con las siguientes ecuaciones.

$$E_d = \frac{1}{N_T} \sum_{k=1}^{N_T} \sqrt{(x_k - x_k^*)^2 + (y_k - y_k^*)^2} \tag{2.82}$$

$$E_a = \frac{1}{N_T} \sum_{k=1}^{N_T} |\phi_k - \phi_k^*| \tag{2.83}$$

Donde $[x_k^*, y_k^*, \phi_k^*]$ son la posición y orientación del robot móvil tomados del algoritmo EKF SLAM sin ruido. N_T es la cantidad de datos que se adquieren durante la navegación.

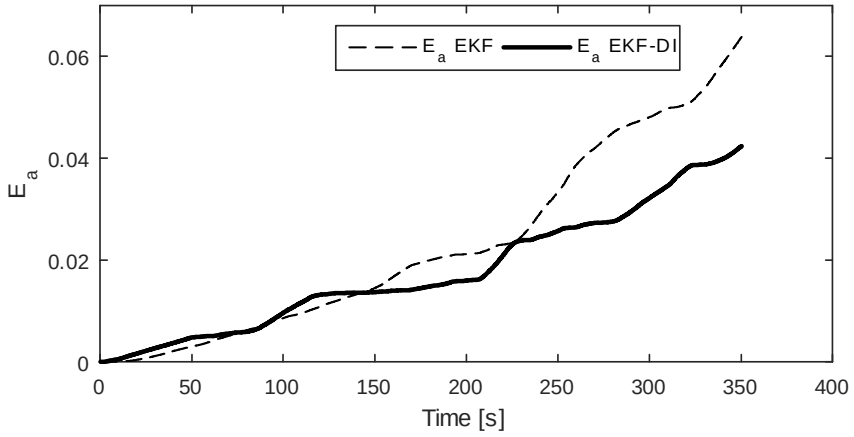

Figura 2.21: Desempeño del SLAM evaluado por el error absoluto de la orientación del robot móvil.

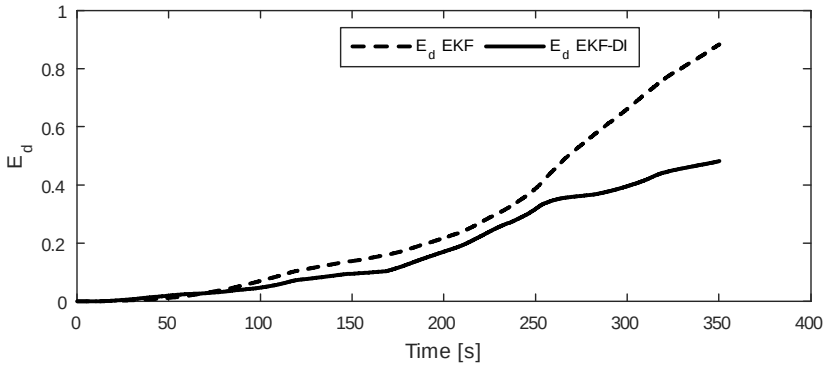

Figura 2.22: Desempeño del SLAM evaluado por el error Euclidiano del robot móvil.

Proponer técnicas para el problema de localización y mapeo simultáneo en un sistema como el robot móvil o un vehículo aéreo no tripulado con los algoritmos presentados en los capítulos anteriores por si solos no logra convertir estos vehículos como autónomos. El siguiente trabajo es importante para lograr la autonomía de los sistemas y consiste en obtener una trayectoria que deben de seguir dichos sistemas durante la navegación. Al realizar la navegación se obtienen las velocidades angulares o lineales que son entradas a los algoritmos antes presentados para realizar la odometría y poder estimar las posiciones y posteriormente se le realizan ajustes con respecto al mapa que se construye. Por lo tanto, en el siguiente capítulo se presentan los algoritmos de planeación de trayectorias para la navegación de los robots móviles.

Capítulo 3

Planeación de trayectoria.

La navegación autónoma en vehículos: vehículos submarinos autónomos (AUV), vehículos aéreos no tripulados (UAV), robots móviles, entre otros robots se ha convertido en un tema de interés en los últimos años. Resolver el problema de la navegación autónoma es un desafío para la comunidad científica, y se han desarrollado diferentes tipos de navegación según el entorno en el que se aplican o el tipo de robot. Tales como las navegaciones en entornos conocidos donde el objetivo principal es encontrar una trayectoria óptima, entornos parcialmente conocidos o dinámicos que consistan en evitar colisiones y entornos desconocidos en los que se requiere exploración o encontrar una ruta desde un estado inicial a un estado objetivo.

La percepción, la planificación de trayectorias, la localización y el mapeo son parte de la navegación autónoma en los robots. La percepción o la información recopilada a través de la observación son fundamentales para los robots autónomos, ya sea para evitar colisiones o la localización y el mapeo simultáneo (SLAM) considerando que la percepción del robot tiene un rango limitado además con incertidumbres, por lo tanto, no es posible encontrar una trayectoria global para la navegación. En consecuencia, es necesario desarrollar un algoritmo de planeación de trayectorias que permita dirigir la navegación dentro del entorno.

El objetivo de la navegación autónoma en los robots o agentes es llevar desde un punto de partida S_i hasta un punto de destino S_T dentro de un entorno definido. En este trabajo, se considera que el entorno es totalmente desconocido; por lo tanto, no es posible encontrar una trayectoria óptima como se hace en entornos conocidos. En este sentido, construimos un modelo del entorno y estimamos los estados del agente o robot utilizando SM SLAM y posteriormente utilizamos un algoritmo de planeación de trayectorias que

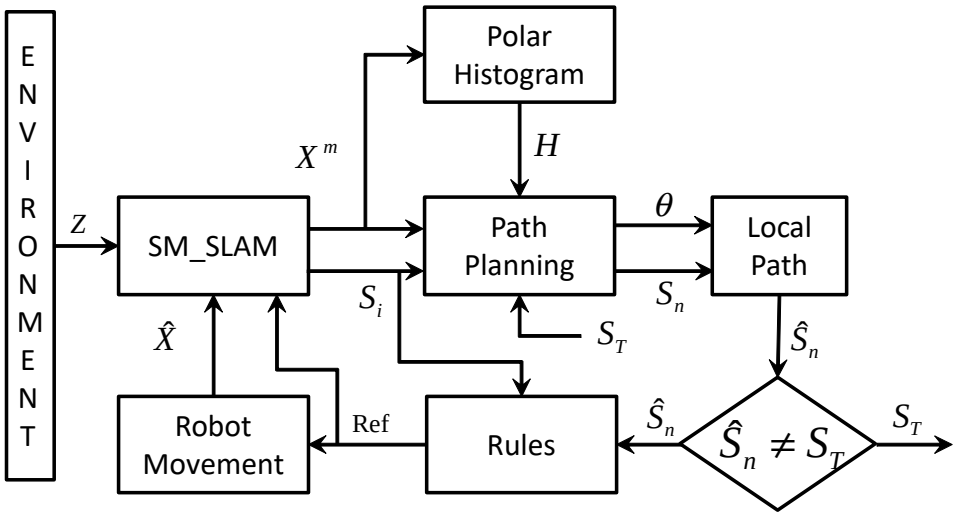

Figura 3.1: Diagrama de un algoritmo de navegación autónoma para los robots.

permite realizar la tarea de navegación autónoma.

Para encontrar la trayectoria en la navegación del robot en entornos desconocidos, utilizamos un algoritmo de planeación de trayectorias. La planificación de trayectorias propuesta permite encontrar una ruta parcial dentro del conjunto explorado por SM SLAM. El algoritmo se desarrolla teniendo en cuenta que los sensores de robot no son capaces de explorar un entorno global, en el que no se puede encontrar una trayectoria óptima y global.

Las principales características de la navegación autónoma son (figura 3.1)

1. Entorno. Un enfoque fundamental para formular y resolver el problema de la planificación de trayectorias o la navegación en entornos conocidos o desconocidos es la configuración del espacio.

$$S_{free}(t) = \{q \in G_O(t) \mid A(q) \cap S_{obs}(t) = 0\}$$

$$S_{obs}(t) = G_O(t)/S_{free}(t)$$

Donde $G_O \subseteq G_E \in \mathbb{R}^n$ es la región observada y depende del tiempo porque representa un mapa local y aumenta la observación a medida que se realiza la navegación, G_E corresponde al entorno, S_{free} es el espacio libre de obstáculos, S_{obs} es el espacio de obstáculos, $A(q)$ es la configuración del agente

2. Tarea de navegación. El objetivo de la navegación es mover un agente desde un estado inicial S_I a un estado objetivo S_T en tiempo finito dentro del entorno G_E, donde $\{G_O(t) \cup G_{UN}(t)\} \subseteq G_E$ con G_{UN} es la región no observada.

3. Planificación del trayectorias. La búsqueda de un camino es esencial para lograr el objetivo de la navegación autónoma. El algoritmo proporciona una trayectoria S_n que el robot debe realizar en cada iteración. Esto se desarrolla con el apoyo de un histograma polar y verificando si el objetivo no es una solución local.

4. Reglas. Un generador de reglas que es capaz de proporcionar las referencias de movimiento del robot.

5. Robot. Es el agente que realiza la tarea de navegación con la configuración del agente $A(q)$, donde q es la forma del robot. Los robots están equipados con sensores, actuadores y otros dispositivos para realizar la tarea de navegación.

La navegación autónoma depende del tipo de entorno en el que se realiza la navegación; Presentamos tres entornos donde se puede aplicar el algoritmo de navegación autónomo propuesto. En un trabajo anterior [106] los ambientes fueron definidos como:

- Entornos conocidos (figure 3.2). Esto es posible encontrar una trayectoria global para la tarea de navegación, es decir, el siguiente estado S_n es conocido $\forall S_i$.

- Entornos parcialmente conocidos (figure 3.3). En este caso, el entorno observado G_O puede cambiar o tiene diferentes obstáculos relacionados con el mapa anterior G_{map}. Por lo tanto, la trayectoria debe adaptarse a la región observada en caso de cambios en la región no observada.

- Entornos desconocidos (figure 3.4). No es posible encontrar una trayectoria óptima y global dentro de un entorno desconocido debido a la falta de información. Por lo tanto, el estado objetivo se convierte en una solución local dentro de la región observada.

La descomposición de rejillas se utiliza para configurar el entorno y así clasificar el espacio libre de obstáculos y el espacio de obstáculos, de esta manera, se obtiene un modelo del entorno. En entornos conocidos, la tarea de navegación autónoma es fácil de resolver, utilizando el método de ocupación de rejilla (cada borde de incidentes es una solución) o métodos de campo potenciales donde el algoritmo de planificación de

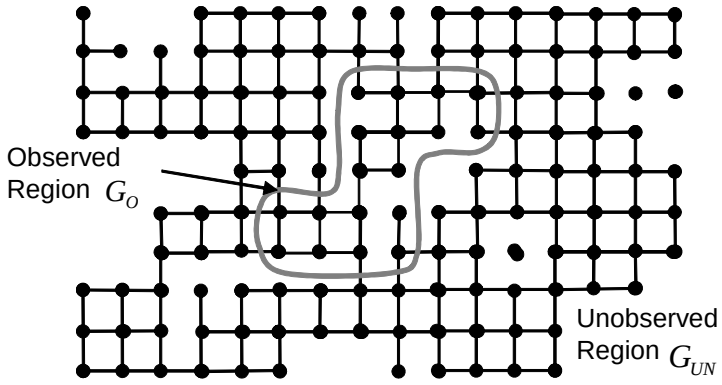

Figura 3.2: Configuración del entorno conocido.

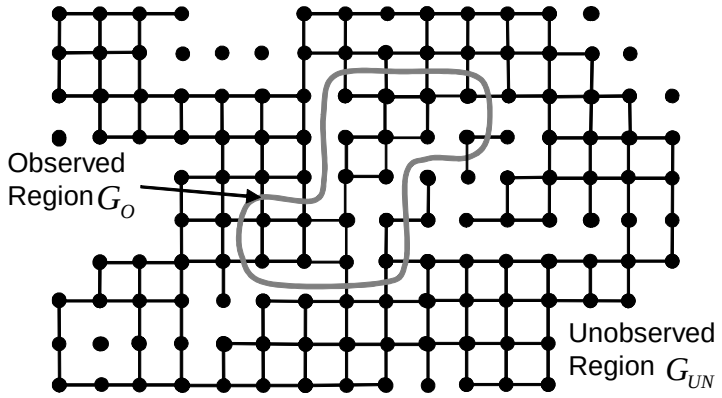

Figura 3.3: Configuración del entorno parcialmente conocido.

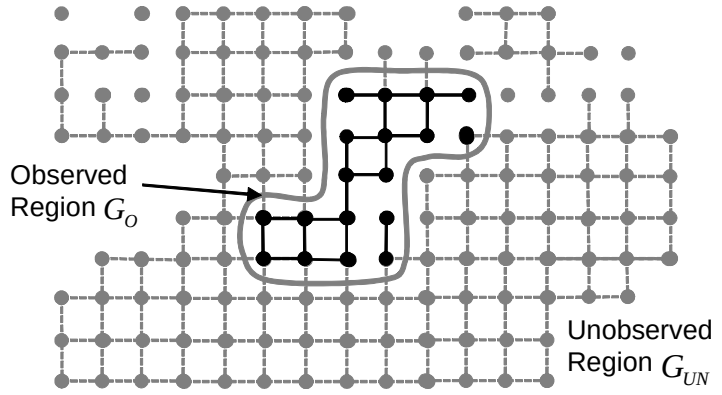

Figura 3.4: Configuración del entorno desconocido.

trayectorias es responsable de optimizar la ruta desde el estado inicial hasta el estado objetivo. Y en el caso de los entornos parcialmente conocidos la trayectoria sufre cambios si el modelo experimenta un cambio. Sin embargo, en entornos desconocidos el modelo carece de información; por lo tanto no es posible generar una trayectoria óptima. En este caso, debe observarse la región para clasificar el entorno y luego realizar la navegación dentro del entorno observado.

Podríamos construir un mapa con la región observada $G_O(t)$ que es una observación parcial del entorno. Entonces el mapa sería la unión de todas las observaciones realizadas por el robot desde el estado inicial al estado objetivo, es decir, $G_{map}(t_k) = G_O(t_0) \cup G_O(t_1) \cup ... \cup G_O(t_k)$. Sin embargo, el mapa y el entorno mostrarían inconsistencias debido a errores de ubicación, mapeo y percepción. Para atenuar estos errores, proponemos utilizar el algoritmo SLAM de modo deslizante, que es un algoritmo robusto contra las perturbaciones. El modo deslizante SLAM realiza la estimación del mapa local y la ubicación del robot que posteriormente utiliza el algoritmo de planificación de trayectorias para encontrar una ruta en la tarea de navegación.

3.1. Histograma polar.

Las siguientes definiciones se utilizan para el algoritmo de planificación de trayectorias:

- Histograma polar: $H(\theta)$ se define como un histograma polar para algún $t \geq t_0$. El histograma polar (figura 3.5) nos proporciona un conjunto de posibles direcciones de búsqueda para el siguiente estado de S_n. Donde, $H(\theta_i) := \{\sigma_i \in S_{obs}(t), P_a \in S_{free}(t) \mid H(\theta_i) = r_O - d_E$ aquí d_E es la función de la distancia euclidiana entre el obstáculo σ_i y la posición actual del robot P_a en la dirección de observación θ_i. r_O es el radio de observación máximo para el agente.

- Umbral: Definir el umbral T_h nos permite encontrar el conjunto $G_{Search} \subseteq G_O$ donde $G_{search} := \{\theta \in [0, 2\pi] \mid H(\theta) \leq T_h\}$ en este conjunto se realiza la operación de búsqueda.

- Frontera: un estado S_f está en el borde $G_F(t)$ si y solo $S_f \in (G_{neigbors} \cap G_{UO}(t))$, donde $G_{neigbors} \subset G_{search}$ es el conjunto de todos los vecinos inmediatos de S_f.

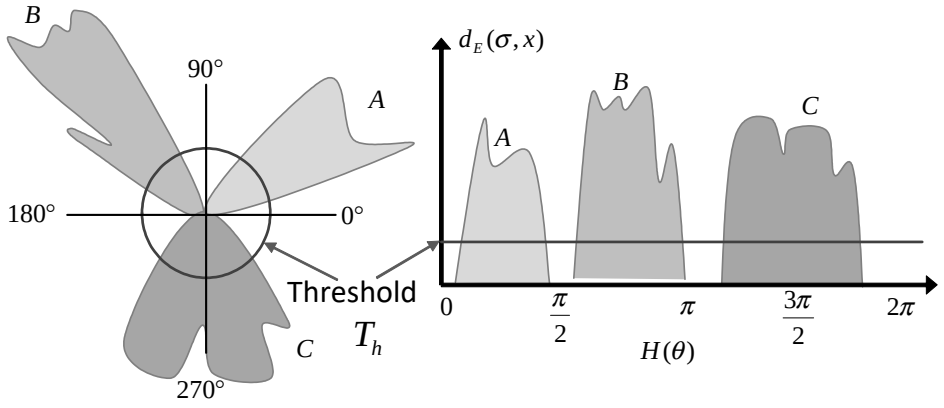

Figura 3.5: Histograma polar P_a para algún $t > t_0$.

- Planeación de trayectorias: en un entorno desconocido $S_T \in G_{UO}$ para algún $t > t_0$, la planeación de trayectoria elige un objetivo parcial $S_n(t) \in G_O$, que se obtiene al optimizar la distancia del conjunto de la frontera con el estado objetivo, es decir, $S_n(t) := \text{mín}(d_E(S_F, S_T))$ esto se hace en cada iteración de la navegación . Y cuando $S_T \in G_O(t)$, entonces el objetivo de la planificación de trayectorias es elegir S_T para completar la tarea de navegación.

Los siguientes teoremas son condiciones necesarias para el algoritmo de planeación de trayectorias para el histograma polar.

Teorema 3.1 *El siguiente estado u objetivo de la trayectoria subsiguiente $S_n \in G_O(t)$ para algún $t \geq t_0$ existe si y solo si el conjunto $G_{search} \notin \emptyset$.*

Demostración. La prueba del teorema se lleva a cabo considerando dos casos particulares: el primer caso, si la región observada $G_O(t)$ está libre de obstáculos, entonces el espacio de búsqueda corresponde a la región observada, es decir, $G_{search} \subseteq G_O$, donde $G_O \notin \emptyset$. En consecuencia, el siguiente estado corresponde a un valor del conjunto de fronteras, $S_n \in G_F \subset G_{search}$. El segundo caso, es el caso extremo cuando la región observada contiene obstáculos, en este caso, el histograma polar nos ayuda a definir nuestra búsqueda, luego si $\exists\theta$ va de 0 a 2π de manera que $H(\theta) \leq T_h$ ó $H(\theta) = 0$, entonces G_{search} no es un conjunto vacío, donde la dirección θ es un espacio de búsqueda para encontrar el siguiente estado. ■

Teorema 3.2 *Sea $\|G_E\| < \alpha$ y el agente $A(q)$ de un estado inicial $S_I \in G_O(t_0)$, el sistema de navegación en un entorno desconocido se completa si $S_n := S_T$ por un tiempo finito.*

Demostración. El entorno G_E es un conjunto compacto invariante en el tiempo porque no es posible explorar fuera de este conjunto para alcanzar el estado de destino S_T de la tarea de navegación. Por otro lado, el conjunto $G_O(t)$ es una variante en el tiempo y se espera que crezca para incluir el estado objetivo. De lo contrario, si la región observada no aumenta, esto implica que el estado objetivo de la tarea de navegación en el conjunto de observación no se incluirá. Esto es:

$$\|G_O(t_0)\| < \|G_O(t_k)\| < \|G_O(t_{k+1})\| < \alpha \text{ para } k = 1, 2, 3... \text{ con } 0 < \alpha < \infty \tag{3.1}$$

Si se cumple la ecuación (3.1) para cada iteración, podemos encontrar un estado siguiente. Es decir, para $G_O(t_0) \to S_1$, $G_O(t_{k-1}) \to S_k$ y $G_O(t_k) \to S_{k+1}$ si esto se hace recursivamente, luego $S_n := S_T$ para n en tiempo finito debido que $\lim_{t\to\infty} \|G_O(t)\| \leq \|G_E\| < \alpha$. ■

La planeación de trayectorias es una tarea esencial en la navegación autónoma para alcanzar el estado objetivo. El siguiente algoritmo corresponde a la planeación de trayectorias dentro de la región observada $G_O(t)$. El primer paso en este algoritmo es construir un histograma $H(\theta)$ que permita reducir el espacio de búsqueda para el siguiente estado S_n en la región observada, es decir, en presencia de obstáculos hay regiones que deben omitirse para la búsqueda de S_n esto se debe a que en esas regiones no encontraremos una solución. Por lo tanto, obtenemos un conjunto de búsqueda reducido $G_{search}(t)$ en comparación con un conjunto sin obstáculos. En el segundo paso, buscamos la frontera del conjunto de búsqueda con la región no observada. Finalmente, seleccionamos una solución mínima dentro del conjunto de la frontera. Por otro lado, calculamos la dirección de la solución para descartar que se trate de una solución local.

Algoritmo 3.1 *Path_Planning_Histogram*(Map, S_i, S_T)

$G_{search} = \emptyset, G_F = \emptyset$
$H \leftarrow Histogram(Map, S_i)$
for *all* $\theta \in [0, 2\pi]$ ***do***
    if $H(\theta) < T_h$ *or* $H(\theta) = 0$ ***then***
        $G_{search}(i) = H(\theta)$
    end if
end for
if $G_{search} \neq \emptyset$ ***then***
    $G_F \leftarrow ComputeFrontier(G_{search})$
else

break
end if
$S_n = \arg\min_{n \in G_F}(d_E(G_F(n), S_T))$
$\theta_i = \arccos\left(\frac{S_i \cdot S_n}{\|S_i\|\|S_n\|}\right)$
return S_n, θ_i

La navegación autónoma en entornos desconocidos es una tarea compleja debido a la incertidumbre y la falta de información del entorno no explorado. Las incertidumbres pueden generarse por la instrumentación de los robots (sensores o actuadores), incertidumbres en la construcción del mapa, y también hay errores en los modelados matemáticos de los sistemas. a continuación se plantea un algoritmo de navegación utilizando la planeación de trayectorias presentando en esta sección y con el SLAM.

Navegación autónoma con histograma polar

La navegación autónoma con el algoritmo SM SLAM y la planeación de trayectorias corresponde a un algoritmo de navegación en entornos desconocidos. Donde, el algoritmo de planeación de trayectorias consiste en generar un histograma polar $H(\theta)$ que produce un conjunto de búsqueda G_{search} el cual permite encontrar una ruta parcial para la tarea de navegación del robot móvil. Por otra parte, para reducir la busqueda dentro del conjunto G_{search} obtenemos un subconjunto $G_F \subset G_{search}$ que consiste en encontrar las fronteras del espacio observado el cual permite un avance parcial de S_n máximo en cada iteración, en consecuencia, buscamos minimizar la distancia euclidiana del estado objetivo con el subconjunto de la frontera. Sin embargo, la navegación puede caer en una solución local, y si este es el caso, usamos la función $ComputeOutsideLocal$ que proporciona una solución $\hat{S}_n$ fuera de la solución local. Por otro lado, la planeación de trayectorias requiere un mapa y la localización del robot, así como el estado objetivo. Para construir un mapa y conocer la localización del robot, utilizamos el algoritmo SLAM de modo deslizante (SM SLAM) presentado en el capítulo anterior.

Algoritmo 3.2 *Autonomous Navagation with Histogram*
$S_I :=$*initial,* $S_T :=$*Target*
$P^- :=$ *covariance*(R_1), $P^+ :=$ *covariance*(R_2)
$\rho :=$ *gain sliding mode,* $r_O :=$ *search radius-*
$Map =$*search_ obstacles*(S_I, r_O)

$S_n, \hat{S}_n = Path\_Planning(Map, S_I, S_T)$
$R_0 = Rules(S_I, S_n)$
while $\hat{S}_n \neq S_T$
$\left[Map_k, \hat{X}_k\right] = SM\_SLAM\ \left(S_k, S_n, R_k, P_k^-, P_k^+, \rho, z_k\right)$
$S_i = \hat{X}_k(end\ state)$
$Map\,(end : length(Map_k)) = Map_k$
$[S_n \ \ \theta_i]$ =$Path\_Planning(Map, S_i, S_T)$
if $\theta_i > \pi$
$\hat{S}_n = ComputeOutsideLocal(Map, S_i, S_T, \theta_i)$
else
$\hat{S}_n = S_n$
end if;
$R_k = Rules(S_i, \hat{S}_n)$
end while
return $\hat{S}_n, \hat{X}_k$

3.2. Roadmap Algoritmo Genético.

La planeación de trayectorias es un problema de optimización que involucra la búsqueda de una trayectoria libre de colisiones y óptima entre dos localizaciones. Donde la planeación de trayectorias tiene diversas áreas de aplicaciones tales como la manufactura, ensambles, transportes y servicios. Sin embargo, las aplicaciones más destacadas en la actualidad se enfocan en los robots autónomos donde son implementados en los entornos de difícil acceso o de alto riego para los humanos así como la exploración de entornos desconocidos y en los transportes autónomos donde se requiere de una planeación de trayectorias para lograr con el objetivo de autonomía en los robots.

Existen tres tipos fundamentales de planeación de trayectorias dependiendo del entorno de aplicación: primero, entornos conocidos, donde la solución de la planeación de trayectorias consiste en encontrar una ruta de navegación global, el problema se reduce a un problema de optimización desde un punto inicial a un punto objetivo. Segundo, entornos parciamente desconocidos, cual permite un robot autónomo a responder a situaciones inesperadas mientras realiza el proceso de navegación, es decir, el algoritmo es capaz de encontrar una nueva ruta libre de obstáculos durante la navegación en caso de

un obstáculo en la solución previa [104]. Tercero, entornos totalmente desconocidos, aquí la solución que se proporciona es parcial por lo tanto la navegación consiste en alcanzar las soluciones parciales, sin embargo, la planeación de trayectorias depende de la navegación el cual se vuelve una solución recursiva hasta que se alcanza el punto objetivo. Para los entornos parcialmente conocidos o desconocidos se requiere de un algoritmo de localización y mapeo simultaneo (SLAM) que permiten modelar el entorno[97].

En cualquier de los casos, la trayectoria debe ser óptimo bajo algún criterio como la longitud de la trayectoria, el tiempo de ejecución o el consumo de energía en la navegación. En este sentido, se han realizado diversos algoritmos incluidos los algoritmos meta heurísticos como las redes neuronales, optimización por colonia de hormigas (ACO) [3], optimización por cúmulo de partículas (PSO) [73], [105] y los algoritmos genéticos (GA) [76] los cuales pueden ser implementados en línea y también los algoritmos fuera de línea como el Roadmap [2], descomposición de celdas [87] o campo potencial [74]. El algoritmo genético tiene la característica de fácil aplicación ya que se adaptan a diversos problemas de optimización o en encontrar soluciones a funciones no determinísticos [45], robustos ante incertidumbres en los modelos, convergen en tiempo finito [69] además encontrar soluciones globales lo cual, hace que sea un algoritmo popular entre los investigadores [88]. Se realiza análisis de convergencia de los algoritmos evolutivos como una cadena de Markov finito [9],[38].

Navegación autónoma con Roadmap algoritmo genético

En este trabajo se resuelve el problema de la planeación de trayectorias para la navegación autónoma en los robots, es decir, una navegación principalmente en los robots autónomos requieren de una ruta previa libre de obstáculos desde un punto inicial x_S a un punto objetivo x_T dentro de un conjunto definido B el cual se configura con dos subconjuntos: el espacio libre de obstáculos $B_{free} = \{z_r \in B \mid A(z_r) \cap B_{obs} = \emptyset\}$ y $B_{obs} = \frac{B}{B_{free}}$ el conjunto de los obstáculos, donde z_r es la forma del robot y $A(z_r)$ área del robot. Realizada la configuración del entorno el objetivo principal es encontrar una trayectoria $f(x, x_S, x_T) \in B_{free}$ que permite al robot realizar la navegación. Para implementar el algoritmo de planeación de trayectoria dentro del espacio libre B_{free} se requiere de un modelo que definen los estados x con x_S y x_T definidos previamente.

Para modelar el entorno utilizamos el método Roadmap, el proceso consiste en generar objetivos aleatorios de manera distribuido dentro del espacio libre; $x_i \in B_{free}$ para

todo $i = 1, 2, \ldots, h$. Entonces h es el número total de puntos generados. Posteriormente, los puntos generados son conectados, donde cada conexión es una trayectoria $g(x_i, x_j)$ con $j = 1, 2, \ldots, h$ que no tiene una intersección con los obstáculos, es decir, $D = \{x_i, x_j \in B_{free} \mid g(x_i, x_j) \cap B_{obs} = \emptyset\}$ para todo $i \neq j$. Finalmente, se implementa la planeación de trayectorias que consiste en buscar una solución $f(x, x_S, x_T)$ dentro del conjunto D que es un conjunto finito. Cabe mencionar, en este método h es un número finito donde no se define de manera analítica se obtiene con respecto a la experiencia del entorno. Sin embargo, si el entorno es conocido y la forma de los obstáculos son conocidos $g_n(z_{obs}) \subseteq B_{obs}$ entonces existen maneras de proponer los puntos objetivos, por ejemplo: $x_i \in B_f$, donde B_f es un conjunto de la frontera del obstáculo, $B_f = \left\{ z_{obs} \in B_{obs} \mid B_f^n = \frac{\partial g_n(z_{obs})}{\partial z_{obs}} \right\}$ para las n formas conocidas si y sólo si $g_n(z_{obs})$ es continuamente diferenciable. En caso contrario, se buscan otras características de los obstáculos como: las aristas o intersecciones entre las tangentes.

El proceso antes planteado se aplica en entornos conocidos, es decir, $B_{free} \subset B$ es conocido e invariante en el tiempo t. Sin embargo, existen entornos que cambian con el tiempo, $B_{free}(t)$ o entornos totalmente desconocidos, $B_{free}(0) = \emptyset$. Para resolver el problema de planeación de trayectorias en entornos parcialmente conocidos o totalmente desconocidos utilizamos un algoritmo de localización y construcción de mapa simultáneamente (SLAM); los autores han realizado diversos trabajos en esta área [100], [106]. El algoritmo SLAM nos permite construir un mapa; encontrar el conjunto de los obstáculos $B_{obs}(t)$ y conocer la posición del robot $x_r(t)$ en el proceso de navegación. Por lo tanto, si $B_{obs}(t)$ es construido por el SLAM, tenemos que $B_{free}(t) = B \setminus B_{obs}(t)$. Entonces, la planeación de trayectorias para estos entornos son variantes con el tiempo $f(x(t), x_S, x_T)$ con $x_S = x_r(t)$. De la misma manera la planeación de trayectorias en entornos desconocidos se requiere de un mapa previo $B_{free}(t)$ y para construir un mapa se requiere de una ruta $f(x(t), x_S, x_T)$ para realizar el proceso de navegación. Por lo tanto, se propone el siguiente algoritmo para la navegación que resuelve el problema utilizando una planeación de trayectorias con Roadmap algoritmo genético (PP_RGA) y el algoritmo SLAM.

Algoritmo 3.3 *Navigation with PP_RGA*

$x_S = x_r = [x_0, y_0]; x_T = [x_{goal}, y_{goal}]$
$B_{obs}(0) = \emptyset$
while $x_r \neq x_T$
    $f(x(t), x_S, x_T) = PP\_RGA(B_{obs}, x_S, x_T)$
    while $x_r \notin B_{obs} \parallel x_r \neq x_T$
        $[B_{obs}, x_r] = SLAM(f(x(t), x_S, x_T), x_S, x_T)$

$x_S = x_r$

end

end

El algoritmo anterior nos permite resolver el problema navegación en los entornos desconocidos. Dicho algoritmo también es posible aplicar en los entornos parcialmente conocidos donde $B_{obs}(0) \neq \emptyset$. Donde el algoritmo SLAM nos proporciona el mapa y la posición del robot en el tiempo t y posteriormente se plantea para resolver el problema de planeación de trayectorias.

Planeación de trayectorias basado en el RGA

La planeación de trayectorias que se propone es implementada en la navegación de los robots autónomos dentro de entornos conocidos, parcialmente conocidos o desconocidos. En el entorno conocido la trayectoria que se obtiene es una trayectoria global. En los entornos parcialmente conocidos la trayectoria puede cambiar si el modelo del entorno cambia $B_{obs}(t)$ e interfiere con la trayectoria previa. En los entornos desconocidos la trayectoria es parcial debido a que el mapa que se construye con el algoritmo SLAM son mapas locales. Al configurar el entorno conocemos los conjuntos B_{free} y B_{obs} donde B_{free} se modela a través del proceso de roadmap para conocer el conjunto D que se considera como el espacio de búsqueda para el algoritmo genético en encontrar una trayectoria óptima. Es decir, encontrar $f(x, x_S, x_T)$ optimo dentro del conjunto D. El algoritmo genético es un algoritmo que resuelve un problema de optimización.

$$\min_{x \in D} f(x, x_S, x_T), \text{ Donde } f : D \to R \tag{3.2}$$

El algoritmo genético es un método de búsqueda estocástico basado en el principio del sistema genético natural. el GA tiene tres operaciones principales que se desarrollan de manera iterativa: i) reproducción/selección, ii) Cruzamiento/Combinación, y iii) Mutación. En cada iteración k, el GA comienza con una población seleccionado de manera aleatoria $P(k) = \{S_1^k, S_2^k, .., S_m^k\}$ con m el tamaño de la población que representa las posibles soluciones en forma de cromosomas o cadenas. Cada cromosoma S_i^t corresponde una solución dentro del conjunto D y tiene la forma

$$S_i^k = [\varsigma_l, \varsigma_{l-1}, \ldots, \varsigma_2, \varsigma_1] \text{ donde } \varsigma_i \in D \quad \forall i = 1, 2, \ldots, l$$

Donde l es el tamaño de la cromosoma y $\varsigma_\cdot = g(\cdot,\cdot)$ componente que pertenece al conjunto D.

La segunda operación del algoritmo genético consiste en realizar el cruzamiento de los cromosomas. Un par de cromosomas son candidatos a realizar el cruzamiento si existe una intersección en sus elementos, es decir, si $S_a^k = [\varsigma_l^a, \varsigma_{l-1}^a, \ldots, \varsigma_2^a, \varsigma_1^a]$ y $S_b^k = [\varsigma_l^b, \varsigma_{l-1}^b, \ldots, \varsigma_2^b, \varsigma_1^b]$ pertenecen a D tal que $S_a^t \cap S_b^t \neq \emptyset$ entonces

$$\begin{aligned} S_{a'}^k &= [\varsigma_l^a, \varsigma_{l-1}^a, \ldots \varsigma_i^{ab}, \ldots, \varsigma_2^b, \varsigma_1^b] \\ S_{b'}^k &= [\varsigma_l^b, \varsigma_{l-1}^b, \ldots \varsigma_j^{ab}, \ldots, \varsigma_2^a, \varsigma_1^a] \end{aligned}$$

Donde $S_{a'}^t$ y $S_{b'}^t$ son los hijos creados por la operación de cruzamiento entres dos cromosomas compatibles. Este proceso nos generan nuevas cromosomas que son posibles soluciones óptimas.

La tercera operación consiste en realizar una mutación. La mutación nos permite buscar soluciones globales y consiste en remplazar un número de cromosomas con aptitudes pobres de la población por cromosomas tomadas del conjunto D de forma aleatoria.

Teorema 3.3 *La operación de mutación del RGA permite encontrar una solución global dentro del conjunto D si $n \to \infty$.*

Demostración. Se calcula la aptitud de cada cromosoma de la operación de mutación $P(Mut) = [fit(S_1^{Mut}), fit(S_2^{Mut}), \ldots, fit(S_n^{Mut})]$ donde n es el número de mutaciones que se realizan y fit es la aptitud que representa la distancia euclidiana. La aptitud total es $Fit = \sum_{i=1}^{n} fit(S_i^{Mut})$. Por lo tanto, la probabilidad de selección p_i de una cromosoma S_i para $i = 1, 2, \ldots, n$ tenemos

$$p_i = \frac{fit(S_i^{Mut})}{Fit} \tag{3.3}$$

Entonces la probabilidad de encontrar una solución óptima p^M dentro del conjunto D por medio de la mutación se puede expresar de la siguiente manera.

$$p^M = \lim_{n\to\infty} (1 - p_i) = \lim_{n\to\infty} \left(1 - \frac{fit(S_i^M)}{\sum_{i=1}^{n} fit(S_i^M)} \right) = 1 \tag{3.4}$$

Podemos asegurar que dentro de la operación de mutación podemos encontrar una solución óptima con probabilidad $p^M = 1$ en una operación infinita. En la práctica este

resultado es poco favorable debido que implica una búsqueda global dentro del conjunto D. ■

El algoritmo RGA tiene diferentes operaciones que permite encontrar soluciones favorables en el proceso de optimización. Las etapas principales del RGA se describen como sigue:

1. Generar una población inicial aleatoria $P(k)$ de tamaño m que pertenecen en el conjunto D.
2. Calcular el valor de aptitud fit, es decir, $fit(S_i) = dis(S_i)$ distancia euclidiana de cada cromosoma S_i $(i = 1, 2, \ldots, m)$.
3. Ordenar la población de menor a mayor aptitud tal que: $fit(S_1) \geq fit(S_2) \geq \cdots \geq fit(S_M)$.
4. Obtener un conjunto en el cruzamiento entre las cromosomas que tienen intersecciones, $S_i \cap S_j \rightarrow S_{ij}, S_{ji}$.
5. Construir la siguiente población $P(k+1)$ remplazando las cromosomas con aptitudes pobres con el conjunto realizado por el cruzamiento.
6. Realizar mutación aleatoria a las cromosomas con aptitudes pobres.
7. Ir a la etapa dos.

Donde D es un conjunto del proceso Roadmap. Estas etapas conforman el algoritmo genético con la combinación del modelado Roadmap para la búsqueda de una trayectoria óptima. En la siguiente sección comprobamos la convergencia del algoritmo propuesto.

Prueba de convergencia del RGA

En esta sección presentamos la prueba de convergencia del algoritmo RGA. Dicho algoritmo realiza una búsqueda dentro de un conjunto D para encontrar una solución óptima $f(\cdot)$ donde la probabilidad de encontrar la solución óptima global es de uno cuando el número de iteraciones va a infinito. El algoritmo comienza con una población definida aleatoriamente y esta población mejora sus aptitudes en cada iteración. Este procedimiento se puede plantear como una cadena de Markov, debido a que cada iteración no es independiente de la población inicial. Cada cromosoma de la población tiene una aptitud

que se denomina como fit, al cuantificar cada cromosoma podemos encontrar relación entre dos cromosomas, es decir, dado dos cromosomas $S_a, S_b \in D$ es posible encontrar una de las siguientes relaciones.

$$\begin{aligned} fit(S_a) &> fit(S_b) \\ fit(S_a) &= fit(S_b) \\ fit(S_a) &< fit(S_b) \end{aligned}$$

De esta manera, es posible clasificar cada cromosoma de acuerdo a su aptitud. En el caso del algoritmo RGA se considera como un algoritmo elitista donde sobreviven los cromosomas con mejor aptitud; en el problema RGA se refiere a las trayectorias con distancias euclidianas cortas. Estas tienen una mayor probabilidad de sobrevivir a la siguiente iteración. Por otra parte, cuando las cromosomas tienen la misma aptitud una de ellas es eliminada, de tal forma que el conjunto de posibles soluciones $P(k)$ del espacio de búsqueda D son linealmente independientes si $\forall S_i \in D, S_i \notin \langle D - S_i \rangle$.

El modelo matemático del problema RGA se puede representar como un modelo estocástico de la cadena de Markov finito, figura 3.6. Donde cada cromosoma tiene la probabilidad de sobrevivir a la transición del estado Q_{ij} al estado $Q_{i(j+1)}$. Que representa una transición en la cadena de Markov donde la probabilidad de ocurrencia $\rho_{ji,ik} > 0$ para $i = 1, 2, \ldots, n$ y ,$k, j = 1, 2, \ldots, m$. En el algoritmo RGA

Los operadores genéticos (selección, cruzamiento y mutación) en alguna iteración k o generación crea una población $P(k)$ con la probabilidad de solución f^k, dicha población preserva a los mejores cromosomas de la población previa $P(k-1)$ con f^{k-1}. Es decir, de la creación de $P(k+1)$ sobre la población $P(k)$ puede verse como una transición de Markov de $H(k)$ a $H(k+1)$.

$$H\{Q_{k+1} = f^{k+1} | Q_k = f^k\} = H(f^{k+1}, f^k) \tag{3.5}$$

Teorema 3.4 *Si el algoritmo genético Roadmap RGA es un proceso elitista entonces la probabilidad de encontrar una solución f^* dentro del conjunto de búsqueda D es exponencial.*

Demostración. El proceso genético es elitista donde el estado Q_1 de la primera iteración puede cambiar con el resto de las cromosomas con una probabilidad de transición

$$H(Q_1 = f^* \forall 0 < \tau \leq m) = \sum_{i=2}^{m} \rho_{i1,11} = \frac{m-1}{m} \tag{3.6}$$

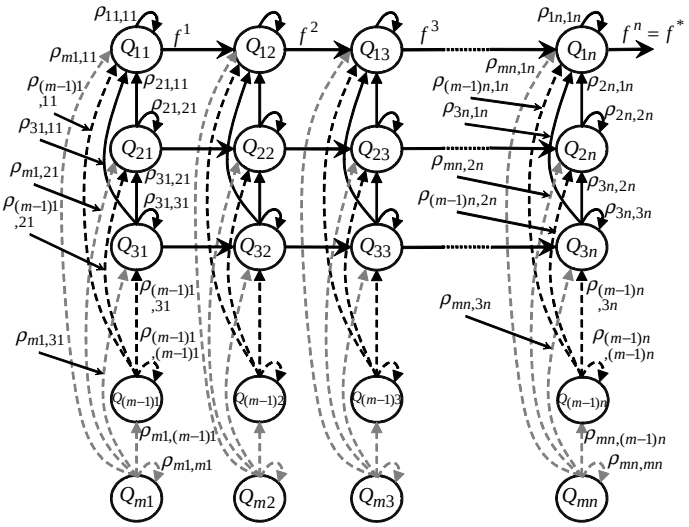

Figura 3.6: RGA modelado como una cadena de Markov finito.

Demostración. Donde m es el tamaño de la población. Si para todo $\alpha, \beta \in D$, existe $0 < \tau \leq m$ tal que $H^\tau(\alpha, \beta) \geq \epsilon > 0$.con

$$\epsilon = \min \left\{ H^\tau(\alpha, \beta) \forall 0 < \tau \leq m \right\} \leq 1 \tag{3.7}$$

Esto implica que dado algún estado Q_t la probabilidad de transición en el tiempo t, esta entre t y $t + m$ y es al menos ϵ, o sin pérdida de generalidad el complemento

$$H(Q_t \neq f^* \forall t < \tau \leq t + m) \geq 1 - \epsilon \tag{3.8}$$

■

En general, la transición en la iteración $k + 1$, se desarrolla como.

$$\begin{aligned} H(Q_{k+1}) &= H(Q_t \neq f^* \forall 0 < t \leq (k+1)m) \\ &= H(Q_t \neq f^* \forall 0 < t \leq km) H(Q_t \neq f^* \forall km < t \leq (k+1)m) \end{aligned} \tag{3.9}$$

Utilizando la ecuación 3.8 tenemos que:

$$\begin{aligned} &\leq H(Q_t \neq f^* \forall 0 < t \leq km)(1 - \epsilon) \\ &\leq H(Q_t \neq f^* \forall 0 < t \leq (k-1)m)(1 - \epsilon)^2 \\ &\leq H(Q_t \neq f^* \forall 0 < t \leq 0) H(Q_t \neq f^* \forall 0 < t \leq m)(1 - \epsilon)^k \\ &= \frac{1}{m}(1 - \epsilon)^k \end{aligned}$$

Donde $H(P_t \neq f^* \forall 0 < t \leq 0) = 1$ por verdad vacua. Entonces

$$\begin{aligned} \lim_{k\to\infty} H(Q_{k+1}) &\leq \lim_{k\to\infty} \frac{1}{m}(1-\epsilon)^k \\ &= \frac{1}{m}\lim_{k\to\infty}(1-\epsilon)^k = 0 \end{aligned} \tag{3.10}$$

Por definición $0 < \epsilon \leq 1$ lo que significa que el algoritmo RGA converge y encuentra una solución f^*, además converge de manera exponencialmente con dos parámetros: el tamaño de la población m y la cantidad de iteraciones k. ■

A continuación se presentan los resultados obtenidos de los algoritmos antes presentados.

3.3. Resultados

En esta sección mostramos los resultados obtenidos de los algoritmos de planeación de trayectorias propuesto en este trabajo. En la primera parte se presentan los resultados del algoritmo histograma polar. Posteriormente, los resultados del Roadmap Algoritmo genético. Estos algoritmos se presentan en una navegación autónoma de un robot móvil.

Resultados del algoritmo histograma polar.

Definimos un mapa previo conocido con un estado inicial S_I y un estado objetivo S_T para la tarea de navegación como se muestra en la figura (3.7). En esta figura, los obstáculos que deben evitarse son claros para la navegación en el entorno conocido, por lo que encontrar una trayectoria óptima es simple. Sin embargo, estamos interesados en encontrar un camino para la navegación autónoma cuando el entorno mencionado anteriormente es completamente desconocido. Es decir, no se ha observado el entorno, $G_O \in \emptyset$. Por lo tanto, la ruta del siguiente estado S_n no existe $\forall t > t_0$.

El radio de observación del robot es $r_0 = 8{,}3m$ La figura (3.8) corresponde a la primera iteración del algoritmo de navegación autónoma propuesto anteriormente (navegación 1). En este caso $A(q) = q^2$ con $q = 1m$, por lo tanto, los obstáculos se configuran de la misma manera, es decir, un Landmark tiene la forma del robot móvil $A(q) = q^2$ esto para configurar el entorno y evitar colisiones con los obstáculos.

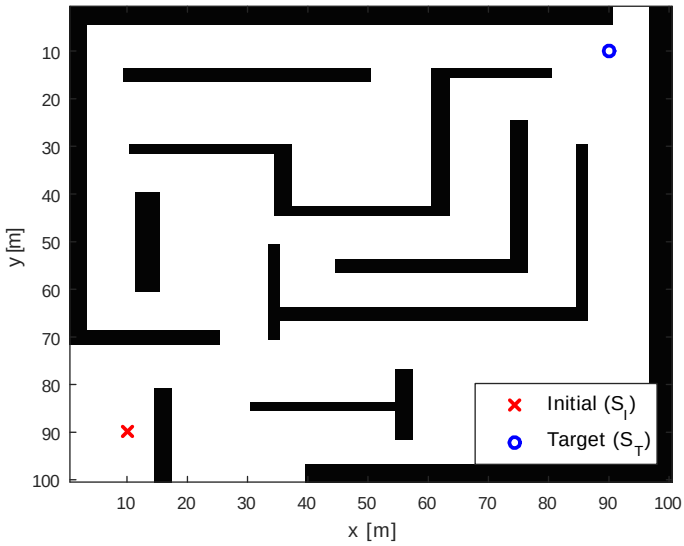

Figura 3.7: Entorno conocido para el histograma polar.

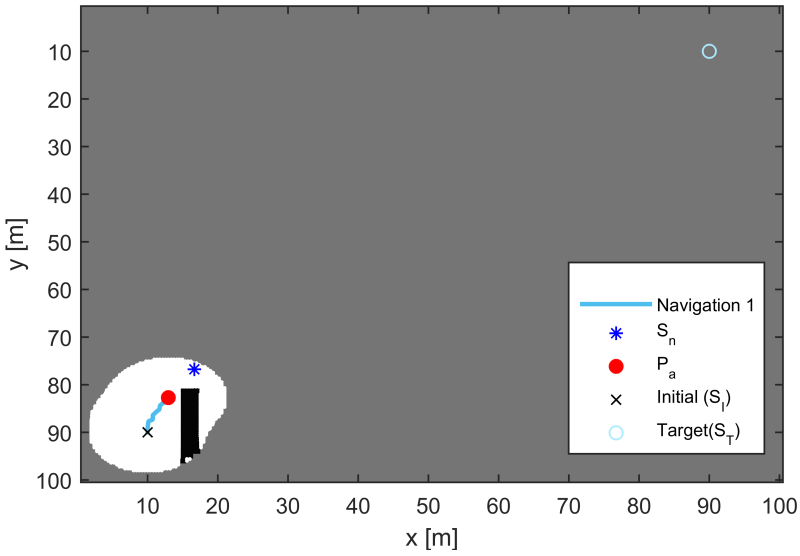

Figura 3.8: Primera iteración de la navegación autónoma con el histograma polar.

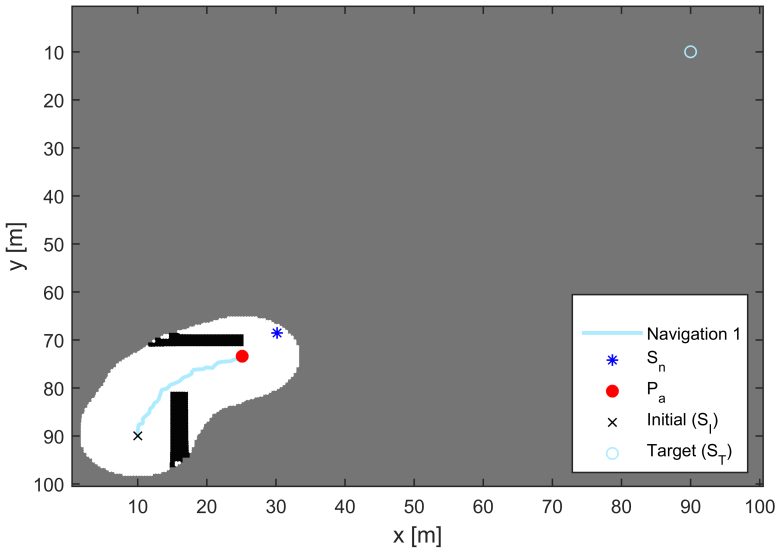

Figura 3.9: tercera iteración de la navegación autónoma con el histograma polar.

La región de observación crece a medida que la navegación evoluciona. Cabe mencionar que en cada iteración, la planeación de trayectorias encuentra un objetivo parcial, en consecuencia, el generador de reglas elige la trayectoria para alcanzar el objetivo parcial, en este caso, se selecciona una línea recta, y en cada $0{,}2m$ se reconstruye el mapa. Por lo tanto, garantizamos una resolución más pequeña que el tamaño del robot. La figura (3.9) muestra la navegación autónoma en la tercera iteración del algoritmo.

Para evaluar el desempeño del algoritmo propuesto. La navegación autónoma se realiza cuando el entorno es totalmente conocido, es decir, que la región observada $G_O \subseteq G_E$. Para esta prueba, existe una trayectoria global y óptima que se compara con el algoritmo de navegación autónoma propuesto para los entornos desconocido, véase la figura (3.10). Como resultado, obtenemos una ruta de navegación muy cercana a la navegación con el entorno conocido. En esta prueba donde se desconoce el entorno, el algoritmo de navegación autónoma requiere 19 iteraciones para alcanzar el punto objetivo.

En este trabajo, también implementamos el algoritmo de navegación autónoma utilizando una planeación de trayectorias de ocupación de rejilla (navegación 2). Algoritmo que se puede encontrar en el trabajo [106] y [91]. Como se muestra en la figura (3.11), este algoritmo toma una trayectoria diferente del algoritmo que propusimos.

Para validar el algoritmo, realizamos varias pruebas para estudiar la longitud de la suboptimización en promedio, E_{PP}. Antes, definimos un índice de la longitud de la trayec-

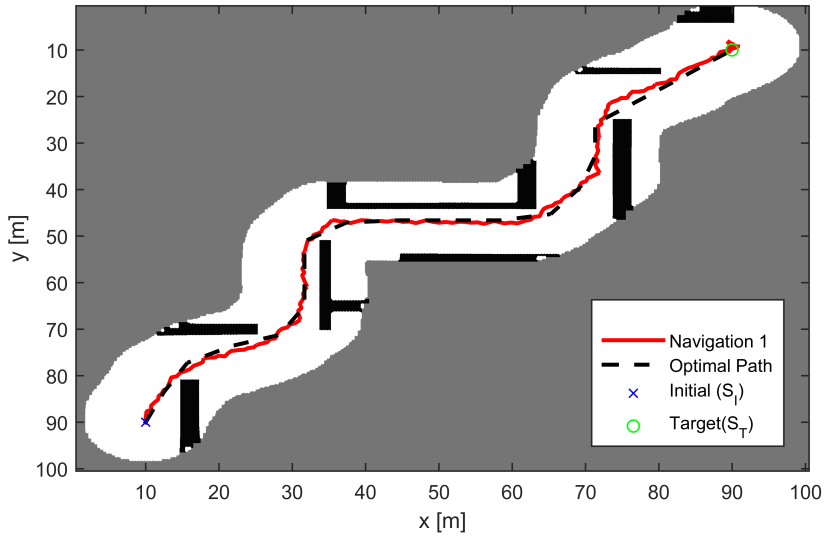

Figura 3.10: La trayectoria óptima (línea discontinua) algoritmo en un entorno conocido y la navegación 1 en el entorno desconocido.

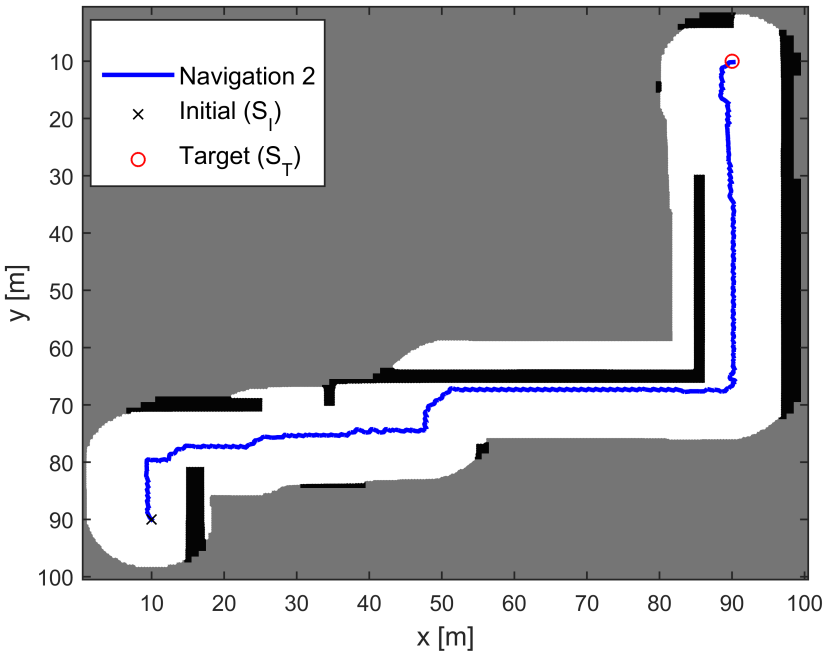

Figura 3.11: Algoritmo de navegación con ocupación de rejilla.

toria como

$$l_{sub} = \frac{effective\ path}{optimal\ path} \tag{3.11}$$

El promedio de longitud l_{sub} se define como $E[l_{sub}]$ y representa la efectividad del algoritmo de navegación. Donde $l_{sub} \geq 1$ porque la trayectoria efectiva del algoritmo de navegación es más larga que la trayectoria óptima.

Para calcular la complejidad de la tarea de navegación, definimos la densidad de obstáculos como

$$d_{obs} = \frac{\sum_{n \in G_O} S_{obs}(n)}{\|G_E\|} \tag{3.12}$$

Si $d_{obs} = 0$, el entorno está libre de obstáculos, si $d_{obs} = 1$, todo el entorno está ocupado por los obstáculos.

Definimos el porcentaje de error de la trayectoria como

$$E_{PP} = \frac{effective\ path - optimal\ path}{100} \tag{3.13}$$

Primero estudiamos el desempeño de los algoritmos utilizando el entorno conocido. Generamos 120 entornos aleatorios con una distribución uniforme, ya que cada entorno es bidimensional con un espacio de rejilla de $100 \times 100m$. El rango de densidad de los obstáculos, ecuación.(3.12) es $[0{,}05, 0{,}6]$ porque la densidad máxima de obstáculos para el algoritmo de navegación 2 es $d_{obs} = 0{,}6$. es decir, si la densidad del obstáculo es mayor que $d_{obs} > 0{,}6$, el algoritmo no encuentra una trayectoria entre el estado inicial y el estado objetivo.

El índice promedio de longitud de la trayectoria $E[l_{sub}]$ en la ecuación (3.11) con respecto a la densidad de los obstáculos en la ecuación (3.12) en un entorno conocido se muestra en la figura (3.12). El promedio de la longitud de trayectoria $E[l_{sub}]$ diverge a medida que aumenta la densidad de los obstáculos. Al comparar la navegación autónoma con el algoritmo SM SLAM (navegación 1) con la navegación 2, podemos observar que la navegación 2 crece más rápido al aumentar la densidad de los obstáculos. Por ejemplo: para la navegación 1 con densidad de obstáculos $d_{obs} = 0{,}3$ tenemos $E[l_{sub}] = 1{,}053$, mientras que para la navegación 2 con la misma densidad, el promedio de la longitud de trayectoria $E[l_{sub}] = 1{,}152$. Si consideramos el error, entonces para la navegación 1, $E_{PP} = 0{,}06$ y para navegación 2, $E_{PP} = 0{,}172$. En otras palabras, el error aumenta de

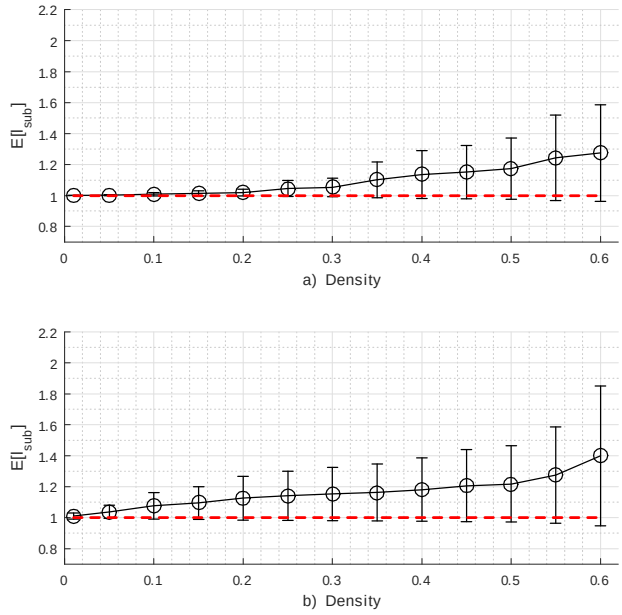

Figura 3.12: Desempeño del algoritmo a) método de histograma polar y b) descomposición de rejilla en entornos conocidos.

6 % a 17,2 %, respectivamente. Y para la densidad del obstáculo $d_{obs} = 0{,}6$, el error para la navegación 1 es 31,13 %, y para la navegación 2 es 45,18 %.

Las pruebas de efectividad de los algoritmos de navegación autónomo en entornos desconocidos se realizan de la misma manera que en los entornos conocidos, los resultados obtenidos se muestran en la figura (3.13). De igual manera, se generan 120 entornos aleatorios con distribución uniforme. Donde el promedio de la longitud de trayectoria aumenta con respecto al aumento en la densidad de los obstáculos. Por ejemplo: para la navegación autónoma con SM SLAM en densidad de obstáculos $d_{obs} = 0{,}3$, tenemos $E[l_{sub}] = 1{,}106$, mientras que para la navegación 2 con la misma densidad, el promedio de la longitud trayectoria es $E[l_{sub}] = 1{,}183$. Si consideramos el error, entonces para la navegación 1, $E_{PP} = 0{,}1206$ y para navegación 2, $E_{PP} = 0{,}2076$. En otras palabras, el error aumenta de 12,06 % a 20,76 %, respectivamente. Y para la densidad del obstáculo $d_{obs} = 0{,}6$, el error para la navegación 1 es de 39,11 %, y para la navegación 2 es del 53,65 %.

En el algoritmo de histograma tenemos el parámetro de ajuste ($Threshold$) que obtiene diferentes resultados; por lo tanto, se realizan varias pruebas con diferentes valores de

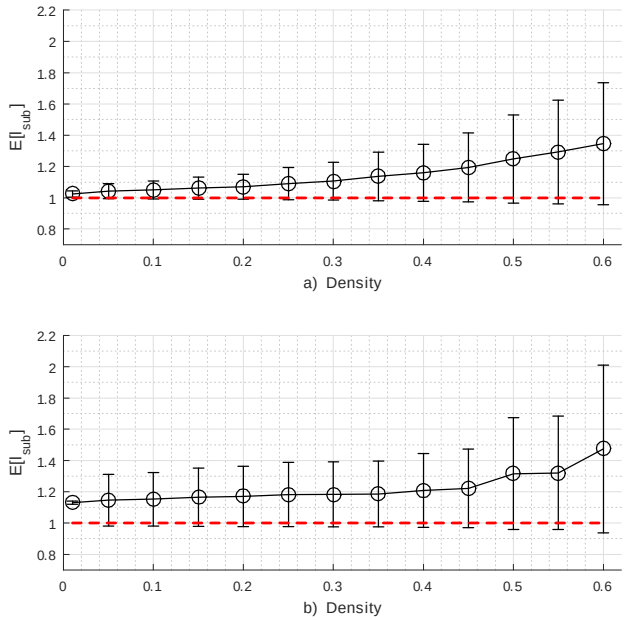

Figura 3.13: Desempeño del algoritmo a) método de histograma polar y b) descomposición de rejilla en entornos desconocidos.

Threshold, para cada análisis se utilizaron 240 entornos diferentes. En consecuencia, el valor *Threshold* permite reducir o aumentar el conjunto de búsqueda. Como resultado, en valores de 2, 3 y 4 de *Threshold*, el error del porcentaje de la trayectoria, $E_{PP} < 0{,}5$. Sin embargo, si el valor del *Threshold* es 5, 6 o 7, entonces el error de porcentaje, $E_{PP} > 1$ en $Density > 0{,}4$, como se muestra en la figura (3.14).

Por otro lado, si la variable *Threshold* es un valor cercano al radio de observación r_O, entonces el número de iteraciones se reduce para encontrar el estado objetivo. En el caso de que el *Threshold* sea un valor cercano al tamaño del robot q, se incrementa el número de iteraciones. El comportamiento mencionado anteriormente se muestra en la figura (3.15).

Los siguientes resultados que se presentan son del algoritmo RGA presentado anteriormente.

Resultados del algoritmo RGA

Los siguientes resultados fueron obtenidos al realizar la simulación en el programa de matlab, el algoritmo RGA propuesto se implementó en tres diferentes entornos: conocidos,

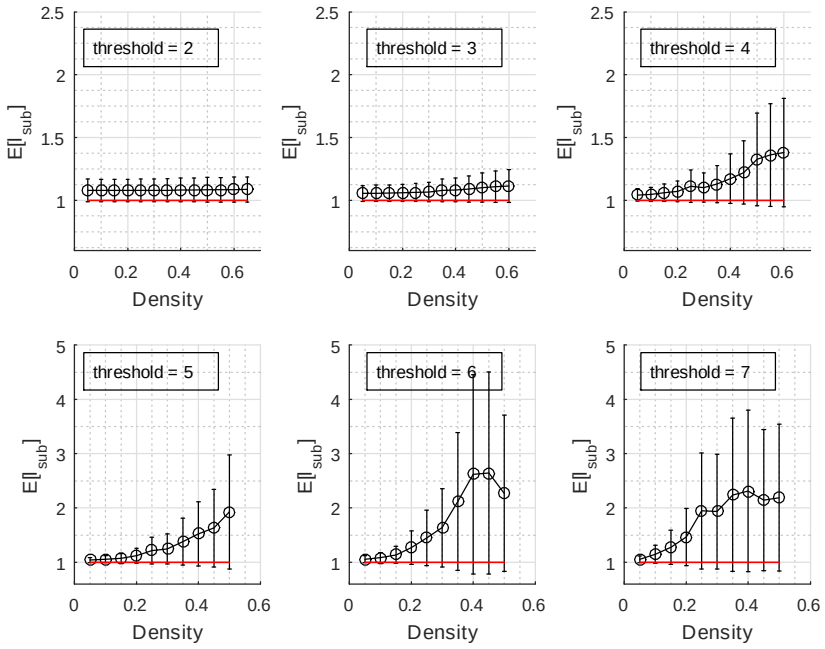

Figura 3.14: Desempeño del algoritmo histograma polar con el parámetro *Threshold*.

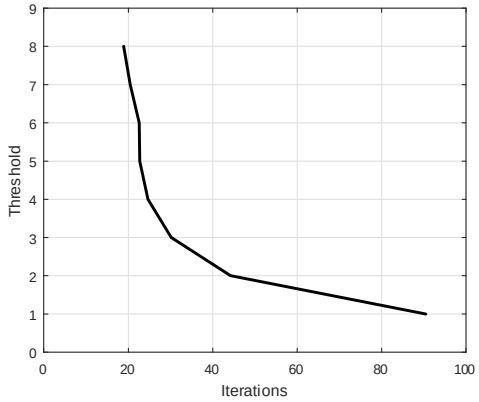

Figura 3.15: El comportamiento de la variable *Threshold* con el número de iteraciones.

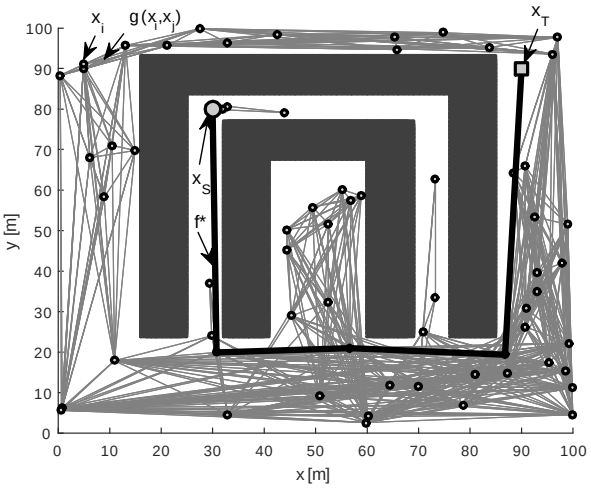

Figura 3.16: Algoritmo RGA en entornos conocidos.

parcialmente conocidos y desconocidos. El tamaño de los entornos son de $100 \times 100m$ en el que se busca una solución para encontrar una trayectoria desde el punto inicial x_S al punto x_T

En la figura 3.16 se muestra una planeación de trayectorias basada en RGA para encontrar una solución f^* dentro del entorno conocido, para el problema de optimización se generan 60 objetivos locales x_i con el espacio de búsqueda D generada por las trayectorias de los objetivos locales que no intersectan con el conjunto de los obstáculos, por lo que se convierte en un problema de optimización para encontrar una trayectoria optima y en tiempo finito como es el caso del algoritmo RGA.

En un entorno con obstáculos generado previamente de $100 \times 100m$ se generan 120 posibles objetivos de manera aleatoria x_i, por lo tanto, el espacio de búsqueda D seria todas las trayectorias $g(x_{i,X_j})$ que no intersecan con el conjunto de los obstáculos, donde el algoritmo genético resuelve el problema de optimización para encontrar una solución al problema de planeación de trayectorias, para el problema presentado anteriormente se encuentra que el algoritmo RGA converge en 40 iteraciones como se muestra en la figura 3.17, para estos resultados se realizaron 100 pruebas para cada número de iteraciones y como se podrá observar en la gráfica, el algoritmo genético converge con mayor probabilidad en los 40 iteraciones

Los siguientes resultados fueron implementados en entornos parcialmente desconocidos, $B_{obs}(0) \neq \emptyset$. La solución f^* de la planeación de trayectorias es parcial debido a que el entorno $B_{obs}(t)$ es variante en el tiempo. En la figura 3.18 muestra una solución parcial

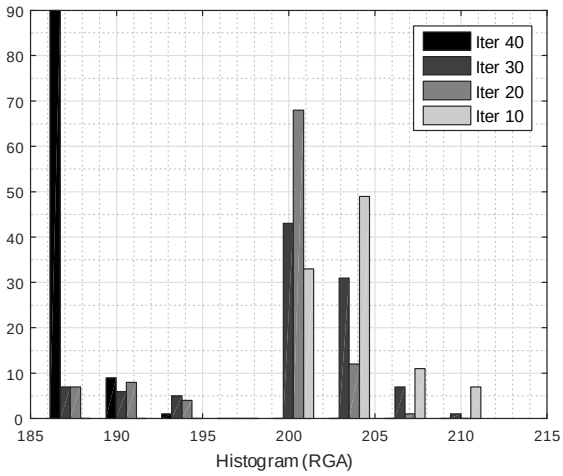

Figura 3.17: Desempeño del algoritmo RGA.

f^* de un punto inicial x_S al punto objetivo x_T para el entorno parcialmente conocido con el algoritmo propuesto RGA.

Para la tarea de navegación del robot o sistema en los entornos parcialmente conocidos o totalmente desconocidos se utilizó el algoritmo SLAM para construir el entorno y conocer la posición del robot. Al comienzo de la navegación en el entorno parcialmente conocido se tiene una trayectoria planeada de navegación por el algoritmo RGA, sin embargo, si sobre la trayectoria planeada se encuentra un obstáculo entonces se requiere utilizar el algoritmo RGA para buscar una nueva ruta dentro del entorno construido por el SLAM, conjunto B_{SLAM}. Este proceso se presenta en la figura 3.19 donde la trayectoria planeada pertenece al conjunto de los obstáculos que impiden llegar a la meta, $f^* \subset B_{obs}(t)$, por lo tanto, se requiere buscar una trayectoria nueva utilizando RGA que permita alcanzar el objetivo.

En la figura 3.20.se muestra el resultado global de la navegación del robot del punto x_S al punto x_T en un entorno variante con un conocimiento inicial de los obstáculos y con el algoritmo SLAM combinado con RGA que permite alcanzar el objetivo y resolver el problema de planeación de trayectorias en entornos parcialmente conocidos.

El algoritmo propuesto RGA también se implementó en entornos totalmente desconocidos, $B_{obs}(0) = \emptyset$. En estos entornos se requiere del algoritmo SLAM para conocer el entorno B_{SLAM} y la posición del robot, de esta manera, cuando se encuentra un obstáculo que contiene la trayectoria planeada entonces se busca una nueva trayectoria con el algo-

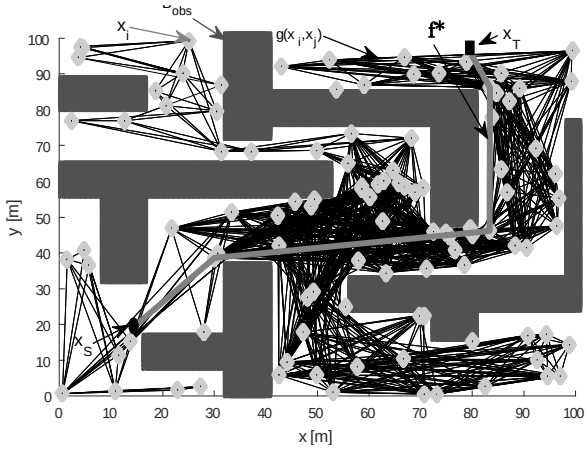

Figura 3.18: Algoritmo RGA en entornos parcialmente conocidos, parte 1.

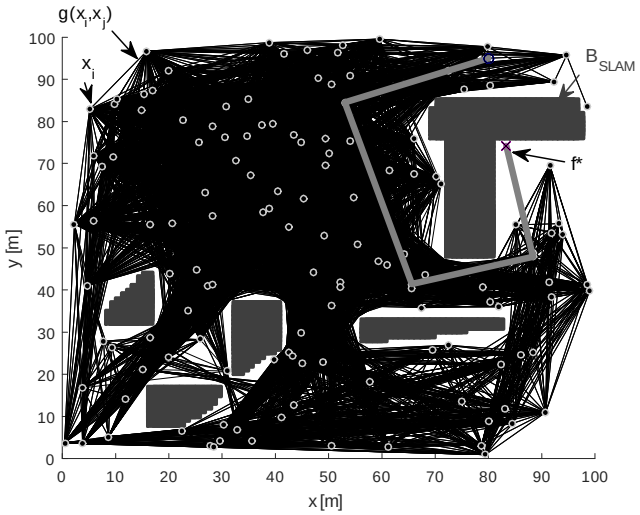

Figura 3.19: Algoritmo RGA en entornos parcialmente conocidos, parte 2.

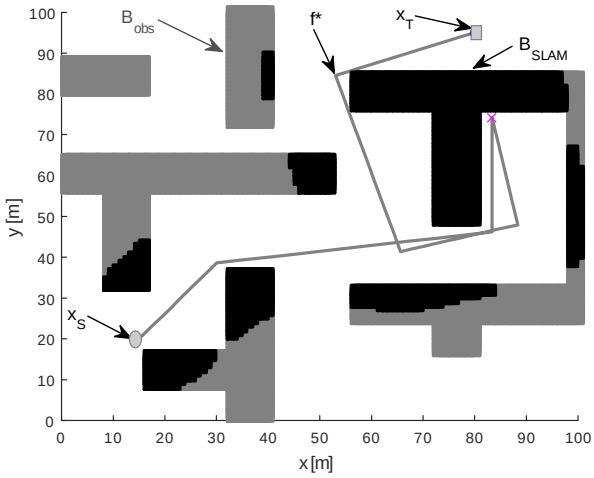

Figura 3.20: Algoritmo RGA en entornos parcialmente conocidos, parte 3.

ritmo RGA sobre el mapa B_{SLAM} hasta que se alcanza el punto objetivo El resultado que se obtiene se muestra en la figura 3.21.

Existen diferentes algoritmos de planeación de trayectorias que se implementan en entornos desconocidos como: histogramas, atracción de campo o descomposición de celdas. Estos algoritmos tienen la desventaja de encontrar soluciones locales, como se muestra en la figura 3.22 donde se implementó un algoritmo de histograma que encontró una solución local.

El algoritmo de planeación de trayectorias es una etapa fundamental para lograr el objetivo de la navegación autónoma de cualquier sistema. Sin embargo, se requiere de un controlador de movimiento en el sistema para llevar a cabo la navegación con la trayectoria encontrada por el algoritmo de planeación de trayectorias. En el siguiente capítulo se propone un algoritmo de seguimientos de trayectorias robusto ante las perturbaciones acotadas e incertidumbres en los sistemas.

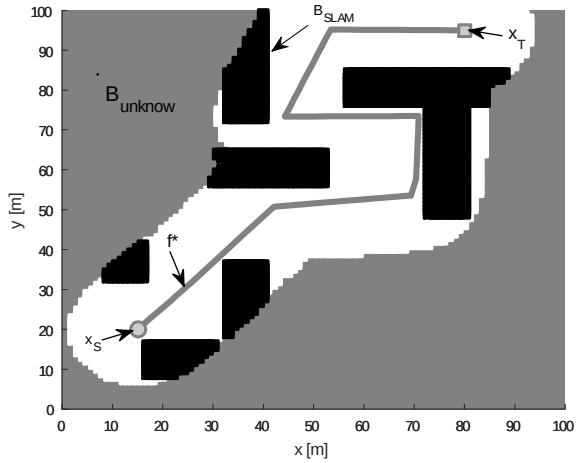

Figura 3.21: Algoritmo RGA en entornos totalmente desconocidos.

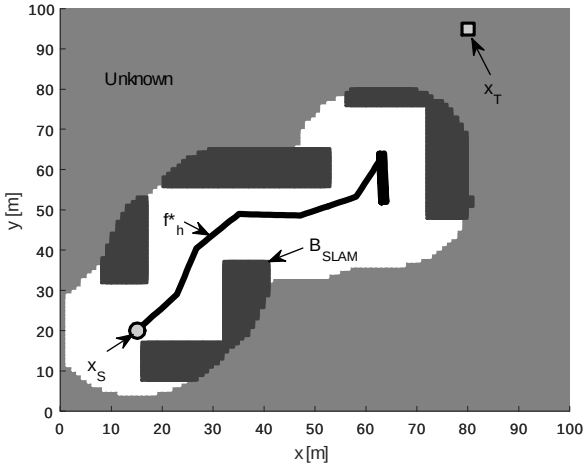

Figura 3.22: Algoritmo histograma polar en entornos desconocidos.

Capítulo 4

Seguimiento de trayectorias.

La navegación autónoma tiene diversas etapas, tales como: la percepción del entorno, localización y mapeo, planeación de trayectorias y por último se requiere de un controlador de movimiento.

El controlador de movimiento se requiere para ejecutar la navegación de los sistemas es decir lograr el seguimiento de trayectorias obtenida del algoritmo de planeación de trayectorias. Existen diversos controladores que realizan esta tarea. Tal como: controladores clásicos (PID: proporcional, integral y derivativo, PD: proporcional, derivativo), controladores no lineales, controlador adaptativo, entre otros. Los controladores de movimiento son esenciales en la navegación autónoma que contribuyen en el desempeño de lograr con los objetivos de navegación en entornos desconocidos. En otras palabras, diseñar un controlador de movimiento robusto ante las incertidumbres y perturbaciones que permitan atenuar los errores en los estados del sistema es importante para lograr los objetivos de navegación. En este capítulo, presentamos un controlador de seguimiento de trayectorias para un cuadricóptero (figura 4.1). Este controlador nos ayuda a proporcionar las velocidades en el marco del cuerpo que se utiliza para resolver el problema del SLAM, es decir, nosotros podemos obtener $U = [u, v, w, p, q, r]^T$ que nos ayuda a realizar la localización del sistema. Nosotros diseñamos un controlador de seguimiento de trayectorias basado en la técnica de modos deslizantes de orden superior o también llamado Super- Twisting generalizado. Esta teoría presenta grandes ventajas ante controladores clásicos, debido a que cuentan con la característica de ser robustos ante las perturbaciones o incertidumbres en los modelos matemáticos de los sistemas.

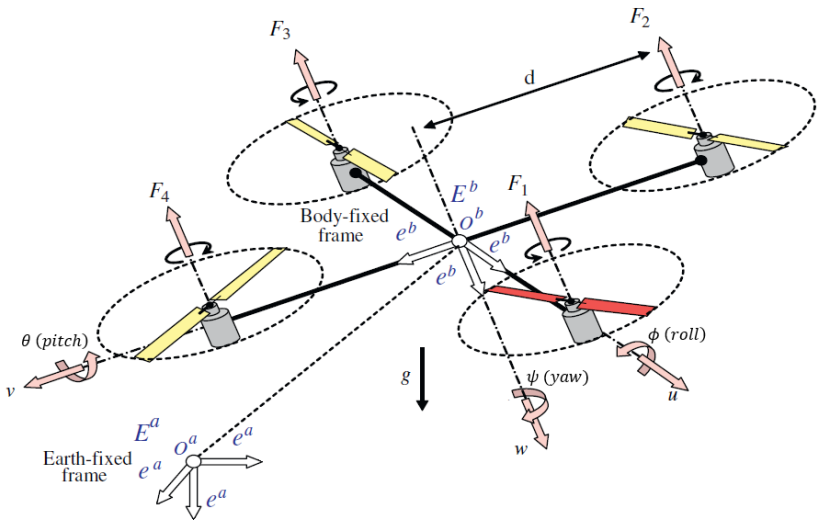

Figura 4.1: Estructura del cuadricóptero tomado de Derafa (2012).

4.1. Modelo matemático.

El modelo dinámico de un helicóptero es estudiado en diversos trabajos [62], [60], [59] y [58] las cuales se utilizan para diseñar técnicas o algoritmos de control antes de una implementación experimental. La posición absoluta y la altura se describen como $X^r = [x, y, z]^T$ y los ángulos de Euler $\Theta = [\phi, \theta, \psi]^T$ correspondientes a una convención aeronáutica. Las ecuaciones dinámicas de un cuadricóptero son dados como:

$$\ddot{X} = \frac{1}{m} R(\Theta) \left[F_{prop} - A_F(U) \right] + g V_3 \tag{4.1}$$

$$\ddot{\Theta} = \left(JM(\Theta) \right)^{-1} \left[T_{prop} - JN(\Theta, \dot{\Theta}) - A_T(U) - \left(M(\Theta)\dot{\Theta} \right) \times \left(JM(\Theta)\dot{\Theta} \right) \right] \tag{4.2}$$

$M(\Theta)$ y $N(\Theta, \dot{\Theta})$ tienen la siguiente forma:

$$M(\Theta) = \begin{bmatrix} 1 & 0 & -S_\theta \\ 0 & C_\phi & C_\theta S_\phi \\ 0 & -S_\phi & C_\theta C_\phi \end{bmatrix} \tag{4.3}$$

$$N(\Theta,\dot{\Theta}) = \begin{bmatrix} -C_\theta\dot{\theta}\dot{\psi} \\ -S_\phi\dot{\phi}\dot{\theta} + C_\phi C_\theta\dot{\phi}\dot{\psi} - S_\phi S_\theta\dot{\theta}\dot{\psi} \\ -C_\phi\dot{\phi}\dot{\theta} - S_\phi C_\theta\dot{\phi}\dot{\psi} - C_\phi S_\theta\dot{\theta}\dot{\psi} \end{bmatrix}$$

La matriz de rotación ortogonal $R(\Theta)$ se define como:

$$R(\Theta) = \begin{bmatrix} C_\psi C_\theta & C_\psi S_\theta S_\phi - S_\psi C_\phi & C_\psi S_\theta C_\phi + S_\psi S_\phi \\ S_\psi C_\theta & S_\psi S_\theta S_\phi - C_\psi C_\phi & S_\psi S_\theta C_\phi - C_\psi S_\phi \\ -S_\theta & C_\theta S_\phi & C_\psi C_\theta \end{bmatrix} \tag{4.4}$$

Donde $cos(\star) = C_\star$ y $sin(\star) = S_\star$. La inercia del cuadricóptero está dada por $J = diag(I_x, I_y, I_z)$. g es la gravedad; el vector $V_3 = [0,0,1]^T$, se representa de esta forma ya que la gravedad sólo actúa en el eje z; m es la masa del vehículo; U es la velocidad del cuadricóptero con respecto al aire; $A_F(U)$ y $A_T(U)$ son vectores representadas por funciones no lineales complejas que describen la fuerza aerodinámica y el torque de las hélices. Finalmente, F_{prop} y T_{prop} son vectores de fuerza y torque del sistema, dadas como:

$$\begin{aligned} F_{prop} &= -\sum_{i=1}^{4} F_i V_3 \\ T_{prop} &= \begin{bmatrix} d(F_2 - F_4) \\ d(F_1 - F_3) \\ c(F_1 - F_2 + F_3 - F_4) \end{bmatrix} \end{aligned}$$

Donde d es la distancia del centro de masa al eje del rotor. $c > 0$ es un factor de resistencia.

4.2. Controlador Super-Twisting Generalizado.

La estrategia de control no lineal para resolver el problema de seguimiento de trayectorias de un vehículo aéreo no tripulado bajo perturbaciones. Donde dicha estrategia se basa en el Algoritmo Super-Twisting Generalizado; es una técnica de modos deslizantes de segundo orden, la cual es capaz de asegurar robustez con respecto a errores de modelado y perturbaciones externas acotadas debido a los términos de corrección lineales añadidos respecto al algoritmo Super Twisting convencional. El objetivo del controlador es conseguir un seguimiento de trayectoria adecuado de las posiciones absolutas deseadas y de los ángulos de Euler, mientras se mantiene la estabilidad del ángulo de inclinación y de alabeo, a pesar de la presencia de perturbaciones y las no linealidades del sistema.

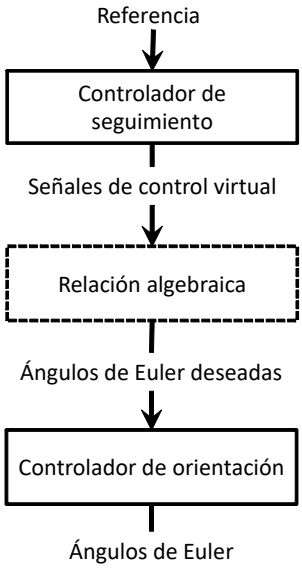

Figura 4.2: Estructura del seguimiento de trayectorias utilizando el controlador Super-Twisting Generalizado.

Para diseñar un controlador de seguimiento de trayectoria [54],[102], [72] en un cuadricóptero dividimos el sistema en dos subsistemas; el primer subsistema consiste en diseñar un controlador para los ángulos de Euler y el segundo subsistema nos ayuda a diseñar un controlador para el posicionamiento del cuadricóptero. En ambos subsistemas utilizamos la teoría de modos deslizantes de orden superior conocida como Super-Twisting en su forma generalizada.

En la figura (4.2) se muestra la estructura de los controladores Super-Twisting generalizado para lograr el seguimiento de trayectorias de una referencia dada (trayectoria deseada). Al implementar un controlador de seguimiento de trayectorias podemos obtener los controles virtuales $[U_x, U_y, U_z]^T$ al aplicar una relación algebraica de los estados con dichos controladores se puede obtener los ángulos de Euler deseados $[\phi_d(t), \theta_d(t), \psi_d(t)]^T$ que posteriormente diseñamos un controlador de orientación para lograr obtener los ángulos deseados en cada estado de la orientación y de esta manera el cuadricóptero comience a navegar en el espacio. Estos ángulos modifican la traslación y rotación del sistema como se presentó en la parte del modelado del cuadricóptero. En consecuencia, se obtiene el seguimiento de trayectorias.

Diseño del controlador de orientación

Diseño de un controlador capaz de manipular los ángulos de Euler. Del modelo (4.2), nosotros definimos la siguiente variable de estados:

$$q_1 = \Theta \qquad q_2 = \dot{\Theta}$$

Entonces el espacio de estados de este modelo está dado por:

$$\dot{q}_1 = q_2 \tag{4.5}$$
$$\dot{q}_2 = f(q) + p(q)u + w(t) \tag{4.6}$$

Donde $w(t) = [w_1, w_2, w_3]^T$ es el vector de perturbaciones; $T_{prop} = u = [u_1, u_2, u_3]^T$ es el vector de la señal de control. los vectores $f(q)$ y $p(q)$ se definen como:

$$\begin{aligned} f(q) &= -\left(JM(\Theta)\right)^{-1}\left[JN(\Theta,\dot{\Theta}) + (M(\Theta)\dot{\Theta}) \times \left(JM(\Theta)\dot{\Theta}\right)\right] \\ p(q) &= \left(JM(\Theta)\right)^{-1} \end{aligned} \tag{4.7}$$

Para el diseño del control, se deben de considerar las siguientes restricciones [57]:

- Restricción 1: X,Θ,$\dot{\Theta}$, pueden medirse por un sensor a bordo del sistema.
- Restricción 2: La cuantificación de $\sum_{i=1}^{4} F_i$ representa el empuje total sobre el cuerpo en el eje z y es estrictamente positivo para superar la gravedad.
- Restricción 3: Los ángulos *pitch*, *roll* y *yaw* tienen un límite de operación
- Restricción 4: Las trayectorias deseadas y su primera derivada, así como su segunda derivada son acotados.
- Restricción 5: La velocidad $\dot{U}$ y la aceleración U del cuadricóptero con respecto al aire están acotadas.

De acuerdo a las restricciones, la matriz (4.7) es no singular y la siguiente desigualdad se satisface.

$$|\dot{w}_i(t)| \leq \delta_i \ , \qquad i = 1, 2, 3.$$

El controlador se diseña con una superficie de deslizamiento dada de la forma.

$$\sigma = \dot{e} + \Gamma e \tag{4.8}$$

Donde $e = q_1^d - q_1$ y $\dot{e} = \dot{q}_1^d - q_2$ es el error y la derivada del error con $q_1^d(t) = [\phi_d(t), \theta_d(t), \psi_d(t)]^T$. $\Gamma \in \mathbb{R}^{3\times 3}$ es una matriz definida positiva.

La ley de control propuesta para controlador el subsistema (4.5) tiene la siguiente forma:

$$u = -p(q)^{-1}\left(v + \ddot{q}_1^d - \Gamma\dot{e} - f(q)\right)$$

Donde v es el controlador Super-Twisting Generalizado de la teoría de modos deslizantes, que se describe como:

$$\begin{aligned} v &= -K_1\phi_1(\sigma) + \lambda & (4.9)\\ \dot{\lambda} &= -K_2\phi_2(\sigma) & (4.10)\end{aligned}$$

Con

$$\begin{aligned} \phi_1(\sigma) &= |\sigma|^{\frac{1}{2}}\, sgn(\sigma) + \sigma \\ \phi_2(\sigma) &= \frac{1}{2}sgn(\sigma) + \frac{3}{2}\,|\sigma|^{\frac{1}{2}}\, sgn(\sigma) + \sigma & (4.11)\end{aligned}$$

Donde $sgn(\cdot)$ es la función signo, descrita como:

$$sgn(x) = \begin{cases} 1 & \text{if } x > 0 \\ -1 & \text{if } x < 0 \end{cases}$$

$K_1 = diag[k_{11}, k_{12}, k_{13}]$ y $K_2 = diag[k_{21}, k_{22}, k_{23}]$ son matrices definidas positivas de ganancias del controlador Super-Twisting Generalizado.

Diseño del controlador de seguimiento.

Diseño del controlador para el posicionamiento del cuadricóptero. Del modelo (4.1) se puede reescribir como:

$$\begin{aligned} \ddot{x} &= \left(\frac{h_1}{m}\right)(F_{prop} - A_F(U)) \\ \ddot{y} &= \left(\frac{h_2}{m}\right)(F_{prop} - A_F(U)) \\ \ddot{z} &= -g + \left(\frac{h_3}{m}\right)(F_{prop} - A_F(U)) & (4.12)\end{aligned}$$

Donde

$$\begin{aligned} h_1 &= sin\theta cos\psi cos\phi + sin\psi sin\phi \\ h_2 &= sin\theta sin\psi cos\phi - cos\psi sin\phi \\ h_3 &= cos\theta cos\phi \end{aligned}$$

La variable de estados se define como:

$$\begin{bmatrix} x_1 \\ x_2 \\ y_1 \\ y_2 \\ z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \\ z \\ \dot{z} \end{bmatrix}$$

Ahora, el sistema dado en su forma matricial se describe de la siguiente manera.

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{y}_1 \\ \dot{y}_2 \\ \dot{z}_1 \\ \dot{z}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ \left(\frac{h_1}{m}\right)(F_{prop} - A_F(U)) \\ y_2 \\ \left(\frac{h_2}{m}\right)(F_{prop} - A_F(U)) \\ z_2 \\ -g + \left(\frac{h_3}{m}\right)(F_{prop} - A_F(U)) \end{bmatrix} \tag{4.13}$$

Para este controlador se utiliza la misma teoría de la sección anterior, tomando como superficie de deslizamiento ς dado como:

$$\varsigma = \dot{r} + \Delta r \tag{4.14}$$

Donde $r = [x_1^d - x_1, y_1^d - y_1, z_1^d - z_1]^T$ y $\dot{r} = [\dot{x}_1^d - x_2, \dot{y}_1^d - y_2, \dot{z}_1^d - z_2]^T$ es un vector de error de posición y la derivada de este error. $\Delta \in \mathbb{R}^{3\times 3}$ es una matriz definida positiva. $A_F(U)$ se considera como una perturbación acotada, por lo tanto, se puede despreciar este término cuando se diseña el controlador debido a que se considera que el sistema (4.13) es libre de perturbaciones. Despreciar el término $A_F(U)$ es posible debido a que el controlador Super-Twisting Generalizado tiene la característica de ser robusto ante perturbaciones acotadas. El controlador Super-Twisting Generalizado está dado como:

$$\begin{aligned} \upsilon_i &= -K_3\phi_3(\varsigma) + \alpha_i \\ \dot{\alpha}_i &= -K_4\phi_4(\varsigma) \end{aligned} \tag{4.15}$$

Con $i = 1, 2, 3$. Donde

$$\begin{aligned} \phi_3(\varsigma) &= |\varsigma|^{\frac{1}{2}}\, sgn(\varsigma) + \varsigma \\ \phi_4(\varsigma) &= \frac{1}{2} sgn(\varsigma) + \frac{3}{2}|\varsigma|^{\frac{1}{2}}\, sgn(\varsigma) + \varsigma \end{aligned} \tag{4.16}$$

$K_3 = diag[k_{31}, k_{32}, k_{33}]$ y $K_4 = diag[k_{41}, k_{42}, k_{43}]$ son matrices definidas positivas de ganancias del controlador.

Ahora diseñamos el controlador para el subsistema (4.13), para esto diseñamos un controlador por cada variable de estado, es decir, $[U_x, U_y, U_z]^T$. Para controlar el eje z se propone el siguiente control.

$$U_z = F_{prop} = \left(\frac{m}{h_3}\right)(g + \upsilon_3) \tag{4.17}$$

Como se puede notar en la ecuación (4.13) es posible controlar el eje z con F_{prop} directamente debido a la relación con T_{prop}, es decir, no tenemos que encontrar los ángulos Euler deseados. Sin embargo, no es posible controlar directamente $(x - y)$ con F_{prop} para lograr un desplazamiento en estos ejes es necesario cambiar los ángulos de Euler con el controlador de orientación, de manera específica, se debe cambiar los ángulos $(\phi - \theta)$. Antes de manipular los ángulos de Euler, debemos de diseñar controladores capaces de manipular la posición de los ejes x y y, como no es posible controlar estos ejes directamente, entonces se diseñan controles virtuales U_x y U_y.

$$U_x = h_1 = \left(\frac{m}{U_z}\right)\upsilon_1 \tag{4.18}$$

$$U_y = h_2 = \left(\frac{m}{U_z}\right)\upsilon_2 \tag{4.19}$$

De las ecuaciones anteriores, U_z tiende a cero si el cuadricóptero está en el suelo. Por lo tanto, U_x y U_y no se indeterminan.

El siguiente paso es encontrar la relación algebraica entre los controladores virtuales y los ángulos deseados de Euler. De (4.13) sin considerar el efecto de la perturbación y manteniendo la variable ψ_d libre, se obtienen las siguientes expresiones:

$$tan\theta_d = \frac{\ddot{x}cos\psi_d + \ddot{y}sin\psi_d}{\ddot{z} + g} \tag{4.20}$$

$$sin\phi_d = \frac{m(\ddot{x}sin\psi_d + \ddot{y}cos\psi_d)}{\sqrt{\ddot{x}^2 + \ddot{y}^2 + (\ddot{z} + g)^2}} \tag{4.21}$$

Prueba de estabilidad del algoritmo Super-Twisting Generalizado.

Se realiza la estabilidad de estabilidad del algoritmo Super-Twisting generalizado basado en el trabajo de Derafa et al. [27]. De la ecuación (4.8) y (4.14), las dinámicas de las superficies de deslizamiento se pueden escribir de la siguiente forma:

$$\dot{\sigma} = w(t) - K_1 \left[|\sigma|^{\frac{1}{2}} \, sgn(\sigma) + \sigma \right] - K_2 \int_0^t \left[\frac{1}{2} sgn(\tau) + \frac{3}{2} |\tau|^{\frac{1}{2}} \, sgn(\tau) + \tau \right] d\tau \tag{4.22}$$

$$\dot{\varsigma} = \varpi(t) - K_3 \left[|\varsigma|^{\frac{1}{2}} \, sgn(\varsigma) + \varsigma \right] - K_4 \int_0^t \left[\frac{1}{2} sgn(\tau) + \frac{3}{2} |\tau|^{\frac{1}{2}} \, sgn(\tau) + \tau \right] d\tau \tag{4.23}$$

Si definimos las siguientes variables:

$$\begin{aligned}
\mu_{1i} &= \sigma_i \\
\mu_{2i} &= -k_{2i} \int_0^t \left[\frac{1}{2} sgn(\tau) + \frac{3}{2} |\tau|^{\frac{1}{2}} \, sgn(\tau) + \tau \right] d\tau \\
\dot{w}(t) &= \beta_i(t) \\
\mu_{3j} &= \varsigma_j \\
\mu_{4j} &= -k_{4j} \int_0^t \left[\frac{1}{2} sgn(\tau) + \frac{3}{2} |\tau|^{\frac{1}{2}} \, sgn(\tau) + \tau \right] d\tau \\
\dot{\varpi}(t) &= \gamma_j(t)
\end{aligned}$$

Entonces (4.22) y (4.23) se puede rescribir en su forma escalar como $(i, j = \overline{1,3})$ como:

$$\begin{aligned}
\dot{\mu}_{1i} &= -k_{1i} \left[|\mu_{1i}|^{\frac{1}{2}} \, sgn(\mu_{1i}) + \mu_{1i} \right] + \mu_{2i} \\
\dot{\mu}_{2i} &= -k_{2i} \left[\frac{1}{2} sgn(\mu_{1i}) + \frac{3}{2} |\mu_{1i}|^{\frac{1}{2}} \, sgn(\mu_{1i}) + \mu_{1i} \right] + \beta_i(t) \\
\dot{\mu}_{3j} &= -k_{3j} \left[|\mu_{3j}|^{\frac{1}{2}} \, sgn(\mu_{3j}) + \mu_{3j} \right] + \mu_{4j} \\
\dot{\mu}_{4j} &= -k_{4j} \left[\frac{1}{2} sgn(\mu_{3j}) + \frac{3}{2} |\mu_{3j}|^{\frac{1}{2}} \, sgn(\mu_{3j}) + \mu_{3j} \right] + \gamma_j(t)
\end{aligned}$$

Sin pérdida de generalidad, nosotros podemos probar la estabilidad de un subsistema de la ecuación anterior y simplificando la notación, tenemos

$$\begin{aligned}
\dot{\mu}_1 &= -\bar{k}_1 \left[|\mu_1|^{\frac{1}{2}} \, sgn(\mu_1) + \mu_1 \right] + \mu_2 \\
\dot{\mu}_2 &= -\bar{k}_2 \left[\frac{1}{2} sgn(\mu_1) + \frac{3}{2} |\mu_1|^{\frac{1}{2}} \, sgn(\mu_1) + \mu_1 \right] + \beta(t)
\end{aligned} \tag{4.24}$$

La función candidata de Lyaponov se define como:

$$V(\mu) = \frac{1}{2}\rho_2 \left(\bar{k}_1 |\mu_1|^{\frac{1}{2}} sgn(\mu_1) + \bar{k}_1\mu_1 - \mu_2^2\right)^2 + \frac{1}{2}\rho_1\mu_2^2 + (\rho_1+\rho_2)\bar{k}_2 \left[|\mu_1| + \frac{1}{2}|\mu_1|^2 + |\mu_1|^{\frac{3}{2}}\right] \tag{4.25}$$

La función candidata de Lyapunov tiene la característica de ser una función continua, definida positiva y diferenciable excepto en $\mu_1 = 0$. Esta ecuación se puede expresar de forma cuadrática $V(\mu) = \xi^T P \xi$ donde $\xi^T = \left[|\mu_1|^{\frac{1}{2}} sgn(\mu_1), \mu_1, \mu_2\right]$, entonces

$$P = \frac{1}{2}\begin{bmatrix} \rho_2\bar{k}_1^2 + 2L\bar{k}_2 & \rho_2\bar{k}_1^2 + L\bar{k}_2 & -\rho_2\bar{k}_1 \\ \rho_2\bar{k}_1^2 + L\bar{k}_2 & \rho_2\bar{k}_1^2 + L\bar{k}_2 & -\rho_2\bar{k}_1 \\ -\rho_2\bar{k}_1 & -\rho_2\bar{k}_1 & L \end{bmatrix}$$

Donde $L = \rho_1 + \rho_2$. Los parámetros ρ_1, ρ_2, $\bar{k}_1$ y $\bar{k}_2$ son escalares y si son positivos entonces $V(\mu)$ es una matriz definida positiva. Por lo tanto, se satisface la siguiente desigualdad.

$$\lambda_{min}(P)\|\xi\|_2^2 \leq V(\mu) \leq \lambda_{max}(P)\|\xi\|_2^2 \tag{4.26}$$

Donde $\|\xi\|_2^2 = |\mu_1| + \mu_1^2 + \mu_2^2$ es la norma Euclidiana de ξ. Entonces la derivada de la función de Lyapunov con respecto al tiempo a lo largo de la trayectoria del sistema, está dada como:

$$\dot{V} \leq -\xi^T [Q_1 + Q_2]\xi - \frac{1}{|\mu_1|^{\frac{1}{2}}}\xi^T [R_1 + R_2]\xi + \frac{1}{|\mu_1|^{\frac{1}{2}}}\nu_1^T\xi \tag{4.27}$$

Con

$$\begin{aligned}
Q_1 &= \bar{k}_1 \begin{bmatrix} 2\rho_2\bar{k}_1^2 & 0 & 0 \\ 0 & \rho_2\bar{k}_1^2 & -\rho_2\bar{k}_1 \\ 0 & -\rho_2\bar{k}_1 & \frac{1}{2}\rho_2 \end{bmatrix} \\
Q_2 &= \bar{k}_1 \begin{bmatrix} \bar{k}_2(\frac{5}{2}\rho_1 + \frac{1}{2}\rho_2) & 0 & 0 \\ 0 & \bar{k}_2(2\rho_1+\rho_2) & -\frac{1}{2}L\frac{\bar{k}_2}{\bar{k}_1} \\ 0 & -\frac{1}{2}L\frac{\bar{k}_2}{\bar{k}_1} & \frac{1}{2}\rho_2 \end{bmatrix} \\
R_1 &= \bar{k}_1 \begin{bmatrix} \rho_1\bar{k}_1^2 & 0 & 0 \\ 0 & 5\rho_2\bar{k}_1^2 & -3\rho_2\bar{k}_1 \\ 0 & -3\rho_2\bar{k}_1 & \frac{1}{2}\rho_2 \end{bmatrix} \\
R_2 &= \bar{k}_1 \begin{bmatrix} \bar{k}_2(2\rho_1+\rho_2) & 0 & -\frac{1}{2}L\frac{\bar{k}_2}{\bar{k}_1} \\ 0 & \bar{k}_2(7\rho_1+2\rho_2) & 0 \\ -\frac{1}{2}L\frac{\bar{k}_2}{\bar{k}_1} & 0 & \frac{1}{2}\rho_2 \end{bmatrix} \\
\nu_1^T &= \delta \begin{bmatrix} \rho_2\bar{k}_1 & \rho_2\bar{k}_1 & L \end{bmatrix}
\end{aligned}$$

Suponga que el término de la perturbación 4.24 es globalmente acotado como se menciona en la restricción 5. Entonces, La derivada de Lyapunov con respecto al tiempo se reduce a:

$$\dot{V}(\mu) \leq -\xi^T Q\xi - \frac{1}{|\mu_1|^{\frac{1}{2}}}\xi^T R\xi$$

Con $Q = Q_1 + Q_2$ y $R = R_1 + R_2 - \delta R_3$. Y la matriz R_3 es de la forma:

$$R_3 = \begin{bmatrix} \rho_2\bar{k}_1 & \frac{1}{2}\rho_2\bar{k}_1 & \frac{1}{2}L \\ \frac{1}{2}\rho_2\bar{k}_1 & 0 & 0 \\ \frac{1}{2}L & 0 & 0 \end{bmatrix}$$

Si los parámetros $\rho_1 > 0$, $\rho_2 > 0$, las ganancias $\bar{k}_1 > 0$, $\bar{k}_2 > \delta$ y suficientemente grandes, y las matrices Q y R son definidas positivas. Entonces, se obtiene que

$$\dot{V}(\mu) \leq -\frac{1}{|\mu_1|^{\frac{1}{2}}}\xi^T R\xi \leq -\frac{1}{|\mu_1|^{\frac{1}{2}}}\xi^T \lambda_{min}(R)\,\|\xi\|_2^2 \tag{4.28}$$

Usando (4.26) y (4.28), obtenemos que

$$|\mu_1|^{\frac{1}{2}} \leq \|\xi\|_2 \geq \frac{V^{\frac{1}{2}}(\mu)}{\lambda_{min}^{\frac{1}{2}}(P)}$$

Finalmente, tenemos que la derivada de la función de Lyaponov como

$$\dot{V}(\mu) \leq -\epsilon V^{\frac{1}{2}}(s)$$

Con $\epsilon = \frac{\lambda_{min}^{\frac{1}{2}}(P)\lambda_{min}(\widetilde{Q}_2)}{\lambda_{max}(\widetilde{Q}_2)}$.

Ahora, resolviendo la ecuación diferencial

$$\dot{v} = -\epsilon v^{\frac{1}{2}}, \qquad v(0) = v_0 \geq 0$$

Tenemos la solución

$$v(t) = \left(v_0^{\frac{1}{2}} - \frac{\epsilon}{2}t\right)^2 \tag{4.29}$$

Si usamos el principio de comparación, obtenemos que la solución converge en tiempo finito al origen con $T = 2V^{\frac{1}{2}}(\mu_0)/\epsilon$ unidades de tiempo. Es decir, $\mu_i = 0$ en tiempo finito donde implica que $\lim_{t\to\infty} e = 0$ y $\lim_{t\to\infty} \dot{e} = 0$. Y por lo tanto, $\lim_{t\to\infty} r = 0$ y $\lim_{t\to\infty} \dot{r} = 0$.

4.3. Resultados del controlador.

Ahora presentamos los parámetros utilizados para el controlador de seguimiento de trayectorias robusta ante perturbaciones. Los parámetros del sistema son: $m = 0{,}468kg$, $I_x = I_y = 3{,}83 \times 10^{-3} kg \cdot m^2$, $I_z = 7{,}13 \times 10^{-3} kg \cdot m^2$ y $g = 9{,}81 m/s^2$. Los parámetros del controlador de orientación tienen los siguiente valores: $\Gamma = diag[2, 2, 2]$, $K_1 = 4\sqrt{W}$, $K_2 = 2W$, donde $W = diag[12, 12, 8]$. Mientras que los parámetros del controlador de posición son: $\Delta = \Gamma$, $K_3 = 8\sqrt{W}$, $K_4 = 4W$.

Una función espiral (4.30) de tres dimensiones se propone como la trayectoria deseada que el controlador de seguimiento de trayectorias debe seguir, está función se define como:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} sin(2t) \\ cos(2t) - 1 \\ t \end{bmatrix} \tag{4.30}$$

Las perturbaciones externas tienen la siguiente forma:

$$A_F(U) = 7H(x) \tag{4.31}$$

Con

$$H(x) = \begin{cases} 0 & \text{if, } x < 0 \\ 1 & \text{if, } x \geq 0 \end{cases}$$

Las perturbaciones externas se introducen en un intervalo de tiempo de 3 a 4 segundos.

En la figura (4.3) presentamos los resultados de seguimiento de trayectorias de un cuadricóptero utilizando la teoría de modos deslizantes de segundo orden.

Los estados del sistema se muestran en la figura (4.4) con condiciones iniciales $[x(0), y0), z(0)]^T$ donde es posible observar que las variaciones de los estados deseados con los estados obtenidos por el controlador son mínimas a pesar de las perturbaciones que se inyectan en el sistema.

Como se ha mencionado, los controladores virtuales nos proporcionan ángulos deseas que son los estados deseados de la orientación del cuadricóptero su evolución con el controlador super-twisting generalizado se muestra en la figura (4.5).

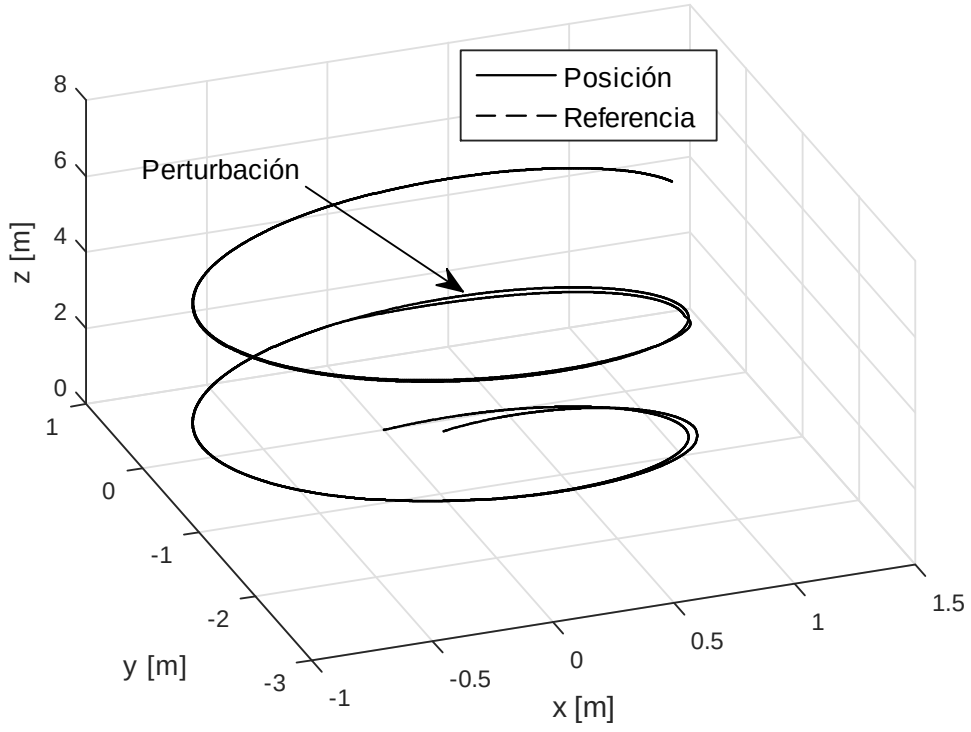

Figura 4.3: Controlador Super-Twisting Generalizado de seguimiento de trayectorias para un UAV.

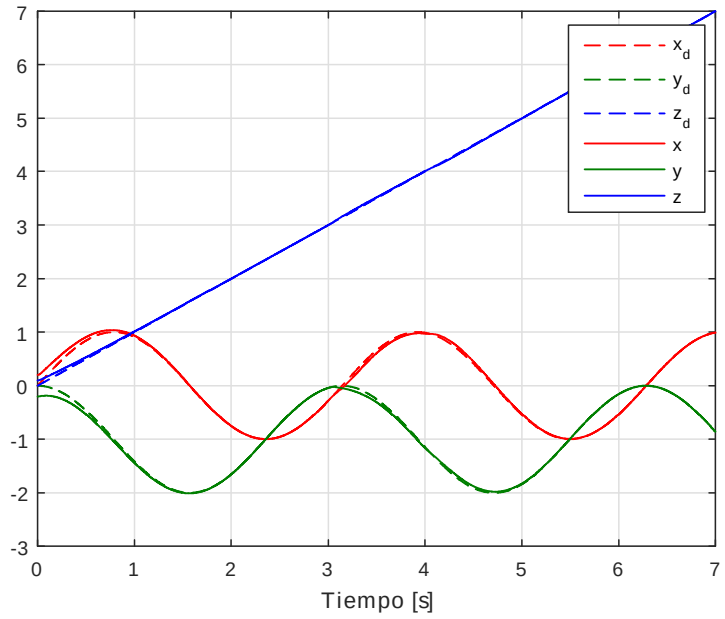

Figura 4.4: Estados de seguimiento de trayectoria del cuadricóptero.

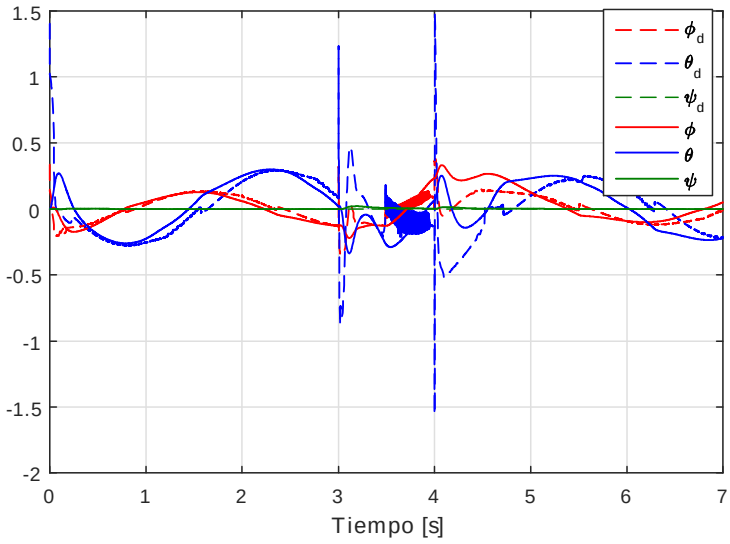

Figura 4.5: Los estados de orientación del sistema cuadricóptero.

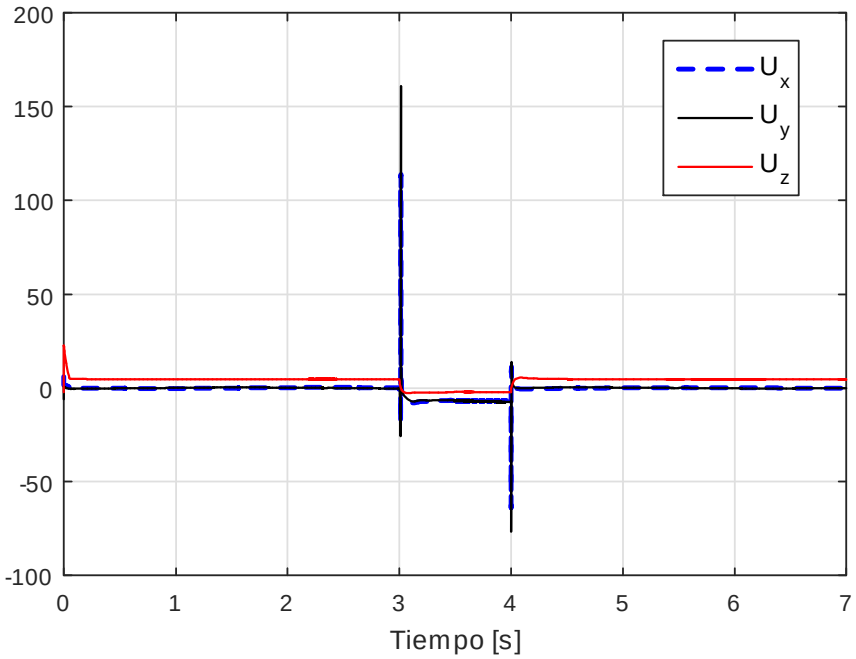

Figura 4.6: Señales de control de seguimiento de trayectoria para el cuadricóptero.

En las figuras (4.6) y (4.7) nos muestran las señales de control, el de seguimiento de trayectoria y de orientación, respectivamente. Las señales del control de seguimiento de trayectorias son virtuales en el sentido que no están en función directa con el torque de los motores del cuadricóptero. En consecuencia, se obtienen los ángulos deseados de Euler donde se requiere de aplicar el controlador de orientación para actuar sobre estos estados debido a que estas señales de control si están en función al torque de los motores que de esta manera se obtiene el desplazamiento de la trayectoria deseada o de referencia.

En la figura (4.8) nos muestra el error de seguimiento de trayectoria. Podemos notar que la amplitud del error es menor a 0.1m, por lo tanto, mostramos la robustez del sistema, además converge en tiempo finito.

Esta parte es el final de este trabajo para cumplir con los objetivos planteados donde los objetivos presentados pueden resumir en resolver el problema de navegación autónoma. En este capítulo, se presentó un controlador de seguimiento de trayectorias con base en la teoría de modos deslizantes de orden superior requerida en la última etapa para lograr la navegación autónoma de un robot. Dicho controlador de seguimiento es robusto ante perturbaciones acotadas y se pueden aplicar a diversos sistemas no lineales. De acuerdo a los resultados presentados en este capítulo el efecto chaterring son atenuadas debido a

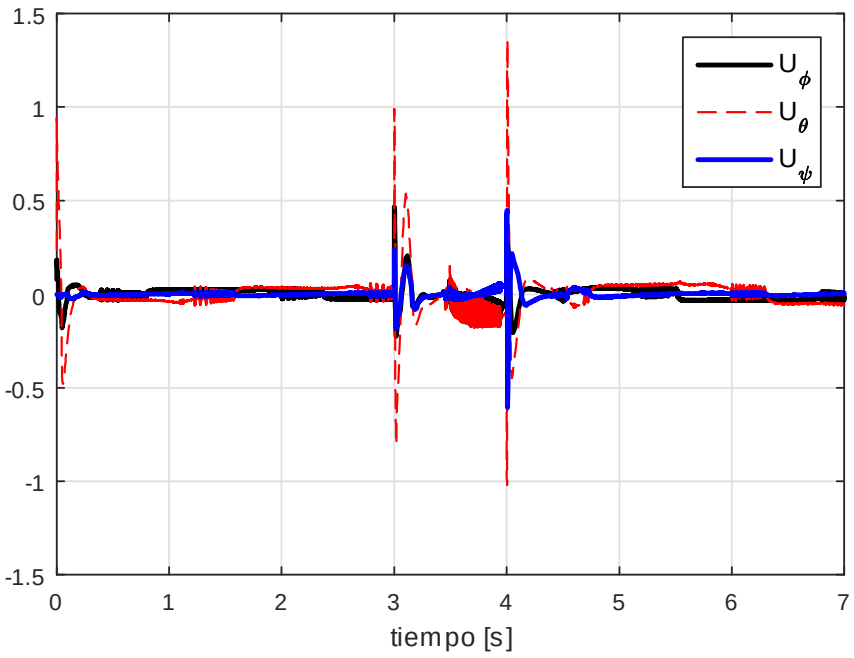

Figura 4.7: Señales de control de orientación para el cuadricóptero.

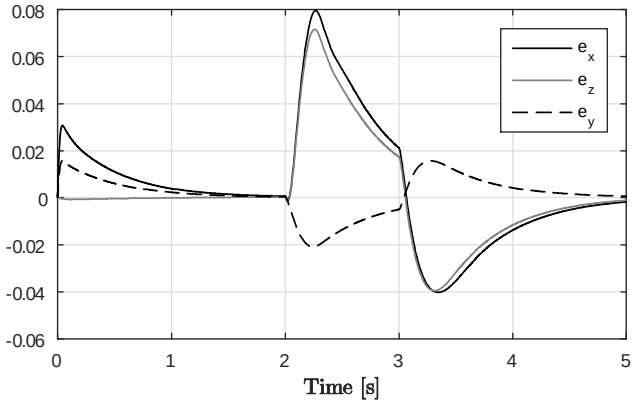

Figura 4.8: Error de seguimiento de trayectorias para el UAV.

que las señales de control no son drásticos, es decir, las señales de control no son prendido o apagado sino puede genera un intervalo de valores.

De este modo, concluimos con los objetivos y se procede a presentar las conclusiones generales de esta tesis.

Capítulo 5

Conclusiones

Este trabajo se dividió en 4 capítulos principales para presentar una nueva técnica para la navegación autónoma de sistemas no lineales. Para lograr la navegación autónoma de dichos sistemas el problema de divide en 4 etapas principales:

- La primera etapa consiste en modelar el sistema de observación que permite cuantificar las características deseadas del entorno como las marcas fijas (Landmarks).
- En la segunda etapa, se diseña un algoritmo para estimar la localización del sistema y de manera simultánea construir un mapa, a este problema se conoce como SLAM, donde se propuso modificar el Filtro de Kalman Extendido aplicando una inyección discontinua toma de la teoría de modos deslizantes para suavizar las restricciones que existe en un algoritmo EKF SLAM, dicha restricción consiste en que los ruidos de entrada deben tener una distribución gaussiana que en la realidad esta restricción es difícil de cumplir, en los resultados obtenidos se prueba que los errores en la estimación de los estados de un robot móvil fueron atenuados aplicando la teoría de modos deslizantes además aprovechando las características de esta teoría se obtiene que los ruidos o perturbaciones sólo requieren ser acotadas, en consecuencia, se realizó la prueba de estabilidad para probar la convergencia del algoritmo que propuso llamada Sliding Mode SLAM (SM SLAM). El algoritmo SM SLAM propuesto se implementa en un sistema más complejo que es el caso de un vehículo aéreo no tripulado (UAV) de manera específica en un cuadricóptero, donde las dimensiones del algoritmo SLAM incrementan considerablemente ya que incrementa los estados de estimación al igual que la dimensión del entorno, sin embargo, los resultados que se obtuvieron fueron favorables para el algoritmo SM SLAM.

- En la tercera etapa, se plantea resolver la planeación de trayectorias el cual es esencial en la navegación autónoma para alcanzar un punto objetivo y evitar los obstáculos durante el proceso. En el cual se propuso dos algoritmos de planeación de trayectorias: el primer algoritmo consistió en utilizar un histograma polar que presento desempeños favorables con respeto a los algoritmos de ocupación de rejilla, sin embargo, en la navegación dentro de los entornos desconocidos se obtienen soluciones locales cuando la densidad de obstáculos supera 0,6, es decir, si $d_{obs} > 0{,}6$ el algoritmo de planeación de trayectorias con histograma polar obtiene soluciones locales. Para resolver este problema, se plantea un segundo algoritmo llamada Roadmap algoritmos genéticos que consiste en modelar en entorno observado con el método Roadmap y posteriormente plantear el problema de planeación de trayectorias como un problema de optimización, el cual, utilizamos los algoritmos genéticos para encontrar soluciones globales que es una característica de este metodología. Además, se realiza la prueba de convergencia planteando como una cadena de Markov finito debido a que el algoritmo genético es elitista, es decir, que sólo sobreviven los cromosomas con mejor aptitud y se logra probar la convergencia exponencial.

- En la última etapa, consiste en diseñar un controlador de movimiento que permite realizar el seguimiento de trayectorias obtenidas por la planeación de trayectorias. En este trabajo se presenta un controlador por modos deslizantes de orden superior o también llamada Super-Twisting Generalizado donde tienen la ventaja de ser robustos ante las perturbaciones externas acotada. De igual manera, se realiza la prueba de estabilidad del dicho controlador. Esta etapa es esencial en la navegación ya que se encarga en realizar la tarea de la navegación dentro del entorno que permita controlar todas las dinámicas que se presentan, así como las perturbaciones o errores del modelado matemático en los sistemas.

El desempeño de la navegación autónoma de cualquier sistema, depende del desempeño de cada una de las etapas de la navegación. En este trabajo cada una de las etapas presentadas con algoritmos nuevos se realizó la prueba de convergencia y se encuentro bajo que restricciones estos algoritmos pueden desempeñarse.

Trabajos a futuro.

Los problemas de navegación autónoma son múltiples.

- El algoritmo SM SLAM incrementa la capacidad computacional conforme la navegación incrementa. Este problema puede ser una limitante debido a la capacidad computacional que restringe el algoritmo en navegaciones con grandes trayectorias, por lo tanto, se debe resolver este problema.
- Implementar experimentalmente los algoritmos presentados en un sistema como el UAV.

Bibliografía

[1] S. Ahn, J. Choi, N. Doh and W. Chung, .A practical approach for EKF-SLAM in an indoor environment: fusing ultrasonic sensors and stereo camera", Autonomous Robots, vol. 24, no. 3, pp. 315-335, 2008.

[2] M. D. S. Arantes, J. D. S. Arantes, C. F. M. Toledo and B. C. Williams, .A hybrid multi-population genetic algorithm for uav path planning", In Proceedings of the Genetic and Evolutionary Computation Conference 2016 (pp. 853-860), 2016.

[3] T. Arora, Y. Gigras and V. Arora, Robotic path planning using genetic algorithm in dynamic environment", International Journal of Computer Applications, 89(11), 9-12, 2017.

[4] M. Ataei and A.Yousefi-Koma, "Three-dimensional optimal path planning for waypoint guidance of an autonomous underwater vehicle". Robotics and Autonomous Systems, 67, 23-32, 2015.

[5] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (SLAM): part II", IEEE Robotics & Automation Magazine, vol. 13, no. 3, pp. 108-117, 2006.

[6] M. Begum, G. Mann and R. Gosine, "Integrated fuzzy logic and genetic algorithmic approach for simultaneous localization and mapping of mobile robots", Applied Soft Computing, vol. 8, no. 1, pp. 150-165, 2008.

[7] J. Borenstein and Y. Koren, "The vector field histogram- fast obstacle avoidance for mobile robots", IEEE Transactions on Robotics and automation, vol. 7, no. 3, pp 278-288, 1991.

[8] L. Besnard, Y. B. Shtessel, and B. Landrum, "Quadrotor vehicle control via sliding mode controller driven by sliding mode disturbance observer", Journal of the Franklin Institute, vol. 349, no. 2, pp. 658–684, 2012.

[9] D. Bhandari, C. A. Murthy and S. K. Pal, "Genetic algorithm with elitist model and its convergence", International Journal of Pattern Recognition and Artificial Intelligence, 10(06), 731-747, 1996.

[10] H. Bouadi, M. Bouchoucha, and M. Tadjine, "Sliding mode control based on backstepping approach for an uav type-quadrotor", World Academy of Science, Engineering and Technology, vol. 26, no. 5, pp. 22– 27, 2007.

[11] K. Bipin, V. Duggal, and K. M. Krishna, .Autonomous navigation of generic monocular quadcopter in natural environment". In Robotics and Automation (ICRA), 2015 IEEE International Conference on. IEEE, p. 1063-1070. 2015.

[12] S. Bouabdallah and R. Siegwart, "Backstepping and sliding-mode techniques applied to an indoor micro quadrotor", in Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on, pp. 2247–2252, IEEE, 2005.

[13] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid and J. Leonard, "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age", IEEE Transactions on Robotics, vol. 32, no. 6, pp. 1309-1332, 2016.

[14] A. Chatterjee and F. Matsuno, .A Neuro-Fuzzy Assisted Extended Kalman Filter-Based Approach for Simultaneous Localization and Mapping (SLAM) Problems", IEEE Transactions on Fuzzy Systems, vol. 15, no. 5, pp. 984-997, 2007.

[15] M. A. Contreras-Cruz, V. Ayala-Ramirez and U. H. Hernandez-Belmonte, "Mobile robot path planning using artificial bee colony and evolutionary programming", Applied Soft Computing, vol. 30, pp. 319-328, 2015.

[16] J. Clemens, T. Reineking and T. Kluth, .An evidential approach to SLAM, path planning, and active exploration", International Journal of Approximate Reasoning, 73, 1-26, 2016.

[17] G. Cai, B. M. Chen, K. Peng, M. Dong and T. H. Lee, "Modeling and control of the yaw channel of a uav helicopter", Industrial Electronics, IEEE Transactions on, vol. 55, no. 9, pp. 3426–3434, 2008.

[18] C. K. Chui, and G. Chen. "Kalman filtering". Springer International Publishing, 2017.

[19] S. Chen, "Kalman Filter for Robot Vision: A Survey", IEEE Transactions on Industrial Electronics, vol. 59, no. 11, pp. 4409-4420, 2012.

[20] C. C. E. Chewu, and V. M. Kumar,. .Autonomous navigation of a mobile robot in dynamic indoor environments using SLAM and reinforcement learning". In IOP Conference Series: Materials Science and Engineering (Vol. 402, No. 1, p. 012022). IOP Publishing, 2018.

[21] K. Choi and S. Lee, .Enhanced SLAM for a mobile robot using extended Kalman Filter and neural networks", International Journal of Precision Engineering and Manufacturing, vol. 11, no. 2, pp. 255-264, 2010.

[22] H. Choset and K. Nagatani, "Topological simultaneous localization and mapping (SLAM): toward exact localization without explicit localization", IEEE Transactions on Robotics and Automation, vol. 17, no. 2, pp. 125-137, 2001.

[23] C. Doer, G. Scholz, and G. F. Trommer, "Indoor laser-based SLAM for micro aerial vehicles". Gyroscopy and Navigation, vol. 8, no 3, p. 181-189, 2017.

[24] A. Davison, I. Reid, N. Molton and O. Stasse, "MonoSLAM: Real-Time Single Camera SLAM", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 29, no. 6, pp. 1052-1067, 2007.

[25] A. Davison and D. Murray, "Simultaneous localization and map-building using active vision", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 7, pp. 865-880, 2002.

[26] M. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte and M. Csorba, .A solution to the simultaneous localization and map building (SLAM) problem", IEEE Transactions on Robotics and Automation, vol. 17, no. 3, pp. 229-241, 2001.

[27] L. Derafa, A. Benallegue and L. Fridman, "Super twisting control algorithm for the attitude tracking of a four rotors uav", Journal of the Franklin Institute, vol. 349, no. 2, pp. 685–699, 2012.

[28] B. A. Erol, S. Vaishnav, J. D. Labrado, P. Benavidez, and M. Jamshidi, Çloud-based Control and vSLAM through cooperative Mapping and Localization". In World Automation Congress (WAC) IEEE, 2016. p. 1-6, 2016.

[29] C. Estrada, J. Neira and J. Tardos, "Hierarchical SLAM: real-time accurate mapping of large environments", IEEE Transactions on Robotics, vol. 21, no. 4, pp. 588-596, 2005.

[30] S. A. Gautam and N. Verma, "Path planning for unmanned aerial vehicle based on genetic algorithm & artificial neural network in 3D". In 2014 International Conference on Data Mining and Intelligent Computing (ICDMIC) (pp. 1-5). IEEE, 2015.

[31] A. Gelb, .ªpplied Optimal Estimation", Cambridge, MA: MIT Press, 1984.

[32] J. K. Goyal and K. S. Nagla, .ª new approach of path planning for mobile robots". In 2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI) (pp. 863-867). IEEE, 2014.

[33] J. N. Guerrero-Tavares, S. Ortiz-Santos, and J. L. Ortiz-Solano "Quadrotor robust path tracking without velocity measurements using the generalized super-twisting control". Revista Politécnica, Vol. 13, no 25, p. 115-124, 2017.

[34] G. Giorgio, S. Cyrill and B. Wolfram, "Improved techniques for grid mapping with rao-blackwellized particle filters", IEEE transactions on Robotics, vol. 23, no 1, p. 34-46, 2007.

[35] S. García, M. E. López, R. Barea, L. M. Bergasa, A.Gómez, and E. J. Molinos, "Indoor SLAM for micro aerial vehicles control using monocular camera and sensor fusion". In Autonomous Robot Systems and Competitions (ICARSC), International Conference on. IEEE, 2016. p. 205-210, 2016.

[36] E. Galceran and M. Carreras, .ª survey on coverage path planning for robotics", Robotics and Autonomous Systems, vol. 61, no 12, p. 1258-1276, 2013.

[37] G. Huang, A. Mourikis and S. Roumeliotis, .ª Quadratic-Complexity Observability-Constrained Unscented Kalman Filter for SLAM", IEEE Transactions on Robotics, vol. 29, no. 5, pp. 1226-1243, 2013.

[38] Z. Hu, S. Xiong, Q. Su and Z. Fang, "Finite Markov chain analysis of classical differential evolution algorithm", Journal of Computational and Applied Mathematics, vol. 268, p. 121-134, 2014.

[39] Q. Hong, X. Ke, and A. Takacs, .An improved genetic algorithm with co-evolutionary strategy for global path planning of multiple mobile robots", Neurocomputing, vol. 120, pp 509-517, 2013.

[40] W. Henry, B. Will N., C, and Dale A., "Learned action slam: Sharing slam through learned path planning information between heterogeneous robotic platforms", Applied Soft Computing, vol. 50, p. 313-326, 2017.

[41] M. Juliá, A. Gil and O. Reinoso, .A comparison of path planning strategies for autonomous exploration and mapping of unknown environments". Autonomous Robots, vol. 33, no 4, p. 427-444, 2012.

[42] C. Kanellakis, and G.Nikolakopoulos, "Survey on computer vision for UAVs: Current developments and trends". Journal of Intelligent & Robotic Systems, 87(1), 141-168, 2017.

[43] R. Kandepu, B. Foss and L. Imsland, .Applying the unscented Kalman filter for nonlinear state estimation", Journal of Process Control, vol. 18, no. 7-8, pp. 753-768, 2008.

[44] http://www.k-team.com/, *K-Team Corporation,* 2013

[45] A. H. Karami and M. Hasanzadeh, .An adaptive genetic algorithm for robot motion planning in 2D complex environments", Computers & Electrical Engineering, 43, 317-329, 2015.

[46] S. Kluge, K. Reif and M. Brokate, "Stochastic Stability of the Extended Kalman Filter With Intermittent Observations", IEEE Transactions on Automatic Control, vol. 55, no. 2, pp. 514-518, 2010.

[47] Y. Kuwata, M. T. Wolf, D. Zarzhitsky, and T. L. Huntsberger, "Safe maritime autonomous navigation with COLREGS, using velocity obstacles", IEEE Journal of Oceanic Engineering, 2014, vol. 39, no 1, p. 110-119, 2014.

[48] B. Ko, H. J. Choi, C. Hong, J. H. Kim, O. C. Kwon, and C. D Yoo. "Neural network-based autonomous navigation for a homecare mobile robot". In 2017 IEEE International Conference on Big Data and Smart Computing (BigComp) (pp. 403-406). IEEE, 2017.

[49] S. Koenig, C. Tovey and Y. Smirnov, "Performance bounds for planning in unknown terrain", Artificial Intelligence, vol. 147, no 1-2, p. 253-279, 2003.

[50] C. Luca, J. Du, M. K. Ng, B. Bona, and M. Indri, .ᴬctive SLAM and exploration with particle filters using Kullback-Leibler divergence", Journal of Intelligent & Robotic Systems, vol. 75, no 2, p. 291-311, 2014.

[51] E. López, S. García, R. Barea, L. M. Bergasa, E. J. Molinos, R. Arroyo, and S. Pardo, .ᴬ Multi-Sensorial Simultaneous Localization and Mapping (SLAM) System for Low-Cost Micro Aerial Vehicles in GPS-Denied Environments", Sensors, vol. 17, no 4, p. 802, 2017.

[52] C. Leung, S. Huang, N. Kwok and G. Dissanayake, "Planning under uncertainty using model predictive control for information gathering", Robotics and Autonomous Systems, vol. 54, no. 11, pp. 898-910, 2006.

[53] L. Fermin-Leon, J. Neira and J. A. Castellanos, "Path planning in graph SLAM using Expected uncertainty", 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), p. 4594-4601, 2016.

[54] S.h. Lee, S. H. Kang, and Y. Kim, "Trajectory tracking control of quadrotor uav", in Control, Automation and Systems (ICCAS), 2011 11th International Conference on, pp. 281–285, IEEE, 2011.

[55] L. Lin and M. A. Goodrich, "Sliding autonomy for UAV path-planning: Adding new dimensions to autonomy management". In Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems (pp. 1615-1624). International Foundation for Autonomous Agents and Multiagent Systems, 2015.

[56] L. Luque-Vega, B. Castillo-Toledo and A. G. Loukianov, Robust block second order sliding mode control for a quadrotor", Journal of the Franklin Institute, vol. 349, no. 2, pp. 719–739, 2012.

[57] T. Madani, and A. Benallegue, "Backstepping sliding mode control applied to a miniature quadrotor flying robot, in IEEE Industrial Electronics", IECON 2006-32nd Annual Conference on, pp. 700–705, IEEE, 2006.

[58] T. Madani and A. Benallegue, "Sliding mode observer and backstepping control for a quadrotor unmanned aerial vehicles", in American Control Conference, 2007. ACC'07, pp. 5887–5892, IEEE, 2007.

[59] D. Mellinger, N. Michael and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors", The International Journal of Robotics Research, 2012.

[60] P. McKerrow, "Modelling the draganflyer four-rotor helicopter", in Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on, vol. 4, pp. 3596–3601, IEEE, 2004.

[61] O. Montiel, U. Orozco-Rosas and R. Sepúlveda, "Path planning for mobile robots using Bacterial Potential Field for avoiding static and dynamic obstacles", Expert System with Applications, vol. 42, pp. 5177-5191, 2015.

[62] J. A. Moreno, "Lyapunov approach for analysis and design of second order sliding mode algorithms", in Sliding Modes after the first decade of the 21st Century, pp. 113–149, Springer, 2011.

[63] J. A. Moreno and M. Osorio, .A lyapunov approach to second-order sliding mode controllers and observers", in Decision and Control, 2008. CDC 2008. 47th IEEE conference on, pp. 2856–2861, IEEE, 2008.

[64] A. Nemra and N. Aouf, Robust airborne 3D visual simultaneous localization and mapping with observability and consistency analysis". Journal of Intelligent and Robotic Systems, vol. 55, no 4-5, p. 345-376, 2009.

[65] http://www.naturalpoint.com/optitrack/, *Natural Point Inc.*, 2013

[66] S. Ön and A. Yazici, .A comparative study of smooth path planning for a mobile robot considering kinematic constraints". In 2011 international symposium on innovations in intelligent systems and applications (pp. 565-569), 2011

[67] A. Pandey, S. Pandey, and D. R. Parhi, "Mobile robot navigation and obstacle avoidance techniques: A review". Int Rob Auto J, 2(3), 00022, 2017.

[68] F. J. Perez-Grau, F. Caballero, L. Merino, and A. Viguria, "Multi-modal mapping and localization of unmanned aerial robots based on ultra-wideband and RGB-D sensing". In Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on. IEEE. p. 3495-3502, 2017.

[69] A. G. Pereira and B. B. de Andrade, .On the genetic algorithm with adaptive mutation rate and selected statistical applications", Computational Statistics, 30(1), 131-150, 2015.

[70] K. Reif and R. Unbehauen, "The extended Kalman filter as an exponential observer for nonlinear systems", IEEE Transactions on Signal Processing, vol. 47, no. 8, pp. 2324-2328, 1999.

[71] K. Reif, S. Gunther, E. Yaz and R. Unbehauen, "Stochastic stability of the discrete-time extended Kalman filter", IEEE Transactions on Automatic Control, vol. 44, no. 4, pp. 714-728, 1999.

[72] G. V. Raffo, M. G. Ortega and F. R. Rubio, "Backstepping/nonlinear h control for path tracking of a quadrotor unmanned aerial vehicle", in American Control Conference, 2008, pp. 3356–3361, IEEE, 2008.

[73] V. Roberge, M. Tarbouchi, and G. Labonté, Çomparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning", IEEE Transactions on Industrial Informatics, vol. 9 no.1, pp. 132-141, 2013.

[74] R. Raja, A. Dutta and K. S. Venkatesh, "New potential field method for rough terrain path planning using genetic algorithm for a 6-wheel rover", Robotics and Autonomous Systems, 72, 295-306, 2015.

[75] Shoudong Huang and Gamini Dissanayake, Çonvergence and Consistency Analysis for Extended Kalman Filter Based SLAM", IEEE Transactions on Robotics, vol. 23, no. 5, pp. 1036-1049, 2007.

[76] B. Song, Z. Wang and L. Sheng, .A new genetic algorithm approach to smooth path planning for mobile robots", Assembly Automation, 36(2), 138-145, 2016.

[77] A. Santamaria-Navarro, G. Loianno, J. Solà, V. Kumar, and J. Andrade-Cetto, .Autonomous navigation of micro aerial vehicles using high-rate and low-cost sensors", Autonomous Robots, 2017, p. 1-18, 2017.

[78] K. D Sebesta, and N. Boizot, .A real-time adaptive high-gain EKF, applied to a quadcopter inertial navigation system". IEEE Transactions on Industrial Electronics, vol. 61, no 1, p. 495-503, 2014.

[79] S. A. Hiremath, G. W. Van Der Heijden, F. K. Van Evert, A. Stein, A, and C. J.Ter Braak, C. J. "Laser range finder model for autonomous navigation of a robot in a maize field using a particle filter", Computers and Electronics in Agriculture, vol. 100, p. 41-50, 2014.

[80] S. A. Sadat, K. Chutskoff, D. Jungic, J. Wawerla and R. Vaughan, "Feature-rich path planning for robust navigation of mavs with mono-slam", 2014 IEEE International Conference on Robotics and Automation (ICRA), p. 3870-3875, 2014.

[81] A. Shum, K. Morris, and A. Khajepour, "Direction-dependent optimal path planning for autonomous vehicles". Robotics and Autonomous Systems, 70, 202-214, 2015.

[82] T. Shioya, K. Kogure, T. Iwata, and N. Ohta,.Autonomous Mobile Robot Navigation Using Scene Matching with Local Features". Journal of Robotics and Mechatronics, 28(6), 887-898, 2016.

[83] M. Sutoh, M. Otsuki, S. Wakabayashi, T. Hoshino, and T. Hashimoto, "The right path: comprehensive path planning for lunar exploration rovers". IEEE Robotics & Automation Magazine, 22(1), 22-33, 2015.

[84] L. Spalazzi and P Traverso, .A dynamic logic for acting, sensing, and planning". Journal of Logic and Computation, 10(6), 787-821, 2000.

[85] H.G. Todoran and M. Bader, .Extended Kalman Filter (EKF)-Based Local SLAM in Dynamic Environments: A Framework", Advances in Robot Design and Intelligent Control, vol. 371, pp. 459-469, 2016.

[86] Tzyh-Jong Tarn and Y. Rasis, .Observers for nonlinear stochastic systems", IEEE Transactions on Automatic Control, vol. 21, no. 4, pp. 441-448, 1976.

[87] A. Tuncer and M. Yildirim, "Dynamic path planning of mobile robots with improved genetic algorithm", Computers & Electrical Engineering, 38(6), 1564-1572, 2012.

[88] C. C. Tsai, H. C. Huang and C. K. Chan, "Parallel elite genetic algorithm and its application to global path planning for autonomous robot navigation", IEEE Transactions on Industrial Electronics, 58(10), 4813-4821, 2011.

[89] P. Thanpattranon, T. Ahamed, and T. Takigawa, "Navigation of autonomous tractor for orchards and plantations using a laser range finder", Automatic control of trailer position with tractor. Biosystems Engineering, vol. 147, p. 90-103, 2016.

[90] V. Utkin, J. Guldner, and J. Shi, "Sliding mode control in electro-mechanical systems". CRC press, 2009.

[91] R. Valencia and J. Andrade-Cetto, "Mapping, planning and exploration with Pose SLAM", Springer, p.60-84, 2018.

[92] C. K. Volos, I. M. Kyprianidis and I. N. Stouboulos, .A chaotic path planning generator for autonomous mobile robots". Robotics and Autonomous Systems, 60(4), 651-656, 2012.

[93] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I", IEEE robotics & automation magazine, vol. 13, no 2, p. 99-110, 2006.

[94] X. Wang, Y. Shi, D. Ding and X. Gu, "Double global optimum genetic algorithm–particle swarm optimization-based welding robot path planning", Engineering Optimization, 48(2), 299-316, 2016.

[95] X. B. Wang, C. Song, G. R. Zhao, and J. L. Huang, .Obstacles avoidance for UAV SLAM based on improved artificial potential field". In Applied Mechanics and Materials. Trans Tech Publications, p. 1118-1121, 2013.

[96] M. Wang, H. Shan, R. Lu, R. Zhang, X. Shen, and F. Bai, Real-time path planning based on hybrid-VANET-enhanced transportation system". IEEE Transactions on Vehicular Technology, 64(5), 1664-1678, 2015.

[97] H. Williams, W. N. Browne and D. A. Carnegie, "Learned action slam: Sharing slam through learned path planning information between heterogeneous robotic platforms", Applied Soft Computing, 50, 313-326, 2017.

[98] N. Wen, L. Zhao, X. Su and P. Ma, ÜAV online path planning algorithm in a low altitude dangerous environment". IEEE/CAA Journal of Automatica Sinica, 2(2), 173-185, 2015.

[99] W. L. Yao, W. Shao, B. Y. Li, and A. L. Chen, .Autonomous navigation of cruise phase based on IST-SCKF", In Control Conference (CCC), 36th Chinese. IEEE, 2017. p. 10414-10417, 2017.

[100] W. Yu, E. Zamora and A. Soria, ."Ellipsoid SLAM: a novel set membership method for simultaneous localization and mapping", Autonomous Robots, vol. 40, no 1, p. 125-137, 2016.

[101] W. Yu, M. A. Moreno, X.Li, Observer Based Neuro Identifier, *IEE Proceedings - Control Theory and Applications*, Vol.147, No.2, 145-152, 2000.

[102] Z. Zuo, "Trajectory tracking control design with command-filtered compensation for a quadrotor", Control Theory & Applications, IET, vol. 4, no. 11, pp. 2343–2355, 2010.

[103] Z. Zuo, ."Adaptive trajectory tracking control design with command filtered compensation for a quadrotor", Journal of Vibration and Control, vol. 19, no. 1, pp. 94–108, 2013.

[104] X. Zhang, Y. Zhao, N. Deng and K. Guo, "Dynamic path planning algorithm for a mobile robot based on visible space and an improved genetic algorithm", International Journal of Advanced Robotic Systems, 13(3), 91, 2016.

[105] Z. Yong, G. Dun-wei and Z. Jian-hua, Robot path planning in uncertain environment using multi-objective particle swarm optimization. Neurocomputing", vol. 103, p. 172-185, 2013.

[106] E. Zamora and W. Yu, "Novel autonomous navigation algorithms in dynamic and unknown environments", Cybernetics and Systems, vol. 47, no 7, p. 523-543, 2016.

Capítulo 6

Apéndice

6.1. Cinemática del robot móvil

En esta sección se muestra el análisis para encontrar la cinemática de un robot móvil de tracción diferencial. Este robot móvil tiene los siguientes parámetros a considerar: L que es la longitud entre las ruedas del vehículo y r es el radio de la rueda (como se muestra en la figura 6.1).

En el modelo de este sistema se considera que el robot móvil se moverá en dos dimensiones como se muestra en la figura 6.2. La localización del robot se determina por las coordenadas (x, y). Por otra parte, v es la velocidad lineal del vehículo con v_l y v_r que son las velocidades tangenciales de cada una de las ruedas.

Las velocidades tangenciales de cada una de las ruedas se pueden calcular por la relación que existe de las velocidades angulares con el radio de cada rueda, como se muestra en la siguiente ecuación.

$$\begin{aligned} V_l &= \gamma_l \cdot r \\ V_r &= \gamma_r \cdot r \end{aligned} \tag{6.1}$$

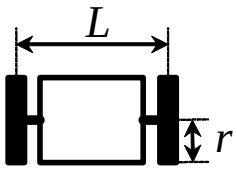

Figura 6.1: Perfil del robot móvil.

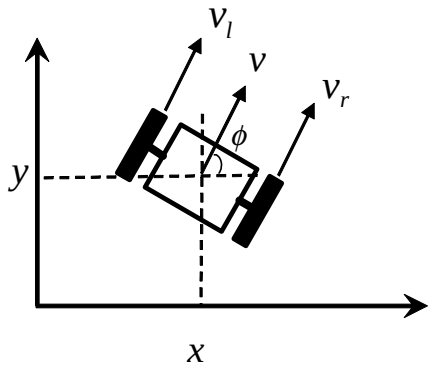

Figura 6.2: Odometría del robot móvil.

Donde γ_l y γ_r son las velocidades angulares de cada una de las ruedas.

Para el análisis de la cinemática nos interesa en conocer la velocidad lineal y la velocidad angular del centro del robot móvil. Estos se obtienen con las siguientes ecuaciones, respectivamente.

$$\begin{aligned} v &= \frac{V_r + V_l}{2} = \frac{(\gamma_l + \gamma_r) \cdot r}{2} \\ \gamma &= \frac{V_r - V_l}{L} = \frac{(\gamma_l - \gamma_r) \cdot r}{L} \end{aligned} \tag{6.2}$$

La ecuación anterior tiene ciertas restricciones tales como; el robot se mueve en una superficie plano, sin deslizamiento y los ejes de las ruedas deben ser perpendiculares a la superficie plana. tomando estas consideraciones el modelo dinámico del robot móvil está dado como:

$$\begin{aligned} \dot{\phi} &= \gamma \\ \dot{x}_r &= v \cos\phi \\ \dot{y}_r &= v \sin\phi \end{aligned} \tag{6.3}$$

Por otra parte, para calcular las velocidades angulares de cada una de las ruedas se realiza mediante la medición con sensores de forma directa o indirectamente. En este caso, la medición se realiza de manera indirecta utilizando sensores ópticos llamados Encoders. Este sensor nos proporciona la cantidad de pulsos que realiza en un intervalo de tiempo asignado. Por lo que es necesario hacer la conversión de pulsos al desplazamiento lineal.

El uso de estos sensores puede traer ciertas limitaciones, debido a que el factor de conversión viene directamente afectado por la resolución del Encoder, de tal forma que la

estimación podría ser de igual manera ser afectado. En la siguiente ecuación se presenta el factor de conversión.

$$\kappa = 2\pi r/nC \tag{6.4}$$

Donde r es el radio de la rueda, n es la reducción del eje/rueda y C es la resolución del Encoder.

Conociendo κ podemos encontrar la velocidad lineal de cada rueda para un intervalo de tiempo, considerando que el sensor toma lectura en intervalos regulares de tiempo, entonces tenemos que:

$$\begin{aligned} V_l &= \kappa N_l \\ V_r &= \kappa N_r \end{aligned} \tag{6.5}$$

Donde N_l y N_r son el número de cuentas del Encoder en el intervalo de tiempo $(k-1, k)$.