MAPEO Y LOCALIZACIÓN SIMULTÁNEOS UTILIZANDO UN SISTEMA DE VISIÓN

LUIS CARLOS ALVEAR VEGA
LUIS FELIPE VELASCO MUÑOZ

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES
DEPARTAMENTO DE ELECTRÓNICA, INSTRUMENTACIÓN Y CONTROL
LÍNEA DE INVESTIGACIÓN EN APLICACIÓN DE TECNOLOGÍAS INTELIGENTES
POPAYÁN
2008

MAPEO Y LOCALIZACIÓN SIMULTÁNEOS UTILIZANDO UN SISTEMA DE VISIÓN

Luís Carlos Alvear Vega
Luís Felipe Velasco Muñoz

Trabajo de Grado para Optar al Titulo de
Ingeniero en Automática Industrial.

Director
Elena Muñoz España
Ingeniera en Electrónica y Telecomunicaciones

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES
DEPARTAMENTO DE ELECTRÓNICA, INSTRUMENTACIÓN Y CONTROL
LÍNEA DE INVESTIGACIÓN EN APLICACIÓN DE TECNOLOGÍAS INTELIGENTES
POPAYÁN
2008

Tabla de Contenido

Pág.

Lista de Figuras

Lista de Tablas

Resumen

En el presente trabajo se diseña e implementa un sistema capaz de realizar el mapeado y la localización simultánea para un robot móvil que se desplaza a través de un ambiente estructurado.

Dicho trabajo inicia con la descripción de los principales tipos de mapas que se pueden levantar de un ambiente; luego, se lleva a cabo un estudio de las técnicas probabilísticas base de los algoritmos que mejores resultados han dado al problema del SLAM. Posteriormente, se procede a analizar con mayor detalle los principales algoritmos existentes en la literatura para implementar SLAM.

De otra parte, se elabora un estudio detallado de las principales técnicas existentes en el procesamiento digital de imágenes, necesarias para proceder a elaborar una implementación básica del sistema SLAM basada en visión artificial y en el algoritmo del filtro extendido de Kalman.

Los resultados obtenidos dieron una muy buena aproximación en la construcción de mapas y en la localización del robot móvil dentro del entorno establecido. Además, la implementación propuesta sirve como base para trabajos posteriores.

Introducción

Un robot móvil es un dispositivo mecánico capaz de desplazarse por un entorno mientras realiza una determinada tarea, la principal habilidad con la que debe contar un robot móvil es la de moverse de forma autónoma por un entorno, evitando obstáculos, objetos y personas en movimiento [1].

Para que un robot móvil pueda ser realmente autónomo deberá contar con la habilidad esencial de explorar su entorno y crear un mapa de el. El problema de crear un mapa mientras el robot se localiza dentro de este, se denomina Mapeado y Localización Simultanea (SLAM). Con el presente trabajo se busca sentar las bases necesarias para llevar a cabo el proceso de SLAM utilizando un sistema de visión, realizar su implementación a partir de uno de los algoritmos existentes y aportar experiencia para trabajos futuros.

En el primer capítulo se realiza una descripción del problema SLAM, de los diferentes tipos de mapas que se pueden generar, se realiza una descripción detallada de la formulación bayesiana necesaria para entender los principales algoritmos existentes para el SLAM. En el segundo capítulo, se describe con mayor detalle dos de los principales algoritmos que dan solución al problema del SLAM, el Filtro Extendido de Kalman y el Filtro de Partículas.

En el tercer capitulo, se describe la implementación llevada a cabo para conseguir el proceso de SLAM basado en el Filtro Extendido de Kalman EKF y en técnicas de visión artificial. Por último, en el cuarto capitulo se presenta toda la experimentación llevada a cabo y las conclusiones obtenidas durante su desarrollo.

1. LOCALIZACIÓN Y MAPEO SIMULTÁNEOS SLAM

1.1 Introducción

La robótica móvil estudia a los robots que realizan tareas desplazándose autónomamente en ambientes dinámicos y complejos, es decir, desde espacios interiores como oficinas y casas, hasta exteriores como espacios terrestres, submarinos y sustentados sobre el aire. Estas habilidades llevan implícita la capacidad de razonar sobre una representación interna del mundo y la autonomía significa la no intervención de ningún operador humano en el proceso de movimiento del robot.

Los mecánicos fueron los pioneros creando máquinas capaces de tener movimientos repetidos, elegantes, controlados, calculados y predecibles. Luego llegaron los electrónicos, quienes dieron a esos movimientos un significado de automatización; ahora las máquinas podrían interactuar en ambientes más complejos, recibiendo retroalimentación, manipulando más variables. Finalmente llega la gente de la computación, tratando de dotar de cierto grado de inteligencia a esas máquinas; en una palabra, autonomía. Se intenta pasar de la automatización y teleoperación a la autonomía [2]

Uno de los problemas abarcado en la literatura de la robótica móvil, y el enfoque de este trabajo, son los robots que interactúan en espacios interiores [1]. El paso entre un robot que haga tareas eficientemente y un robot que sólo pueda moverse y percibir su ambiente es grande y no trivial, ya que implica el razonamiento sobre un ambiente para realizar una planeación inteligente de sus movimientos. Para poder realizar este razonamiento de su espacio de operación, el robot debe tener una representación del mismo, es decir, un mapa de su ambiente. Con el mapa del ambiente, el robot podría realizar tareas de localización y planeación de movimientos [2].

Inicialmente los investigadores daban al robot una representación del ambiente. Estos mapas eran hechos de manera externa al robot y por lo mismo eran difíciles de empatar con la perspectiva del robot. Nuestra percepción es distinta a la del robot. Por lo que la

tendencia es que el robot genere su propia representación del ambiente y se localice dentro de éste, utilizando sus propios sensores con sus limitaciones intrínsecas y alcances [2], surgiendo así el problema de la localización y mapeo simultáneos en la robótica móvil.

1.2 Justificación

El problema del mapeado y de localización simultánea, mejor conocido en la literatura como SLAM debido a sus siglas en inglés (*Simultaneous Localization and Mapping*), es un tema con bastante trabajo realizado; sin embargo, aún no se llega a un algoritmo que sea aceptado ampliamente por la comunidad científica, aunque las soluciones con enfoques probabilistas son las más utilizadas.

Para comenzar a describir el problema de SLAM, como ya se ha dicho, el objetivo es crear una representación del ambiente donde trabajará el robot, en otras palabras, un mapa. Para hacer esto de manera autónoma, el robot se vale de sus sensores para percibir el mundo. Una vez que el robot se encuentra en un punto y sensa su ambiente, el robot debe cambiar su ubicación para descubrir la topología del nuevo punto y finalmente integrar ambas representaciones en un mapa general. A esta forma de generar las representaciones del ambiente se les llama incrementales, ya que el procesamiento del mapa ocurre cada vez que se tiene un nuevo punto de vista del ambiente.

Sin embargo se presenta el problema de que la información de los sensores contiene mucho ruido que conduce a errores e inexactitudes en las mediciones. Este problema es más notorio al tratarse de las medidas relacionadas con la odometría[1] del móvil, ya que este error aumenta con el tiempo y la distancia recorrida, por lo que sus lecturas no son en lo absoluto confiables. Entonces, si se desea integrar dos representaciones vistas por el robot en tiempos distintos, se debe conocer exactamente la posición donde el robot

[1] Sistema capaz de estimar el movimiento relativo incremental del robot.

hizo las mediciones. No obstante, el trabajo de localización requiere un mapa previo para relacionar las mediciones con la representación. Por estas razones el problema obliga resolver simultáneamente los aspectos de localización y mapeo, ya que la localización no se puede conocer de manera exacta con simples mediciones y para que el robot realice un mapeado del entorno es necesario tener conocimiento de su localización.

1.3. Descripción del problema

Este trabajo de grado cubre principalmente el estudio de las técnicas más importantes para resolver el problema del SLAM (Mapeado y Localización Simultáneos); posterior a ello se realizará la selección de una de estas técnicas para desarrollar una implementación básica utilizando como elemento principal de medición del entorno, una cámara y como elementos de medida de desplazamiento, el sistema odométrico del robot ATIBOT desarrollado por el grupo de investigación ATI de la Universidad del Cauca.

1.3.1. El problema de la construcción de un mapa

Para adquirir el mapa del entorno, un robot móvil debe disponer de sensores estereoceptivos, que le permitan percibir los objetos del mundo que le rodea. Entre estos sensores se pueden citar los sonares o ultrasonidos, infrarrojos, escáneres láser, cámaras mono o estéreo y sensores de contacto. Todos estos sensores presentan como característica fundamental la existencia de ruido en sus medidas, medidas que son referenciadas relativamente a la posición (en la que se incluye la orientación) del robot [3]. Otra característica importante es el a menudo limitado rango de los mismos, con alcances efectivos de pocos metros.

Aunque no son absolutamente necesarios, los robots móviles también suelen incorporar sensores propioceptivos que le permiten obtener medidas relacionadas con su estado: velocidad, incremento de posición y aceleraciones. Algunos de estos sensores son los *encoders* ópticos incrementales, acelerómetros, giróscopos y sensores de velocidad relativos al exterior. Estos sensores permiten obtener una estimación incremental del movimiento del robot, pero debido a su ruido inherente no son capaces de mantener la

posición del robot correctamente a lo largo del tiempo, ya que los errores son acumulativos [3].

En un proceso típico de construcción de un mapa de un entorno, se parte de un mapa vacío o desconocido en el que el robot parte de una posición conocida, típicamente el origen de coordenadas. Comienza el proceso, y el robot es capaz de construir la parte del mapa cercana a su posición, pero debido al limitado rango de los sensores, el robot debe moverse hacia las zonas que quieran ser incluidas en el mapa. Según se va desplazando hacia esas nuevas áreas, observa los objetos presentes en ellas y va actualizando el mapa del entorno. El ruido de las medidas hace que el mapa no sea exacto, y a su vez el robot se mueve con desplazamientos de los que solo se tiene una estimación, con lo que su posición tampoco es exacta. Cuando se realizan observaciones de objetos nuevos, la estimación de la posición de esos objetos depende tanto de la incertidumbre de la posición del robot como de la incertidumbre de las medidas realizadas [3].

El procesamiento del mapa del entorno se puede realizar después de la adquisición y almacenamiento de los datos sensoriales, con lo que el tiempo de procesamiento no es critico, o se puede realizar incrementalmente, es decir a medida que llegan los datos de los sensores. Esta segunda forma de procesamiento requiere que el tiempo de procesamiento empleado en la actualización del mapa con la llegada de cada información sensorial, sea inferior al periodo de muestreo de dichos sensores, si no se desea incurrir en perdida de información [3].

En la figura1.1, se observa un esquema que refleja de manera comprensible las operaciones involucradas en el problema del SLAM, mientras que en la tabla 1 se describen las variables utilizadas en el esquema.

Para el problema incremental del SLAM, al robot que se encuentra en la posición X_{k-1} se le aplica un vector de control U_k en el tiempo $k-1$. El tiempo k se considera discreto (1, 2,…,). En la nueva posición X_k, el robot, utilizando sus sensores de distancia, toma mediciones Z_{k-1} de los objetos (m_i) que componen el ambiente visible para el robot [2].

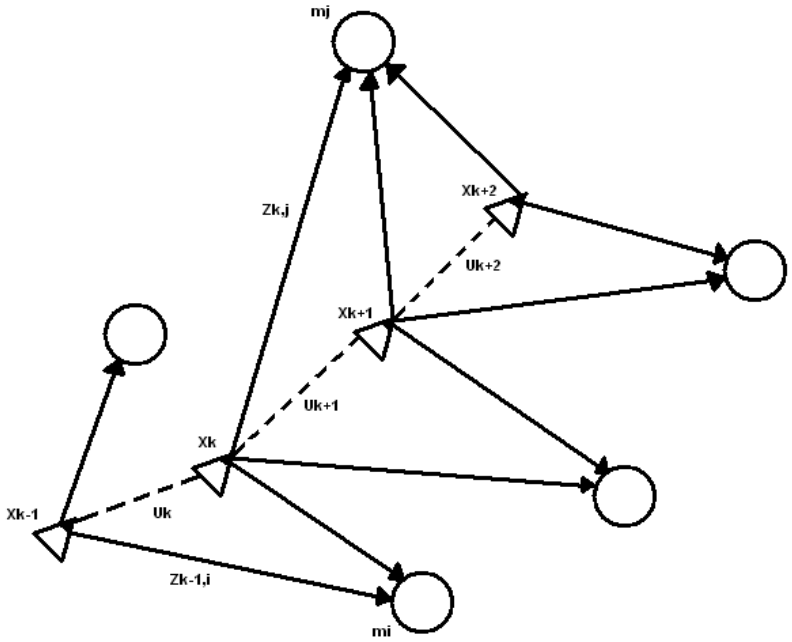

Figura 1. 1. . Descripción del problema SLAM [2].

Tabla 1. 1. Notación general de la descripción del SLAM.

Variable	Descripción
x_k	Posición del robot en el tiempo k
U_k	Vector de control aplicado al tiempo k-1 para mover al robot de X_{k-1} a X_k en el tiempo k
m_i	Posición de las marcas (elemento del mapa) m_i
$Z_{k,i}$	Observación (medición) de la marca m_i desde la posición X_k al tiempo k.

Esta labor tiene múltiples complicaciones que han convertido a este problema en un tema muy atractivo para la comunidad científica. Estas complicaciones se pueden agrupar en cinco categorías:

- El ruido de medición. Los sensores de distancia son inexactos, susceptibles a ruido y muy dependientes de sus capacidades físicas. La odometría es más

compleja ya que su ruido es estadísticamente dependiente, debido a que su error es acumulativo.

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

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

- Ambientes dinámicos. Una de las restricciones aplicadas al problema de la exploración es que el ambiente debe estar estacionario al momento de hacer esta tarea.

1.4 Tipos de mapas a reconstruir

Uno de los temas pendientes en el problema del SLAM es una forma unificada de representación de ambientes. Sin embargo, actualmente dependiendo del tipo de ambiente a mapearse, se hace uso de un tipo de representación.

A grandes rasgos se pueden categorizar los ambientes en dos grupos:

1. Interiores y exteriores
2. Estructurados y no estructurados

Los ambientes interiores son relativamente pequeños, con muchas estructuras rectangulares; el piso es plano y su dinámica es ligeramente controlable y predecible. En cambio los ambientes exteriores son espacios grandes con obstáculos muy dispersos, de formas irregulares; el terreno es disparejo y es imposible de controlar y predecir.

Los ambientes estructurados tienen regiones claramente distinguibles (cuadros, oficinas, marcas etc.). En los ambientes no estructurados no se pueden hacer suposiciones previas sobre el espacio.

Teniendo en cuenta las combinaciones de las dos categorías expuestas, se puede elegir el tipo de representación, considerando las restricciones de memoria y velocidad del sistema de cómputo a utilizar [2].

De manera general se pueden agrupar las diferentes representaciones del ambiente en tres grandes grupos:

- *Geométricos*: representan el ambiente por medio de primitivas geométricas, tal como puntos, celdas, líneas o polígonos. Estos tipos de mapas son los más detallados y ocupan más recursos de memoria.

- *Topológicos*: representa al espacio como un conjunto de grafos, donde cada marca natural es un vértice y sus arcos son conexiones entre ellos. Son más compactos pero eliminan mucha información métrica que puede ser importante.

- *Semánticos*: en esta representación el grafo compuesto por los lugares significativos del entorno y sus conexiones, son meramente etiquetados con atributos lingüísticos.

La inmensa mayoría de algoritmos existentes trabajan con la hipótesis de un mundo bidimensional, hipótesis condicionada en gran medida por la capacidad limitada de los sistemas sensoriales del robot. Siendo esta bidimensionalidad una simplificación considerable. A partir de ella, se plantean estos tres niveles posibles de representación en la definición de un mapa: Geométrico, Topológico y Semántico.

1.4.1 Nivel de Representación Geométrico

La información integrada en un mapa métrico se compone de elementos geométricos tales como puntos, segmentos, planos etc. En este sentido, el modelo de mapa resulta menos abstracto, esto es, más cercano al entorno real. De esta forma se simplifica en gran medida la adquisición de observaciones, ya que la información extraída de los sensores suele ser también información métrica.

En la figura.1.2 se representa un mapa basado en características geométricas de un entorno típico de interiores. En dicha figura, el mapa obtenido esta compuesto por segmentos que representan las paredes del entorno [3].

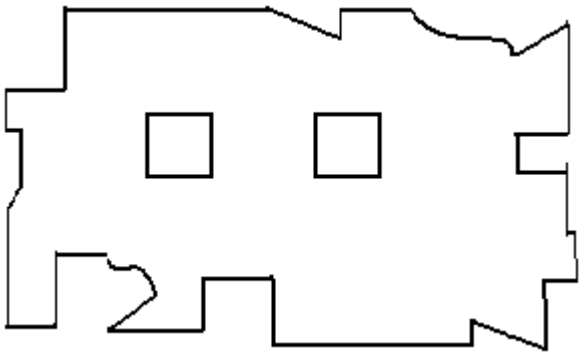

Figura 1.2. Mapa geométrico [3].

Esta representación es bastante útil en entornos estructurados como los de interiores en los que sea posible una extracción lo suficientemente robusta de características geométricas. Tiene como gran ventaja que al realizar una extracción previa de características, se produce un filtrado de los objetos dinámicos del entorno, sobre todo cuando se considera que las características extraídas como segmentos (paredes) o puntos (barras o esquinas) siempre están fijas, mientras que objetos dinámicos como las personas son automáticamente eliminados en el pre-procesado. No obstante, estos algoritmos requieren de sensores con menor ruido como los escáneres láser o más complejos como la visión, ya que los sensores como los ultrasonidos son difícilmente utilizables a la hora de distinguir objetos en el entorno [3].

Entre sus desventajas se pueden destacar su incapacidad para realizar un modelo completo del entorno. Queriendo decir que todo lo que no sea una característica geométrica será descartado y no considerado en el mapa, es decir, se pierde gran parte de la información sensorial en aras de la compacidad[2] y la robustez. Esto provoca que estos mapas a menudo no sean útiles para tareas comunes en navegación de robots móviles [3].

[2] La compacidad es aquello que manifiesta la calidad de compacto. El adjetivo compacto representa una masa muy unida; un agregado cuyos elementos constituyentes están muy poco o nada separados los unos de los otros.

1.4.2 Nivel de Representación Topológico

Aunque es común obtener la representación topológica de un entorno, a partir de una previa representación métrica, existen numerosas aproximaciones que contemplan el problema de construir directamente un grafo topo-geométrico del entorno, en el que se modelan espacios representativos del mismo (habitaciones, pasillos, esquinas, etc.) y las conexiones entre los mismos. Son muy comunes las representaciones basadas en el Diagrama de Voronoi[3] ya que representa el esqueleto de la figura geométrica contemplada, es decir el mapa completo del entorno como lo ilustra la figura.1.3. [3].

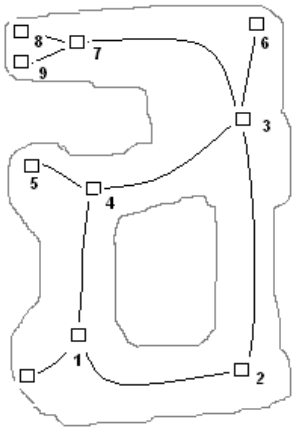

Figura 1.3. Mapa Topológico basado en el diagrama de Voronoi [3].

La compacidad obtenida por esta representación es todavía mayor que con los mapas basados en características geométricas. Sin embargo, las habilidades sensoriales del robot deben ser incrementadas hasta el punto de poder distinguir los complejos espacios representativos [3].

1.4.3 Nivel de representación Semántico

Se distingue del topológico en que toda la información geométrica es totalmente descartada. Queda pues en esta representación el grafo compuesto por los lugares significativos del entorno y sus conexiones, meramente etiquetados con atributos lingüísticos. En la figura 1.4 se observan los mapas topológico y semántico de un entorno de interiores [3].

[3] El diagrama de Voronoi es una construcción geométrica que permite construir una partición del plano euclídeo. Usualmente se utilizan para determinar áreas de influencia.

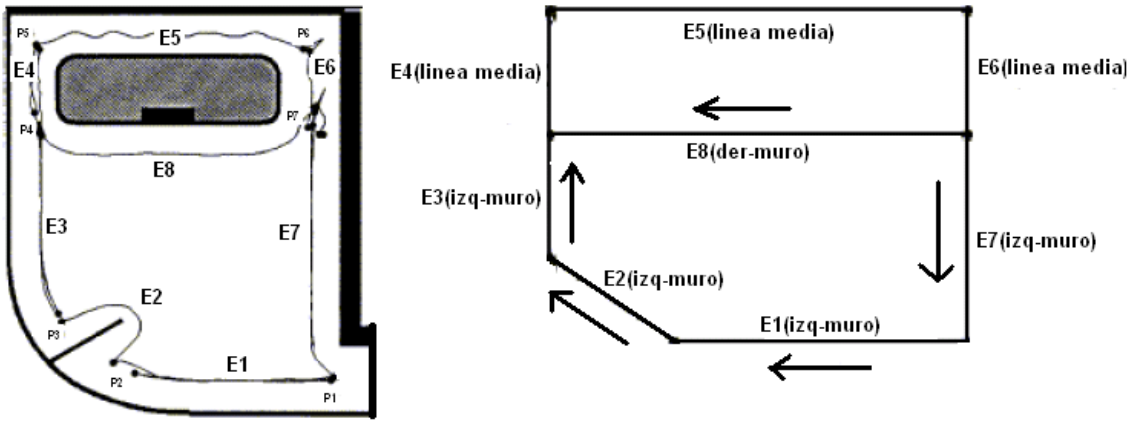

Figura 1.4. Mapa Topológico y Semántico [3].

1.5 Mapeado y Localización Simultáneos (SLAM) con técnicas probabilísticas.

Los principales algoritmos existentes en la literatura, basan su solución al problema SLAM en técnicas probabilísticas. Son este tipo de algoritmos, los que alcanzan los mejores resultados. Cualquier solución al problema SLAM que utilice estas técnicas está fundamentada en el teorema de BAYES [3].

1.5.1 Formulación bayesiana del problema SLAM

Todos los algoritmos de cierta relevancia existentes representan tanto la posición del robot como el entorno probabilísticamente, y todos ellos utilizan la inferencia probabilística como herramienta para transformar las medidas obtenidas en un mapa (también probabilístico) del entorno. La propia incertidumbre de los sensores y modelos del sistema deriva naturalmente en una formulación basada en técnicas probabilísticas [2].

La formulación del problema SLAM basado en el teorema de Bayes, recibe comúnmente el nombre de Filtro de Bayes, el cual se puede entender como una generalización temporal del teorema de Bayes.

Teorema de Bayes:

$$P(A_i|B) = \frac{P(A_i)P(B|A_i)}{\sum_{i=1}^{k} P(A_i)P(B|A_i)} = \frac{P(A_i)P(B|A_i)}{P(B)} \qquad (1.1)$$

Analíticamente, el problema de la Localización y Mapeo Simultáneos consiste en determinar el conjunto de puntos que forman el mapa y las posiciones del robot s_t dadas las mediciones de los sensores y las instrucciones de control. De forma general se puede ver el problema como una regla de Bayes de la siguiente manera:

$$P(x|d) = \eta P(d|x)P(x) \qquad (1.2)$$

Donde se desea aprender de la cantidad *X*, basado en datos de medición *d*. Entonces la regla Bayes indica que el problema puede ser resuelto multiplicando $P(d|x)$ y $P(x)$. El término $P(d|x)$ es el modelo genérico, que describe el proceso de generar mediciones de los sensores bajo diferentes mundos *X*. El término $P(x)$ es llamado antecedente y especifica el deseo de asumir que *X* es la instancia en el mundo antes que lleguen los datos. Finalmente η es un normalizador para asegurar que el lado izquierdo de la regla de Bayes es una distribución de probabilidad valida (entre 0 y1).

En un robot móvil que construye un mapa se pueden distinguir 2 tipos de datos:

1. las medidas propioceptivas del movimiento del robot (odometría) u_t.
2. las medidas estereoceptivos, es decir, de los sensores externos z_t.

Se puede asumir sin perdida de generalidad que dichas medidas llegan secuencial y alternativamente en el tiempo.

$$u_1, z_1, u_2, z_2, \ldots, u_t, z_t \qquad (1.3)$$

Donde estos términos conforman el termino 'd' en la ecuación 1.2. La 'u' representa un desplazamiento relativo del robot y la 'z' una medida sensorial, y cuyos subíndices

representan el instante de tiempo en el que fueron obtenidas, siendo el tiempo 't' el instante actual [4].

Entonces, exponiendo la regla de Bayes anterior con la notación descrita y asumiendo que el comando de movimiento 'u' es independiente de la posición, se tiene:

$$P(x|z,u) = P(z|x,u)P(x|u) \qquad (1.4)$$

Ahora, también se asume que los comandos de movimiento son independientes de las mediciones de distancia, por lo que la anterior ecuación se puede formular así:

$$P(x|z,u) = P(z|x)P(x|u) \qquad (1.5)$$

Haciendo uso de la propiedad Markoviana[4] propia del problema, se puede saber la posición del robot $P(x|u)$ utilizando la posición en el instante anterior.

$$P(x|u) = \sum P(x_t|u_t,x_{t-1})P(x_{t-1}) \qquad (1.6)$$

Se puede suponer que $P(x_{t-1}) = P(x_{t-1}|z^{t-1},u^{t-1})$ considerando que el exponente ^{t} se refiere a todos los datos que existen antes del tiempo *t*, es decir:

$$z^t = \{z_1, z_2, \dots, z_t\}$$

$$u^t = \{u_1, u_2, \dots, u_t\}$$

Por lo que, finalmente, si se sustituye 1.6 en 1.5 se tiene que el problema se puede representar con la siguiente formulación dentro de un espacio continuo:

$$P(x_t|z^t,u^t) = \eta P(z_t|x_t) \int P(x_t|u_t,x_{t-1})P(x_{t-1}|z^{t-1},u^{t-1})dx_{t-1} \qquad (1.7)$$

[4] La condición de Markov determina que el nuevo estado del modelo dinámico depende únicamente del estado y la acción inmediatamente anteriores, es decir que el conocimiento de las acciones previas a las inmediatamente anteriores, no proporciona ninguna información adicional.

La probabilidad posterior $P(x_t|z^t,u^t)$ es calculada utilizando la probabilidad obtenida en el tiempo anterior (propiedad Markoviana). La probabilidad inicial en el tiempo t = 0 es $P(x_t|z^t,u^t) = P(x_0)$.

Es requisito indispensable para un filtro Bayesiano que el estado x_t contenga todas las cantidades desconocidas, esto es el mapa y la posición del robot. Ahora, usando m para designar el mapa y s para describir la posición del robot, se obtiene el siguiente filtro de Bayes [4]:

$$P(s_t,m_t|z^t,u^t)$$
$$= \eta P(z_t|s_t,m_t) \iint P(s_t,m_t|u_t,s_{t-1},m_{t-1}) P(s_{t-1},m_{t-1}|z^{t-1},u^{t-1}) ds_{t-1} dm_{t-1} \quad (1.8)$$

La mayoría de los algoritmos cartográficos asumen un mundo estático, lo que implica que el índice de tiempo en el mapa m no es necesario. Además la mayoría de las propuestas de solución asumen que el movimiento del robot es independiente del mapa [2]. Esto conduce a una forma más conveniente del filtro:

$$P(s_t,m|z^t,u^t) = \eta P(z_t|s_t,m) \int P(s_t|u_t,s_{t-1}) P(s_{t-1},m|z^{t-1},u^{t-1})\, ds_{t-1} \quad (1.9)$$

El problema así formulado puede ser abordado recursivamente[5], necesitando en cada instante de tiempo únicamente los datos sensoriales obtenidos en ese instante y la función de probabilidad del estado en el instante anterior [2].

La expresión (1.9) recibe el nombre de Filtro de BAYES en el problema SLAM. Para que esta expresión pueda ser evaluada se requieren dos modelos o funciones de probabilidad denominadas genéricas, y que corresponden a la información tanto del sistema odométrico como del sistema de percepción externo.

[5] Recursividad hace relación a la palabra recurrencia que significa la aparición repetitiva de algo, proceso en el cual se calcula cualquier término conociendo los precedentes.

Modelo de medida: $P(z_t|s_t,m_t)$

Modelo de movimiento: $P(s_t|u_t,s_{t-1})$

El principal problema es que la ecuación (1.9) no puede ser resuelta ni implementada en su forma general. Por lo tanto es necesaria la realización de simplificaciones, suposiciones o hipótesis añadidas para su solución. Son estas suposiciones o la forma de abordar esta ecuación las que establecen las diferencias entre los distintos algoritmos de construcción de mapas con robots.

Condición de Markov:

El modelo dinámico del robot determina la información que las acciones y los estados previos proporcionan sobre el estado actual. La formulación de este modelo se expresa con una función de probabilidad condicional,

$$P(x_t|x_1,\dots,x_{t-1},u_1,\dots,u_{t-1})$$

La condición de Markov sobre el modelo dinámico determina que el nuevo estado depende únicamente del estado y de la acción anterior. Esto es:

$$P(x_t|x_1,\dots,x_{t-1},u_1,\dots,u_{t-1}) = P(x_t|x_{t-1},u_{t-1})$$

La condición de Markov establece que el conocimiento de las acciones y posiciones previas $(u_1,\dots,u_{t-2},x_1,\dots,x_{t-2})$, no proporciona ninguna información adicional a la derivada de conocer la posición y acción inmediatamente previa [4].

2. ALGORITMOS PARA SLAM.

En este capítulo se presentan las principales técnicas existentes que dan solución al problema del SLAM. La primera de ellas es el Filtro de Kalman (FK), para aplicaciones lineales, continuando con el Filtro Extendido de Kalman (EKF) que no es más que la extensión del Filtro de Kalman para aplicaciones no lineales. Por último, el Filtro de Partículas utilizado para el seguimiento de características del entorno en secuencias de imágenes y que se basa al igual que el anterior en el Filtro de Kalman para estimar la posición del robot.

2.1 Filtro de Kalman.

El filtro de Kalman es un conjunto de ecuaciones matemáticas que proveen una solución recursiva eficiente del método de mínimos cuadrados[6] desarrollado por Rudolf E. Kalman.

Esta solución permite calcular un estimador lineal, insesgado[7] y óptimo del estado de un proceso en cada momento del tiempo con base en la información disponible en el momento t-1, y actualizar dichas estimaciones con la información adicional disponible en el momento t. Este filtro es el principal algoritmo para estimar sistemas dinámicos especificados en la forma de estado-espacio [5]. El filtro de Kalman estima el proceso anterior utilizando una especie de control de retroalimentación, esto es, estima el proceso a algún momento en el tiempo y entonces obtiene la retroalimentación por medio de los datos observados.

Desde este punto de vista las ecuaciones que se utilizan para derivar el filtro de Kalman se pueden dividir en dos grupos: las que actualizan el tiempo o *ecuaciones de predicción* y las que actualizan los datos observados o *ecuaciones de actualización*. Las del primer grupo son responsables de la proyección del estado al momento t tomando como

[6] El método de mínimos cuadrados es el procedimiento mas objetivo para ajustar una recta a un conjunto de datos representados en un diagrama de dispersión.
[7] Un estimador es insesgado si el valor de su esperanza coincide con el parámetro que se desea estimar.

referencia el estado en el momento t-1 y de la actualización intermedia de la matriz de covarianza del estado. El segundo grupo de ecuaciones son responsables de la retroalimentación, es decir, incorporan nueva información dentro de la estimación anterior con lo cual se llega a una estimación mejorada del estado [5].

Las ecuaciones que actualizan el tiempo pueden también ser pensadas como ecuaciones de pronóstico, mientras que las ecuaciones que incorporan nueva información pueden considerarse como ecuaciones de corrección. Efectivamente, el algoritmo de estimación final puede definirse como un algoritmo de pronóstico-corrección para resolver numerosos problemas. Así el filtro de Kalman funciona por medio de un mecanismo de proyección y corrección al pronosticar el nuevo estado y su incertidumbre y corregir la proyección con la nueva medida. Este ciclo se muestra en la figura 2.1 [5].

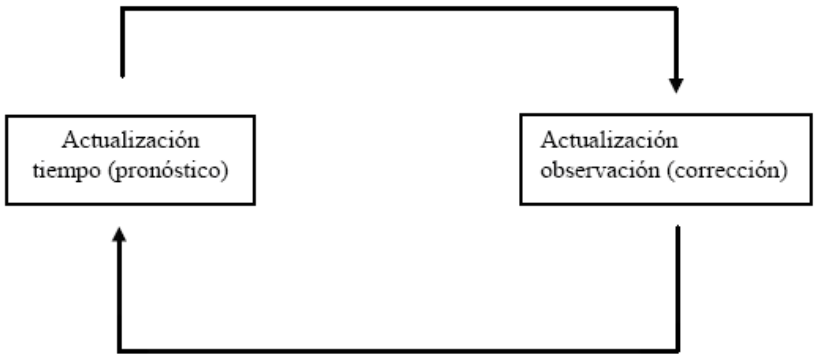

Figura 2.1. El ciclo del filtro de Kalman [5].

El primer paso consiste en generar un pronóstico del estado hacia adelante en el tiempo tomando en cuenta toda la información disponible en ese momento y en un segundo paso, se genera un pronóstico mejorado del estado, de tal manera que el error es minimizado estadísticamente [5].

El filtro de Kalman tiene como ventaja principal su aplicabilidad a diferentes problemas en inferencia estadística, pero su gran poder de aplicación es debido a la complejidad de sus ecuaciones lo que puede ser visto como una de sus principales desventajas [5].

El filtro de Kalman es una herramienta muy extendida en el área de la robótica, la

estimación con incertidumbre y la fusión sensorial[8]. La aplicación para sistemas no lineales se hace mediante la extensión del mismo en el que las ecuaciones son previamente linealizadas, y recibe el nombre de Filtro Extendido de Kalman (EKF).

2.2 Filtro Extendido de Kalman (EKF).

El EKF es utilizado en aplicaciones de Localización y Mapeo Simultáneos, para estimar la posición de un robot a partir de las ecuaciones de la odometría y de las medidas de los sensores. Mediante una etapa de predicción en la que se tiene en cuenta únicamente la odometría, y una etapa de corrección en la que se utilizan también las medidas de los sensores y las distribuciones de los errores, para encontrar la nueva posición [3].

El EKF tiene como objetivo obtener un estimador que se "acerque" lo suficiente en términos del error de estimación al proceso deseado o en pocas palabras que converja al proceso deseado en forma consistente[9]. Este trabaja con procesos que pueden ser descritos por una representación en variables de estado y por ecuaciones de estado y de observación, que rigen la dinámica del sistema.

Con este filtro se obtiene una función de densidad de probabilidad, que modela conjuntamente la posición de las características del entorno y la del vehículo, con una cierta incertidumbre. Esta incertidumbre viene determinada por el modelo de movimiento para el vehículo y por el modelo de observación de las características.

Aunque se trate de una solución muy extendida, la crítica más importante al EKF radica en su complejidad.

El EKF requiere para el cálculo de su matriz de covarianzas que los modelos de movimiento y de medida, que típicamente son no lineales, sean previamente linealizados. Generalmente se supone que esta linealización es aceptable, y que los errores cometidos son pequeños, lo que es asumido por la totalidad de las implementaciones existentes [3].

[8] La fusión sensorial combina lecturas de distintos sensores en una estructura de datos uniforme.
[9] Un estimador es consistente cuando el valor esperado del parámetro estimado tiende al valor real cuando el número de muestras tiende a infinito.

El SLAM-EKF tiene como ventaja fundamental que es capaz de mantener la estimación a posteriori completa explícitamente de forma incremental, a costa de realizar la suposición de que tanto el ruido del sistema odométrico como del sistema sensorial pueden ser aproximados por una distribución normal.

Para dar solución al problema del SLAM las técnicas más usadas son las basadas en el EKF (FILTRO EXTENDIDO DE KALMAN), debido a su fuerte respaldo teórico, lo que le brinda su fortaleza con respecto a otros filtros o estimadores.

En general en un desarrollo SLAM, las medidas de movimiento del vehículo y las medidas de las características del mapa son leídas en tiempo discreto k, por lo que es posible representar el estado del robot y las marcas observadas en el entorno de la siguiente manera:

$$x_{k+1} = f(x_k, u_k, v_k) \qquad (2.1)$$

$$z_k = h(x_k) + w_k \qquad (2.2)$$

El vector de estados $x_k \in \mathbb{R}^{m+dn}$ contiene la posición del vehículo $x_{r,k} \in \mathbb{R}^m$ en el tiempo k, y contiene un vector de n marcas d – dimensionales $x_f \in \mathbb{R}^{dn}$,

$$x_k = \begin{bmatrix} x_{r,k} \\ x_f \end{bmatrix} \qquad (2.3)$$

Para éste caso $m = 3$ por ser estos los tres factores tomados en cuenta para determinar la posición del robot, estos factores son la posición del robot en el eje x, la posición del robot en el eje y, y por último la orientación del robot θ. En cuanto a d el valor para esta aplicación es de 2, que corresponde a la posición de la marca en el eje x, y la posición de la marca en el eje y; Por lo anterior el vector de estados $\mathbf{x_k} \in \mathbb{R}^{3+2n}$, donde n es el número de marcas observadas está conformado por la posición del vehículo $x_{r,k} \in \mathbb{R}^3$ y el vector de características $\mathbf{x_f} \in \mathbb{R}^{2n}$ [19].

El vector de entrada $u_k \in \mathbb{R}^l$ es el comando de control del vehículo y $v_k \in \mathbb{R}^l$ es un vector aleatorio gaussiano con media cero y una matriz de covarianza $Q \in \mathbb{R}^{lxl}$, que representa dinámicas no modeladas del vehículo y el ruido del proceso. La función $\mathbb{R}^{m+dn} \rightarrow \mathbb{R}^{m+dn}$ es una posible ecuación diferencial no lineal que describe el movimiento del vehículo [32].

El vector aleatorio gaussiano $W_k \in \mathbb{R}^{dn}$ representa las inexactitudes de la posible no linealidad del modelo de medida **h:** $\mathbb{R}^{dn+m} \rightarrow \mathbb{R}^{dn}$ y el ruido en la medida con media cero y matriz de covarianza $\boldsymbol{R} \in \mathbb{R}^{dn \times dn}$.

Asumiendo que están disponibles un grupo de medidas $Z^i = \{z_1, \ldots\ldots, z_i\}$ para el cálculo del mapa actual estimado $x_{k|k}$, la expresión,

$$x_{k+1|k} = f(x_{k|k}, u_k, 0) \qquad (2.4)$$

Esta expresión genera una estimación a priori libre de ruido de las nuevas localizaciones del robot y de las características del mapa, más adelante se introduce al sistema un comando de control del vehículo u_k. De forma similar,

$$z_{k+1|k} = h(x_{k+1|k}) + 0 \qquad (2.5)$$

constituye una estimación a priori libre de ruido de las mediciones de los sensores [32].

La aproximación EKF al SLAM requiere de una linealización de la planta y de los modelos de movimiento. Dichas linealizaciones son formuladas como aproximaciones en series de Taylor con los términos de orden superior disminuidos,

$$x_{k+1} \approx x_{k+1|k} + F(x_k - x_{k|k}) + G_{vk} \qquad (2.6)$$

$$z_{k+1} \approx z_{k+1|k} + H(x_{k+1} - x_{k+1|k}) + w_{k+1} \qquad (2.7)$$

Las matrices Jacobianas $F, G\ y\ H$ contienen las derivadas parciales de f con respecto a x, y, al ruido del proceso v, y h con respecto a x, respectivamente [32]:

$$F = \left.\frac{\partial f}{\partial x}\right|_{(x_{k|k}, u_k, 0)} \tag{2.8}$$

$$G = \left.\frac{\partial f}{\partial v}\right|_{(x_{k|k}, u_k, 0)} \tag{2.9}$$

$$H = \left.\frac{\partial h}{\partial x}\right|_{(x_{k+1|k}, 0)} \tag{2.10}$$

Dado que las marcas del entorno son consideradas estacionarias, su estimación a priori es simple,

$$x_{f,k+1|k} = x_{f,k|k} \tag{2.11}$$

Por lo tanto, los elementos del modelo de transición lineal no-estacionario del vehículo y de las dinámicas del mapa en (2.6) y (2.7) toman la forma

$$\begin{bmatrix} x_{r,k+1} \\ x_f \end{bmatrix} = \begin{bmatrix} x_{r,k+1|k} \\ x_{f,k|k} \end{bmatrix} + \underbrace{\begin{bmatrix} F_r & \\ & I \end{bmatrix}}_{F} \begin{bmatrix} \tilde{x}_{r,k|k} \\ \tilde{x}_{f,k|k} \end{bmatrix} + \underbrace{\begin{bmatrix} G_r \\ 0 \end{bmatrix}}_{G} \begin{bmatrix} v_k \\ 0 \end{bmatrix} \tag{2.12}$$

$$z_{k+1} \approx z_{k+1|k} + \underbrace{\begin{bmatrix} H_r & H_f \end{bmatrix}}_{H} \begin{bmatrix} \tilde{x}_{r,k+1|k} \\ \tilde{x}_{f,k+1|k} \end{bmatrix} + w_{k+1} \tag{2.13}$$

El EKF requiere de representaciones gaussianas para todas las variables aleatorias que conforman el mapa (posición del vehículo y posición de marcas). Además, sus covarianzas deben ser pequeñas para poder aproximar todas las funciones no lineales a sus formas linealizadas.

En términos de las matrices de covarianzas, los pasos del filtro de Kalman son los siguientes: El primero es la predicción del estado a priori,

$$P_{k+1|k} = F P_{k|k} F^T + Q_k, \tag{2.14}$$

Donde P es la covarianza para el estado del robot (x, y, θ), *F* es la derivada de *f* con respecto a x, es decir, la Jacobiana del modelo de predicción y Q representa el ruido del proceso.

La innovación de la covarianza S esta dada por,

$$S_{k+1} = R_{k+1} + HP_{k+1|k}H^T, \tag{2.15}$$

Donde R es el ruido generado en la medición de las marcas, H es la derivada de h con respecto a x, es decir, la jacobiana del modelo de medida y P es la covarianza del estado del robot como se mencionó anteriormente.

Y finalmente la ganancia del filtro K,

$$K = P_{k+1|k}H_{k+1}^T S_{k+1}^{-1} \tag{2.16}$$

El objetivo del SLAM es estimar x_k usando el EKF que calcula recursivamente un estimador de mínimos cuadrados. La figura 2.2 muestra una representación del algoritmo SLAM [19]. El estimador de mínimos cuadrados es designado por el estado estimado del sistema $x_{k|k} = \left[x_{r,k|k}, x_{f,k|k}\right]^T$, y su matriz de covarianzas de estado,

$$P_{k|k} = \begin{bmatrix} P_{r,k|k} & P_{rf,k|k} \\ P_{rf,k|k} & P_{f,k|k} \end{bmatrix} \tag{2.17}$$

Donde $x_{r,k|k}$ y la covarianza $P_{r,k|k}$ hacen referencia a la parte del estado estimado del vehículo y donde $x_{f,k|k} = \left[x_{f,k|k}^1, \dots, x_{f,k|k}^i\right]^T$ y su covarianza $P_{f,k|k}$ hacen referencia a la parte del estado estimado del mapa y el término $P_{rf,k|k}$ es la covarianza cruzada entre el vehículo y el mapa como se muestra en la figura 2.2.

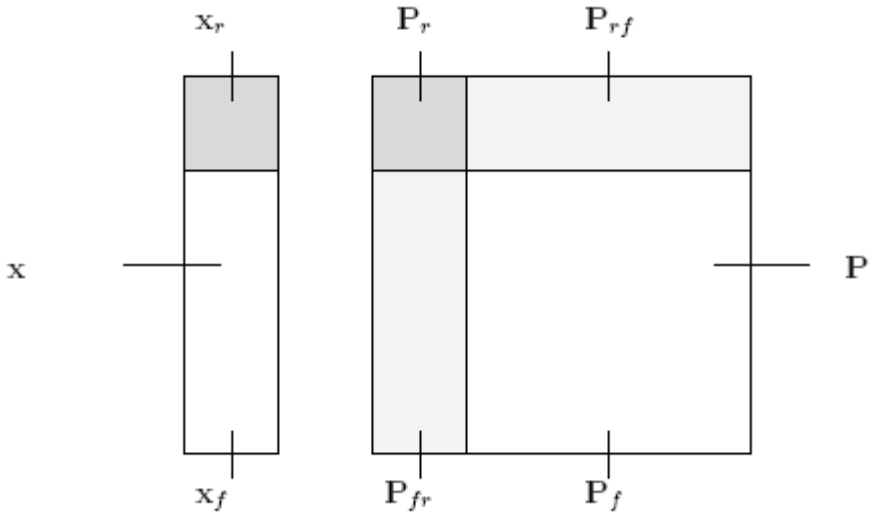

Figura 2. 2. Representación del vector de estados y la matriz de covarianzas del EKF [19].

Introducción de nuevas marcas

Dado un grupo de medidas i, $Z^i = [z_1; z_2; \dots z_i]$ se desea inicializar una nueva entidad x_f^i (marca) en un mapa modelado estocásticamente con una media $x_{k|k}$ y la matriz de covarianza $P_{k|k}$ [19].

La inicialización consiste en apilar cada posición de cada nueva marca x_f^i en el mapa como sigue:

$$x^+ = \begin{bmatrix} x \\ x_f^i \end{bmatrix} \qquad (2.18)$$

Y definiendo la *fdp* (función de densidad de probabilidad) de este nuevo estado condicionado a la medida z_i. Considerando el caso de un modelo de medida dado, una expresión implícita que relaciona una nueva variable m_f es definida como sigue,

$$h'(x_r, x_f, m_f) = 0 \qquad (2.19)$$

la medida asociada a la nueva marca, donde $z = m_f + v_k$. La solución h' para valores fijos de x_r y m_f, para obtener una expresión explicita para una nueva marca [19].

$$x_f = j(x_r, m_f) \tag{2.20}$$

Note que para el rango y el ángulo medido, la solución de esta expresión es un punto, se asume que x_f es aproximadamente una variable aleatoria gaussiana con media y matrices de covarianzas definidas por,

$$x_{f,k|k} = j(x_{r,k|k}, z) \tag{2.21}$$

$$P_f = J_r P_r \tag{2.22}$$

$$P_{ff} = J_r P_{rr} J_r^T + J_m P_{rr} J_m^T \tag{2.23}$$

Donde $P_r = [P_{rr} \quad P_{rf}]$ y $J_r = \frac{\partial j}{\partial x_r}$ y $J_m = \frac{\partial j}{\partial m_f}$. El mapa aumentado esta finalmente especificado por,

$$x_{k|k}^+ = \begin{bmatrix} x_{k|k} \\ x_{f,k|k}^i \end{bmatrix} \qquad P_{k|k}^+ = \begin{bmatrix} P_{k|k} & P_{f,k|k}^T \\ P_{f,k|k} & P_{ff,k|k} \end{bmatrix} \tag{2.24}$$

Algoritmo del filtro de Kalman

A continuación se realiza una breve descripción de los pasos que se deben seguir para desarrollar SLAM a partir del Filtro Extendido de Kalman.

Paso 0: *Extracción de marcas del entorno.*

En este paso se extrae la posición (x, y)de cada una de las marcas observadas en cada posición donde se encuentra el robot.

Asociación de datos.

A medida que se van observando las marcas del entorno, se realiza una comparación de cada nueva marca con las marcas almacenadas en el vector de estados, con el fin de diferenciar las marcas nuevas de las ya existentes en el entorno.

Paso 1: Actualización del estado mediante datos de odometría.

Se actualiza el estado actual del robot posición (x, y, θ) mediante la información entregada por el sistema odométrico del robot. A continuación se actualiza la jacobiana del modelo de predicción A; el ruido del proceso Q, y finalmente la covarianza para la posición del robot P_{rr}, además de la correlación cruzada entre robot-marcas P_{ri}.

Paso 2: Actualización del estado mediante marcas re-observadas.
Se calcula la jacobiana del modelo de medida H, el ruido en la medida R, con lo anterior se halla la ganancia de Kalman y finalmente se re-calcula el nuevo vector de estados usando la ganancia de Kalman.

Paso 3: Adición de nuevas marcas al mapa.
En este paso se actualiza la información correspondiente a las posiciones(x, y) de cada nueva marca en el vector de estados, se calcula la covarianza para cada nueva marca P^{N+1N+1}, luego la covarianza robot-marca para la nueva marca P^{rN+1}, la covarianza contraria a esta P^{N+1r}, las covarianzas entre marca-marca P^{N+1i}, y por último la contraria de esta P^{iN+1}.

Dentro de sus desventajas se encuentra, que el coste computacional del SLAM EKF crece al cuadrado con el número 'n' de objetos del mapa siendo $O(n^2)$, lo que significa que no puede ser aplicado para mapas arbitrariamente grandes.

Además la asociación de datos, encargada de realizar la correspondencia entre las observaciones realizadas y los objetos existentes en el mapa, no queda resuelta por el algoritmo. Esta asociación de datos debe ser realizada en el instante en el que se realizan las medidas, con la estimación del mapa disponible en ese momento, por lo que la correspondencia puede llegar a ser bastante incierta, teniendo en cuenta que una asociación de datos errónea lleva casi sin lugar a dudas a la divergencia del mapa. En entornos de interiores las características geométricas del entorno (paredes, marcas) suelen estar suficientemente diferenciadas para no presentar el problema de la ambigüedad en las observaciones [3].

Con el fin de combatir algunas de estas desventajas surge el denominado FAST SLAM o Filtro de Partículas, descrito a continuación.

2.2 Filtro de Partículas FAST SLAM.

Este filtro fue propuesto por N. Gordon, D. Salmond y A. Smith en 1993, como *Filtro Bootstrap* para implementar *Filtros Bayesianos recursivos* [10]. El propósito general del algoritmo del filtro de partículas es representar funciones de densidad de probabilidad (FDP) y su evolución en el tiempo [6]. Estas representaciones están basadas en muestras discretas de las funciones que son modeladas; estas muestras discretas se denominan partículas, y están formadas por un estado (x) y un peso (π)[7]. Estas partículas son estados posibles del proceso, que se pueden representar como puntos en el espacio de estados de dicho proceso.

De otra parte, todos los algoritmos existentes son propuestas similares que propagan las partículas utilizando el modelo de movimiento $P(x_t|x_{t-1},u_t)$ y el modelo de medida $P(z_t|x_t)$.

La evolución temporal de las partículas se lleva a cabo a través de un procedimiento iterativo de:

- Actualización
- Predicción

Como lo indica la figura 2.3.

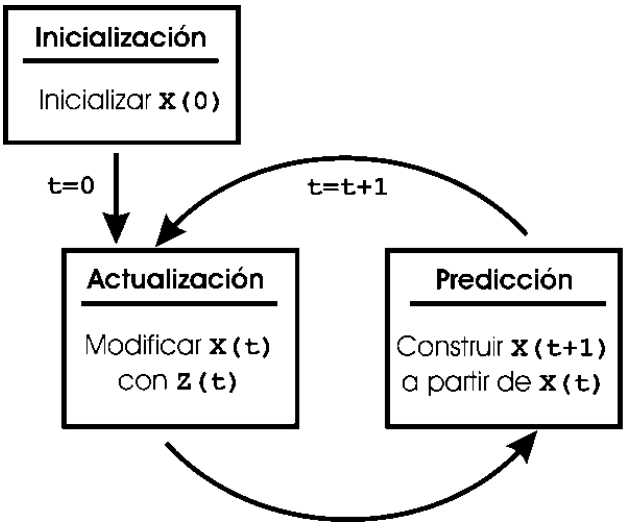

Figura 2. 3. Evolución temporal de las partículas [6]

En concreto, el *Filtro de Partículas* representa la densidad a posteriori mediante un conjunto discreto de N partículas $(m_1, \dots, m_N)$ y sus probabilidades asociadas $(\pi_1, \dots, \pi_N)$ [6].

Inicialmente, el conjunto de partículas se escoge a partir de la distribución a priori $p(x_0)$. Si no existe información a priori, entonces las partículas se distribuyen uniformemente por el espacio de estados. Posteriormente, en cada instante de tiempo t, se actualizan las N partículas en función de la acción anterior u_{t-1} y la observación actual Z_t. Para ello, se aplica el modelo de movimiento $P(x_t|x_{t-1},u_{t-1})$ a cada una de las N partículas, generando un nuevo conjunto de partículas. Las partículas nuevas representan la predicción de la variable de estado, sin considerar la observación, se obtiene el peso π^i asociado a cada partícula.

El conjunto de pesos de cada partícula es proporcional a la probabilidad de su estado y a la suma normalizada de sus pesos. La densidad de los pesos es igual al producto de la densidad previa (del muestreo) y la probabilidad (de los pesos).

En La Figura 2.4 se representa un muestreo discreto de una función de densidad de probabilidad continua mediante partículas, cuyo tamaño hace referencia al peso asignado a las mismas [6].

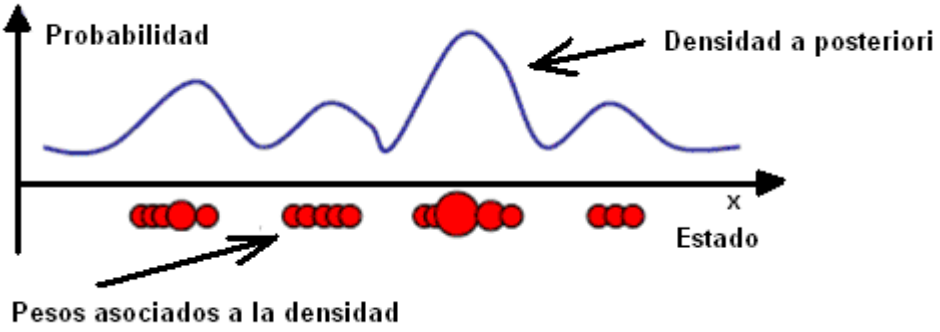

Figura 2. 4. Representa el muestreo y los pesos obtenidos al aplicar el Filtro de Partículas [6].

En un último paso, se remuestrea el conjunto de partículas, extrayendo (con reemplazo) N partículas del conjunto actual, proporcional al peso de cada una. En este nuevo conjunto tendrán más probabilidad de desaparecer aquellas partículas para las que no hay evidencia de verosimilitud o, lo que es lo mismo, que tenga menor peso. Una vez construido el nuevo conjunto de partículas, según la probabilidad de éstas se asocia un peso a cada una. Este nuevo conjunto de partículas constituye una representación muestral de la probabilidad a posteriori. En la figura 2.5 se puede observar el comportamiento de un filtro de partículas [10].

En la figura 2.5, se puede observar una población de partículas ponderadas uniformemente x_t^i, N^{-1}, que aproximan la densidad $P(x_t|z_{t-1})$. En este momento se reciben nuevas medidas (z_t) y se calcula el peso para cada partícula, que involucra la función de densidad de probabilidad $P(z_t|x_t)$.

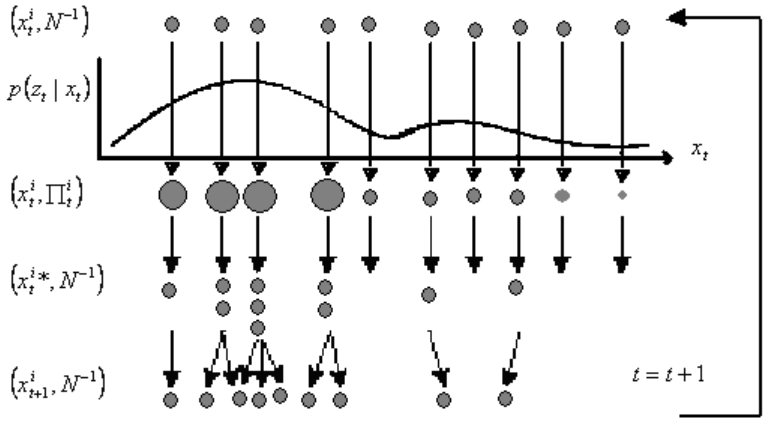

Figura 2. 5. Comportamiento del filtro de partículas [7].

El resultado es un conjunto de partículas con pesos asociados (x_t^i, π_t^i) que constituyen una aproximación discreta de $P(x_t|z_t)$. A continuación, se lleva a cabo el remuestreo.

En este paso, se seleccionan las partículas con mayor peso para obtener un conjunto de partículas $(x_t^i *, N^{-1})$ ponderadas uniformemente. El último paso es la predicción, cuyo objetivo es adaptar el conjunto de partículas al nuevo instante, aproximando así la *fdp* $P(x_{t+1}|z_t)$.

Por lo tanto, los filtros de partículas son algoritmos que manipulan la evolución de un conjunto de partículas a través del tiempo. Las partículas se mueven de acuerdo a un modelo de movimiento y sobreviven con una probabilidad proporcional a su peso [7].

A continuación, se describen los pasos a seguir para desarrollar el filtro de partículas al problema del SLAM.

Algoritmo del Filtro de Partículas.

Paso 0: Se parte de un conjunto ponderado de partículas con igual peso $1/N$.

Paso 1: Se propaga cada partícula m utilizando el modelo de movimiento F del objeto para obtener un conjunto actualizado de partículas $\{m_i^*\}$.

Paso 2: Se Obtiene un nuevo vector de medidas Z y evaluar la densidad de probabilidad a posteriori de cada partícula:

$$p(m_i^*|Z) = \frac{p(Z|m_i^*)p(m_i^*)}{p(Z)}$$

donde, $p(Z)$ es la probabilidad a priori de la medida que se asume constante y conocida y $p(m_i^*) = \frac{1}{N}$

Paso 3: Se vuelve a muestrear a partir del conjunto $\{m_i^*\}$ con probabilidades π_i^* y generar un nuevo conjunto bien ponderado $\{m_i'\}$ con pesos iguales $(1/N)$ para cada partícula. El muestreo de cada partícula consiste en un estado, peso y covarianza (opcional).

Paso 4: Se Repiten los pasos 1 y 3 [6].

En la figura 2.6. Se observa la evolución y el comportamiento dinámico del Filtro de Partículas.

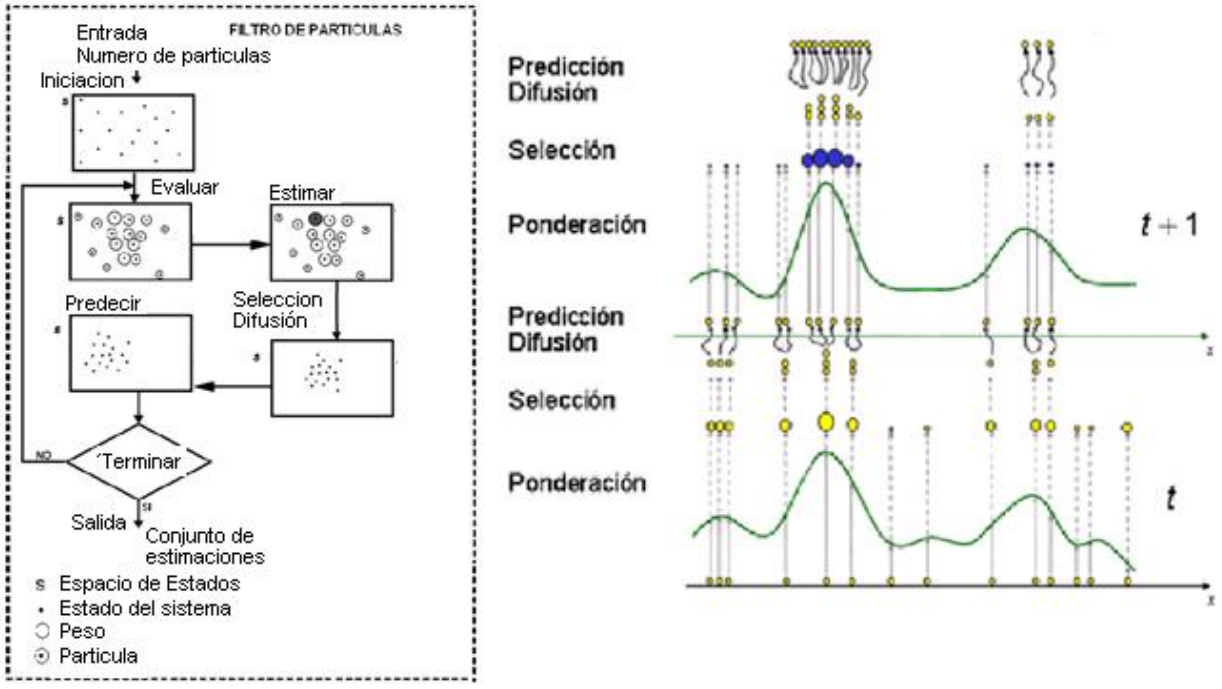

Figura 2. 6. Comportamiento del filtro de partículas diagrama de flujo (izquierda), a nivel de estados con cada una de sus etapas (derecha) [13].

Ejemplo de aplicación al seguimiento visual.

Suponiendo que se desea seguir un móvil a través de una secuencia de imágenes, utilizando un Filtro de Partículas como se indica en la figura 2.7:

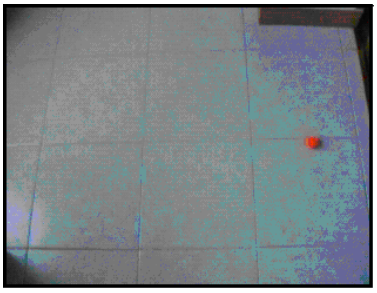

Figura 2. 7. Seguimiento de un móvil a través de una secuencia de imágenes

Modelo de partícula

Una partícula está formada por un estado (x) y un peso (π) (figura 2.8). En seguimiento visual de un móvil, una partícula será entonces:

- Estado: (x, y)
- Peso: (π)

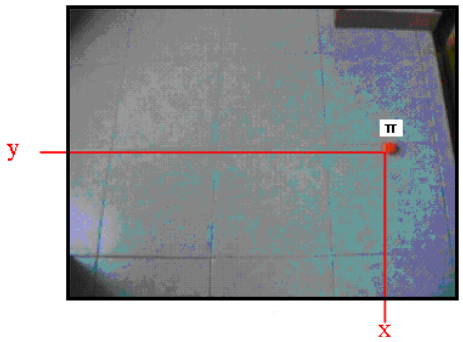

Figura 2. 8. Modelo de partícula.

Modelo de medida

Se necesita encontrar una característica del objeto que lo distinga del resto. En este caso, se ha comprobado que el color del objeto cumple la siguiente condición (figura 2.9): Canal R >120 & canal G<100 & canal B<100

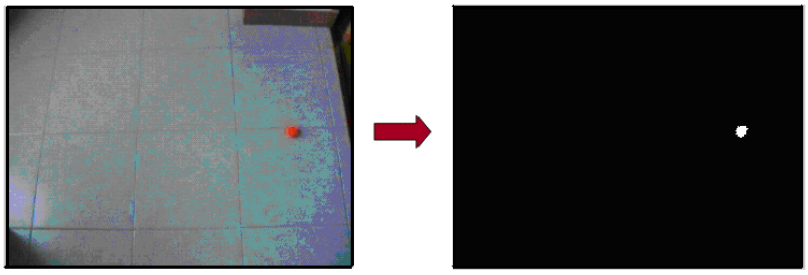

Figura 2. 9. Cuadro del objeto (izquierda), Característica propia del objeto (derecha)

El algoritmo del filtro de partículas se puede describir de la siguiente manera:

- Inicializar: muestrea una población inicial de N partículas de una distribución inicial (figura 2.10).

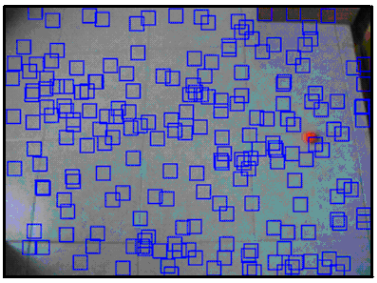

Figura 2. 10. Muestreo de una población inicial de N partículas.

- Evaluar: asigna un peso a cada partícula en función de su correspondencia con la medida (figura 2.11).

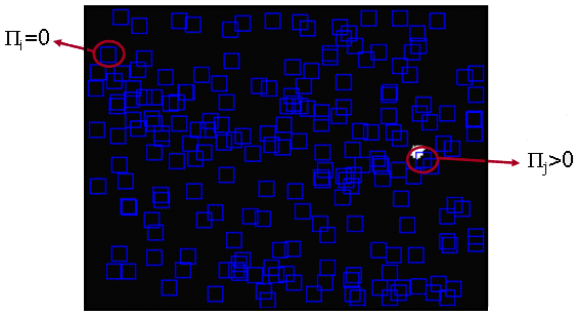

Figura 2. 11. Asignación de peso a cada una de las N partículas.

- Elegir: elige las partículas que formarán la población en el siguiente instante. La probabilidad de elegir una partícula es proporcional a su peso (figura 2.12).

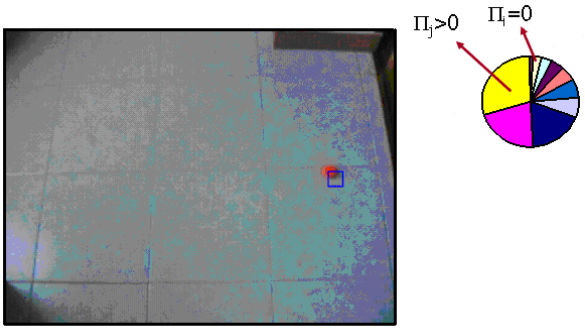

Figura 2. 12. Elección de las partículas con mayor probabilidad.

- Difundir: como cada partícula puede ser elegida más de una vez, se aplica un desplazamiento aleatorio a cada una para evitar coincidencias (figura 2.13).

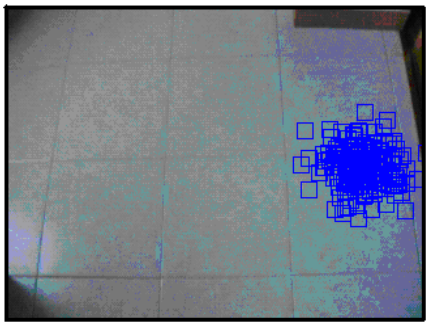

Figura 2. 13. Desplazamiento aleatorio de cada partícula para evitar ser elegida nuevamente.

- Inicializar Movimiento: aplica el modelo del sistema para construir el conjunto de partículas en el instante siguiente (figura 2.14).

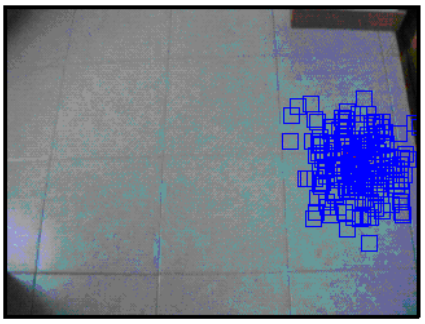

Figura 2. 14. Aplicación del modelo del sistema para construir el conjunto de partículas en el instante siguiente.

En el ejemplo anterior se puede observar que los filtros de partículas son algoritmos que permiten llevar a cabo seguimiento de objetos en secuencias de imágenes, además el mecanismo de predicción-actualización modela la evolución de los objetos móviles en la escena. Es posible encontrar un valor estimado del estado del sistema para cada instante. En el caso de seguimiento, este estado viene dado por las coordenadas del objeto [x, y].

El funcionamiento del filtro de partículas permite centrar la atención en una región de la imagen donde es probable encontrar al objeto, despreciando la exploración del resto [13].

Una vez estudiadas las principales técnicas existentes para llevar a cabo SLAM, se procede a dar una justificación de porque se utiliza el Filtro Extendido de Kalman para desarrollar el sistema de mapeado y localización simultánea expuesto en este trabajo de grado.

El Filtro Extendido de Kalman puede ser visto como una variante de un filtro bayesiano. El EKF brinda una estimación recursiva del estado de la dinámica del sistema o precisamente, resuelve un problema de estimación no lineal. El estado del sistema X en un tiempo t puede ser considerado como una variable aleatoria donde la incertidumbre de este estado puede ser representada por una distribución de probabilidad.

El FastSLAM parte del hecho que se conoce el recorrido del robot por lo que las medidas individuales de las marcas se hacen independientes del recorrido del robot. El FastSLAM descompone el problema del SLAM en un problema de localización del robot, y una colección de k problemas de estimación de marcas.

Básicamente, el EKF SLAM y el FastSLAM resuelven el mismo problema haciendo uso de modelos de movimiento y de medida similares, ambos usan el Filtro de Kalman: el EKF SLAM aplica el filtro una vez a un gran problema de mapeo y localización; mientras el FastSLAM aplica pequeños EKFS a cada partícula. En este orden de ideas, se decide utilizar el EKF SLAM para desarrollar la aplicación propuesta en este trabajo de grado, por ser este filtro la base del FastSLAM. Con el estudio e implementación de una aplicación EKF, se dan a conocer los fundamentos del funcionamiento de esta metodología para la resolución del problema SLAM, lo que deja abierta la posibilidad de realizar futuras implementaciones basadas en éste filtro como lo hace el FastSLAM.

3.DISEÑO E IMPLEMENTACIÓN

En este capítulo se describe el proceso de implementación de la aplicación SLAM basada en el EKF, tomando como base el documento “SLAM for Dummies” y en técnicas de visión artificial.

3.1 Hardware Utilizado

El hardware del robot es muy importante. A continuación se describe tanto el robot como el dispositivo de medición de características del entorno, necesarios para llevar a cabo el proceso de SLAM.

Para llevar a cabo la implementación del sistema SLAM, se utilizó el robot ATIBOT (figura 3.1) construido por el grupo de investigación en robótica ATI de la Facultad de Ingeniera Electrónica y Telecomunicaciones de la Universidad del Cauca.

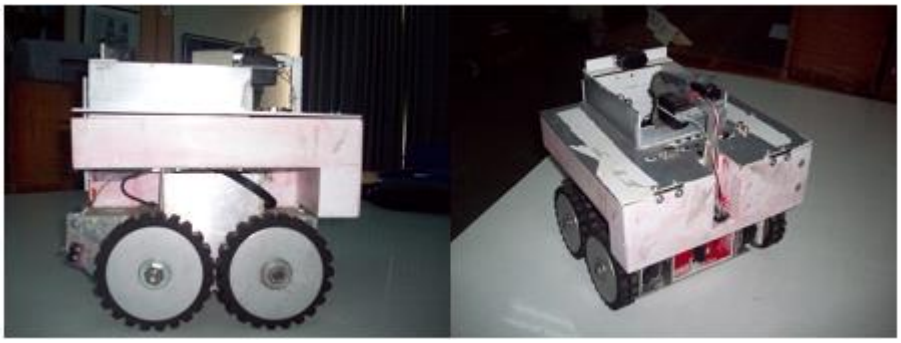

Figura 3.1. Robot ATIBOT.

La figura 3.2 muestra el robot ATIBOT con sus respectivas dimensiones.

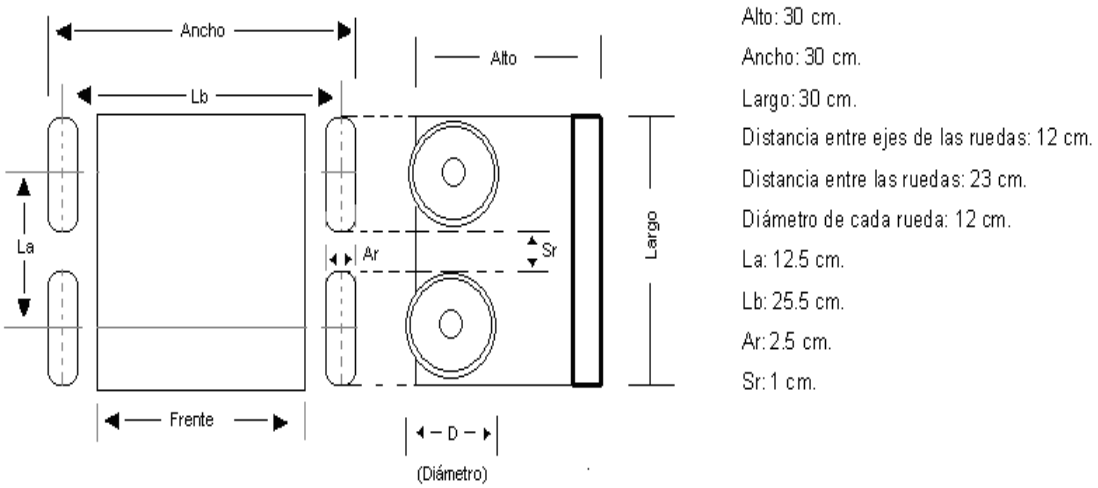

Figura 3.2. Dimensiones del Robot ATIBOT [31].

.

En la configuración del robot es muy importante la relación geométrica *La / Lb* ya que esta establece la facilidad de girar del vehículo, y depende tanto de parámetros geométricos como físicos del robot. La relación establecida para obtener un buen desempeño de giro es de 0.5, la cual proporciona una buena distribución de masa entre los cuatro puntos de contacto y permite la utilización sin mayores dificultades de ruedas comerciales. Dicha relación establece las dimensiones del chasis [31].

El robot posee una plataforma de locomoción "*skid steering*" de cuatro ruedas y una arquitectura de direccionamiento diferencial, la cual emplea la diferencia de velocidades de las ruedas del lado izquierdo con respecto a las del lado derecho, para realizar todos sus movimientos [31].

En skid steering se disponen varias ruedas en cada lado del vehículo que actúan de forma simultánea. El movimiento es el resultado de combinar las velocidades de las ruedas de la izquierda con las de la derecha [29]. Las ruedas del mismo lado del chasis deben girar con igual sentido y velocidad y no cambian su orientación con respecto al chasis (figura 3.3).

El direccionamiento diferencial viene dado por la diferencia de velocidades de las ruedas laterales. La tracción se consigue también con éstas mismas ruedas. Dos ruedas montadas en un único eje son independientemente propulsadas y controladas, proporcionando ambas tracción y direccionamiento. Este sistema es muy útil si se

considera la habilidad de movimiento del móvil, presentando la posibilidad de cambiar su orientación sin movimientos de translación, lo que se podría llamar cambio de *spin*. [29].

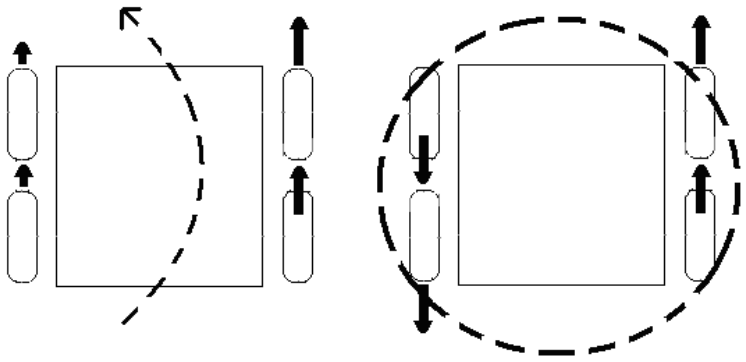

Figura 3. 3. Movimientos *skid steering* de la arquitectura mecánica del robot ATIBOT [31].

Para que el robot ATIBOT pueda disponer de buena carga útil, éste cuenta con actuadores que le proporcionaran buena tracción. Para tal propósito se emplean moto-reductores SPAL, motores DC de gran capacidad.

Cada rueda se une por medio de un sistema de buje con un *encoder* incremental y su respectivo moto-reductor [31]. El sistema "actuador" así formado es totalmente independiente y se une muy fácilmente a través de tornillos en el espacio destinado a cada actuador en el chasis.

El robot ATIBOT, cuenta con *encoders* incrementales como sensores de odometría, para poder determinar su posición. Los codificadores ópticos o *encoders incrementales* se utilizan fundamentalmente para el cálculo de la posición angular. Básicamente constan de un disco transparente, el cual tiene una serie de marcas opacas colocadas radialmente y equidistantes entre si; de un elemento emisor de luz (como un diodo LED); y de un elemento fotosensible que actúa como receptor [33]. El eje cuya posición angular se va a medir va acoplado al disco. En la figura 3.4 se muestran los componentes de un *encoder* incremental.

Cuando el sistema comienza a funcionar, el emisor de luz empieza a emitir; a medida que el eje vaya girando, se producirán una serie de pulsos de luz en el receptor, correspondientes a la luz que atraviesa los huecos entre las marcas. Llevando una cuenta de esos pulsos es posible conocer la posición del eje. Las medidas entregadas por el

sistema odométrico permiten calcular su posición, a través de la rotación de sus ruedas [33].

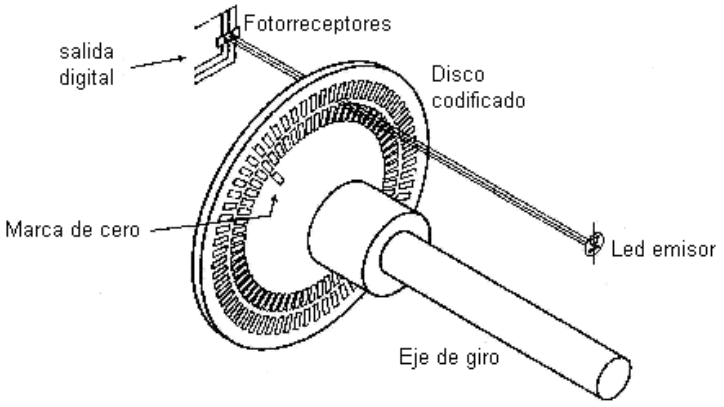

Figura 3.4. Encoder incremental [33].

3.2 Software Utilizado

El proyecto de mapeado y localización simultanea (SLAM) se desarrolla utilizando la plataforma .NET y su herramienta Visual Studio la cual contiene al lenguaje de programación C# (C Sharp).NET.

La plataforma .NET es una capa de software que se coloca entre el Sistema Operativo (SO) y el programador (figura 3.5) y que abstrae los detalles internos del SO [25].

El lenguaje de programación *Visual C# .NET (C Sharp),* introducido por Microsoft en la plataforma .NET, es el que mejor se adapta a la plataforma ya que ha sido exclusivamente creado para trabajar sobre ella. Sus principales creadores son Scout Wiltamuth y Anders Hejlsberg [25].

Es un lenguaje de programación que toma las mejores características de lenguajes preexistentes como Visual Basic, Java o C++ y las combina en uno solo; por lo cual su sintáxis y estructuración es muy parecida a la de estos lenguajes de programación [26].

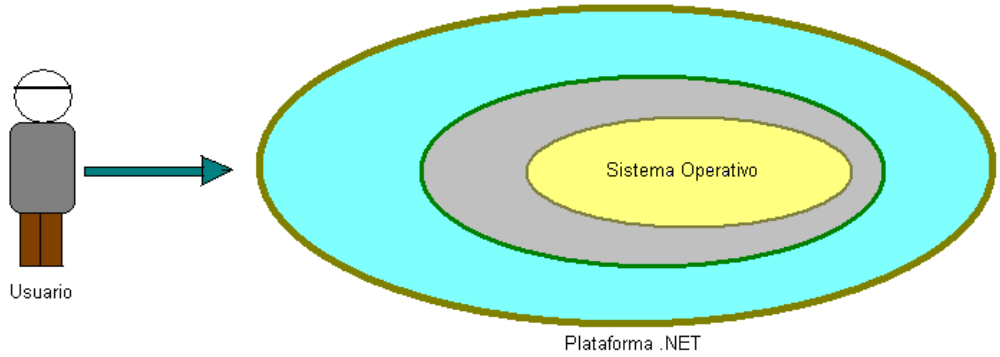

Figura 3.5. La plataforma .NET abstrae al programador de SO [25].

3.3 Datos del sistema de Visión

El primer paso en el proceso SLAM, es obtener información sobre el entorno en el cual se desplaza el robot. El método más común para realizar estas labores es utilizar un *scanner Laser*. Un ejemplo de este tipo de sensores es el *scanner laser SICK*, que se puede configurar para que entregue medidas en un ángulo de 100º o 180º con una resolución de 0,25º, 0,5º o 1º. La salida típica de un scanner de este tipo puede ser: 2,98, 2,99, 3,00, 3,01, 3,00, 3,49, 3,50,…2,20, 8,17, 2,21, siendo estas medidas de distancia correspondientes a cada ángulo. La figura 3.8 ilustra el método de medición más común de un *scanner laser*.

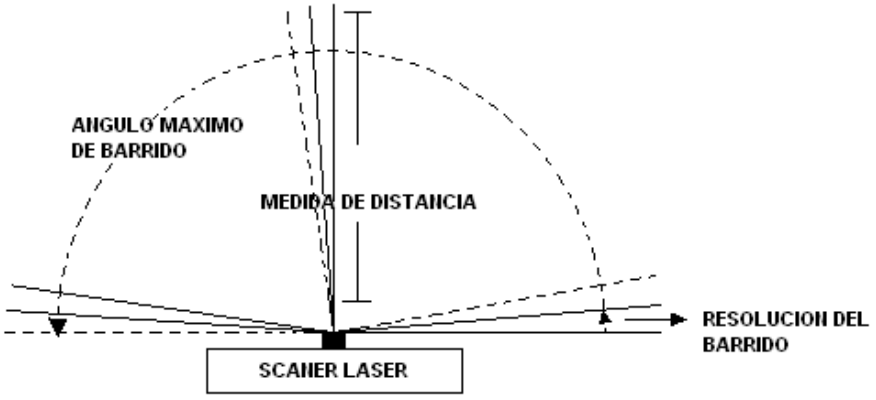

Figura 3.8. Mediciones entregadas por un Scanner Láser.

La salida de un *scanner láser* entrega medidas de distancia de derecha a izquierda en unidades de metros. Si el *scanner láser* por alguna razón no puede entregar la medida

exacta de distancia a un ángulo dado, este retorna un valor alto para indicar que ha ocurrido un error en la medida de distancia.

Los *scanner láser* son muy rápidos, usando el puerto serial de un computador para recibir la información, estos puede trabajar hasta una frecuencia de 11Hz.

A pesar de ser el *scanner láser* el método más común para realizar las mediciones del entorno en el proceso de SLAM, uno de los objetivos principales de este trabajo de grado es realizar el proceso de SLAM utilizando un sistema de visión, lo que implica la utilización de una cámara como elemento principal de medición para realizar estas labores de extracción de características del entorno.

El algoritmo SLAM se alimenta con la información extraída de las imágenes del entorno tomadas por una cámara web, se procesa cada imagen para detectar las marcas presentes dentro del ángulo de visión de la cámara y a partir de su posición dentro de la imagen y su tamaño, se calcula el ángulo al cual se encuentra la marca, con respecto a la cámara y la medida de distancia desde la cámara hasta la marca respectivamente. La figura 3.9 muestra el tipo de información que se requiere de la imagen capturada por la cámara web.

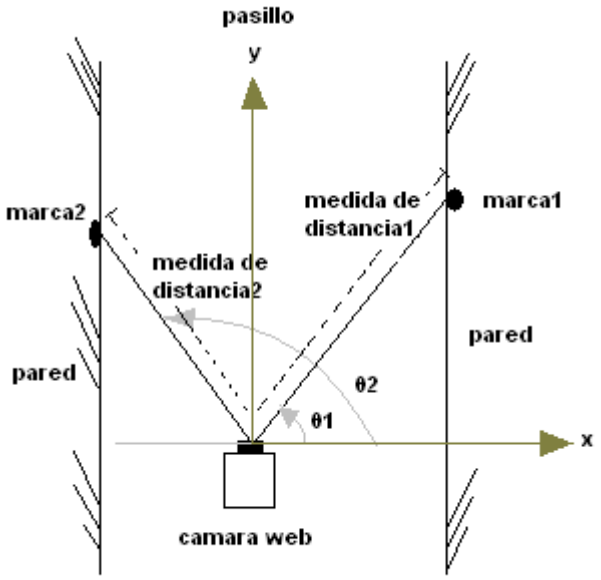

Figura 3.9. Información entregada por la cámara web.

De este modo se garantiza la entrega de la información del entorno mediante una cámara, de la misma manera como se entregaría si se contara con sensores de tipo *scanner láser*, en términos de ángulo y medida de distancia.

3.4 Datos de Odometría

Otro aspecto muy importante en el proceso de SLAM, son los datos de odometría. El objetivo de los datos de odometría es brindar una posición aproximada del robot mediante la medida de movimiento de las ruedas del mismo, estos datos son usados en el primer paso del EKF.

Para el desarrollo del sistema SLAM, la información de odometría relevante que debe suministrar el robot ATIBOT es su orientación y su desplazamiento. Es importante mencionar que dicha información debe ser adquirida cada cierto intervalo de tiempo establecido a través de una interrupción generada por la aplicación.

La adquisición de dicha información se realiza de la siguiente manera:

1- Se genera una interrupción por software desde la aplicación en C# hacia el microcontrolador del robot para establecer la petición de la información.

2- Se verifica la petición de la información.

   a. Si la petición es aceptada:

      i. Se envía un bit de cabecera para indicar que no existe error en la comunicación en éste caso se envía un uno (1).

      ii. Una vez establecida la comunicación se envía un byte (8 bits) con la información de la orientación del robot, compuesto de dos partes importantes, por un lado el estado (4 bits más significativos) y por otro la orientación del giro (4 bits menos significativos) de acuerdo a la siguiente codificación.

Estado	Orientación
a) Procesando: 1100	a) 90 grados _ derecha: 1100
b) Completado: 0011	b) 90 grados _ izquierda: 0011
	c) No _ giro: 1111

Ejemplo de orientación:

11001100: vehículo girando 90 grados_Derecha.

00111100: vehículo giró 90 grados_Derecha.

00111111: no se ha realizado giro.

iii. En otros dos bytes se envía la información de su desplazamiento (en centímetros) con respecto al eje X y al eje Y

b. Si la petición es rechazada, se debe esperar para volver a intentarlo.

Es importante aclarar que la información de orientación y desplazamiento entregada por el microcontrolador del robot fue generada y proporcionada por el grupo de desarrollo en robótica (ATI) de la Facultad de Ingeniería Electrónica y Telecomunicaciones.

3.5 Marcas del Entorno

En el proceso de definir el tipo de marca ideal que facilite el proceso de mapeo del entorno sobre el cual se desplaza el robot, primero se realizaron pruebas preliminares con marcas de tipo rectangular pegadas en las paredes del entorno con el fin de determinar la calidad de la información generada con este método, como se mencionó anteriormente, el objetivo era entregar la información de manera similar a la suministrada por un *scanner láser* pero utilizando una cámara web. En la figura 3.13. Se muestra un esquema de éstas primeras pruebas.

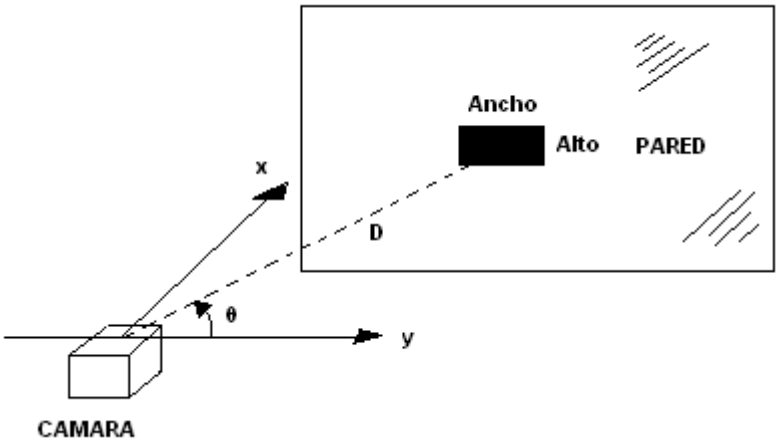

Figura 3.13. Medición de una marca rectangular.

Para inferir la distancia desde la cámara hasta la marca, se calculó el área mediante el conteo de píxeles de la imagen capturada por la cámara. Para esta aplicación se definió que el tamaño de la imagen seria de 320x240 píxeles fijos y como el área de la imagen varía al acercar o alejar la cámara de la marca, fue posible inferir mediante la medición del área de la marca la distancia existente entre cámara y marca. La figura 3.14, ilustra el procedimiento para calcular la distancia entre una marca y la cámara, a partir de una imagen.

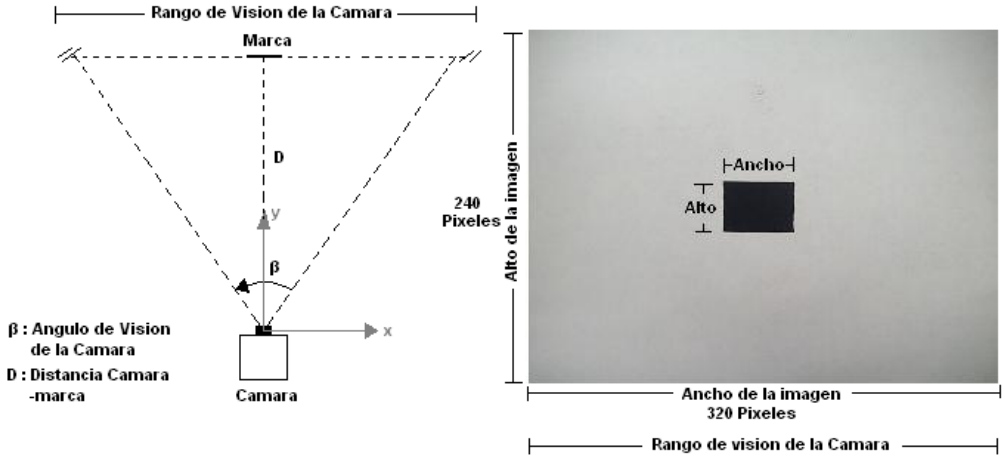

Figura 3. 14. Marca extraída del Entorno.

El problema principal de usar marcas rectangulares pegadas en la pared, es que el método para calcular distancias funciona siempre y cuando la marca siempre se mire paralela al eje x de la cámara como se muestra en la imagen anterior, si la marca se mira

con la cámara inclinada en un cierto ángulo, el área de la marca varia a medida que cambia la perspectiva con que se mira. Como el área es el dato en el que se basa el cálculo de la distancia, una marca rectangular no serviría porque el área en píxeles de una marca a una distancia fija varia con respecto a la perspectiva con que se mire. Este inconveniente se ilustra en la figura 3.15.

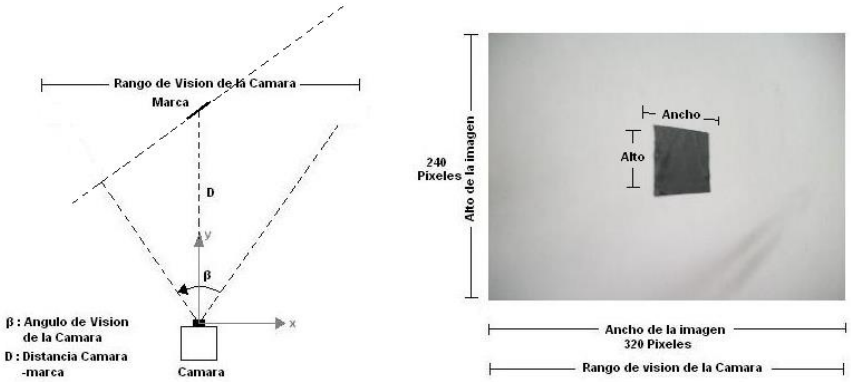

Figura 3. 15. Marca inclinada. Variación del área de la marca con la perspectiva con que se mira.

En la figura anterior se observa la diferencia que se presenta en el área de la marca con respecto al área de la marca de la figura 3.14, a pesar de que ambas se encuentran a igual distancia. Debido a este problema, se buscó otro tipo de marca que no variara con respecto a la perspectiva con que se mire. La figura geométrica que cumple con estas características es la esfera y es por ello que se decidió que las marcas puestas en el entorno serían de forma esférica.

Debido a que la cámara representa el mundo 3D en un plano 2D, la representación de una esfera en 2D sería una circunferencia y el área de ésta no varía con respecto a la perspectiva con que se mire. Esto hace a la esfera como la marca más indicada para ésta aplicación. En la figura 3.16 se ilustra la esfera como el tipo de marca elegida para el muestreo del entorno.

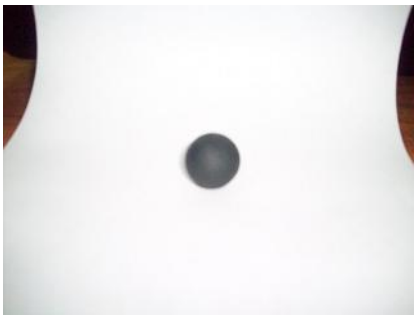

Figura 3.16. Marca esférica. La marca esférica es capturada por la cámara como una circunferencia que no varía su área con respecto a la perspectiva con que se mire.

3.6 Extracción de marcas

El objetivo del proceso de extracción de marcas es poder determinar la ubicación en un plano x, y global cada una de las marcas captadas por la *webcam*.

Antes de poder realizar el proceso de extracción de marcas del entorno es necesario preparar la imagen aplicando algunas técnicas de procesamiento digital de imágenes para mejorar las marcas presentes en cada toma del entorno. Debido a que el ambiente fue definido desde el inicio como estructurado y con características controladas, los pasos requeridos en el proceso de mejoramiento de la imagen y detección de objetos fueron sólo 5: *mejoramiento de contraste*, *conversión a escala de grises*, *umbralización*, *filtrado de ruido* y *segmentación*.

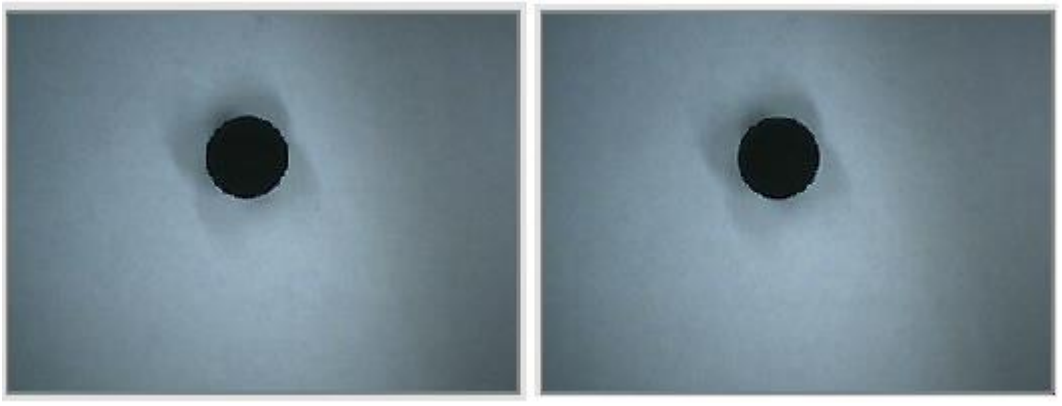

Figura 3.17. Captura de una imagen. En el lado izquierdo se observa la señal en vivo de la webcam y al lado derecho la imagen capturada sin procesamiento alguno.

El proceso de captura de la imagen, se realiza almacenando la imagen existente en memoria, dentro de una variable de tipo *Bitmap*. La figura 3.17 muestra el proceso de

captura de la imagen. Una vez capturada la imagen en la variable de tipo *Bitmap*, se procede a realzar los objetos mejorando el contraste. Este procedimiento se realiza llamando al método que ejecuta esta función, a éste se le entrega la variable de tipo *Bitmap* correspondiente a la imagen capturada, la imagen resultado se almacena en una segunda variable de tipo *Bitmap* llamada "*contrast*". En la figura 3.18 se muestra el cuadro en vivo observado por la *webcam* y la imagen con el ajuste de contraste.

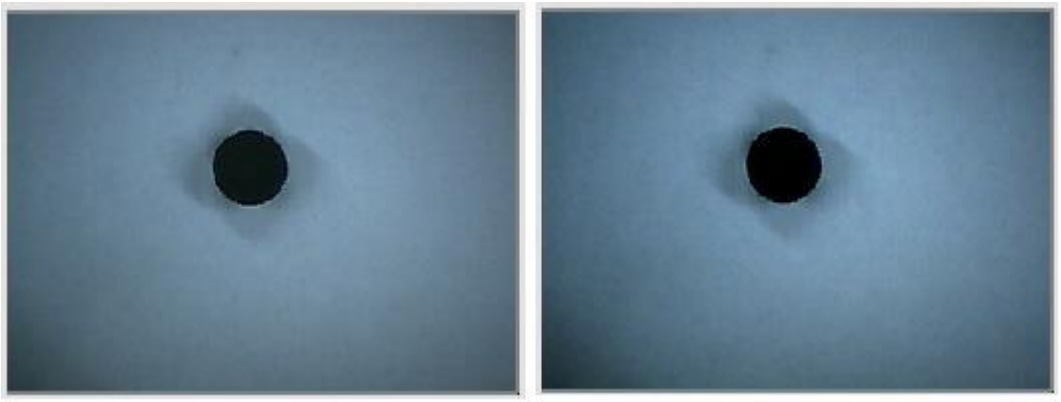

Figura 3.18. Ajuste de contraste. En la parte izquierda la señal en vivo de la Webcam, en la parte derecha la imagen con el contraste ajustado.

Luego de ajustar el contraste, se procede a la conversión del *Bitmap* a escala de grises para facilitar la diferenciación entre los objetos y el entorno. El entorno fue definido binario, es decir, el fondo (paredes) de color blanco y las marcas de color negro. Llevar la imagen a escala de grises sirve para no confundir las sombras con las marcas. La imagen en escala de grises, se almacena en la tercera variable de *Bitmap* llamada "*gris*". En la figura 3.19 se muestra la imagen en vivo de la webcam y la imagen convertida a escala de grises.

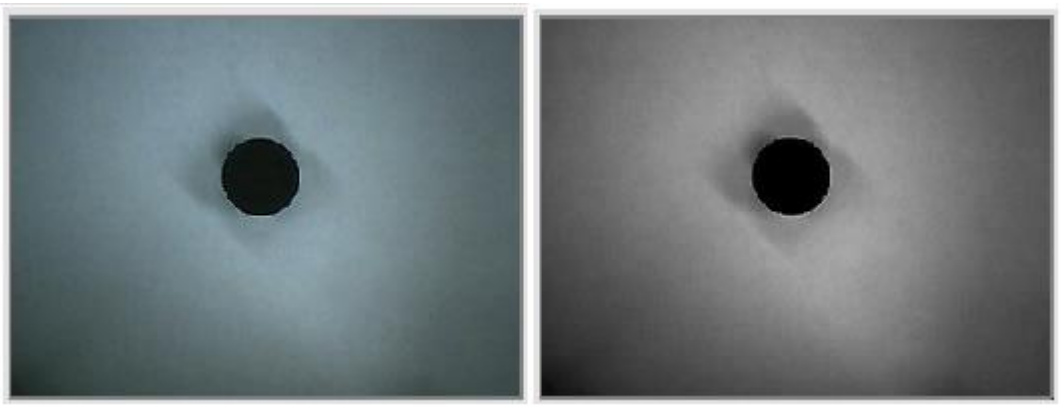

Figura 3.19. Escala de Grises. En la parte izquierda se muestra la señal en vivo de la webcam, y en la parte derecha la imagen convertida a escala de grises.

Una vez llevada la imagen a escala de grises, se procede a la binarización de la misma, para esto se establece un umbral, se recorre la imagen píxel por píxel y el píxel que este por encima de ese umbral fija su valor al máximo (255), y el píxel que este por debajo de ese umbral fija su valor al mínimo (0). De esta manera se diferencia totalmente el objeto del fondo y luego se procede a almacenar esta imagen en la cuarta variable de tipo *Bitmap* llamada "*umbral*". En la figura 3.20. Se observa la imagen en vivo y la imagen umbralizada.

Figura 3.20. Umbralización de la imagen. En la parte izquierda se indica la imagen en vivo y en la parte derecha la imagen umbralizada.

Luego de umbralizar la imagen, se aplica el filtro de "sal y pimienta" para eliminar los posibles ruidos en los bordes de cada objeto. Este método retorna la imagen filtrada que se almacena en la quinta variable de tipo *Bitmap* llamada "*filtrada*". Este procedimiento se puede observar en la figura 3.21.

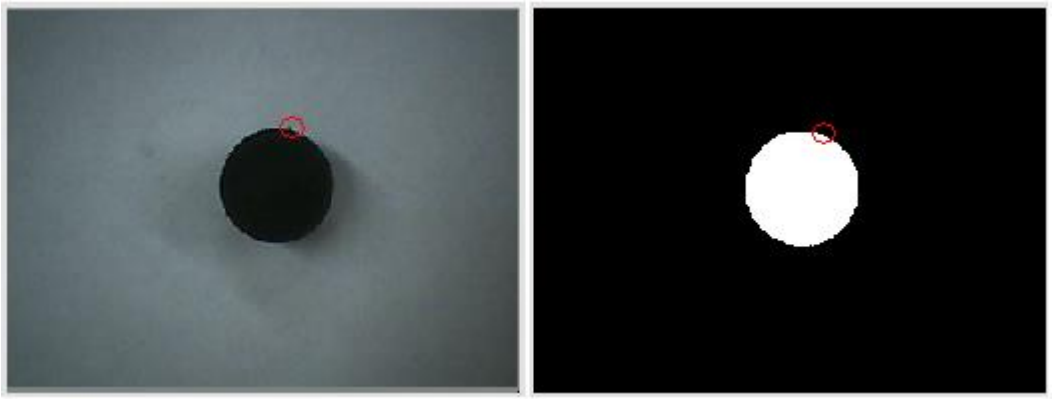

Figura 3.21. Imagen filtrada. En el circulo rojo de la imagen izquierda se observa unos cuantos píxeles que se entienden como ruido en la detección del objeto, y en la parte derecha se observa el objeto umbralizado y libre de ruido.

Por último se recorre la imagen y se realiza la segmentación de cada uno de los objetos presentes en este cuadro. Al método encargado de esta función se le pasa como argumento la imagen filtrada y este retorna una imagen donde se observan cada uno de los objetos diferenciados mediante un cuadro rojo que se muestra alrededor de cada objeto. En este punto se almacena cada una de las características de cada objeto, ancho, alto y área, posición en *x* y posición en *y*, con las cuales en pasos siguientes se inferirán las medidas de distancia y ángulo de cada marca. En la figura 3.22 se observa la segmentación de una imagen con tres objetos.

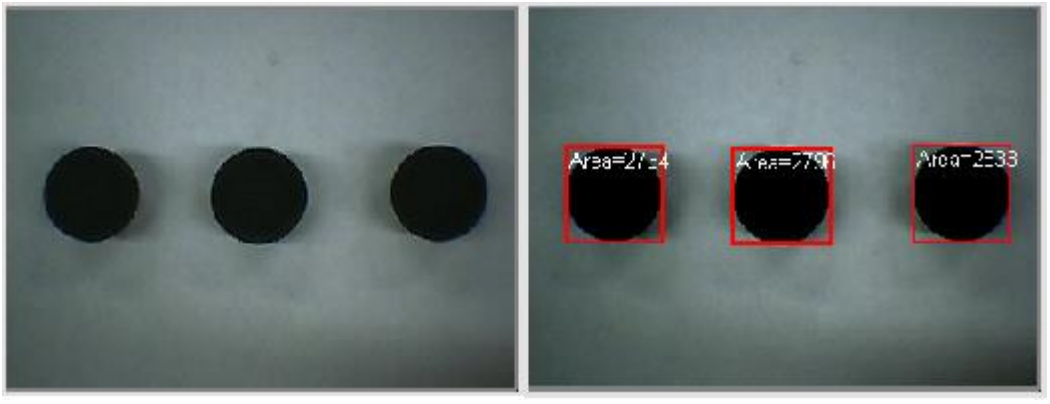

Figura 3.22. Segmentación. En la parte izquierda se observa la imagen en vivo y en la parte derecha se observa la misma imagen donde son diferenciados y caracterizados cada uno de los objetos presentes en el cuadro.

Al final de estos cinco pasos se obtiene la información más importante de cada objeto, con la cual en los pasos siguientes se realiza la extracción de características, entendiéndose ésta como la relación de cada una de las marcas detectadas hacia el

plano de referencia de la cámara y luego hacia un plano de referencia global donde se moverá la cámara montada en el robot.

Una vez obtenidas las características de cada objeto como se mostró en los pasos anteriores, se procede al manejo de esta información para realizar el cálculo de distancia y ángulo al cual se encuentra cada marca.

Primero, teniendo la medida de área de cada marca se realiza el cálculo de la distancia entre ejes paralelos de la cámara y la marca como se muestra en la figura. 3.23.

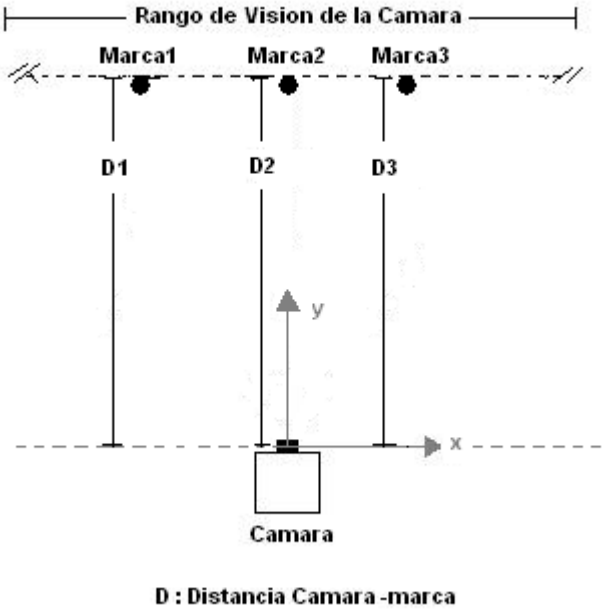

Figura 3.23. Medida de distancia Cámara-Marca. La medida de distancia entregada es con respecto a los ejes paralelos de la marca y la cámara.

Esta distancia se calcula a partir del área de la marca observada. Cabe aclarar que el área se obtiene sumando todos los píxeles que componen cada marca. El procedimiento para realizar el cálculo de esta distancia fue totalmente experimental. Se definió un espacio de 123 centímetros desde una pared donde había una marca fija; se alinea la cámara con la marca y se divide el espacio de 123 centímetros en secciones de 10 centímetros y en cada intervalo se toma la medida de área en píxeles de la marca y la distancia correspondiente; de esta manera se puede establecer la relación entre área y distancia. Con esta información se traza la curva correspondiente, se realiza una

linealización por tramos y se establece una ecuación para cada sección recta de la curva con la cual, a partir de una entrada en términos de área en píxeles se obtiene una medida de distancia en centímetros. La figura 3.24 ilustra el montaje realizado para la toma de datos necesarios para encontrar la relación distancia-área.

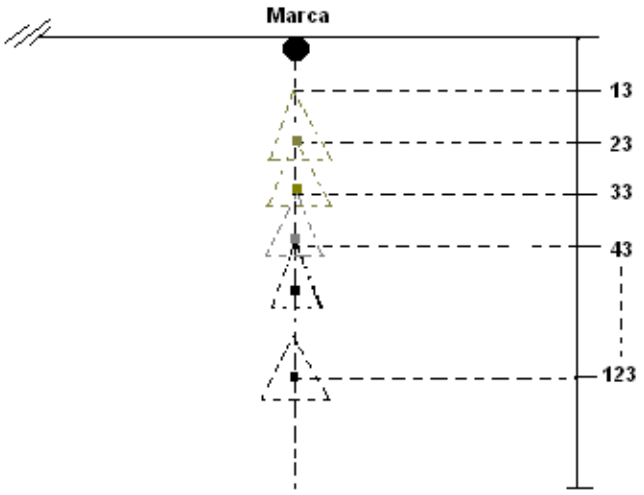

Figura 3.24. Toma de medidas de área y de distancias.

Las medidas obtenidas son las siguientes:

	A[X]	B[Y]
1	5869	13
2	2149	23
3	1017	33
4	550	43
5	325	53
6	245	63
7	176	73
8	113	83
9	96	93
10	78	103
11	39	113
12	22	123

Figura 3.25. Datos resultantes de la medición de distancia-área.

En la figura anterior, en el eje *X* se muestran los datos medidos correspondientes al área en píxeles de la marca observada y en la columna *Y,* se muestran los datos medidos correspondientes a la distancia a que fue tomada cada media de área. En la figura 3.26 se observa la curva obtenida a partir de los datos de distancia-área. En el eje *X* mediciones en píxeles y en el eje *Y* mediciones de distancia en centímetros.

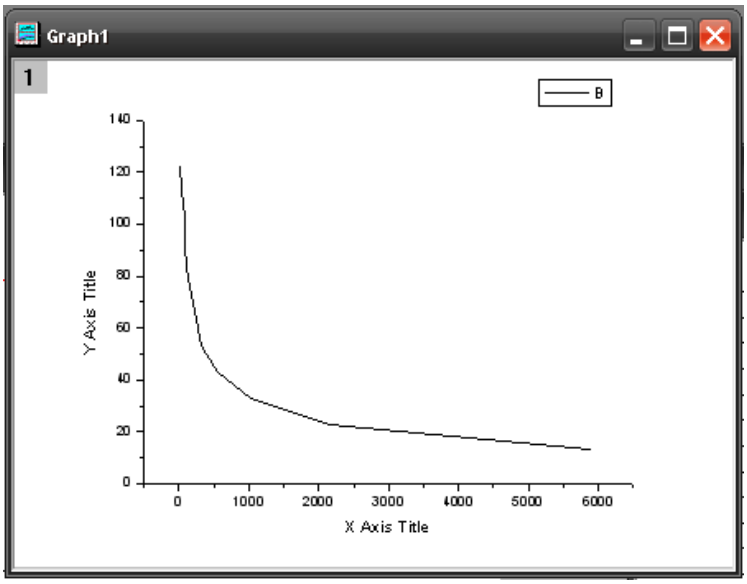

Figura 3.26. Curva área Vs distancia.

La curva mostrada en la figura anterior fue seccionada en seis partes, se aproxima cada parte a una recta y se calcula la ecuación para cada una de estas rectas utilizando el software *Microcal Origin 6.0.*

El primer tramo linealizado de esta curva junto con su respectiva ecuación, se muestra en la figura 3.27, este tramo comprende desde los 13 hasta los 23 centímetros.

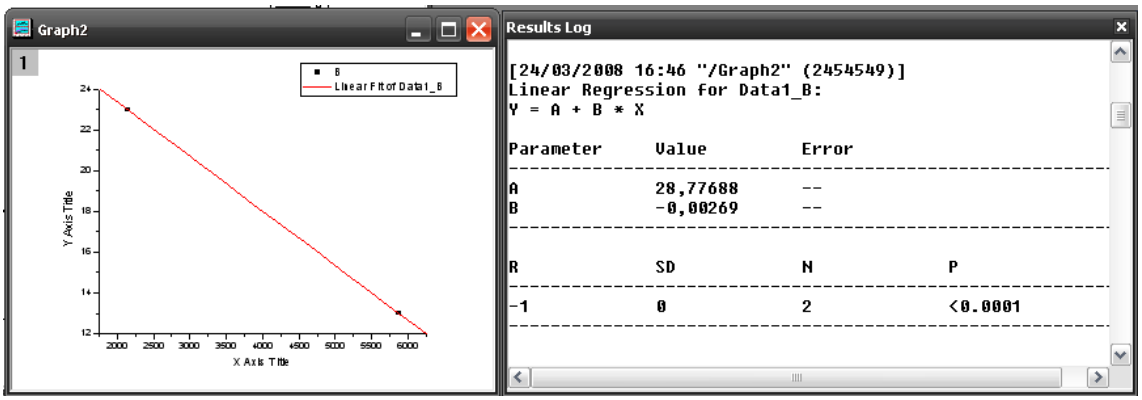

Figura 3.27. Ecuación de la recta1. A la derecha la primera sección de las 8 partes en que fue divida la curva de la figura 3.26, a la izquierda su ecuación lineal.

De esta misma manera fueron extraídas las 8 ecuaciones de las 8 secciones en que fue dividida la curva de la figura 3.26. Una vez obtenidas estas 8 ecuaciones se creo un método al cual se le entrega como argumento, el área de la marca detectada, dentro de él, se plantean condicionales para tener 8 posibles rangos de áreas, y si el área que entra

al método esta dentro de estos rangos se aplica la ecuación respectiva, de esta manera se tiene que ante una entrada en unidades de píxeles se obtiene su equivalente en distancia en centímetros. El rango para el cual se plantearon estas ecuaciones esta entre 22 píxeles y 5869 píxeles o su equivalente en distancia 123 centímetros y 13 centímetros.

Hasta este punto se tiene la distancia recta entre los ejes paralelos de la marca a la cámara. Pero se necesita la distancia diagonal *R* entre la marca y la cámara y el ángulo al cual esta siendo observada la marca con respecto al eje x de la cámara, la figura 3.28 ilustra los datos a obtener a partir de la imagen.

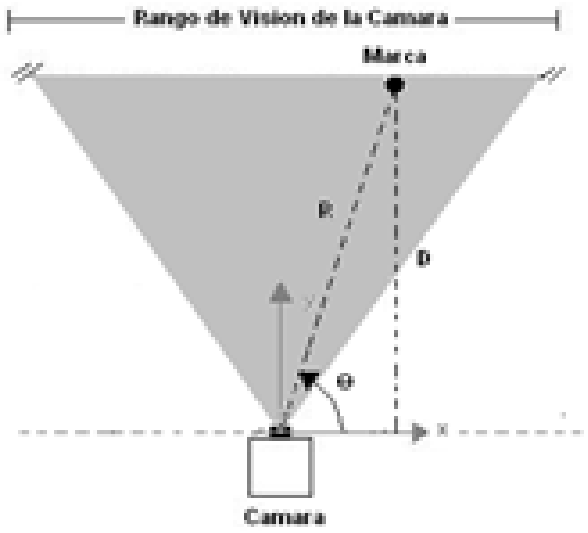

Figura 3.28. Datos de las marcas obtenidos de la imagen. R es la distancia diagonal entre la cámara y la marca, D es la distancia entre ejes paralelos de la cámara y la marca, y θ es el ángulo al cual esta siendo observada la marca con respecto al eje x de la cámara.

El valor de la distancia *R*, se calcula a partir de la distancia entre ejes paralelos *D* y del ángulo al cual se encuentra la marca con respecto al centro de la imagen (figura 3.29).

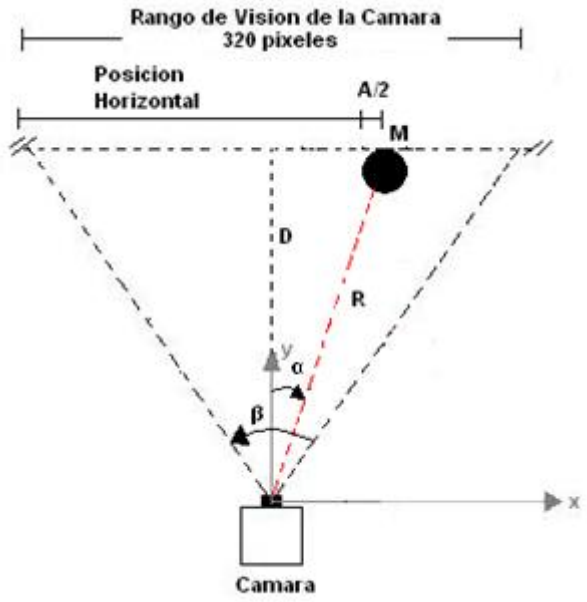

Figura 3.29. Esquema para calcular la distancia diagonal R. A/2 es el punto medio de la marca, α es el ángulo al que se encuentra la marca con respecto al centro de la imagen y β es el ángulo de visión de la cámara.

Para calcular el ángulo al cual se encuentra la marca dentro de la imagen es necesario determinar la posición de la marca dentro de la imagen; este procedimiento se realiza de la siguiente manera:

Primero, como se mencionó anteriormente, de la etapa de procesamiento de la imagen se puede extraer la ubicación en los ejes *x, y* de cada marca en el plano de la imagen. Esta ubicación esta dada en píxeles. También se puede extraer el alto y el ancho de cada marca en píxeles. Estos datos son suficientes para calcular la posición del centro de cada marca con respecto a la imagen de donde fueron extraídas. La posición horizontal del centro de cada marca esta dada como $Posx_i$ y se calcula,

$$Posx_i = PosH_i + \left(\frac{A_i}{2}\right) \qquad (4.1)$$

Donde $PosH_i$ es la posición de la *i-ésima* marca desde el punto cero hasta donde comienza la marca y A_i es el ancho en píxeles de la *i-ésima* marca. En la figura 3.30 se ilustra un ejemplo de una marca que comienza en el píxel 190, esta tiene un ancho de 20 píxeles y se calculará la posición de su punto central.

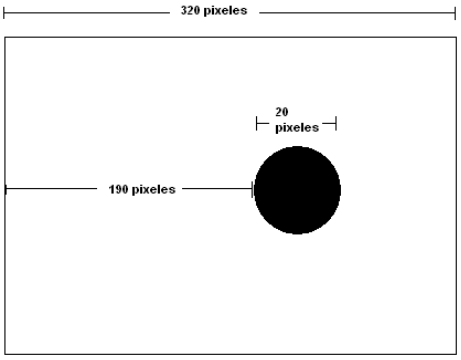

Figura 3.30. Ejemplo de cálculo de posición de una marca.

En la figura anterior el valor de $Posx_1$ será,

$$Posx_1 = 190 + \frac{20}{2} = 200$$

Este valor es la posición del punto medio de la marca dentro de la imagen que va de 0 a 320 píxeles, la posición del centro de la marca en la imagen es 200, con este valor de posición del centro de la marca, se establece una relación con respecto al ángulo con que esta siendo observada la marca desde la cámara.

En la figura 3.31 se observa que el ángulo de visión tiene relación directa con el ángulo al cual esta siendo observada la marca. Se establece una relación directa entre el ancho de la imagen, para este caso 320 píxeles y el ángulo de visión de la cámara β.

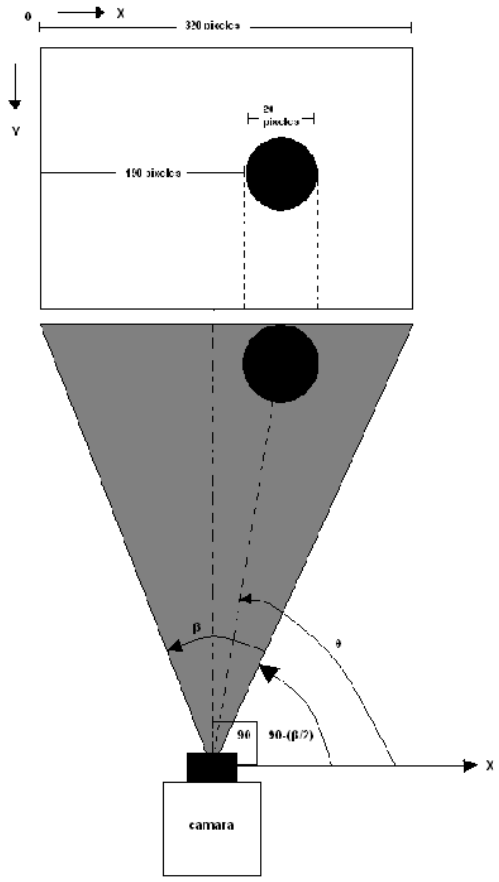

Figura 3.31. Relación entre θ y la posición de la marca en la imagen.

De la figura anterior se tiene que β es el ángulo de visión de la cámara, y que el centro de la imagen coincide con un ángulo de 90° con respecto al eje x de la cámara. Con estos datos resulta fácil calcular el ángulo θ al cual esta la marca con respecto al eje x de la cámara.

Para calcular θ primero se debe convertir la posición de la marca en la imagen a un ángulo dentro del ángulo de visión de la cámara θ',

$$320Pixeles \cong \beta$$

$$\theta'_i = \frac{\left(-\left(\left(Posx_i + \left(\frac{A_i}{2}\right)\right) - 320\right) \times \beta\right)}{320}$$

Una vez obtenido el ángulo al cual se encuentra la marca dentro de la imagen θ', se procede a calcular el ángulo al cual se encuentra la marca con respecto al eje *x* de la cámara.

$$\theta_i = \theta'_i + \left(90 - \left(\beta/_2\right)\right)$$

Mediante esta relación se puede obtener el ángulo al cual se encuentra una marca con respecto al eje *x* de la cámara dada la posición de la marca dentro de la imagen.

Una vez obtenido el valor de θ, el paso siguiente es calcular *R*, que también se relacionaba con la posición a la cual se encuentra la marca dentro de la imagen. El valor α de la figura 3.38 es el ángulo al cual se encuentra la marca con respecto al centro de la imagen, que sería el eje *y* de la cámara.

$$\alpha_i = \frac{\left(-\left(\left(PosH_i + \left(\frac{A_i}{2}\right)\right) - 160\right) \times \left(\frac{\beta}{2}\right)\right)}{160}$$

Con este ángulo se puede calcular la distancia diagonal R,

$$R_i = D_i/\cos(\alpha_i)$$

Con el valor de distancia diagonal *R* y del ángulo al cual se observa una marca, se completa la información que entregaría un *scanner láser*, pero usando un sistema de visión. En la figura 3.32 se ilustran todos los factores presentes en el proceso de extracción de dos marcas a partir de una imagen capturada por una cámara web, a una resolución de 320x240 píxeles.

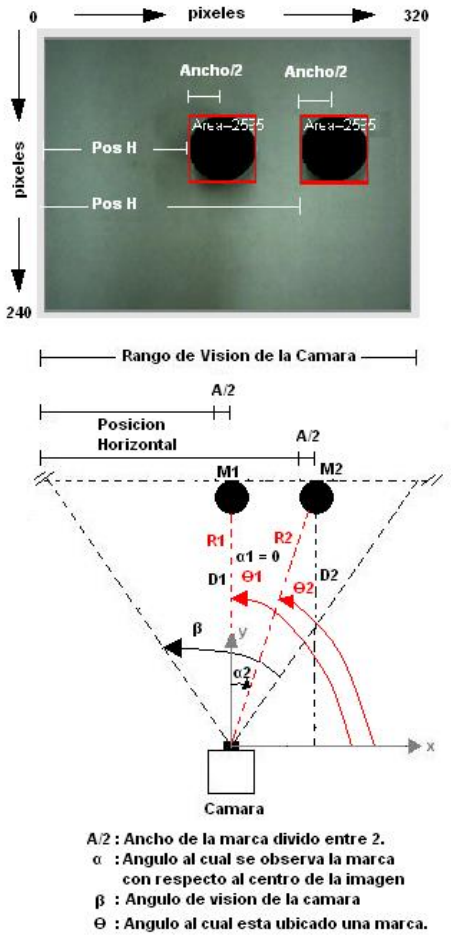

Figura 3.32. Términos presentes en la extracción de marcas del entorno a partir de una imagen.

Por último se convierten estos datos hacia un plano general donde se realiza el proceso de Localización y Mapeo Simultáneos.

Para representar el entorno se utiliza un objeto del directorio *System.Windows.Forms* llamado Panel, este objeto despliega un cuadro de tamaño variable dentro del cual se puede pintar cualquier figura. Dentro de este panel se dibuja el robot en su posición actual, y cada una de las marcas vistas por el robot durante su travesía por el entorno.

Cada una de las marcas se define como un punto dentro del panel y su posición *x, y* como la posición dentro del panel. En este panel se trabaja a una escala de 1 píxel por centímetro, el tamaño del panel se define de 400x400 píxeles lo que significa que el sistema puede pintar entornos de máximo 4x4 metros, a pesar de que si el robot sigue

avanzando por encima de este limite, la información se almacena en el vector de estados. La conversión con respecto al mapa global se realiza de la siguiente manera:

$$Xmarca[i] = (VecEst[0] + Xtriangulo) + R_i \times \sin(VecEst[2] + \alpha_i)$$
$$Ymarca[i] = (VecEst[1] + Ytriangulo) + R_i \times \cos(VecEst[2] + \alpha_i)$$

Donde *Xmarca[i]* es la posición en x de la i-ésima marca dentro de panel, *VecEst[0]* es la posición del centro del robot en el eje x, *Xtriángulo* es la posición en x de la punta del robot, esto significa que la cámara ira montada en la parte frontal del robot, R_i es la distancia diagonal marca-cámara, *VecEst[2]* es la orientación actual del robot θ y α_i es el ángulo al cual se encuentra la marca con respecto al centro de la marca. Los valores *Ymarca[i], VecEst[1], Ytriángulo,* tienen el mismo significado que los valores anteriores, pero para el eje y.

De esta manera a cada una de las marcas observadas por la cámara se le asocia un punto dentro del panel donde se realiza el mapeo del entorno.

Para pintar el robot también es necesario realizar cierta cantidad de cálculos, pues para pintar un triángulo dentro del panel es necesario definir tres puntos y luego unirlos mediante tres líneas. Estos puntos deben variar con la posición en *x e y* del robot y su orientación θ, las ecuaciones para pintar este triángulo en cualquier punto dentro del panel y con cualquier orientación θ son las siguientes:

$$Xtriangulo = 18{,}6 \times \cos(VecEst[2] + 36{,}249)$$
$$Ytriangulo = 18{,}6 \times \sin(VecEst[2] + 36{,}249)$$
$$Xtriangulo1 = 18{,}6 \times \cos(VecEst[2] + 143{,}751)$$
$$Ytriangulo1 = 18{,}6 \times \sin(VecEst[2] + 143{,}751)$$
$$Xtriangulo2 = 18{,}6 \times \cos(VecEst[2] + 270)$$
$$Ytriangulo2 = 18{,}6 \times \sin(VecEst[2] + 270)$$

Estos son los 3 puntos del triángulo que variaran con la orientación del robot θ, los valores constantes de estas ecuaciones están dados a partir de las dimensiones reales del robot que visto desde arriba son 30x30 centímetros. La figura 3.33 muestra la representación

del robot a partir de sus dimensiones reales, ahí se presenta la procedencia de los valores constantes de las anteriores ecuaciones.

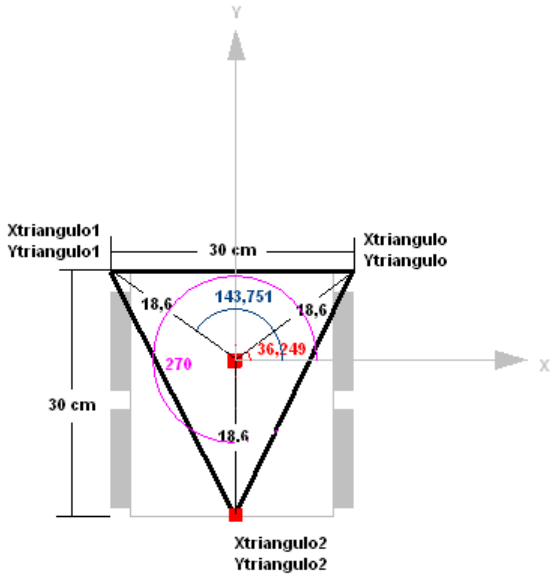

Figura 3.33. Representación del Robot en el panel de mapeo.

El siguiente paso es hacer que el triángulo que representa al robot varié con respecto a la posición en *x e y* del mismo, para ello se definen los tres puntos del triángulo cada uno con su posición *x, y,* la siguiente ecuación muestra esta relación:

$$Punto1 = \big((VecEst[0] + Xtriangulo1), (VecEst[1] + Ytriangulo1)\big)$$
$$Punto2 = \big((VecEst[0] + Xtriangulo), (VecEst[1] + Ytriangulo)\big)$$
$$Punto3 = \big((VecEst[0] + Xtriangulo2), (VecEst[1] + Ytriangulo2)\big)$$

Con las anteriores ecuaciones se obtienen los tres puntos necesarios para pintar el triángulo que representa al robot dentro del plano, este triangulo se podrá pintar en cualquier posición del plano y con cualquier orientación dependiente del vector de estados. La figura 3.34 ilustra al robot ubicado en la posición x, y de 200cm, 100cm y con una orientación de 75° con respecto al origen, el punto rojo en el triángulo representa la ubicación de la cámara en el robot y el punto rojo fuera de él representa una marca observada frente a él.

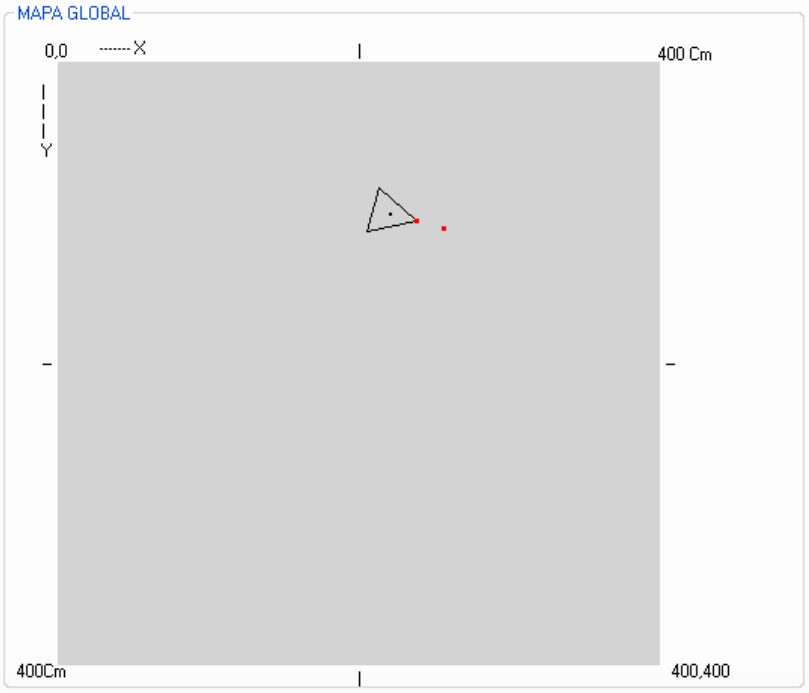

Figura 3.34. Representación del robot dentro del panel general en una ubicación x, y de 200cm, 100cm y con una orientación de 75°.

El paso siguiente consiste en diferenciar las marcas nuevas observadas en el entorno de las repetidas, pero eso corresponde a la asociación de datos que se muestra a continuación.

3.7 Asociación de Datos

El problema de la asociación de datos consiste en que el robot puede observar la misma marca más de una vez y seria erróneo agregarla al mapa sabiendo que es una sola marca vista varias veces. Un ejemplo de esto es que el robot en una posición observe una marca dada, y luego de su recorrido pase por el mismo punto por donde inició y vuelva a observar la misma marca que vio al inicio de su recorrido. En éste punto es necesario que el robot tenga la capacidad de diferenciar si una marca ya fue observada en pasos anteriores, y proceder a su posterior filtrado. Además, el segundo paso del EKF requiere que en un punto dado se pueda diferencia las marcas vistas por primera vez de las re-observadas en el instante inmediatamente anterior para ajustar la posición del robot.

En la práctica los problemas que abarca la asociación de datos son los siguientes:

- Es posible que no se puedan re-observar marcas en cada iteración.
- Es posible observar una marca al comienzo, pero no siempre es posible mirarla nuevamente.
- Es posible asociar erróneamente una marca a una observada anteriormente.

Para tener en cuenta estos posibles problemas se procede a realizar el desarrollo de asociación de datos, para ello se divide el problema en dos partes; la primera parte consiste en generar un vector con todas las posiciones de las marcas nuevas del entorno, es decir, este vector contiene solo posiciones de marcas diferentes, es decir no existe posiciones repetidas dentro de él. La segunda parte consiste en generar un vector que almacena temporalmente las marcas observadas en el instante inmediatamente anterior, para diferenciar éstas de las marcas nuevas.

Para realizar el llenado del primer vector, se procede de la siguiente manera, primero se llena un vector auxiliar con todas las marcas que va observando la cámara, este proceso se ilustra en la figura 3.35.

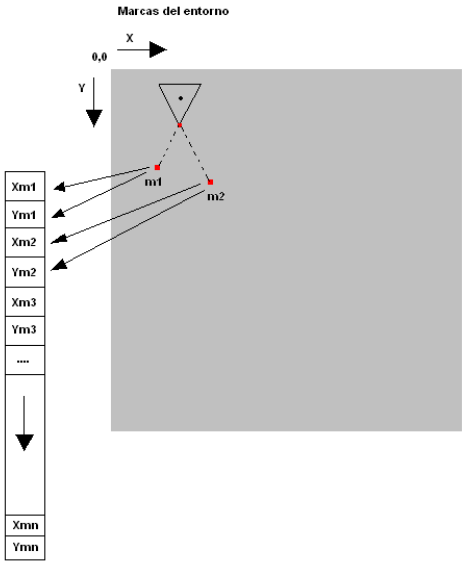

Figura 3.35. Almacenamiento de marcas en un vector auxiliar.

De esta manera se tiene un vector que reúne todas las posiciones de las marcas aun si estas fueran repetidas, el segundo paso consiste en filtrar estas marcas repetidas

recorriendo el vector comparando campo por campo y estableciendo un umbral para determinar si la posición x, y de una marca corresponde a una anteriormente observada.

Se establece un umbral de 3 centímetros, si la diferencia entre las dos posiciones comparadas es menor de 3 centímetros, se coloca en cero la última medida comparada ya que es la misma marca. La figura 3.36 muestra un ejemplo de esta comparación.

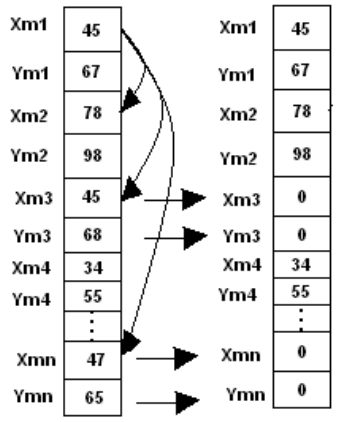

Figura 3.36. Filtrado de marcas repetidas del Vector de Estados Auxiliar.

Una vez puestas en cero las posiciones de las marcas repetidas el paso siguiente consiste en eliminar los ceros y dejar el vector libre de marcas repetidas. La figura 3.37 ilustra el método para suprimir los ceros del vector de estados auxiliar, esto se hace subiendo las marcas que están por debajo de los ceros para que ocupen su posición dentro del vector.

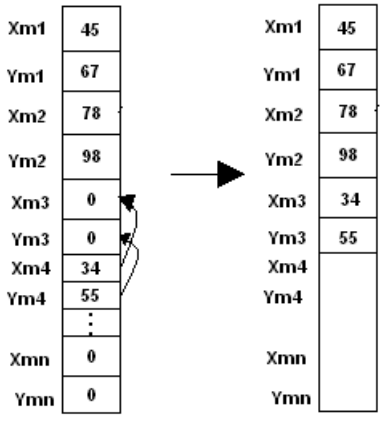

Figura 3.37. Eliminación de ceros en el vector de estados auxiliar.

De esta manera se obtiene un vector de estados auxiliar que posee dentro de sí solo marcas diferentes, es decir, nunca se encuentran dos posiciones x, y repetidas dentro de este vector.

La segunda parte de la asociación de datos consiste en diferenciar las marcas nuevas de las re-observadas en el instante inmediatamente anterior. En este punto, se establece el mismo umbral de 3 centímetros, pero para catalogar una marca como nueva se procede de la siguiente manera.

Primero se crea un vector que almacene las marcas inmediatamente anteriores, después de cada ciclo, éste vector es vaciado y después llenado con las marcas anteriores para el instante siguiente. Luego, a medida que se van observando nuevas marcas, estas se van comparando con el vector que almacena las marcas inmediatamente anteriores, si el resultado de esta comparación esta por encima del umbral, esta marca se cataloga como nueva, este proceso se realiza de la siguiente manera.

$$if\big((VecEstAX[i] - Xmarca[i]) > 3 || (VecEstAY[i] - Ymarca[i]) > 3\big)$$

Donde *VecEstAX* es el vector que contiene las posiciones *x* de las marcas en el instante inmediatamente anterior, *Xmarca* es el vector donde se almacenan las posiciones *x* de las marcas actualmente vistas, *VecEstAY* y *Ymarca* tienen el mismo significado de lo anterior pero aplicado a la posición *Y* de las marcas. Si se cumple que la diferencia entre las posiciones en X o en las posiciones en Y (cualquiera de las dos) es mayor que 3 centímetros, la posición X de la marca se almacena en el vector *VecMarcasNuevasX,* y la posición Y de la marca se almacena en el vector *VecMarcasNuevasY*. Solo basta con que una de las coordenadas de la marca sea distinta para que se asuma que es una marca nueva.

De igual forma se utilizan los vectores anteriormente expuestos para clasificar las marcas repetidas, para esto se utiliza el mismo umbral de 3 centímetros, pero esta vez se catalogan como marcas repetidas si el producto de la comparación entre las posiciones de las marcas anteriores y las posiciones de las marcas actuales es menor al umbral. La siguiente ecuación muestra la comparación que se realiza para llevar a cabo éste procedimiento.

$$if\big((VecEstAX[i] - Xmarca[i]) < 3 \;\&\&\; (VecEstAY[i] - Ymarca[i]) < 3\big)$$

El significado de *VecEstAX, Xmarca, VecEstAY, Ymarca,* es el mismo de los mencionados anteriormente. Si se cumple que la diferencia entre posiciones en X y en las posiciones en Y (solo las dos) es menor que 3 centímetros, la posición X de la marca se almacena en el vector *VecMarcasRepetidasX,* y la posición Y de la marca se almacena en el vector *VecMarcasRepetidasY.* Solo si las dos coordenadas de las marcas comparadas son similares, esta se asume como una marca repetida.

De esta manera se realiza la asociación de datos, luego de este proceso, ya se tiene claramente diferenciadas las marcas nuevas de las marcas anteriormente vistas, además se obtiene un vector que contiene solo marcas diferentes del entorno.

3.8 EKF

La implementación del sistema se realiza utilizando el Filtro Extendido de Kalman con base en la argumentación expuesta en el capitulo 2 de este trabajo.

A continuación se realiza una descripción de las matrices que se emplean para realizar SLAM a partir del Filtro Extendido de Kalman.

VECTOR DE ESTADOS DEL SISTEMA X:

El vector de estado del sistema X es probablemente uno de los más importantes vectores junto a la matriz de covarianza P. Este contiene la posición del robot x_r, y_r y θ_r. Además contiene la posición x y y de cada marca (figura.3.38).

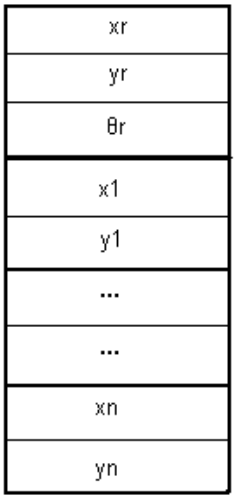

Figura 3.38. Vector de estado del sistema X [24].

El tamaño del vector X es $3 + 2 * n$ filas, donde n es el número de marcas. Normalmente los valores de x y y se guardan en centímetros; y el ángulo de rotación θ_r se guarda en grados.

MATRIZ DE COVARIANZA P:

La matriz de covarianza[10] P es la matriz central del sistema (figura 3.39) contiene la covarianza de la posición del robot, la covarianza de las marcas, la covarianza entre la posición del robot y las marcas y la covarianza entre las marcas.

[10] La covarianza de dos variables proporciona una medida de que tan fuertemente correlacionadas estas dos variables están. La correlación es un concepto utilizado para medir el grado de dependencia lineal entre variables.

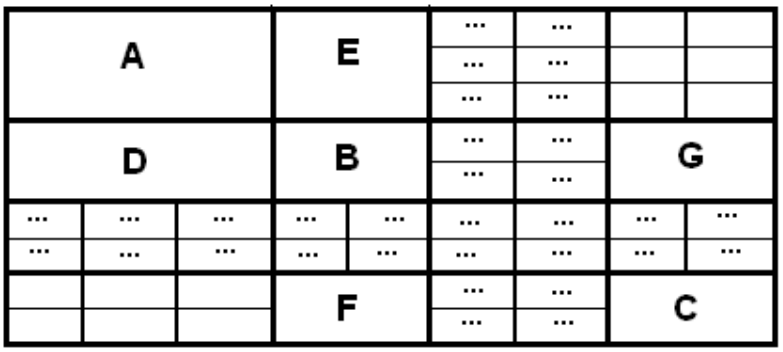

Figura 3.39. Matriz de covarianzas P [24].

La primera celda **A** contiene la covarianza de la posición del robot, ésta es una matriz de 3x3 (x_r, y_r, θ_r); **B** es la covarianza de la primera marca; ésta es una matriz de 2x2 ya que la marca no tiene una orientación theta (θ), la covarianza de las demás marcas se ubican en la diagonal hasta **C,** la cual es la covarianza de la última marca.

La celda **D** contiene la covarianza entre el estado del robot y la primera marca. La celda **E** contiene la covarianza entre la primera marca y el estado del robot. **E** puede ser deducida a partir de la transpuesta de **D**.

F contiene la covarianza entre la última marca y la primera marca, mientras **G** contiene la covarianza entre la primera marca y la última marca, **G** puede ser deducida a partir de la transpuesta de **F**.

Así, aunque la matriz de covarianza puede parecer complicada de realizar, es en realidad desarrollada sistemáticamente.

Inicialmente el robot no ha visto ninguna marca por lo cual, la matriz de covarianza P solo incluye la matriz **A.** La matriz de covarianza debe ser inicializada usando algunos valores por defecto para la diagonal. Esto refleja la incertidumbre en la posición inicial, dependiendo de la implementación será frecuente un error singular, si la incertidumbre inicial no es incluida en algunos de los cálculos, es conveniente incluir algún error inicial aunque existan razones para creer que la posición inicial del robot es la correcta.

GANANCIA DE KALMAN K

La ganancia de Kalman K se calcula para determinar cuanto se quiere ganar a partir del conocimiento de las marcas re-observadas con el fin de mejorar el estado actual del robot. Por ejemplo, si se observa que el robot se ha movido 10 centímetros a la derecha, de acuerdo a las marcas re-observadas, se utiliza la ganancia de Kalman para determinar la posición en la que realmente se encuentra; ésta puede ser únicamente de 5 centímetros porque no se confía en las posiciones de las marcas completamente, por lo tanto, se debe encontrar un compromiso entre la odometría y la corrección de las marcas.

Esto se hace usando la incertidumbre de las marcas observadas junto con una medida de la calidad de la medida del dispositivo de medición de marcas y el desempeño de la odometría del robot.

Si la medida del dispositivo de medición de marcas es realmente mala comparada con la odometría desarrollada por el robot, la ganancia de Kalman puede ser baja. Por el contrario, si la medida del dispositivo de medición de marcas es muy buena comparada al desarrollo de la odometría del robot la ganancia de Kalman podría ser alta. La matriz de la ganancia de Kalman se observa en la figura 3.40.

Xr	Xb
Yr	Yb
tethar	tethab
X1,r	X1,b
Y1,r	Y1,b
…	…
…	…
Xn,r	Xn,b
Yn,r	Yn,b

Figura 3.40. Matriz de la ganancia de Kalman [24].

La primera fila muestra cuanto podría ganarse desde la innovación para la primera fila del vector de estado X. La primera columna en la primera fila describe cuanto podría ganarse desde la innovación en términos del rango, la segunda columna en la primera fila describe cuanto podría ganarse desde la innovación en términos del ángulo.

Tanto la ganancia a partir del rango como la ganancia a partir del ángulo son para la primera fila en el vector de estados, lo cual es el valor de X de la posición del robot. La matriz continua bajando a partir de la posición del robot; las primeras tres filas; y las marcas cada dos nuevas filas.

El tamaño de la matriz es 2 columnas y 3+2*n filas, donde n es el número de marcas.

JACOBIANA DEL MODELO DE MEDIDA H:

La jacobiana del modelo de medida esta estrechamente relacionada con el modelo de medida. El modelo de medida (h) define la forma de calcular la magnitud y ángulo de las mediciones (posición de las marcas observadas) esperadas. Esto se hace utilizando la siguiente fórmula, la cual se denota como h.

$$h = \begin{bmatrix} magnitud \\ ángulo \end{bmatrix} = \begin{bmatrix} \sqrt{(\lambda_x - x)^2 + (\lambda_y - y)^2} + v_r \\ tan^{-1}\left(\frac{\lambda_y - y}{\lambda_x - x}\right) - \theta + v_\theta \end{bmatrix}$$

Donde λ_x, λ_y definen la posición de la marca; x_r, y_r corresponden a la posición actual estimada del robot, y θ es la rotación del robot. Esto podría dar la medida predicha de la magnitud y del ángulo de la marca con respecto a la posición estimada del robot. La jacobiana de esta matriz con respecto a x, y y θ es:

$$H = \begin{bmatrix} \frac{x - \lambda_x}{r} & \frac{y - \lambda_y}{r} & 0 \\ \frac{\lambda_y - y}{r^2} & \frac{\lambda_x - x}{r^2} & -1 \end{bmatrix}$$

H indica como la magnitud y el ángulo cambian con los cambios que tienen x_r, y_r y θ. El primer elemento en la primera fila es el cambio en la magnitud con respecto al cambio en la componente x. El segundo elemento es con respecto a cambios en la componente y. El último elemento es con respecto a cambios en θ la rotación del robot. Por supuesto, este

valor es cero ya que la magnitud no cambia con la rotación del robot. La segunda fila proporciona la misma información, excepto que muestra el cambio en ángulo para la marca. Este es el contenido normal de H para la estimación del estado del EKF normal.

Cuando se realiza SLAM a partir del EKF, se hacen necesarios algunos valores adicionales para las marcas como lo indica la figura 3.41.

Xr	Yr	Tr	X1	Y1	X2	Y2	X3	Y3
A	B	C	0	0	- A	- B	0	0
D	E	F	0	0	- D	- E	0	0

Figura 3.41. Valores adicionales para las marcas [24]

Cuando se utiliza la matriz H por ejemplo, para la marca número 2 se podría usar la matriz de arriba. La fila superior es únicamente para propósitos de información, no hace parte de la matriz.

Las tres primeras columnas conforman una matriz H común, para una estimación EKF. Para cada marca se adicionan dos columnas. Cuando se utiliza la matriz H para la marca dos se llena la matriz como se indica anteriormente, X2 con –A y –D y Y2 con –B y –E. Las columnas para el resto de marcas son cero. Solamente se usan dos términos, X2 y Y2, porque las marcas no tienen ninguna rotación.

JACOBIANA DEL MODELO DE PREDICCIÓN A:

La jacobiana del modelo de predicción está relacionada con su modelo de predicción, por ésta razón, inicialmente se revisa este modelo.

El modelo de predicción define la manera de calcular la posición del robot esperada, dada la posición anterior y los desplazamientos del robot. Esto se realiza utilizando la siguiente formula, denotada como f:

$$f = \begin{bmatrix} x_r + \Delta t cos\theta + q\Delta t cos\theta \\ y_r + \Delta t sin\theta + q\Delta t sin\theta \\ \theta + \Delta\theta + q\Delta\theta \end{bmatrix}$$

Donde x_r, y_r es la posición del robot, θ es la rotación del robot, Δt es la magnitud del desplazamiento instantáneo y q es el término del error. Se utilizan los cambios en la posición directamente de la entrada de la odometría del robot Δx, Δy y $\Delta\theta$ y el proceso del ruido q el cual se describe más adelante:

$$\begin{bmatrix} x_r + \Delta x + \Delta x * \mathrm{q} \\ y_r + \Delta y + \Delta y * q \\ \theta + \Delta\theta + \Delta\theta * q \end{bmatrix}$$

En todo caso se utiliza la versión linealizada a la hora de calcular el rendimiento de la jacobiana A.

$$A = \begin{bmatrix} 1 & 0 & -\Delta t sin\theta \\ 0 & 1 & \Delta t cos\theta \\ 0 & 0 & 1 \end{bmatrix}$$

Los cálculos son los mismos que para la matriz H , excepto que ahora se tiene una fila más para la rotación del robot. Dado que sólo se utiliza para la predicción de la posición del robot por lo cual no se extenderá para el resto de las marcas. Como puede verse en la matriz del modelo de predicción, el término $-\Delta t sin\theta$ es el mismo $-\Delta y$ en éste caso y $\Delta t cos\theta$ es el mismo Δx por ser ésta la versión linealizada del modelo de predicción (figura. 3.42). De esta manera, la jacobiana A queda en función de las variaciones de los desplazamientos.

1	0	-Δy
0	1	Δx
0	0	1

Figura 3.42. Matriz jacobiana del modelo de predicción [24].

JACOBIANAS ESPECÍFICAS DEL SLAM Jxr y Jz:

Cuando se está haciendo SLAM se hace necesaria la utilización estas jacobianas. Para la integración de nuevas marcas. La primera de estas jacobianas específicas es J_{xr}, ésta es básicamente la misma que la jacobiana del modelo de predicción, excepto que no incluye el término de rotación theta. Esta matriz jacobiana hace referencia a la posición de las marcas con respecto a la posición del robot.

$$J_{xr} = \begin{bmatrix} 1 & 0 & -\Delta t sin\theta \\ 0 & 1 & \Delta t cos\theta \end{bmatrix}$$

La jacobiana J_z es también la jacobiana del modelo de predicción para las marcas, pero esta vez con respecto a su observación en magnitud y ángulo.

$$J_z = \begin{bmatrix} cos(\theta + \Delta\theta) & -\Delta t * sin(\theta + \Delta\theta) \\ sin(\theta + \Delta\theta) & \Delta t * cos(\theta + \Delta\theta) \end{bmatrix}$$

RUIDO DEL PROCESO Q y W:

Se supone que el proceso tiene ruido gaussiano proporcional a las señales de control Δx, Δy y Δt. El ruido se denota con Q, y es una matriz de 3x3 (figura 3.43). Normalmente se calcula multiplicando algunas muestras gaussianas C con W y W transpuesta:

$c\Delta x^2$		
	$c\Delta y^2$	
		$c\Delta\theta^2$

Figura 3.43. Matriz de ruido Q [24].

$$W = [\Delta t cos\theta \quad \Delta t sin\theta \quad \Delta\theta]$$

$$Q = WCW^T$$

C es una representación de que tan exacta es la odometría. Su valor debe establecerse de acuerdo con el desempeño de la odometría del robot y es normalmente fácil de configurar de manera experimental.

RUIDO EN LA MEDIDA R y V:

Se asume también que el dispositivo de medida de las marcas tiene ruido gaussiano proporcional a la magnitud y al ángulo. Este se calcula como VRV^T. Siendo V una matriz identidad de 2x2. R es una matriz de 2x2 con números solamente en la diagonal. A continuación se muestra el rango, r, multiplicado por algunas constantes c y d (figura 3.44) las constantes deben representar la exactitud en la medición del dispositivo.

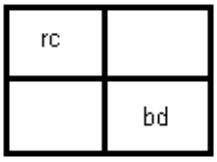

Figura 3.44. Ruido asociado al dispositivo de medida [24].

Con las matrices anteriormente explicadas se lleva a cabo el proceso de SLAM usando el Filtro Extendido de Kalman.

Los tres pasos siguientes conforman el desarrollo del algoritmo del EKF para realizar SLAM; y se implementan para el desarrollo de este trabajo.

3.8.1. Pasos del EKF

Una vez realizada la extracción de características (marcas) y la asociación de datos, el proceso SLAM se desarrolla en tres pasos:

1. Actualización del estado actual estimado usando los datos de odometría.
2. Actualización del estado estimado a partir de las marcas re-observadas.
3. Adición de nuevas marcas al estado actual.

En el primer paso se adicionan los desplazamientos del robot en el estado actual (x_r, y_r) al estado anterior del robot. Por ejemplo, el robot esta en un punto (x_r, y_r) con rotación θ (theta) y los desplazamientos son (dx, dy) y el cambio de rotación es $d\theta$. El resultado del primer paso del EKF es el nuevo estado del robot $(x_r + dx,\ y_r + dy)$ con rotación $(\theta + d\theta)$.

En el segundo paso se consideran las marcas re-observadas. Usando la posición actual estimada y las marcas re-observadas es posible corregir la estimación del paso 1 del EKF, debido a que los datos de odometría no son del todo confiables. Normalmente, existen algunas diferencias entre lo estimado y lo real, esto es llamado la innovación. Esta definición se aplica tanto a la diferencia entre la posición estimada del robot y la posición actual del robot, como a la diferencia entre lo que el robot es capaz de observar y lo que realmente hay en el entorno. En el segundo paso, la incertidumbre de cada marca re-observada es actualizada para reflejar cambios recientes.

En el tercer paso se adicionan nuevas marcas al estado actual del robot. Esto se hace usando información acerca de la posición actual del robot y adicionando información derivada de la relación entre las nuevas marcas y las antiguas.

PRIMER PASO DEL EKF: ACTUALIZACIÓN DEL ESTADO ACTUAL USANDO LOS DATOS DE ODOMETRÍA.

En este paso también llamado predicción del estado, se actualiza el estado actual del robot usando los datos de odometría. Se usan los movimientos del robot para calcular una estimación de la nueva posición del robot.

Para actualizar el estado actual se utiliza la siguiente ecuación:

$$\begin{bmatrix} x_r + \Delta t cos\theta + q\Delta t cos\theta \\ y_r + \Delta t sin\theta + q\Delta t sin\theta \\ \theta + \Delta\theta + q * \Delta\theta \end{bmatrix}$$

En el modelo de odometría simplemente se puede añadir los movimientos realizados por el robot.

$$\begin{bmatrix} x_r + \Delta x \\ y_r + \Delta y \\ \theta + \Delta\theta \end{bmatrix}$$

Utilizando este modelo, se actualiza la posición y orientación del robot, correspondiente a los tres primeros espacios en el vector de estados X. La figura 3.45 muestra, en la parte superior la actualización de la posición inicial del robot; éste parte de una posición inicial x_r, y_r, θ_r de 20, 0, 0, respectivamente. En la parte inferior se muestra el avance del robot de 100, 200, 45, en x_r, y_r, θ_r, respectivamente, siendo el modelo de movimiento utilizado en la aplicación desarrollada en este trabajo.

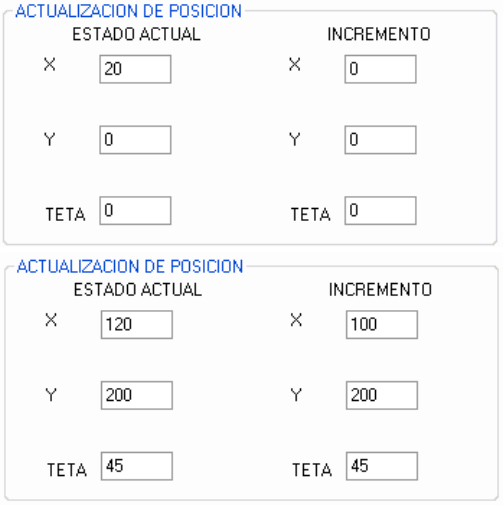

Figura 3.45. Modelo de movimiento del robot aplicado.

También se hace necesario actualizar la jacobiana del modelo de predicción *A* (figura 3.46), en cada iteración:

1	0	$-\Delta y$
0	1	Δx
0	0	1

Figura 3.46. Jacobiana del modelo de predicción A.

Utilizando el modelo anterior, se realiza la actualización de la matriz correspondiente a la jacobiana del modelo de predicción A, la figura 3.47 muestra como se desarrolla este paso.

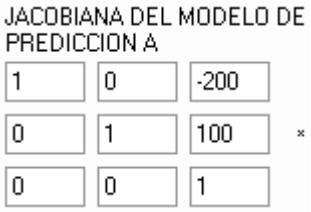

Figura 3.47. Jacobiana del modelo de predicción A aplicada.

La matriz A anterior es la resultante después del movimiento del robot mostrado en la parte inferior de la figura 3.45, es decir, después de un movimiento del robot de 100, 200, 45 en x_r, y_r, θ_r respectivamente.

También se actualiza Q(ruido del proceso) (figura 3.48) (c= 0.3 y Δt la variación del ángulo de rotación) para reflejar los términos del movimiento del robot, $\Delta x, \Delta y$ y Δt :

$c\Delta x^2$	$c\Delta x\Delta y$	$c\Delta x\Delta\theta$
$cc\Delta y\Delta x$	$c\Delta y^2$	$c\Delta y\Delta\theta$
$c\Delta\theta\Delta x$	$c\Delta\theta\Delta y$	$c\Delta\theta^2$

Figura 3.48. Ruido del proceso Q [24].

La figura 3.49 muestra la actualización de la matriz de ruido del proceso en la aplicación desarrollada en este trabajo.

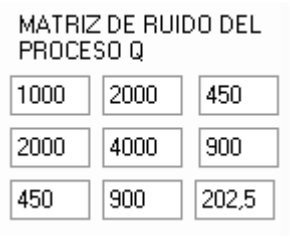

Figura 3.49. Matriz de Ruido del Proceso Q aplicada.

Finalmente se calcula la nueva covarianza para la posición del robot. Esta es una matriz de 3x3 ubicada en la parte superior izquierda de la matriz P y se actualiza de la siguiente manera:

$$P^{rr} = AP^{rr}A^T + Q$$

Donde P^{rr} es la matriz 3x3 del extremo superior izquierdo de P. En la figura 3.50. Se muestra el cálculo de esta etapa en la aplicación desarrollada.

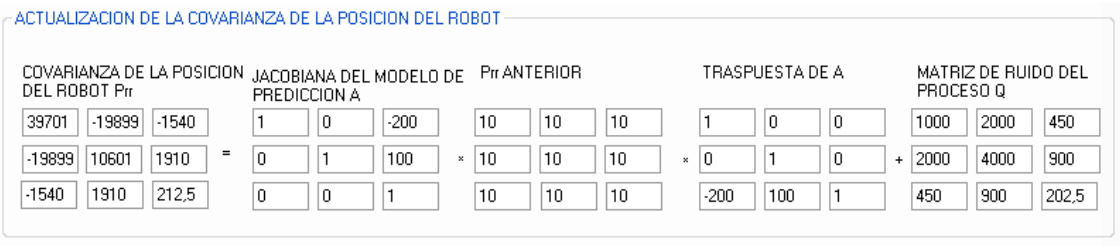

Figura 3.50. Actualización de la covarianza de la posición del robot.

Se ha actualizado la posición estimada del robot y la covarianza para la posición estimada. También se hace necesario actualizar la correlación cruzada del robot hacia las marcas. Esta correlación se ubica en las tres filas superiores de la matriz de covarianza P.

$$P^{ri} = AP^{ri}$$

En la figura 3.51 se muestra la actualización de este valor realizada en la aplicación desarrollada.

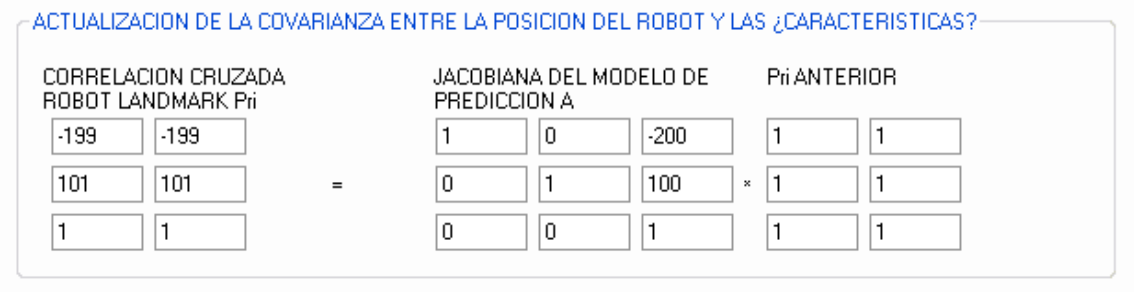

Figura 3.51. Actualización de la covarianza entre la posición del robot y las marcas.

Una vez terminado el primer paso del EKF se procede al desarrollo del segundo paso del EKF.

SEGUNDO PASO DEL EKF: ACTUALIZACIÓN DEL ESTADO ESTIMADO A PARTIR DE LAS MARCAS RE-OBSERVADAS.

La estimación obtenida de la posición del robot no es completamente exacta debido a los errores en la odometría del robot. El objetivo de este paso es compensar estos errores. Esto se consigue utilizando las marcas re-observadas en el estado inmediatamente anterior.

El tema de las marcas ya fue discutido, incluyendo como observarlas y como asociarlas con marcas ya existentes. Usando las marcas asociadas se puede calcular el desplazamiento del robot comparado con la posición en la cual el robot supone que se encuentra. Usando el desplazamiento se puede actualizar la posición del robot. Esto es lo que se hace en el paso 2. Este paso se debe ejecutar para cada marca re-observada.

Las marcas nuevas se abordan en el paso 3 del algoritmo.

Retardando la incorporación de nuevas marcas hasta el paso siguiente, decrece el costo computacional utilizado en este paso, debido a que la matriz de covarianza, P y el estado del sistema X, no requieren de mucho esfuerzo computacional para su actualización.

Primero, se va a tratar de predecir donde esta la marca, utilizando la posición actual estimada del robot (x, y) y la posición guardada de la marca (λ_x, λ_y). Con la siguiente formula:

$$\begin{bmatrix} magnitud \\ ángulo \end{bmatrix} = \begin{bmatrix} \sqrt{(\lambda_x - x)^2 + (\lambda_y - y)^2} + v_r \\ tan^{-1}\left(\frac{\lambda_y - y}{\lambda_x - x}\right) - \theta + v_\theta \end{bmatrix}$$

En la aplicación desarrollada, la forma de calcular estos valores se lleva a cabo como se presenta en la figura 3.52.

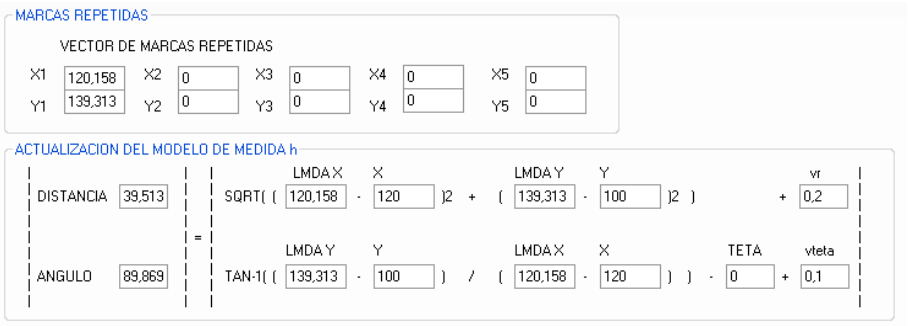

Figura 3.52. Ejemplo de la actualización del modelo de medida para una marca re-observada.

Se obtiene la magnitud y el ángulo para la marca y se representa como h, con este modelo se calcula la jacobiana H. Esta información puede ser comparada con la magnitud y el ángulo para la marca que se obtuvo en la asociación de datos, que se denota como z. Pero primero se hacen necesarios algunos cálculos más. Del capítulo anterior se tiene la jacobiana H:

Xr	Yr	Tr	X1	Y1	X2	Y2	X3	Y3
A	B	C	0	0	- A	- B	0	0
D	E	F	0	0	- D	- E	0	0

Como se indicó anteriormente los valores son calculados como sigue:

$$H = \begin{bmatrix} \frac{x-\lambda_x}{r} & \frac{y-\lambda_y}{r} & 0 \\ \frac{\lambda_y - y}{r^2} & \frac{\lambda_x - x}{r^2} & -1 \end{bmatrix}$$

Cabe recordar que solo se deben llenar las primeras tres columnas y las columnas validas para la marca. En la figura 3.53 se indica el proceso de actualización de la jacobiana del modelo de medida H, realizado por la aplicación desarrollada.

Figura 3.53. Ejemplo del cálculo de H para una marca repetida.

El error de la matriz R (figura 3.54) también debe ser actualizado para reflejar la magnitud y el ángulo en la medición actual. Un buen valor inicial para rc es el valor de la magnitud multiplicado por 0.01, esto significa que la medida de magnitud presenta un error del 1%. Un buen valor de error para bd es 1, lo que significa que hay un error de 1 grado en la medida. Este error puede no ser proporcional con el tamaño del ángulo medido.

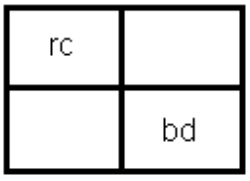

Figura 3.54. Matriz de error en la medida [24].

En la aplicación desarrollada, esta matriz de error en la medida se desarrolla como se muestra en la figura 3.55.

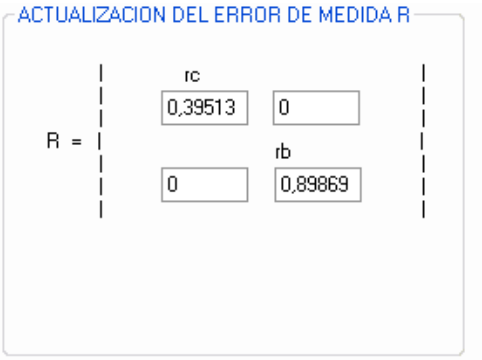

Figura 3.55. Ejemplo de definición de error en la medición de marcas.

Ahora se procede a calcular la ganancia de Kalman. Para lo cual se utiliza la siguiente expresión:

$$K = P * H^T * (H * P * H^T + V * R * V^T)^{-1}$$

La ganancia de Kalman ahora contiene un conjunto de números que indican en cuanto debe actualizarse cada una de las posiciones de la marca y la posición del robot de acuerdo con la marca re-observada. El término $(H * P * H^T + V * R * V^T)$ es llamado la innovación de la covarianza. En la figura 3.56 se observa, en la parte izquierda el cálculo de la covarianza de la innovación S y en la parte derecha el cálculo de la ganancia de Kalman K para una marca re-observada.

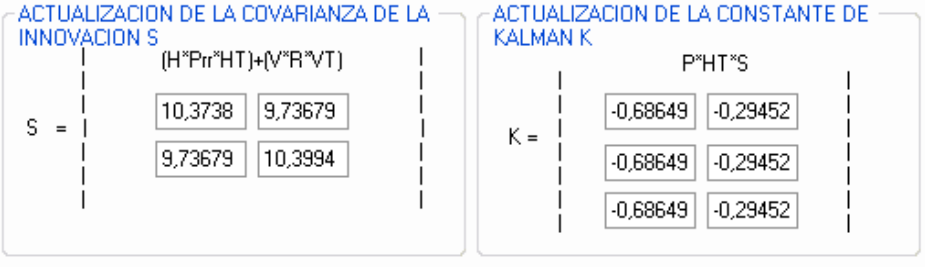

Figura 3.56. Cálculo de la covarianza de la innovación y de la constante de Kalman para una marca re-observada.

Finalmente se calcula un nuevo vector de estado X utilizando la ganancia de Kalman:

$$X = X + K * (z - h)$$

El proceso de actualización del estado del robot desarrollado en la aplicación se muestra en la figura 3.57.

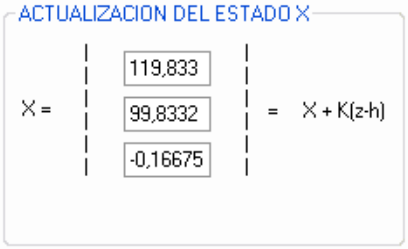

Figura 3.57. Ejemplo de la actualización del estado del robot para una marca re-observada.

Esta operación permite actualizar la posición del robot utilizando todas las posiciones de las marcas re-observadas, dado que el resultado del término $(z-h)$ no este en $(0,0)$. Nótese que $(z-h)$ da como resultado dos números los cuales son la diferencia entre lo medido y lo estimado en magnitud y ángulo, denotado como v. Este proceso se repite para cada marca asociada.

Una vez actualizado el estado del robot mediante las marcas re-observadas, es necesario adicionar al mapa las nuevas marcas observadas por el robot. Este proceso se realiza en el tercer paso del EKF descrito a continuación.

TERCER PASO DEL EKF: ADICIÓN DE NUEVAS MARCAS AL ESTADO ACTUAL.

En este paso se actualiza el vector de estado X y la matriz de covarianza P adicionando nuevas marcas. El propósito es tener más marcas que puedan ser asociadas al estado global del robot.

Primero se adiciona la nueva marca al vector de estado X; este proceso se observa en la figura 3.58. Donde se cataloga una marca observada como nueva.

$$X = [X\ x_N y_N]^T$$

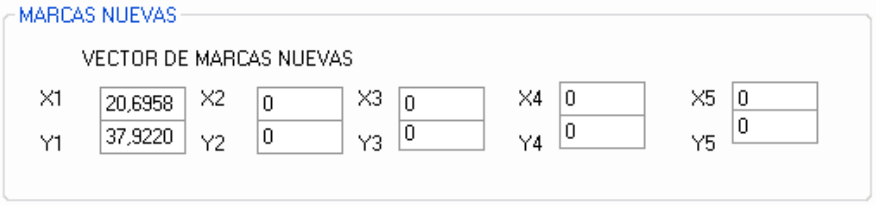

Figura 3.58. Ejemplo de una marca catalogada como nueva en el mapa.

Al adicionar una nueva marca al mapa global, se hace necesario adicionar una nueva fila y columna a la matriz de covarianza, como se indica en la figura 3.59 pintada con gris.

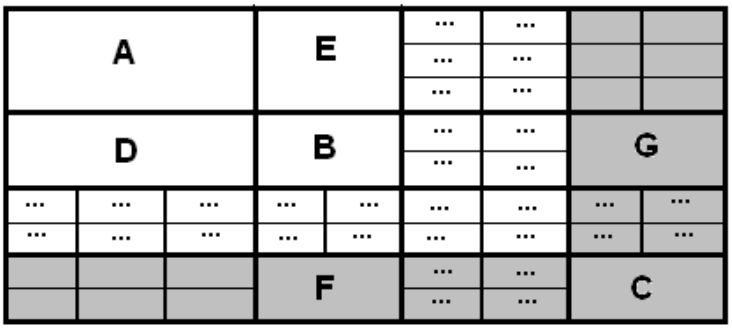

Figura 3.59. Matriz de Covarianzas P [24].

Primero se adiciona la covarianza para la nueva marca en la celda C, también llamada P^{N+1N+1}, esta es la covarianza para la $N+1$ marca:

$$P^{N+1N+1}=J_{xr}PJ_{xr}^{T}+J_{Z}RJ_{Z}^{T}$$

Pero antes de hallar el resultado de la covarianza de la nueva marca es necesario calcular el valor de las Jacobianas J_{xr} y J_z. Este proceso se muestra en la figura 3.60.

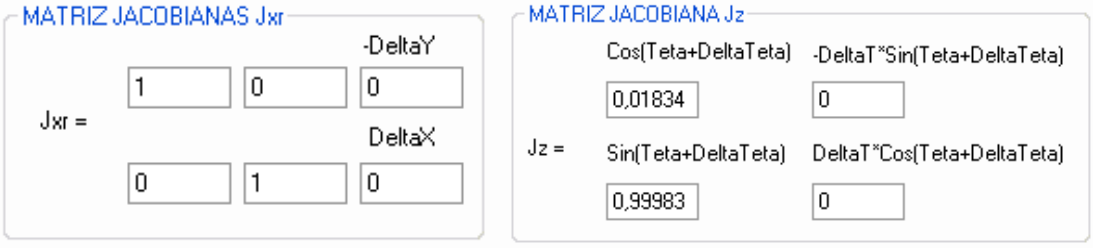

Figura 3.60. Cálculo de las jacobianas J_{xr} y J_z.

Estas matrices se calculan siguiendo el procedimiento mostrado en la sección de descripción de las matrices que componen el EKF. Además se hace necesario calcular el error en la medida de cada marca nueva, la figura 3.61 muestra este proceso.

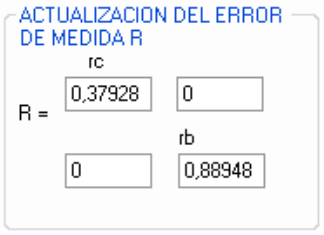

Figura 3.61. Ejemplo del cálculo del error de medida para una marca nueva.

En la figura 3.62 se observa el cálculo de la covarianza para una nueva marca adicionada al mapa.

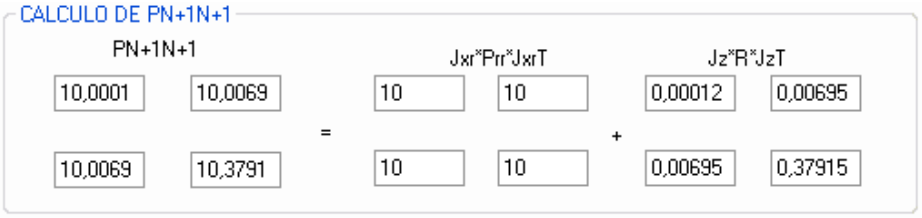

Figura 3.62. Cálculo de la covarianza para una nueva marca.

Luego se adiciona la covarianza robot-marca ante una nueva marca. Esta corresponde a la esquina superior derecha de la matriz de covarianza. Esta se calcula de la siguiente manera:

$$P^{rN+1} = P^{rr} J_{xr}^{T}$$

La covarianza marca-robot es el valor transpuesto de la covarianza robot-marca, corresponde a la esquina inferior izquierda de la matriz de covarianza.

$$P^{N+1r} = (P^{rN+1})^{T}$$

En la aplicación desarrollada estas matrices se calculan como lo muestra la figura 3.63.

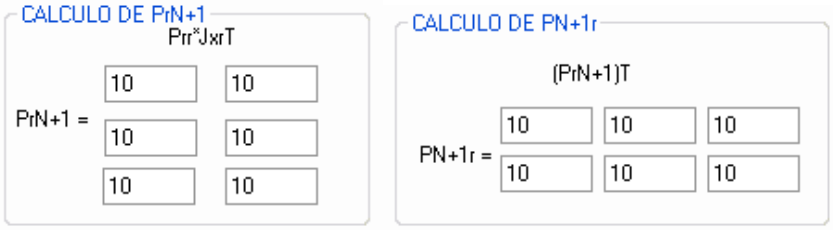

Figura 3.63. Ejemplo del cálculo de la covarianza robot-marca (izquierda) y marca-robot (derecha).

Finalmente se adiciona la covarianza marca-marca esta corresponde a la fila más baja:

$$P^{N+1i} = J_{xr}\left(P^{ri}\right)^{T}$$

La covarianza marca-marca en el otro lado de la diagonal de la matriz es el valor transpuesto:

$$P^{iN+1} = \left(P^{N+1i}\right)^{T}$$

En la figura 3.64 se observa el cálculo desarrollado en nuestra aplicación de la covarianza marca-marca y su valor transpuesto.

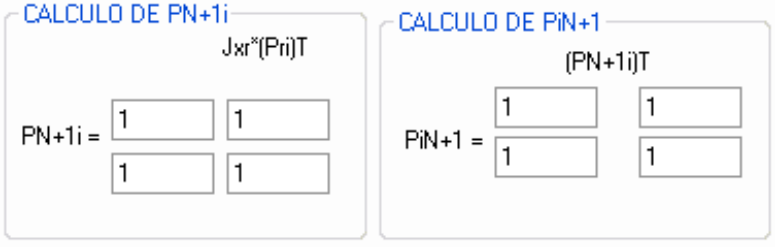

Figura 3.64. Ejemplo del cálculo de la covarianza marca-marca para una nueva marca vista.

Esto completa el último paso del proceso del SLAM. El robot esta listo para moverse nuevamente, observar marcas, asociarlas, actualizar el estado del sistema usando odometría, actualizar el estado del sistema usando marcas re-observadas y finalmente adicionar nuevas marcas.

3.9 DESCRIPCIÓN DE LA APLICACIÓN.

En esta parte se realiza la descripción de la interfaz desarrollada para el proceso de Localización y Mapeo Simultáneos utilizando una cámara web como dispositivo de

medición del entorno. La figura 3.65 muestra la pantalla de trabajo general con todas las partes que la conforman.

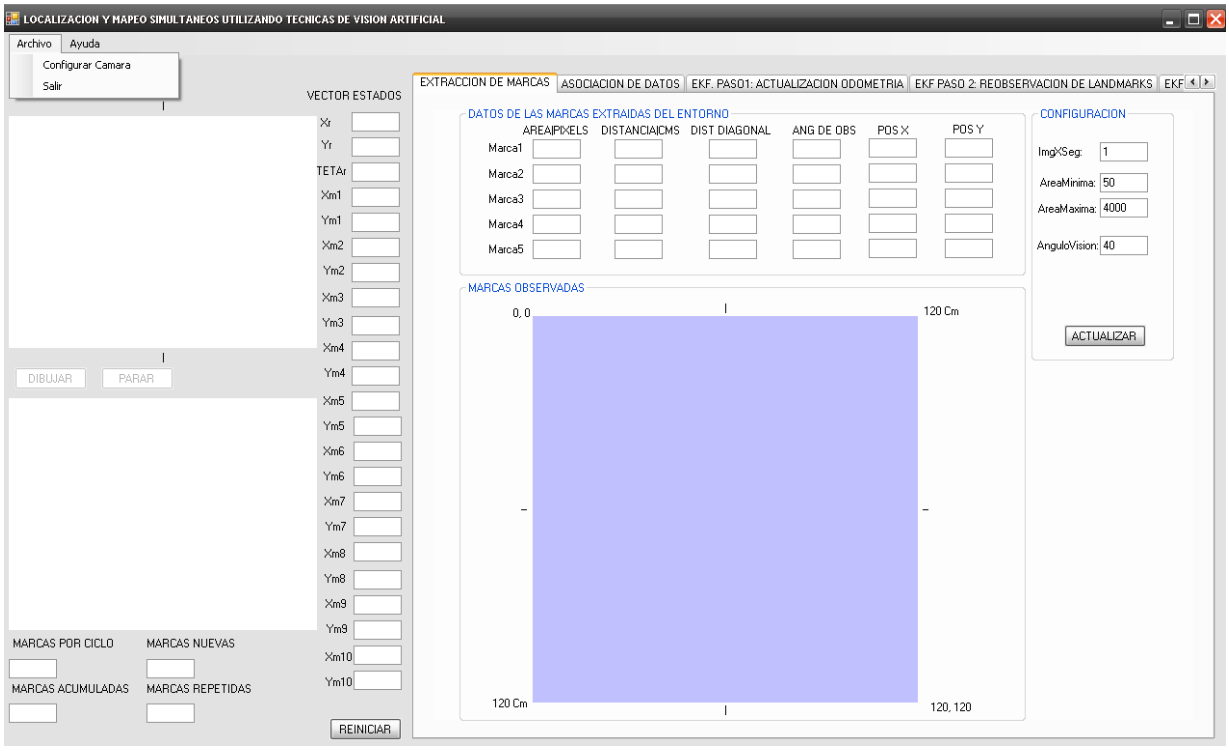

Figura 3.65. Interfaz de trabajo del Software desarrollado para la Localización y Mapeo Simultáneos utilizando una cámara web.

La interfaz consta principalmente de 4 zonas, la primera de ellas se encuentra en la parte superior de la ventana donde se disponen un botón “archivo” de donde se despliegan las opciones “configurar cámara” y “salir”, la opción “configurar cámara” despliega una nueva ventana donde se escoge una de las cámaras que puedan estar conectadas al puerto USB del computador donde se corra la aplicación, además en esta ventana se configuran el tamaño de la imagen capturada y la velocidad de captura (imágenes/seg). Cabe anotar que se eligió trabajar con un tamaño de imagen de 320x240 que es un tamaño pequeño, dado que en el entorno recorrido no se iban a muestrear figuras complejas que pudieran requerir de gran resolución, además trabajar con imágenes pequeñas aumenta el rendimiento del algoritmo en términos de velocidad de procesamiento. Además se eligió una velocidad de captura de imágenes de 15 imágenes por segundo dado que el robot se mueve a una velocidad de 33.3 cm/seg, y con esta tasa de captura se asegura baja perdida de información ante el movimiento del robot. El otro botón, “Ayuda” despliega un

manual donde se expone la utilidad de cada uno de los componentes que conforman la aplicación.

La segunda zona ubicada en la parte izquierda de la aplicación muestra, en la parte superior la visión que tiene el robot del ambiente y en la parte inferior las imágenes procesadas, con los objetos claramente diferenciados del entorno donde se encuentran. Debajo de estos dos cuadros se muestra la información correspondiente a las marcas observadas por ciclo de captura, las marcas observadas acumuladas, las marcas re-observadas en el instante anterior y las marcas nuevas.

Junto a la segunda zona se encuentra la tercera que es la zona destinada a mostrar el vector de estados. El vector de estados despliega en su parte superior el estado del robot, los termino *Xr, Yr, y TETAr* corresponden a la posición del robot en eje *X, Y,* y a su orientación θ respectivamente. Debajo de los términos expresados anteriormente, en el vector de estados se almacena las posiciones *x, y* de las marcas observadas por el robot. Cabe anotar que solo se almacenan las posiciones de marcas diferentes, es decir, en este vector no existen marcas repetidas.

La tercera zona esta compuesta por un fichero donde se desarrollan todos los pasos necesarios para llevar a cabo el EKF.

El primer paso es el de extracción de marcas (figura 3.66), donde muestra una ilustración del robot con la representación en 2D de las marcas observadas a su alrededor, además se presentan los datos extraídos de cada marca. Cabe anotar que el programa fue diseñado para observar un máximo de 5 marcas por ciclo, asumiendo este número de marcas como el número máximo de marcas que se pudieran presentar en una posición dada del robot dentro del entorno.

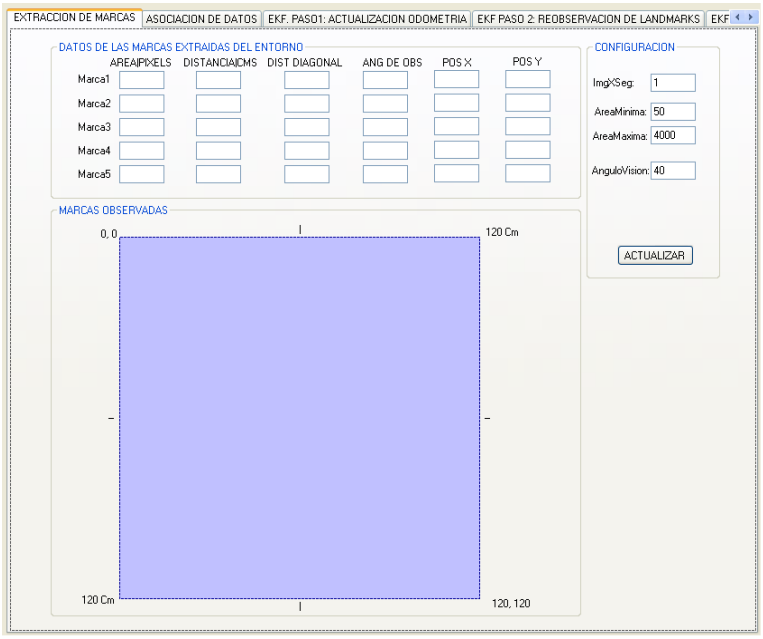

Figura 3.66. Extracción de marcas.

El segundo paso es la asociación de datos (figura 3.67) donde se muestra el robot en un mapa global, las marcas observadas también se referencian hacia este mapa global para poder determinar a partir de su posición x, y si ya han sido vistas o si definitivamente son marcas nuevas.

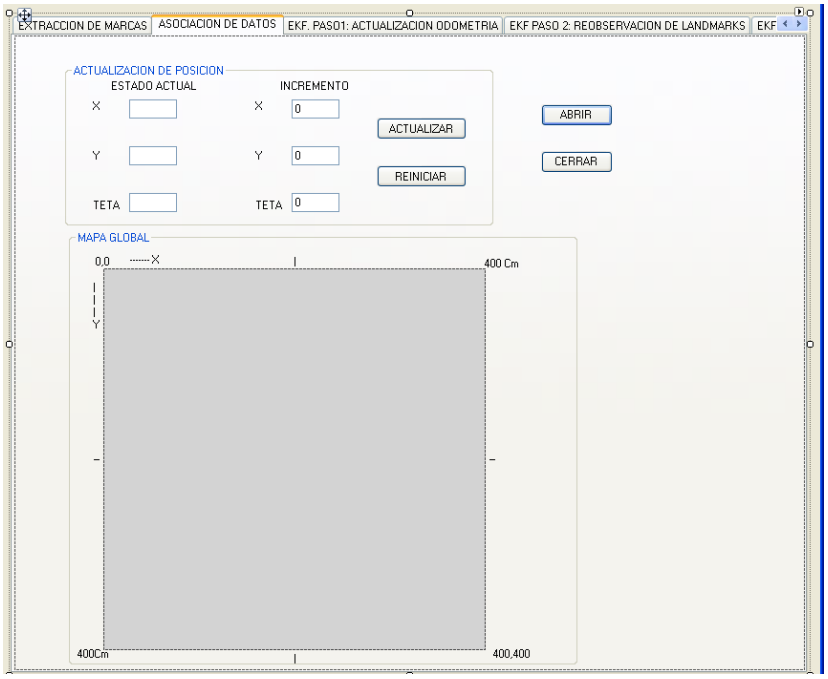

Figura 3.67. Asociación de datos.

Luego de estos dos pasos preliminares, en el fichero se pueden observar los tres pasos principales del EKF, los cuales son:

Actualización del estado mediante los datos de odometría (figura 3.68).

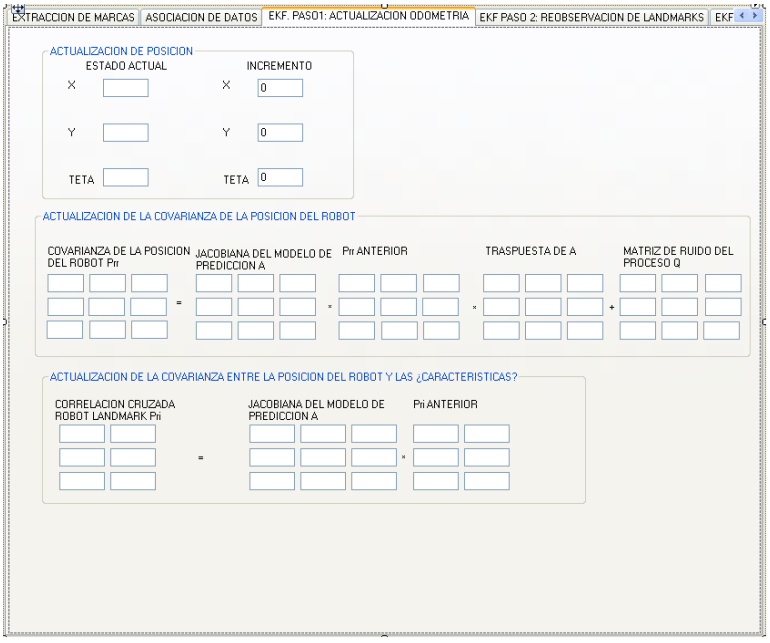

Figura 3.68. Actualización del estado del robot a través de la odometría.

Actualización del estado mediante las marcas re-observadas (figura 3.69).

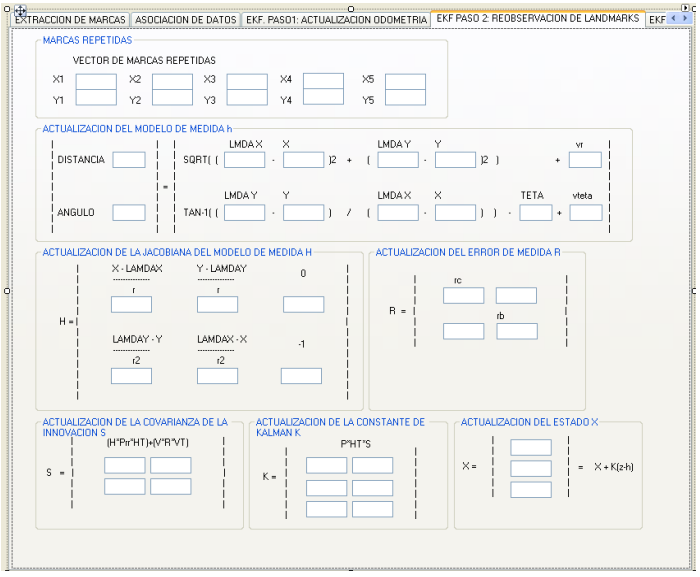

Figura 3.69. Actualización del estado a partir de la re-observación de marcas.

Adición de nuevas marcas al estado actual (figura 3.70).

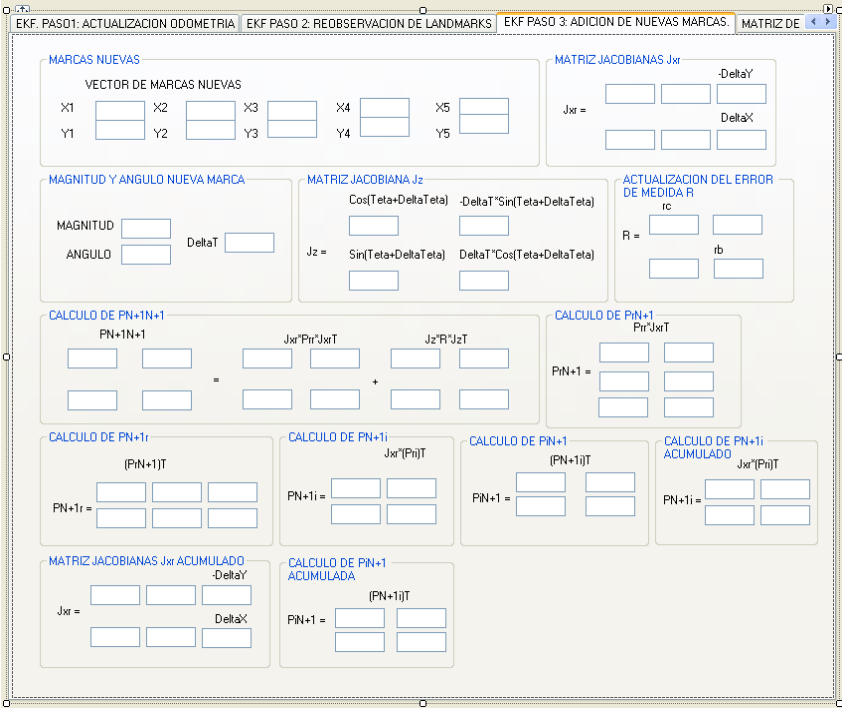

Figura 3.70. Adición de nuevas marcas al estado actual.

Estos tres pasos del Filtro Extendido de Kalman (EKF) fueron explicados con anterioridad.

Una vez desarrollados los tres pasos del EKF, en el fichero, también se puede observar como se elabora la matriz de covarianzas del sistema para las primeras seis marcas observadas en el ambiente (figura 3.71).

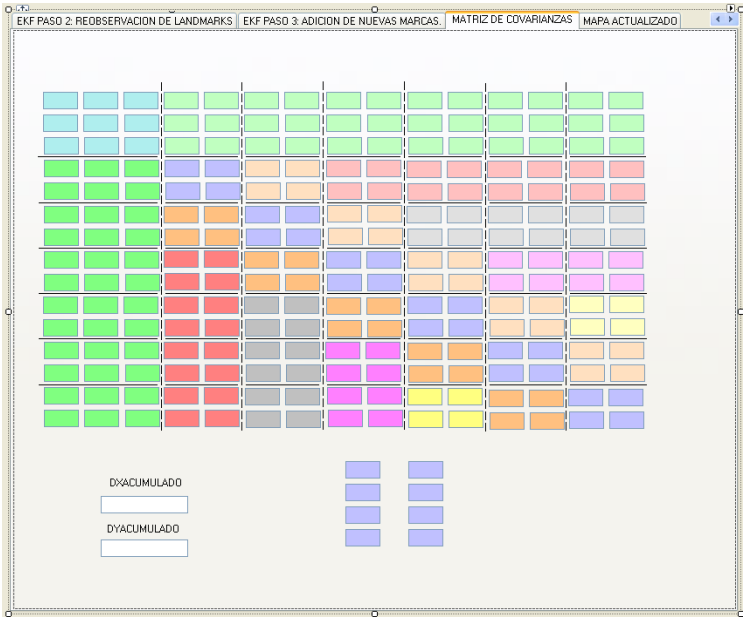

Figura 3.71. Matriz de covarianzas desarrollada por el sistema.

De otra parte, el sistema permite observar el mapa final creado por el sistema y el recorrido total realizado por el robot (figura 3.72).

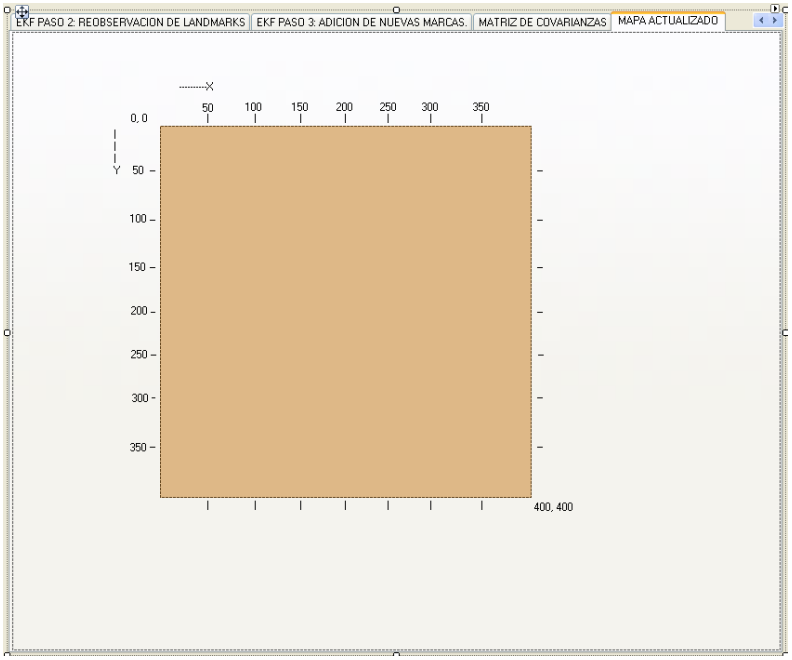

Figura 3.72. Mapa actualizado del sistema.

4. EXPERIMENTACIÓN, RESULTADOS, ANÁLISIS Y CONCLUSIONES

En este capítulo se muestra el proceso de experimentación al que fue sometida la aplicación de localización y mapeo simultáneos SLAM utilizando un sistema de visión. Además se da a conocer los resultados obtenidos a partir de dicha experimentación y, por último, se realiza el análisis de los resultados obtenidos de la mano de las conclusiones adquiridas después de realizado el proyecto.

4.1 Experimentación y Resultados

En este punto es necesario describir el entorno en el cual el robot realiza el proceso de localización y mapeo simultáneos. Para tal propósito, en la figura 4.1 se ilustra el diseño y las dimensiones finales bajo las cuales se construyó el ambiente a través del cual se movería el robot ATIBOT.

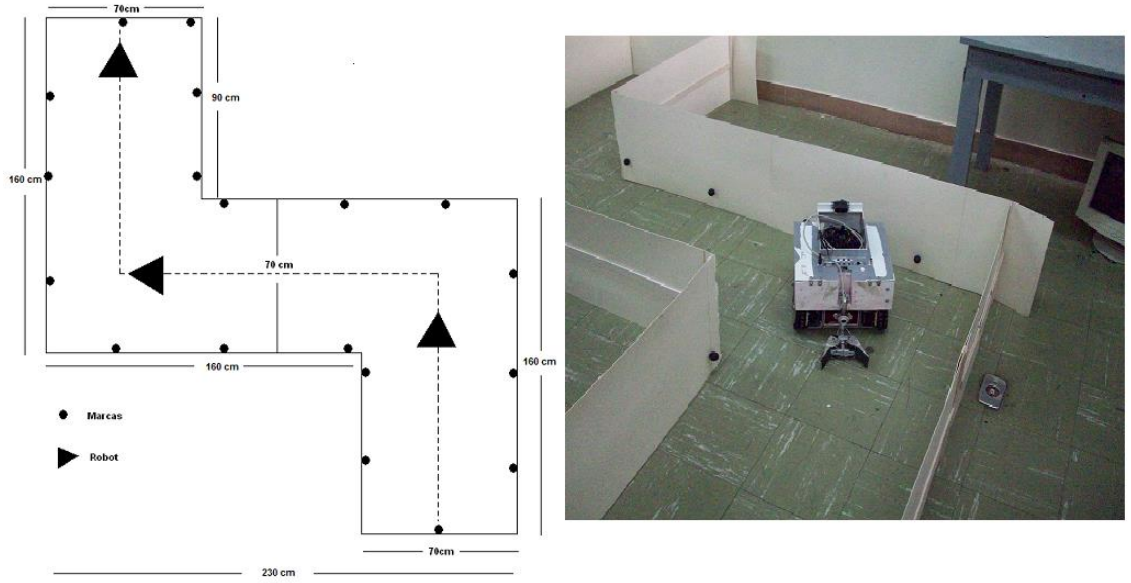

Figura 4. 1. Ambiente de prueba para el algoritmo de Localización y Mapeo Simultáneos SLAM.

El proceso de experimentación consistió en rodar el robot a través del entorno anteriormente expuesto. Cabe anotar que en este punto era necesario garantizar buenas condiciones de iluminación para el experimento, debido a que el procedimiento para calcular la distancia a la que se encontraban las marcas, con respecto al robot, está basado en la observación de las marcas mediante una cámara web y las medidas de distancias podrían verse alteradas al no contar con unas condiciones de luz propicias. El objetivo de la experimentación fue determinar qué tan exacta resulta la construcción de un mapa y la ubicación del robot dentro de este ambiente, utilizando la aplicación desarrollada durante el transcurso de este trabajo. Con este propósito se equipó al robot con una cámara web en su parte frontal, de tal manera que mediante la información recogida por ésta y mediante la información obtenida de los *encoders* con los que cuenta el robot, se pudiera realizar el proceso de localización y mapeo simultáneos SLAM. La figura 4.2 muestra la ubicación de la cámara *web* en el robot ATIBOT, el puerto de comunicación serial a través del cual éste entrega la información referente a su desplazamiento y la conexión vía manos libres con un celular para enviar al robot las ordenes de movimiento.

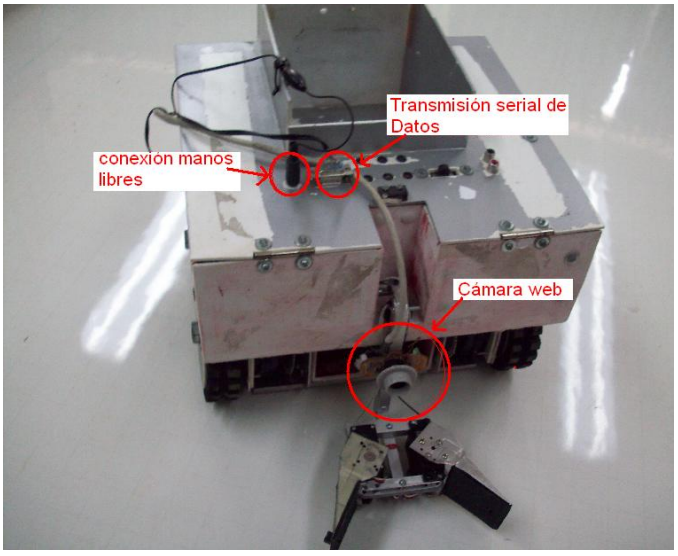

Figura 4 2. Ubicación de la cámara web en el robot ATIBOT.

El primer paso en el proceso de experimentación consistió en ajustar las medidas de los sensores odométricos y estereoceptivos del robot; para este caso, una cámara web. Este ajuste fue necesario debido al alto grado del error en la medida generado por el sistema odométrico del robot y, aunque el algoritmo SLAM EKF tiene como objetivo mejorar estos

errores, los datos entregados por el robot se encontraban bastante lejos de la realidad. Con este procedimiento, se pretendía acercar a la realidad los datos entregados por el robot más no disminuir totalmente el error entre lo real y lo medido, porque entonces no tendría sentido la utilización del Filtro de Extendido de Kalman(EKF) para mejorar su ubicación.

El ajuste de los sensores odométricos se realizó de manera experimental, es decir, se le transmitió la orden al robot para que tuviera desplazamientos de diferentes magnitudes y se realizó la lectura de los datos generados por sus sensores odométricos. Una vez obtenida esta información se pudo constatar que ésta no era lo suficientemente fiel como para garantizar el correcto funcionamiento del algoritmo SLAM. Por lo que fue necesario ajustar los datos de desplazamiento entregados por el robot a su desplazamiento real.

Estos errores se presentaban de diferentes maneras al muestrear los datos de odometría en diferentes periodos, pues al mover el robot una distancia corta, alrededor de 20cm, y muestrear, se observó que la medida entregada estaba muy por debajo del desplazamiento real del robot, mientras que al muestrear después de un movimiento mayor, alrededor de 1m, se obtenía una medida más cercana al desplazamiento real del robot. Por lo anterior, antes de realizar el procedimiento de experimentación se corrigió la información de odometría entregada por el robot, muestreando los desplazamientos reales del robot y comparándolos con los datos entregados por el robot en una gráfica para posteriormente sacar una función que permitiera aproximar los datos medidos al desplazamiento real del robot. La figura 4.3 muestra los datos medidos vs datos reales en ambos ejes, y la función hallada para realizar la aproximación de los datos medidos al desplazamiento real del robot.

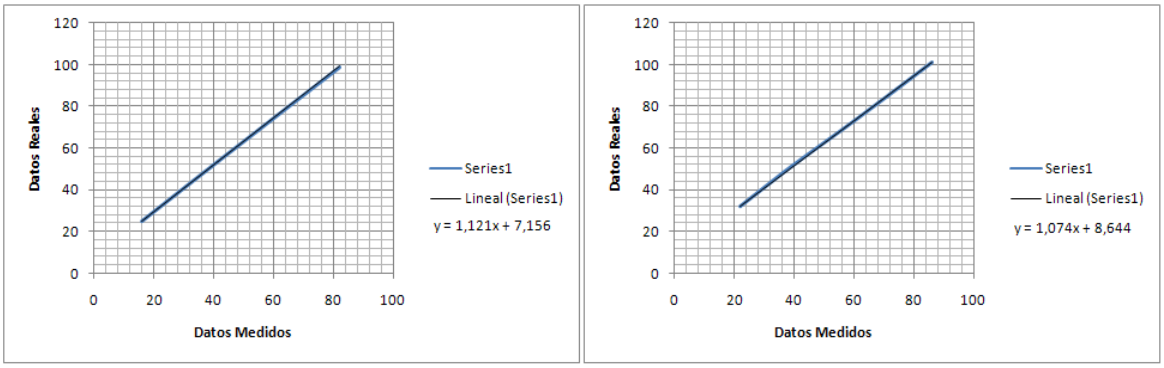

Figura 4 3. Rectas mediante las cuales se realizo el mejoramiento en las medidas de desplazamiento del robot. A la izquierda la función con la que se mejoraron los datos de

desplazamiento en el eje Y, a la derecha la función con la que se mejoraron los datos de desplazamiento en el eje X.

En la figura anterior, *X* representa los datos medidos ante un desplazamiento del robot, y *Y* representa el desplazamiento real realizado por el robot, la idea de hallar estas funciones es poder aplicarlas a los datos medidos por el robot, para que como resultado generen un dato de desplazamiento más cercano a desplazamiento real. Con esta función se logró disminuir el error en la medida de desplazamiento a 3 cm aproximadamente, mientras que sin aplicar esta ecuación se presentaban errores hasta de 25 cm.

Otro inconveniente que presentó el sistema odométrico con respecto al desplazamiento real del robot, fue el desfase en su trayectoria lineal. Este problema se presentó por la incoherencia entre los datos medidos de desplazamiento del robot y sus desplazamientos reales por cuestiones mecánicas y de programación, pues mientras los datos de los sensores odométricos indicaban un desplazamiento en línea recta, el desplazamiento real del robot era diagonal. Este fue otro punto a corregir durante el proceso de experimentación. La figura 4.4 muestra la diferencia entre la trayectoria real realizada por el robot y la trayectoria medida según sus datos de odometría.

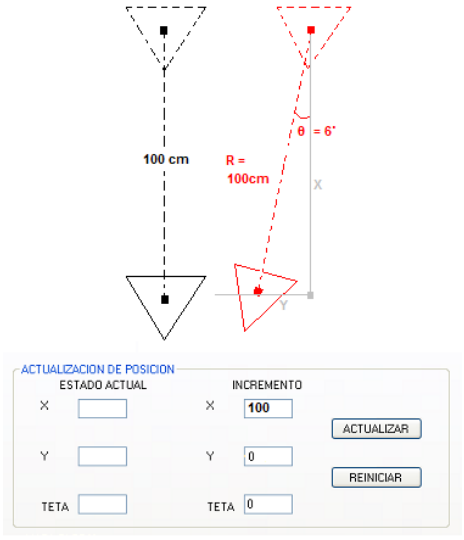

Figura 4 4. Diferencia entre movimiento real del Robot y datos de desplazamiento entregados por él. En la parte superior, a la izquierda el desplazamiento según los datos del robot y en la parte derecha el desplazamiento real. En la parte inferior los datos entregados por el robot luego del desplazamiento de 1 metro.

Por aspectos mecánicos del robot, cuando se le ordenaba avanzar en línea recta hacia adelante, éste realizaba un avance que resultaba desfasado en un ángulo de 6˚

aproximadamente con respecto al avance ordenado como se muestra en la figura 4.4. Los datos entregados por el robot estaban dados para movimientos solo en el eje X o solo en el eje Y, pero no en ambos ejes simultáneamente.

Según lo anterior el robot entregaba los datos de desplazamiento en un solo eje con los que la figura resultante de su desplazamiento sería una línea recta en uno de los dos ejes, cuando en realidad esta línea recta tendría desplazamientos tanto en el eje X como en el eje Y. Por lo anterior se realizó la siguiente corrección en la entrada de datos al sistema.

$$y = R \sin \theta$$

Donde y es el desplazamiento en el eje Y no considerado en la figura 4.4; R es el desplazamiento leído de los sensores de odometría y θ es el ángulo de desfase en el desplazamiento. Con esta ecuación se pudo actualizar el desplazamiento en el eje Y que no se consideraba según las medidas entregadas por el robot. Fue así, como se consiguió una representación más cercana del desplazamiento real del robot, la figura 4.5 muestra las mejoras obtenidas al aplicar la ecuación para hallar el desplazamiento en el eje Y.

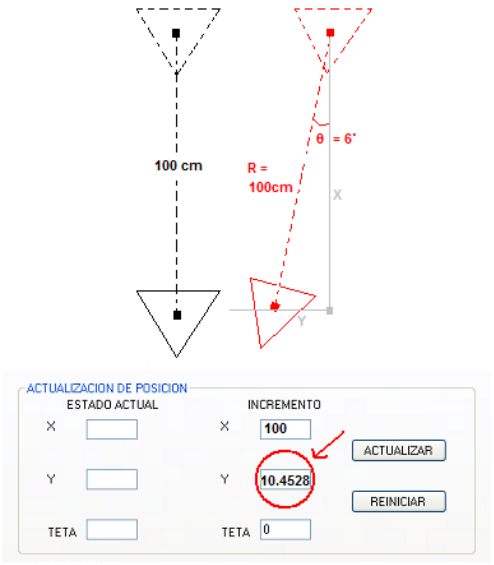

Figura 4 5. Corrección en la medida de desplazamiento entregado por el robot.

Siguiendo con la experimentación cuando al robot se le daba la orden de retroceder o de moverse hacia atrás, se observó un comportamiento aun más variable y complejo de mejorar vía software. Por esta razón, se determinó que el robot únicamente se

desplazaría hacia adelante durante el desarrollo de la experimentación, para conseguir así los mejores resultados posibles.

Además del desfase en el desplazamiento del robot, fue necesario compensar el desfase en su orientación, para ello se hizo necesario sumarle 6 grados a la orientación actual del robot cuando este comenzaba a moverse en uno u otro eje, este ángulo se adicionó solo para la primera vez que el robot se moviera y se mantuvo así, siempre y cuando el movimiento del robot se mantuviera en la misma orientación, cuando el robot realizaba un giro, recordemos que por su programación, el robot solo puede realizar giros de 90 grados o de -90 grados; si se había modificado la orientación del robot, también era necesario compensar este desfase en las medidas de giro del mismo. Este procedimiento se realizó como se muestra en la figura 4.6.

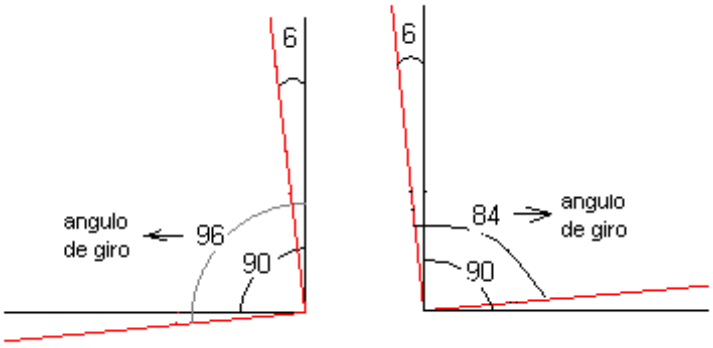

Figura 4 6. Compensación en la orientación del robot al realizar un giro.

Luego de estas correcciones se obtuvo una representación más cercana del movimiento real del robot. La figura 4.7 muestra la diferencia entre la representación del movimiento del robot sin correcciones y con las correcciones sobre las medidas de desplazamiento y orientación.

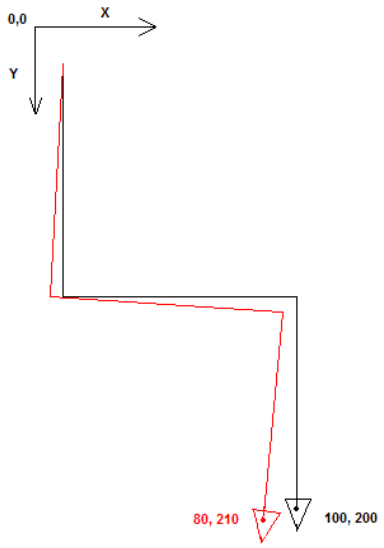

Figura 4 7. Corrección de datos de odometría entregados por el robot. La línea negra representa los datos entregados por el robot ante un movimiento en forma de s, que no se acerca tanto al desplazamiento real como los datos representados por la línea roja donde se realiza la corrección tanto en posición como en orientación.

Una vez fueron ajustados los datos de odometría entregados por el robot, se procedió a realizar el ajuste de los datos de medida de las marcas del entorno, este proceso se realizó re-muestreando los datos de área (en pixeles) y de distancia (en centímetros) para con ellos realizar el ajuste de las constantes de las ecuaciones descritas en el proceso de extracción de características (capitulo 3).

Habiendo mejorado las medidas de los sensores tanto odométricos como estereoceptivos se procedió a la etapa de experimentación.

En el proceso de experimentación se realizaron varios procedimientos, el primero de ellos consistió en generar un mapa moviendo el robot en intervalos de 50 cm aproximadamente. Luego de mover el robot por todo el entorno en los intervalos anteriormente mencionados, se obtuvo el mapa que se muestra en la figura 4.8.

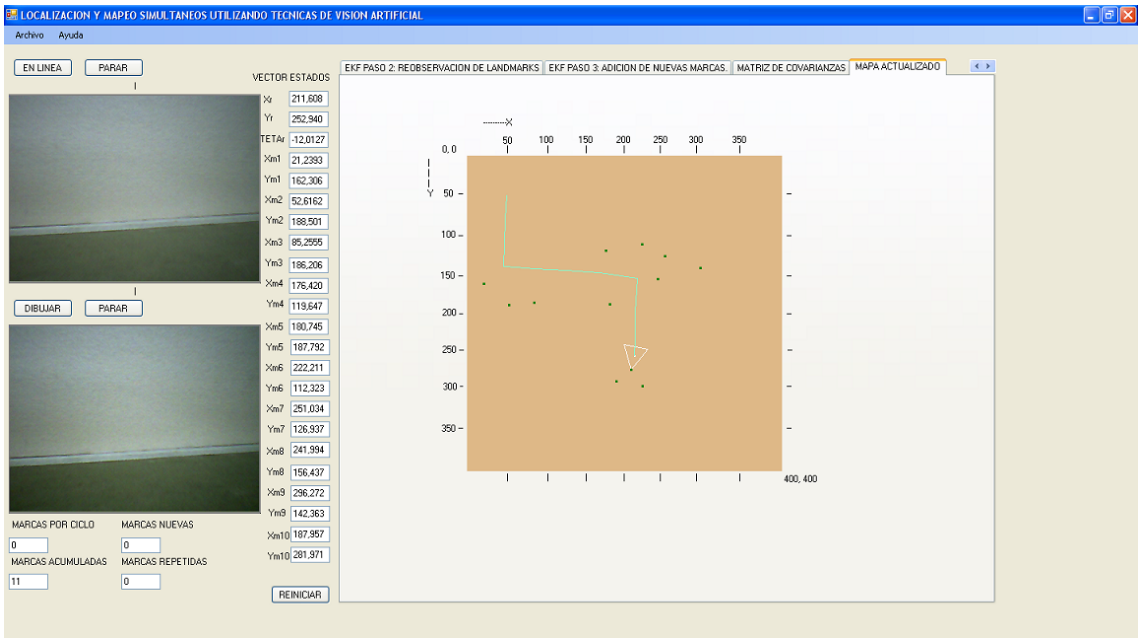

Figura 4 8. Mapeado del entorno, realizado en intervalos de 50 cm.

En la figura anterior se puede observar el mapa global pintado a partir de las marcas ubicadas en el entorno, en el lado izquierdo se observa la última imagen vista por el sistema de visión en la cual no se observa ninguna marca por estar muy cerca de la marca que se encuentra frente a él, en la parte central se observa el vector de estados, el cual presenta en sus primeras tres elementos, la posición X_r, Y_r del robot y su orientación (Tetha) y de ahí en adelante presenta las posiciones X, Y de todas las marcas existentes en el entorno excluyendo las que fueron observadas más de una vez. Por último en la parte derecha de la figura 4.8 se observa la pestaña correspondiente al gráfico que representa el entorno observado luego de ser recorrido y de habérsele aplicado el Filtro Extendido de Kalman. A continuación se describe paso a paso el proceso de SLAM realizado en este experimento.

4.1.1 Experimento 1.

Primero, hay que aclarar que en el gráfico, se definió como posición inicial del robot la posición 50, 50, para que se pudieran observar las marcas vistas por el robot tanto a la derecha como a la izquierda de su eje central de observación. Se le ordenó al robot moverse frontalmente en intervalos de 30 cm aproximadamente y en las pausas se realizó el proceso de SLAM. La figura 4.9 muestra el desplazamiento frontal inicial que se le dió al robot a través del entorno.

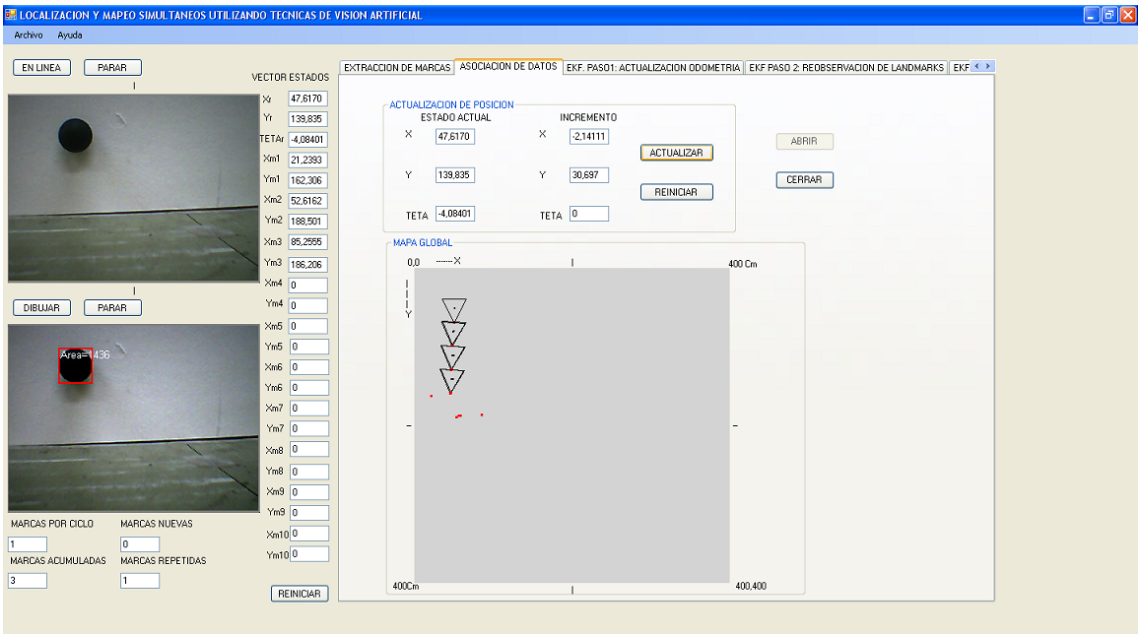

Figura 4 9. Desplazamiento frontal realizado por el robot.

En la figura anterior se puede observar los tres movimientos frontales realizados por el robot, cuando se realizó la actualización del sistema en la posición inicial, el robot observó las tres marcas del entorno ubicadas frente a él catalogándolas como marcas nuevas, agregándolas al vector de estados durante el tercer paso del EKF, la figura 4.10 muestra las primeras tres marcas del entorno observadas por el robot. Luego se movió el robot unos 30 cm adelante en el eje Y del mapa global, en éste punto el robot observó una sola marca (la del centro), la cual clasificó como marca re-observada por lo que no la agrego al vector de estados y con la cual realizó la corrección de la posición del robot en el segundo paso del EKF. Este procedimiento se repitió en dos ocasiones moviendo el robot dos veces más en intervalos de 30 cm aproximadamente. En la figura anterior se observa que en las tres últimas posiciones en las que estuvo el robot, se realizó la corrección del estado del robot correspondiente al paso dos del EKF.

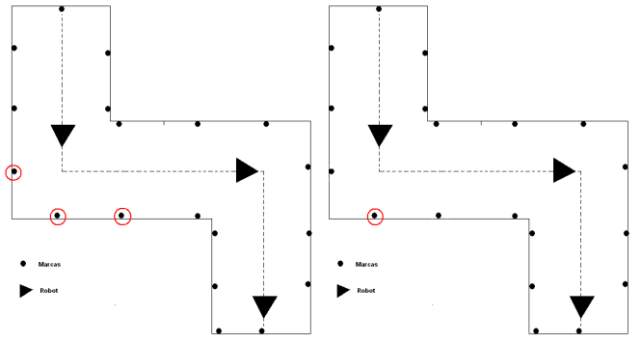

Figura 4 10. Marcas observadas en la primera trayectoria del Robot. La figura de la izquierda muestra las tres marcas observadas por el robot en la posición inicial y la figura de la derecha muestra la marca re-observada en las siguientes tres posiciones.

En el momento en que se actualizó el estado del robot en la posición inicial, el algoritmo ejecutó todos los pasos involucrados en el proceso de EKF SLAM, desde la adquisición de la información estereoceptiva y odométrica pasando por la asociación de datos, hasta los 3 pasos del EKF propiamente dichos. La figura 4.11 ilustra los pasos del EKF involucrados en el muestreo de la posición inicial del robot.

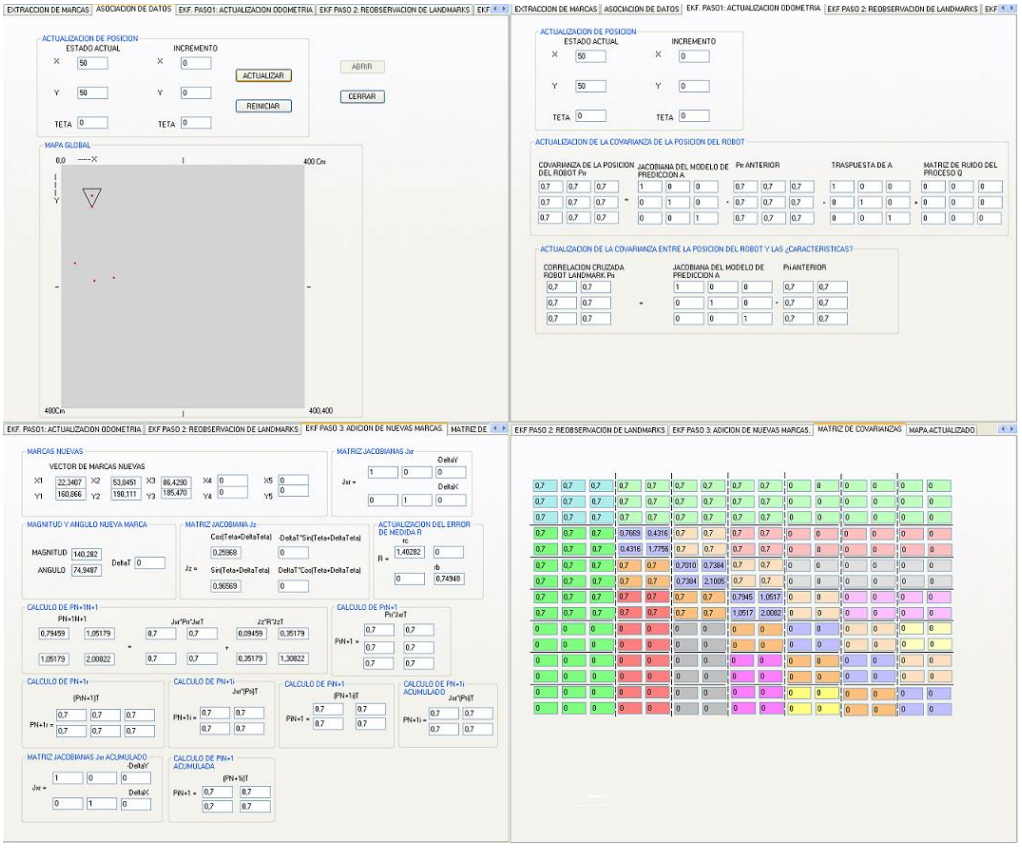

Figura 4 11. Pasos EKF involucrados en el muestreo de la posición inicial del robot.

En la figura anterior se observa la actualización del estado actual, la actualización del paso 1 y 3 del EKF y la actualización de la matriz de covarianzas con las marcas actualmente vistas. El segundo paso del EKF no se muestra, porque en este punto no aportó ninguna alteración al estado del sistema al no existir marcas re-observadas.

Al mover el robot hacia adelante y muestrear, se observó una marca como repetida, es en este punto donde el EKF SLAM realizó la corrección del estado actual del robot valiéndose de la información de posición de esta marca re-observada. En la figura 4.12 se observa los pasos involucrados en la re-observación de marcas realizada en el primer avance del robot.

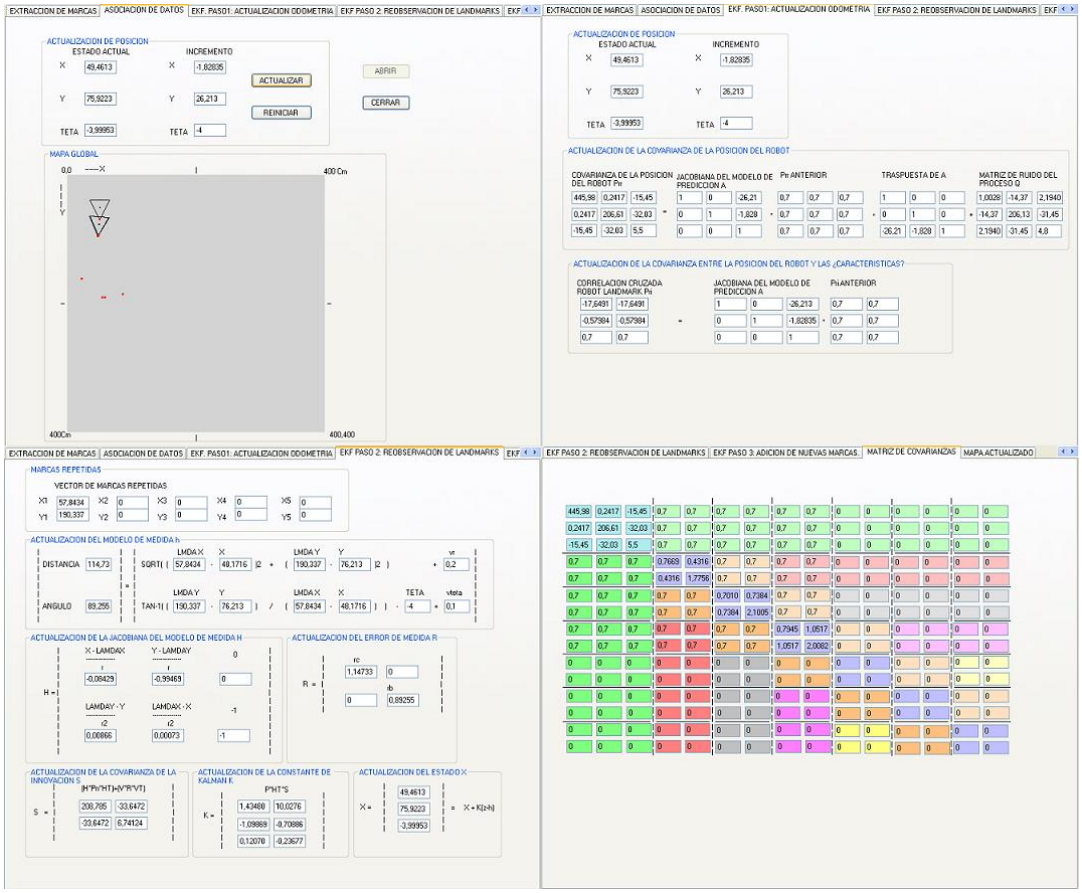

Figura 4 12. Pasos EKF involucrados en el primer movimiento del robot.

En la figura anterior se observan los pasos EKF involucrados ante la re-observación de marcas en el entorno. El tercer paso no se muestra pues este no altera el estado del sistema al no existir observación de marcas nuevas en este punto.

Se observa que la matriz de covarianzas no se altera pues la marca que esta siendo observada, ya ha sido vista con anterioridad y por lo tanto ya ha sido adicionada al estado general del sistema.

Luego de mover el robot hacia adelante, se procedió a girarlo para mapear la segunda sección del entorno. En la primera actualización del estado luego del giro, el robot observó dos marcas nuevas las cuales se agregaron al vector de estados del sistema, estas marcas se observan en la figura 4.13.

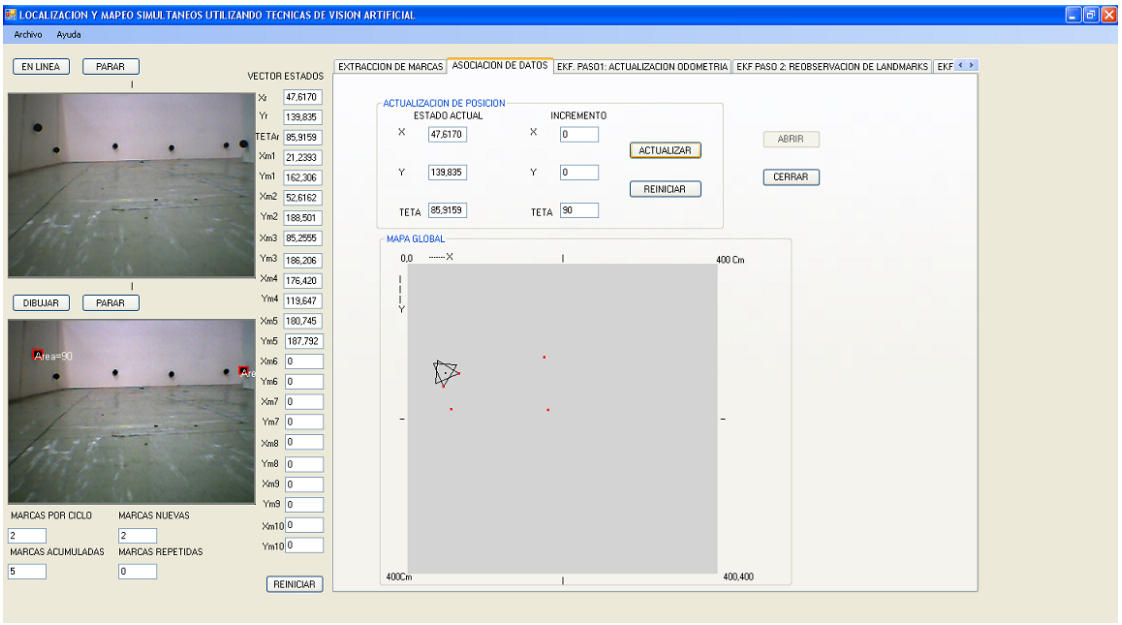

Figura 4 13. Primeras marcas observadas en la segunda sección del entorno.

Las posiciones correspondientes a las marcas observadas en la figura anterior se muestran en la figura 4.15.

Una vez realizada la primera observación luego del giro del robot, se procedió a mapear el resto de esta sección del entorno, realizando actualizaciones del sistema cada 40 cm aproximadamente, durante este proceso solo se corrigió la posición del robot en la última observación, dado que hasta este punto se observaron como repetidas dos marcas del entorno. En la figura 4.14 se muestran las marcas observadas en el segundo tramo del entorno.

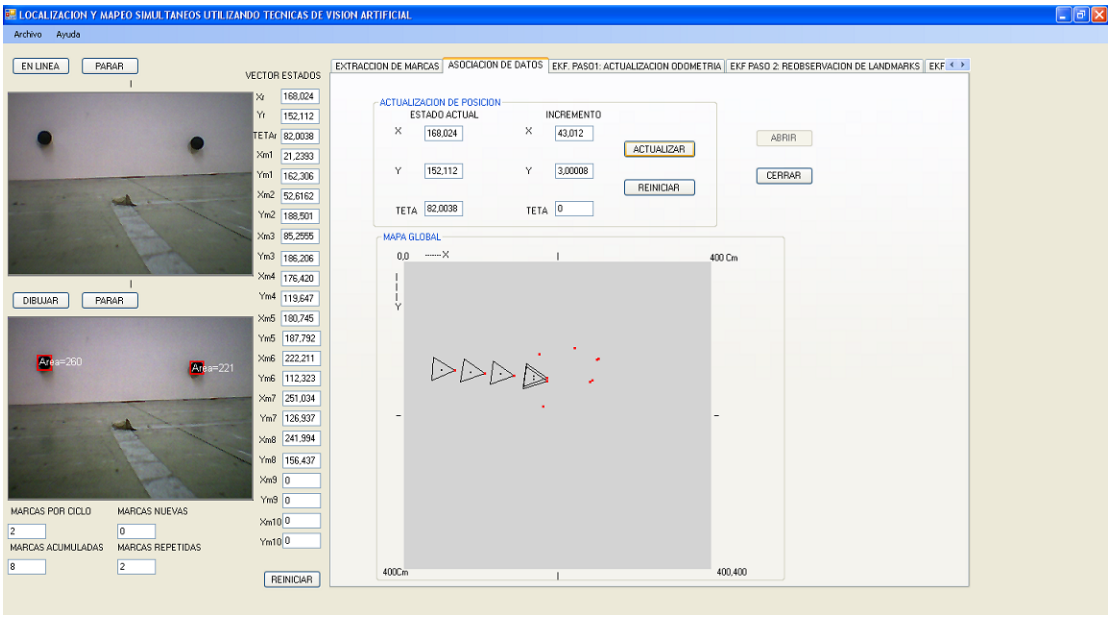

Figura 4 14. Marcas observadas durante la travesía del segundo tramo del entorno.

En la figura anterior se observan las primeras dos marcas observadas luego del primer giro a la izquierda del robot, más adelante en el siguiente desplazamiento del robot, este observó una marca a su derecha; en su segundo desplazamiento el robot observó dos marcas ubicadas en la pared frente a él y en el último movimiento el robot observó las mismas dos marcas vistas en el estado anterior por lo que se realizó la respectiva corrección de posición y orientación del estado actual del robot. En la figura 4.15 se evidencian las marcas en el entorno observadas por el robot durante su travesía por el segundo tramo del ambiente.

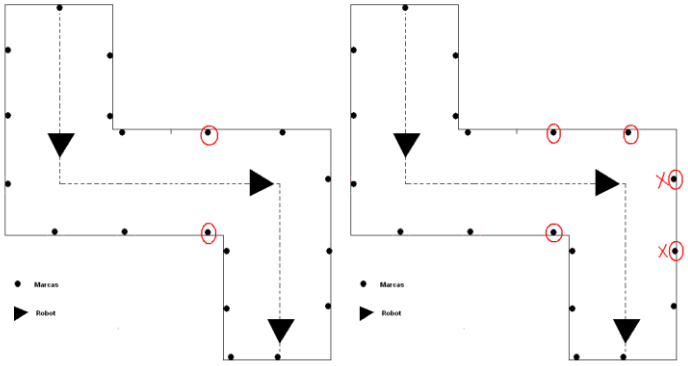

Figura 4 15. Marcas observadas durante la travesía del segundo tramo del entorno. A la izquierda, las marcas observadas en la primera observación luego del primer giro, a la derecha las marcas observadas en los avances posteriores, las marcas con una x son las marcas re-observadas en el último punto.

Para poder realizar el giro a la derecha que permitiera explorar la última sección del entorno fue necesario avanzar un poco más en el eje en el que se encontraba el robot, en este punto se pudo observar la detección de una marca indeseada, pues lo que detectó el sistema de visión no fue una marca propiamente dicha, sino una contaminación producida por las sombras generadas por no tener condiciones de luz adecuadas. La figura 4.16 muestra el ruido observado en el último movimiento del robot antes de realizar un segundo giro.

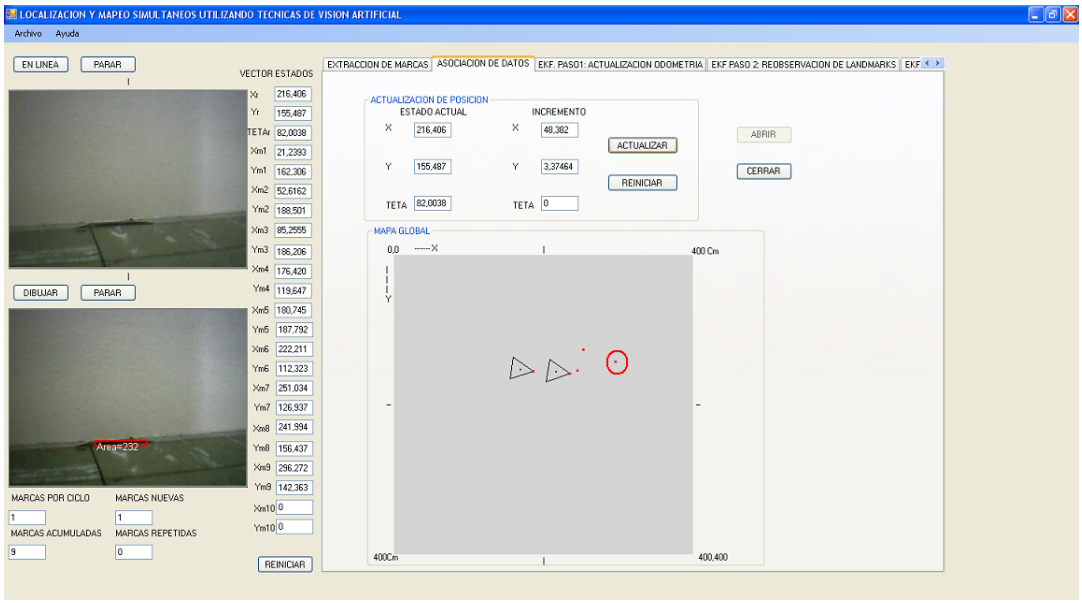

Figura 4 16. Ruido observado en el último desplazamiento de la segunda sección.

En la figura anterior, la marca encerrada en el círculo rojo es el ruido observado en la imagen capturada por el sistema de visión. En la imagen capturada, se puede observar que la razón por la cual se encuentra alejada la marca (ruido) del robot es por su área, es decir, el número de píxeles que la componen es pequeña.

En este punto se realizó un giro a la derecha para proceder a la exploración del último tramo del entorno, entonces se observaron 2 nuevas marcas ubicadas al final del pasillo, el algoritmo las clasificó como marcas nuevas y se les dió el tratamiento correspondiente en el tercer paso del EKF. Luego se hizo un avance de 40 cm aproximadamente, punto en el cual fue re-observada una de las marcas anteriormente vistas, razón por la cual el sistema realizó la corrección del estado del robot mediante el segundo paso del EKF. La figura 4.17 muestra las marcas observadas y re-observadas en la travesía del último tramo del entorno.

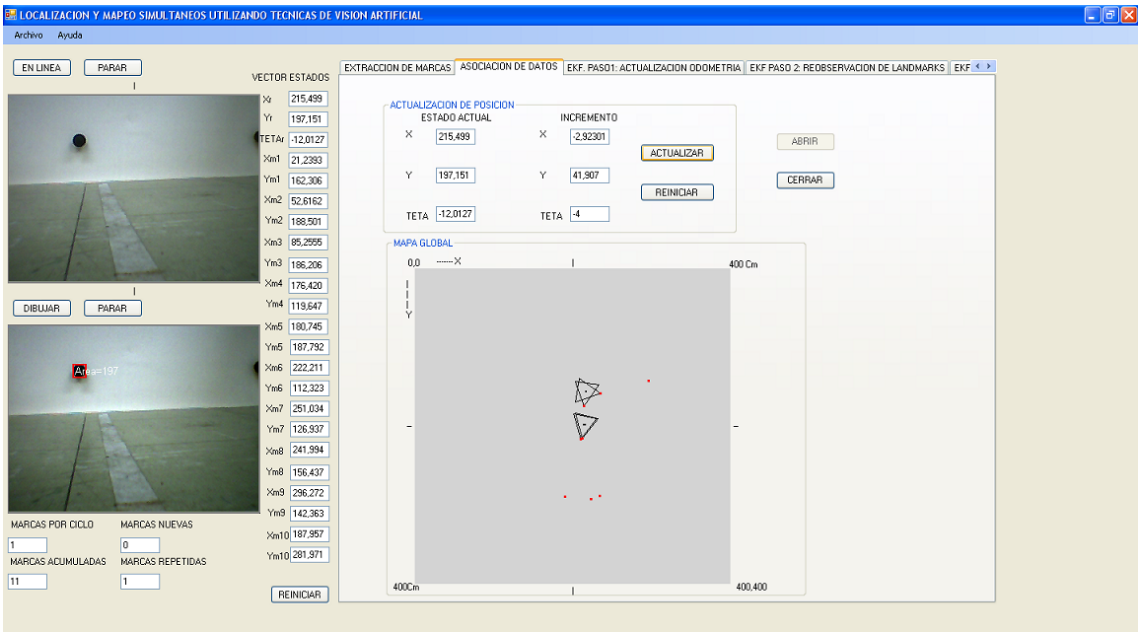

Figura 4 17. Marcas observadas en el último tramo del entorno.

En la figura anterior se observa como el robot luego de realizar un giro a la derecha, detecta dos marcas en el fondo de esta sección, una vez se realiza un avance frontal, este detecta nuevamente una de las marcas anteriormente vistas. En la figura 4.17 la marca re-observada es la que se encuentra más alejada. La ubicación en el entorno de las dos marcas observadas en esta última sección se muestra en la figura 4.18.

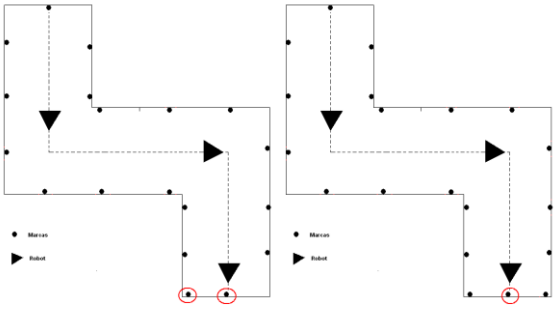

Figura 4 18. Marcas observadas en la travesía del tercer tramo del entorno. La imagen de la izquierda muestra las marcas observadas en el primer muestreo luego del giro a la derecha y la imagen de la derecha muestra la imagen que fue re-observada luego de una avance frontal de 40 cm aproximadamente.

Por último se realizó un avance hacia delante de 20 cm aproximadamente, pero luego de éste, el sistema de visión no detectó ninguna marca por estar muy cerca de la marca que se encontraba frente a él. En este punto el sistema solo aplicó el primer paso del EKF (Actualización del estado mediante datos de Odometría) dado que no se observaron ni

marcas nuevas ni marcas repetidas. La figura 4.19 muestra el último avance realizado por el robot en este experimento.

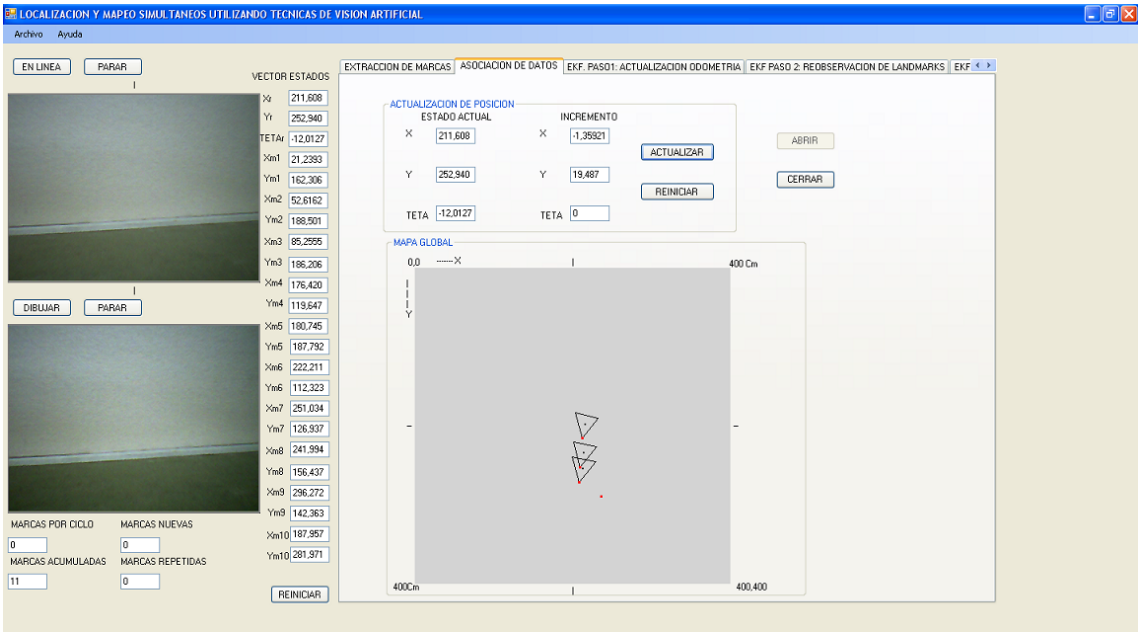

Figura 4 19. Ultimo avance del robot en el tercer tramo del entorno.

De esta manera se generó el mapa del entorno y se localizó el robot en cada punto de actualización utilizando el Filtro Extendido de Kalman y la cámara web como sistema de visión.

Para realizar la comparación entre lo real y lo estimado, se referenció el entorno real al sistema de coordenadas manejado en la ventana de mapeado de la aplicación SLAM, es decir se estableció una posición de referencia 0,0 en el entorno real que coincidiera con la posición inicial 0,0 del espacio de mapeo en la aplicación. Este procedimiento se observa en la figura 4.20.

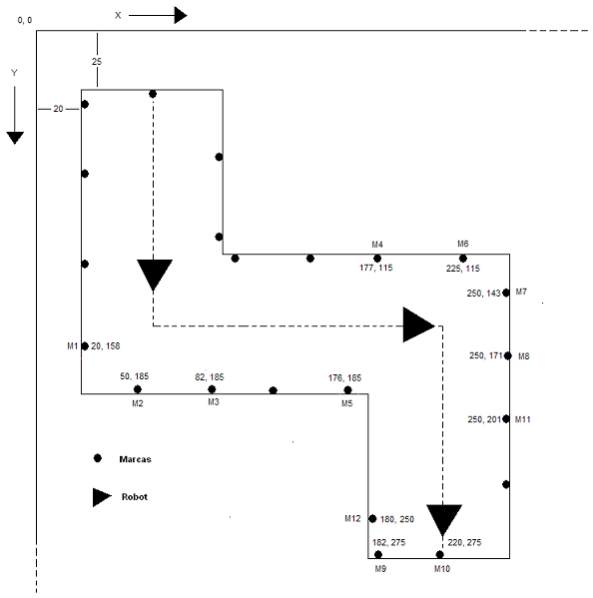

Figura 4 20. Posiciones de las marcas en el entorno real.

Fue así como se estableció el entorno con el que se compararían los resultados obtenidos durante el proceso de experimentación. La figura 4.21 muestra la diferencia entre el entorno real y el mapa generado por la aplicación desarrollada en este proyecto.

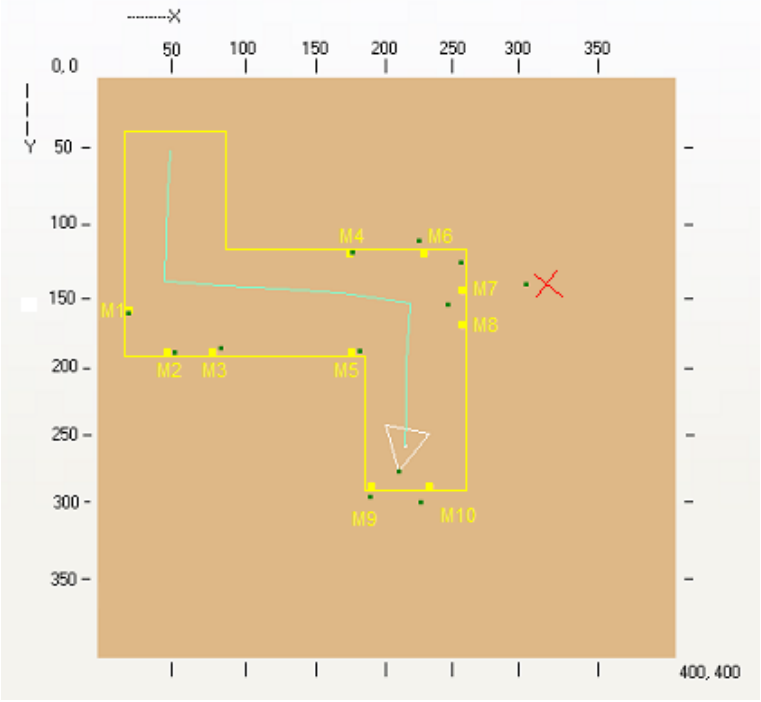

Figura 4 21. Comparación entre el entorno real y el mapa generado por la aplicación SLAM Experimento1.

En la gráfica anterior, se puede observar que en la posición inicial, las marcas del mapa generado (marcas verdes), estaban muy cercanas a las posiciones de las mismas marcas pero en el entorno real, que está representado por las líneas y por las marcas amarillas. A medida que el robot avanzó en su recorrido, las medidas de las posiciones de las marcas del mapa generado, fueron alejándose de las posiciones de las mismas del entorno real. Esto, debido a los errores acumulados en la medición de odometría; sin embargo, cada vez que el sistema observó marcas repetidas, este realizó la corrección de la posición actual del robot con lo que se compensó en gran medida este efecto. La tabla 4.1 muestra la diferencia entre las posiciones de las marcas reales y las marcas calculadas por la aplicación. Cabe aclarar que no se comparó la marca detectada entre las marcas 7 y 8, por ser ésta una medida de ruido en el entorno.

Tabla 4. 1. Diferencia entre los datos reales y los datos calculados mediante el EKF SLAM Experimento1.

	POSICIÓN X DE LAS MARCAS EN EL ENTORNO REAL(cm)	POSICIÓN X DE LAS MARCAS EN EL MAPA GENERADO(cm)	ERRORES EN LAS MEDIDAS(cm)	POSICIÓN Y DE LAS MARCAS EN EL ENTORNO REAL(cm)	POSICIÓN Y DE LAS MARCAS EN EL MAPA GENERADO(cm)	ERRORES EN LAS MEDIDAS(cm
	Xreal	Xmed	Error X	Yreal	Ymed	Error Y
MARCA1	20	21	-1	158	162	-4
MARCA2	50	53	-3	185	188	-3
MARCA3	82	85	-3	185	186	-1
MARCA4	177	176	1	115	120	-5
MARCA5	176	181	-5	185	188	-3
MARCA6	225	222	3	115	112	3
MARCA7	250	251	-1	143	127	16
MARCA8	250	242	8	171	156	15
MARCA10	182	188	-6	275	282	-7
POS FINAL ROBOT TRAYECTORIA X				**POS FINAL ROBOT TRAYECTORIA Y**		
POS ROBOT	205	212	-7	259	253	6
ÁNGULO DE GIRO DEL ROBOT						
Θ real		Θ medido		Error Θ		
-20		**-12**		**-8**		

En la tabla anterior, se observa que la diferencia entre los datos de posición de las marcas del entorno real y los datos de posición de las marcas presentes en el mapa generado, en algunos casos, es muy pequeña (marcas 1, 2, 3 y 6), lo que resulta favorable en esta etapa después de la experimentación; pero en otros casos, esta diferencia es muy alta

(marcas 7 y 8). Esto, es evidencia de errores en la toma de medidas de los sensores, tanto de entorno como de desplazamiento o presencia de comportamientos indeseados del robot durante el experimento. Para generalizar el comportamiento de la aplicación, se realizaron dos experimentos más, de los cuales se mostrarán los resultados luego de analizar el por qué de los resultados del experimento 1.

4.1.1.1 Análisis.

Según la tabla 4.1 donde se muestra la diferencia entre el entorno real y el mapa generado en el Experimento 1, se observa que las primeras 3 marcas mapeadas durante la primera sección de la trayectoria presentan niveles de error muy pequeños, definiendo error como la diferencia entre las posiciones reales de las marcas y las posiciones de las marcas mapeadas. Este nivel de error bajo (de 1 a 4 cm) se debe a que, al realizar SLAM por primera vez, no se presentan errores de odometría por lo que las mediciones de las posiciones de las marcas resultan más cercanas a la realidad. Sin embargo, el error en este punto existe, pero se debe a ruidos que contaminan el sistema de visión, presentes por una iluminación deficiente que genera sombras, las cuales alteran las mediciones de distancia relativas al robot.

Cuando el robot realizó un avance de 30 cm aproximadamente, se introdujeron los primeros errores en las mediciones debidos a los errores introducidos por los sensores de odometría. De esta manera, la posición de la marca re-observada en este punto, fue actualizada en una posición cercana a su posición detectada en el paso anterior pero, debido al umbral de 3 cm establecido para determinar si una marca es repetida o nueva, esta marca pudo ser clasificada como una que ya se encontraba en el vector de estados del sistema, por lo que no se agregó a este vector; en cambio, a partir de ésta, se actualizó la posición y orientación del robot en ese punto. Cabe recordar que a los datos entregados por estos sensores, se les realizó un ajuste para compensar algunos de los desfases en el desplazamiento del robot debidos a cuestiones mecánicas. En magnitud de desplazamiento, el error se logró reducir hasta 3 cm aproximadamente; pero el error debido a la inercia del giro del robot, no logró ser reducido en igual medida vía software. Esto fue causado por el comportamiento heterogéneo del robot en el proceso de giro.

Cuando el robot realizó el primer giro para tomar el segundo tramo del entorno, se presentó un alto error odométrico debido al comportamiento impredecible del robot durante el giro. Por esta razón, el error en las mediciones de las marcas detectadas en este punto (marcas 4 y 5), aumentó de un máximo de 3cm, presente en las marcas 1, 2 y 3, a un máximo de 5cm. A medida que el robot avanzó en este sentido, el error odométrico aumentó, por ser éste de carácter acumulativo.

Luego de girar, se le ordenó al robot moverse hacia adelante aproximadamente 40 cm. En este punto, el robot observó la marca 6 en la cual se redujo el error en la medida de 3 cm. Esto se produjo porque al mover el robot hacia adelante, se compensó en cierto grado el error generado durante el giro pues, hay que recordar, que en el avance el robot presenta un error de 6 grados aproximadamente por cada metro que avance.

En los dos siguientes avances ordenados al robot, este observó las marcas 7 y 8 en las cuales aumentó su error de medida a un máximo de 15 cm. Esto fue debido al ruido introducido en la odometría del robot, al modificar de manera manual aproximadamente 2 grados la orientación del robot, para evitar que chocara con las marcas del entorno en sus primeros desplazamientos.

Luego se ordenó al robot desplazarse hasta el final de esta sección del entorno, en este punto, se introdujo la medida de una marca indeseada por las condiciones de iluminación deficientes presentes en el entorno. Por esta razón, la marca 9 del vector de estados mostrado en la interfaz del sistema, no fue comparada con ninguna marca presente en el entorno real.

Una vez el robot fue ubicado en el final de la segunda sección del entorno; se ordenó girar para mapear el último tramo del laberinto. En este punto, el robot observó las dos últimas marcas (marcas 9 y 10). Debido a que la interfaz de trabajo sólo muestra las primeras 10 posiciones de las marcas almacenadas en el vector de estados, sólo se pudo comparar la marca 10 con su equivalente en el entorno real. El resultado de esta comparación fue de un máximo de 7cm, que resultó ser bajo teniendo en cuenta que el error de odometría es acumulativo y que en la segunda sección se adicionó ruido al cambiar la orientación del robot manualmente.

En cuanto a la localización del robot, se comparó la posición final del robot en el entorno real, con la posición indicada por la aplicación y se observó que el mayor error se presentó en la orientación del robot que, aunque fue corregida en varias ocasiones mediante las marcas re-observadas, el resultado final fue de 8 grados. Esto debido a la manipulación que se tuvo del robot en el segundo tramo del entorno. En cuanto a posición X, Y del robot, se observó un error máximo de 7 cm que reflejan los errores en su odometría, a pesar de que la aplicación realiza correcciones en estas medidas.

Los resultados de este experimento, aunque fueron buenos en términos generales con un error máximo de 16 cm, no fueron los mejores debido a la manipulación que se le realizó al robot para evitar el choque con el entorno, lo que generó el aumento en el error en las medidas siguientes a este hecho. Lo ideal sería obtener errores de menos de 5cm para conseguir un mapeado muy cercano a la realidad, sin embargo, por las condiciones de manejo y de comportamiento del robot, en este experimento se logró estar cerca pero no lo suficiente del entorno global real.

4.1.2 Experimento 2.

En este experimento se siguió el mismo procedimiento de movimiento del robot que se siguió en el Experimento 1, pero tratando en lo posible de no introducir ruidos en la odometría debidos a la manipulación directa del robot. El mecanismo para determinar el error en los mapas generados es comparando las posiciones de las marcas medidas con las posiciones de las marcas en el entorno real como se realizó en el anterior experimento. La figura 4.22 y la tabla 4.2 muestran los resultados obtenidos en el segundo experimento.

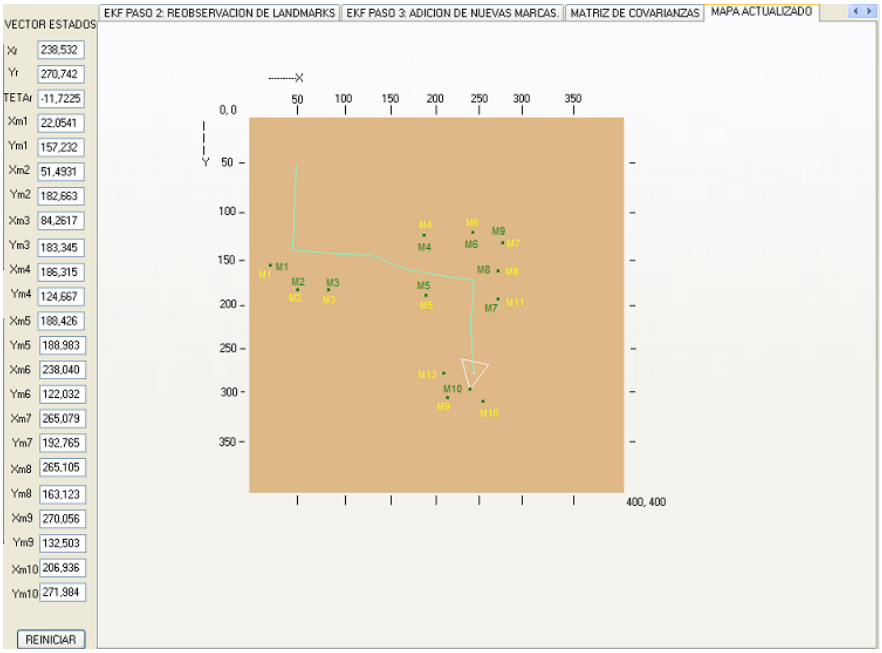

Figura 4 22. Mapa generado en el Experimento 2.

En la figura anterior, las etiquetas amarillas representan las marcas del entorno real de las cuales se muestra su ubicación en la figura 4.10, mientras que las etiquetas y puntos verdes representan el orden en el vector de estados y las marcas mapeadas por la aplicación SLAM. La tabla 4.2 muestra los resultados obtenidos luego del segundo experimento.

Tabla 4. 2. Diferencia entre los datos reales y los datos calculados mediante el EKF SLAM Experimento2.

	POSICIÓN X DE LAS MARCAS EN EL ENTORNO REAL(cm)	POSICIÓN X DE LAS MARCAS EN EL MAPA GENERADO(cm)	ERRORES EN LAS MEDIDAS(cm)	POSICIÓN Y DE LAS MARCAS EN EL ENTORNO REAL(cm)	POSICIÓN Y DE LAS MARCAS EN EL MAPA GENERADO(cm)	ERRORES EN LAS MEDIDAS(cm)
	Xreal	Xmed	Error X	Yreal	Ymed	Error Y
MARCA1	20	22	-2	158	157	1
MARCA2	50	51	-1	185	183	2
MARCA3	82	84	-2	185	183	2
MARCA4	177	186	-9	115	125	-10
MARCA5	176	188	-12	185	189	-4
MARCA6	225	238	-13	115	122	-7
MARCA7	250	265	-15	201	193	8
MARCA8	250	265	-15	171	163	8
MARCA9	250	270	-20	143	132	11
MARCA10	182	207	-25	275	271	4
POS ROBOT TRAYECTORIA X				**POS ROBOT TRAYECTORIA Y**		

POS ROBOT	200	220	-20	275	266	9
ÁNGULO DE GIRO DEL ROBOT						
Θ real		**Θ medido**		Error Θ		
-11		**-5**		**-6**		

4.1.2.1 Análisis.

Según la tabla 4.2, se observa que el error presente en las mediciones de las marcas 1, 2 y 3, al igual que en el primer experimento es muy bajo, de 2cm máximo. Esto debido a la razón expuesta en el análisis de estas tres marcas en el experimento anterior. Cuando el robot realizó el primer giro detecto las marcas 4, 5, en las cuales el error aumentó a un máximo de 12 cm, mayor al presentado en estas mismas marcas en el experimento 1. Este aumento en el error de las medidas de estas marcas fue debido a un error presente en la odometría del robot al realizar el giro, esta inconsistencia se presentó porque al girar, las ruedas del robot patinaron más de lo normal por lo tanto la odometría del robot indicó que este, además de haber realizado un giro, había tenido un avance de 10 cm aproximadamente, cuando en realidad lo único que había hecho era girar sobre su eje. Este inconveniente se observa en la figura 4.23 encerrado en el círculo 1.

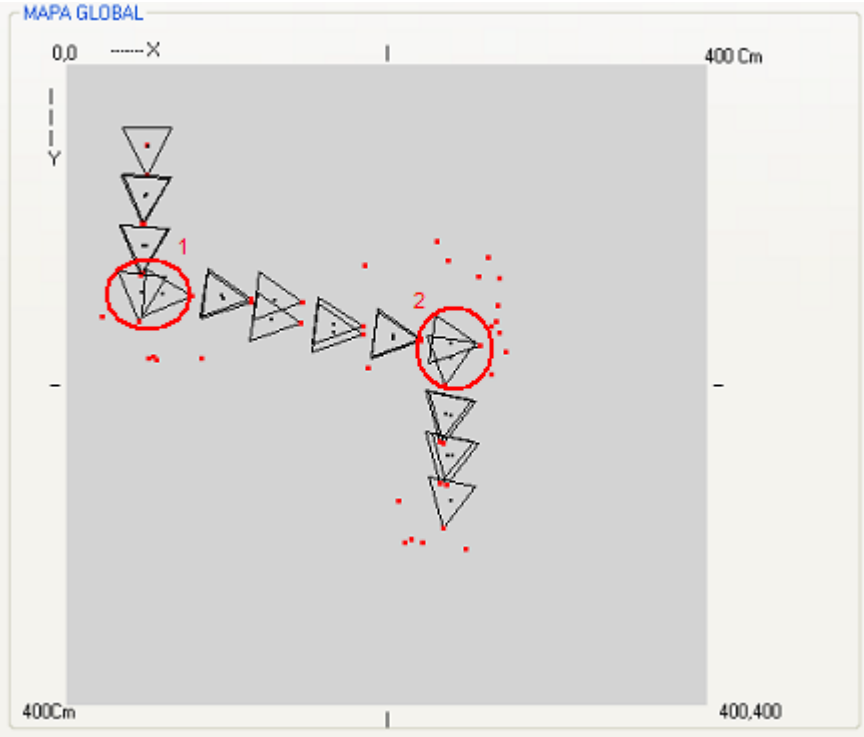

Figura 4 23. Error en la medida de giro entregada por los sensores de odometría.

.

De aquí en adelante todos los errores de observación de las siguientes marcas fueron en aumento, debido esta inconsistencia y los errores acumulativos presentes en cada actualización de la odometría. Este es el caso de las marcas 6, 7, 8 y 9, en las cuales aumentó su error de observación a un máximo de 20 cm justamente en el eje x donde la odometría entregó una medida de desplazamiento que nunca existió en el proceso de giro.

El error en la medida de la marca 10 siguió en aumento, logrando el más alto índice, 25 cm máximo, esto debido a que se sumó al error acumulado anteriormente, el mismo problema presentado durante el primer giro del robot. Cuando el robot giró por segunda vez para tomar el último tramo del entorno, aparte del giro, la odometría del robot indicó un desplazamiento en el eje Y que nunca existió. El circulo con el numero 2 de la figura 4.23 indica esta inconsistencia.

En cuanto a la localización del robot, se presentó un error máximo de 20 cm en el eje X y de 9 cm en el eje Y, debido a los inconvenientes de odometría presentados en la figura 4.23, en la cual se observa que este efecto fue más fuerte en el primer giro realizado por el robot, por lo que se afectó en mayor medida la medida de odometría en el eje X

entregada por el robot. En cuanto a orientación el error no fue tan alto, dado que el robot no fue manipulado durante el proceso de SLAM, sin embargo se presentó un error de 6 grados debido a cuestiones mecánicas del mismo.

Los resultados de este experimento arrojaron errores más altos que los presentes en el experimento 1, pero la razón por la cual ocurrió este incremento fue por el comportamiento indeseado del robot durante los giros al realizar este segundo experimento.

4.1.3 Experimento 3.

El mapeado del entorno y localización del robot en el experimento 3 se llevó a cabo de la misma manera que en los anteriores experimentos. Los resultados obtenidos se muestran en la figura 4.24 y en la tabla 4.3.

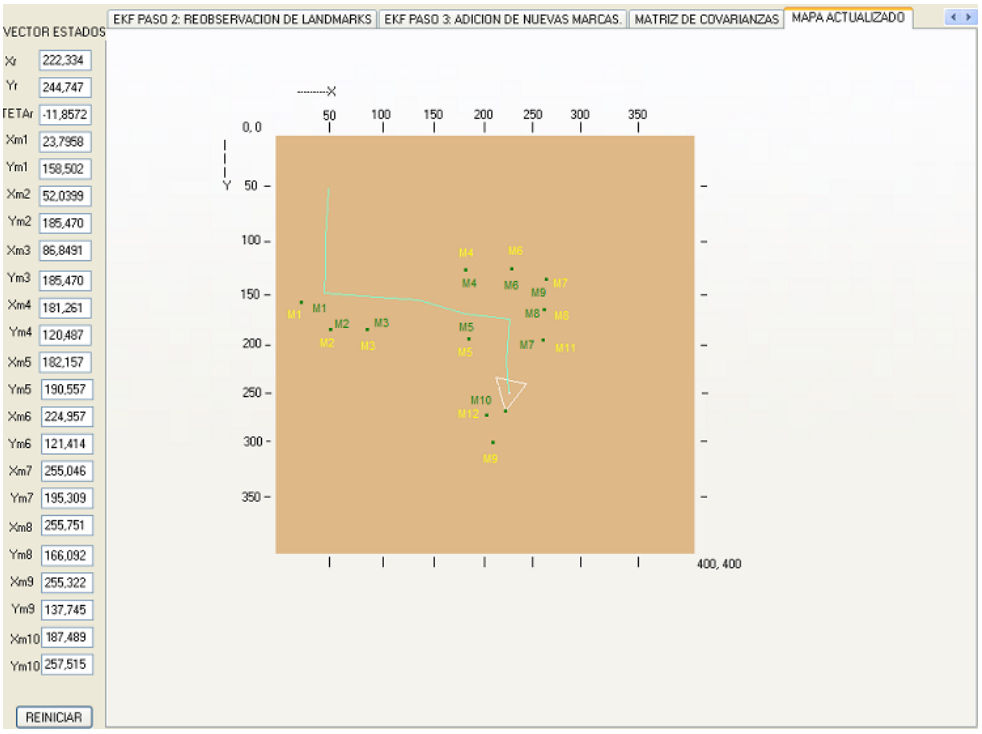

Figura 4 24. Mapa generado en el Experimento 3.

La interpretación de la figura 4.24, es la misma que en la figura 4.22. La diferencia entre los datos obtenidos en este experimento y los datos del entorno real se muestra en la tabla 4.3.

Tabla 4. 3. Diferencia entre los datos reales y los datos calculados mediante el EKF SLAM Experimento3.

	POSICIÓN X DE LAS MARCAS EN EL ENTORNO REAL	POSICIÓN X DE LAS MARCAS EN EL MAPA GENERADO	ERRORES EN LAS MEDIDAS	POSICIÓN Y DE LAS MARCAS EN EL ENTORNO REAL	POSICIÓN Y DE LAS MARCAS EN EL MAPA GENERADO	ERRORES EN LAS MEDIDAS
	Xreal	Xmed	Error X	Yreal	Ymed	Error Y
MARCA1	20	24	-4	158	158	0
MARCA2	50	52	-2	185	185	0
MARCA3	82	87	-5	185	185	0
MARCA4	177	181	-4	115	120	-5
MARCA5	176	182	-6	185	190	-5
MARCA6	225	225	0	115	121	-6
MARCA7	250	255	-5	201	195	6
MARCA8	250	256	-6	171	166	5
MARCA9	250	255	-5	143	137	6
MARCA10	180	187	-7	250	257	-7
POS ROBOT TRAYECTORIA X				**POS ROBOT TRAYECTORIA Y**		
POS ROBOT	214	222	-8	252	245	7
ÁNGULO DE GIRO DEL ROBOT						
Θ real		Θ medido		Error Θ		
-10		-5		-5		

4.1.3.1 Análisis.

Según la tabla 4.3, el tercer experimento arrojó mejores resultados que los dos primeros. Se observa que en las marcas 1, 2 y 3, el error de medida es cero, esto no es del todo cierto porque las medidas tomadas del entorno real están condicionadas al error del ojo humano, pues estas medidas fueron tomadas con el laberinto montado y de esta manera pueden existir errores de apreciación que aunque no superan 1 cm, es necesario mencionarlos.

Los errores en las medidas de las marcas 4 y 5 se mantuvieron bajos (6 cm máximo), debido a que en esta ocasión se instaló el entorno en otro lugar donde fueron mejoradas las condiciones del piso para garantizar el buen funcionamiento del robot y su odometría

durante los giros. De igual forma los errores en las marcas 6, 7, 8, 9, se mantuvieron bajos con un nivel de error máximo de 6 cm, debido a que en este experimento no se presentó la contaminación en los datos de odometría al realizar el giro como si ocurrió en el experimento 2. En la observación de la marca 10 se presentó el mayor índice de error en la medición de las marcas debido a los pequeños errores acumulados de odometría y del sistema de visión, que aunque el objetivo del algoritmo SLAM EKF es reducirlo al máximo solo pudo ser atenuado hasta un mínimo de 7cm.

En cuanto a la localización del robot, el error máximo se presentó en el desplazamiento del robot en el eje X y este fue de 8 cm, y en la orientación fue de 5 grados. El error generado en la medida de localización del robot siempre va a prevalecer, debido a que el error debido a la odometría es acumulativo, es decir, en cada medición de odometría se agregan nuevos errores a las mediciones acumuladas que también contienen sus propios errores.

4.2 Conclusiones, Aportes y Trabajos Futuros.

Teniendo en cuenta el objetivo principal de este trabajo de grado que consistía en desarrollar un sistema de localización y mapeo simultáneos utilizando un sistema de visión, y luego de avanzar por este proceso, se puede concluir en primer lugar que el SLAM es un campo demasiado amplio, donde se pueden encontrar una gran variedad de enfoques para abordar este problema. Pero, la mayoría de estos enfoques lo abordan desde el punto de vista de simulación, por lo que el desarrollo de la aplicación llevada a cabo en éste trabajo de grado, aporta en gran medida corroborando en la práctica real, el funcionamiento de uno de éstos algoritmos teóricos, en este caso el Filtro Extendido de Kalman (EKF), desarrollado para dar solución al problema de mapeado y localización simultáneos SLAM.

Las aplicaciones estudiadas en el transcurso de este trabajo, utilizan como sistemas de captura de información del entorno, sensores tales como ultrasonido o *scanners laser* que requieren de un bajo nivel de procesamiento para la extracción de características del mismo; de otra parte, para cumplir con ésta función, deben ser instalados varios sobre el robot; por lo que los costos financieros aumentan. Otra opción es la de utilizar sensores

más avanzados como sistemas de visión estereoscópicos los cuales incluyen arreglos de dos o más cámaras. Este tipo de sensores requieren de altos niveles de procesamiento para realizar la extracción de características del entorno, además, son difíciles de adquirir debido a su elevado costo. En este sentido la aplicación desarrollada en este trabajo de grado, logra reconstruir un ambiente y localizar un robot móvil dentro de éste, utilizando como sistema de visión una cámara web configurada a baja resolución para extraer características del entorno y, como sensores de desplazamiento, se utilizó *encoders* incrementales instalados en el robot de trabajo ATIBOT. Resaltando, que este tipo de sensores requieren de costos computacionales y financieros bajos, lo cual da pie para avanzar en el estudio de esta temática dentro de la Universidad.

Las mediciones de las características del entorno, pueden ser fácilmente tomadas mediante un sistema de visión relativamente simple, para este caso una cámara web, debido al tipo de ambiente definido donde se desarrolló el proceso de SLAM. El entorno de trabajo es definido binario, es decir, las paredes del entorno son blancas y las marcas de color negro; esto, con el fin de que el procesamiento necesario para extraer las características del entorno no sea demasiado complejo. Bajo estas condiciones, el sistema de visión arrojó excelentes resultados demostrando que, con un sistema de visión basado en procesamiento de imágenes no tan complejo, se pueden extraer de manera eficiente, las medidas de distancia necesarias para llevar a cabo el proceso de SLAM.

Aunque el proceso de SLAM EKF funciona asumiendo la existencia de ruido que contamina al sistema, es necesario garantizar que el error introducido a través de los sensores, tanto de odometría como estereoceptivos, esté dentro de niveles tolerables para garantizar una representación fiel del entorno sobre el cual se está desarrollando el proceso de Localización y Mapeo Simultáneos.

Dado que la aplicación, generada en este trabajo de grado fue desarrollada e implementada siguiendo la metodología de diseño orientado a objetos, la cual la hace portable y escalable, y que la estructura del Filtro Extendido de Kalman utilizado, está conformada por secciones independientes unas de otras, es posible realizar mejoras futuras a la aplicación con gran facilidad o incluso, tomarla como base para desarrollos futuros.

Luego del proceso de experimentación, se observa que el filtro extendido de Kalman se comporta como se esperaba, dando una mejor estimación del mapa cuando el error existente en las medidas de las marcas y de la odometría disminuye. También, se observa que al disminuir el error odométrico, los valores calculados de la constante de Kalman disminuyen, debido a que al presentarse este hecho, debe hacerse menos corrección en el estado actual del robot. La cámara web utilizada como sistema de visión, aporta la información suficiente para reconstruir el mapa del entorno establecido. Para hacer esto posible, la posición y orientación del robot deben tener un nivel de error bajo, a pesar de que el filtro extendido de Kalman corrige los errores de odometría. Esta fue una de las principales razones para no conseguir mejores resultados.

Queda abierta la posibilidad de trabajos futuros en esta rama, utilizando sistemas de visión más avanzados, como por ejemplo trabajar con sistemas de visión estéreo o incluso fusionar diferentes tipos de sensores para realizar el proceso de extracción de características del entorno.

A menor escala, se podría modificar el algoritmo para que permitiera la lectura de información de una segunda cámara web, con el fin de ampliar el ángulo de observación y de ésta manera poder capturar más información del entorno en cada punto de muestreo. Además, se podría trabajar en la aplicación de técnicas más avanzadas de procesamiento de imágenes como utilizando técnicas de detección de bordes y esquinas para la construcción de mapas más elaborados.

Por último se podría trabajar en la utilización de estos mapas generados mediante técnicas SLAM para labores de navegación de robots móviles en entornos desconocidos.

BIBLIOGRAFIA

[1] GIL, A y REINOSO, O. “Influencia de los Parámetros de un filtro de partículas en la solución al problema SLAM”. En: IEEE latin América Transactions, vol. 6, No. 1. Alicante, 2008.

[2] JAQUEZ, V. “construcción de mapas y localización simultánea con robots móviles”. Master’s thesis, ITESM Campus Morelos, México, 2005.

[3] RODRIGUEZ, DIEGO Y GONZÁLEZ, LOZADA. “SLAM geométrico en tiempo real para robots móviles en interiores basado en el EKF”. Universidad Politécnica de Madrid. Departamento de Automática, Ingeniería Electrónica e Informática Industrial, Escuela Técnica Superior de Ingenieros Industriales. Madrid, 2004.

[4] GALLARDO LÓPEZ, DOMINGO.”Aplicación del muestreo bayesiano en Robots Móviles: Estrategias para la localización y estimación de mapas del entorno”. Universidad de Alicante. Departamento de Tecnología Informática y Computación. Alicante, 1999.

[5] SOLERA RAMIREZ, ÁLVARO. “El Filtro de Kalman”. Nota Económica. Banco Central de Costa Rica. División Económica. Departamento de Investigaciones Económicas. Costa Rica, 2003.

[6] BARGUEÑO RAIGAL, EVA. “Aplicación del Filtro de Partículas al Seguimiento de Objetos en Secuencias de Imágenes”. Universidad Rey Juan Carlos. Ingeniería Técnica en Informática de Sistemas. 2002-2003.

[7] PANTRIGO, JUAN JOSE; DUARTE ABRAHAM y SÁNCHEZ ÁNGEL. “Hibridación entre El Filtro de Partículas y búsqueda Dispersa: aplicación al TSP Dinámico”. Universidad Rey Juan Carlos. ESCET. Departamento de Informática, Estadística y Telemática. Madrid, 2001.

[8] GONZALEZ VARGAS, NÉSTOR ANDRÉS. “Sistema de Visión por Computadora para la Medición de Distancia e Inclinación de Obstáculos para Robots Móviles”. Pontificia Universidad Javeriana. ISSN 0123-2126. Vol. 9, No.2. Bogotá, 2005.

[9] ZAMORA, M.A, TÓMAS BALIBREA L.M y MARTÍNEZ, H, SHARMETA, A,G. “Navegación Planificada de un Robot Móvil en Entornos de Interiores desconocidos”. Universidad de Murcia. Facultad de Informática. Murcia.

[10] MONTEMERLO, MICHAEL. “FastSLAM: A Factores Solution to the Simultaneos Localization and Mapping Problem UIT Unknown Data Association”. Carnegie Mellon University. Pittsburg, 2003.

[11] THRUN, SEBASTIAN; MONTEMERLO, MICHAEL; KOLLER, DAPHNE; WEGBREIT, BEN; NIETO, JUAN and NEBOT EDUARDO. “FastSLAM: A Factores Solution to the Simultaneos Localization and Mapping Problem UIT Unknown Data Association”. University Stanford, University of Sydney.

[12] CASTILLO, RICARDO ANDRES; DÍAZ LÓPEZ, IVÁN ANDRÉS y HUERTAS LEÓN MAURICIO. “Localización Espacial de un Punto en XYZ mediante Visión Artificial”. Universidad Militar Nueva Granada. Vol. 16, No.1. 2006.

[13] BARGUEÑO RAIGAL, EVA. “Aplicación del Filtro de Partículas al Seguimiento de Objetos en Secuencias de Imágenes”. Universidad Rey Juan Carlos. Ingeniería Técnica en Informática de Sistemas. 2002-2003.

[14] OIZA NAVARRO, IÑAKI. “Localización de un Robot Móvil con el Filtro Extendido de Kalman”. 14 de Julio de 2005.

[15] MARTÍNEZ GÓMEZ, LUIS ALFREDO. “Sistema de Visión para el equipo de Robots Autónomos del ITAM”. Instituto Tecnológico Autónomo de México. México D.F. 2004.

[16] ESCALERA HUESO, ARTURO. “Visión por Computador, Fundamentos y Métodos”. Pearson Educación. S.A. Madrid, 2001.

[17] ESCALA DE GRISES-WIKIPEDIA. La Enciclopedia Libre. Disponible en: http://es.wikipedia.org/wiki/Escala_de_grises. Fecha de actualización: 22 de febrero de 2008. Fecha de acceso: 11 de marzo de 2008.

[18] CAMPUS VIRTUAL, Enseñanza Virtual y Laboratorios Tecnológicos.”Tratamiento Digital de la Imagen, El Histograma de una imagen”. Disponible en: http://campusvirtual.uma.es/tdi/www_netscape/TEMAS/Tdi_30/. Fecha de actualización: 8 de marzo de 2008. Fecha de acceso: 4 de marzo de 2008.

[19] VIDAL CALLEJA, TERESA A. “Visual Navigation in Unknown Environments ”. Universitat Politécnica de Catalunya. Institut de Robótica i Informática Industrial. Barcelona, 2007.

[20] HERNANDEZ HOYOS, MARCELA. “Procesamiento y análisis de Imágenes Digitales(PAID)”. Compugráfica. 2004.

[21] GRUPO DE TECNOLOGIA INDUSTRIAL. “Características de una Imagen”. Universidad Miguel Hernández. Visión por Computador. División de Ingeniería de Sistemas y Automática.

[22] CASTRO PEREZ, DAVID y DOPICO RODRIGUEZ, PATRICIA. “Segmentación, Umbralización, Regiones y Clustering”. Universidad Carlos III. Madrid, 2004.

[23] MARTIN, MARCOS. “Técnicas Clásicas de Segmentación de Imagen”. Mayo, 2002.

[24] RIISGAARD, SOREN and RUFUS BLAS, MORTEN. “Slam for Dummies (a tutorial Approach to Simultaneous Localization and Mapping)”.

[25] SOTOMAYOR BASILIO, BORJA y EXTREMO BALGORRI, UNAI. “La Plataforma .Net: ¿el futuro de la Web?”. Disponible en: http://people.cs.uchicago.edu/~borja/pubs/revistaeside2002.pdf.

[26]”El Rincón en Español de C#”. Disponible en: http://www.programacion.net/votar/id=537&obj=enlace/

[27] GUTIERREZ MAYORAL, ANTONIO. “Lenguaje de Programación C#”. Disponible en: http://gsyc.es/~agutierr/pfc-tecnica-html/node20.html

[28] GONZALES, RAFAEL C and WOODS, RICHARD E. “Tratamiento Digital de Imágenes”. Addison-Wesley. Madrid, 1996.

[29] BAÑÓ AZCÓN, ALBERTO. “Análisis y Diseño del Control de posición de un Robot Móvil con Traccion Diferencial”. Universitat Rovira i Virgili. Departament d’ Enginyeria Electrónica i Automàtica. 2003.

[30] ARRAS, KIA O. “The CAS Robot Navigation Toolbox”. Versión 1. 2004.

[31] ALVIRA, CRISTIAN D; SEGURA, JUAN MAUEL; VELASCO, OSCAR; FLORES, JUAN F.”Atibot: UGV SKID STEERING de Arquitectura Distribuida”. Universidad del Cauca. Grupo de Investigación ATI. Popayá, 2007.

[32] ZUNINO, GUIDO. “Simultáneous Localization and Mapping for Navigation in Realistic Environments”. Universitet Stockholms. Licentiate Thesis, Royal Institute of Technology. Numerical Analysis and Computer Sciencie. Stockholm, 2002.

[33] ELTRA. “Encoder Incremental Descripción General”. SILGE ELECTRONICA S.A. Buenos Aires, Argentina. 2000.