Universidad Nacional Autónoma de México

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

Localización de un robot móvil utilizando el filtro de Kalman extendido y marcas visuales en el ambiente.

T E S I S

QUE PARA OPTAR POR EL GRADO DE:

Maestro en Ciencia e Ingeniería de la Computación.

P R E S E N T A :

Diego Alejandro Cordero Castro

TUTOR

Dr. Jésus Savage Carmona

Facultad de Ingeniería

Ciudad Universitaria, Cd. Mx., Enero 2021

Agradecimientos

- A mis padres por darme todo su apoyo, por formarme como una persona de bien, por todo su cariño, consejos y tiempo.
- A mis hermanos Gaby y Mando por apoyarme y motivarme día a día y por toda su alegría que me dan.
- A Luisa <3 por compartir su tiempo conmigo, por todos los buenos momentos, por motivarme cada día, por su cariño y por ayudarme a revisar la tesis.
- A todos los miembros del laboratorio de Biorobotica por sus buenos consejos, por todo el apoyo, por la amistad que me han brindado y por todos los buenos momentos que he pasado con ustedes.
- Al doctor Jesús Savage por orientarme en mi formación profesional y todo el apoyo que me ha brindado.
- Al IIMAS y toda su comunidad por enriquecer mi formación profesional y las buenas experiencias.
- A DGAPA-UNAM por el apoyo recibido a través del proyecto PAPIIT IG100818

Resumen

La localización de un robot, es una herramienta fundamental para la navegación de un robot móvil. Existen métodos de localización basados en diferentes fuentes de información y componentes de hardware, pues no hay un método que funcione adecuadamente en cualquier escenario y para cualquier robot. En el presente trabajo se aborda la localización de un robot móvil utilizando el filtro de Kalman extendido (EKF por sus siglas en inglés) y marcas en el ambiente del robot. También se describen los conceptos necesarios para el desarrollo de esta tesis y se hace énfasis en los métodos de localización para robots móviles. El filtro de Kalman es un algoritmo que permite conocer el estado oculto de sistemas dinámicos. Asimismo, este algoritmo se ha utilizado en la localización de naves espaciales, en economía y en diversas aplicaciones de tecnología. En el presente trabajo, el reconocimiento de marcas se realiza a través de técnicas de visión computacional y en conjunto con el filtro de Kalman, se desarrolla una metodología para la localización de robots móviles. Además, se presenta la implementación del método en un robot móvil real, mencionando las características mas relevantes de implementación. Aunado a ello, se presenta un análisis del método propuesto haciendo una comparación entre los resultados obtenidos y los datos teóricos.

Abstract

The localization of a robot is a fundamental tool for mobile robot navigation . There are localization methods based on different sources of information and hardware components, as there is no method that works properly in any scenario and for any robot. In this work, the location of a mobile robot is approached using the extended Kalman filter (EKF) and landmarks in the robot's environment. The concepts necessary for the development of this thesis are also described and emphasis is placed on location methods for mobile robots. The Kalman filter is an algorithm that allows knowing the hidden state of dynamic systems. The Kalman filter has been used in spaceships localization, economics, and various technology applications. In this document, landmark recognition is carried out through computer vision techniques and the Kalman filter, a methodology is developed for mobile robots localization. In addition, the implementation of the method in a real mobile robot is shown, focusing in the most relevant implementation characteristics. An analysis of the proposed method is also presented, making a comparison between the results obtained and the theoretical data.

Índice general

Capítulo 1

Introducción

La robótica es la ciencia de percibir y manipular el mundo físico a través de dispositivos controlados por computadora [1] . Dichos dispositivos son conocidos como robots, que a su vez están constituidos por componentes como motores y sensores. Cada robot tiene una finalidad diferente, por ejemplo, los brazos robóticos están dotados de varias articulaciones que le permiten manipular objetos con gran precisión, este tipo de robots son utilizados en lineas de producción para soldar piezas metálicas, empacar productos y clasificar componentes, entre otras tareas. Otro ejemplo de robot son los drones, los cuales tienen como objetivo el entretenimiento o grabar una escena con una cámara digital, por lo tanto, dependiendo del objetivo del robot, este contará con diferentes componentes y en consecuencia tendrá capacidades diferentes.

Los robots móviles son aquellos que se desplazan en su entorno, los hay teleoperados o autónomos, como los robots aspiradora o los que se utilizan en muchas clases de robótica para principiantes que usualmente cuentan con dos llantas que le permiten la movilidad y un dispositivo que controla sus movimientos. Los robots teleoperados no requieren de un sistema de control de movimientos, pues el operador es el que controla los movimientos por medio de algún control remoto. Controlar los movimientos de un robot móvil autónomo pude ser una tarea sencilla o muy compleja, dependiendo de la finalidad del robot, es decir, si únicamente el robot va a evadir obstáculos sin importar su destino no se requieren un control de movimientos muy complejo, sin embargo, si un robot móvil autónomo tiene como objetivo recolectar muestras de rocas que se

encuentran a un kilometro de distancia en Marte, donde el suelo es irregular y puede haber muchos obstáculos en su camino, el control de sus movimientos debe ser muy bueno.

Un robot móvil adquiere información de su entorno a través de sus sensores, dicha información se procesa por medio de algoritmos computacionales y da como resultado acciones, las cuales son ejecutadas por los actuadores del robot para poder interactuar con su entorno.

Una cualidad importante en un robot móvil, es la navegación, que es la habilidad de desplazarse por su entorno de manera eficiente. Para que el robot pueda navegar de manera precisa es conveniente tener un mapa del lugar donde el robot interactuara, ya que si no se cuenta con un mapa el problema de llegar a un punto especifico seria una tarea mucho más compleja, incluso dicha tarea sería complicada para una persona que se encuentre en una ciudad donde nunca ha estado y que además no pueda preguntar a otra persona alguna pista para llegar al lugar deseado. Teniendo un mapa se puede planear una ruta antes de iniciar la navegación y una vez planeada, ejecutarla, pero en la ejecución se pueden cometer errores, por ejemplo si a un robot móvil se le dice que avance tres metros, lo más probable es que avance un poco más o un poco menos pero no exactamente lo indicado, esto mismo pasaría al girar, por lo tanto, después de varios movimientos el robot estaría en un lugar muy diferente al que se esperaría. Los errores de movimiento son causados por múltiples factores, por ejemplo para los robots que tienen un par de llantas una fuente de errores es el tipo de suelo, la precisión de sus motores, el peso del robot, el viento, entre otros. En consecuencia se requiere de un método de localización para que el robot llegue a su destino y no se pierda a causa de los errores acumulados por sus movimientos.

Algunos métodos de localización que se utilizan en robots móviles tienen como base un sensor láser 2D que registra la forma de paredes u objetos que se encuentran a su alrededor y con la ayuda de un mapa se logra localizar, sin embargo, si el robot se encuentra en un pasillo largo donde la forma de las paredes no varia mucho, el robot es incapaz de localizarse con éxito o si el entorno cuenta con muchas personas, alteraría las lecturas del sensor láser, ya que no podría ver las paredes, lo cual provocaría fallas

en el método de localización.

En este documento se abordara la investigación y desarrollo de un método de localización por medio de marcas conocidas en el ambiente, por ejemplo en el interior de un edificio de una universidad el robot móvil podría reconocer marcas que no cambien a lo largo del tiempo, por ejemplo, los señalamientos de evacuación en caso de sismo, letreros con la matricula de salones, puertas, ventanas, pizarrones, extintores, señalamientos de sanitarios, entre otros. Este método no pretende ser el mejor de todos, pues también tiene casos en los que no podrá localizarse de manera correcta, sin embargo este estudio servirá como una herramienta en la investigación de métodos de localización en robots móviles.

1.1. Justificación

La interacción de un robot móvil con su entorno está relacionado con su localización. En consecuencia es trascendente estudiar esta característica de la robótica. Si bien es cierto que existen métodos excelentes, no hay uno definitivo que resuelva todas las problemáticas implicadas en la localización. Es por ello, que se retoma el filtro de Kalman extendido y la visión computacional, con la finalidad de otorgar elementos teóricos y operacionales para futuros estudios comparativos entre las diversas metodologías.

1.2. Hipótesis

¿Es posible que un robot móvil pueda estimar su pose en un ambiente conocido por medio del filtro de Kalman extendido, y puntos de referencia? Los cuales son encontradas por medio de técnicas de visión computacional, de tal modo que el robot puede navegar en un ambiente conocido de manera exitosa.

1.3. Objetivo

Desarrollar una metodología para estimar la localización de un robot móvil basada en el filtro de Kalman extendido y utilizando marcas en el entorno como puntos de referencia, las marcas son reconocidas por medio de técnicas de visión computacional.

1.4. Estructura de la tesis

En el capítulo dos se explican las características, capacidades y componentes de un robot móvil, así como los conceptos necesarios para comprender de mejor manera el desarrollo del presente trabajo.

En el capítulo tres se mencionan los métodos probabilísticos de localización, haciendo énfasis en el Filtro de Kalman y el Filtro de Kalman Extendido. Se da una explicación detallada con ejemplos y algoritmos del funcionamiento de el método.

En el capítulo 4 se mencionan técnicas de visión computacional como el procesamiento digital de imágenes que son necesarios para el reconocimiento de marcas para la metodología propuesta.

En el capítulo 5 se describe la metodología que se propone para este trabajo de tesis, basándose en conceptos que se mencionan en capítulos anteriores, además se mencionan detalles de implementación en un robot móvil.

En el capítulo 6 se muestran las pruebas hechas con un robot móvil bajo diferentes situaciones, también se muestran los resultados obtenidos y una breve interpretación.

En el capítulo 7 se muestran las conclusiones obtenidas de la experimentación y también se exponen ideas para un trabajo a futuro.

Capítulo 2

Marco Teórico

En este capítulo se menciona la teoría, conceptos y métodos necesarios para sustentar el presente trabajo de tesis.

2.1. Robots móviles

Los robots móviles son aquellos que se desplazan en su entorno de manera autónoma o teleoperada. Este capítulo se centra en los robots móviles autónomos, los cuales requieren de un sistema de control sofisticado para poder evitar obstáculos, saber en que lugar se encuentra (localización) y así poder moverse de un lugar a otro sin dificultades.

El componente principal de un robot para que pueda moverse es el sistema o mecanismo de locomoción. Ejemplo de ello, sería un par de llantas acopladas a motores o unas piernas robóticas conformadas por servomotores y un chasis.

Otro componente fundamental es el sistema de software que se encarga del control de los movimientos del robot, el cual es un programa que se ejecuta en un microcontrolador o microprocesador. Para que el sistema de control pueda generar movimientos con el mínimo error, se debe de tomar en cuenta el modelo cinemático del sistema de locomoción del robot, para poder lograr un sistema lo mas preciso posible.

Es necesario que un robot móvil autónomo cuente con sensores para conocer su entorno y también para obtener información de él mismo. Los sensores internos son

útiles para conocer el estado interno del robot, por ejemplo, saber en que posición se encuentran sus llantas o conocer la carga de la batería. Los sensores externos sirven para conocer el entorno del robot como posibles obstáculos, personas, sonidos, fuentes de luz, entre otros. A demás de los componentes ya mencionados, los robots móviles cuentan con otros componentes de hardware que son necesarios para su funcionamiento como la batería, reguladores de voltaje, fusibles, microcontroladores, microprocesadores, etc. En conjunto todos los componentes forman y dan funcionamiento al robot móvil y con ayuda de un control preciso puede llegar a ser un robot móvil autónomo.

2.2. Pose de un robot

En este trabajo se utilizará el concepto de 'pose' para referirse a las coordenadas de la posición del robot respecto un eje coordenado. También el concepto 'pose' engloba la orientación del robot. Al tratarse este trabajo de un robot móvil que se desplaza en un plano, el termino pose se refería en la mayoría de los casos a las componentes (x, y, θ), a menos que se indique lo contrario. Las componentes x y y son valores reales que tienen como referencia un punto fijo en el espacio. Por otro lado, la componente θ hace referencia al ángulo en el que se encuentra orientado el robot respecto al eje x del sistema coordenado. El termino posición hace referencia únicamente a las componentes x y y. Mientras que el termino orientación hace referencia únicamente a la componente θ.

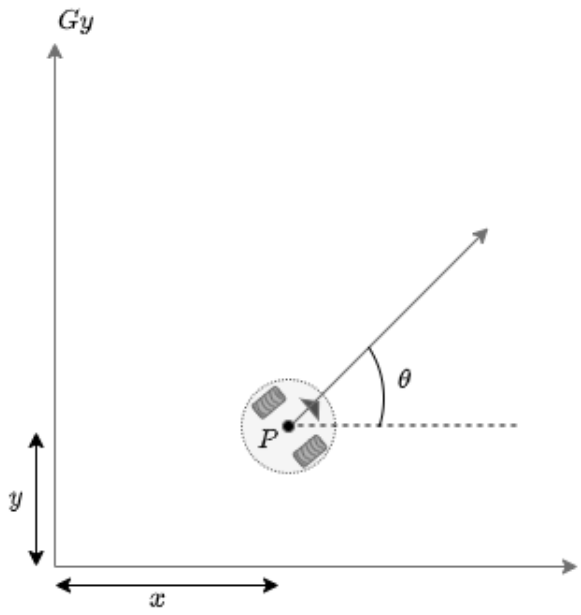


Figura 2.1: Robot móvil diferencial con marco de referencia global G. La pose del robot es: (x,y,θ). La posición del robot es el punto P: (x,y) y la orientación del robot es: θ

2.3. Locomoción

Los robots móviles necesitan de un sistema de locomoción para poder desplazarse, existen robots que pueden caminar, nadar, rodar, saltar, girar, etc. Por ejemplo, los robots humanoides pueden caminar debido a piernas con múltiples servomotores que le permiten equilibrarse gracias a un sistema de control sofisticado. Cabe mencionar que esta tesis se enfocará a los robots móviles que utilizan motores de corriente directa con llantas acopladas para poder desplazarse. Existen diversas configuraciones de locomoción para robots móviles con llantas, en la tabla 2.1 se presentan algunas de ellas.

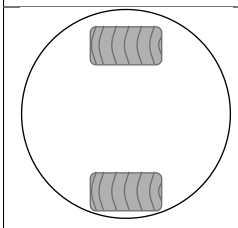
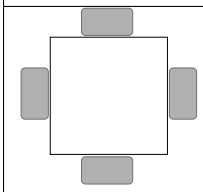
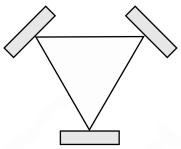
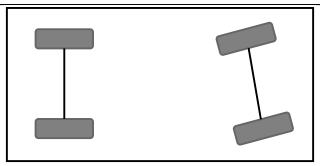
Configuración	Descripción
	Un volante en la parte delantera, una rueda de tracción en la parte trasera
	Robot de par diferencial, dos llantas acoplada a un motor, ambos motores sobre una linea recta
	Robot con cuatro llantas omnidireccionales acopladas a los motores en cada uno de los lados del robot.
	Tres llantas omnidireccionales a 120 grados de separación
	Dos ruedas motorizadas en la parte delantera, 2 ruedas libres en la parte trasera.

Tabla 2.1: Diferentes configuraciones de robots móviles con llantas

Los robots omnidireccionales son aquellos que se pueden desplazar en cualquier dirección sobre un plano sin la necesidad de rotar, este tipo de movimiento se logra a través de llantas omnidireccionales (Fig. 2.2), las cuales son llantas que cuentan con rodillos para poder permitir el movimiento sobre el eje del motor y no únicamente en dirección perpendicular a el, como es el caso de las llantas comúnmente utilizadas en los automóviles.

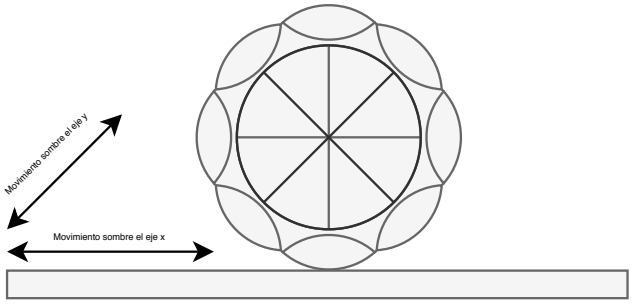


Figura 2.2: Rueda omnidireccional que permite el movimiento dirección x y y sombre un plano.

Los robots móviles de par diferencial, cuentan con dos motores, cada uno con una llanta acoplada, y permiten el movimiento únicamente en dirección perpendicular al eje de su motor. En la figura (Fig. 2.3) se muestra los movimientos que debe hacer un robot de par diferencial para desplazarse de un punto a otro y terminar con la misma orientación que tenia antes de iniciar el desplazamiento. El mismo movimiento es ejemplificado en la figura (Fig. 2.4) para un robot omnidireccional.

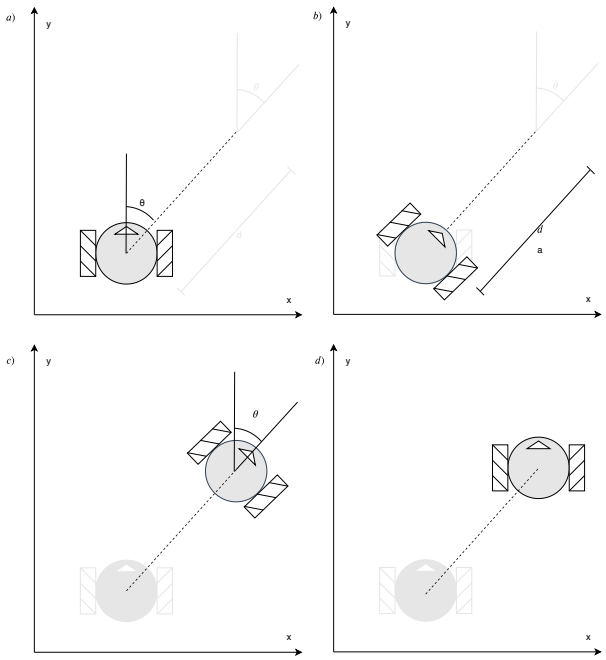


Figura 2.3: A) Pose original del robot. B)Giro sobre el eje del robot . C) Desplazamiento del robot. D)Giro sobre eje del robot para obtener la misma orientación que en un inicio.

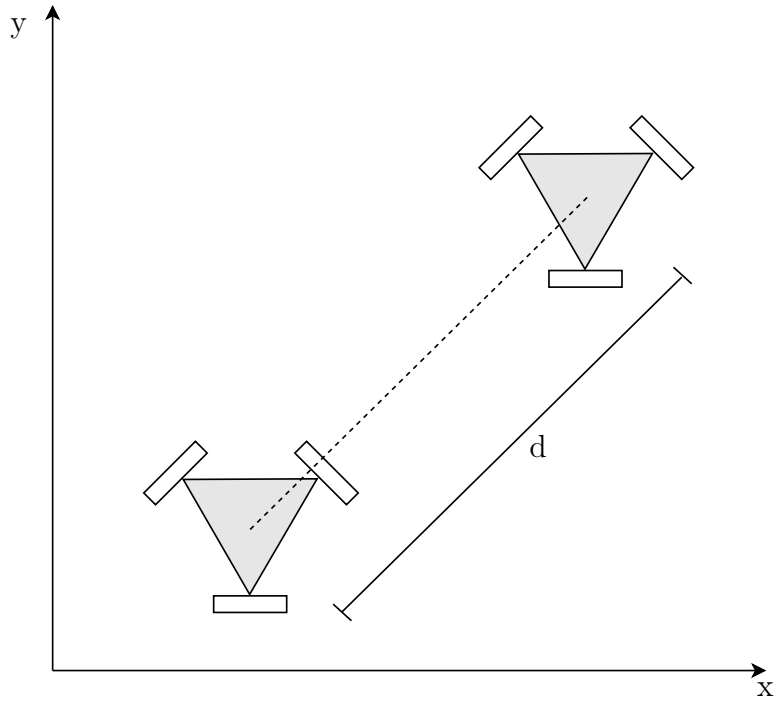


Figura 2.4: Desplazamiento de un robot móvil omnidireccional

Como se puede ver en las figuras (Fig. 2.3) y (Fig. 2.4) es mayor el número de movimientos hechos por un robot de par diferencial en comparación al omnidireccional, sin embargo, el robot omnidireccional requiere mover tres motores lo cual requiere mayor energía, también un control de movimientos mas sofisticado y, por supuesto al requerir mas componentes su costo monetario será mayor. Es importante conocer los sistemas de locomoción, pues éstos influyen en el modelo cinemático del robot, el cual es fundamental para muchos métodos de localización.

2.4. Cinemática de robot móvil

Conocer la cinemática de un robot móvil es fundamental para comprender el comportamiento del sistema mecánico del robot y también es importante conocer la cinemática del robot al momento de desarrollar un sistema de control, ya que dependiendo de su locomoción, el robot móvil tendrá restricciones de movimiento, por ejemplo, un automóvil no puede girar sobre si mismo, incluso no puede dar giros muy cerrados en comparación a un robot omnidireccional, donde la libertad de sus

movimientos es mayor a la de un automóvil. Por lo mencionado es importante conocer la cinemática del robot cuando se esta diseñando uno sistema de localización.

Para conocer la posición de un robot móvil se requiere de un sistema de localización. Como primer recurso se puede capturar sus movimientos a lo largo del tiempo para poder conocer su posición. Desafortunadamente ocurren pequeños errores en los movimientos del robot que originan que al cabo de varios movimientos el robot no se encuentre donde se esperaría.

Un sistema de localización se debe basar en la cinemática del robot teniendo en cuenta los errores que se pueden ocurrir y otros métodos que serán mencionados mas adelante. Es por eso la importancia de la cinemática del robot móvil en el desarrollo de un sistema de localización.

2.4.1. Cinemática de un robot móvil diferencial

Para ejemplificar el proceso de describir la cinemática del robot se propone un robot móvil diferencial como el de la figura (Fig. 2.5) el cual tiene un marco de referencia L y un marco de referencia global G. EL marco de referencia L siempre va con el robot, es decir, es un marco de referencia local cuyo origen, mientras el marco de referencia G es global y θ es el ángulo entre los dos marcos de referencia. Para hacer un seguimiento del robot a través del tiempo es necesario tomar un punto del robot como referencia, dicho punto será el punto P que se encuentra en el eje de rotación del robot.

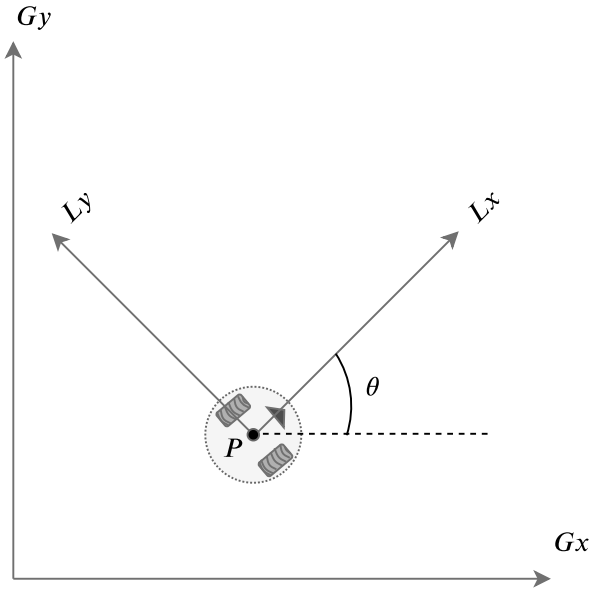


Figura 2.5: Robot móvil diferencial con marco de referencia local L y marco de referencia global G

Para poder describir la posición del robot en el marco de referencia global se utiliza un vector con tres componentes:

$$\vec{\xi}_G = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{2.1}$$

Para describir el movimiento en termino de sus componentes es necesario mapear los movimientos sobre los ejes del marco de referencia global al marco de referencia local, dicho mapeo se puede hacer con la matriz de rotación ortogonal:

$$R(\theta) = \begin{bmatrix} cos(\theta) & sin(\theta) & 0 \\ -sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

De la siguiente manera se obtendrá la relación entre el marco de referencia global al local dadas las velocidad :

$$\dot{\vec{\xi}}_R = R(\theta)\dot{\vec{\xi}}_G \tag{2.3}$$

A manera de ejemplo se revisará cuando el ángulo θ es igual a $\pi/2$ como se muestra en la figura [2.6].

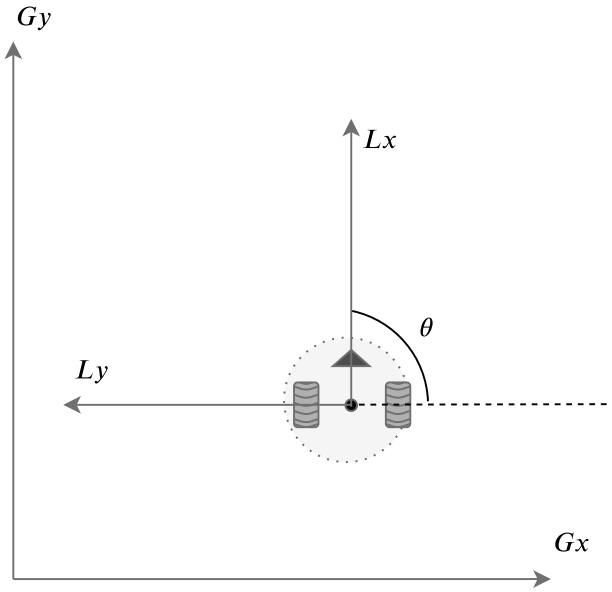


Figura 2.6: Robot móvil diferencial con marco de referencia local L y marco de referencia global G con un ángulo θ entre ellos de $\pi/2$

Para este caso dadas las velocidades $(\dot{x}, \dot{y}, \dot{\theta})$, siguiendo la ecuación [2.3] se tiene:

$$\dot{\vec{\xi}}_L = R(\pi/2)\dot{\vec{\xi}}_G = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{y} \\ -\dot{x} \\ \dot{\theta} \end{bmatrix} \tag{2.4}$$

Con esto se puede ver claramente como un movimiento sobre el eje L_x es igual a $\dot{y}$ y un movimiento en el eje L_y es igual a $-\dot{x}$.

2.4.2. Influencia del movimiento de cada rueda en el movimiento del robot móvil

Para comprender como es el movimiento del robot, es necesario entender la influencia de cada una de sus llantas en el movimiento del robot, específicamente como influye en el movimiento del punto P del chasis del robot. EL robot móvil diferencial tiene dos llantas, cada una de diámetro r, cada llanta está a una distancia l del punto

P y una velocidad de giro ω, un modelo cinemático calculará la velocidad general del robot en el marco de referencia global dadas las variables anteriores.

$$\dot{\vec{\xi}}_G = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = f(l, r, \theta, \dot{\omega}_1, \dot{\omega}_2) \tag{2.5}$$

De la ecuación (2.4) podemos encontrar el movimiento en el marco de referencia global a partir del marco local, dicha relación se expresa en la siguiente ecuación:

$$\dot{\vec{\xi}}_G = R(\pi/2)^{-1}\dot{\vec{\xi}}_L \tag{2.6}$$

Suponga que el el marco de referencia local esta alineado al robot como se ha mostrado anteriormente, primero considere la contribución de la velocidad de cada una de las llantas a la traslación del punto P en dirección de Lx, si una llanta se queda quieta y la otra se mueve, como P se encuentra en un punto medio entre ambas llantas , se moverá con la mitad de la velocidad: $\dot{x}_{r1} = (1/2)r\dot{w}_1$ y $\dot{x}_{r2} = (1/2)r\dot{w}_2$ en un robot diferencial estas dos expresiones pueden ser simplemente sumadas para calcular la componente Lx de $\dot{\vec{\xi}}_L$. En cuanto a los movimientos en el eje Ly es igual a cero dado las restricciones de la locomoción del robot. Finalmente, la velocidad angular de $\dot{\vec{\xi}}_L$ puede ser calculada por el movimiento independiente de cada llanta y después sumarlas, dicha contribución puede ser calculada de la siguiente forma, si la llanta derecha tiene una velocidad distinta de cero y la llanta izquierda tiene una velocidad angular igual a cero, entonces el movimiento general del robot en el marco de referencia general girará en contra del movimiento de las manecillas del reloj con referencia al punto P, por tal motivo la velocidad angular con respecto al punto P puede ser calculado por el movimiento a lo largo del arco de un círculo de radio $2l$, por lo tanto, se obtiene la siguiente expresión:

$$\dot{\vec{\xi}}_G = R(\theta)^{-1} \begin{bmatrix} \frac{r\dot{\omega}_1}{2} + \frac{r\dot{\omega}_2}{2} \\ 0 \\ \frac{r\dot{\omega}_1}{2l} + \frac{-r\dot{\omega}_2}{2l} \end{bmatrix} \tag{2.7}$$

Note el signo negativo en la segunda parte de la suma para el componente de velocidad angular, esto se debe a que si la llanta derecha se queda detenida mientras la otra gira en sentido positivo, el robot en el marco general se moverá en sentido de las manecillas del reloj. Siguiendo la ecuación (2.6) necesitamos calcular la matriz inversa de $R(\theta)$ la cual es :

$$R(\theta)^{-1} = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.8}$$

Suponga que el robot esta posicionado con un ángulo $\theta = \pi/2$, $r = 1$, $l = 1$ y con las velocidades $\omega_1 = 4$ y $\omega_2 = 2$ en sus llantas. Se puede obtener la velocidad en el marco de referencia global d ela siguiente manera:

$$\dot{\vec{\xi}}_G = R(\pi/2)^{-1}\dot{\vec{\xi}}_L = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \\ 1 \end{bmatrix} \tag{2.9}$$

Por lo tanto, el robot se moverá instantáneamente sobre el eje y del marco de referencia global a una velocidad de 3 y una velocidad angular de 1. Con este análisis se puede obtener la posición del robot a lo largo del tiempo a partir de la velocidad de cada una de sus llantas, sin embargo, existen restricciones de los movimientos del robot por ejemplo, una vuelta muy cerrada a alta velocidad podría hacer que el robot derrape o que se volcara, es por eso que se requiere de un análisis mas profundo para poder comprender con mayor precisión los movimientos y restricciones del robot, todo ello para poder generar movimientos eficientes y seguros.

2.5. Sensores

Los robots móviles necesitan información de su entorno para poder tomar decisiones. Para adquirir información del entorno un robot necesita sensores. Existen diversos tipos de sensores y cada uno tiene un funcionamiento diferente bajo ciertas

circunstancias, es decir, mientras que un sensor se desempeña correctamente en un entorno luminoso, otro se comporta de una manera no deseada. Es por ello que cada sensor es utilizado con una finalidad especifica, y por lo tanto es de importancia conocer algunos de ellos, en especial los que se utilizan comúnmente en los robots móviles.

2.5.1. Sensores de contacto

El parachoque es uno de los sensores más importantes en un robot móvil, su función es detectar las colisiones entre el robot y algún objeto, con la finalidad de detenerse inmediatamente o tomar alguna acción de evasión. Para un robot autónomo, es de suma importancia prevenir colisiones y si llegara a ocurrir una colisión, es necesario tomar acciones inmediatamente para evitar accidentes. Para detectar las colisiones basta utilizar un sistema de contacto el cual puede ser un botón acoplado a una carcasa que accione el botón al haber una colisión. El botón de paro de emergencias también es un sensor de contacto que al ser presionado desactiva las funciones de un robot por seguridad, este tipo de botones es necesario por cualquier problema que pueda ocurrir.

2.5.2. Sensores de distancia

Los sensores de distancia son fundamentales para poder detectar posibles colisiones o incluso para reconocer objetos. El sistema básico de funcionamiento de un sensor de distancia es ejemplificado en la figura (Fig. 2.7).

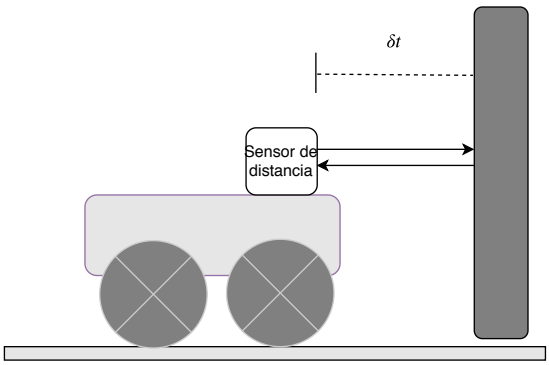


Figura 2.7: Funcionamiento de un sensor de distancia para reconocer el entorno de un robot. Un sensor emite una señal la cual rebota en una superficie y posteriormente es detectada por este mismo

Los sensores infrarrojos cuentan con un fototransistor que emite una señal infrarroja, imperceptible para el ojo humano, la cual rebota en una superficie y un receptor captura la señal de regreso. La distancia es proporcional al tiempo que tarda la señal desde que fue emitida hasta ser detectada por el receptor.

Los sensores ultrasónicos tienen un funcionamiento parecido a los infrarrojos en cuanto a calcular la distancia a un objeto midiendo el tiempo entre la emisión y la recepción de una señal, sin embargo, los sensores ultrasónicos en vez de emitir una señal infrarroja emiten una señal de audio de alta frecuencia, inaudible para el oído humano, al rededor de 40Khz.

Sin embargo este tipo de sensores pueden verse afectados por diversos factores. Por ejemplo, las características de la superficie donde rebota la señal pueden afectar la lectura del sensor. En el caso de los infrarrojos su lectura dependerá del color de la superficie , si es reflectante como un espejo o traslucida como un vidrio. El ángulo con el cual insidie la señal sobre una superficie también tiene importancia en la lectura de los sensores de distancia, pues dependiendo de ese ángulo de incidencia, la señal podría no regresar al receptor. En ocasiones si la superficie de incidencia es demasiado pequeña la señal podría no regresar al receptor. También dependiendo de la características del emisor y receptor, si la superficie se encuentran muy lejos o muy cerca, las lecturas podrían ser erróneas. Los errores mencionado anteriormente son ejemplificados en la figura (2.8).

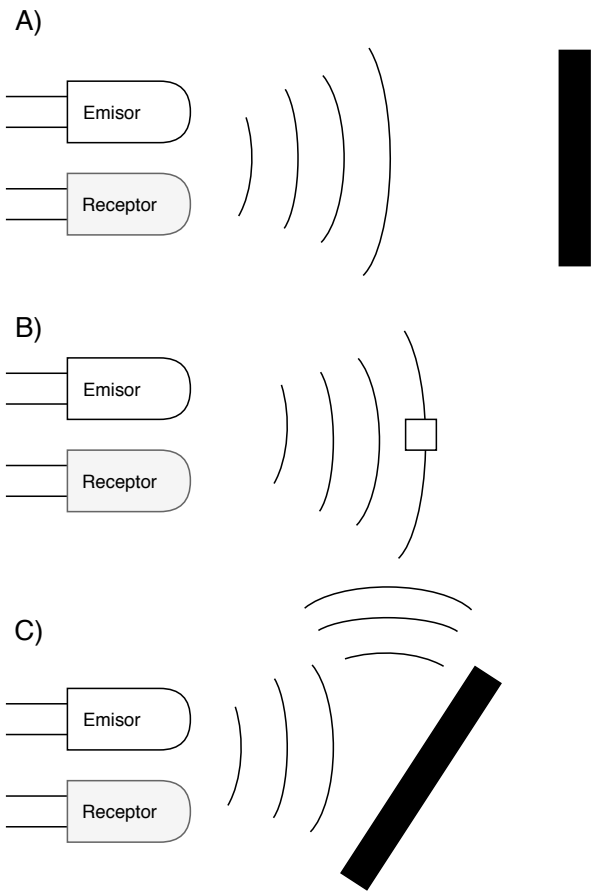


Figura 2.8: a) Distancia muy grande entre el sensor y la superficie, B) Objeto muy pequeño para ser detectado por el sensor C) Superficie en ángulo evitando que la señal regrese al sensor.

Otro tipo de dispositivos utilizados en los robots móviles son los escáner en dos dimensiones, que son sensores de distancia pero que hacen lecturas en diferentes direcciones, usualmente funcionan con un láser imperceptible al ojo humano y un motor que hace girar el sensor, estos sensores son más caros en corporación a los sensores de distancia comunes, pues la frecuencia de lecturas debe de ser mayor para que la resolución angular de sus lecturas puedan dar un escaneo aceptable del entorno.

2.6. Sensores en motores

El encoder es un transductor rotativo, que mediante una señal eléctrica sirve para indicar la posición angular del eje del rotor de un motor [2.9].

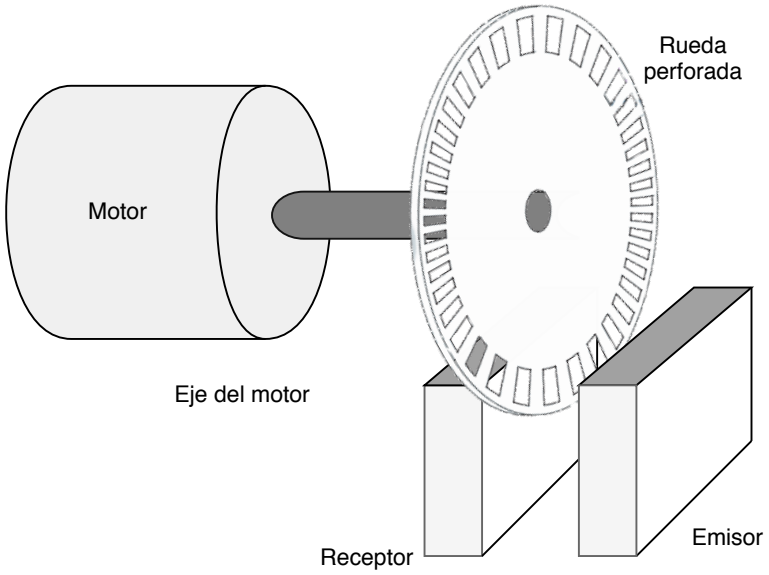


Figura 2.9: Funcionamiento básico de un encoder rotativo, donde la señal recibida por el receptor es bloqueada por el disco perforado.

2.6.1. Encoder incremental

Existen encoders llamados incrementales que generan una señal cuadrada, la cual es modificada por un disco dentado acoplado al eje del motor, el disco gira con el eje para permitir o impedir el paso de una señal infrarroja que es emitida por un emisor que se encuentra de un lado del disco dentado y que es detectada por un receptor que se encuentra del otro lado del disco (Fig. 2.9). Este tipo de encoder es utilizado en los motores de las llanta de los robots móviles, pero cuenta con dos emisores y dos receptores en el disco dentado para conocer también el sentido de giro. Hay encoders que en vez de utilizar luz infrarroja utilizan sensores magnéticos y en vez del disco dentado, utiliza un pequeño imán acoplado al eje del motor (Fig.2.10).

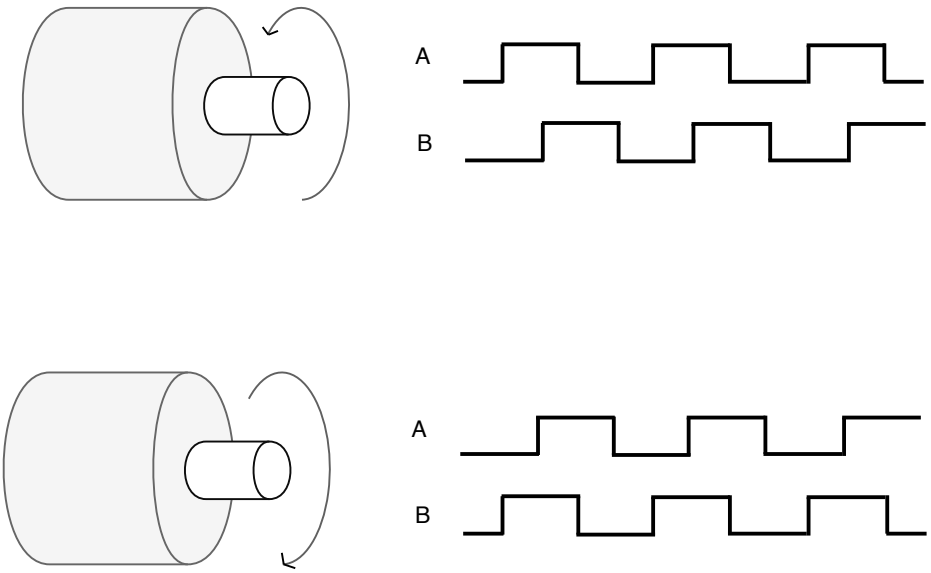


Figura 2.10: Tren de pulsos generado por cada sensor (A y B) con lo cual podemos distinguir la dirección en la que se encuentran girando los motores.

2.6.2. Encoder absoluto

También hay encoders absolutos que para cada posición del eje del motor tiene un valor diferente, de esta forma siempre se conoce cual la posición en la que se encuentra el eje del motor. Esto se logra con un disco que tiene un código único binario y el cual se lee por múltiples sensores infrarrojos posicionados a lo largo del disco (Fig. 2.12).

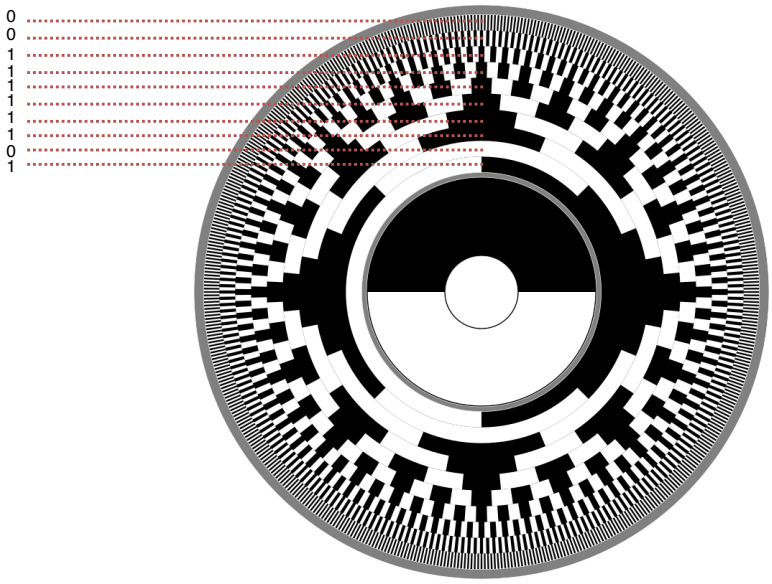


Figura 2.11: Disco de un encoder absoluto, donde idealmente cada posición del motor tiene un código asignado

2.7. Cámaras

Las cámaras son un tipo de sensor ampliamente utilizado en los robots móviles con diferentes finalidades, por ejemplo, reconocer objetos, personas, colores, gestos, tareas relacionadas con la localización de robot, entre otras. Su uso ha incrementado desde que las redes neuronales convoluciones profundas comenzaron a tener mucha importancia en el área de investigación de la visión computacional. Por lo tanto, es de suma importancia conocer el tipo de cámaras utilizadas en el área de la robótica.

2.7.1. Camaras RGB

Las cámaras más utilizadas son las cámaras RGB (del inglés red, green y blue) las cuales detectan, por medio de un sensor hecho de materiales fotosensibles, las diferentes longitudes de honda de la luz del ambiente dando lugar a una imagen a color y dependiendo del tipo de lente la imagen puede ser distorsionada y dar como resultado una imagen con un amplio rango de visión, este tipo de sensores son conocidos como pasivos pues no necesitan emitir ninguna señal para poder funcionar,

debido a que únicamente detectan las señales naturales del medio ambiente.

2.7.2. Cámaras estereoscópicas

Las cámaras estereoscópicas son aquellas que cuentan con dos cámaras separadas por una cierta distancia, similar a los ojos de muchos seres vivos como los seres humanos, debido a que tenemos dos ojos podemos percibir la distancia a la que se encuentran los objetos a nuestro alrededor. En las cámaras estereoscópicas se comparan las dos imágenes capturadas para reconstruir una escena en tres dimensiones. El procesamiento computacional que requieren este tipo de cámaras para obtener una reconstrucción en tres dimensiones, es bastante alto debido a que se requiere encontrar la correspondencia de cada uno de los píxeles de una cámara en la otra creando así una imagen que se le llama mapa de disparidad (Fig:2.13), donde los píxeles mas lejanos tienen una menor disparidad y los mas cercanos una mayor disparidad, este tipo de sensores es utilizado para detectar objetos sobre superficies como una mesa o obstáculos en el piso. Además su, funcionamiento de las cámaras estereoscópicas es ideal para funcionar en exteriores, pues la luz del sol no modifica su comportamiento.

Figura 2.12: Cámara estereoscópica comercial

Figura 2.13: Lado izquierdo imagen RGB. Lado derecho imagen de disparidad

2.7.3. Cámaras RGB-D

Existen cámaras que cuentan con más sensores que permiten calcular la profundidad de la imagen. Las cámaras RGB-D son un conjunto de sensores, uno emite luz infrarroja en varias direcciones y otro sensor recibe esas señales y por medio de algoritmos de triangulación se obtiene la distancia a la que se encuentra cada uno de los píxeles de una imagen, creando así una mapa de profundidad. Este tipo de sensores son activos, pues requieren emitir una señal al ambiente para poder funcionar. Un ejemplo de este tipo de cámaras es el sensor Kinect (Fig. 2.14).

Figura 2.14: Cámara RGB-D, sensor Kinect

Los sensores anteriormente mencionados se han descrito de manera breve pues cada uno tiene variaciones y métodos de funcionamiento muy particulares para ser presentados y analizados en este documento, además de que es probable que el funcionamiento de muchos sensores en especial las cámaras, tengan un funcionamiento muy particular para cada modelo. También es importante mencionar que un sensor ideal es aquel que no tiene ningún tipo de error en sus lecturas, sin embargo, todos

los sensores mencionado tienen errores debido a factores físicos y de fabricación, por tal motivo es importante procesar toda la información recibida por los sensores.

2.8. Localización

La localización de un robot móvil es una tarea en la cual el robot tiene que estimar la pose en la que se encuentra respecto a su eje de referencia, para poder localizarse se han propuesto diversas técnicas aunque ninguna es definitiva. Uno de los primeros métodos que se han utilizado para la localización de un robot es contabilizar los movimientos que dan las llantas del robot a través de sus encoders y así se sabría en que lugar se encuentra el robot a partir de su punto de inicio, sin embargo, en la vida real esto no ocurre debido a los pequeños errores originados por factores como: el tipo de suelo, diferentes velocidades en los motores, el viento, etc. Por lo cual, si se le indicar a un robot que debe de moverse sobre el perímetro de un cuadrado de dos metros de lado, los movimientos que debe de realizar simplemente son: avanzar dos metros, girar a la derecha noventa grados y repetir tantas veces como se le indique. Tal vez en los primeros movimientos el robot regrese al lugar de origen, sin embargo al cabo de varios movimientos la acumulación de los pequeños errores seria notoria en la posición del robot y así el robot podría reportar que está en una posición cuando en realidad se encuentra en una posición distinta (Fig. 2.15). A esta técnica se le conoce como *Dead reckoning* (navegación por estima).

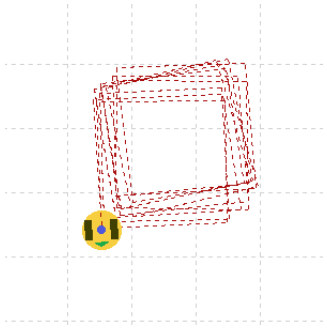


Figura 2.15: Resultado de la trayectoria de un robot móvil simulado, con error en sus movimientos al desplazarse sobre el perímetro de un cuadrado

2.8.1. Odometría

La odometría en un robot móvil se refiere a la técnica de calcular la posición del robot con referencia a los movimientos que tienen su llantas a lo largo del tiempo. La distancia reportada únicamente por los movimientos de las llantas suele tener errores proporcionales a la distancia recorrida es decir, a mayor distancia mayor será el error en la medición. Estos errores pueden ser causados por factores propios del robot móvil, algunos de esos factores son: diferente diámetro de las llantas, llantas no alineadas, llantas no balanceadas, la discretización de los encoders entre otros. También los errores pueden ser causados por razones ajenas al robot como son: suelos resbaladizos, sobre aceleración, derrapes, entre otros.

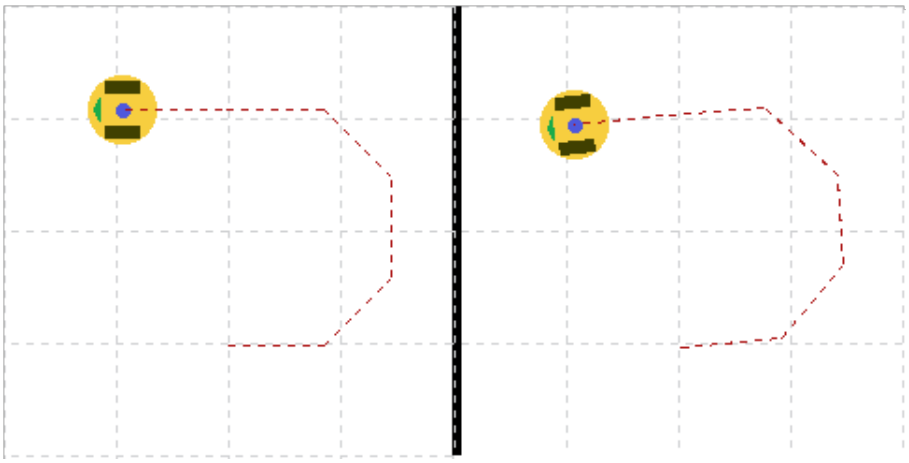


Figura 2.16: Lado izquierdo representa el caso ideal de los movimientos de un robot móvil y el lado derecho muestra los movimientos la posición del robot real después de una serie de movimientos

Es por estos errores acumulados que se requieren técnicas de localización con el menor error posible en conjunto con otras técnicas.

2.8.2. Localización por medio de marcas

Otra forma de estimar la posición y orientación de un robot, es por medio de *landmarks* (punto de referencia) en el ambiente, dichas marcas pueden ser puntos clave de algoritmos como *SURF (Speeded-Up Robust Features)* u objetos reconocidos por

una red neuronal convolucional. Dichas marcas pueden ser de cualquier tipo, sin embargo las técnicas de visión computacional requieren que dichas marcas sean visuales. Reconociendo marcas en el ambiente y con las técnicas adecuadas es posible calcular la posición de la cámara. Para que el robot pueda reportar su pose, es indispensable tener un mapa que contenga la posición de las marcas (Fig. 2.17).

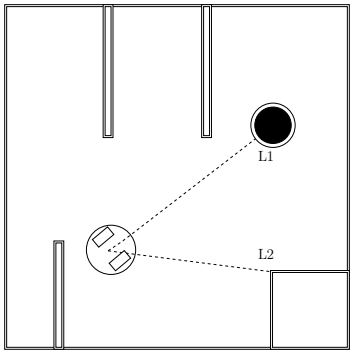


Figura 2.17: Localización por medio de marcas conocidas

Un método para estimar la posición de un robot es por medio de circunferencias, es decir conociendo la posición de las marcas en el mapa y además la distancia del robot a cada marca es posible encontrar la posición del robot [2]. Si la distancia entre el robot y las marcas representa el radio de una circunferencia, se puede encontrar el punto de intersección de las circunferencias, que seria igual a la posición del robot en un caso ideal.

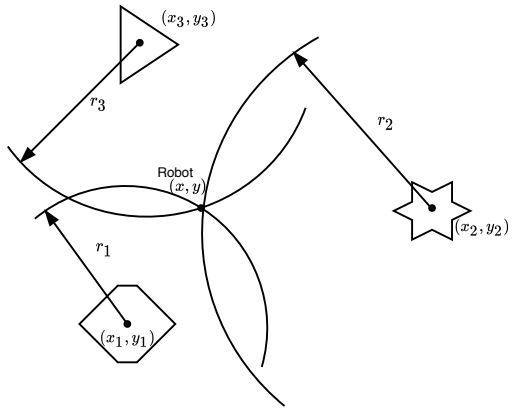


Figura 2.18: Localización de un robot móvil por medio de la intersección de las circunferencias descritas por la distancia entre el robot y cada marca detectada en el ambiente.

En la figura (Fig. 2.18) se muestra que la posición del robot es (x, y) y (x_i, y_i) representa la posicion de la i-esima marca detectad. Por lo cual las ecuaciones de cada una de las ecuaciones son:

$$(x - x_1)^2 + (y - y_1)^2 = r_1^2 \tag{2.10}$$

$$(x - x_2)^2 + (y - y_2)^2 = r_2^2 \tag{2.11}$$

$$(x - x_3)^2 + (y - y_3)^2 = r_3^2 \tag{2.12}$$

desarrollando los binomios y restando 2.11 de 2.10 y de 2.13 se obtiene:

$$2x(x_2 - x_1) + x^2{}_1 - x^2{}_2 + 2y(y_2 - y_1) + y^2{}_1 - y^2{}_2 = r^2{}_1 - x^2{}_2 \tag{2.13}$$

$$2x(x_2 - x_3) + x^2{}_3 - x^2{}_2 + 2y(y_2 - y_3) + y^2{}_3 - y^2{}_2 = r^2{}_3 - x^2{}_2 \tag{2.14}$$

las ecuaciones anteriores representan 2 rectas en el plano cartesiano. Debido a los errores del sistema y del ambiente no se asegura que las circunferencias coincidan en un

punto, Las rectas obtenidas son aquellas que pasan por los dos puntos de intersección de las circunferencias o perpendiculares a la recta que pasa por sus centros y que pasan por el espacio intermedio entre ellas (Fig. 2.19)

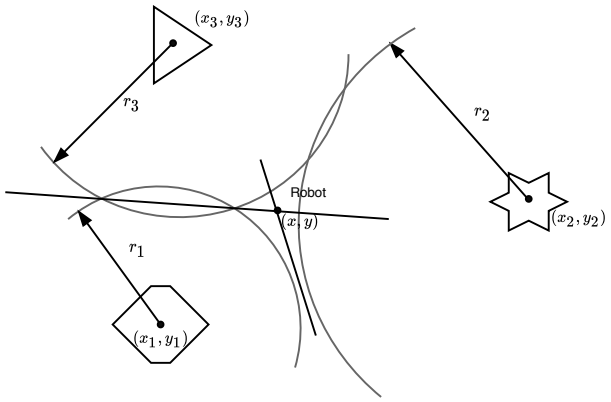


Figura 2.19: Rectas resultantes de las circunferencias descritas por r_1, r_2 y r_3. El punto de intersección representa la posición del robot.

Resolviendo las ecuaciones 2.13 y 2.14 se obtiene las coordenadas (x, y) correspondientes a la posición del robot.

$$x = \frac{(y_2 - y_1)(r^2{}_2 - r^2{}_3 - x^2{}_2 + x^2{}_3 - y^2{}_2 + y^2{}_3) - (y_3 - y_2)(r^2{}_1 - r^2{}_2 + x^2{}_1 + x^2{}_2 - y^2{}_2 1 + y^2{}_2)}{2(x_3 - x_2)(y_2 - y_1) - 2(x_3 - x_2)(y_3 - y_2)} \tag{2.15}$$

$$y = \frac{(x_2 - x_1)(r^2{}_2 - r^2{}_3 - x^2{}_2 + x^2{}_3 - y^2{}_2 + y^2{}_3) - (x_3 - x_2)(r^2{}_1 - r^2{}_2 + x^2{}_1 + x^2{}_2 - y^2{}_2 1 + y^2{}_2)}{2(x_2 - x_1)(y_3 - y_2) - 2(x_3 - x_2)(y_2 - y_1)} \tag{2.16}$$

2.8.3. Representación del ambiente

Se ha mencionado que un robot requiere de un mapa para poder localizarse sin embargo, no se ha mencionada como está constituido un mapa. Dependiendo de el

tipo de *landmarks* el mapa debe de contener informacion relevante de su posicion. Por ejemplo si los *landmarks* son señalaminetos de carreteras, es importante que el mapa contenga las coordenadas de la posición de los señalamientos. Si los *landmarks* son las paredes y objetos al rededor del robot, los cuales son detectados a través de un escáner 2D infrarrojo, el mapa debería de ser una versión discreta del espacio en celdas donde cada una tiene la propiedad de estar ocupada o vacía [2.20]. El tamaño de las celdas es dado por un equilibrio entre practicidad y espacio de almacenamiento, pues de nada serviría tener celdas de un milímetro de diámetro cuando en practica es únicamente necesario celdas de 5cm, por mencionar un ejemplo. El algoritmo *SLAM (del inglés simultaneous localization and mapping)* [1, Pagina:337] es un algoritmo que permite crear mapas y al mismo tiempo localizarse, pues ¿Cómo se puede crear un mapa sin saber donde se encuentra el robot? y ¿Cómo se puede saber donde está el robot sin un mapa?, este es el dilema que resuelve el algoritmo por medio de un enfoque probabilístico.

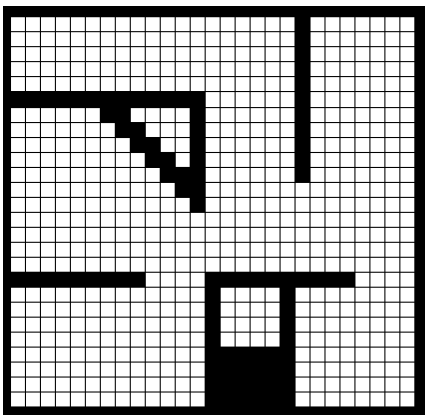


Figura 2.20: Mapa de ocupación, representación discreta del entorno real

2.8.4. Entornos estáticos y dinámicos

La localización de un robot en un entorno casi estático se da cuando el entorno sufre muy pocos cambio, de este modo el robot puede localizarse exitosamente con referencia a un mapa, por ejemplo un robot que se localiza por medio de un escáner 2D (*lidar*) comparando las paredes (en su mayoría) y objetos de un lugar con el mapa previamente hecho. Aquellos objetos que no se encuentran en el mapa pero que son

registrados por el sensor son considerados como ruido en las lecturas, de esta forma es posible localizarse con éxito, por ejemplo, en el caso del mapa de una oficina en el cual no se encontraban personas al momento de crearlo y que al momento de navegar por la oficina se encuentran algunas personas y sillas fuera de su lugar, estos cambios son considerados como ruido y de esta manera el robot puede continuar localizándose exitosamente.

Por otro lado si el robot se encuentra en un entorno donde hay muchas personas o muchos puntos de referencia se han movido de su lugar original al momento de crear el mapa, el robot no podrá localizarse adecuadamente y por tal motivo puede que nunca llegue a su destino.

Al iniciar el sistema de navegación de un robot móvil se le indica de antemano en que lugar se encuentra, pues es complicado que se localice con las lecturas de los sensores, a menos que navegue lo suficiente para encontrar marcas conocidas y así determinar con éxito su posición, ese mismo caso podría darse cuando por alguna razón el robot se localiza erróneamente, para poderse recuperar de esos errores, es necesario que el robot tenga un sistema de recuperación ante fallos, en tal caso se requeriría un sistema de localización mas sofisticado para poder determinar su pose de manera correcta.

Capítulo 3

Métodos probabilísticos de localización

3.1. Introducción

Los métodos probabilísticos de localización, han tenido resultados importantes en el área de la robótica, ya que no se enfocan en calcular el estado exacto del robot, por el contrario, se enfocan en estimar el estado más probable, dependiendo de la lectura de sus sensores y sus poses en estados anteriores.

3.2. Filtro de Kalman

El filtro de Kalman es un algoritmo cuyo objetivo es identificar el estado oculto de un sistema dinámico lineal, dicho algoritmo fue desarrollado por Rudolf E. Kalman en 1960.

3.2.1. Predicción

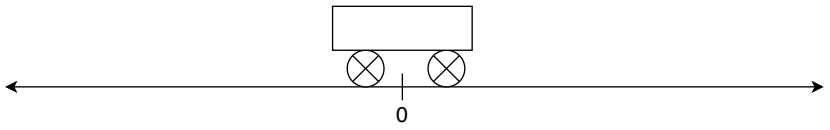


Figura 3.1: Movimiento de un robot móvil sobre una linea recta

El algoritmo cuenta con dos etapas, la primera de ellas es la predicción y la segunda la actualización. Para ejemplificar el algoritmo de Kalman se plantea la estimación de la posición de un robot móvil que se desplaza en linea recta sobre el eje x (Fig. 3.1), la variable x_k representa la posición del robot en el eje x en el momento k y la variable v_k representa la velocidad del robot en el momento k, estas dos variables son representadas por el vector $\vec{x}_k$ (3.1) que llamaremos vector de estado el cual es un vector vertical.

$$\vec{x}_k = \begin{bmatrix} x_k \\ v_k \end{bmatrix} \tag{3.1}$$

Note la diferencia entre $\vec{x}_k$ que es el vector de estado y x_k que representa la posición del robot. Al no tener otro punto de referencia mas que la posición inicial, a modo de ejemplo se puede decir que tanto la posición como la velocidad inicial son cero.

El filtro de Kalman asume que ambas variables (posición y velocidad) son aleatorias y tienen una distribución normal (fig. 3.2), cada variable tiene una media μ, la cual es el centro de la distribución y una varianza σ^2 la cual representa la incertidumbre.

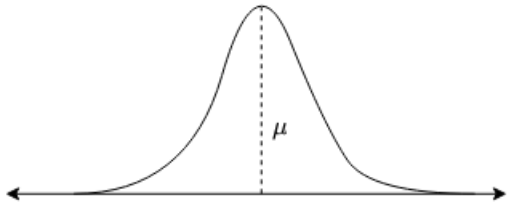


Figura 3.2: Distribución normal

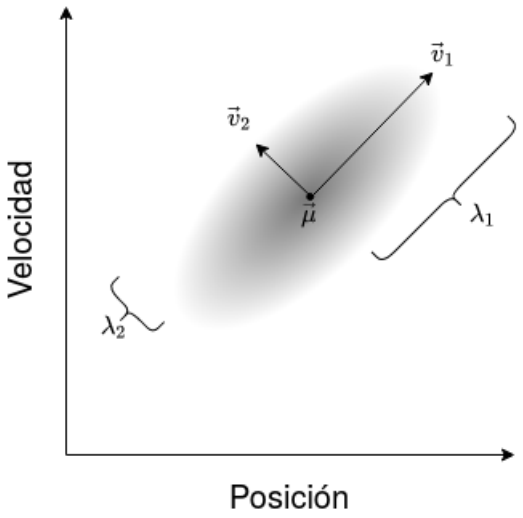


Figura 3.3: Correlación entre variables

Como se muestra en la figura (Fig. 3.3) existe una correlación entre velocidad y posición. Si la velocidad incrementa la distancia recorrida será mayor, de la misma forma si el robot se desplaza a una velocidad baja, el movimiento será menor. Este tipo de relación es importante pues el valor de una variable afecta en sentido positivo o negativo a otra. Dicha correlación es representada por la matriz de correlaciones Σ_k (3.2) donde cada uno de sus elementos muestra el grado de correlación entre las variables del vector de estado, esto quiere decir que siempre es una matriz cuadrada de tamaño nxn donde n es el tamaño del vector de estado. La forma de la distribución que se muestra en la figura (Fig. 3.3) es obtenida por los vectores y valores característicos de dicha correlación centrado en la media de cada variable. Con esta información desconocemos la posición y velocidad exacta, pero sabemos que existe un rango de posibilidades donde algunos estados (los más cercanos a la media) son más probables que otros.

$$\Sigma_k = \begin{bmatrix} \Sigma_{xx} & \Sigma_{xv} \\ \Sigma_{vx} & \Sigma_{vv} \end{bmatrix} \tag{3.2}$$

Con el estado anterior $k-1$ se debe de predecir el estado siguiente k siguiendo el modelo del sistema, para este ejemplo el sistema se basa en las ecuaciones del movimiento rectilíneo uniforme.

$$x_k = x_{k-1} + \Delta t v_{k-1} \tag{3.3}$$

$$v_k = v_{k-1} \tag{3.4}$$

Lo anterior expresado en forma matricial se puede ver como.

$$\vec{x}_k = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \vec{x}_{k-1} \tag{3.5}$$

$$\vec{x}_k = F_{k-1}\vec{x}_{k-1} + \vec{\gamma}_{k-1} \tag{3.6}$$

El vector $\vec{\gamma}_k$ es de las mismas dimensiones que el vector de estado y representa el ruido gaussiano añadido al sistema por el entorno. Existen fuerzas externas al sistema pero que podemos conocer de cierta manera, por ejemplo, los comandos que se le dan al robot. En este caso podríamos suponer que conocemos la aceleración del sistema, pues se conoce con antelación cual es el comando que el robot va a ejecutar. Por tanto la ecuación (3.3) se actualiza a:

$$x_k = x_{k-1} + \Delta t v_{k-1} + \frac{1}{2} a \Delta t^2 \tag{3.7}$$

Y la ecuación (3.4) se actualiza a:

$$v_k = v_{k-1} + a\Delta t \tag{3.8}$$

En forma matricial se tiene $\vec{x}_k$:

$$\vec{x}_k = F_{k-1}\vec{x}_{k-1} + \begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{bmatrix} a \tag{3.9}$$

$$\vec{x}_k = F_{k-1}\vec{x}_{k-1} + B_{k-1}\vec{u}_{k-1} + \vec{\gamma}_{k-1} \tag{3.10}$$

La ecuación (3.10) representa la probabilidad de transición de estado $p(x_k|u_t, x_{k-1})$

donde u_t es el comando de movimiento que se le indica al robot, B_k se le conoce como la matriz de control y $\vec{u}_k$ como el vector de control, para sistemas sin fuerzas externas o sistemas muy sencillos se pueden omitir ambos valores, como es el caso de este ejemplo.

Hasta el momento las fuerzas que se incluyen en las ecuaciones son fuerzas conocidas. El algoritmo de Kalman es capaz de lidiar con fuerzas desconocidas, dichas fuerzas pueden ser el viento, la fricción entre llantas y el suelo, deformidades sobre la superficie en la que el robot se desplaza, entre muchas otras, las cuales se representaran como $\vec{\gamma}_k$ esta variable representa el ruido en el sistema como ya se ha comentado.

$$\vec{x}_k = F_k\vec{x}_{k-1} + \vec{\gamma}_{k-1} \tag{3.11}$$

Los valores del vector $\vec{\gamma}$ desconocen y de conocerse no habría necesidad de utilizar técnicas como el algoritmo de Kalman. Por lo tanto, para predecir el nuevo estado $\vec{x}_k$ se utilizara la ecuación (3.6) y para actualizar la matriz Σ_k se actualiza por medio de la siguiente ecuación:

$$\Sigma_k = F_{k-1}\Sigma_{k-1}F_{k-1}^T \tag{3.12}$$

La matriz Q representa la incertidumbre de la nueva posición del robot en el estado Σ_k.

$$\Sigma_k = F_{k-1}\Sigma_{k-1}F_{k-1}^T + Q \tag{3.13}$$

En conclusión para la etapa de predicción, la nueva estimación $\vec{x}_k$ es una predicción hecha por la estimación previa $\vec{x}_{k-1}$ más una corrección por las fuerzas conocidas $B_{k-1}u_{k-1}$ (que en este ejemplo no se toman en cuenta por simplicidad) y la matriz de incertidumbre Σ_k es actualizada por la incertidumbre del estado anterior Σ_{k-1} más la incertidumbre del ambiente Q.

3.2.2. Actualización

En esta etapa se hace un ajuste mayor con las lecturas de los sensores disponibles. El primer paso consiste en obtener las lecturas de los sensores esperadas dado que el robot se encuentra en el estado resultante de la etapa de predicción. Esto se logrará con una matriz que modela a partir de la predicción anterior la lectura de los sensores esperada, nos referiremos a esta matriz como C_k (Fig. 3.4) y al resultado de esta operación μ_e, el ruido sera representado por una matriz de correlación que llamaremos Σ_e para calcular ambas valores, se sigue una metodología similar a la que se utilizó en el paso de predicción, por lo cual tendremos las siguientes ecuaciones:

$$\vec{\mu}_e = C_k \vec{x}_k \tag{3.14}$$

$$\Sigma_e = C_k \Sigma_k C_k^T \tag{3.15}$$

Las unidades de los resultados de estas ecuaciones corresponden a las unidades de los sensores y no a las del vector de estado. En otras palabras, esto se puede ver como simular las lecturas que deberían tener los sensores dado que el robot se encuentra en la posición predicha en la etapa de predicción.

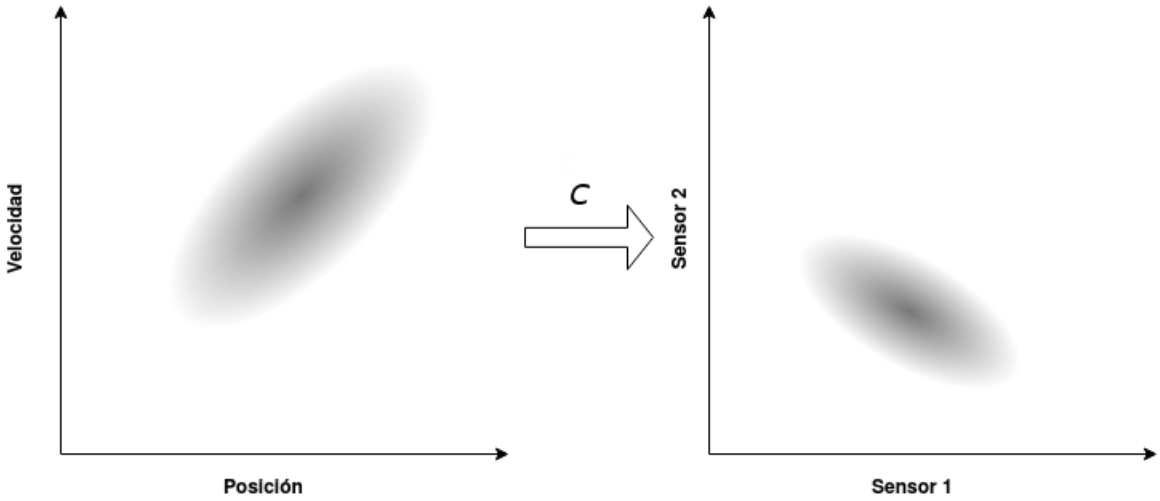


Figura 3.4: Lecturas de sensores modeladas por medio de la matriz C_k

A los valores reportados por los sensores reales los englovaremos en un vector

$\vec{z}_k$ (fig:3.5) que a la vez será el centro de una distribución del estado reportado por los sensores y la matriz de covarianza de dicha distribución será representada por la matriz R_k la cual se obtiene experimentalmente y representa la incertidumbre de las lecturas.

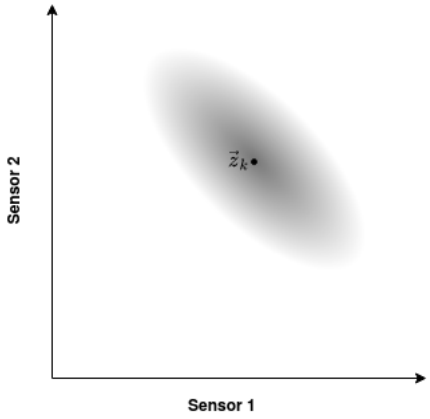


Figura 3.5: Lecturas de sensores modeladas por medio de la matriz C_k

Para combinar las distribuciones de los sensores esperados y los de los sensores reales se tomara un enfoque donde se combina por medio de la multiplicación de dos distribuciones gaussianas en una sola dimensión, para después extrapolarlo a casos donde las distribuciones son multivariadas, este enfoque esta basado en [3] .

La ecuación de una campana gaussiana con varianza σ^2 y media μ es:

$$N(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x-\mu^2}{2\sigma^2}} \tag{3.16}$$

Lo que se desea es encontrar el valor resultante de la intersección de dos distribuciones gaussianas [fig:3.6].

$$N(x, \mu', \sigma') = N(x, \mu_0, \sigma_0) * N(x, \mu_1, \sigma_1) \tag{3.17}$$

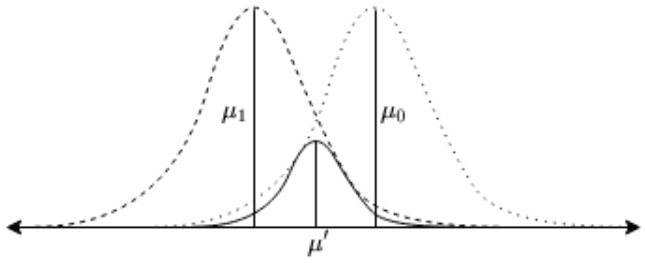


Figura 3.6: Producto de dos distribuciones normales

sutituyendo la ecuación 3.16 en la ecuación anterior se tiene:

$$\mu' = \mu_0 + \frac{\sigma_0^2(\mu_1 - \mu_0)}{\sigma_0^2 + \sigma_1^2} \tag{3.18}$$

$$\sigma' = \sigma_0^2 - \frac{\sigma_0^4}{\sigma_0^2 + \sigma_1^2} \tag{3.19}$$

ahora llamaremos K al siguiente termino:

$$K = \frac{\sigma_0^2}{\sigma_0^2 + \sigma_1^2} \tag{3.20}$$

Sustituyendo K en (3.18) y (3.19)

$$\mu' = \mu_0 + K(\mu_1 - \mu_0) \tag{3.21}$$

$$\sigma' = \sigma_0^2 - K\sigma_0^2 \tag{3.22}$$

Expresando lo anterior en forma matricial se tiene:

$$K = \Sigma_0(\Sigma_0 + \Sigma_1)^{-1} \tag{3.23}$$

$$\vec{\mu}' = \vec{\mu}_0 + K(\vec{\mu}_1 - \vec{\mu}_0) \tag{3.24}$$

$$\Sigma' = \Sigma_0 - K\Sigma_0 \tag{3.25}$$

La variable K es llamada la ganancia de Kalman. Regresando al ejemplo del robot que se mueve en linea recta , se tienen dos distribuciones, la esperada $(\mu_e, \Sigma_e) = (C_k\vec{x}_k, C_k\Sigma_k C_k^T)$ y la observada $(\vec{z}_k, R_k)$

Exponiendo lo anterior pero con la notación matricial mencionada en (3.23)y (3.24) se tiene:

$$C_k\vec{x}_k = C_k\vec{x}_k - K(\vec{z}_k - C_k\vec{x}_k) \tag{3.26}$$

$$C_k\Sigma_k C_k^T = C_k\Sigma_k C_k^T - KC_k\Sigma_k C_k^T \tag{3.27}$$

La ganancia de Kalman vista en (3.25) pero para este ejemplo es:

$$K = C_k\Sigma_k C_k^T(C_k\Sigma_k C_k^T + R_k)^{-1} \tag{3.28}$$

Resumiendo las ecuaciones anteriores y eliminando el termino C_k de cada ecuación, se tiene:

$$K = \Sigma_k C_k^T(C_k\Sigma_k C_k^T + R_k)^{-1} \tag{3.29}$$

$$\vec{x}_k = \vec{x}_k - K(\vec{z}_k - C_k\vec{x}_k) \tag{3.30}$$

$$\Sigma_k = \Sigma_k - KC_k\Sigma_k \tag{3.31}$$

De esta forma se obtiene la predicción final en $\vec{x}_k$ y Σ_k. Con el nuevo comando de movimiento , se tendrá que volver a iterar sobre el algoritmo presentado donde $\vec{x}_k$ pasa a ser $\vec{x}_{k-1}$ y Σ_k a Σ_{k-1}. El procedimiento anterior se basa en [4]. El procedimiento mencionado se muestra en la figura (Fig. 3.7).

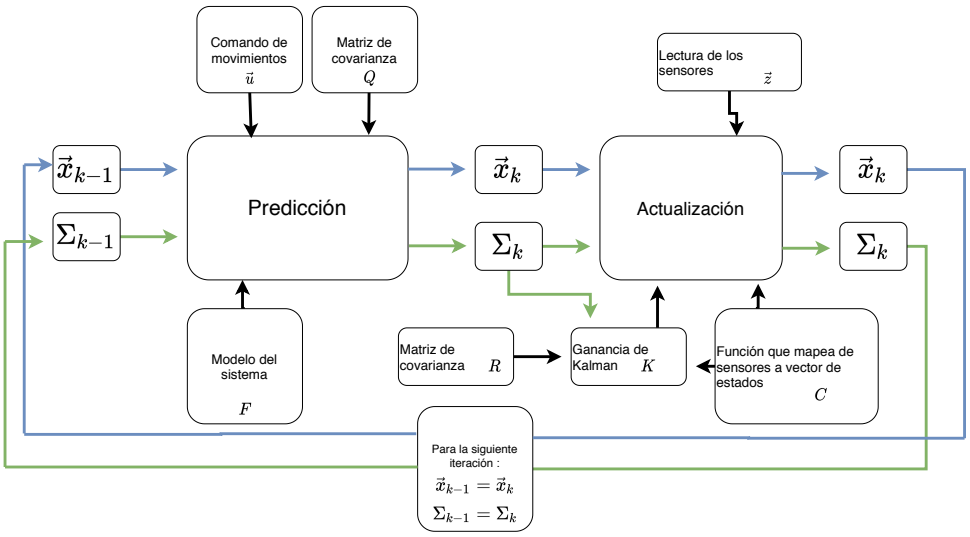


Figura 3.7: Diagrama de flujo del Filtro de Kalman

El procedimiento completo se muestra en el algoritmo 1.

Algorithm 1 Filtro de Kalmam

procedure KF($\vec{x}_{k-1}, \Sigma_{k-1}, \vec{u}_k, \vec{z}_k$)
    Predicción
    $\vec{x}_k = F_{k-1}\vec{x}_{k-1} + B_{k-1}\vec{u}_{k-1}$
    $\Sigma_k = F_{k-1}\Sigma_{k-1}F_{k-1}^T + Q$
    Actualización
    $K = \Sigma_k C_k^T (C_k \Sigma_k C_k^T + R)^{-1}$
    $\vec{x}_k = \vec{x}_k - K'(\vec{z}_k - C_k\vec{x}_k)$
    $\Sigma_k = \Sigma_k - K'C_k\Sigma_k$
    return $(\vec{x}_k, \Sigma_k)$
end procedure

3.3. Filtro de Kalman Extendido

El KF (Filtro de Kalman) trabaja adecuadamente bajo el supuesto de que las lecturas de los sensores son una función lineal del estado y además de que el estado siguiente es una función lineal del estado anterior, esto con el fin de asegurar que siempre se tendrá una distribución gaussiana y el algoritmo funcione correctamente, pues se basa en distribuciones gaussianas de varias variables. Al evaluar una distribución gaussiana en una función lineal, se asegura que el resultado es otra distribución

gaussiana (Fig. 3.8)

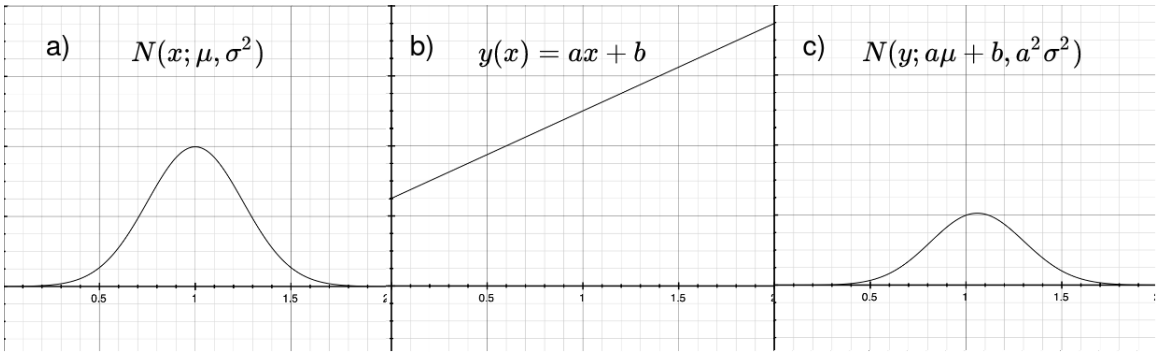


Figura 3.8: a)Distribución gaussiana con media μ y varianza σ^2 b) Función lineal del tipo $ax + b$ c) El resultado de despejar x de $y(x)$ y evaluarlo en **a)**

Por otro lado, si la función no es lineal, entonces el resultado será una función diferente a una distribución gaussiana (Fig. 3.9), por tal motivo se debe de asegurar que el resultado tanto en la etapa de predicción y actualización es una distribución normal.

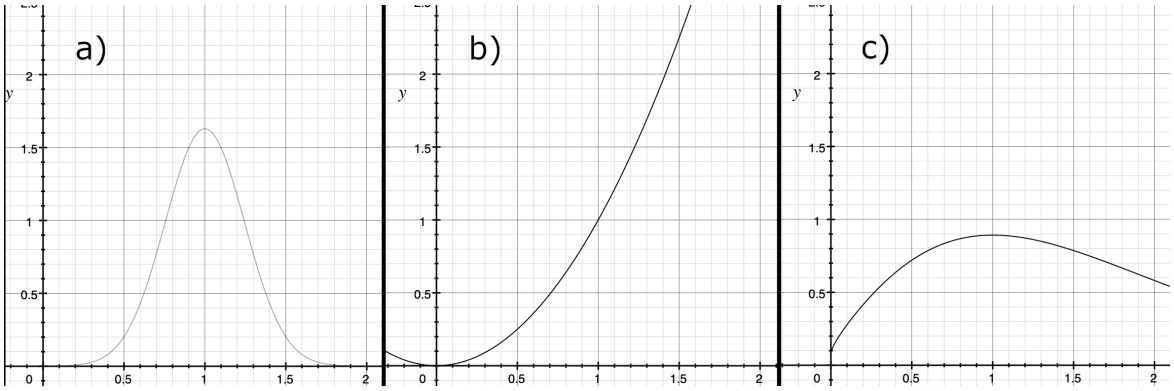


Figura 3.9: a)Distribución gaussiana con media μ y varianza σ^2 b) Función no lineal del tipo x^2 c) El resultado de despejar x de $y(x)$ y evaluarlo en **a)**, como se puede ver el resultado no es una distribución gausiana

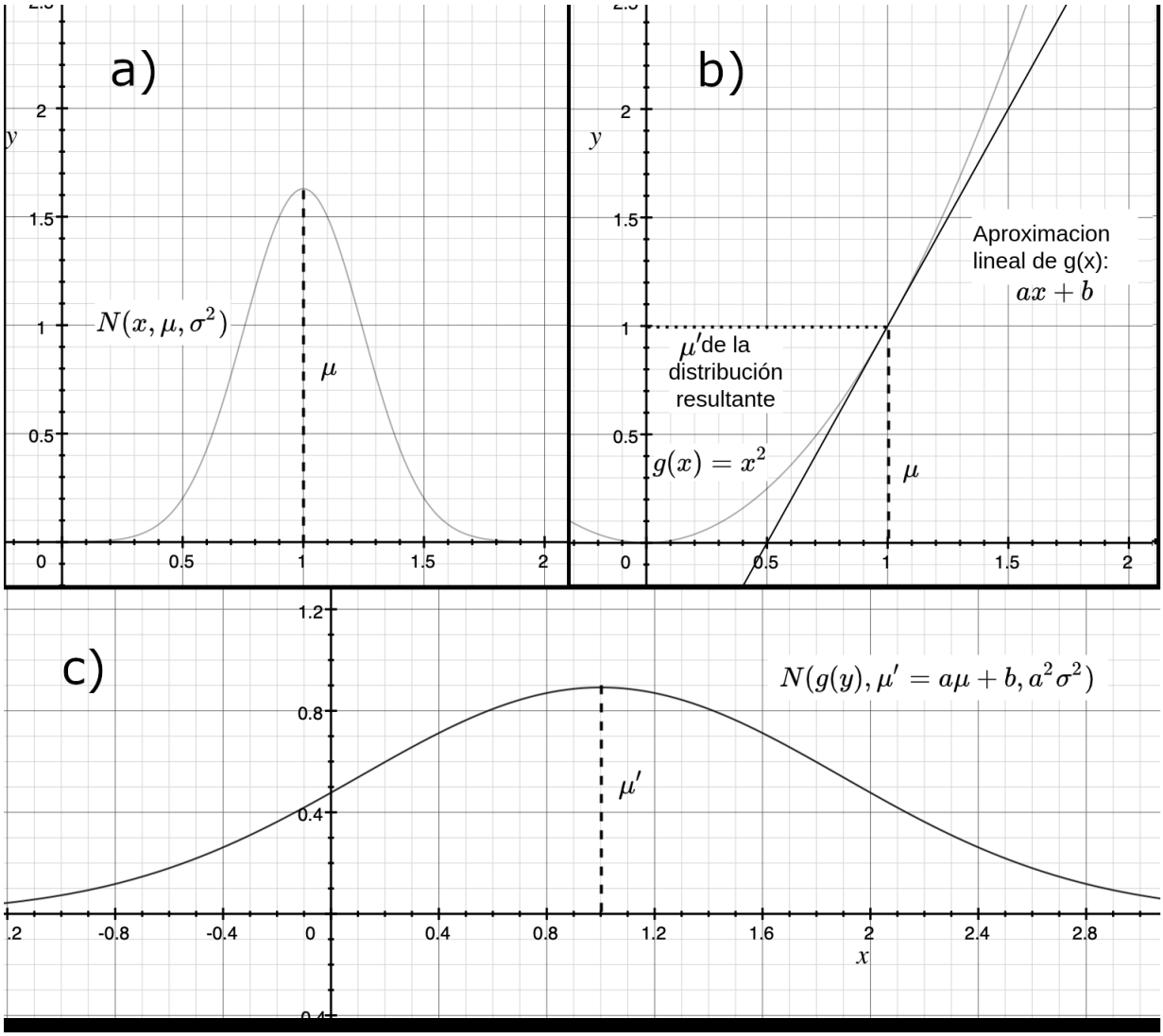


Figura 3.10: a)Distribución normal $N(x, \mu, \sigma^2)$ b) Recta $ax + b$ resultante de la linelización de $g(x)$ en el punto $g(\mu)$ c) Distribución normal resultante $N(g(y), \mu' = a\mu + b, a^2\sigma^2)$

El EKF (filtro de Kalman extendido) resuelve el problema de trabajar únicamente para sistemas lineales. Para resolver dicho problema, el EKF trata de aproximar una función no lineal en un punto por medio de la recta tangente a el. El punto en el cual se debe de hacer dicha aproximación, es en el punto mas probable de una distribución normal, esto es en la media μ esto se ve ejemplificado en (Fig. [3.10]):

Una de las principales diferencias entre el EKF y KF radica en la etapa de predicción (3.10) que es reescrita aquí por conveniencia:

$$\vec{x}_k = F_k\vec{x}_{k-1} + B_{k-1}\vec{u}_{k-1} + \gamma_k \tag{3.32}$$

pasa a ser una función g:

$$\vec{x}_k = g(u_t, \vec{x}_{k-1}) + \gamma_k \tag{3.33}$$

Dicha función es no lineal y tiene que ser linealizada por su matriz jacobina.

$$G = J_{g(\vec{u}_k, \vec{x}_{k-1})} \tag{3.34}$$

De la misma forma la ecuación (3.14) que es reescrita aquí por conveniencia:

$$\mu_e = C_k\vec{x}_k \tag{3.35}$$

se convierte en la función no lineal:

$$\mu_e = h(\vec{x}_k) + \delta_k \tag{3.36}$$

y su matriz Jacobiana es:

$$H = J_{h(\vec{x}_k)} \tag{3.37}$$

La matriz jacobiana es una matriz formada por las derivadas parciales de primer orden de una función. Con esta matriz se aproxima linealmente a la función en un punto. En este sentido, el jacobiano representa la derivada de una función multivariable. De esta forma la matriz jacobiana G corresponde a las matriz F, además la matriz jacobiana H corresponde a la matriz C. A continuación se muestra el algoritmo para el filtro de Kalman extendido.

Algorithm 2 Filtro de Kalman Extendido

procedure EKF($\vec{x}_{k-1}, \Sigma_{k-1}, \vec{u_k}, \vec{z_k}$)
    Predicción
    $\vec{x}_k = g(u_t, \vec{x}_{k-1})$
    $\Sigma_k = G_k \Sigma_k G_k^T + Q$
    Actualización
    $K = \Sigma_k H_k^T (H_k \Sigma_k H_k^T + R_k)^{-1}$
    $\vec{x}_k = \vec{x}_k - K(\vec{z}_k - h(\vec{x}_k))$
    $\Sigma_k = \Sigma_k - K H_k \Sigma_k$
    return $(\vec{x}_k, \Sigma_k)$
end procedure

3.4. Localización de un robot móvil con el Filtro de Kalman Extendido

A continuación se muestra la implementación del filtro de Kalman Extendido en un problema de localización para un robot móvil en un simulador (Apéndice A), con la finalidad de mostrar los detalles de implementación, es importante mencionar que dicho simulador se desarrolló para hacer las pruebas simuladas de esta tesis..

En el simulador (fig: 3.13) se cuenta con un robot móvil de par diferencial que recibe comandos de giro y de avance (θ, d), además cuenta con sensores. El primero de ellos es el sensor de luz que identifica en que dirección se encuentra la fuente luminosa respecto al robot, además cuenta con un sensor que simula un escáner láser en dos dimensiones, con el cual reconoce los obstáculos. Las marcas conocidas son los polígonos que se encuentran en el entorno y que son capturados por el sensor láser. El robot se mueve por medio de un comportamiento reactivo done su objetivo es llegar a la fuente luminosa y además evitar los obstáculos. Al navegar el robot reporta su posición por medio del EKF. Dicho algoritmo se describe a en el algoritmo 3.

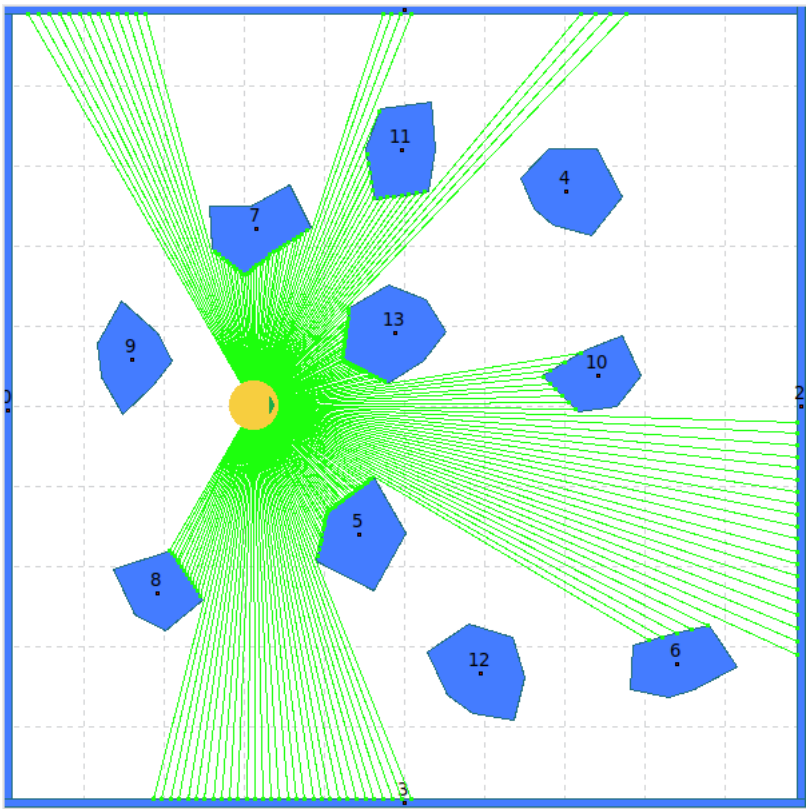


Figura 3.11: Robot móvil de par diferencial simulado, con lecturas de un escáner en dos dimensiones simulado

El algoritmo requiere 4 parámetros para funcionar, el primero de ellos es la pose de la iteración anterior $\vec{x}_{k-1}$, si es el primer movimiento dicho vector contendrá la pose inicial del robot. El segundo parámetro es la Matriz de covarianza de la iteración anterior Σ_{k-1}, si es el primer movimiento los valores de dicha matriz serán todos ceros. El tercer parámetro es el comando $\vec{u}$ que deberá ejecutar el robot, dicho vector contiene dos componentes un valor de avance u_d y uno de giro u_θ.

$$\vec{u} = \begin{bmatrix} u_d \\ u_\theta \end{bmatrix} \tag{3.38}$$

El ultimo parámetro S son los landmarks reconocidos por el robot.

$$S = \begin{bmatrix} x_1 & y_1 & r_1 & \phi_1 & m_1 \\ x_2 & y_2 & r_2 & \phi_2 & m_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & r_n & \phi_n & m_n \end{bmatrix} \tag{3.39}$$

La matriz S es de tamaño 5xn donde el número de renglones n corresponde al número de landmarks detectados, las cinco columnas representan las características de cada landmark. Donde x_i y y_i representan la posición de los landmarks idealmente, r_i es la distancia del robot al landmark i y θ_i la orientación del landmark respecto al robot, r_i y ϕ_i son los valores obtenidos a partir de la lectura de los sensores, m_i representa el identificador del landmark.

Las lineas 3 4 y 5 del algoritmo 3 representan la predicción del vector de estado $\vec{x}_k$ reescribiendo dichas lineas en una expresión mas rigurosa se tiene:

$$\begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \theta_{k-1} \end{bmatrix} + \begin{bmatrix} u_d * cos(\theta_{k-1} + u_\theta) \\ u_d * sin(\theta_{k-1} + u_\theta) \\ \theta_{k-1} + u_\theta \end{bmatrix} \tag{3.40}$$

escrito de otra forma esto es:

$$\vec{x}_k = g(\vec{u}_k, \vec{x}_{k-1}) \tag{3.41}$$

Lo cual representa que el vector de estado es actualizado por medio de una función g que involucra el estado anterior $\vec{x}_{k-1}$ y un comando $\vec{u}_k$. Al ser g una función no lineal es necesario linealizarla por medio de su matriz jacobiana:

$$G = \frac{\partial g(\vec{u}_k, \vec{x}_{k-1})}{\partial \vec{x}_{k-1}} = \begin{bmatrix} \frac{\partial x_k}{\partial x_{k-1}} & \frac{\partial x_k}{\partial y_{k-1}} & \frac{\partial x_k}{\partial \theta_{k-1}} \\ \frac{\partial y_k}{\partial x_{k-1}} & \frac{\partial y_k}{\partial y_{k-1}} & \frac{\partial y_k}{\partial \theta_{k-1}} \\ \frac{\partial \theta_k}{\partial x_{k-1}} & \frac{\partial \theta_k}{\partial y_{k-1}} & \frac{\partial \theta_k}{\partial \theta_{k-1}} \end{bmatrix} \tag{3.42}$$

La etapa de actualización se lleva a cabo por cada uno de los landmarks recono-

cidos, donde los componentes de $\hat{z}$ representan la distancia y el ángulo a partir del vector de estado obtenido en la etapa de predicción, dichos valores son los esperados a partir de la posición donde se encuentra el robot, sin embargo, esto no se cumple debido al ruido del ambiente, el vector z representa los valores de distancia s_r y ángulo s_ϕ reportados por los sensores. La linea 10 del algoritmo 3 representa lo siguiente:

$$\hat{\vec{z}}_i = h(S_{i,*}, \vec{x}_k) = \begin{bmatrix} r \\ \phi \\ m \end{bmatrix} = \begin{bmatrix} \sqrt{(S_{i,x} - x_k)^2 + (S_{i,y} - y_k)^2} \\ atan2(S_{i,y} - y_k, S_{i,x} - x_k) - \theta_k \\ S_{i,m} \end{bmatrix} \tag{3.43}$$

Para poder linealizar esta función se requiere de su matriz jacobiana:

$$H_i = \frac{\partial h(S_{i,*}, \vec{x}_k)}{\partial \vec{x}_k} = \begin{bmatrix} \frac{\partial r}{\partial x_k} & \frac{\partial r}{\partial y_k} & \frac{\partial r}{\partial \theta_k} \\ \frac{\partial \phi}{\partial x_k} & \frac{\partial \phi}{\partial y_k} & \frac{\partial \phi}{\partial \theta_k} \\ \frac{\partial m}{\partial x_k} & \frac{\partial m}{\partial y_k} & \frac{\partial m}{\partial \theta_k} \end{bmatrix} \tag{3.44}$$

$$H_i = \begin{bmatrix} \frac{-(S_{i,x} - x_k)}{\sqrt{(S_{i,x} - x_k)^2 + (S_{i,y} - y_k)^2}} & \frac{-(S_{i,y} - y_k)}{\sqrt{(S_{i,x} - x_k)^2 + (S_{i,y} - y_k)^2}} & 0 \\ \frac{(S_{i,y} - y_k)}{(S_{i,x} - x_k)^2 + (S_{i,y} - y_k)^2} & \frac{-(S_{i,x} - x_k)}{(S_{i,x} - x_k)^2 + (S_{i,y} - y_k)^2} & -1 \\ 0 & 0 & 0 \end{bmatrix} \tag{3.45}$$

La siguiente lineas de código corresponden al algoritmo 2 de la sección 3.3. La etapa de actualización se repite tantas veces como landmarks son observados, esto bajo el supuesto de que cada observación es independiente.

Algorithm 3 Filtro de Kalman Extendido

procedure EKF($\vec{x}_{k-1}, \Sigma_{k-1}, \vec{u}, S$)
  Predicción
  $\theta_k \leftarrow \theta_{k-1} + u_\theta$
  $x_k \leftarrow x_{k-1} + u_d * cos(\theta_k)$
  $y_k \leftarrow y_{k-1} + u_d * sin(\theta_k)$
  $\mathrm{G} \leftarrow \begin{bmatrix} 1 & 0 & -u_d * sin(\theta_k) \\ 0 & 1 & u_d * cos(\theta_k) \\ 0 & 0 & 1 \end{bmatrix}$
  $\Sigma_k \leftarrow G\Sigma_{k-1}G^T + Q$
  Actualización
  for $(i = 0 \ ; \ i < n \ ; \ i++ \ )$ **do**
    $\hat{z} \leftarrow \begin{bmatrix} \sqrt{(S_{i,x} - x_k)^2 + (S_{i,y} - y_k)^2} \\ atan2(S_{i,y} - y_k, S_{i,x} - x_k) - \theta_k \\ S_{i,m} \end{bmatrix}$
    $z \leftarrow \begin{bmatrix} S_{i,r} \\ S_{i,\phi} \\ S_{i,m} \end{bmatrix}$
    $H \leftarrow \begin{bmatrix} \frac{-(S_{i,x} - x_k)}{\sqrt{(S_{i,x} - x_k)^2 + (S_{i,y} - y_k)^2}} & \frac{-(S_{i,y} - y_k)}{\sqrt{(S_{i,x} - x_k)^2 + (S_{i,y} - y_k)^2}} & 0 \\ \frac{(S_{i,y} - y_k)}{(S_{i,x} - x_k)^2 + (S_{i,y} - y_k)^2} & \frac{-(S_{i,x} - x_k)}{(S_{i,x} - x_k)^2 + (S_{i,y} - y_k)^2} & -1 \\ 0 & 0 & 0 \end{bmatrix}$
    $L \leftarrow (H * \Sigma * H^T) + R$
    $K \leftarrow \Sigma * H^T * L^{-1}$
    $V \leftarrow z - \hat{z}$
    $\vec{x}_k \leftarrow \vec{x}_k + (K * V)$
    $\Sigma_k \leftarrow (I - K * H) * \Sigma_k$
  end for
  return $(\vec{x}_k, \Sigma_k)$
end procedure

El diagrama de flujo del algoritmo presentado anteriormente se muestra en la figura (Fig. 3.12).

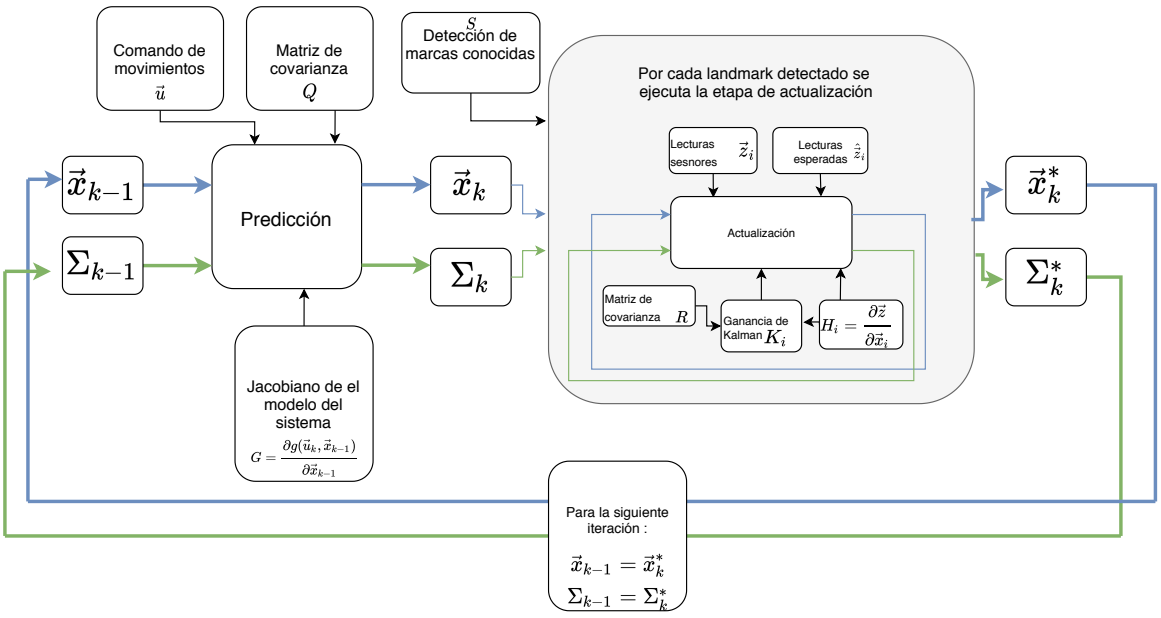


Figura 3.12: Diagrama de flujo para el filtro de kalman extendido

La precisión y exactitud dependerá de la cantidad de marcas reconocidas, a mayor marcas detectadas por el robot, este tendrá una localización más precisa, sin embargo, si el robot reconoce pocas marcas la incertidumbre de la estimación será mayor y por lo tanto menos precisa, además si el robot no reconoce ninguna marca, únicamente se guiará por los comandos que recibe el robot, es decir, únicamente se localizará con el paso de predicción, el filtro de Kalman extendido puede lidiar con estos caso siempre y cuando sean breves los momentos en los que no se reconoce ninguna marca, pues al no reconocer ninguna marca en un lapso de tiempo largo la localización tendría los problemas de *Dead reckoning* (navegación por estima) los cuales fueron mencionados en la sección 2.8.

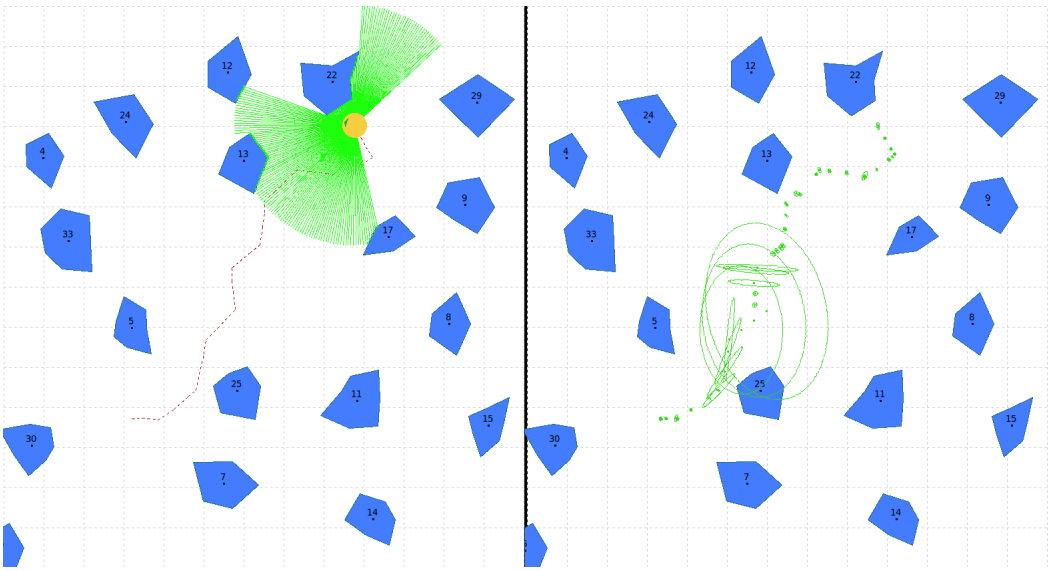


Figura 3.13: Simulación de un robot móvil localizándose por medio del filtro de kalman bajo condiciones de ruido en sus lecturas

En la figura (Fig. 3.13) del lado izquierdo se muestra un robot móvil que ha descrito una trayectoria (linea punteada). Por medio de sus sensores (color verde) reconoce los diferentes landmarks (centroide de cada polígono) que se encuentran en el mapa, como se puede apreciar el rango no es tan amplio para reconocer todas las marcas a su alrededor. Del lado derecho se muestra las posiciones obtenidas por el Filtro de Kalman Extendido, se aprecia que las posiciones son muy cercanas a las descritas por el robot (lado izquierdo) sin embargo, la incertidumbre crece cuando no puede reconocer marcas cercanas, la incertidumbre se representa por las elipses verdes. Esta simulación cuenta con ruido gaussiano en los movimientos del robot y las lectura de sus sensores. En esta simulación el id de cada una de las marcas es proporcionada por el simulador, sin embargo, en la practica con un robot real, se requiere de un sistema de visión que cumpla con la tarea de identificar marcas en el ambiente.

3.5. Otros métodos probabilísticos

También existen otros métodos de localización probabilísticos: El filtro de partículas AMCL [5] (Adaptative Monte Carlo Localization) o por medio de Modelos Ocultos de Markov. Sin embargo, en este trabajo de tesis no se abordaran, pues la

finalidad de este trabajo de tesis se enfoca en el desarrollo de una metodología de localización con el filtro de Kalman, la cual permita posteriormente el análisis entre diferentes métodos probabilísticos de localización.

Capítulo 4

Técnicas de visión computacional

4.1. Introducción

La Visión Computacional tiene dos objetivos. Desde el punto de vista de la ciencia biológica, la visión por computadora tiene como objetivo generar modelos computacionales del sistema visual humano. Desde el punto de vista de la ingeniería, la visión por computadora tiene como objetivo construir sistemas autónomos que puedan realizar algunas de las tareas que el sistema visual humano puede realizar. [6] Este trabajo se enfoca en el segundo punto de vista, pues se utilizar varias técnicas que en conjunto resuelven una tarea que para un ser humano es muy fácil de realizar, dicha tarea es el reconocimiento de marcas en el ambiente.

El reconocimiento de un objeto es una tarea fácil para un ser humano, por ejemplo si se le pidiera a una persona mencionar todos los objetos en un escritorio de oficina, la persona podrá con mucha facilidad realizar esta tarea, sin embargo, al analizar cada uno de los pasos que conlleva esta tarea, nos daremos cuenta que la persona es capaz de diferenciar un escritorio del resto de objetos en la oficina, también identificar cada uno de los objetos aunque partes de ellos no se encuentran visibles y asociar cada objeto con su nombre (Fig. 4.2). Realizar este trabajo para una computadora es un tarea compleja. A demás, una imagen es una representación en dos dimensiones mientras que los seres humanos percibimos el mundo en tres dimensiones. Para que una computadora pueda realizar esta tarea, es necesario seguir una metodología con

diferentes procesos y ejecución de algoritmos, es por esta razón que es importante conocer algunos de los métodos que hacen que una computadora pueda reconocer objetos y en consecuencia un robot pueda identificar objetos o marcas de interés a su alrededor.

Figura 4.1: a) Imagen original b) Reconocimiento del área de interés (escritorio) c) Reconocimiento de cada objeto

4.2. Procesamiento digital de imágenes

Al conjunto de técnicas y procesos para descubrir o hacer resaltar información contenida en una imagen usando como herramienta principal una computadora se le conoce como procesamiento digital de imágenes (PDI). Hoy en día, el PDI es un área de investigación muy específica en computación y está muy relacionada con el procesamiento digital de señales. Esta relación estriba en el hecho de que en esencia el PDI es una forma muy especial del procesamiento digital de señales en dos o tres dimensiones.[7] El procesamiento de imágenes es muy utilizado en el área de la medicina y cada vez tiene un mayor impacto en la vida cotidiana, pues los avances

de la tecnología en edición de imágenes ha llegado al alcance de muchas personas a través de un teléfono inteligente. En al área de visión computacional el PDI es una herramienta fundamental, pues muchos de los métodos requieren de un procesamiento de la imagen para resaltar la información que se busca y posteriormente aplicar otro tipo de técnicas a la imagen.

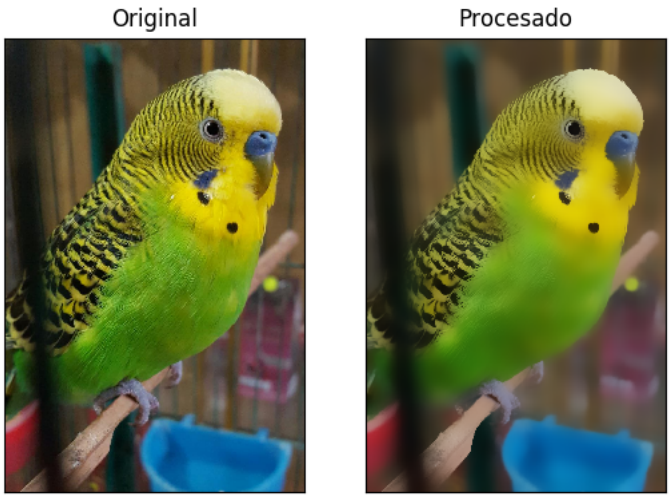


Figura 4.2: Izquierda, imagen original. Derecha, imagen después de aplicar un filtro bilateral (Imagen capturada y procesada para uso de este trabajo de tesis)

El procesamiento de imágenes se pude interpretar como una función que recibe una o mas imágenes (por ejemplo una imagen a color que cuenta con tres canales rojo, verde y azul) y como resultado produce una o más imágenes. En el dominio continuo esto se representa de la siguiente manera.

$$g(x) = h(f(x)) \tag{4.1}$$

o

$$g(x) = h(f_0(x), ..., f_n(x)) \tag{4.2}$$

Para el dominio discreto se tiene un número finito de píxeles por lo cual se tiene.

$$g(i, j) = h(f(i, j)) \tag{4.3}$$

Dependiendo del resultado deseado se debe de aplicar un procedimiento u otro,

por ejemplo uno de los procesamientos mas básicos son la multiplicación y adición de constantes.

$$g(x) = af(x) + b \tag{4.4}$$

donde a debe de ser mayor a cero, los parámetros a y b son usualmente llamados control de contraste y brillo respectivamente[8]. El anterior ejemplo es un caso de procesamiento por píxel, el cual, genera un nuevo valor dependiendo del valor anterior del mismo píxel con cierta modificación.

De esta manera se pueden hacer correcciones o sobresaltar la información relevante, no obstante hay técnicas mas complejas como los filtros lineales.

4.2.1. Filtros Lineales

Los filtros lineales son aquellos que modifican el valor de un píxel dependiendo del valor de sus vecinos, es decir, que implican combinaciones ponderadas de píxeles en regiones aledañas pequeñas.

$$g(i,j) = h(f(i,j)) \tag{4.5}$$

la función h asigna a cada píxel un valor que depende de un número de píxeles a su alrededor.

$$g(i,j) = \sum_{k=0}^{m}\sum_{l=0}^{n} f(i+k, j+l)h(k,l) \tag{4.6}$$

o mostrado en una notación mas breve.

$$g = f \otimes h \tag{4.7}$$

A esta operación se le conoce como convolución, donde h representa una matriz conocida como kernel o núcleo, cuyos elementos son conocidos como los coeficientes del filtro. Usualmente todos los valores de un núcleo son divididos por una constante para normalizar el píxel resultante. Este tipo de filtros son utilizados para difuminar

imágenes y también al contrario, para definir más la imagen, también se utilizan para resaltar bordes o esquinas en una imagen entre otros usos. En la figura (Fig. 4.3) se muestra el proceso de filtrado de una imagen.

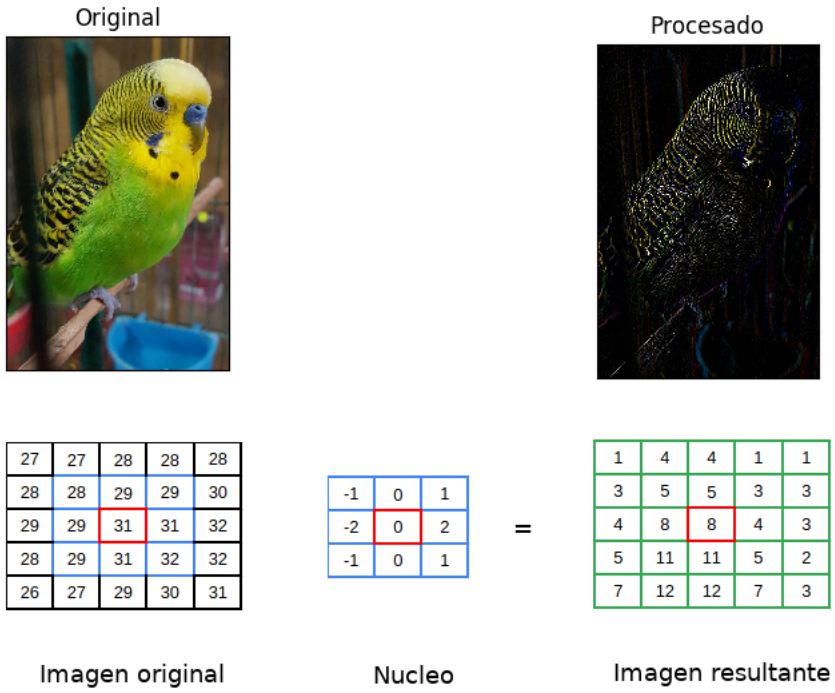


Figura 4.3: Convolución de la imagen de la izquierda por un núcleo detector de bordes dando como resultado la imagen de la derecha. El procedimiento para llegar al resultado es el siguiente : $(-1)(28)+(0)(29)+(1)(29)+(-2)(29)+(0)(31)+(2)(31)+(-1)(29)+(0)(31)+(1)(32) = 8$

4.2.2. Filtros separables

Cuando se realiza la convolución con núcleos cuadrados donde cualquiera de sus lados mide l el numero de operaciones realizadas es l, sin embargo existen ciertos núcleos que pueden ser separables en dos núcleos de menores dimensiones. Por ejemplo el núcleo mostrado en la figura (Fig: 4.4). puede ser dividido en un núcleo horizontal de una dimensión y otro filtro vertical de una dimensión. Y para obtener la misma imagen que un filtro no separado, únicamente se debe de efectuar la convolución con el primer filtro horizontal y posteriormente efectuar lo mismo con el filtro vertical. Con este cambio en el procesamiento, el resultado en la imagen es el mismo, sin embargo, el procedimiento es mas eficiente, pues únicamente se realizan $2l$ operaciones por

píxel en la imagen, en comparación con l operaciones por píxel en el caso de un filtro cuadrado.

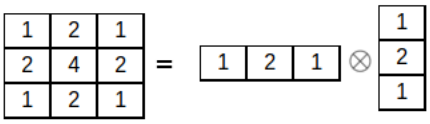


Figura 4.4: Separación de un núcleo en dos núcleos de menores dimensiones, lo cual resulta en una reducción de operaciones

4.2.3. Filtros no lineales

Utilizando filtros que no son lineales se pueden obtener mejores resultados en ciertos escenarios. En estos casos los núcleos con los que se convolucionará una imagen no son una combinación lineal de sus vecinos.

Por ejemplo el filtro bilateral funciona como un filtro difuminado, sin embargo, mantiene los bordes de la imagen. Esto debido a que rechaza los píxeles que son distantes en intensidad del píxel evaluado, esta es la idea principal del filtro bilateral.

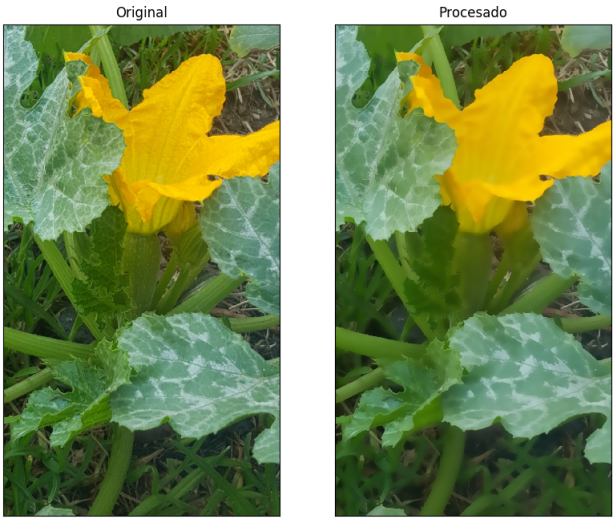


Figura 4.5: Imagen procesada con un filtro bilateral (Imagen capturada y procesada para uso de este trabajo de tesis)

Filtro bilateral

El filtro bilaterla se define de la siguiente manera:

$$g(i,j) = \frac{\Sigma_{k,l} f(k,l) w(i,j,k,l)}{\Sigma_{k,l} w(i,j,k,l)} \tag{4.8}$$

El coeficiente ponderado $w(i,j,k,l)$ depende del dominio del kernel que usualmente es una función gausiana.

$$d(i,j,k,l) = exp\left(-\frac{(i-k)^2 + (j-l)^2}{2{\sigma_d}^2}\right) \tag{4.9}$$

y también depende del el rango del núcleo.

$$r(i,j,k,l) = exp\left(-\frac{||f(i,j) - f(k,l)||^2}{2{\sigma_r}^2}\right) \tag{4.10}$$

multiplicando las ultimas dos expresiones se obtiene el filtro bilateral.

$$w(i,j,k,l) = exp\left(-\frac{||f(i,j) - f(k,l)||^2}{2{\sigma_r}^2} - \frac{(i-k)^2 + (j-l)^2}{2{\sigma_d}^2}\right) \tag{4.11}$$

De esta manera es como el filtro bilateral realiza la difuminación solo en los píxeles cuyo valor se encuentre en el rango definido por σ_r si este valor incrementa el resultado se aproximará a un filtro de difuminasion gaussiano, mientras que si incrementa σ_d se difuminaran mayores áreas en común.

4.2.4. Otros filtros no lineales

Filtrado en frecuencia

Como los ejemplos mencionados anteriormente también existen diversas técnicas de procesamiento de imágenes, por ejemplo el filtrado en frecuencia, donde una imagen se filtra en el domino de la frecuencia y en consecuencia es relativamente sencillo aplicar filtros paso bajas, paso altas o paso bandas. El inconveniente de trabajar en el dominio de la frecuencia es que se debe de realizar la transformada de Fourier, que a pesar de que en la actualidad existan algoritmos muy eficientes, es mas fácil en muchos casos aplicar un filtro convolucional, además de que un filtrado en frecuencia

puede generar artefactos con facilidad en la imagen resultante.

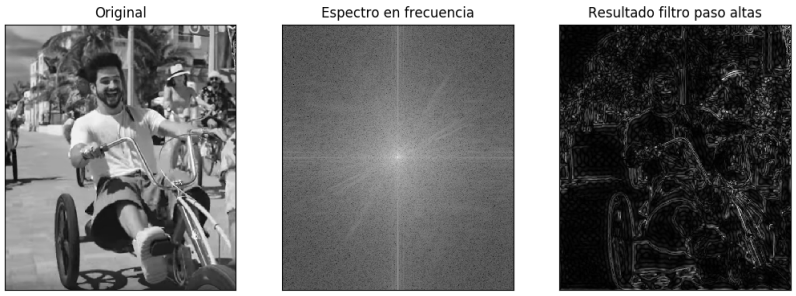


Figura 4.6: Filtro paso altas en frecuencia

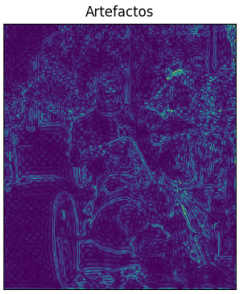


Figura 4.7: Artefactos producidos por el filtrado en frecuencia (Mapa de colores modificada para apreciar mejor los artefactos)

Operadores morfológico

A pesar de que muchas veces se trabaja con imágenes de color, hay casos en los que se prefiere trabajar con imágenes binarias, por ejemplo cuando se escanea un documento escrito a mano y se quiere digitalizar por un proceso automático, en dicho caso se pueden aplicar filtros para mejorar el trazo de las letras en el documento. Este tipo de operaciones tienen un umbral para obtener los resultados deseados, es decir, el umbral define el valor de cada píxel bajo ciertas normas. En imágenes binarias las operaciones mas comunes se llaman operaciones morfológicas, las cuales consisten en , convolucionar la imagen binaria con una estructura elemental binaria, dicha estructura puede ser de cualquier forma desde un caja de 3x3 pixeles hasta cualquiera mas compleja. Una función umbral se define como:

$$\theta(f,t) = \begin{cases} 1 & \text{si } f \geq t, \\ 0 & \text{en otro caso} \end{cases} \tag{4.12}$$

Los operadores morfológicos se definen a continuación.

Dilatación:

$$d(f,s) = \theta(c,1) \tag{4.13}$$

Erosión:

$$e(f,s) = \theta(c,S) \tag{4.14}$$

Cierre:

$$c(f,s) = e(d(f,s),s) \tag{4.15}$$

Apertura:

$$c(f,s) = d(e(f,s),s) \tag{4.16}$$

Donde $c = f \otimes s$ es un valor entero que cuenta el numero de unos dentro de cada estructura elemental conforme se va recorriendo la imagen y S es el tamaño de elementos de la estructura, es decir el número de píxeles.

Figura 4.8: Ejemplo de operadores morfológicos

Como es de esperar, los procesamientos de imágenes que se acaban de mencionar son solo algunos de una gran cantidad de métodos. Si se requiere profundizar en este tema se puede consultar en la literatura como (Lim, Jae S 1990)[9] (Bhabatosh et. al. 2011) [10].

4.3. Procesamiento de imágenes 3D

Como se vio en el capítulo 2 en la descripción de algunos de los sensores para robots móviles se mencionan las sensores que proporcionan información del ambiente en tres dimensiones espaciales (x, y, z), entre ellos se encuentran las cámaras 3D y las cámaras estereoscópicas. Ambos sensores proporcionan una imagen de color y además la posición (x, y, z) de cada uno de los píxeles en la imagen. Este tipo de información se utiliza comúnmente para escanear y tener una representación 3D de un espacio de interés. Además en la robótica, este tipo de sensores se utiliza para reconocer superficies con algoritmos como RANSAC [11], o también para calcular distancias a una superficie u objeto. Usualmente se le llama nube de puntos a la información espacial de cada pixel (Fig. 4.9).

Figura 4.9: Izquierda: Nube de puntos. Derecha:Imagen a color.

Existe una rama de investigación para el análisis y procesamiento de nube de puntos, algunas de estas ramas se enfocan en la segmentación o al reconocimiento de objetos y superficies en ellas. Un enfoque reciente para la segmentación de nubes de puntos, es el aprendizaje profundo que en años recientes se ha convertido en una herramienta muy fuerte en el análisis de imágenes en dos dimensione y en tres dimensiones, como la red neuronal artificial PointNet [12].

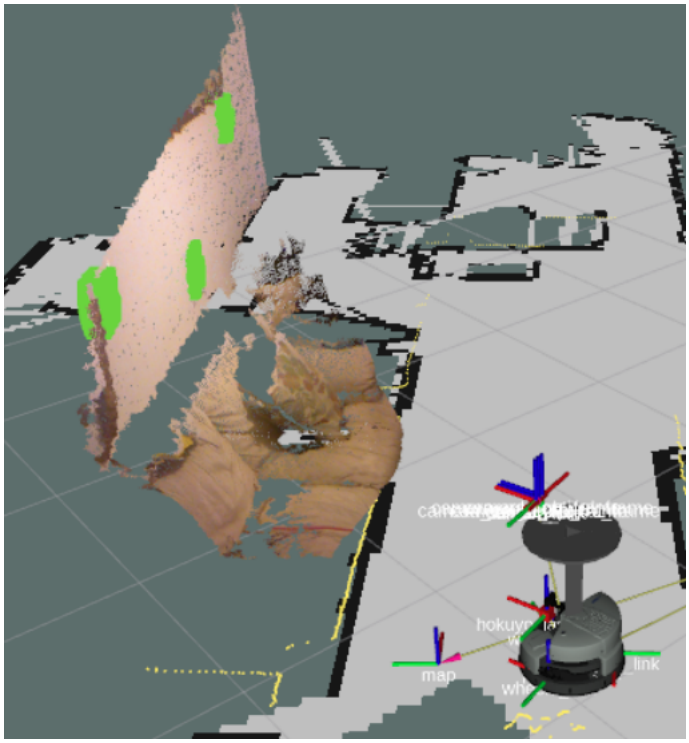


Figura 4.10: Robot móvil y la nube de puntos detectada por sus sensores, segmentando (en color verde) regiones de interés.

4.4. Homógrafa

El termino homógrafa en el campo de la visión computacional, se refiere a una transformación h entre dos imágenes , la primera de ellas A es tomada por una cámara con un punto focal C y la segunda imagen B por una cámara con punto focal C' ambas cámaras toman una fotografía sobre un plano π

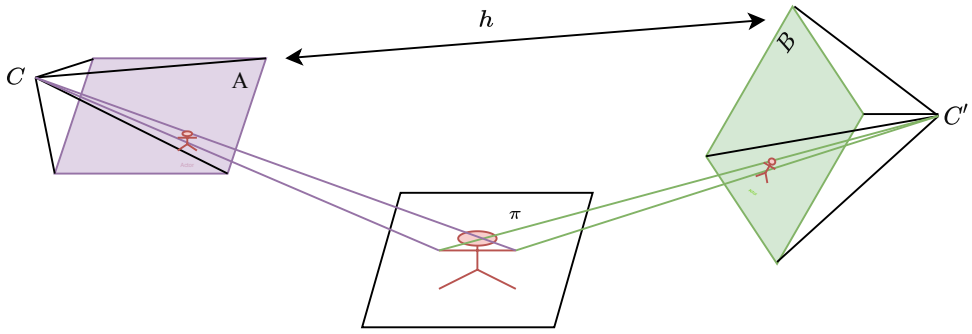


Figura 4.11

En la practica cualquiera de las dos cámaras o incluso las dos pueden ser virtuales, es decir la imagen no necesariamente debe de provenir de una cámara, sino que puede ser una imagen diseñada por computadora. El resultado en cualquier caso seria una transformación, tal que la imagen resultante tiene la misma perspectiva que otra imagen.

La transformación para un punto p en la imagen A en un punto p' en B se define como:

$$p' = h \times p \tag{4.17}$$

o bien de manera más explicita:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

(4.18)

La ecuación anterior proviene de una transformación de traslación, rotación y perspectiva para mayor información consultar [13] y [14].

Por ejemplo en la figura (FIg. 4.12) se muestran dos imágenes tomadas desde diferente posición, para mostrar la segunda imagen con la misma perspectiva que la primera se requiere calcular la matriz homógrafa que está definida por una matriz 3x3 que mapea todos los puntos de la segunda imagen en la primera.

Figura 4.12: Fotografías de la misma imagen con perspectivas diferentes, los puntos marcados son los mismos en cada imagen pero con coordenadas diferentes. (Imagenes capturadas para uso de este trabajo de tesis)

Un procedimiento para encontrar la relación de perspectiva entre dos imágenes es utilizando al menos 4 puntos de cada imagen, donde cada punto en la primera imagen sea el mismo en la segunda (Fig. 4.12). Por lo tanto para un punto se tiene:

$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}} \tag{4.19}$$

$$y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}} \tag{4.20}$$

Igualando a cero se tiene:

$$h_{11}x + h_{12}y + h_{13} - h_{31}xx' - h_{32}yx' + h_{33}x' = 0 \tag{4.21}$$

$$h_{21}x + h_{22}y + h_{23} - h_{31}xy' - h_{32}yy' + h_{33}y' = 0 \tag{4.22}$$

Exponiendo lo anterior en forma matricial, se tiene:

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x'_1 & -y_1x'_1 & x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y'_1 & -y_1y'_1 & y'_1 \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Como se mencionó, es necesario 4 puntos para efectuar este método, por lo tanto:

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x'_1 & -y_1x'_1 & x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y'_1 & -y_1y'_1 & y'_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2x'_2 & -y_2x'_2 & x'_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2y'_2 & -y_2y'_2 & y'_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3x'_3 & -y_3x'_3 & x'_3 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3y'_3 & -y_3y'_3 & y'_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4x'_4 & -y_4x'_4 & x'_4 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x_4y'_4 & -y_4y'_4 & y'_4 \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

exponiendo lo anterior de manera reducida:

$$A \times h = 0 \tag{4.23}$$

en caso de que se tengan más puntos se pueden agregar como renglones de la matriz A.

Agregando la matriz A^T en ambos lados de la ecuación, se tiene:

$$A^T A \times h = A^T \times 0 \tag{4.24}$$

$$A^T A \times h = 0 \tag{4.25}$$

La matriz h es igual a el eigenvector cuyo eigenvalor es el menor de todos, para 4 puntos, este debería de ser cero.

Ejemplo: Dando los puntos marcado en la figura (4.12) la imagen izquierda A y la derecha B se tiene:

	Puntos en A	Puntos en B
1	(541,1105)	(853, 1087)
2	(2580, 1167)	(2257, 1881)
3	(2643, 4338)	(2646, 4526)
4	(392, 4309)	(479, 4231)

Para dichas parejas de puntos se tiene la matriz h:

$$h = \begin{bmatrix} 1.01492503 & 1.34520296e^{-01} & -4.86057913e^{+02} \\ -9.69272965e^{-01} & 1.53330031 & 2.34243705e^{+02} \\ -1.97390806e^{-04} & 1.29215260e^{-04} & 1.00000000 \end{bmatrix} \tag{4.26}$$

Es importante mencionar que algunos métodos para la obtención de los eigenvalores y eigenvectores el resultado de la matriz h puede no estar normalizado, para lo cual, es necesario dividir cada elemento de la matriz h por el valor h_{33} el cual representa la escala de la imagen resultante. La imagen el resultante se muestra en la figura (Fig:4.13).

Figura 4.13: A) Imagen de referencia, B) Imagen tomada desde una perspectiva diferente a A, C)Resultado de procesar la imagen B con la matriz de homografía calculada a partir de la imagen A y B

Capítulo 5

Metodología propuesta

En este capitulo se describe la metodología propuesta para la localización de un robot móvil con el filtro de Kalman extendido y marcas en el ambiente.

El robot propuesto es un robot móvil con configuración de par diferencial el cual tiene su eje de referencia en el centro de rotación, el cual se encuentra en el punto medio entre las dos llantas donde el eje X se encuentra apuntando positivamente al frente del robot y los demás ejes siguen la regla de la mano derecha.

5.1. Etapa de predición EKF

La metodología que se propone en este documento se basa en el filtro de Kalman extendido para la localización de un robot móvil,el cual se mencionó a profundidad en la sección 3.4. En dicha sección se mencionó que el filtro de Kalman enfocado en la localización de un robot móvil, requiere de un comando de movimiento y el reconocimiento de marcas para su funcionamiento.

El comando u que se le dará al robot es un comando conformado por dos subcomandos, el primero de ellos indica un giro respecto al centro de rotación del robot, un giro positivo es en sentido antihorario y uno negativo en sentido horario. El segundo

subcomando indica una traslación sobre al eje X del robot.

$$\vec{u}_k = \begin{bmatrix} u_\theta \\ u_d \end{bmatrix} \tag{5.1}$$

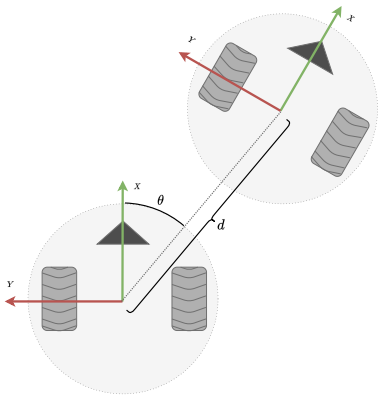


Figura 5.1

Dado que este sistema se encarga de localizar al robot, el vector de estado $\vec{x}$ estará compuesto por la pose del robot respecto a un punto fijo.

$$\vec{x}_k = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{5.2}$$

Por lo cual, en la etapa de predicción la matriz F, que describe de manera ideal la posición del robot después de ejecutar un comando, se define como:

$$g(\vec{x}_k, \vec{u}_k) = \begin{bmatrix} 1 & 0 & u_d * cos(\theta_k + u_\theta) \\ 0 & 1 & u_d * sin(\theta_k + u_\theta) \\ 0 & 0 & \theta_k + u_\theta \end{bmatrix} \tag{5.3}$$

La función F debe ser linealizada mediante su Jacobiano. Por lo tanto.

$$G = J_{g(\vec{x}_k,\vec{u}_k)} = \begin{bmatrix} 1 & 0 & -u_d * sin(\theta_k) \\ 0 & 1 & u_d * cos(\theta_k) \\ 0 & 0 & 1 \end{bmatrix} \tag{5.4}$$

Por lo tanto, el vector de estado se actualiza de la siguiente manera para la etapa de predicción:

$$\vec{x}_k = g(\vec{x}_{k-1}, \vec{u}_k) \tag{5.5}$$

.

y la matriz de covarianza:

$$\Sigma_k = G_{k-1}\Sigma_{k-1}G_{k-1}^T + Q \tag{5.6}$$

Donde la matriz Q representa las fuerzas externas al sistema, en otras palabras representan la incertidumbre de la posición final después de la ejecución de un comando.

5.2. Detección de marcas

Las marcas propuestas para este trabajo son códigos gráficos de dos dimensiones los cuales forman una matriz binaria cuadrada donde sus elementos son colores bien contrastados entre ellos, por ejemplo blanco y negro. Algunos códigos con estas características son: QR, Aruco, BIDI y Datamatrix.

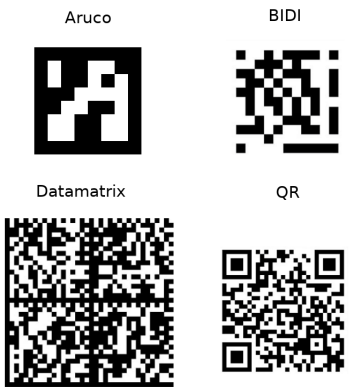


Figura 5.2: Diferentes tipos de códigos bidimensionales binarios

Los códigos bidimensionales binarios, son relativamente fáciles de detectar y de decodificar a partir de una imagen gracias a las técnicas de procesamiento digital de imágenes como se vio en la sección 4.2. Es importante mencionar que cada uno de los diferentes códigos tiene características que son ventajas o desventajas sobre los demás códigos bajo circunstancias diferentes.

Una vez que se detecta un código en una imagen 2D es necesario calcular la distancia y ángulo respecto al robot, el punto de referencia en el código sera su centro, para ello se requerirá de un sensor que nos permita realizar esa tarea como los mencionados en la sección 2.7

Una cámara RGB de la cual se conocen sus características intrínsecas, es decir la cámara se encuentra calibrada, y con los 4 puntos de las esquinas de los códigos, es posible resolver el problema *Perspective-n-Point* el cual determina la pose de la cámara respecto a puntos en el espacio, para mayor información consultar [15].

Con una cámara estereoscópica se puede calcular la distancia a una código segmentando la el área correspondiente al código en la imagen RGB sobre la nube de puntos, usualmente la nube de puntos se calcula respecto a uno de los lentes de la cámara (típicamente el lente izquierdo), por lo tanto la posición $< x, y, z >$ del código será respecto a dicha cámara. Una vez que se tengan los puntos de la nube que corresponden al código, se calcula su centroide haciendo un promedio de las coordenadas de todos los puntos. El método anterior es aplicable directamente a las cámaras RGB-D pues la información que proporcionan es la misma (imagen RGB y nube de puntos)

a pesar de que su funcionamiento sea diferente.

5.3. Etapa de actualización EKF

Después de la ejecución de un comando, el robot debe de revisar a su alrededor las marcas conocidas. El robot debe de tener conocimiento de las marcas en el ambiente, es decir, debe conocer previamente su posición $< x, y, z >$ para que en la etapa de actualización del EKF pueda comparar con las lecturas de sus sensores, que en este caso son las posiciones $< x, y, z >$ de los códigos reconocidos por alguno de los métodos mencionados en la sección 5.2. En consecuencia la etapa de actualización es idéntica a la mencionada en la sección 3.4, incluso el algoritmo propuesto del filtro de Kalman extendido es el mencionado en dicha sección.

5.4. Detalles de implementación

La implementación de el sistema propuesto tiene dos ramas importantes, la primera de ellas es referente al hardware y la segunda al software.

5.4.1. Hardware

El hardware principal utilizado en esta investigación consta de un robot omnidireccional (consultar sección 2.3) Robotino de la marca Festo. El robot móvil cuenta con 3 motores y 3 llantas omnidireccionales que permiten su movimiento en diferentes direcciones (Fig: 5.3), además cuenta con una computadora a bordo con un procesador intel core i5y 8 gb de memoria RAM.

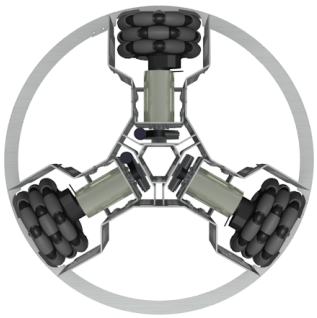


Figura 5.3: Configuración de motores y llantas en la base del robot de pruebas

Sobre la plataforma del robot se encuentra montado un sensor Kinect, el cual cuenta con un emisor y un receptor infrarrojo, el cual registra la profundidad de cada píxel en la imagen RGB.

Como unidad de procesamiento principal se utiliza una tarjeta de desarrollo Nvidia Jetson Tx2, en la cual se ejecuta el EKF y los programas necesarios de visión computacional para reconocer los códigos ArUco en el entorno (Fig. 5.4).

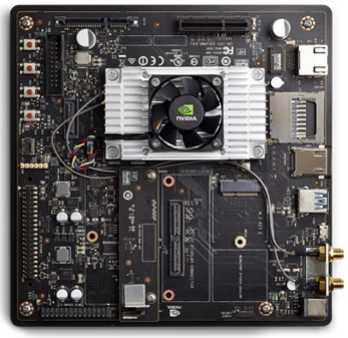


Figura 5.4: Unidad principal de procesamiento, Nvidia Jetson Tx2

También cuenta con un escáner laser 2D (consultar sección 2.5.2) el cual para objetivos de este trabajo no tiene mayor relevancia, sin embargo su utilidad en los robots móviles es muy importante, pues se pueden detectar obstáculos y crear mapas del entorno, sin embargo, estos dos temas están fuera del alcance de este trabajo.

Figura 5.5: Robot de pruebas

5.4.2. Software

El software principal utilizado en este trabajo es ROS (Robot Operating System) que es un framework de desarrollo de robots. ROS consta de librerías, herramientas y buenas practicas de desarrollo de software para robótica. Por las características de el framework ROS se eligió como parte fundamental del software de este proyecto, pues simplifica algunas tareas importantes, hace que el código sea modular y por tanto hasta cierto punto el software desarrollado es independiente del robot.

Cada uno de los sensores se encuentra posicionado en cierta parte del robot, por lo tanto es necesario tener un punto de referencia a partir del cual se encuentren posicionados sus sensores y actuadores. El centro de giro del robot se toma como su origen, es decir, el punto de intersección entre el eje de cada uno de sus motores y a partir de este punto se toma la posición de sus sensores como se muestra en la figura (Fig. 5.6).

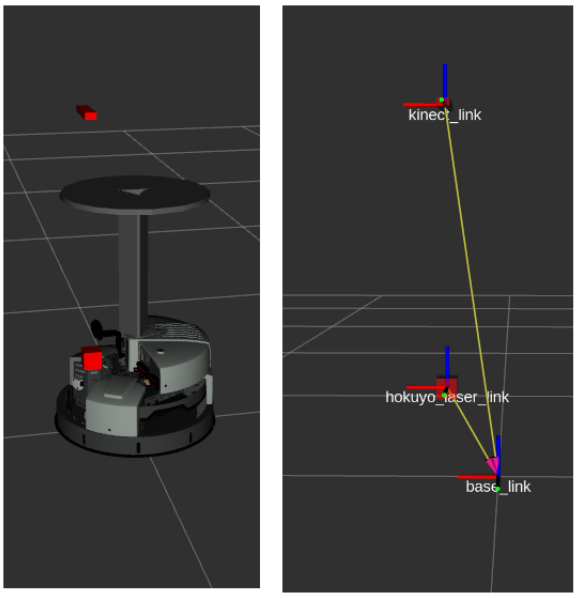


Figura 5.6: Lado izquierdo se muestran los ejes de referencia de los sensores respecto al origen del robot. Lado derecho muestra la reparcelación gráfica del robot y sus sensores

Los ejes de referencia de los sensores usualmente tienen como punto de referencia el origen del robot al igual que las partes moviles del robot como llantas o manipuladores. Por ejemplo si el sensor kinect detecta un objeto en las coordenadas $< x_o, y_o, z_o >$ respecto de el mismo, y se requiere saber que tan lejos se encuentra del origen del robot basta sumar la posición del eje del sensor respecto al origen del robot $< x_k, y_k, z_k >$, dando como resultado la posición del objeto respecto del origen $< x_o + x_k, y_o + y_k, z_o + z_k >$ esta característica es una de las herramientas que ROS proporciona, pues en ocasiones se requiere saber la posición de algún objeto respecto a un punto del robot. Este tipo de transformaciones de coordenadas es necesario para brazos robóticos, pues es importante saber la posición de sus articulaciones para recoger un objeto que se ha detectado con un sensor que tiene su propio eje de referencia.

De la misma forma que las transformación son necesarias entre los sensores del robot, en el tema de localización también se requiere un eje de referencia a partir del cual el robot estime su posición.

En primera instancia el robot se debe de localizar respecto a su punto inicial antes de ejecutar cualquier movimiento, esto se logra a través de la odometría (ver sección

2.8.1) que calcula la posición del robot a partir de los encoders (ver sección 2.6) en sus motores. La odometría es una parte fundamental de la etapa de predicción, pues a partir de la odometría se calcula el comando u al cabo de un cierto desplazamiento, giro o tiempo, tema que se abordará más adelante.

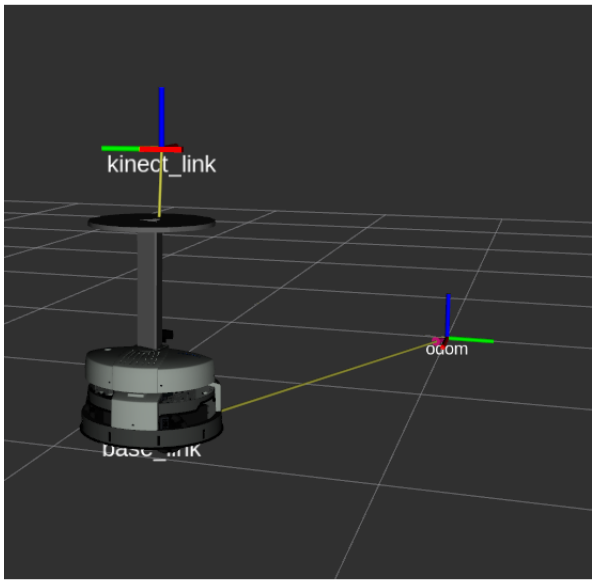


Figura 5.7: Transformación del origen de los movimientos (Odom) al eje del robot

El sistema de localización tiene que calcular el error de la odometría a lo largo del tiempo y hacer la transformación desde un punto en el ambiente del robot, al eje de la odometría. Si la odometría fuera perfecta no habría transformación entre el punto fijo en el ambiente y el eje de odometría, pero como en la práctica esto es imposible es necesario un sistema de localización que se encargue de ajustar dicha trasformación.

El punto fijo del ambiente debe de ser conocido de alguna manera por el robot, es por eso que se requiere de un mapa del entorno, dicho mapa puede contener cualquier tipo de información que sea útil para el robot. Para este trabajo el mapa necesario es el de las marcas en el ambiente de las cuales se hablara en breve, sin embargo para la navegación del robot es necesario un mapa que describa el espacio libre navegable para poder trazar rutas y evadir obstáculos, el cual puede ser fácilmente hecho con paquetes de ROS, un escáner láser 2D y un robot móvil. Un algoritmo muy famoso para construir mapas es SLAM (Simultaneous Localization and Mapping) si se quiere

profundizar en este tema se puede consultar [1, Pagina:337]

5.4.3. Reconocimiento de marcas

Para este trabajo los códigos utilizados son los ArUco [16] los cuales representan un índice único el cual representa un valor numérico. Los códigos ArUco son detectados y decodificados velozmente por una serie de pasos descritos a continuación.

En un principio con el sensor Kinect se buscan los posibles códigos en cada cuadro del vídeo.

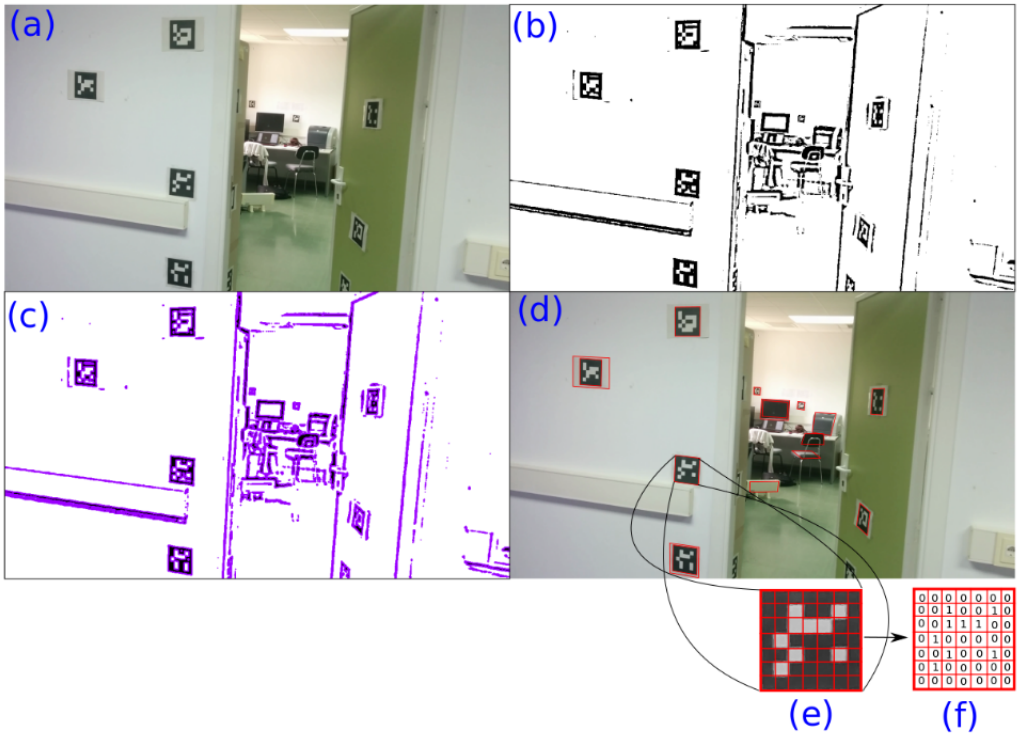


Figura 5.8: Reconocimiento e identificación de códigos ArUco a) Imagen original. b) Umbralización de la imagen por un filtro adaptativo. c) Extracción de contornos. d) Filtrado de contornos que se aproximan a un polígono de 4 vértices. e) Imagen transformada a su forma canónica f)Binarización de la imagen para su decodificación. Imagen tomada de [16]

.

En la imagen (Fig. 5.8) se muestra de manera resumida los pasos para el reconocimiento y la decodificación de los códigos ArUco, en estos pasos se requiere utilizar técnicas de visión computacional(sección 5). El primer paso para reconocer un código

ArUco es aplicar un umbral adaptativo a la imagen, para resaltar características de interés por medio de la binarización. El siguiente paso consiste en reconocer contornos en la imagen. Después se seleccionan aquellos contornos que se aproximan a un polígono de cuatro vértices. En el paso siguiente, cada polígono detectado se transforma en su forma canónica, es decir, como si el código fuera visto de frente a la cámara, esto se logra calculando la matriz de homografía (sección 4.4) para el polígono y la forma canónica. Finalmente se aplica la umbralización de Otsu [17] para binarizar la imagen y poder decodificarla. En la etapa de decodificación se descartan las áreas de la imagen detectadas como posibles códigos ArUco, en caso de ser un código valido, se guardan en memoria el identificador, y los cuatro puntos de las esquinas. Para una explicación mas detallada de este método consultar [16].

5.4.4. Posición de las marcas respecto al robot

Cuando se tiene el identificador de cada código, así como sus cuatro puntos que lo delimitan en la imagen original, se procede a calcular su posición respecto al sensor. Como se comentó anteriormente, las coordenadas que identifican las posición $< x, y, z >$ de un código es su punto central. Una vez identificados los códigos en la imagen RGB se procede a segmentar los puntos correspondientes a los códigos en la nube de puntos generada por el sensor, posteriormente por cada región segmentada se calcula un centroide haciendo un promedio de las coordenada de los puntos correspondientes. Para manipular las nubes de puntos se utilizo PCL(Point Cloud Library) una biblioteca de C++ para el procesamiento de nubes de puntos. Finalmente cada centroide representa la posición de cada marca en el espacio.

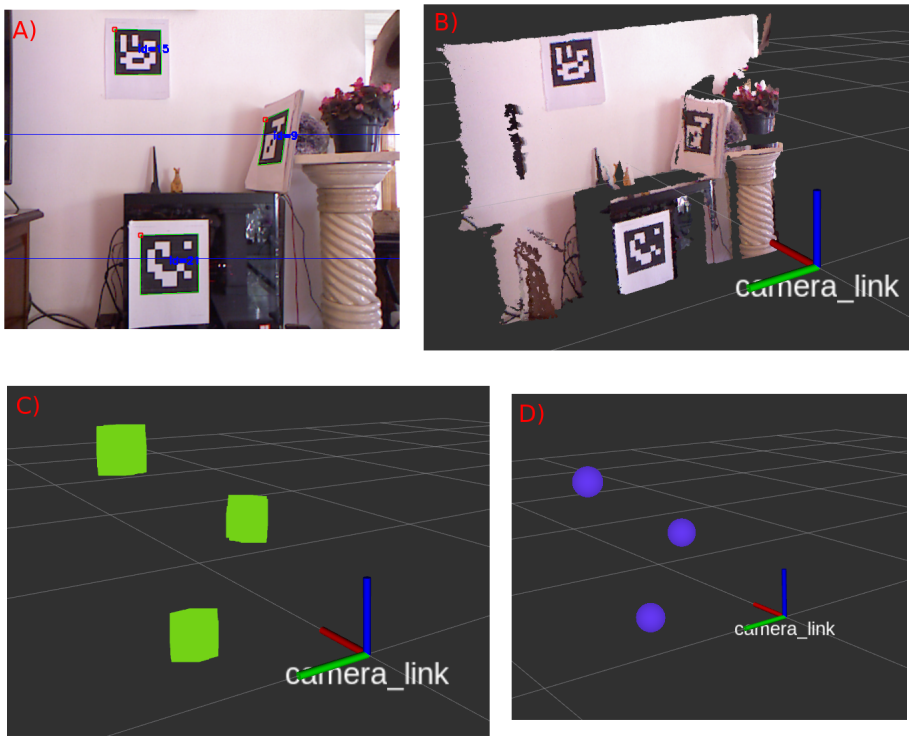


Figura 5.9: Cálculo de la posición de cada código. A)Imagen RGB original. B)Nube de puntos correspondiente a la imagen RGB C)Regiones segmentadas D)Centroides resultantes

Una vez obtenidos las posiciones de las marcas reconocidas se hace la transformación de esas posiciones respecto al eje del robot.

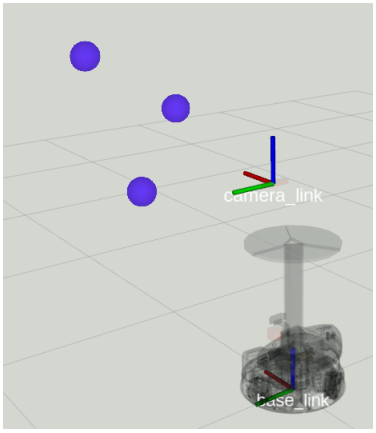


Figura 5.10: Cálculo de la posición de cada código respecto al eje de del robot.

Conocer el comportamiento de un sensor es importante para saber el rango de operación óptimo. El la figura (Fig. 5.11) se muestra la gráfica de error del sensor

Kinect al calcular la distancia a los landmarks. La gráfica que se muestra a continuación se obtuvo posicionando un código ArUco frente al sensor y comparando el valor reportado por el sensor y la medición un una cinta métrica, el proceso se realizo para diferentes distancias entre el sensor y el landmark.

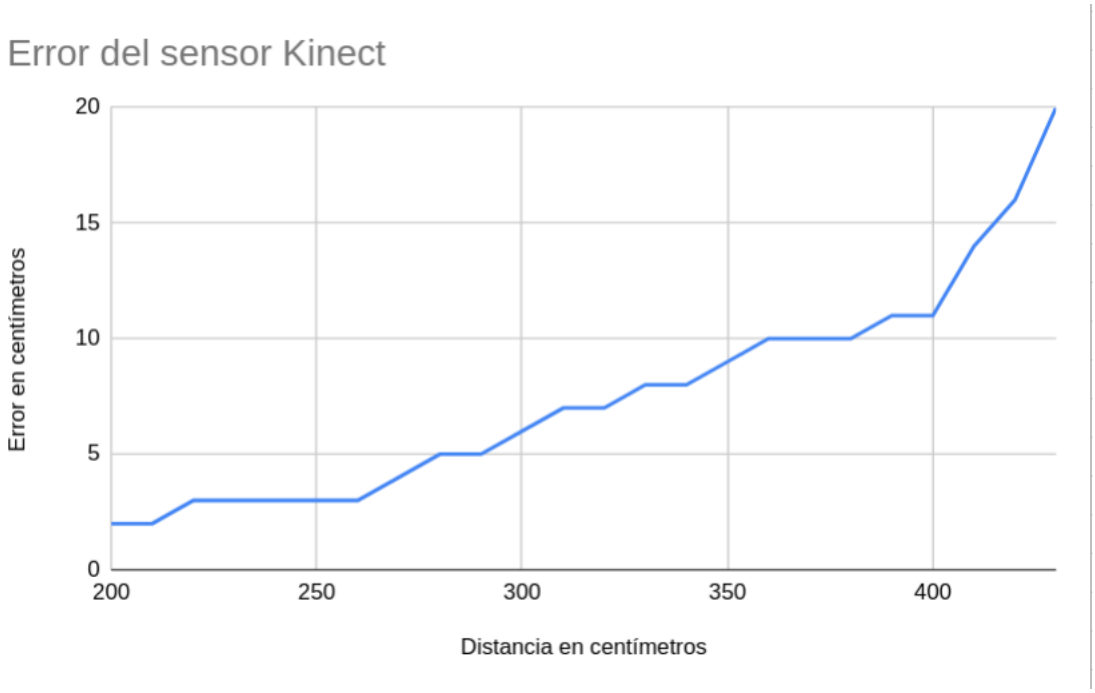


Figura 5.11: Error del sensor Kinect al calcular la posición de un código.

El EKF de debe ser capas de estimar la posición a partir de las lecturas y los movimientos del robot a pesar del error inherente de sus sensores.

5.4.5. Representación del ambiente

Como se mencionó en secciones anteriores, para que el robot pueda navegar y localizarse por medio del EKF, es necesario un mapa que contenga información del área navegable y de los landmarks en el ambiente.

Respecto al mapa del área navegable, en este trabajo se utilizó el algoritmo SLAM(Simultaneous Localization and Mapping) proporcionado por los paquetes de software de ROS [1, Pagina:337]. En la figura (fig:5.13) se muestra el mapa resultante, en el cual se puede apreciar que es una representación discreta del área navegable

donde la unidad mínima es un cuadrado de 5cm de lado, dependiendo del color se identifica cada cuadrado como un área ocupada (negro), libre (blanco) o se desconoce (gris).

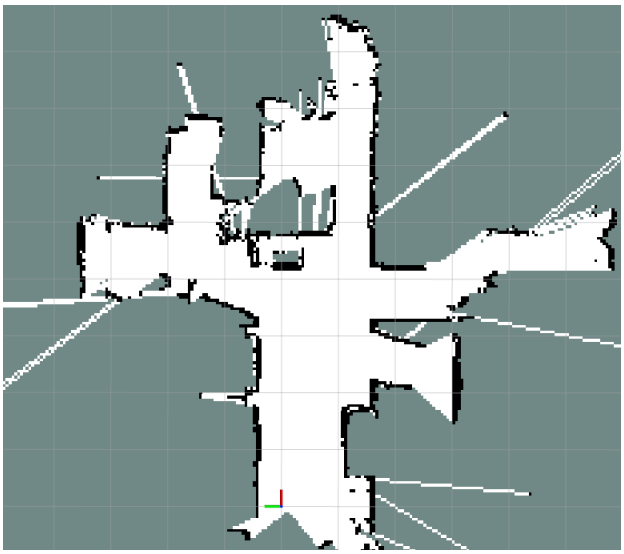


Figura 5.12: Mapa generado por el algoritmo SLAM

A partir del mapa generado con el algoritmo SLAM, se utilizó el sistema de localización AMCL(Adapatative Monte Carlo localization) [5] para mover el robot por el área navegable y registrar la posición de las marcas respecto al mapa.

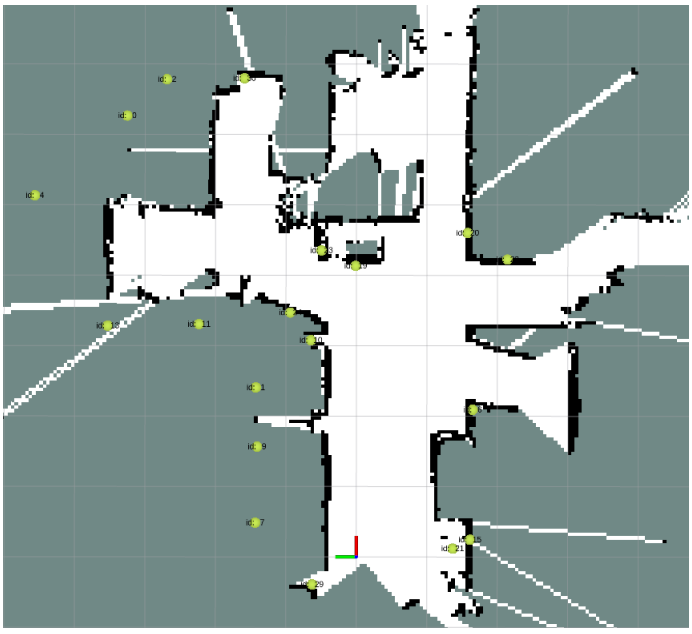


Figura 5.13: Mapa de posiciones de los códigos ArUco en el ambiente

Aunque el mapa de posiciones también se puede hacer de manera manual midiendo la posición por medio de una cinta métrica o alguna otra herramienta de medición.

5.4.6. Planeador de rutas

Para que el robot pueda navegar por su entorno es necesario un sistema de navegación, que planee una ruta entre dos puntos del mapa y otro sistema que se encargue de que el robot siga la ruta planeada.

El procedimiento que se utiliza en este trabajo para calcular rutas entre dos puntos, consiste en marcar como ocupadas las celdas en las cuales si el robot se posicionara colisionaría con algún objeto. Las celdas con dicha característica son las que se encuentran a una distancia r (igual al radio del robot) de una celda ocupada, dando como resultado un mapa con menos celdas libres.

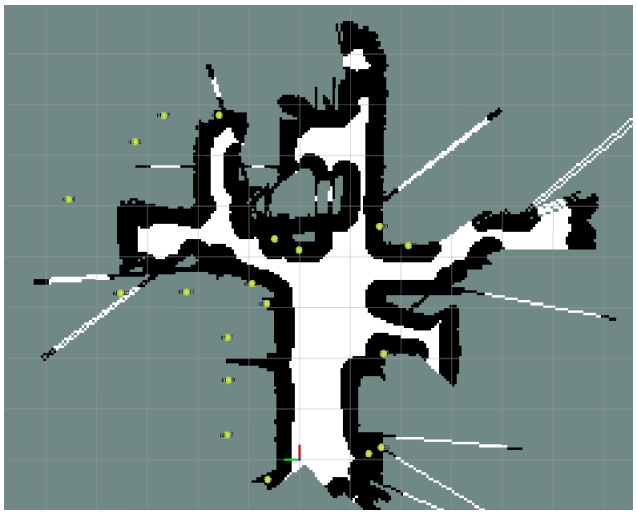


Figura 5.14: Mapa con celdas ocupadas aumentadas para calcular rutas sin colisiones.

Para calcular una ruta entre dos puntos se utiliza el algoritmo de Dijkstra para encontrar la ruta más corta entre dos puntos, sin embargo también es posible utilizar otros algoritmos de búsqueda, por ejemplo, A estrella (A*) en el cual se puede utilizar una función heurística, por ejemplo encontrar un camino en una ciudad donde lo que se busca es minimizar el tiempo y no la distancia.

Los nodos para el algoritmo de Dijkstra corresponden a cada una de las celdas del espacio libre y se encuentran conectadas por sus vecinos, es decir, cada uno de los

nodos tiene a lo más 8 conexiones y el peso de de cada transición es la distancia en centímetros.

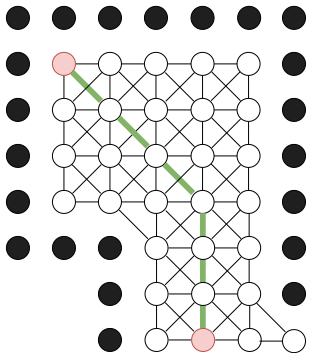


Figura 5.15: Conexiones entre los nodos navegables, cada nodo representa una celda. Linea verde representa la ruta calculada por el algoritmo de Dijkstra

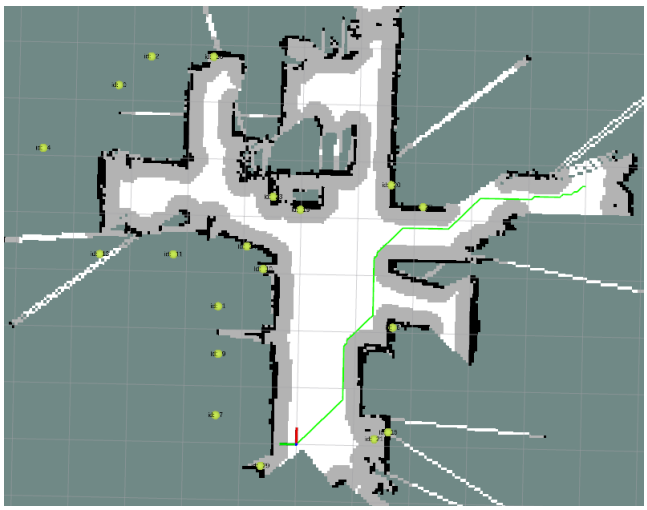


Figura 5.16: Ruta calculada por el algoritmo de Dijkstra sobre el mapa.

5.4.7. Navegación

Para este trabajo el sistema de navegación consiste en movimientos lineales sobre el eje x y angulares sobre el eje z, los movimientos son especificados por medio de velocidades. En la computadora integrada del robot, un sistema interno se encarga de mover cada uno de sus motores a la velocidad adecuada, para que el robot se mueva a las velocidades indicada.

Las velocidades tanto lineales y angulares son calculadas por un control proporcional básico, pero funcional para las pruebas realizadas en el presente trabajo.

$$v_x = \sqrt{(r_x - o_x)^2 + (r_y - o_y)^2} \tag{5.7}$$

$$v_\theta = 1.3 * atan2(r_y - o_y, r_x - o_x) + r_\theta \tag{5.8}$$

Donde r_x es la posición del robot en el eje x, r_y es la posición del robot en el eje y, o_x es la posición del objetivo en el eje x, o_y es la posición del objetivo en el eje y. A primera vista estos valores podrían parecer exagerados, incluso velocidades inalcanzables para un robot, sin embargo su efectividad se basa en que el punto objetivo nunca estará a un distancia mayor a 20 cm, pues las rutas se calculan desde la posición del robot al punto objetivo y el robot sigue la ruta teniendo como objetivo un punto cercano a él, dicho punto va cambiando conforme el robot avanza sobre la ruta planeada. A continuación, se muestra el algoritmo para seguir una ruta.

Algorithm 4 Seguidor de rutas

1: $P \leftarrow Ruta$
2: $i \leftarrow 0$
3: **while** $i \neq P.size$ **do**
4: $\quad d \leftarrow \sqrt{(r_x - P[i]_x)^2 + (r_y - P[i]_y)^2}$
5: $\quad$ **if** $d > corte$ **then**
6: $\quad\quad v_x \leftarrow prop_x * \sqrt{(r_x - P[i]_x)^2 + (r_y - P[i]_y)^2}$
7: $\quad\quad v_\theta \leftarrow prop_t * atan2(r_y - P[i]_y, r_x - P[i]_x) + r_\theta$
8: $\quad$ **else**
9: $\quad\quad$ i = i + 1
10: $\quad$ **end if**
11: **end while**
12: $v_\theta \leftarrow 0$
13: $v_x \leftarrow 0$

El valor de *corte*, $prop_x$ y $prop_t$ para los experimentos se estableció de manera experimental. La variable *corte* se estableció con un valor de 0.1m en la condicional de la linea 5 ya que las velocidad lineal sería muy pequeña si este valor fuera menor y en consecuencia, el robot se moverá muy lento. El valor de $prop_x$ es igual a 1 y el

valor de $prop_t$ es 1.3.

constante 1.3 en la linea 7 se determino de manera experimental, del mismo modo el valor .

5.5. Muestreo de la odometría

Como se mencionó, el EKF requiere de un comando u, para la etapa de predicción, sin embargo no es óptimo que un robot se mueva por comandos de giro y de avance ya que si el robot tiene que seguir una trayectoria curva esto no sería posible. Por tal motivo en este trabajo se utilizó una función para que a partir de dos poses se calcule el comando que el robot debió haber ejecutado de acuerdo a los comando de giro y de avance. Para mayor información sobre esta función, consultar [1, Pag:136]. La función consiste en muestrear la pose del robot como si se tratara de la ejecución de comandos de giro y de avance.

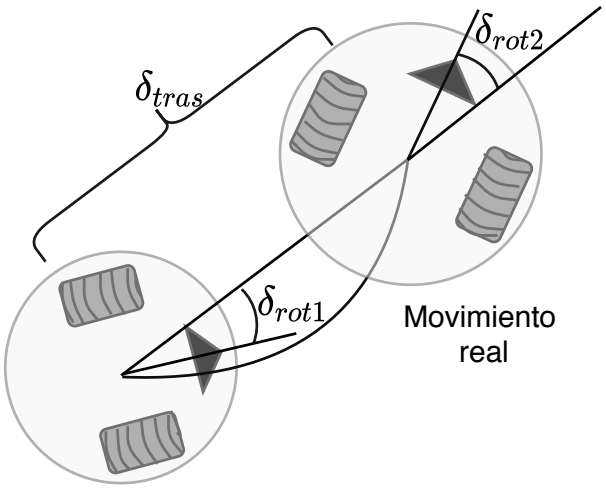


Figura 5.17: Muestreo de los movimientos del robot, el robot sigue una trayectoria curva, la cual puede ser aproximada a una rotación seguida de una traslación y otra rotación

En el algoritmo 5 se recibe como parámetro el comando u_t el cual esta conformado

Algorithm 5 Muestreo odometría

procedure MUESTREOODOMETRIA(u_t, x_{t-1})
  $\delta_{rot1} \leftarrow atan2(\bar{y}' - \bar{y}, \bar{x}' - \bar{x}) - \bar{\theta}$
  $\delta_{tras} \leftarrow \sqrt{(\bar{x} - \bar{x}')^2 + (\bar{y} - \bar{y}')^2}$
  $\delta_{rot2} \leftarrow \bar{\theta}' - \bar{\theta} - \delta_{rot1}$

  $\hat{\delta}_{rot1} \leftarrow \delta_{rot1} - sample(\alpha_1\delta^2_{rot1} + \alpha_2\delta^2_{tras})$
  $\hat{\delta}_{tras} \leftarrow \delta_{tras} - sample(\alpha_3\delta^2_{tras} + \alpha_4\delta^2_{rot1} + \alpha_4\delta^2_{rot2})$
  $\hat{\delta}_{rot2} \leftarrow \delta_{rot2} - sample(\alpha_1\delta^2_{rot2} + \alpha_2\delta^2_{tras})$

  $x' \leftarrow x + \hat{\delta}_{tras} cos(\theta + \hat{\delta}_{rot1})$
  $y' \leftarrow y + \hat{\delta}_{tras} sin(\theta + \hat{\delta}_{rot1})$
  $\theta' \leftarrow \theta + \hat{\delta}_{rot1} + \hat{\delta}_{rot2})$
  return $x_t \leftarrow (x', y', \theta')^T$
end procedure

por dos poses reportadas por la odometría, $u_t = (\bar{x}_{t-1} \quad \bar{x}_t)^T$, con $\bar{x}_{t-1} = (\bar{x} \quad \bar{y} \quad \bar{\theta})$ y $\bar{x}_t = (\bar{x}' \quad \bar{y}' \quad \bar{\theta}')$. El segundo parámetro representa la ultima estimación de la posición $x_{t-1} = (x \quad y \quad \theta)^T$. La función sample en el algoritmo representa una función de densidad de probabilidad $sample(b^2)$ con varianza b^2, dicha función de probabilidad puede ser una función normal, o alguna otra que modele el comportamiento del robot. La constante α_1 especifica el ruido esperado en la estimación de rotación de la odometría, a partir del componente de rotación del movimiento del robot. La constante α_2 especifica el ruido esperado en la estimación de rotación de la odometría, a partir del componente de traslación del movimiento del robot, α_3 especifica el ruido esperado en la estimación de traslación de la odometría, a partir de la componente de traslación del movimiento del robot, y α_4 especifica el ruido esperado en la estimación de traslación de la odometría, a partir de la componente de rotación del movimiento del robot.

Como el sistema funciona por muestreo de la odometría y no por medio de comandos, es necesario tener una condición para que se ejecute una iteración del EKF, dicha condición es una distancia recorrida o un giro determinado, es decir, el sistema hará una iteración cuando el robot se haya desplazado una distancia d o haya rotado un ángulo θ. Para el robot de pruebas estas constantes son $d = 0.2m$ y $\theta = 0.174533rad$.

5.6. Filtro de Kalman Extendido para el sistema propuesto

Por lo mencionado en los capítulos y secciones anteriores, el sistema propuesto en esta investigación se muestra en el algoritmo 6.

Algorithm 6 Filtro de Kalman Extendido para el sistema propuesto

1: **procedure** EKF($\vec{x}_{k-1}, \Sigma_{k-1}, \vec{u}, S$)
2: Predicción
3: $\vec{x}_k \leftarrow muestreoOdometria(\mathrm{u}_k, x_{k-1})$
4: $\mathrm{G} \leftarrow \begin{bmatrix} 1 & 0 & -u_d * sin(\theta_k) \\ 0 & 1 & u_d * cos(\theta_k) \\ 0 & 0 & 1 \end{bmatrix}$
5: $\Sigma_k \leftarrow G\Sigma_{k-1}G^T + Q$
6: Actualización
7: **for** $(i = 0 \ ; \ i < n \ ; \ i++)$ **do**
8: $\hat{\vec{z}} \leftarrow \begin{bmatrix} \sqrt{(S_{i,x} - x_k)^2 + (S_{i,y} - y_k)^2} \\ atan2(S_{i,y} - y_k, S_{i,x} - x_k) - \theta_k \\ S_{i,m} \end{bmatrix}$
9: $\vec{z} \leftarrow \begin{bmatrix} S_{i,r} \\ S_{i,\phi} \\ S_{i,m} \end{bmatrix}$
10: $H \leftarrow \begin{bmatrix} \frac{-(S_{i,x} - x_k)}{\sqrt{(S_{i,x} - x_k)^2 + (S_{i,y} - y_k)^2}} & \frac{-(S_{i,y} - y_k)}{\sqrt{(S_{i,x} - x_k)^2 + (S_{i,y} - y_k)^2}} & 0 \\ \frac{(S_{i,y} - y_k)}{(S_{i,x} - x_k)^2 + (S_{i,y} - y_k)^2} & \frac{-(S_{i,x} - x_k)}{(S_{i,x} - x_k)^2 + (S_{i,y} - y_k)^2} & -1 \\ 0 & 0 & 0 \end{bmatrix}$
11: $L \leftarrow (H * \Sigma * H^T) + R$
12: $K \leftarrow \Sigma * H^T * L^{-1}$
13: $V \leftarrow \vec{z} - \hat{\vec{z}}$
14: $\vec{x}_k \leftarrow \vec{x}_k + (K * V)$
15: $\Sigma_k \leftarrow (I - K * H) * \Sigma_k$
16: **end for**
17: **return** $(\vec{x}_k, \Sigma_k)$
18: **end procedure**

Es importante mencionar que en la linea 7 del algoritmo anterior el valor de n reprecenta el numero de renglones en la matriz S, la cual es de tamaño nx5. En la figura (Fig. 5.18), se muestra un diagrama de la metodología propuesta:

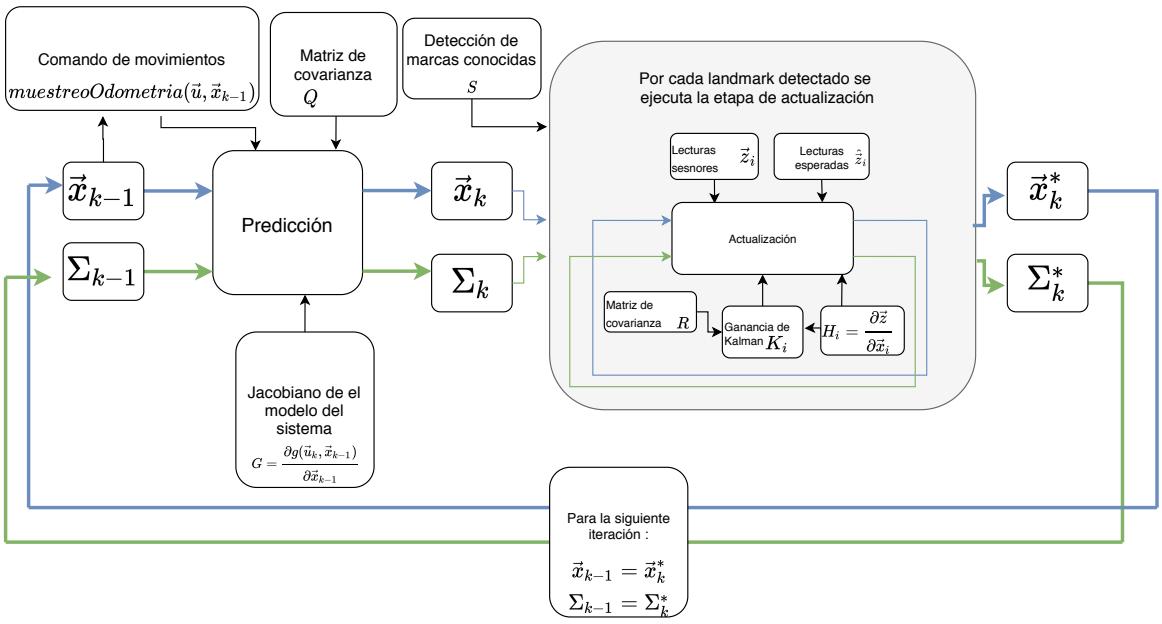


Figura 5.18: Diagrama del método propuesto.

Cuando el sistema no reconoce ninguna marca en el ambiente y se cumplen las condiciones para ejecutar el EKF, entonces únicamente se ejecuta la etapa de predicción.

Capítulo 6

Pruebas y resultados

Las pruebas del método propuesto tienen como finalidad evaluar su desempeño en diferentes situaciones. A continuación se describen las características de las pruebas y gráficas de resultados. Las constantes utilizadas la experimentación se muestran a continuación las cuales. Los valores alfa se recomiendan como valores estándar en la literatura, las matrices Q y R fueron obtenidas de manera experimental partiendo del calculo teórico.

Constante	Valor
α_1	0.2
α_2	0.2
α_3	0.2
α_4	0.2
Q	$\begin{bmatrix} 0.0025 & 0 & 0 \\ 0 & 0.0025 & 0 \\ 0 & 0 & 0.2 \end{bmatrix}$
R	$\begin{bmatrix} 0.0009 & 0 & 0 \\ 0 & 0.0009 & 0 \\ 0 & 0 & 0.000001 \end{bmatrix}$

En la figura (Fig 6.1) se muestran las marcas dispuestas en el entorno del robot, los códigos fueron dispuestos a lo largo de las paredes de manera aleatoria.

Figura 6.1: Código ArUco dispues en el entorno

6.1. Primera prueba

Para la etapa de pruebas se realizó un mapa con 22 códigos ArUco dispuestos en el ambiente(Fig: 6.2).

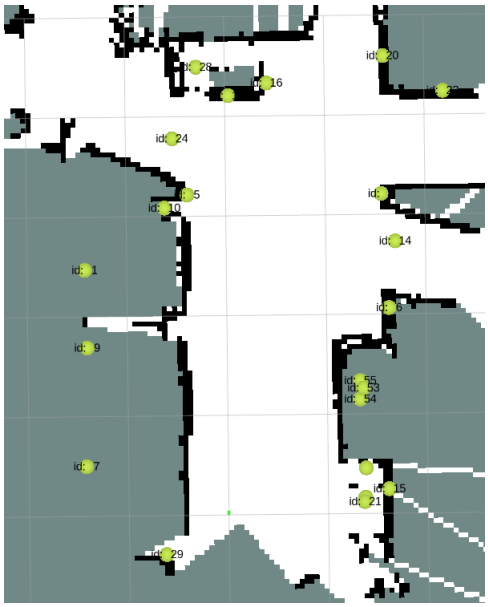


Figura 6.2: Posición de 22 códigos ArUco dispuestos en el entorno del robot

Se muestreo cada dos segundos la pose reportadas por la odometría del robot y la

pose calculada por el EKF y con esa información se obtuvieron los siguientes datos.

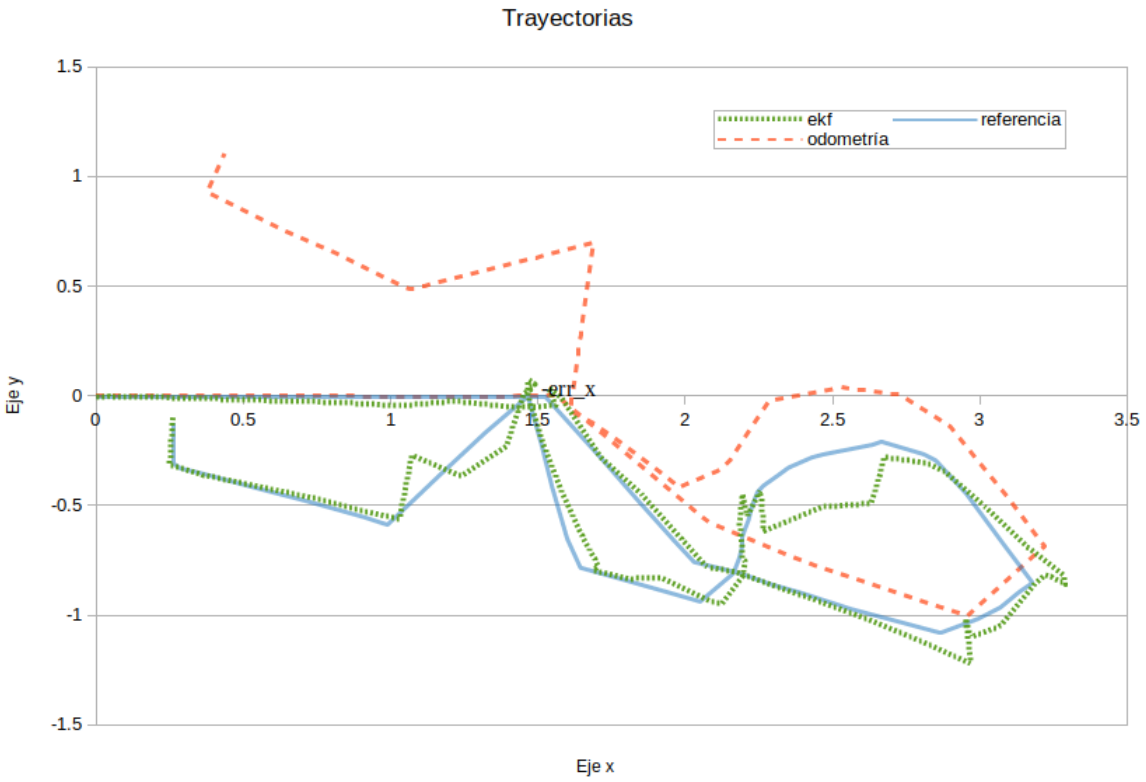


Figura 6.3: Se muestran tres trayectorias, azul trayectoria de referencia, rojo punteado odometría, verde EKF.

En la figura (Fig: 6.3) se muestra las trayectorias descritas por la odometría reportada por el robot, por el método propuesto y además la ruta de referencia, que representa la trayectoria real del robot la cual se obtubo por medio de mediciones manuales. Como se puede apreciar, el robot tiene un error significativo en el ángulo sobre el eje z reportado por la odometría, lo que da como resultado que dicha trayectoria vaya divergiendo cada vez más de la ruta de referencia. La ruta obtenida por el método propuesto a primera vista parece corregir los errores acumulados por la odometría, sin embargo, la trayectoria muestra puntos donde difiere considerablemente de la ruta de referencia. A continuación se muestra un análisis más detallado.

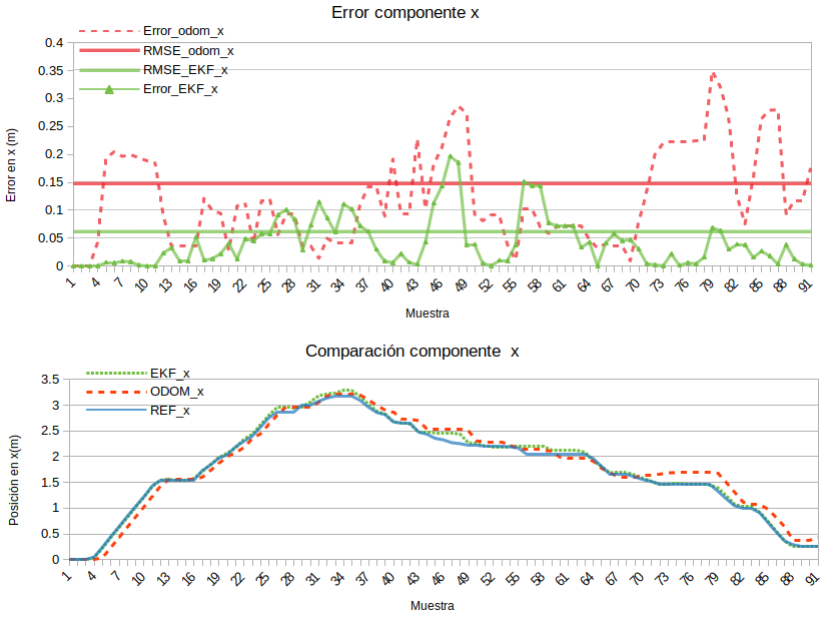


Figura 6.4: La gráfica superior muestra el error sobre el eje x respecto a la referencia. La gráfica inferior es comparativa entre la componente x de la odometría, el método propuesto y la referencia.

En la figura (Fig: 6.4) se muestran las diferencias tanto de la odometría como del método propuesto para la componente x respecto a la referencia. Calculando la raíz cuadrada del error cuadrático medio se puede observar que el error es menor para el método propuesto, que utilizando únicamente la odometría como método de localización.

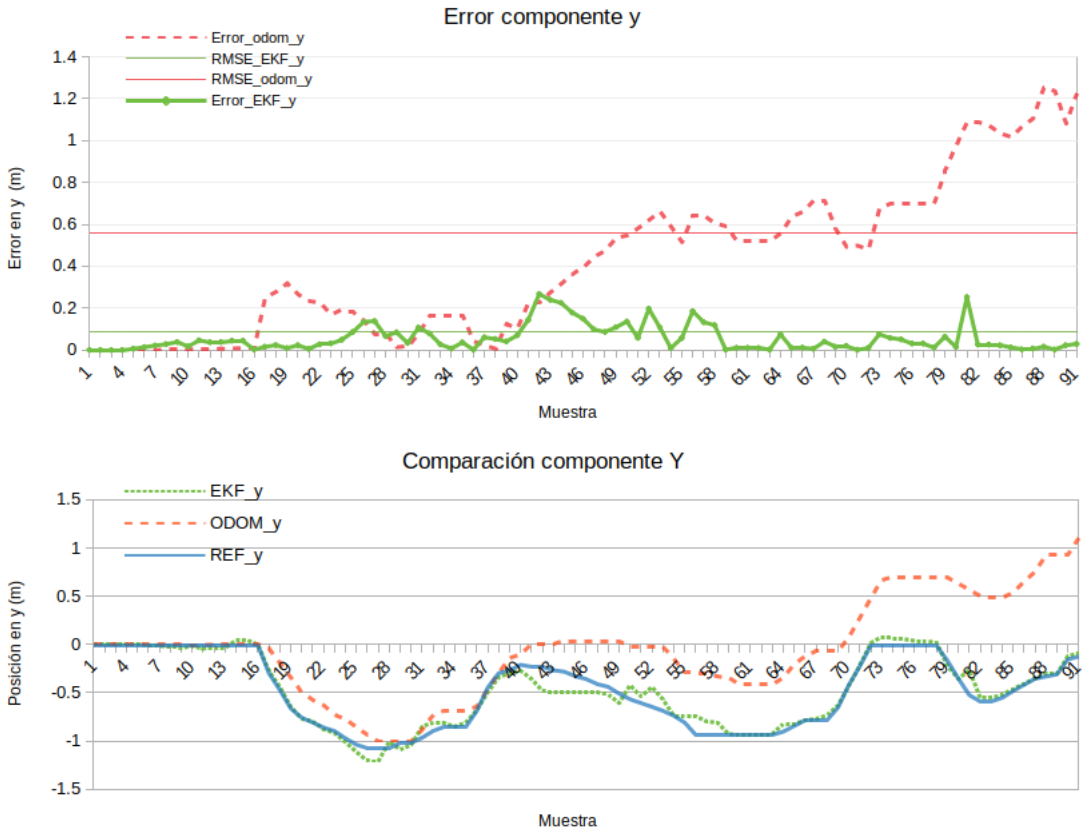


Figura 6.5: La gráfica superior muestra el error sobre el eje y respecto a la referencia. La gráfica inferior es comparativa entre la componente y de la odometría, el método propuesto y la referencia.

En la figura (Fig: 6.5) se muestran las diferencias tanto de la odometría como del método propuesto para la componente y respecto a la referencia. Calculando la raíz cuadrada del error cuadrático medio se puede observar que el error es menor para el método propuesto, que utilizando únicamente la odometría como método de localización.

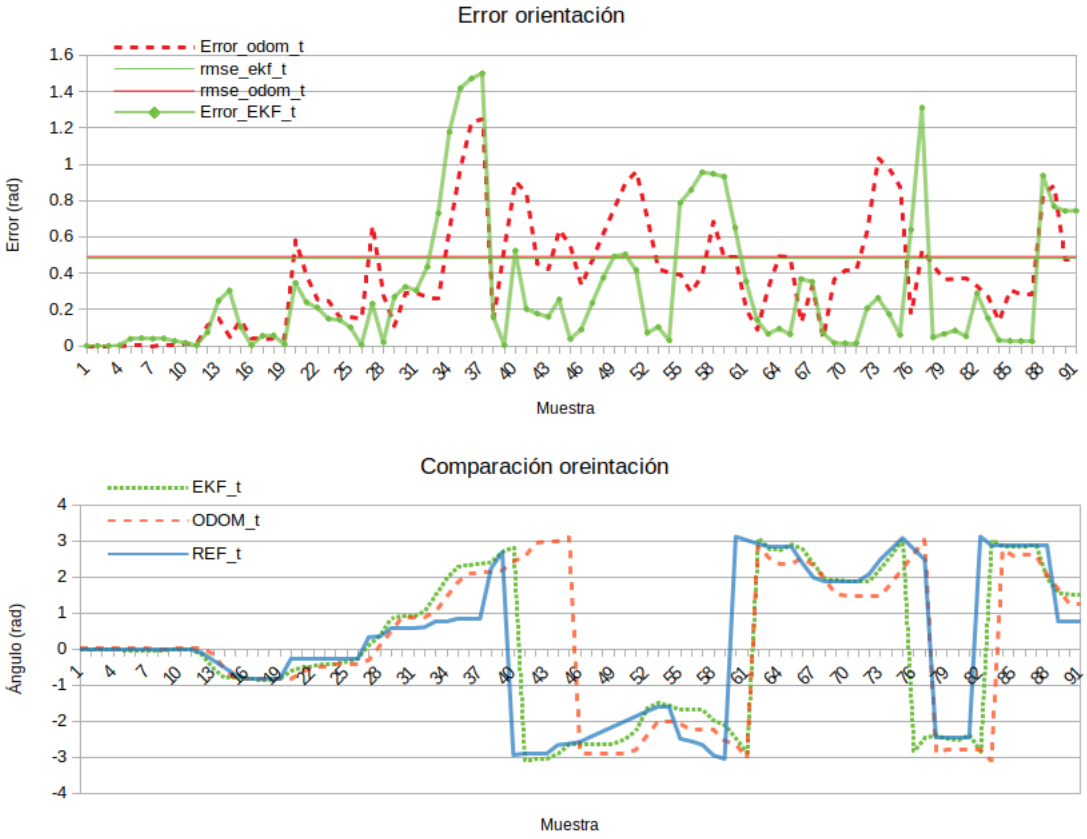


Figura 6.6: La gráfica superior muestra el error en el ángulo de orientación del robot respecto a la referencia. La gráfica inferior es comparativa entre la componente x de la odometría, el método propuesto y la referencia.

En la figura (Fig: 6.6) se muestran las diferencias tanto de la odometría como del método propuesto para la componente θ respecto a la referencia. Calculando la raíz cuadrada del error cuadrático medio se puede observar que el error es menor para el método propuesto, que utilizando únicamente la odometría como método de localización. Los saltos entre valores positivo y negativos se deben a que la orientación del robot se mide entre $[\pi, -\pi]$.

6.2. Segunda prueba

En esta prueba se redujo la cantidad de marcas en el ambiente a 13.

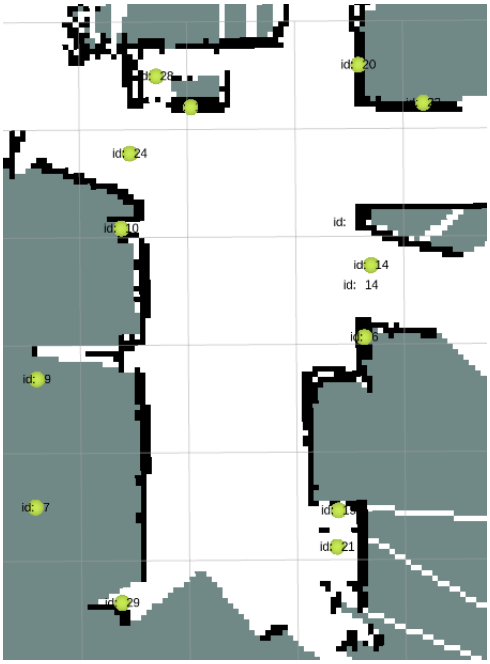


Figura 6.7: Posición de los códigos ArUco dispuestos en el entorno del robot

Al igual que en la primera prueba muestreo cada dos segundos la pose reportadas por la odometría del robot y la pose calculada por el EKF y con esa información se obtuvieron los siguientes datos.

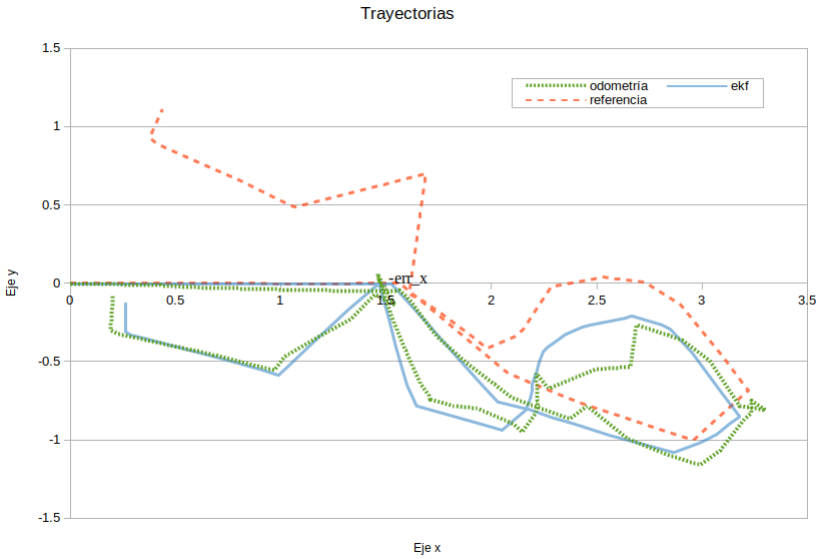


Figura 6.8: Se muestran tres trayectorias, azul trayectoria de referencia, rojo punteado odometría, verde EKF.

Como se puede apreciar en la figura (Fig: 6.13) a pesar de que el número de marcas se disminuyo de 22 a 13 el resultado obtenido es semejante a la prueba anterior, a continuación se muestra el análisis de resultados a detalle.

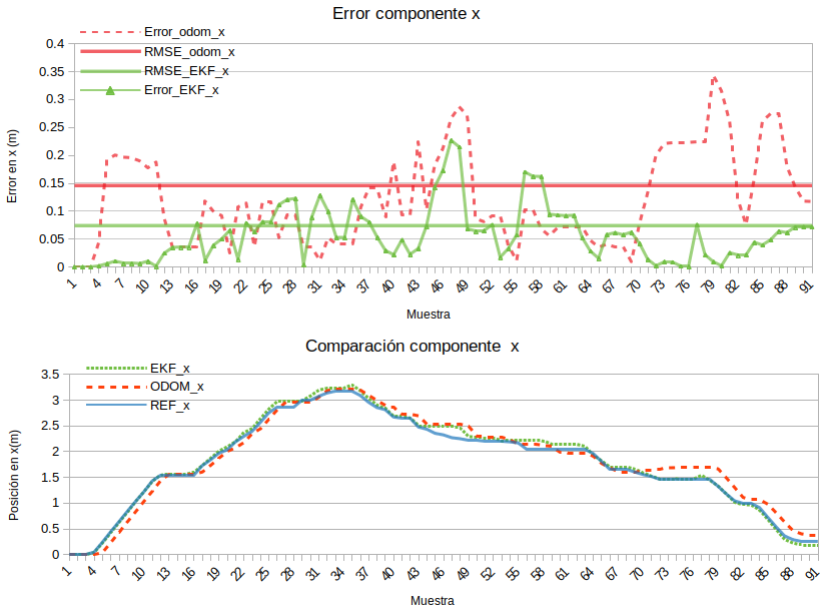


Figura 6.9: La gráfica superior muestra el error sobre el eje x respecto a la referencia. La gráfica inferior es comparativa entre la componente x de la odometría, el método propuesto y la referencia, para esta prueba se colocaron 13 marcas en el ambiente.

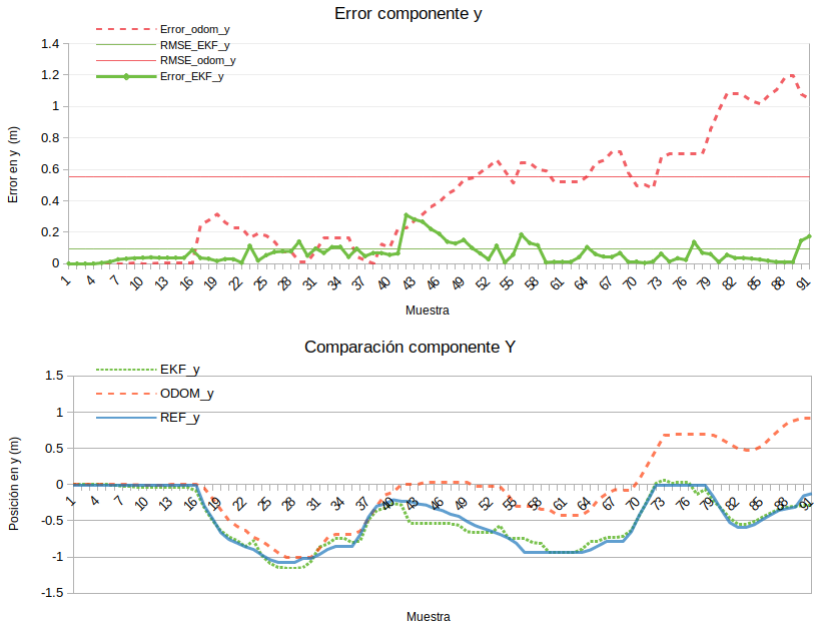


Figura 6.10: La gráfica superior muestra el error sobre el eje y respecto a la referencia. La gráfica inferior es comparativa entre la componente y de la odometría, el método propuesto y la referencia, para esta prueba se colocaron 13 marcas en el ambiente.

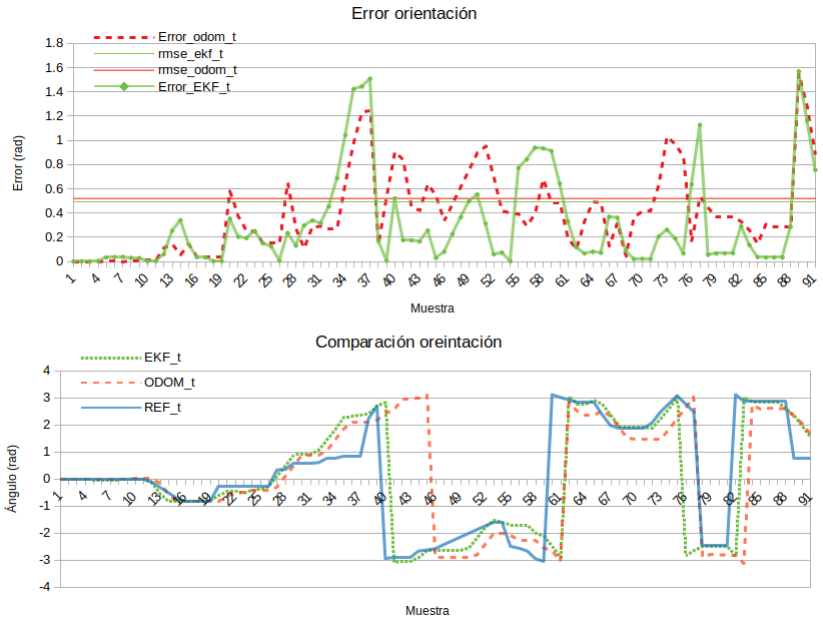


Figura 6.11: La gráfica superior muestra el error en el ángulo de orientación del robot respecto a la referencia. La gráfica inferior es comparativa entre la componente x de la odometría, el método propuesto y la referencia, para esta prueba se colocaron 13 marcas en el ambiente.

En las figuras (Fig: 6.9), (Fig: 6.10) y (Fig: 6.11) se muestran las diferencias tanto

de la odometría como del método propuesto para las componente x, y y θ respectivamente, respecto a la referencia. Calculando la raíz cuadrada del error cuadrático medio se puede observar que el error es menor para el método propuesto en las tres componentes. Sin embargo a pesar de que en esta prueba se redujo la cantidad de marcas en el ambiente el aumento del error no es tan significativo.

6.3. Tercera prueba

Para la tercera prueba se redujeron las marcas a 5 como se muestra en la figura (Fig: 6.12).

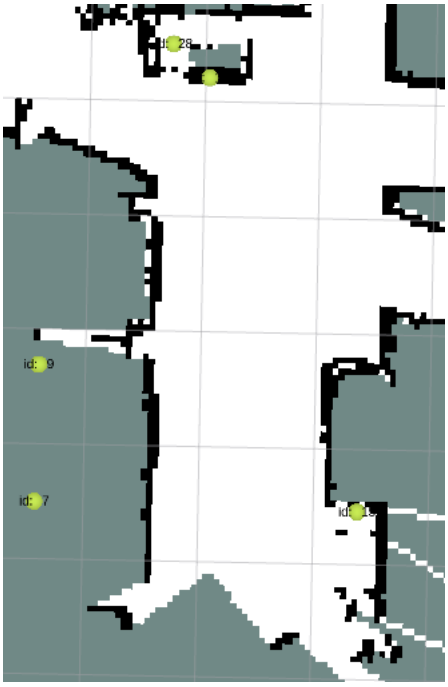


Figura 6.12: Posición de los 5 códigos ArUco dispuestos en el entorno del robot

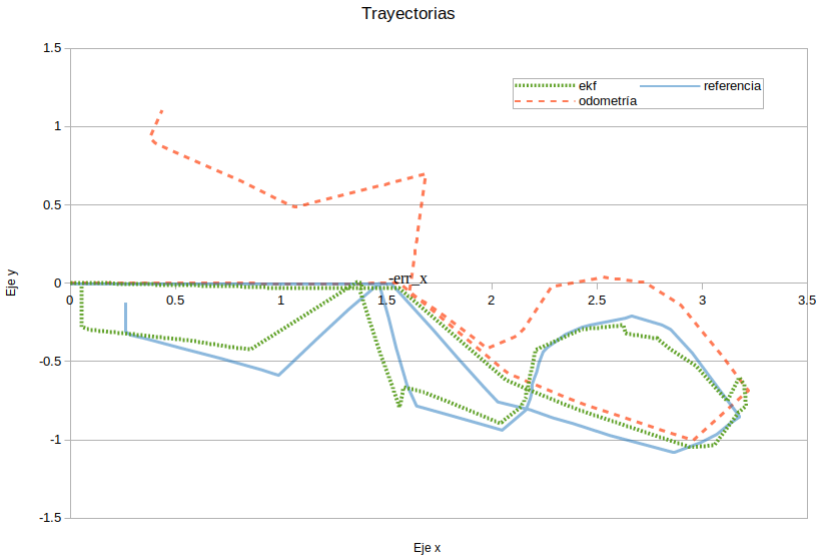


Figura 6.13: Se muestran tres trayectorias, azul trayectoria de referencia, rojo punteado odometría, verde EKF.

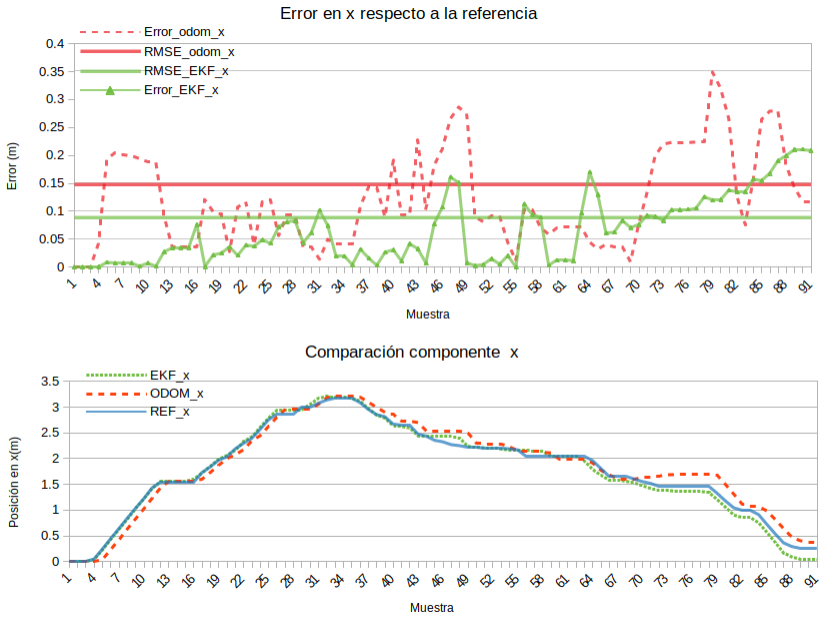


Figura 6.14: La gráfica superior muestra el error sobre el eje x respecto a la referencia. La gráfica inferior es comparativa entre la componente x de la odometría, el método propuesto y la referencia, para esta prueba se colocaron 5 marcas en el ambiente.

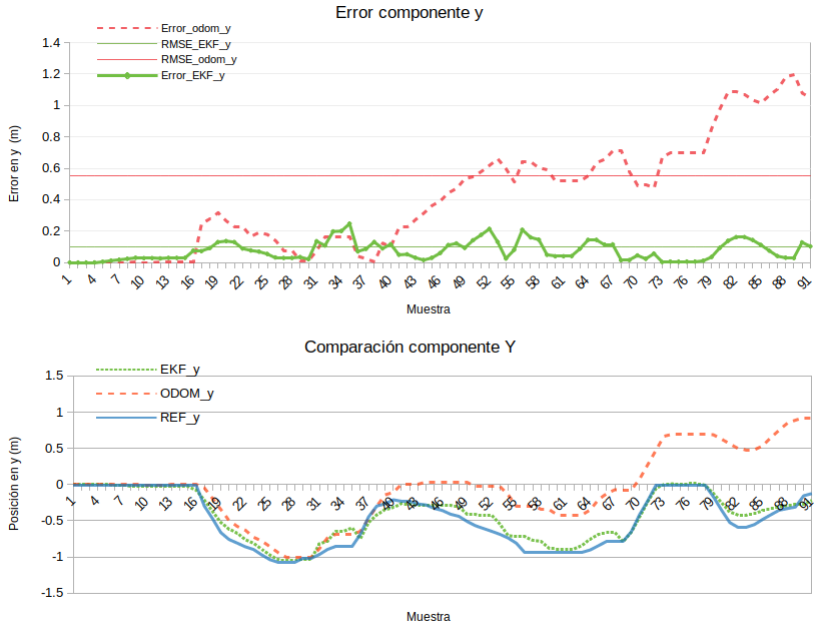


Figura 6.15: La gráfica superior muestra el error sobre el eje y respecto a la referencia. La gráfica inferior es comparativa entre la componente y de la odometría, el método propuesto y la referencia, para esta prueba se colocaron 5 marcas en el ambiente.

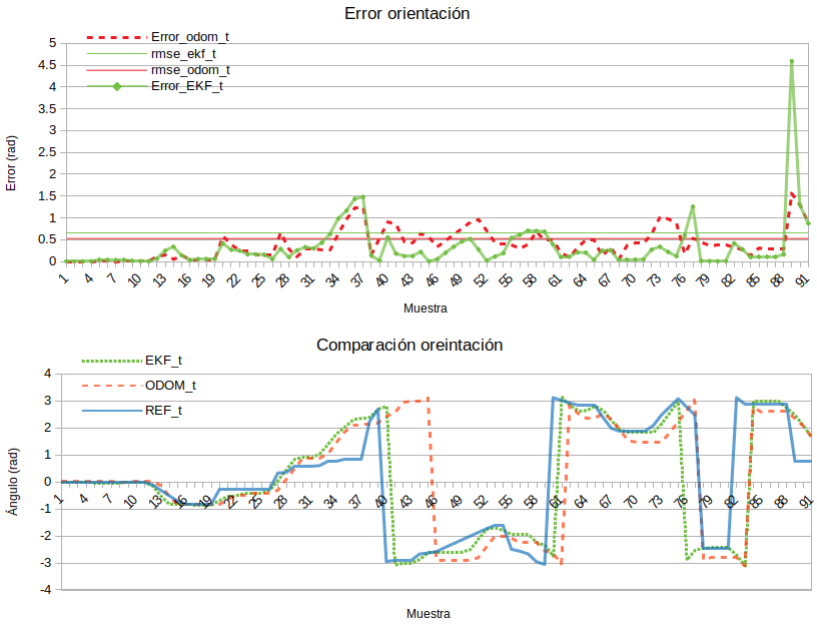


Figura 6.16: La gráfica superior muestra el error en el ángulo de orientación del robot respecto a la referencia. La gráfica inferior es comparativa entre la componente x de la odometría, el método propuesto y la referencia, para esta prueba se colocaron 5 marcas en el ambiente.

En las figuras (Fig: 6.14), (Fig: 6.15) y (Fig: 6.16) se muestran las diferencias tanto de la odometría como del método propuesto para las componente x, y y θ respectivamente, respecto a la referencia. Si bien el método propuesto parece seguir la trayectoria de referencia, el aumento del error respecto a las pruebas previas es mayor incluso el RMSE de la componente θ es mayor en el método propuesto, sin embargo, la trayectoria resultante no diverge tanto como en la odometría.

6.4. Conclusiones de las pruebas 1, 2 y 3

En la siguiente tabla se muestra una comparación de la raíz cuadrada del error cuadrático medio (RMSE) para las tres pruebas anteriores donde la variación fue el numero de marcas dispuestas en el entorno del robot. Es importante recalcar que para estas pruebas el robot siguió una trayectoria y se guardaron las lecturas de sus sensores, para después poder variar el número de marcas sin modificar la trayectoria de referencia.

Comparativa del RMSE para las diferentes componentes						
Marcas	Odom x	EKF x	Odom y	EKF y	Odom θ	EKF θ
22	0.1460 m	0.0608 m	0.5563 m	0.0827 m	0.5212 rad	0.4748 rad
13	0.1460 m	0.0747 m	0.5563 m	0.0905 m	0.5212 rad	0.4857 rad
5	0.1460 m	0.0874 m	0.5363 m	0.0951 m	0.5212 rad	0.6442 rad

Como se puede apreciar en la tabla anterior, el valor RMSE aumenta en las tres componentes conforme disminuye el número de marcas, sin embargo no se aprecia un comportamiento lineal en relación con el número de marcas. Las trayectorias generadas son diferentes debido a que a lo largo del tiempo el error oscila y estas oscilaciones derivan en una aproximación de la pose. También se notó durante la experimentación que mientras los parámetros $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ y las matrices Q y R sean adecuados para el robot y sus sensores, si al menos se ve una marca durante movimientos de traslación el método estima de manera aceptable su pose. Para los movimientos de rotación es necesario al menos dos marcas, pues si la incertidumbre aumenta al no ver marcas y

posterior a un movimiento de rotación se detecta una sola marca, el método no puede aproximar con exactitud la pose del robot, debido a que a pesar de que el método sea capaz de aproximar la pose con la distancia d a la marca, existen varias posiciones x, y donde se cumple que la marca este a la distancia d.

Capítulo 7

Conclusiones

7.1. Conclusión

En el presente trabajo se desarrolló la metodología para que un robot móvil se localizará en un entorno conocido. Por medio de la experimentación se llegó a la conclusión de que un sistema basado en el reconocimiento de códigos ArUco y el EKF funciona correctamente bajo las siguientes condiciones:

- El robot debe de detectar al menos dos marcas al momento de ejecutar el EKF, de lo contrario la incertidumbre aumentará y por tal motivo el error en la estimación de la pose incrementaría.
- Las constantes $\alpha_1, \alpha_2, \alpha_3, \alpha_4, Q$ y R mencionadas en el capitulo 5 varían de un robot a otro y es de suma importancia hacer un equilibrio entre los valores teóricos y prácticos.

El método de detección de marcas es rápido pues tiene un velocidad de muestreo de aproximadamente 25 hz. La detección de marcas es la parte que requiere más poder de computo del sistema propuesto. Por lo tanto, se puede decir que el método es rápido para los componentes de hardware con los que se hicieron las pruebas.

El éxito de una navegación estable depende de el número de marcas en el ambiente reconocidas el momento de ejecutar el EKF. Un mayor número marcas utilizadas en

la etapa de actualización, da como resultado una mejor estimación de la pose del robot.

7.2. Trabajo futuro

Como trabajo futuro se plantea modificar las condiciones que se deben de cumplir para ejecutar el EKF. En el método propuesto no se asegura que en el momento de la ejecución el robot reconozca marcas. De esta manera se puede aprovechar en mayor medida las marcas reconocidas y mejorar la estimación de la pose. También se plantea hacer una comparación entre diferentes métodos de localización, los cuales quedaron fuera de este trabajo, pues se considera que para hacer dicha comparación se debe de dar una explicación detallada de cada método de localización, lo cual es un tema extenso. El método propuesto únicamente fue analizado para la adquisición a través de una cámara RGB-D, sin embargo un método que utilice únicamente una cámara RGB seria un factor importante pues no se requerirá manipular nubes de puntos, lo cual es una de las partes que consumen mayores recursos. Un sistema basado únicamente en una cámara RGB disminuiría los costos de un robot, incluso se podría utilizar este método en robots móviles de menor tamaño y además en robots que trabajen en exteriores. El uso de este método para sistemas 3D es otra área que es importante abordar con el método propuesto, pues a pesar de que el reconocimiento de las marcas sea en tres dimensiones en el sistema actual el EKF únicamente funciona para dos dimensiones.

Apéndice A

Simulador de robots móviles

El simulador presentado en esta tesis, es un simulador de robots móviles que ejecutan comandos de giro y de avance. La interfaz gráfica fue realizada con el lenguaje de programación Python y los subprogramas necesarios para funcionar están hechos en el lenguaje de programación C.

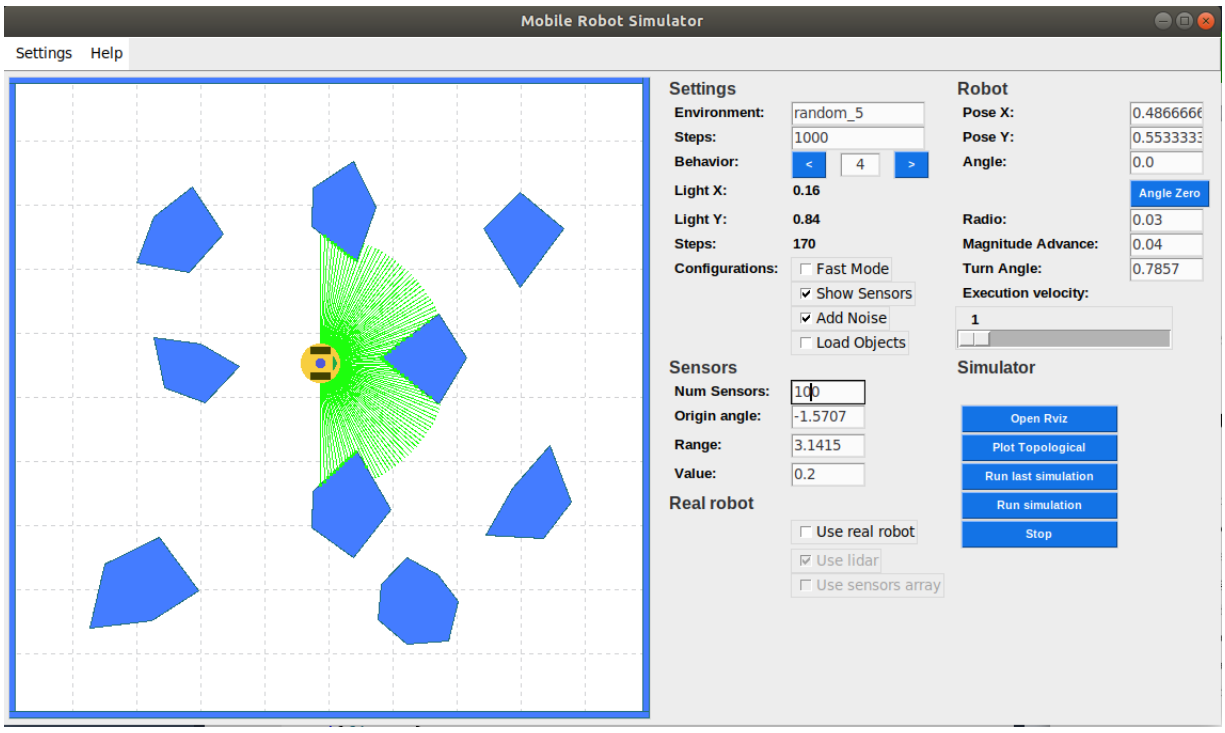


Figura A.1: Interfaz gráfica del simulador

El robot es capaz de girar y desplazarse en linea recta tanto positiva como nega-

tivamente, sin embargo. El robot no es capaz de ejecutar un movimiento lineal y uno angular al mismo tiempo.

El robot simulado tiene un configuración de par diferencial y cuenta con la simulación de un escáner 2D. El escáner 2d es configurable por medio de tres parámetros, número de lecturas, inicio de las lecturas y el rango de operación. El origen del escáner se encuentra en el centro de rotación del robot simulado. El inicio de las lecturas es indicado en radianes. Un valor positivo es en sentido contrario a las manecillas del reloj y un valor negativo en sentido de las manecillas del reloj, en la figura (Fig. A.2) y (Fig. A.3) se muestran diferentes configuraciones.

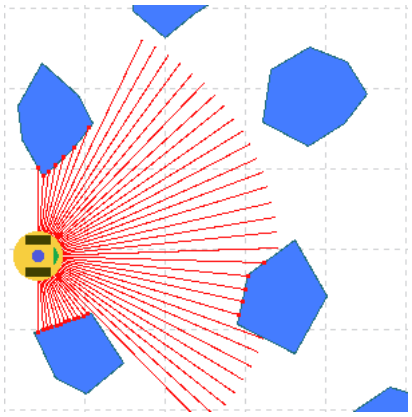


Figura A.2: Configuración del sensor láser con valores: inicio de las lecturas igual a -1.5707 rad, número de lecturas igual a 50, rango de operación igual a π y rango de operación igual a 0.3 m.

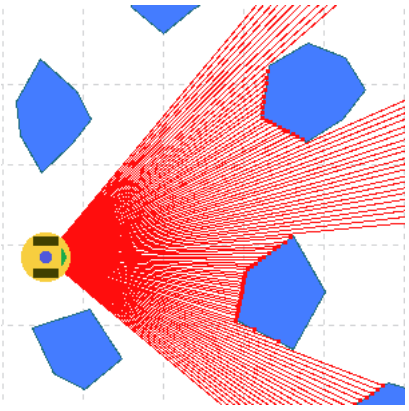


Figura A.3: Configuración del sensor láser con valores: inicio de las lecturas igual a -0.7 rad, número de lecturas igual a 50, rango de operación igual a 1.5707 y rango de operación igual a 0.6 m.

En el entorno del robot se encuentran polígonos convexos que son definidos en archivos de configuración, en los cuales, se describen los puntos que conforman cada uno de los polígonos. Los archivos de configuración tienen como extinción .wrl y son archivos de texto.

Los archivos wrl son utilizados por el programa encargado de calcular la lectura del escáner por medio de un algoritmo de Ray Casting.

El robot simulado también cuenta con 8 sensores que capturan la intensidad de la luz para saber en que dirección se encuentra la fuente luminosa, dichos sensores están simulados equidistante mente en el contorno del robot. La fuente luminosa es indicada por el usuario a través del puntero.

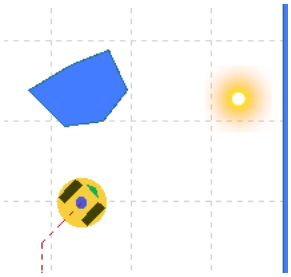


Figura A.4: Robot móvil en dirección a la fuente luminosa detectada por sus 8 sensores de intensidad luminosa

El simulador recibe comandos de movimiento a través de el framework ROS donde un nodo se encarga de procesar las lecturas de los sensores y tomar decisiones, dicho nodo funciona a través de maquinas de estados donde se definen comportamientos reactivos para llegar a la fuente luminosa, evadir, obstáculos, etc.

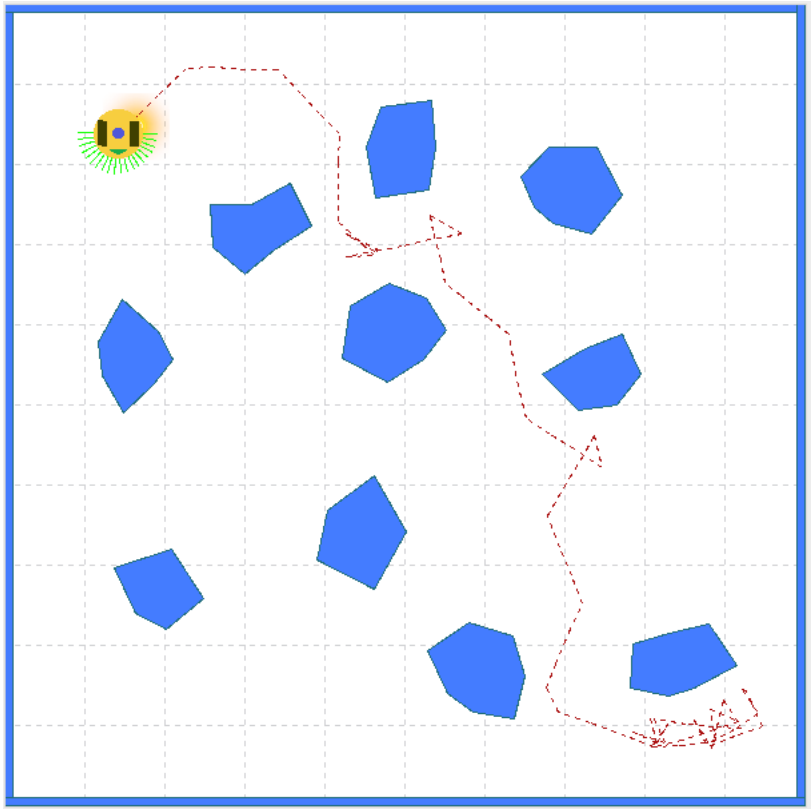


Figura A.5: En la linea punteada se muestra la trayectoria que realizó el robot para llegar a la fuente luminosa esquivando obstáculos.

El simulador también cuenta con algoritmos para encontrar rutas, uno de ellos es el algoritmo de Dijkstra, el cual recibe como entrada un mapa topológico donde se indican puntos en el mapa y las conexiones entre ellos, estos punto y conexiones son navegables por el robot. El mapa topologico se describe en un archivo te texto con estencion .top. Un ejemplo de mapa topológico se muestra en la figura (Fig. A.6) y un ruta calculada se muestra en la figura (Fig. A.7).

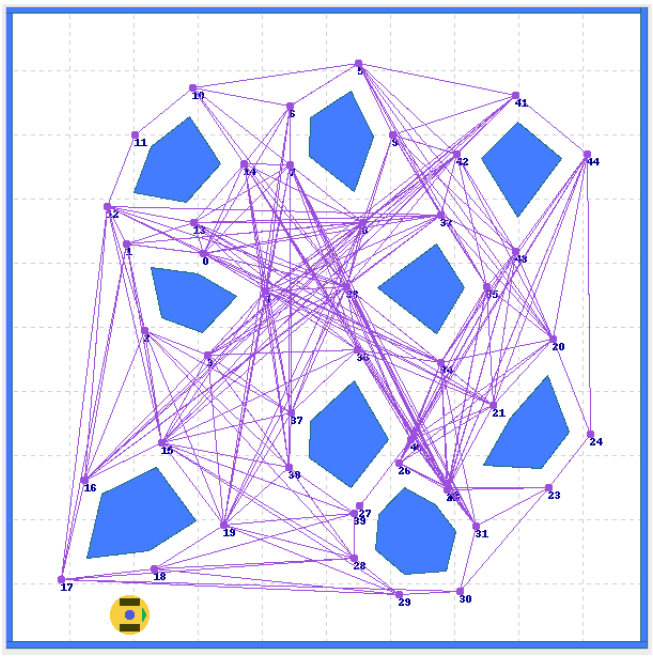


Figura A.6: Mapa topológico.

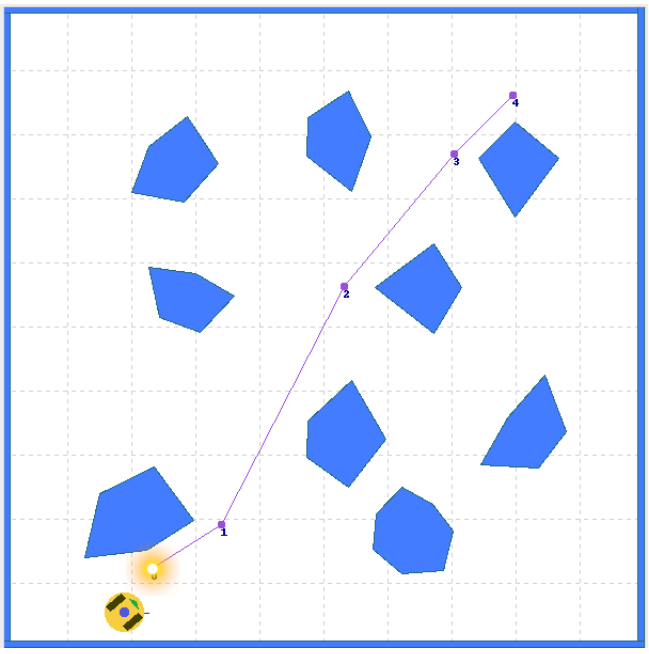


Figura A.7: Ruta calculada por el algoritmo de Dijkstra

El simulador fue utilizado para hacer las primeras pruebas del método propuesto con el filtro de Kalman que se presenta en este trabajo de tesis.

Bibliografía

[1] Thrun, Sebastian, Wolfram Burgard y Dieter Fox: *Probabilistic Robotics*. The MIT Press Cambridge, Massachusetts London, England, 2006.

[2] Yukihiro Minami Koyama, Jesús Savage Carmona: *Determinacion de la posicion de un robot movil usando transmisiones ultrasonicos*. Tesis de Licenciatura, Universidad Nacional Autónoma de México,, 2004. Pag. 36 y 37.

[3] Faragher, Ramsey *y cols.*: *Understanding the basis of the Kalman filter via a simple and intuitive derivation*. IEEE Signal processing magazine, 29(5):128–132, 2012.

[4] Savage, Jesús: *Filtro de Kalman*. `https://biorobotics.fi-p.unam.mx/wp-content/uploads/Courses/robots_moviles/2020_2/lecciones/Leccion11_robots_moviles_2020_2.pdf`, 2019.

[5] Pfaff, Patrick, Wolfram Burgard y Dieter Fox: *Robust monte-carlo localization using adaptive likelihood models*. En *European robotics symposium 2006*, páginas 181–194. Springer, 2006.

[6] Huang, Thomas: *Computer vision: Evolution and promise*. 1996.

[7] Torres, Alejandro Domínguez: *Procesamiento digital de imágenes*. Perfiles Educativos, (72), 1996.

[8] Szeliski, Richard: *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[9] Lim, Jae S: *Two-dimensional signal and image processing*. ph, 1990.

[10] Bhabatosh, Chanda *y cols.*: *Digital image processing and analysis*. PHI Learning Pvt. Ltd., 2011.

[11] Yang, Michael Ying y Wolfgang Förstner: *Plane detection in point cloud data*. En *Proceedings of the 2nd int conf on machine control guidance, Bonn*, volumen 1, páginas 95–104, 2010.

[12] Qi, Charles R, Hao Su, Kaichun Mo y Leonidas J Guibas: *Pointnet: Deep learning on point sets for 3d classification and segmentation*. En *Proceedings of the IEEE conference on computer vision and pattern recognition*, páginas 652–660, 2017.

[13] Agarwal, Anubhav, CV Jawahar y PJ Narayanan: *A survey of planar homography estimation techniques*. Centre for Visual Information Technology, Tech. Rep. IIIT/TR/2005/12, 2005.

[14] Dubrofsky, Elan: *Homography estimation*. Diplomová práce. Vancouver: Univerzita Britské Kolumbie, 5, 2009.

[15] Lu, Xiao Xin: *A review of solutions for perspective-n-point problem in camera pose estimation*. En *Journal of Physics: Conference Series*, volumen 1087, página 052009, 2018.

[16] Romero-Ramirez, Francisco J, Rafael Muñoz-Salinas y Rafael Medina-Carnicer: *Speeded up detection of squared fiducial markers*. Image and vision Computing, 76:38–47, 2018.

[17] Nina, Oliver, Bryan Morse y William Barrett: *A recursive Otsu thresholding method for scanned document binarization*. En *2011 IEEE Workshop on Applications of Computer Vision (WACV)*, páginas 307–314. IEEE, 2011.