Universidad Politécnica de Puebla

Tesis de Maestría

EXPLORACIÓN CON UN ROBOT AÉREO PARA IDENTIFICACIÓN DE OBJETOS.

Autor:
Luis Alejandro Zetina De la Cruz

Director de Tesis:
Dr. Antonio Benítez Ruiz

Esta tesis se presenta como un requisito
para obtener el grado de Maestría en Ingeniería

en

Sistemas y computo inteligente
Departamento de Posgrado

Juan C. Bonilla, Puebla. *19 de octubre de 2015*

Agradecimientos

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Planteamiento del problema de investigación

1.1. Introducción

La ciencia de la computación es aún joven en comparación con otras ciencias, pero en gran auge debido a las variadas áreas en que se puede aplicar. Cuando se construyeron las primeras computadoras, tenían asociadas tareas específicas de cálculos complejos y se buscó que estas operaciones se calcularan con más rapidez y precisión. La ciencia de la computación tiene un amplio campo de aplicación como son: las redes de computadoras; las bases de datos; la graficación por computadora; el análisis de algoritmos; la inteligencia artificial; la teoría de la computación; la computación concurrente paralela y distribuida; la computación científica, por mencionar algunas.

Una de las áreas de mayor desarrollo de la ciencia de la computación es la Inteligencia Artificial (IA), que se especializa en desarrollar sistemas y máquinas con cierto grado de inteligencia. Alan Turing [1] fue uno de los pioneros en esta área al publicar su artículo "*Computing machinery and intelligence*" donde plantea la posibilidad de que una máquina simule el comportamiento humano. Algunas de las disciplinas que abarca la IA son: la representación del conocimiento, el razonamiento automático, el aprendizaje automático, la visión computacional y robótica, entre otras.

En la actualidad existe un importante número de aplicaciones comerciales y científicas relacionadas con el aprendizaje automático. El aprendizaje automático busca desarrollar técnicas que brinden a las computadoras la capacidad de aprender; como podría ser un automóvil autónomo que sea capaz de transportarse por una ciudad sin requerir un conductor. Esta disciplina de la inteligencia artificial según Mitchell [2], se refiere a la interrogante de ¿cómo construir programas de computadora que mejoren con la experiencia?, se relaciona con varias áreas como la estadística, la filosofía, la biología, la teoría de la información, por mencionar algunas. El aprendizaje automático utiliza algoritmos de aprendizaje para dotar de inteligencia a las computadoras.

Otra de las áreas de interés en la IA es la robótica, ésta propone técnicas y mecanismos para construir maquinas con cierto grado de inteligencia que puedan asistir a los seres humanos en la ejecución de alguna tarea. Así, un robot se puede definir como una entidad mecánica controlada por una computadora y programada para moverse, manipular objetos y realizar trabajos al interaccionar con su entorno [3].

La palabra robot fue utilizada por primera vez por Karel Čapek [4] en su obra de teatro R.U.R. (*Robots Universales Rossum*) derivada de la palabra checa robota que significa "*esclavo*" o "*trabajo*" y en la actualidad su campo de aplicación es muy variado, como la medicina, la industria, la vigilancia, incluso con fines militares. Su uso es muy importante en áreas donde la integridad humana estaría en peligro, como la exploración de volcanes, exploración del océano a grandes profundidades, manipulación de objetos explosivos o radioactivos, exploración de planetas y de otros cuerpos celestes, y en áreas de reciente aplicación como la vigilancia civil y mapeo de áreas. Para Abdala et al., [5], la clasificación de los robots con respecto a su arquitectura es: poli-articulados, móviles, androides, zoomórficos e híbridos. Dentro de los robots móviles se encuentran los terrestres, marítimos, espaciales y aéreos.

Barrientos [6], menciona que los robots aéreos UAV (del ingles, *Unmanned Aerial Vehicle: Vehículo Aéreo No Tripulado*) son aeronaves que no necesitan de un operador para realizar una actividad, también menciona que un criterio

de clasificación de éste tipo de robots es por el tipo de despegue: vertical y no vertical. Dentro de los primeros se tienen los de hélice (helicópteros y cuatrimotor) y los auto-sustentados (dirigibles y globos). En los de despegue no vertical se encuentran los de ala flexible (parapente y ala delta) y los de ala fija (aeroplanos). El tipo de aéronave más utilizado para el desarrollo de aplicaciones civiles es el cuatrimotor, que es un tipo de helicóptero que su arquitectura tiene cuatro rotores para realizar todos los posibles movimientos y desplazamiento, al realizar variaciones en la velocidad angular de cada rotor; la mayor utilización de este tipo de aéronave frente a otras arquitecturas, se debe principalmente a las ventajas que presenta, como las mencionadas por Hoffmann et al. [7], entre las que destacan; la mejor maniobrabilidad en espacios pequeños, que es muy limitada en aeronaves de ala fija, como por ejemplo, la inspección de edificios y puentes para la detección de grietas, la detección de municiones sin detonar, baliza de seguimiento de rescate. El cuatrimotor es un vehículo fácil de usar con bajos requerimientos de mantenimiento.

Una de las actividades más importantes en la robótica es la exploración de ambientes y la reconstrucción de mapas. Los mapas brindan una referencia de localización para un sistema de navegación de un robot. Por esto, es importante dotar a los robots con la capacidad de generar su propio mapa del ambiente donde se encuentran los objetos de interés. Entre los principales tipos de mapas están los mapas de celda, mapas topológicos y mapas híbridos. La exploración del ambiente para la generación de mapas se basa en sensores que le permitan al robot percibir el ambiente que explora como visuales, infrarrojos, ultrasónicos y de laser, por mencionar algunos [8]. Para este trabajo se utilizarán sensores visuales, debido a que se entrenarán algoritmos de reconocimiento de objetos en imágenes.

Por otro lado, una de las disciplinas de la IA con mayores retos es la de visión computacional; la complejidad radica en que, el proceso de la visión humana es un acto inconsciente y por esto resulta muy difícil la implementación computacional de esta actividad. La visión por computadora trata de describir el mundo a partir de un conjunto de imágenes al extraer y reconstruir sus propiedades como la distribución de colores, la forma y la iluminación [9]. Den-

tro de las actividades de la visión computacional se encuentra la detección automática de objetos, que en conjunto con la visión artificial le facilita a los robots interaccionar con el ambiente [10].

El reconocimiento de objetos es una actividad que combina la visión cumputacional y el aprendizaje automático. Ésta actividad utiliza algoritmos de aprendizaje automático que le brinda al sistema la capacidad de reconocer objetos de interés en imágenes. En el trabajo previo, Olmedo [11], se realizó el entrenamiento del detector Viola-Jones [12] en paralelo para el reconocimiento de tres objetos diferentes y fue implementado en un cuatrimotor AR.Drone [13], controlado por medio de un joystick para realizar el recorrido del ambiente. En este trabajo se aumentará el número de objetos reconocidos, así como también, se busca potencializar las funcionalidades del robot con la implementación de un algoritmo de navegación autónoma.

Este proyecto busca articular varias disciplinas de la IA, como el aprendizaje automático, la visión computacional, la exploración y reconstrucción de mapas en robótica con la finalidad de dotar a un cuatrimotor con la capacidad de explorar y construir un ambiente controlado a partir del reconocimiento de objetos de interés.

1.2. Objetivo General

Explorar de forma autónoma un ambiente controlado para la reconstrucción de mapas

1.3. Objetivos Específicos

- Diseñar una interfaz para el monitoreo de la información de los sensores y operación del cuatrimotor.
- Identificar cuatro objetos diferentes en tiempo real.
- Implementar un algoritmo de exploración autónoma para un cuatrimotor.

- Generar un mapa en 2D del ambiente explorado apoyado en un algoritmo de autolocalización identificando objetos de interés.

1.4. Justificación

El uso de los robots se va generalizando en más áreas, como su incursión en labores de rescate, en desastres humanos o naturales, iniciando en esta área en los atentados ocurridos el 11 de septiembre de 2001, donde se emplearon robots para la excavación y localización de personas. Otro acontecimiento donde demostraron su efectividad y rapidez de localización de personas, fue en el desastre provocado por el huracán Katrina en Nueva Orleans en 2005, donde robots aéreos recorrían grandes extensiones en busca de personas en las azoteas, incluso más rápido que el personal de rescate. Un ejemplo más es el robot aéreo para detectar incendios realizado por Noelia Hernández et al. [14], la vigilancia de fronteras o de cualquier otro objetivo, son ejemplos de otras aplicaciones que pueden tener los robots y que han demostrado ser muy eficientes.

Así, este proyecto busca incrementar la funcionalidad del robot con exploración autónoma de un ambiente controlado y generación de mapa en 2D.

1.5. Contribuciones esperadas

- Desarrollo de una interfaz gráfica para el monitoreo de los sensores y localización del cuatrimotor.
- Implementación de rutinas de exploración autónoma en el cuatrimotor.
- Lograr reconocer cuatro objetos diferentes en tiempo real.
- Reconstrucción del mapa en 2D del ambiente explorado para ubicar los objetos reconocidos.
- Aproximación de la localización de objetos de interés detectados en un ambiente controlado a través de un proceso de muestreo.

Capítulo 2

Robótica aérea y construcción de mapas

La investigación y desarrollo de robótica aérea se ha incrementado debido al interés en su amplio campo de aplicación, que incluye la milicia, aplicaciones civiles como vigilancia y búsqueda y rescate, y de investigación como la exploración y mapeo. Como ya se mencionó dentro de las arquitecturas de robots aéreos existentes se utilizará la arquitectura tipo cuatrimotor.

2.1. Robot tipo cuatrimotor

El cuatrimotor es la arquitectura más utilizada en desarrollos civiles, por su gran versatilidad de vuelo, su capacidad de vuelo estacionario y la relativa facilidad de control de sus rotores en comparación con otras arquitecturas.

- Ventajas.
  - Se puede acercar a objetos con baja posibilidad de colisionar.
  - Puede maniobrar en las tres dimensiones [15], es decir puede realizar rotaciones y desplazamientos en cualquiera de los ejes.
  - Mayor estabilidad.
- Desvantajas.
  Entre las principales desventajas que menciona Barrientos [6], se encuentran:

- Velocidad de desplazamiento menor en comparación con otras arquitecturas.
- Menor autonomía de vuelo con respecto al tiempo de duración de las baterías.
- Poca resistencia a perturbaciones de viento.
- Capacidad de carga limitada.
- Techo de vuelo limitado.

2.2. AR.Drone

El AR.Drone es un cuatrimotor de radiocontrol desarrollado por la empresa Francesa Parrot y fue presentado en la feria International Consumer Electronics Show (*CES*) en Las Vegas en 2010 y la nueva versión fue presentada en la misma del 2012 con buenas expectativas de los visitantes, como expresa la revista Muy interesante en su versión en línea [16]; éste cuenta con un microprocesador y un conjunto de sensores, al igual que un conector wi-fi para la comunicación con dispositivos móviles con sistemas operativos IOS, Android o Linux, que le permitan controlarlo y recibir los datos de lectura de los sensores del AR.Drone.

Las primeras versiones estaban disponibles para ser controlados con sistema IOS, pero la empresa liberó los applet de control bajo código abierto, lo que permitió el desarrollo de aplicaciones para diversos dispositivos.

2.2.1. Ventajas y desventajas

Ventajas

- La navegación del AR.Drone se realiza manipulando las velocidades que actúan sobre los ejes de referencia.
- Es posible la recuperación de las lecturas de los sensores con los que cuenta el cuatrimotor.
- El cuatrimotor cuenta con el control de vuelo ya programados.

Desventajas

- El control del AR.Drone no puede ser modificado.
- Presenta inestabilidad de vuelo siendo mas evidente en vuelos a baja velocidad.
- La carga de las baterías es de poca duración y afecta la estabilidad de vuelo.

2.2.2. Características de hardware

La estructura del AR.Drone permite un cambio muy sencillo de sus piezas e incluso el ensamble total de un equipo sin mayor problema. Su estructura principal es en forma de cruz de tubos de fibra de carbono donde se montan los motores, la placa madre y la placa de navegación; la mayor parte de su cuerpo es de espuma de polipropileno que le brinda mayor protección contra golpes. Cuenta con un casco de protección hecha de plástico para exteriores y otra para interiores que protegen las hélices como se muestra en la Figura 2.1.

(a) (b)

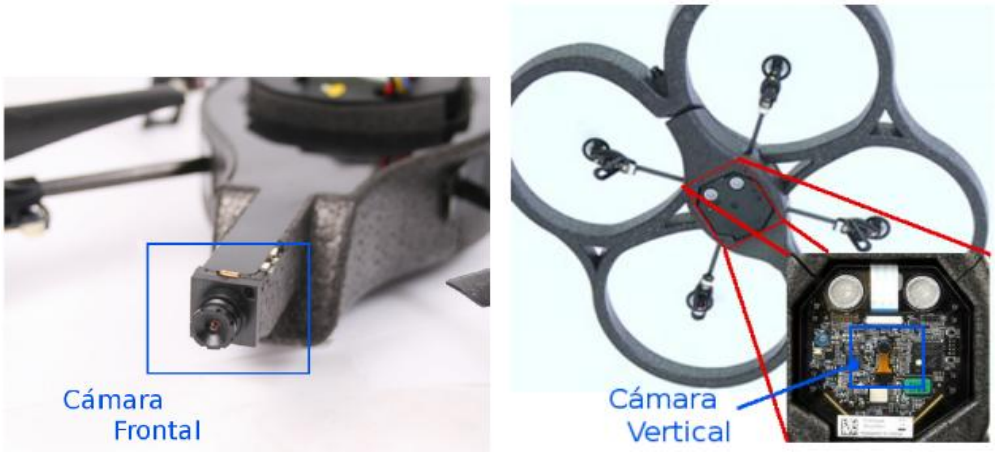
Figura 2.1: AR.Drone con carcasa de interiores (a) y carcasa de exteriores (b).

En la Tabla 2.1 se muestran las especificaciones con que cuenta el AR.Drone. En la Figura 2.2 se destacan las dos cámaras que tiene integradas, una frontal

que envía la imagen de navegación al dispositivo resolución de 640 x 480 pixeles y una vertical de 176 x 144 pixeles de resolución, que utiliza para calcular la velocidad de desplazamiento.

Tabla 2.1: Especificaciones técnicas del AR.Drone 2.0.

Procesador	ARM Cortex A8 de 32 bits a 1 GHz	RAM DDR2 de 1 Gbit a 200 MHz
Conexiones	Wi-Fi b/g/n	USB 2.0 de alta velocidad
Sensores	Acelerómetro de 3 ejes	Giroscopio de 3 ejes
	Sensor de presión	Sensores de ultrasonido para medición de altitud
	Magnetómetro de 3 ejes con precisión de 6°	
Cámaras	Frontal de lente angular	Vertical de lente angular de 93°
	Resolución 1280 x 720 pixeles	Resolucion 176 x 144 pixeles
	30fps	60fps
Sistema Operativo	Linux	



(a) (b)

Figura 2.2: Cámara frontal (a) y vertical (b) del AR.Drone [17].

La velocidad máxima de recorrido es de 18 km/h con un alcance y techo máximo de vuelo limitado por el alcance de la conexión Wi-Fi y una autonomía de vuelo promedio de 12 minutos con la pila estándar y con tiempo de carga de 90 minutos. Tiene integrado un microprocesador ARM9 RISC de 32 bits a 469

MHz, una memoria DDR SDRAM de 128 MB a 200 MHz, sistema operativo con núcleo Linux, modem Wi-Fi y conector USB. Cuenta con 4 motores funcionando a 3,500 rpm con una potencia de 15 w y con un altímetro ultrasónico con alcance de 6 metros para la estabilización vertical.

2.2.3. Características del software

La empresa Parrot desarrolló una interfaz para móviles para manipular el cuatrimotor AR.Drone (véase Figura 2.3) con fines de entretenimiento con opción de juegos de realidad aumentada. Entre otras características cabe resaltar las siguientes más innovadoras:

- Controles táctiles e intuitiva de inclinación de vuelo
- Transmisión de vídeo en directo y sesión de fotos
- Etiquetas integradas de detección para juegos de realidad aumentada

Se encuentra disponible un **kit de desarrollo de software** o **SDK**(del inglés, *Software Development Kit*) de código abierto que permite a cualquier desarrollador de AR.Drone desarrollar y distribuir nuevos juegos basados en el AR.Drone utilizando el wi-fi con dispositivos móviles como consolas de juegos, el Apple iPhone, iPod touch, el Sony PSP, ordenadores personales o teléfonos Android. Para descargar el SDK AR.Drone, se tiene que registrar sin ningún costo en la página y aceptar los términos del acuerdo de licencia y condiciones del SDK AR.Drone.

El SDK incluye lo siguiente:

- Documento que explica cómo utilizar el SDK y describe los protocolos de comunicación del cuatrimotor.
- Biblioteca AR.Drone (*ARDroneLIB*), que proporciona la **interfaz de programación de aplicaciones** o **API** (del inglés, *Application Programming Interface*), necesaria para comunicarse fácilmente y configurar al cuatrimotor.
- La biblioteca herramienta AR.Drone (*ARDroneTool*), que proporciona un cliente drone completamente funcional donde los desarrolladores sólo tienen que insertar su código de aplicación específica.

Figura 2.3: El AR.Drone puede ser manipulado por un móvil [17].

- La biblioteca *AR.Drone Control Engine*, que proporciona una interfaz de control intuitivo desarrollado por Parrot para controlar a distancia al cuatrimotor desde un iPhone.
- Un ejemplo de juego en iPhone de código abierto, varios ejemplos de código abierto que muestran cómo controlar al cuatrimotor desde una computadora con Windows o Linux, y un ejemplo sencillo para los teléfonos Android.

El SDK permite desarrollar aplicaciones para controlar de forma remota el AR.Drone [13]:

- Desde cualquier computadora con conexión Wi-Fi (Linux o Windows)
- Desde un iPhone
- Desde un móvil Android
- Y recientes aplicaciones para móviles Nokia

También se puede desarrollar un control de forma remota del AR.Drone desde cualquier dispositivo programable con tarjeta de red Wi-Fi y una pila TCP/UDP/IP para los dispositivos que no son compatibles con Parrot. Pero el SDK no es compatible con:

- Reescribir un software integrado propio: No está permitido el acceso directo al hardware del drone (sensores, motores).

2.2.4. Biblioteca AR.Drone

Como ya se mencionó, la biblioteca AR.Drone es una biblioteca de código abierto que incluye las API de altos niveles necesarios para acceder al cuatrimotor. Esta biblioteca cuenta con los siguientes componentes:

- SOFT: Código específico del cuatrimotor que incluye:
  - COMMON: Archivos cabecera (.h) que describen la estructura de comunicación utilizados por el cuatrimotor.
  - Lib/ardrone_tool: Conjunto de herramientas para manejar fácilmente al cuatrimotor, como un bucle de comandos AT y el envío de hilo, un navdata recibe el hilo, listo para usar video pipeline y listo para usar la función main.
- VLIB: Es la biblioteca de procesamiento de vídeo. Contiene las funciones de recibir y decodificar el flujo de video.
- VPSDK: Conjunto de bibliotecas de uso general, incluyendo [13]:
  - VPSTAGES: piezas de procesamiento de video, que se puede montar para construir un canal de procesamiento de video.
  - VPOS: Contenedores multiplataforma (Linux/Windows/Parrot plataforma propietaria) para las funciones de nivel de sistema (asignación de memoria, gestión de hilos, etc.)
  - VPCOM: Envolturas multiplataforma para las funciones de comunicación (a través de Wi-Fi, Bluetooth, etc.)
  - VPAPI: Asistentes para manejar video pipelines e hilos.
  - Ardrone_tool.c: Contiene una función main C read-to-use, que inicializa la red Wi-Fi e inicia todas las comunicaciones con el cuatrimotor.
  - UI: Contiene un código de lista para usar la gestión del gamepad.
  - AT: Contiene todas la funciones que se pueden llamar que mantienen el control del cuatrimotor. La mayoría de ellos se refieren directamente a un comando AT que se construye automáticamente con la sintaxis y secuencia de números correctos y transmitidos al manejador de hilos AT.

- NAVDATA: Contiene un navdata listo para usar para recepción y decodificado de sistema.

2.2.5. El AR.Drone Tool

El AR.Drone Tool forma parte de la biblioteca del AR.Drone. Esta biblioteca es la encargada de implementar de forma eficaz los servicios del cuatrimotor, en particular proporciona [13]:

- Un comando AT de gestión de hilo, que recoge los comandos enviados por todos los otros hilos y enviarlos de manera ordenada con números de secuencia correctos.
- Un navdata manejador de hilo que recibe automáticamente la corriente navdata, lo decodifica y proporciona la aplicación cliente con listas para usar los datos de navegación a través de una función de devolución de llamada.
- Un manejador de hilo de video, que recibe automáticamente el flujo de vídeo y proporciona la aplicación cliente con listas para usar los datos de vídeo a través de una función de devolución de llamada.
- Un control de hilo que gestione las peticiones de otros hilos para enviar comandos fiables desde el cuatrimotor, y busca automáticamente las confirmaciones del cuatrimotor.

Estos hilos cuidan de conectar al cuatrimotor desde su creación, y lo hacen mediante el uso de la biblioteca *vp_com* que se encarga de volver a conectar con el cuatrimotor cuando sea necesario.

Estos hilos, así como la inicialización requerida, son creadas y administradas mediante la función principal, también suministrada por el AR.DroneTool en el archivo *ardrone_tool.c*.

Para la implementación de nuevas aplicaciones se tienen que llenar las funciones con código específico, con la limitante que el código es en lenguaje *C*.

2.3. Exploración y reconstrucción de ambientes

Una de las metas de la robótica es la autonomía de los robots y la robótica aérea no es la excepción. Una de las principales características de autonomía es la capacidad del robot de realizar exploración, navegación y creación de mapas por sí mismo, para que pueda realizar planeación de rutas y libramiento de obstáculos. El problema de **Localización y Mapeo Simultaneo** (SLAM, del ingles *Simultaneous Localization and Mapping*) es una de las áreas de mayor interés e importancia si se desea contar con robots con mayor autonomía, que implica la capacidad de realizar planeación de desplazamiento y poder desplazarse a un objetivo específico.

2.3.1. Exploración de ambientes

Para la exploración de ambientes existen varios algoritmos SLAM, mayormente con aplicaciones a la robótica terrestre, sin embargo, a consecuencia del incremento de desarrollos en robótica aérea, se busca resolver este problema aplicando estos algoritmos que puedan ser adaptados al ambiente aéreo. Los algoritmos SLAM utilizan mayormente cámaras en aplicaciones de robótica aérea, debido a las ventajas que presenta en relación con otros sensores, como por ejemplo, menor costo, más ligeras y con buen ahorro de energía, características importantes debido a que las arquitecturas aéreas cuentan con gran limitación de peso y energía [6].

Para elegir un algoritmo SLAM se debe tener presente si el ambiente que explorará la aeronave es un espacio abierto o cerrado, de esto depende la forma de resolver el problema de la autolocalización. En un espacio abierto se podría hacer uso de los sistemas de posicionamiento global, que hacen uso de los satélites geoestacionarios para realizar la triangulación de un dispositivo y obtener su localización. Sin embargo, en un ambiente cerrado donde el error de centímetros es muy significativo hace poco viable la utilización de estos sistemas de posicionamiento global. A continuación se presentan algoritmos SLAM utilizados en robótica aérea en ambientes cerrados.

2.3.1.1. Entornos inteligentes

En este caso el ambiente cuenta con un conjunto de cámaras previamente calibradas y enlazadas [18]. El robot deberá contar con diodos infrarrojos montados en la parte superior de la aeronave con el fin de permitir la ubicación incluso en condiciones de luz tenue. Para la ubicación del robot se lleva a cabo un proceso de identificación de bordes, una vez identificado, las coordenadas del robot se obtienen de los diversos planos de la imagen. Sobre la base de estas coordenadas, se propone un algoritmo que tiene en cuenta la desviación estándar del error producido en las diferentes cámaras en la determinación de las coordenadas de los elementos de la baliza. Se utiliza un metodo para estimar la posición del robot, llamado odometría [19], que su idea fundamental es la integración de la información de movimiento gradual con el tiempo. La odometría se utiliza en la orientación que permite que la posición del robot se estime durante los intervalos de tiempo requeridos para procesar la información visual proporcionada por las cámaras.

Este método fue utilizado por Noelia [20] que realizó una aplicación para cuatrimotor que detecta llamas, en este caso solo utilizó el ambiente inteligente para detectar la ubicación de la aeronave, sin explotar alguna otra utilidad de este.

2.3.1.2. Algoritmo SIFT

El algoritmo SIFT (del ingles, *Scale Invariant Feature Transform*) fue presentado en 1999 por Lowe [21], fue utilizado en sus inicios para reconocimiento de objetos, pero más tarde se extendió su uso a aplicaciones SLAM. La idea principal es que los pixeles o puntos de la imagen pueden ser utilizados para la correspondencia de características o puntos clave. Las características deben ser invariantes de la escala y la rotación, y pueden ser descritos por vectores de descripción. El sistema selecciona las características emblemáticas que tienen distintos vectores descriptores entre los puntos de la entidad. Las ubicaciones de los puntos de referencia se calculan y almacenan en una base de datos de mapa. Con esta información puede ser calculada la posición de una aeronave como por ejemplo el trabajo realizado por Jeong-Oog Lee [22].

2.3.2. Reconstrucción de mapas

Los mapas son de gran importancia para los robots móviles por que con estos puede planear y ejecutar tareas. Un mapa es utilizado para tres objetivos principales:

- Establecer las zonas del ambiente ocupadas por obstáculos y libres para navegación.
- Reconocer regiones o lugares del ambiente.
- Reconocer objetos específicos dentro del ambiente.

Los mapas por lo general son construidos por un robot utilizando los sensores con los que cuenta como: sonares, laser, cámaras, etc. Con éstos, el mapa estará representado de acuerdo a los alcances y limitaciones de sus sensores. Existen varias propuestas para la representación de los mapas que ofrecen ventajas sobre el tipo de ambiente y los sensores de los que dispone el robot. La representación espacial puede ser dividido en dos grupos principales [23]: los basados en una descripción precisa de los objetos, denominados mapas métricos y los basados en una descripción con base a la topología del ambiente llamados mapas topológicos; existen trabajos que buscan explotar las ventajas de ambos tipos creando representaciones híbridas. En la Tabla 2.2 se muestra la comparación de los enfoques métricos y topológicos realizada por Thrun y Büken [23].

2.3.2.1. Representaciones métricas

La representación métrica debe tener alta precisión de modelado de objetos y espacios [24]; esto es, que los mapas mantienen información de las dimensiones, posición y distancia entre los objetos. Estos mapas basan su representación en los datos de distancia extraídos de sus sensores que pueden ser transformados o ajustados a tipos abstractos. Dentro de este grupo se encuentran:

a) Mapas geométricos

Los mapas geométricos se construyen a partir de primitivas geométricas como: líneas, polígonos, puntos, etc. Éstos mapas cuentan con la ventaja de poder representar ambientes de dimensiones grandes, porque, pueden

Tabla 2.2: Comparación de enfoques de mapa métrico y topológico.

Métricos	**Topológicos**
Ventajas	
Fáciles de construir, representar y mantener	Permiten una planeación eficiente, baja complejidad en el espacio
Reconocimiento de lugares, no son ambiguos y puntos de vistas independientes	No requiere determinación del robot
Facilita el computo de caminos más cortos	Representación conveniente para planificadores simbólicos, interfaces con lenguaje natural
Desventajas	
Planificación ineficiente, consumen espacio	Difíciles de construir y mantener en ambientes grandes
Requiere determinación precisa de la posición del robot	Reconocimiento de lugares frecuentemente ambiguo, sensitivo al punto de vista
Interfaz pobre para mucho de los planificadores simbólicos	Puede producir rutas subóptimas

ser representados por un modelo que incluya parámetros que describan las propiedades de las primitivas geométricas [25].

Drumheller [26], fue pionero en la utilización del enfoque geométrico al utilizar en 1987 un sonar para determinar la posición y orientación de un robot dentro de una habitación. El plano de una habitación se modela como una lista de segmentos de rectas extraídas de los datos de los sensores de ultrasonido. La Figura 2.4 muestra el ejemplo de la extracción de rectas de los sensores.

b) Mapas de celdas

El modelo de mapas de celdas fue presentado por Elfes y Moravec [27], convirtiéndose en uno de las más utilizados (véase Figura 2.5). Como men-

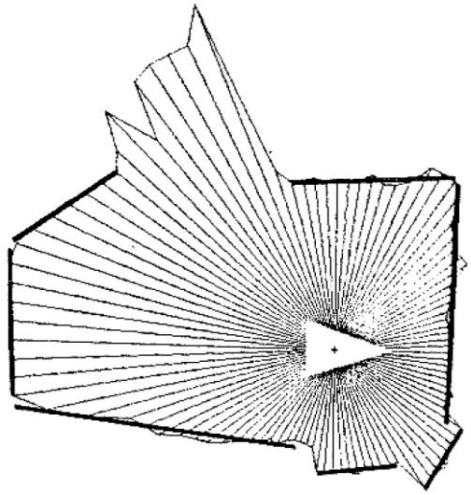


Figura 2.4: Representación de un mapa geométrico.

ciona Thrun y Büken [23], los mapas métricos son de dos dimensiones, las rejillas de ocupación son discretas y cada celda de la cuadricula (x, y) en el mapa tiene un valor de ocupación adjunta, que mide la creencia subjetiva de estar o no ocupada o puede almacenar la probabilidad de estar o no ocupada de acuerdo a las lecturas de los sensores y el robot pueda tomar la decisión de dirigirse a una celda o evitarla.

0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5
0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5
0,5	0,5	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	0,5
0,5	0,9	0,3	0,3	0,3	0,3	0,3	0,3	0,3	0,3	1,0	0,5
0,5	0,9	0,3	0,3	0,3	0,3	0,3	0,3	0,3	0,3	0,9	0,5
0,5	1,0	0,2	0,2	0,2	0,2	0,2	0,2	0,2	0,2	0,8	0,5
0,5	0,9	0,1	0,1	0,1	0,9	0,8	0,8	0,7	0,1	1,0	0,5
0,5	0,9	0,2	0,2	0,2	0,9	0,2	0,2	0,2	0,2	1,0	0,5
0,5	1,0	0,0	0,0	0,0	1,0	0,0	0,0	0,0	0,0	0,9	0,5
0,5	0,9	0,1	0,1	0,1	1,0	1,0	0,1	0,1	0,1	1,0	0,5
0,5	0,9	0,0	0,0	0,0	0,0	1,0	1,0	0,0	0,0	0,9	0,5
0,5	0,9	0,0	0,0	0,0	0,0	0,0	1,0	0,0	0,0	0,9	0,5
0,5	0,8	0,1	0,1	0,1	0,1	0,1	1,0	0,1	0,1	1,0	0,5
0,5	0,9	0,2	0,2	0,2	0,2	0,2	0,2	0,2	0,2	1,0	0,5
0,5	0,9	0,2	0,2	0,2	0,9	1,0	0,9	0,9	0,9	0,5	0,5
0,5	0,5	1,0	1,0	1,0	0,5	0,5	0,5	0,5	0,5	0,5	0,5
0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5

Figura 2.5: Representación de un mapa de celdas. Cada celda tiene asignado un valor correspondiente a la probabilidad de que este o no ocupada.

2.3.2.2. Representaciones topológicas

Las representaciones geométricas se basan en datos métricos, pero estos datos carecen de precisión debido al ruido que pueden tener las lecturas de los sensores. Estos problemas se pretenden evitar con el uso de las representaciones topológicas que se basan en la detección y reconocimiento de zonas del ambiente y su conectividad. Este tipo de mapa facilita la interacción con instrucciones humanas, como por ejemplo *"ir a la habitación A"* [28].

La representación topológica se basa en una abstracción del ambiente en términos de lugares distintivos y su adyacencia. Este modelo describe el ambiente como un grafo donde los vértices representan los lugares reconocidos por el robot y las aristas su conectividad [29]. El mapa se representa con un grafo [30]:

$$G = (V, E)$$

Donde V es un conjunto de n vértices:

$$V = \{v_1, v_2, ..., v_n\}$$

Y E un conjunto de m aristas:

$$E = \{e_1, e_2, ..., e_m\} \text{ donde } e_k = \{v_i, v_j\}$$

Para Kuipers y Byun [31], en un mapa topológico un vértice del grafo es identificado en base a una medida local distinguible y las aristas entre vértices son las estrategias de control que indican la forma de alcanzar los nodos adyacentes. La Figura 2.6 muestra un mapa topológico.

2.3.2.3. Representaciones híbridas

Un mapa híbrido es la combinación de dos o más tipos de mapas. Por lo general los mapas métricos y topológicos son utilizados para crear este tipo de mapa. Por definición, un mapa híbrido tiene más enlaces que conectan los elementos de un mapa a los elementos de otro mapa. De no tener estos enlaces, el

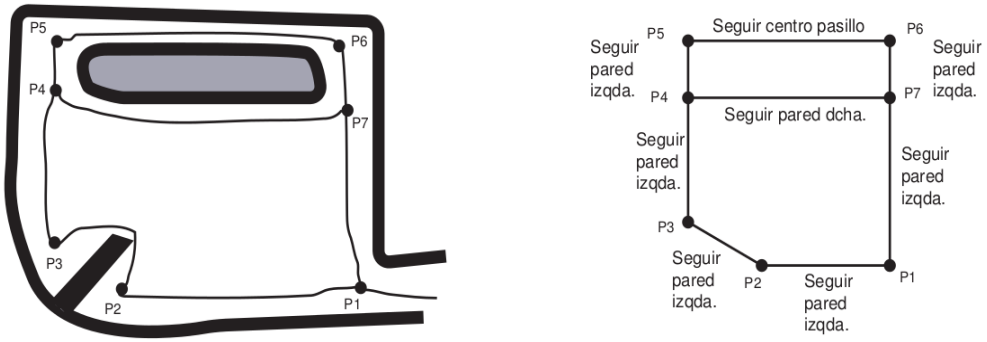


Figura 2.6: Representación de un mapa Topológico. Mapa topológico de Kuipers y Byun.

mapa híbrido seria solo una colección de mapas sin relación alguna. Un ejemplo simple de este tipo de mapa es un mapa de una ciudad combinado con un mapa de carreteras como en la Figura 2.7.

Mencionando a todos los mapas tanto los métricos como los topológicos son fundamentalmente diferentes, así como también tienen fortalezas y debilidades, al igual que ventajas y desventajas descritas en la Tabla 2.2. Es fácil ver la alta densidad que tiene un mapa métrico de objetos en comparación con un mapa topológico. Por esto, el mapa métrico tiene una mayor resolución a un mapa topológico. Y al utilizar un mapa métrico, es posible una localización y planificación de ruta con mayor precisión que utilizando uno topológico.

Por otro lado la menor densidad, junto con la información de relación dada en un mapa topológico hace posible una planificación más simple y más rápida que si se utilizara un mapa métrico. Por estas dos razones es más fácil de ampliar a entornos más grandes que un mapa métrico.

Menciona Thrun [32], que los mapas métricos son más fáciles de construir, representar y mantener que los mapas topológicos. No se requiere de una estimación de la posición precisa para la construcción de mapas topológicos y métodos para la localización. El problema en la práctica es la de asociación de datos para el reconocimiento de nodos que pueden hacer a los mapas topológicos difíciles de construir en entornos grandes.

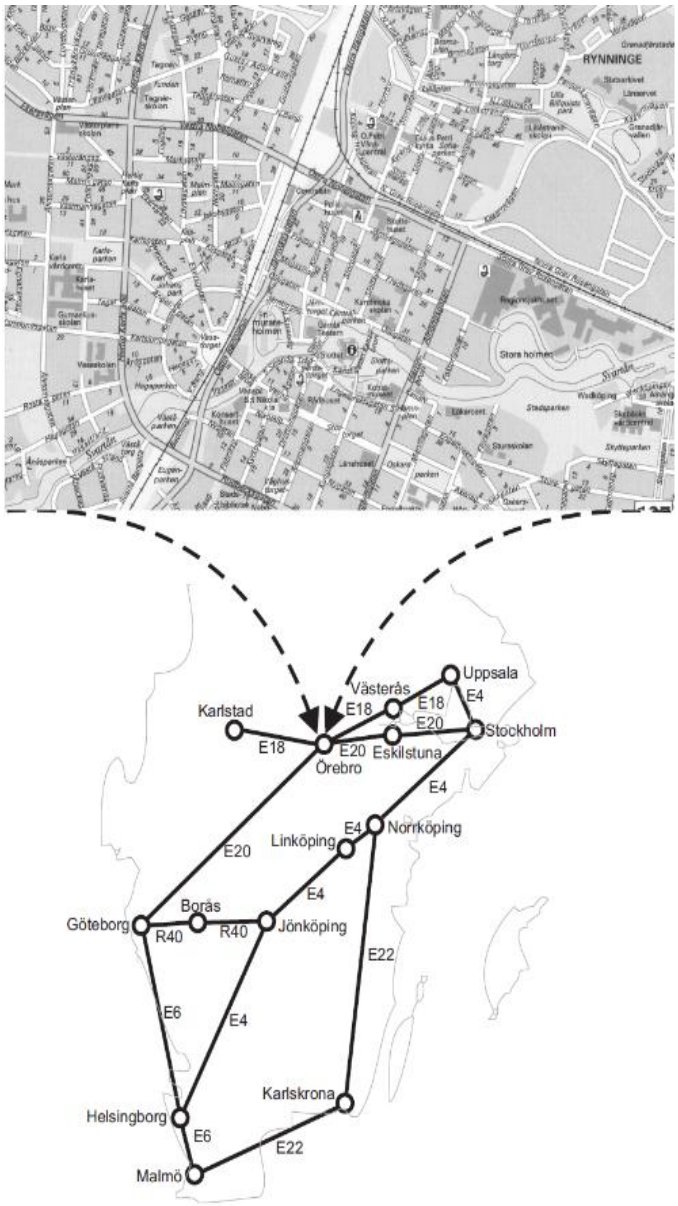


Figura 2.7: Representación de un mapa híbrido. Un mapa híbrido puede ser representado por el conjunto de un mapa de carreteras y un mapa de la ciudad.

Se hace evidente que las ventajas y desventajas mencionadas se complementan de diversas maneras. Lo que es negativo con un mapa métrico, es positivo con un mapa topológico y viceversa. Esto es el motivo principal del desarrollo de los mapas híbridos, que son combinaciones de varios mapas diferentes, como el que se muestra en la Figura 2.8, donde se combinan un mapa de celdas con uno

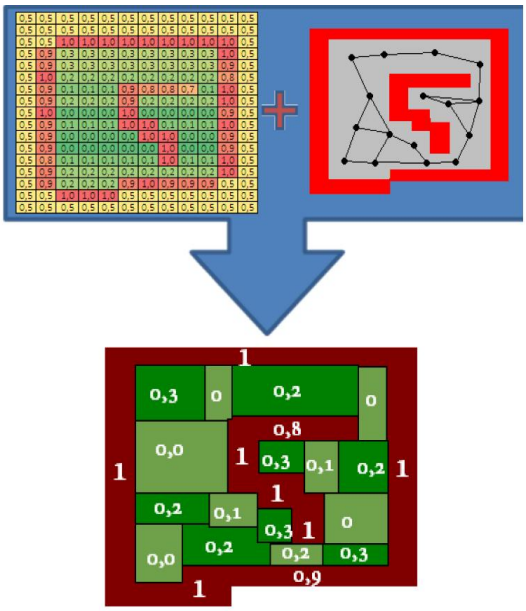


Figura 2.8: Ejemplo de un mapa híbrido. Se mezclan los valores de cada celda del mapa de celdas para representar cada nodo del mapa topológico.

topológico. Un mapa híbrido ayuda a superar las debilidades de cada tipo de mapa y para cada tarea utilizar el componente en el mapa híbrido que sea el más adecuado para la tarea.

2.4. OpenCV

Open Source Computer Vision Library (OpenCV) [33], es una biblioteca de visión por computadora de código abierto, está escrito en C y C++ y corre bajo Linux, Windows y Mac OS X. Existe un desarrollo activo de interfaces para Python, Ruby, Matlab y otros lenguajes. Fue diseñado para la eficiencia computacional y aplicaciones en tiempo real. Su principal objetivo es proporcionar una infraestructura que facilite la utilización de la visión por computadora para ayudar a construir rapidamente aplicaciones de visión sofisticadas.

OpenCV contiene más de 500 funciones utilizadas en muchas áreas de la visión incluyendo el área de imágenes médicas, seguridad, interfaces de usuario, calibración de cámaras, visión estéreo y la robótica. Debido a que la visión por computadora y aprendizaje automático frecuentemente van de la mano,

OpenCV tiene una biblioteca de aprendizaje automático de propósito general, que se centra en el reconocimiento de patrones y clustering.

2.5. Trabajos relacionados

La robótica aérea está siendo objeto de investigaciones innovadoras para aplicaciones civiles muy diversas, como la vigilancia de fronteras, detección de incendios, servicios de rescates en desastres naturales, entre otras. Los robots aéreos pueden ser implementados con diversas características funcionales, como por ejemplo el trabajo realizado por Andriluka y colaboradores [34], en el que un robot aéreo es capaz de reconocer personas recostadas en el suelo, con el fin de ayudar en labores de rescate al poder ubicar en un ambiente a personas recostadas.

Para la exploración de ambientes se pueden utilizar las imágenes recuperadas por cámaras como en el trabajo de Lee [22], que utiliza el algoritmo *SIFT*, que es un método de detección de características basado en visión. El algoritmo primero extrae puntos característicos de los datos de la imagen tomada por una cámara monocular. El sistema selecciona los puntos emblemáticos de características que tienen distintos vectores descriptores entre los puntos de la entidad. Entonces las ubicaciones de los puntos de referencia se calculan y almacenan en una base de datos de mapa. Con base en la información, la posición del robot aéreo es calculada.

Un ejemplo más de la navegación basado en sensores de videos, es el realizado por Kumar [35], en este trabajo la técnica *SLAM* utiliza el algoritmo Filtro de Kalman Extendido(*EKF SLAM* del inglés, Extendend Kalman Filter) [36], que trata de aproximar los modelos de movimiento y sensores no lineales para hacerlos lineal; éste filtro cuenta con dos pasos importantes. El primer paso es crear un mapa usando las mediciones de los sensores; el siguiente paso después de un movimiento y observaciones posteriores de las lecturas de los sensores de los mismos puntos de referencia, utilizando la actualización de movimiento y actualización de la medida de un filtro Kalman [37].

Existen algoritmos de navegación que utilizan otro tipo de sensores como es el caso del trabajo realizado por Bachrach [38], en el que utilizan sensores laser para explorar y generar mapas. Los autores utilizaron la aplicación disponible del algoritmo Gmapping que está disponible en el repositorio OpenSlam, que realiza *SLAM* en *2D*. Gmapping se basa en un modelo de movimiento estándar para robots de tierra con odometría en ruedas, sin embargo, usaron estimaciones calculadas por el módulo de escaneo láser, modificaron Gmapping de movimiento para propagar las partículas, por medio de las incertidumbres calculadas por el módulo de análisis de coincidencia.

Capítulo 3

Algoritmos de reconocimiento de objetos

Los algoritmos de aprendizaje han sido de gran utilidad práctica en una gran variedad de aplicaciones. Utilizados ampliamente en mineria de datos para descubrir conocimiento valioso a partir de grandes bases de datos comerciales que contienen transacciones financieras, registros médicos, entre otros, lugares donde el programa debe adaptarse dinámicamente a las condicionces cambiantes, como por ejemplo, el control de procesos de fabricación [2]. Un ejemplo de algoritmo de aprendizaje es el de *árbol de decisión*, siendo de los más utilizados por su facilidad de implementación y buen desempeño para resolver problemas, el algoritmo puede ser representado con un árbol de decisión, pero también con un conjunto de reglas si-entonces para mejorar la legibilidad.

La utilización de imágenes en el entrenamiento de algoritmos de aprendizaje automático tiene un alto costo computacional, directamente proporcional con las dimensiones de las imágenes. Una forma de reducir este problema, es a través de la extracción de características que Wan y Li [39] describen como la extracción de características irrelevantes y redundantes, utilizando un subconjunto de características con el objetivo de reducir el número de características y manteniendo la información relevante. Para esto, se utilizan algoritmos especializados en extracción de características.

Para Nandhitha et al. [40], el objetivo de un algoritmo de extracción de características es el procesamiento de una imagen digital para la extracción de

la región de interés y describir cuantitativamente las características de la región extraída. Los principales beneficios de la utilización de estos algoritmos es que el número de muestras de entrenamiento no necesita ser muy grande y por otro lado, el algoritmo de clasificación puede funcionar más rápidamente y producir resultados más simples y precisos.

3.1. Algoritmo de detección de objetos

La detección de objetos es descrita por Pratt [41] como la determinación de la presencia o ausencia de objetos de interés en una imagen. Se refiere a la comparación de plantillas, en la que una réplica de un objeto se compara con los objetos de una imagen. Los algoritmos de detección de objetos se encargan de crear las plantillas o en un término más estricto clasificadores, utilizando las características de los objetos de interés que el algoritmo se encarga de extraer para crear los clasificadores.

Los algoritmos de detección de objetos son un tipo de aplicación de los algoritmos de aprendizaje automático. Uno de los principales algoritmos de detección de objetos es el de Viola-Jones [12], que logra una taza de detección de rostros comparables con los mejores algoritmos de detección de rostros, pero a una mayor rapidez.

El algoritmo de Viola-Jones se basa en cuatro puntos principales:

1. Extracción de características utilizando las características de Haar [12].

2. Reducción de tiempo de caracterización basado en la imagen integral [12].

3. Optimización del tiempo de entrenamiento con la estructura de clasificador en cascada [12].

4. Entrenamiento del clasificador con el algoritmo de aprendizaje Adaboost [12].

3.1.1. Extracción de características utilizando las características de Haar

En el año 2000 Papageorgiou y Poggio [42], propusieron la utilización de las características de Haar bajo el concepto de ondícula de Haar, que es una representación de una señal en términos de versiones trasladadas y dilatadas de una onda finita que tiene un valor promedio cero [43].

Para Papageorgiou y Poggio las características de Haar, identifican las características locales, las características orientadas, la diferencia de intensidad en diferentes escalas y es computable eficientemente. Viola y Jones [12], aplicaron este concepto como la diferencia de la suma de los pixeles en dos o más zonas rectangulares adyacentes; con tres tipos de características que se pueden observar en la Figura 3.1

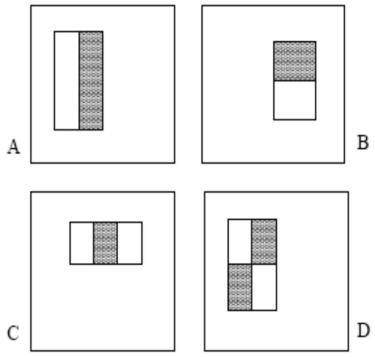


Figura 3.1: Ejemplo rectángulo de características; la suma de los píxeles que se encuentran dentro de los rectángulos blancos se restan de la suma de los píxeles en los rectángulos grises. En (A) y (B), se muestran funciones de características de dos rectángulos. La figura (C) muestra una función de tres rectángulos, y en (D) de cuatro rectángulos.

El valor de una función de dos rectángulos es la diferencia entre la suma de los píxeles dentro de las dos regiones rectangulares. Las regiones tienen el mismo tamaño y forma, y están horizontalmente o verticalmente adyacentes (véase la Figura 3.1). Una de las características de tres rectángulos es que calcula la suma en dos rectángulos externos y restan la suma en un rectángulo central.

Finalmente una característica de cuatro rectángulos calcula la diferencia entre los pares diagonales de los rectángulos [12].

3.1.2. Reducción de tiempo de caracterización basado en la imagen integral

Viola-Jones utilizan la imagen integral para reducir el tiempo necesario para el cálculo de las características de Haar. La imagen integral también conocida como tabla de áreas sumadas, fue propuesta por Crown [44], es un algoritmo que calcula la suma de los valores de un rectángulo de pixeles de un subconjunto de puntos en una matriz. Implica hacer una suma de todos los pixeles que estén arriba y a la izquierda de un punto (x,y) en la imagen. En la Fórmula 3.1 [12], se describe formalmente:

$$ii(x,y) = \sum_{x' \leq x, y' \leq y} i(x',y') \tag{3.1}$$

donde:

ii(x,y) es la imagen integral
i(x',y') es la imagen original.

También se pueden utilizar las siguiente Fórmulas 3.2,3.3 recursivas para construir la imagen integral en un solo recorrido:

$$s(x,y) = s(x,y-1) + i(x,y) \tag{3.2}$$

$$ii(x,y) = ii(x-1,y) + s(x,y) \tag{3.3}$$

En donde *s(x,y)* es la suma de los pixeles de la fila x, también se debe considerar que *s(x,-1)=0*, y *ii(-1,y)=0* [12] [45]. Los pixeles se suman de arriba hacia abajo y de izquierda a derecha por lo que el último cuadro será la suma de todos los anteriores más éste mismo, como se muestra en la Figura 3.2.

Como se muestra en la Figura 3.3, la suma de cualquier área rectangular sobre la imagen y la suma de los pixeles de la región ABCD puede ser calculada mediante la Fórmula 3.4, que requiere cuatro referencias del arreglo de la imagen integral.

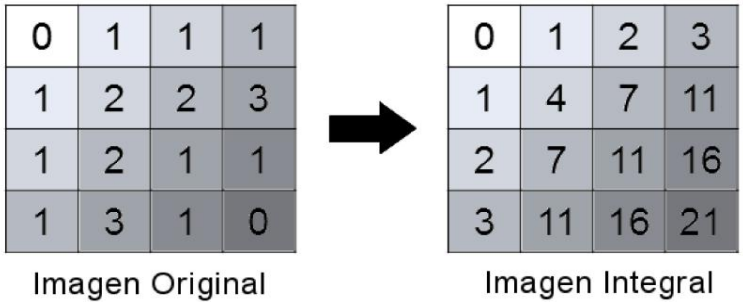


Figura 3.2: Imagen integral respecto de la imagen original. Los pixeles se suman de arriba hacia abajo y de izquierda a derecha.

$$\sum_{(x,y)\in ABCD} i(x,y) = ii(D) + ii(A) - ii(B) - ii(C) \tag{3.4}$$

Viola y Jones utilizaron este concepto para calcular las características rectangulares simples de Haar.

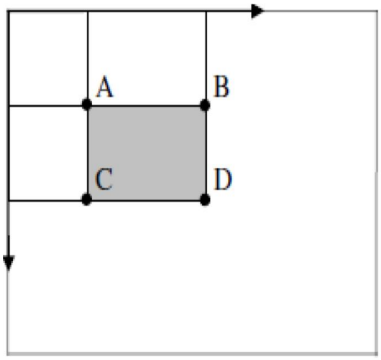


Figura 3.3: Cálculo del área en gris mediante cuatro referencias como *D-(B+C)+A*.

3.1.3. Optimización del tiempo de entrenamiento con la estructura de clasificador en cascada

Viola y Jones utilizan AdaBoost que es muy eficiente para combinar clasificadores débiles que se refieren a clasificadores moderadamente precisos que funcionan levemente mejor que una clasificación aleatoria, mediante un conjunto de características y un conjunto de entrenamiento (imágenes positivas y negativas) para obtener un clasificador fuerte.

Si se cuenta con imágenes de 24 x 24 pixeles, existen aproximadamente 160,000 rectángulos de características por imagen, superando el número de pixeles contenidos en ellas. Hacer el cálculo de las características puede ser muy costoso computacionalmente; por éste motivo Viola y Jones utilizaron una variante de AdaBoost para la selección de características y para el entrenamiento del clasificador [12].

El algoritmo de AdaBoost se utiliza para mejorar el rendimiento de un solo algoritmo de aprendizaje, pero en éste algoritmo se utiliza en conjunto con una cascada de clasificadores débiles para crear un clasificador fuerte. La idea principal es crear niveles donde un clasificador débil rechace ejemplos negativos. Inicialmente, se acepta un grupo de ejemplos positivos y otro de ejemplos negativos es rechazado; el proceso se repite en cada nivel.

El primer clasificador evalúa una subventana y si es aceptada avanza al segundo clasificador, donde se evalúa y si es aceptada avanza al siguiente. Este proceso continúa en cada uno de los niveles hasta que un clasificador rechace la subventana o llegue hasta el último clasificador. Si la subventana es aceptada por el último clasificador entonces es aceptada como un ejemplo positivo. Como se muestra en la Figura 3.4.

Los niveles en la cascada se construyen utilizando Adaboost. Se inicia con un clasificador fuerte de 2 características. Un filtro eficaz puede ser obtenido ajustando el umbral para minimizar detecciones de objetos de interés, de objetos que en realidad no lo son (falsos positivos).

Se inicia con el humbral $\frac{1}{2}\sum_{t=1}^{T}\alpha_t$, diseñado para disminuir el error del entrenamiento.

Un umbral bajo produce un alto número de detecciones al igual que un alto número de falsos positivos. Utilizando un conjunto de entrenamiento de validación, el clasificador de dos características puede ajustarse para detectar el 100 % de objetos con una razón de falsos positivos del 50 % [12].

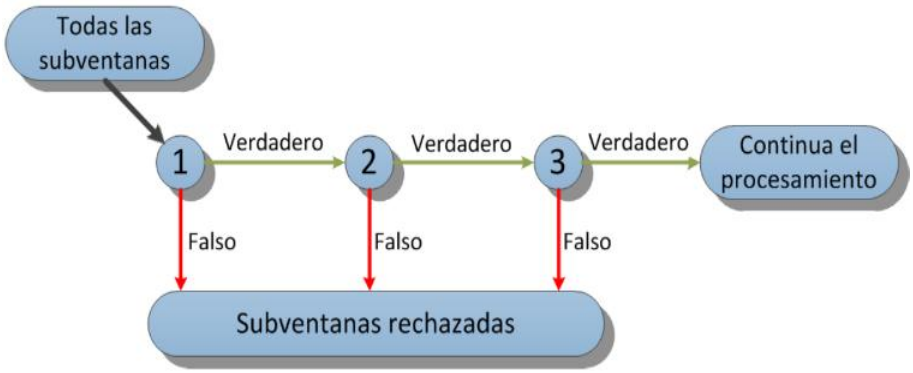


Figura 3.4: Esquema de la detección en cascada.

La estructura de la cascada refleja el hecho que dentro de cualquier imagen existe una gran mayoría de sub-ventanas negativas, que la cascada intenta rechazar el mayor número de ejemplos negativos que sea posible en las etapas más tempranas. Mientras que un ejemplo positivo ejecutará la evaluación de cada clasificador de la cascada.

La cascada es similar a un árbol de decisión, donde los clasificadores posteriores se entrenan utilizando los ejemplos que pasaron por las etapas anteriores. Cada clasificador evalúa las características dentro de una subventana para decidir si avanza al siguiente nivel o no; repitiéndose en cada nivel de la cascada. Esto da como resultado que, el siguiente clasificador tenga un trabajo más difícil que el anterior debido al incremento de características a evaluar [12].

La principal ventaja de este algoritmo, es el hecho de que rechaza ejemplos negativos sin necesidad de que recorra todas las etapas, eficientando el tiempo de ejecución.

3.1.4. Entrenamiento del clasificador con el algoritmo de aprendizaje Adaboost

Boosting [46] convierte un clasificador débil en otro fuerte, construyendo varios clasificadores en niveles sucesivos, que basan sus resultados en los obtenidos en los niveles anteriores, tiene su origen en el modelo PAC(*Probably Approximation Correct*) [47], que muestra el límite del número de ejemplos

de entrenamiento necesarios para obtener un alto grado de aprendizaje y que más tarde evolucionó para hacer posible convertir un algoritmo débil en uno fuerte [46].

En 1999 Freund y Schapire [48], crearon un algoritmo bosting llamado Adaboost (*Adaptive Boosting*). Para Sierra [46], algunos experimentos con Adaboost han demostrado que funciona bien con los árboles de decisión como C4.5.

Viola y Jones utilizaron Adaboost para el entrenamiento. En el Algoritmo 2.1 se muestra y se explica a continuación.

Como parámetros de entrada se tiene un conjunto de entrenamiento etiquetado como positivos y negativos (x_i, y_i) y el resultado final será un clasificador fuerte C. En la línea 1 del algoritmo, se considera un conjunto de entrenamiento $S = \{(x_i, y_i), i = 1, ..., n\}$, donde x_i pertenece a un conjunto de ejemplos X e y_i a un espacio finito de clases $Y = \{1, 0\}$, donde $y_i = 1$ e $y_i = 0$ son ejemplos positivos y negativos, respectivamente. En la línea 2 se inicializa un vector de pesos para cada ejemplo tanto positivo como negativo.

En la línea 3, se inicia un ciclo desde uno hasta el número máximo de niveles del entrenamiento T, en la línea 4, se hace una normalización de todos los pesos. En la línea 5, se hace una selección del mejor clasificador que minimiza el error ϵ_t de la línea 6 y es definido como el clasificador del nivel t en la línea 7. En esta selección para una característica, los ejemplos están ordenados basados en el valor de la misma. El umbral óptimo de la característica se calcula en un solo recorrido sobre esa lista. Al final se tienen 4 sumas: las sumas totales de pesos de los ejemplos positivos T^+, negativos T^-, de los ejemplos positivos anteriores al actual S^+ y los negativos anteriores al actual S^-.

El error de un umbral que divide el rango entre el ejemplo actual y los anteriores de la lista ordenada se muestra en la Fórmula 3.5, usando el de menor error que corresponde al mejor clasificador [12].

Algoritmo 2.1 Algoritmo Adaboost utilizado por Viola-Jones [12].

Entrada Conjunto de entrenamiento $(x_1, y_1), ..., (x_n, y_n)$
Salida Clasificador fuerte $C(x)$

1. Dado un conjunto de entrenamiento $(x_1, y_1), ..., (x_n, y_n)$ donde $x_i \in X$ conjunto de instancias, e $y_i \in Y = 0, 1$ para ejemplos negativos y positivos respectivamente.

2. Inicializa los pesos $w_1 \leftarrow \frac{1}{2m}, \frac{1}{2l}$ para $y = 0, 1$ respectivamente, donde m y l son el total de ejemplos negativos y positivos respectivamente.

3. **Para** los niveles $t \leftarrow 1$ **hasta** T **Hacer**

4. Normalizar pesos $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$

5. Seleccionar el mejor clasificador débil h respecto a su error
$\in_t = min_{f,p,\theta} \sum_i w_i |h(x_i, f, p, \theta) - y_i|$

6. Se define un clasificador $h_t(x) = h(x, f_t, p_t, \theta_t)$ donde f_t, p_t, y θ_t minimizan el error $\in_t$

7. Actualizar los pesos:

$$w_{t+1,i} \leftarrow w_{t,i} \beta_t^{1-e_i}$$

donde $e_i = 0$ si el ejemplo x_i es clasificado correctamente, de otro modo $e_i = 1$ y $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$

8. **Fin Para**

9. El clasificador fuerte final es:

$$C(x) = \begin{cases} \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 \text{ de otro modo} \end{cases}$$

donde $\alpha_t = \log \frac{1}{\beta_t}$

$$e = min(S^+ + (T^- - S^-), S^- + (T^+ - S^+)) \tag{3.5}$$

En la actualización de los pesos en la línea 7, le permite al algoritmo mejorar en la clasificación de falsos positivos para los niveles siguientes; finalmente en la línea 9, se obtiene un clasificador fuerte que es una combinación lineal de todos los clasificadores débiles almacenados y sus pesos.

3.2. Implementación del algoritmo Viola-Jones en el cuatrimotor para el reconocimiento de figuras básicas

Viola-Jones describe un método de detección de objetos rápido en visión computacional [49]. Una de las principales aplicaciones de éste algoritmo es para la detección de rostros, para lo cual es posible una detección práctica con un consumo mínimo de poder computacional. Este algoritmo se basa en la detección de bordes, por lo que el color en la imagen no tiene relevancia para su funcionamiento. El algoritmo Viola-Jones utiliza una serie de clasificadores para llevar a cabo el entrenamiento que dará como resultado un clasificador con mayor exactitud, debido a que es el resultado de la suma de los resultados de los clasificadores utilizados en el entrenamiento.

La implementacion del algoritmo en OpenCV [33], tiene el nombre de Cascadas de Haar y tiene la capacidad de finalizar el entrenamiento en el momento en que los resultados obtenidos por los clasificadores sean suficientes para una buena detección de objetos. Éste algoritmo de reconocimiento utiliza el método de cascada para la identificación, para esto, divide la imagen en una serie de ventanas en las que aplica la identificación con los identificadores que tienen el menor poder de identificación pero con ello demanda menor poder de cómputo, eliminando la mayor parte de las ventanas de forma rápida y concentrando el mayor poder de identificación en las ventanas que necesitan una identificación más eficiente. Por estas características es conveniente su implementación para la identificación de cuatro figuras diferentes al mismo tiempo.

Para éste trabajo, se requiere detectar o reconocer cuatro objetos diferentes. Para lograr esto, se llevo a cabo el entrenamiento utilizando el algoritmo Viola-Jones para cuatro figuras básicas: círculo, cuadrado, rectángulo y triángulo, mostradas en la Figura 3.5. Se integró un conjunto de imagenes por cada objeto para el entrenamiento, conformados por 2687 imágenes negativas y 1000 imágenes positivas, pero los niveles de entrenamiento necesarios no fue el mismo, debido a las características individuales. De cada uno de los entrenamientos se obtiene un identificador para cada objeto entrenado.

(a) Círculo (b) Cuadrado (c) Rectángulo (d) Triángulo

Figura 3.5: Objetos de interés para detección con las cámaras del AR.Drone (a) Círculo (b) Cuadrado (c) Rectángulo (d) Triángulo.

Con los cuatro clasificadores que se obtienen del algoritmo de Cascadas de Haar, se implementa la detección de objetos en el video tomado por alguna de las cámaras del cuatrimotor. En la Figura 3.6 se presenta el diagrama general de la detección de los objetos de interés y a continucación se explica. El cuatrimotor cuenta con dos cámaras que pueden utilizarse para la detección de los objetos. El cuatrimotor envía el video a la computadora por medio de un enlace wi-fi que existe entre ambos. Debido a que el formato de video con el que esta configurado el SDK no puede ser manejado por OpenCV (ya que no es compatible), se hace un cambio de formato para poder ejecutar el reconocimiento de objetos Cascadas de Haar de OpenCV.

Para el proceso de reconocimiento de los objetos, se sigue la secuencia de pasos que a continuación se explican.

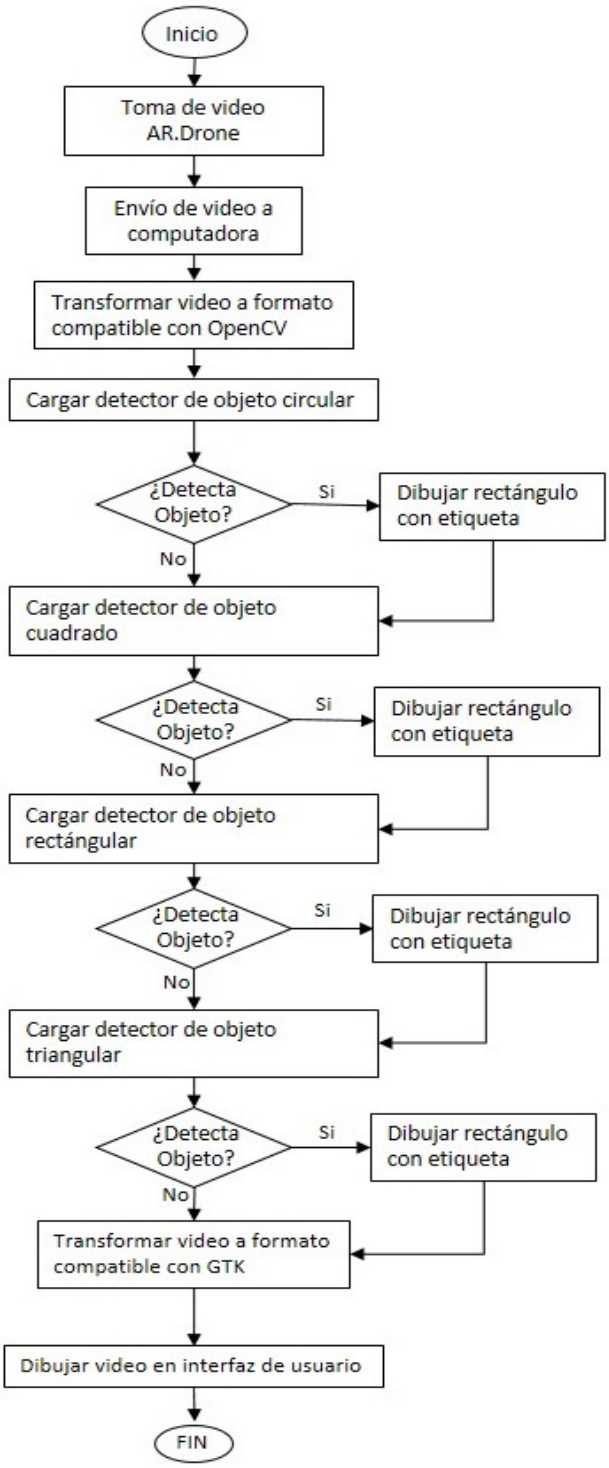


Figura 3.6: Diagrama de la secuencia de pasos para la detección de objetos.

1. Se carga el identificador para el primer objeto (círculo).
2. Se ejecuta el identificador Cascadas de Haar (identificador y video como parámetros).
3. Si se encuentra algún objeto, se dibuja un rectángulo de color predeterminado para el tipo de objeto y le anexa una etiqueta con el nombre del objeto; se le implementó un contador para el número de veces que un objeto es identificado, cada etiqueta tiene el número de identificación del objeto.
4. Se repiten los pasos anteriores para los siguientes identificadores.

Una vez terminado el proceso de reconocimiento para los cuatro objetos de formas simples, el video es transformado a un formato compatible con GTK que es la biblioteca con la que se desarrolló la interfaz para que pueda ser mostrado en la misma.

Esta biblioteca cuenta con un programa para evaluar la exactitud de los clasificadores que permite conocer la cantidad de identificaciones correctas y erroneas de los clasificadores. Para la evaluación se utilizó una muestra de 1000 imágenes por cada objeto, en las que se tenían a los objetos a evaluar su identificación; en la Tabla 3.1 se muestran los resultados de los identificadores.

Tabla 3.1: Resultados de la evaluación de los clasificadores obtenidos del algoritmo cascadas de Haar implementado en OpenCV.

Figura	**Correctos**	**Erroneos**
Círculo	834	166
Cuadrado	929	71
Rectángulo	960	40
Triángulo	933	67

Para poder comprender estos datos obtenidos del evaluador, en relación a la eficiencia de la detección de objetos en el ambiente, es necesario aplicar al menos una métrica de evaluación. La métrica aplicada en este caso es la *exactitud*. La exactitud [50] representa el porcentaje de detecciones correctas. El cálculo de esta métrica se presenta en la Fórmula 3.6.

$$exactitud = \frac{tp + tn}{tp + tn + fp + fn} \tag{3.6}$$

donde:

tp: clasificaciones verdaderas positivas
tn: clasificaciones verdaderas negativas
fp: clasificaciones incorrectas o falsos positivos.
fn: clasificaciones falsas negativas

Donde el resultado puede ser multiplicado por 100 para ser expresado en porcentaje. En la Tabla 3.2 se muestran los resultados de aplicar la Fórmula 3.6 a los resultados mostrados en la Tabla 3.1.

Tabla 3.2: Resultados de la aplicación de la métrica de exactitud a los resultados de los clasificadores obtenidos del algoritmo cascadas de Haar implementado en OpenCV.

Figura	**Porcentaje de exactitud**
Círculo	83 %
Cuadrado	92.9 %
Rectángulo	96 %
Triángulo	93 %

Capítulo 4

Desarrollo de interfaz gráfica de usuario

Una **interfaz gráfica de usuario** [51] (**GUI** del inglés: *Graphical User Interface*), es un tipo de interfaz que permite a los usuarios interactuar o comunicarse con los dispositivos electrónicos que utilizan imágenes en lugar de comandos de texto, facilitando el monitoreo de los sensores con que cuenta un sistema o en éste caso la orientación y posición de un robot aéreo[52] tipo cuatrimotor.

La implementación de una **GUI** es muy importante para la operación del cuatrimotor, por que a través de ésta un usuario puede llevar a cabo la manipulación sin necesidad de conocer los comandos de locomoción, al mostrar al usuario de forma gráfica mediante botones las opciones de navegacion. También facilita el monitoreo de las lecturas de los sensores entre los que se encuentran cámaras de video.

4.1. Herramientas de desarrollo

Se desarrolló una **GUI** para la operación y monitoreo del cuatrimotor. Para llevar a cabo éste desarrollo, fue necesario considerar las herramientas de desarrollo disponibles y compatibles con el SDK del cuatrimotor. La información que los sensores obtienen del ambiente es indispensable para la navegación, al igual que los comandos de locomoción.

Se utilizó GTK (*"GIMP Tool Kit"*) o kit de herramientas GIMP (*"GNU Image Manipulation Program"*) como herramienta de desarrollo para la interfaz. La biblioteca GTK [53], está escrita en C pero fue diseñada para apoyar una amplia gama de lenguajes como C/C++, Perl y Python, C#, JAVA y PHP, entre otros. GTK es una herramienta de software libre, parte del proyecto GNU. La herramienta GTK es compatible con el SDK del AR.Drone escrito en lenguaje C. La interfaz está integrada por un conjunto de pequeñas aplicaciones llamadas Widget, como ventanas, botones, etiquetas, entre otras, manejados por GTK.

El API SDK para desarrollar la interfaz considera las siguientes funcionalidades:

- Comandos de locomoción: Para los movimientos elementales de locomoción del cuatrimotor, se cuenta con desplazamientos en los tres ejes (ver Figura 4.1), para poder llevar a cabo movimientos elementales de locomoción:
  - Control de la altura: hace que el cuatrimotor se eleve o descienda a lo largo del eje z.
  - Giros: el cuatrimotor gira hacia la derecha o a la izquierda, este movimiento actua alrededor del eje z, también conocido como giro de guiñada.
  - Desplazamiento bidireccional: se desplaza hacia adelante o hacia atrás, éste movimiento se lleva a cabo a lo largo del eje x.
  - Desplazamiento derecha-izquierda: el cuatrimotor se desplaza hacia la derecha o izquierda, éste desplazamiento se lleva a cabo a lo largo del eje y.
- Información de sensores: Para mantener un monitoreo de la navegación y estado del cuatrimotor. El SDK recibe información de los sensores que describen la navegación del cuatrimotor en el ambiente, como las velocidades en los tres ejes x, y, z, la elevación del robot, los ángulos de orientación en los tres ejes y el porcentaje de carga de la batería.

- Video: El cuatrimotor cuenta con dos cámaras de las que se puede recuperar video o fotografías del ambiente explorado.

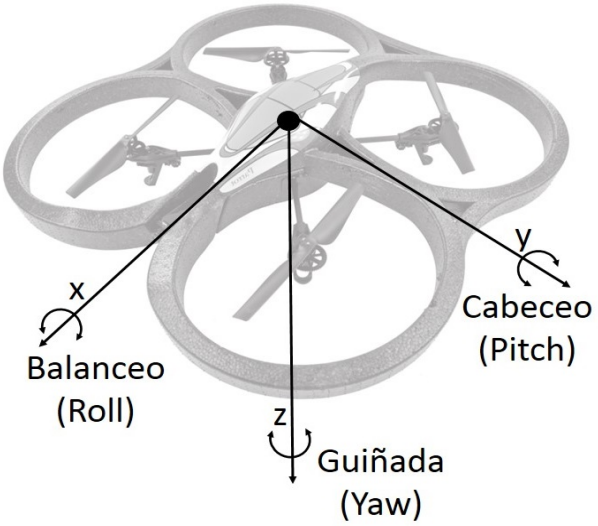


Figura 4.1: Rotaciones del cuatrimotor [54]: Guiñada (Yaw) en el eje Z, cabeceo (Pitch) en el eje Y y balanceo (Roll) en el eje X.

4.2. Diseño de la interfaz

Para el diseño de la interfaz, se requiere implementar secciones para tareas específicas, que permitan cumplir con el objetivo para el cual se esta desarrollando la interfaz gráfica. Las secciones necesarias se mencionan a continuación:

1. Sección de comandos de locomoción
2. Sección de recuperación de video
3. Sección de monitoreo de sensores
4. Sección de mapa

4.2.1. Sección de comandos de locomoción

Para poder enviar los comandos de locomoción al cuatrimotor, es necesario contar con una sección, donde por medio de botones, se pueda navegar con el

cuatrimotor de forma intuitiva. Para ésto, se desarrolló una botonera con las funciones elementales de locomoción del cuatrimotor, como se muestra en la Figura 4.2.

Figura 4.2: Sección de comandos de locomoción para el cuatrimotor.

Los botones se encuentran agrupados de forma que faciliten la comprensión de los movimientos o funciones que realiza cada uno. La principal sección esta formada por botones con flechas y fondos de colores, se puede observar que los botones se encuentran agrupados en pares, de forma que cada par corresponde a movimientos opuestos entre si que actúan sobre el mismo eje. El SDK cuenta con una función que hace posible la navegación del cuatrimotor, esta función envia comandos de locomoción para que el cuatrimotor se desplace.

A continuación se presenta una breve explicación de las funciones implementadas para cada botón de locomoción:

- Botones de control de altura: Estos botones envían comandos de locomoción relacionadas con la altura del cuatrimotor. Éstos botones manipulan la velocidad vertical del cuatrimotor para que se desplace sobre el eje z, el movimiento es representado en la Figura 4.3 por el eje remarcado. Las dirección de las flechas (arriba-abajo) de los botones corresponden a la dirección del desplazamiento del cuatrimotor.

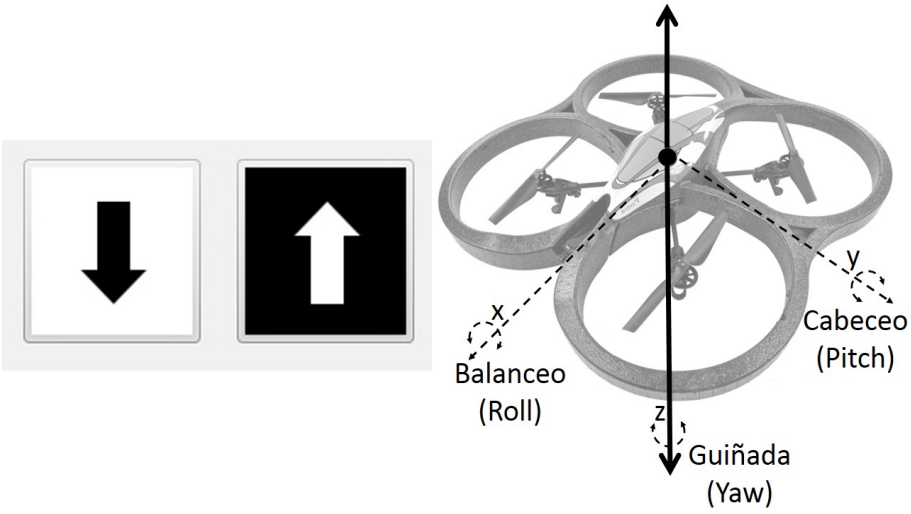


Figura 4.3: Estos botones incrementan o decrementan la velocidad vertical del cuatrimotor para llevar a cabo un desplazamiento sobre el eje z.

- Botones de función de desplazamiento bidireccional: Los botones que se muestran en la Figura 4.4, corresponden a las funciones de traslación sobre el eje x. El botón de fondo verde envía los comandos de locomoción para el desplazamiento hacia el frente, representado por una flecha verde sobre el eje x. El botón de fondo rojo, ejecuta un desplazamiento hacia atras y es representado por una flecha roja en la misma figura. Para poder hacer este desplazamiento, el cuatrimotor ejecuta un giro de *cabeceo*.

- Botones de movimiento de izquierda-derecha: Esta sección comprendida por los botones que se muestran en la Figura 4.5, son para el desplazamiento sobre el eje y; el botón de fondo azul, se encarga de enviar los

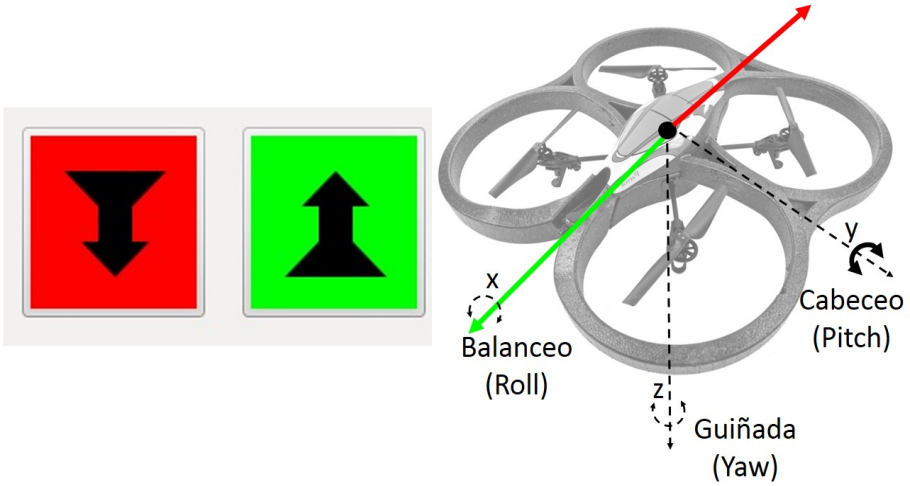


Figura 4.4: Botones que manipulan el avance y retroceso del cuatrimotor, el desplazamiento es sobre el eje x (ver flechas de colores), ejecutado por un giro de *cabeceo*.

comandos de locomoción para el desplazamiento hacia la izquierda y el botón de fondo amarillo envía los comandos para el desplazamiento hacia la derecha. Estos desplazamientos se representan en la figura con las flechas con los colores correspondientes a cada botón.

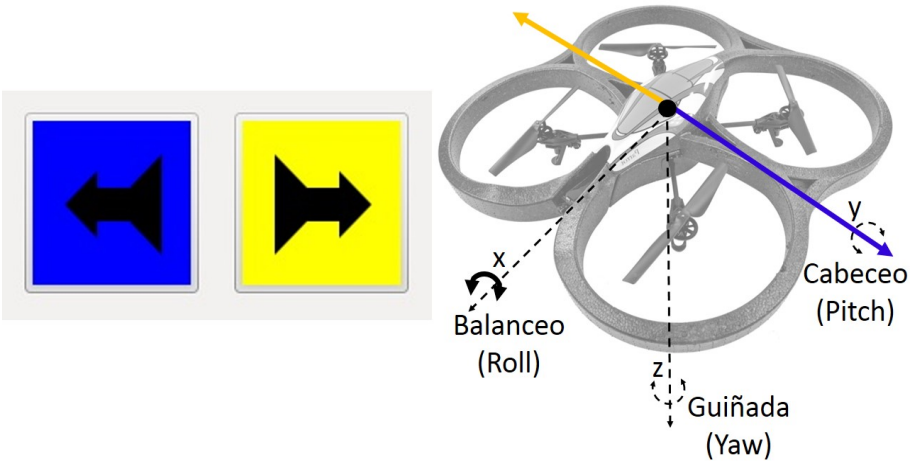


Figura 4.5: Botones para la traslación del cuatrimotor sobre el eje y hacia la derecha(azul) o izquierda(amarillo), basados en el giro de *balanceo*.

- Botones de giros: El último par de botones de esta sección (ver Figura 4.6), envía el comando al cuatrimotor para que gire sobre el eje z (guiñada). Con el botón de fondo blanco gira hacia la izquierda y el de fondo negro hacia la derecha.

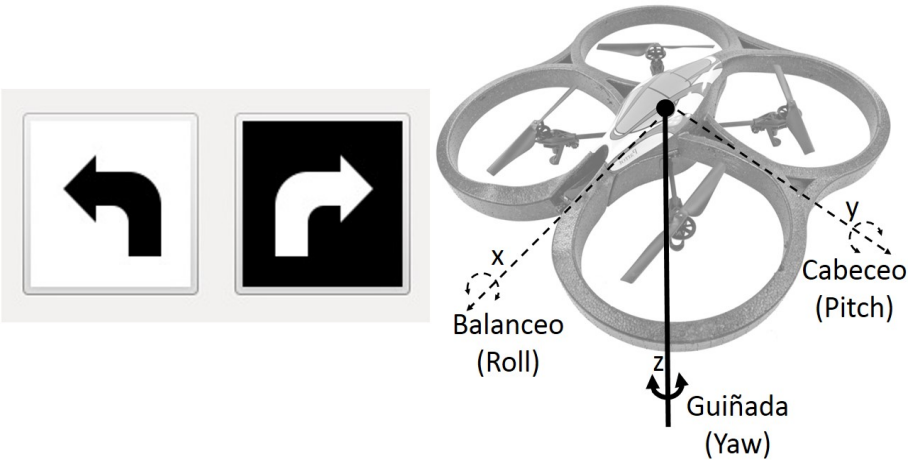


Figura 4.6: Botones para el giro a la izquierda (botón de fondo blanco) o a la derecha (botón de fondo negro), el giro es alrededor del eje z (guiñada) y cambia la orientación del cuatrimotor.

Sin embargo, los botones descritos solo tienen implementado un movimiento a una velocidad constante sin condición de paro. El botón *Alto* (ver Figura 4.2), tiene la función de que el cuatrimotor vuele de forma estacionaria, ésto lo logra al desactivar los desplazamientos en los ejes de referencia. El boton de *emergencia* mostrado en la Figura 4.7, envía el comando para que el cuatrimotor aterrice y reinicie su sistema.

Con el botón *Auto* se ejecuta una rutina de navegación definida para la exploración de seis cuadrantes, identifica objetos de interés con la cámara vertical y dibuja el mapa del ambiente explorado, dibujando los objetos identificados durante la exploración. El boton *MapeoFrontal* (véase la Figura 4.2), activa una secuencia de cuatro giros de 90° cada uno, para hacer identificación de objetos con la cámara frontal del cuatrimotor y dibuja los objetos encontrados en el área de la interfaz correspondiente al mapa.

Figura 4.7: Botón de emergencia, envía el comando para que el cuatrimotor aterrice y reinicie su sistema.

Se puede observar en la Figura 4.2, una sección donde se muestra de forma gráfica el porcentaje de carga de la batería. Debido a que la carga de la batería es de vital importancia para el buen funcionamiento del cuatrimotor y para evitar la reducción del tiempo de vida útil de la batería, se le implementó a ésta representación tres diferentes etiquetas en relación con la carga. La primera etiqueta es "*Batería*". Ésta etiqueta, esta activa mientras la carga de la batería se encuentra por encima del 35 %, cuando la carga baja de éste rango y hasta por encima del 25 % se activa la etiqueta "*Reemplace*" y por último, la etiqueta "*¡¡¡¡ ESTADO CRITICO !!!*" se activa cuando la carga esta por debajo del 25 %.

Por último en la Figura 4.8, el botón *Despegue* envía los comandos de locomoción de despegue al cuatrimotor, inicia el envío de las lecturas de los sensores y lo ubica a un metro de altura del suelo. El botón *Aterrizaje* hace que el cuatrimotor descienda (aterrice), suspenda el envío de las lecturas de los sensores y se ponga en estado de espera de la instrucción de despegue. Estos botones no encienden ni apagan el sistema del cuatrimotor, ésto solo se logra al conectar o desconectar (respectivamente) la batería. Cabe señalar que ambas funciones(despegue y aterrizaje) se encuentran definidas en el SDK del cuatrimotor, no se pueden modificar los parámetros de locomoción que envían al cuatrimotor.

Figura 4.8: Botones para el despegue y aterrizaje del cuatrimotor. El botón de despegue eleva al cuatrimotor a una altura de un metro e inicia el envío de las lecturas de los sensores. El botón de aterrizaje envía los comandos para que el cuatrimotor aterrice sin apagar el sistema.

4.2.2. Sección de recuperación de video

El cuatrimotor cuenta con dos cámaras, una frontal que visualiza el ambiente frente al cuatrimotor y una vertical que visualiza el ambiente que se encuentra debajo del cuatrimotor cuando éste ha despegado. El SDK recibe el video del cuatrimotor, pero para utilizar el video para reconocimiento de objetos es necesario transformar la imagen a la estructura de imagen de OpenCV, para ésto se utilizó la función de transformación que tiene implementada OpenCV; una vez que la identificación de objetos a concluido, el video debe ser transformado a la estructura de imagen de GTK para visualizar el video en la interfaz.

El SDK tiene implementado cuatro diferentes formas de visualizar el video de las cámaras del cuatrimotor. En la Figura 4.9, se muestra la imagen tomada por la cámara frontal, esta cámara toma video del ambiente que se encuentra frente al cuatrimotor.

Figura 4.9: Foto tomada con la cámara frontal del cuatrimotor.

Para visualizar el video que toma la cámara vertical, se debe modificar la configuración de visualización de las cámaras. La Figura 4.10 muestra la imagen tomada con la cámara vertical del ambiente que se encuentra debajo del área que el cuatrimotor sobrevuela.

Figura 4.10: Foto tomada con la cámara vertical del cuatrimotor.

Las otras dos opciones para visualizar el video tomado por las cámaras, son dos combinaciones diferentes de las imágenes tomadas por ambas cámaras visualizadas en una misma imagen. La imagen mostrada en la Figura 4.11 es una combinación de la imagen tomada por ambas cámaras, teniendo la imagen tomada por la cámara frontal como la que abarca el mayor espacio y la imagen que se obtiene de la cámara vertical visualizada en un recuadro en la parte superior izquierda.

Figura 4.11: Combinación de ambas cámaras para la visualización de frente y debajo del cuatrimotor, teniendo como principal plano la imagen tomada por la cámara frontal [17].

Por último, se puede visualizar la combinación de ambas cámaras teniendo como imagen principal la tomada por la cámara vertical y en la parte superior izquierda la imagen tomada por la cámara frontal, dando como resultado una imagen como la mostrada en la Figura 4.12.

Figura 4.12: Combinación de ambas cámaras para la visualización de la parte inferior y superior del cuatrimotor, teniendo como principal plano la imagen tomada por la cámara vertical.

Debido a que se se dibujan dos tipos de mapas diferentes, primero un mapa para los objetos identificados con la cámara frontal y un mapa para los objetos identificados en el ambiente de exploración con la cámara vertical. Es necesario especificar para cada exploración la cámara a utilizar para la detección de objetos. Para la exploración del ambiente, se requiere visualizar el ambiente que se encuentra debajo del cuatrimotor, para éste caso, la cámara que se utiliza es la vertical. En el mapa donde se dibujan los objetos detectados frente al cuatrimotor en intervalos de giros de noventa grados, se configura para que la cámara utilizada para la identificación de objetos en el ambiente sea la frontal.

4.2.3. Sección de monitoreo de sensores

Para poder monitorear el estado de vuelo del cuatrimotor, se debe poder visualizar las lecturas de los sensores con que cuenta. Para la interpretación de las lecturas, es necesario definir un sistema de referencia que sirva para establecer el estado de vuelo del cuatrimotor en base a las lecturas de los sensores. El cuatrimotor define su propio sistema de referencia en el que se basa para calcular la distancia recorrida durante la navegación; la Figura 4.13 muestra el

eje cartesiano utlizado por el cuatrimotor para monitorear el estado de vuelo. Cuando el cuatrimotor enciende su sistema, se coloca al cuatrimotor en el origen del sistema de coordenadas, es decir, el cuatrimotor se coloca en el lugar donde los tres ejes cartesianos se interceptan.

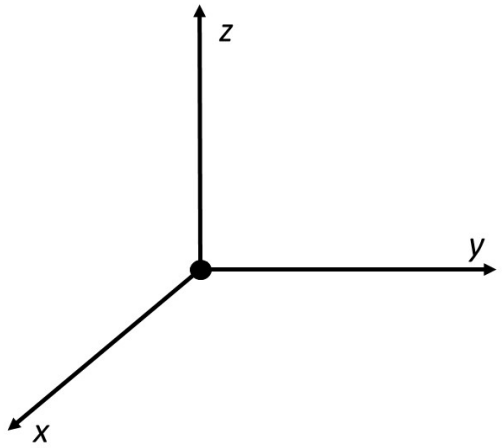


Figura 4.13: Sistema de referencia de vuelo del cuatrimotor (3D).

La Figura 4.14 muestra la sección de monitoreo de la interfaz, donde se pueden observar los datos de posición, velocidad y orientación, datos de gran importancia para verificar el estado de la exploración del cuatrimotor. A continuación se explica brevemente la interpretación de los datos:

- Posición: Muestra la posición en relación a los tres ejes del plano cartesiano del sistema de referencia. Las distancias estan dadas en metros, medidos a partir del punto de origen de las coordenadas y el punto de origen corresponde al punto donde el sistema del cuatrimotor es encendido. Las mediciones pueden ser de signo positivo o negativo, cada uno corresponde a una dirección de desplazamiento a partir del punto de origen de las coordenadas del sistema de referencia. Cuando la magnitud es positiva el desplazamiento fue hacia adelante,hacia la derecha o hacia arriba, según la

	x	y	z
Posición (m)	0,02	-0,03	0,12
Velocidad (m/s)	-0,10	-0,08	-0,04
Orientación (grados)	355,92	4,60	38,54

Figura 4.14: Sección donde se visualizan los datos recuperados por los sensores del cuatrimotor, donde se observan los datos correspondientes a la localización, velocidades y orientación con respecto a los tres ejes del plano cartesiano, medidos a partir del origen definido al momento de encender el sistema del cuatrimotor.

coordenada que represente x, y o z respectivamente. Y cuando es negativa el desplazmiento fue en sentido contrario según cada caso.

- Velocidad: El cuatrimotor utiliza los acelerómetros con que cuenta para calcular la velocidad de desplazamiento en cada uno de los ejes. Dependiendo el signo de la velocidad se refiere el sentido del desplazamiento. Si la velocidad es positiva en el eje x el movimiento es hacia el frente y negativa en sentido contrario. Si la velocidad es positiva en el eje y, el movimiento es hacia la derecha y negativa si es hacia la izquierda. Si la velocidad del cuatrimotor le permite elevarse la velocidad en el eje z es positiva y si desciende es negativa.

- Orientación: La orientación también es inicializada al encender el sistema del cuatrimotor. A partir de la orientación de inicio, las lecturas de los sensores se miden en sentido contrario a las manecillas del reloj desde una perspectiva de frente de los tres ejes del plano cartesiano.

4.2.4. Sección de mapa

Para el desarrollo o implementación de esta sección, se utilizó la biblioteca *GTK-Cairo* que forma parte de GTK. GTK-Cairo, es la biblioteca que brinda las herramientas necesarias para poder dibujar líneas y áreas utilizando diferentes colores. La función de dibujo del mapa, crea una área con fondo blanco (como se puede observar en la Figura 4.15), sobre la cual se van a pintar los objetos

que sean detectados en el ambiente. Se implementaron funciones de dibujo para cada una de las figuras que se van a reconocer (círculo, cuadrado, rectángulo y triángulo). Cuando se detecte alguna figura, se activará la función especifica de dibujo para el objeto encontrado. Es importante decir que fue necesario implementar una función de conversión que determinará en funcion de la altura y la posición del cuatrimotor en que posición será dibujado el objeto.

En el área del mapa, se dibujará el resultado de la exploración del ambiente, los objetos dibujados en el mapa deberán corresponder en ubicación, cantidad y tipo de objeto, al área explorada con la cámara vertical.

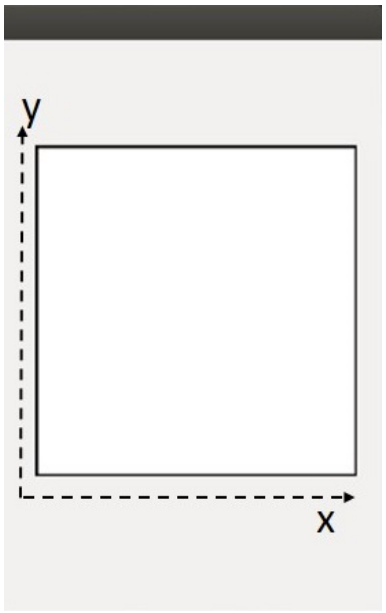


Figura 4.15: Sección donde se dibujará el mapa resultado de la secuencia de mapeo con el cuatrimotor.

4.3. Funciones de exploración del cuatrimotor

Para que el cuatrimotor sea capaz de explorar un ambiente, es necesario utilizar una serie de elementos que le ayuden. El cuatrimotor cuenta con cuatro rotores que además de darle el nombre a esta arquitectura, son los elementos que le dan la capacidad para volar y llevar a cabo la navegación en un ambiente.

Estos se encuentran unidos a los extremos de una base en forma de cruz [54], en donde también se encuentra la batería y el hardware que integran el cuatrimotor (ver Figura 4.16).

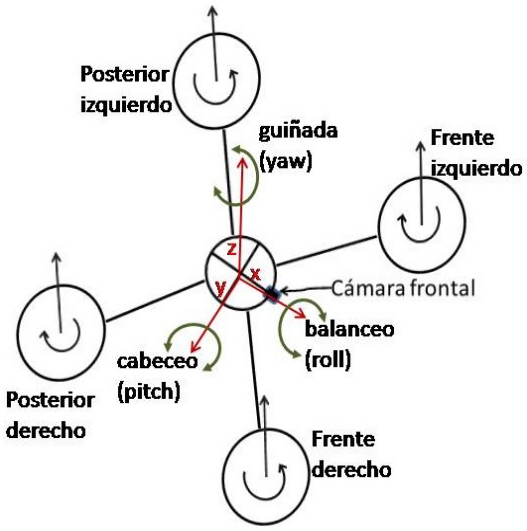


Figura 4.16: Rotación de rotores en sentido contrario en pares opuestos; posicionamiento de los ejes cartesianos con respecto al cuatrimotor [54].

Alternando las velocidades en los rotores, el cuatrimotor puede navegar en los tres ejes para explorar el ambiente; los vehículos aéreos pueden rotar en los tres ejes dimensionales para alterar su dirección [55], como se puede apreciar en la Figura 4.17.

Estos movimientos se pueden describir:

- *Yaw Axis* (eje de guiñada): es perpendicular al plano de las alas (en su caso). Un movimiento de guiñada es un movimiento del frente de la aeronave de lado a lado.

- *Pitch Axis* (eje de cabeceo): es perpendicular al eje de guiñada y es paralelo al plano de las alas con su origen. Un movimiento de cabeceo es un movimiento hacia arriba o hacia abajo del frente de la aeronave.

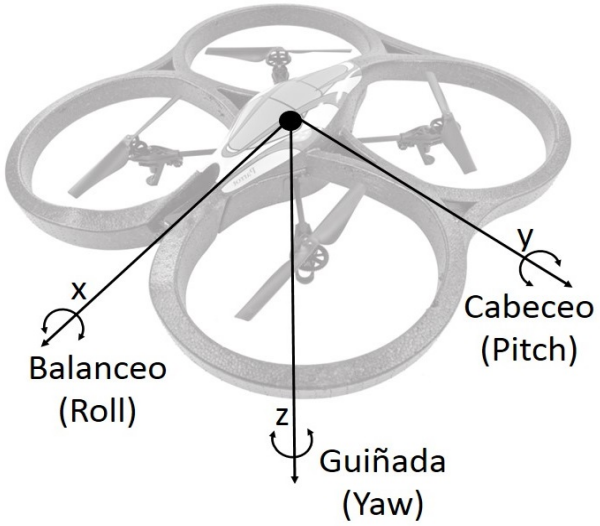


Figura 4.17: Rotaciones del cuatrimotor [54]: Guiñada (Yaw) al rededor del eje Z, cabeceo (Pitch) al rededor del eje Y y balanceo (Roll) al rededor del eje X.

- *Roll Axis* (eje de alabeo o balanceo): es perpendicular a los otros dos ejes. Un movimiento de balanceo es un movimiento hacia arriba y abajo de los costados de la aeronave.

4.3.1. Aproximación de la localización de los objetos

Debido a que el cálculo de la posición de los objetos identificados se obtiene del video, es necesario calcular el área que cubre la cámara vertical en relación con la altura a la cual se encuentre volando en el momento que identifica algún objeto. Para esto, se tomó una muestra de imágenes tomadas con la cámara vertical. Para cada imagen se determinó el área que cubre la cámara en conjunto con la altura a la que fueron tomadas. Para esto, se utilizó el método matemático de *regresión lineal* [56]. El método modela la relación de una variable dependiente en relación con una variable independiente. Es decir, que dos variables x, y están relacionadas, eso quiere decir que al conocer el valor de x, entonces el valor de y puede ser calculado. La relación lineal de dos variables se representa con la *fórmula* 4.1 [56].

$$y = \beta_1 x + \beta_0 \tag{4.1}$$

donde:

β_0 = es la ordenada al origen.
β_1 = es la pendiente de la recta.
x = variable independiente.
y = variable dependiente.

Para determinar el valor de β_0 y β_1 se utilizan las *fórmulas* 4.2 y 4.3.

$$\beta_1 = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n\sum x_i^2 - (\sum x_i)^2} \tag{4.2}$$

$$\beta_0 = \bar{y} - \beta_1 \bar{x} \tag{4.3}$$

Los resultados de la muestra de las imágenes tomadas con la cámara vertical, son presentados en la gráfica de la Figura 4.18, donde se puede observar la distancia que cubren el alto y ancho de la cámara en relación con la altura a la que fueron tomadas cada una de las imágenes.

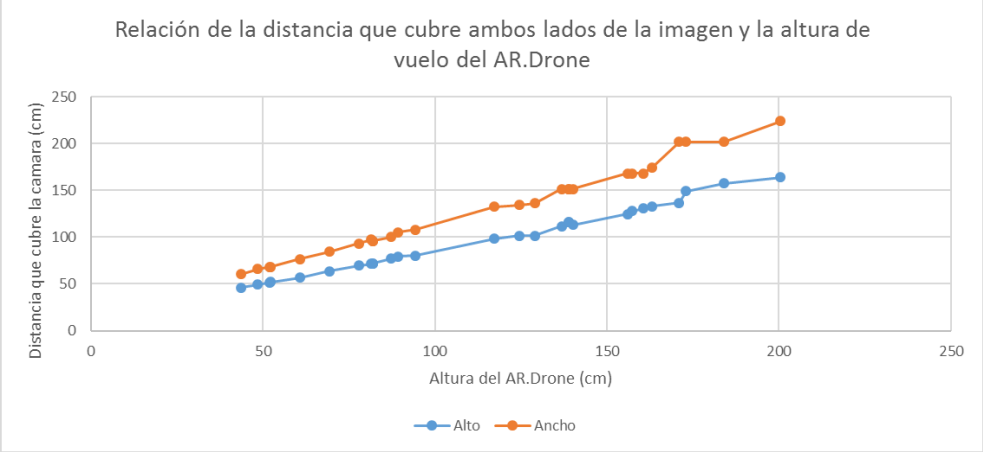


Figura 4.18: Relación de la distancia que cubren el alto y ancho de la cámara vertical a una determinada altura de vuelo del cuatrimotor.

Utilizando los datos de la gráfica de la Figura 4.18 y empleando las fórmulas 4.2 y 4.3, se obtienen las fórmulas de la relación entre la distancia de los lados de la imagen y la altura. La gráfica de la Figura 4.19, muestra la relación que existe entre la distancia que cubre el alto de la imagen con respecto a la altura

a la que se encontraba el cuatrimotor al momento de tomar la foto; se puede observar que la tendencia es lineal con algunas variaciones que son causadas por la inestabilidad detectada en el cuatrimotor que afectaron el ángulo de orientación de la cámara. La linea punteada dentro de la misma gráfica, representa los datos estimados por la fórmula 4.4, producto de la regresión lineal aplicada a los datos de la muestra.

$$y = 0{,}753x + 10{,}78 \tag{4.4}$$

donde:

x = altura de vuelo del cuatrimotor (cm).
y = distancia que cubre el alto de la cámara(cm).

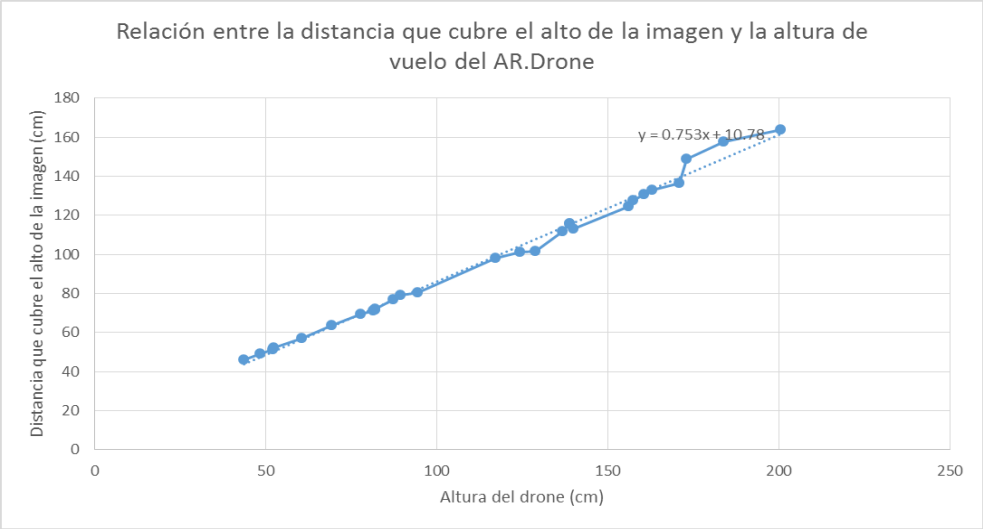


Figura 4.19: Gráfica para visualizar la relación que existe entre la distancia que cubre el alto de la imagen con respecto a la altura de vuelo del cuatrimotor. La relación calculada con la fórmula producto de la regresión lineal, esta representada en línea punteada.

En la gráfica de la Figura 4.20, se muestra la relación que existe entre la distancia que cubre el ancho de la imagen con respecto a la altura a la que se encontraba el cuatrimotor al momento de tomar la muestra; también en este caso la inestabilidad de vuelo del cuatrimotor afectó las mediciones, ésto es evidente en las variaciones que se observan de la tendencia lineal. La linea

punteada dentro de la gráfica representa los datos estimados por la fórmula 4.5, producto de la regresión lineal aplicada a los datos de la muestra.

$$y = 1{,}0167x + 13{,}203 \tag{4.5}$$

donde:

x = altura de vuelo del cuatrimotor (cm).
y = distancia que cubre el ancho de la cámara(cm).

Figura 4.20: Gráfica para visualizar la relación que existe entre la distancia que cubre el ancho de la imagen y la altura de vuelo del cuatrimotor. La relación calculada con la fórmula resultado de la regresión lineal, esta representada por la línea punteada.

Con las fórmulas calculadas con la regresión lineal, se puede determinar la distancia que cubre la imagen de la cámara vertical a una determinada altura, cada una aplicada especificamente a un lado de la imagen que toma la cámara vertical y basadas en la altura de vuelo del cuatrimotor, como se representa en la Figura 4.21.

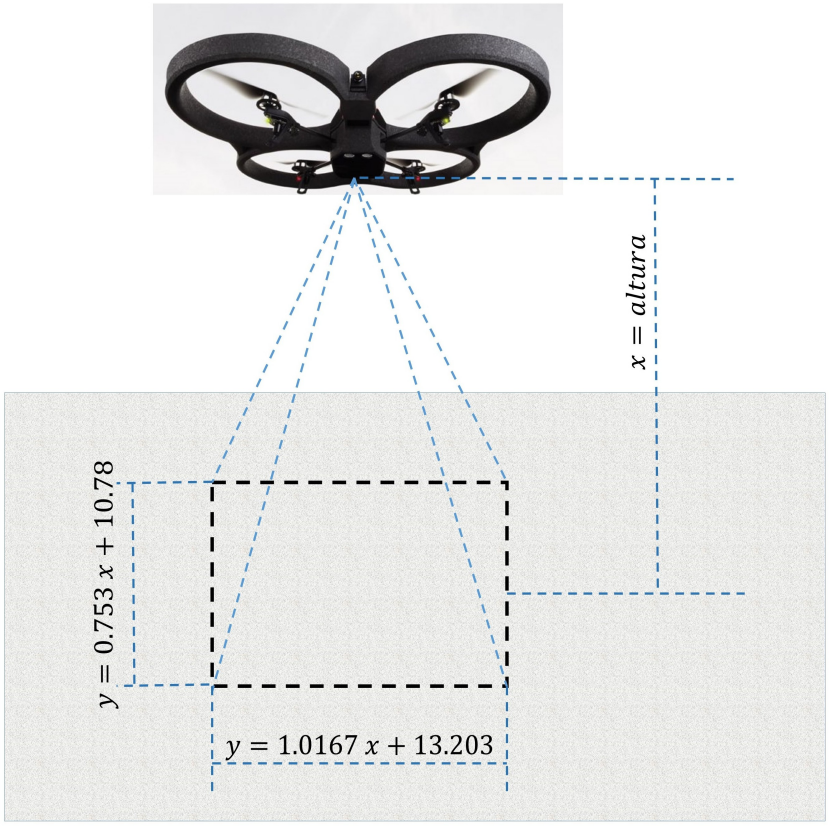


Figura 4.21: Representación de la aplicación de las fórmulas para calcular la distancia que cubren los lados de la imagen tomada por la cámara vertical, ambas fórmulas utilizan la altura de vuelo del cuatrimotor para hacer el cálculo.

4.3.2. Área de exploración

Para poder hacer exploración de un ambiente, es necesario tener en cuenta el área a explorar y la rutina de navegación que cubrira el ambiente objetivo. Después de pruebas de navegación, se determinó que los datos de posición y velocidad que el SDK recibe del cuatrimotor presentan discrepancias, por lo que no pueden ser utilizados para determinar la localización del cuatrimotor en el ambiente durante la exploración.

Para determinar las dimensiones del área a explorar y la rutina de navegación, se hicieron pruebas de vuelo para determinar la distancia recorrida en un determinado tiempo para parámetros de velocidades. Basado en las pruebas de vuelo, se determinó un área de exploración de $24{,}12m^2$, con dimensiones de

$6m$ por 4,02m. Ésta área es cubierta a una altura de 184cm.

En la Figura 4.22 se puede representar la secuencia de navegación implementadas en el cuatrimotor. Cada cuadrante equivale al área que cubre la cámara a la altura definida, recorre hacia el frente 4,5m y la cámara cubre un área de $6m$ por 2,01m, para moverse hacia la siguiente sección de cuadrantes, el cuatrimotor vuela hacia la derecha durante $2s$ que equivale a un desplazamiento de 2,01m. Por último, el cuatrimotor se desplaza hacia atras durante $3s$ el mismo lapso de tiempo que el movimiento hacia adelante.

Para que la estimación de la localización de los objetos sea lo mas apegado al ambiente explorado, se debe activar la identificación de los objetos en los espacios donde no se hayan tomado imagen para la identificación. Por esta razón, para la secuencia de navegación se consideró el área que la cámara cubre del ambiente (véase Figura 4.21) y en base a ésto, cada cuadrante corresponde a un espacio del ambiente que es capturado en imagen por la cámara una única vez y como la cámara esta en la posición para tomar un cuadrante cada segundo de vuelo a la velocidad establecida, la identificación de objetos se activa en este intervalo de tiempo.

En la Figura 4.23 se muestran los pasos a seguir para poder bibujar el mapa, a continuación se describe:

1. Para verificar la identificación por medio del video que se visualiza en la interfaz, se implementó un enmarcado de los objetos identificados con diferentes colores para cada tipo de objeto. También se etiquetan con el nombre del objeto y la cantidad identificados del mismo tipo. Cuando se detecte algún objeto, en el video de la interfaz el objeto se mostrará enmarcado en un cuadro de un color específico para el tipo de objeto.

2. Se implementó una estructura de datos para guardar los datos de la identificación como las coordenadas (pixeles donde se localiza el objeto en la imagen), tipo de objeto detectado y el cuadrante donde esta navegando el cuatrimotor al momento de identificar algún objeto.

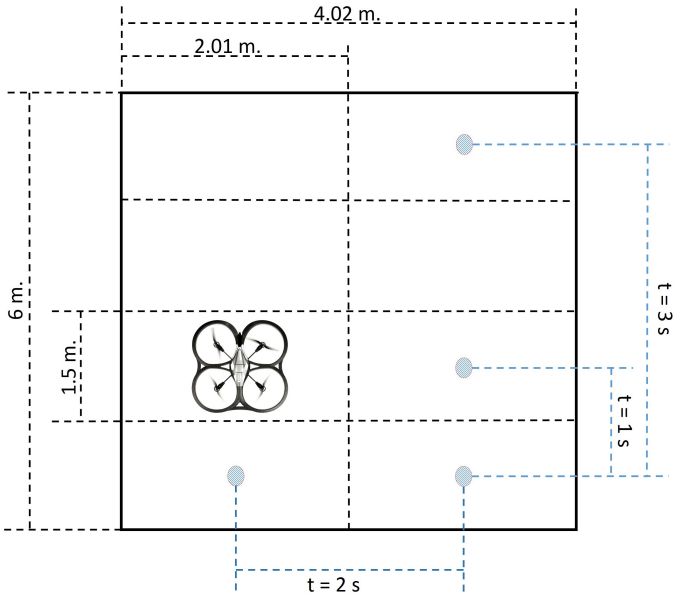


Figura 4.22: Representación de la relación del mapa con tiempo y área recorrida por el cuatrimotor.

3. Apoyado en un factor de desplazamiento para el cuadrante donde se detectó el objeto, se calculan las coordenadas donde se detectó el objeto en relación al ambiente total explorado.

4. Nuevamente con el apoyo de una relación de proporcionalidad se convierten las coordenadas de localización del objeto en el ambiente a coordenadas de localización pero en el mapa, es decir, convertir las coordenadas en centímetros a coordenadas en píxeles.

5. Finalmente se dibuja el objeto en el área del mapa, en las coordenadas calculadas y el tipo de objeto detectado.

4.3.3. Rutina de vuelo alterna

El giro del cuatrimotor durante el vuelo es de vital importancia para la navegación, por esto, es importante validar por medio de un método experimental que el cuatrimotor gire sobre su propio eje para cambiar la dirección de vuelo.

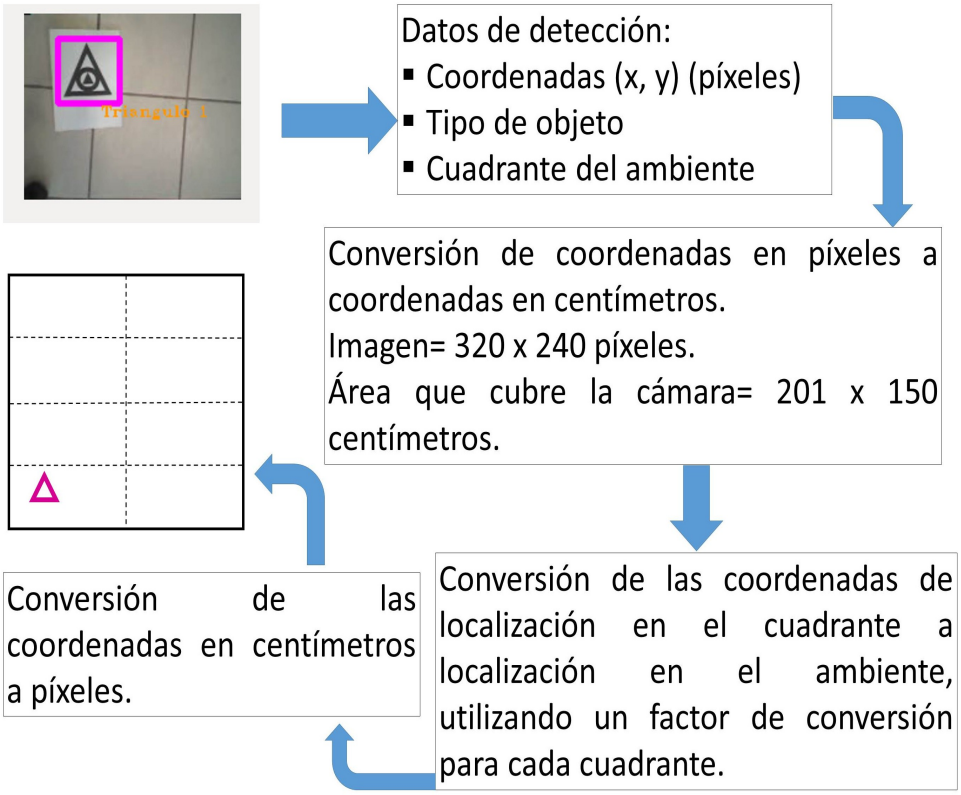


Figura 4.23: Metodología para la generación de mapa basandose en el reconocimiento de objetos en un ambiente controlado explorado por un cuatrimotor.

Para validar el giro, se implemento una secuencia de vuelo que consiste en colocar al cuatrimotor en cuatro diferentes orientaciones en un rango de noventa grados de diferencia uno de otro, en sentido contra reloj y medidos a partir del punto de origen del sistema de referencia establecido al momento de encender el cuatrimotor.

Para el pintado del mapa, resultado de la identificación de objetos con la cámara frontal del cuatrimotor, se dividió el área de pintado de mapa en cuatro cuadrantes, como se puede observar en la Figura 4.25. El SDK registra como orientación inicial del cuatrimotor la orientación que tiene cuando se enciende su sistema. A partir de éste punto, el cuatrimotor gira en sentido contrario a las manecillas del reloj en rangos de movimiento de 90°, cuando se encuentre en la orientación requerida, la detección de objetos se activa.

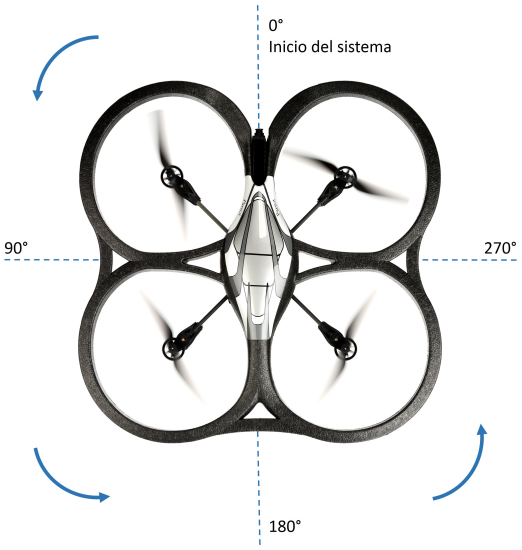


Figura 4.24: El cuatrimotor tiene implementado una secuencia de tres giros en intervalos de 90° en sentido contra reloj.

El cuatrimotor no cuenta con sensores de proximidad que le indiquen la distancia a la que se encuentra un objeto que se encuentra al frente, por esta razón, solo se puede estimar la localización relativa de los objetos en el ambiente, por medio de una relación de proporcionalidad entre el área de la imagen donde fue la detección y el área de dibujo del mapa.

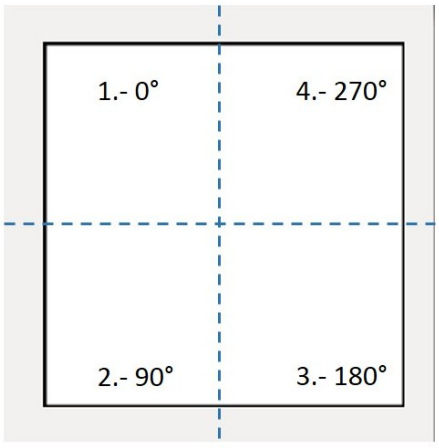


Figura 4.25: Representación de los cuadrantes en que es dividida el área de pintado de mapa para dibujar los objetos identificados con la cámara frontal del cuatrimotor.

Como se aprecia en la Figura 4.25, en el primer cuadrante se dibujarán los objetos encontrados cuando el cuatrimotor se encuentra en una orientación de 0°. En el segundo cuadrante se dibujan los objetos encontrados en el área de los 90°. Para el tercer y cuarto cuadrante se dibujarán los objetos encontrados en la orientación de 180° y 270° respectivamente. Si al iniciar la secuencia el cuatrimotor no se encuentra con una orientación de 0°, el primer paso de la secuencia de navegación, será el posicionar al cuatrimotor con esta orientación inicial.

Capítulo 5

Resultados

A continuación se presentan los resultados de la implementación de la identificación de objetos con el cuatrimotor. La navegación solo se implementó en la exploración con la cámara frontal, debido a problemas técnicos no se completó la exploración del ambiente con la cámara vertical, sin embargo el mapa se dibujó según lo esperado con las identificaciones de objetos.

5.1. Detección de objetos de interés

Con los identificadores que se obtienen del algoritmo de Viola-Jones, se llevaron a cabo un conjunto de pruebas para verificar la exactitud de la identificación. De la prueba de reconocimiento de los cuatro objetos al mismo tiempo, se tomaron cien muestras de identificación con la cámara del cuatrimotor, de los que se obtuvieron los siguientes resultados:

- Círculo: 84 detecciones.
- Cuadrado: 93 detecciones.
- Rectángulo: 82 detecciones.
- Triángulo: 83 detecciones.

El histograma de la Figura 5.1 muestra la frecuencia de detección para cada uno de los objetos. Comparando estos resultados con los de la Tabla 3.2, se puede observar que los resultados del rectángulo y triángulo son los que muestran mayor discrepancias entre la evaluación con OpenCV y con el cuatrimotor.

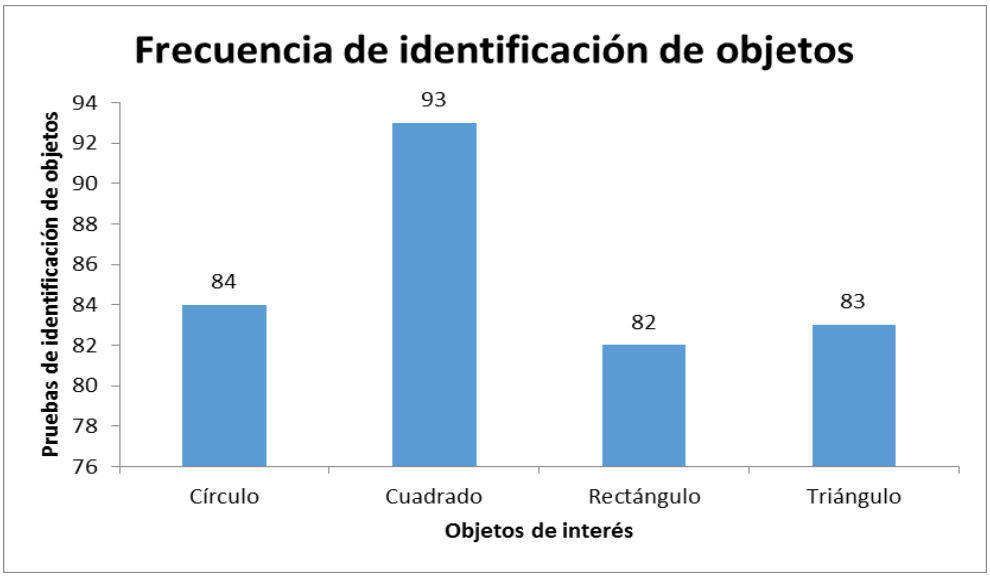


Figura 5.1: Gráfica de detecciones realizadas con la cámara del cuatrimotor.

La Figura 5.2 muestra la detección hecha por el cuatrimotor. La variación entre los resultados del evaluador del OpenCV y la deteccción con el cuatrimotor, se debe principalmente a las variaciones del ángulo del cuatrimotor con respecto a los objetos. Las variaciones de iluminación influyó de igual manera en la exactitud de la identificación, ésto fue evidente al momento de activar la identificación cuando el ambiente se encontraba con poca iluminación, siendo detectados una menor cantidad de objetos en comparación cuando el ambiente se encontraba con mayor iluminación.

La exactitud de la deteccción se ve afectada cuando la distancia entre el cuatrimotor y los objetos varía de forma rapida durante el vuelo. Si la cámara que esta activa para la detección es la vertical, entonces afecta que el cuatrimotor vuele a una altura variable, esto es, que el cuatrimotor no logre volar a una altura constante que permita que el tamaño de los objetos sea lo más constante posible en la imagen tomada.

En el caso de la cámara frontal, se implementó un conjunto de giros estacionarios con deteccción de objetos, pero como el cuatrimotor no logró volar sin desplazarse sobre los ejes de referencia, entonces la identificación presento un

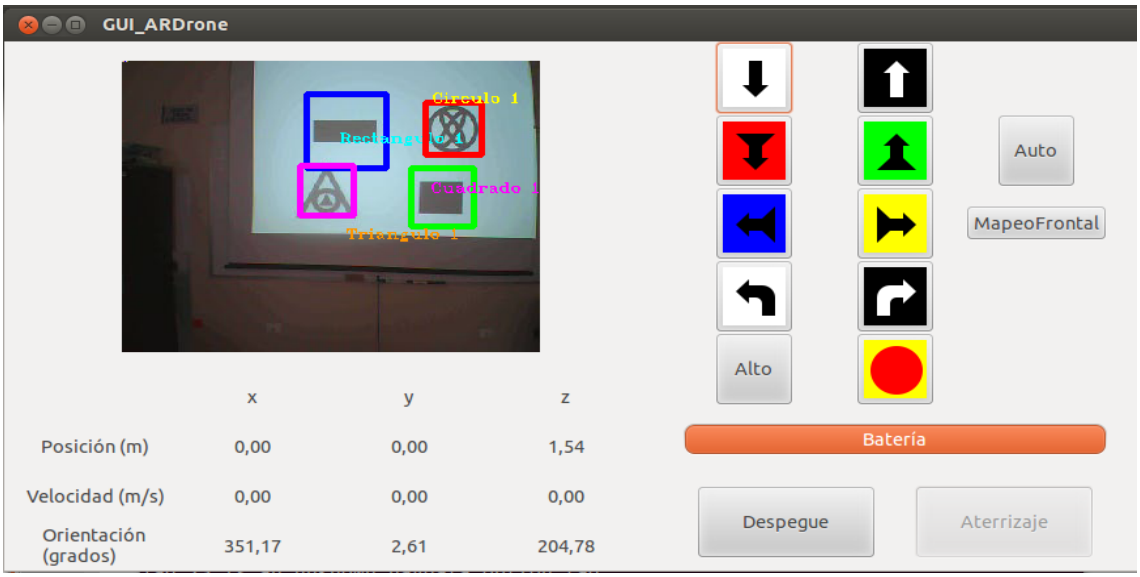


Figura 5.2: Detección de cuatro objetos de interés con la cámara del cuatrimotor.

menor número de deteccciones en relación a los objetos presentes en el ambiente explorado, debido a que el ángulo de la cámara con respecto a los objetos y la distancia entre ellos variaban afectando la detección.

5.2. Interfaz

La interfaz gráfica de usuario que se muestra en la Figura 5.3, lleva a cabo las funciones que le fueron implementadas. Los botones de la sección de comandos de navegación facilitan al usuario la operación del cuatrimotor, sin que tenga que revisar las especificaciones técnicas de vuelo del cuatrimotor, siendo transparente para el usuario el envío de los comandos de locomoción, y de forma intuitiva el usuario identifica la funcionalidad de cada botón, esto principalmente a que es un conjunto de botones ampliamente utilizado entre otras cosas en mandos de control de grúas, de maquinaría pesada y la mayoría de equipos que tienen algún elemento operable que se mueva a lo largo de dos o tres ejes. Los botones son funcionales, éstos envían los comandos de locomoción que le fueron implementadas.

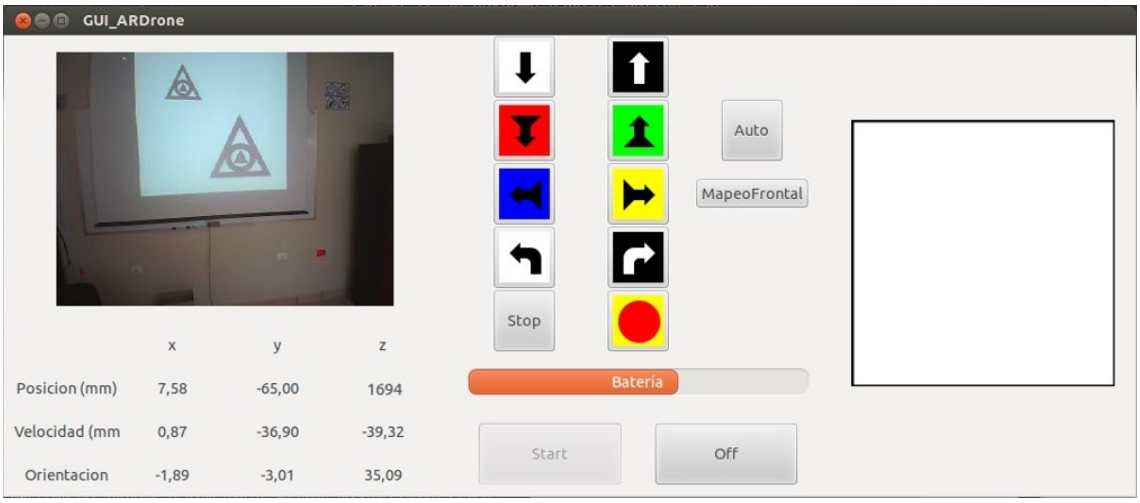


Figura 5.3: Interfaz gráfica de usuario para la operación y monitoreo del cuatrimotor.

La sección de video muestra el video enviado por las cámaras, cuando la cámara que se visualiza es cambiada, la interfaz muestra el video de la cámara seleccionada. Solo se presenta un retardo en la visualización cuando la identificación de todos los objetos se encuentra activa, esto debido al tiempo que le toma a los clasificadores detectar y dibujar los cuadros y etiquetas sobre los objetos detectados, como se observa en la Figura 5.4.

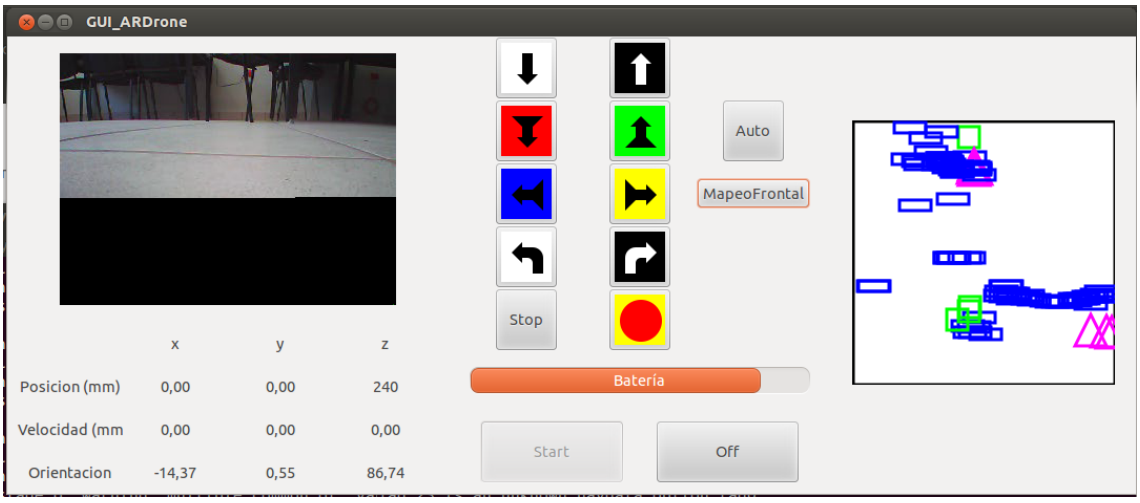


Figura 5.4: Interfaz gráfica de usuario, se presenta retardo en la visualización del video debido a que la detección de los objetos se encuentra activa.

En la sección de monitoreo, la información enviada por el cuatrimotor al SDK se visualiza en tiempo real, se le da formato a la información previamente a ser visualizada, para que pueda ser interpretada por el usuario y cumpla con el objetivo de ser útil para mantener un constante monitoreo de las condiciones del vuelo del cuatrimotor. Aunque la información no es fidedigna debido a que se detectó que la información del desplazamiento del cuatrimotor durante el vuelo no es la que el cuatrimotor envía; pero esto corresponde a la sección de navegación. Finalmente, la sección de mapa dibuja los objetos identificados y borra el mapa si se activa nuevamente una secuencia de exploración, pero se abordará a mayor detalle en los resultados de la navegación.

5.3. Navegación

Para realizar Los valores de los parámetros que manipulan la navegación, se tuvieron que ajustar para asegurar un movimiento uniforme, es decir, durante el movimiento de desplazamiento del cuatrimotor las orientaciones y velocidades en los tres ejes de referencia deben mantenerse sin modificaciones. El principal movimiento que se ve afectado por la velocidad de locomoción es el de los giros sobre el eje z (guiñada) Figura 5.5; se observó que a una velocidad baja el drone presenta gran inestabilidad para realizar el giro en cualquier sentido, pero también se observó que sin importar la velocidad al momento de realizar el giro se percibe una gran distancia de recorrido.

Otro movimiento que se ve bastante afectado por una baja velocidad es el que manipula la elevación o descenso del cuatrimotor. En este caso, se observó que una baja velocidad vertical para realizar el ascenso, no ejecuta ningún desplazamiento sobre el eje z, a través de un proceso experimental se determinó que el valor mínimo que debe ser enviado como parámetro a la función de vuelo para que el drone realice el desplazamiento hacia arriba o hacia abajo es el de 0.3 ó -0.3, según sea el caso, valor positivo para ascenso o valor negativo si se requiere descender.

Para los desplazamientos sobre los ejes x e y, mediante un método experimental se hicieron pruebas de vuelo en ambos ejes para determinar las velocidades

que en base al área que cubre la cámara a una cierta altura pudiera realizar un recorrido cubriendo el ambiente sin dejar áreas sin hacer reconocimiento de objetos. De la serie de recorridos, se eligió la que se desplazaba a una distancia con la que se puede llevar a cabo el reconocimiento sin dejar áreas sin explorar.

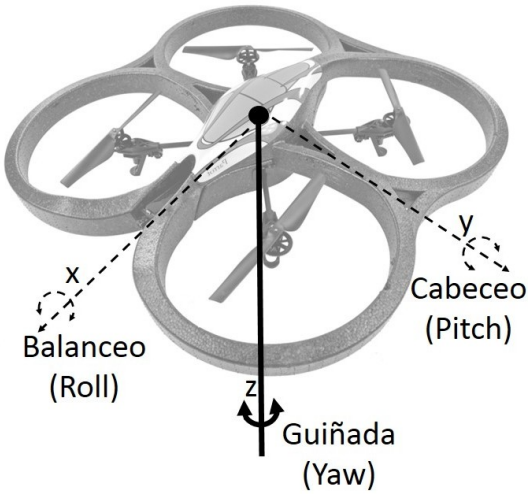


Figura 5.5: Giro a la izquierda o a la derecha sobre el eje z (guiñada) para cambiar la orientación del cuatrimotor.

Con la velocidad y tiempo elegidos se hicieron una serie de pruebas para determinar la estabilidad de este recorrido para implementar posteriormente la exploración. Se realizaron dos conjuntos de cien recorridos para determinar la distancia recorrida en promedio y el tiempo de navegación, de desplazamiento frontal y lateral.

En la Tabla 5.1, se pueden ver los resultados de los recorridos donde se observa que para la mediana, moda y promedio la distancia recorrida en los tres casos fue de 450 cm en un tiempo de recorrido de 3 segundos. Junto con estos datos se tiene que en el 48 % de los casos el recorrido fue de 450 cm, en el 38 % fue en un rango que discrepó hasta en 10 cm bien sea arriba de los 450 cm o por debajo y en un 14 % la discrepancia fue entre 11 y 24 cm por encima o por debajo de los 450 cm. En la Figura 5.6, se puede observar de forma gráfica los resultados de los recorridos experimentales.

Tabla 5.1: Analisis de resultados de las pruebas de navegación frontal.

	Resultados
mediana	450cm
moda	450cm
promedio	450.24cm
Rango de distancia	**Resultados**
450cm	***48 %***
± 10cm	38 %
± 11cm - ± 24cm	14 %

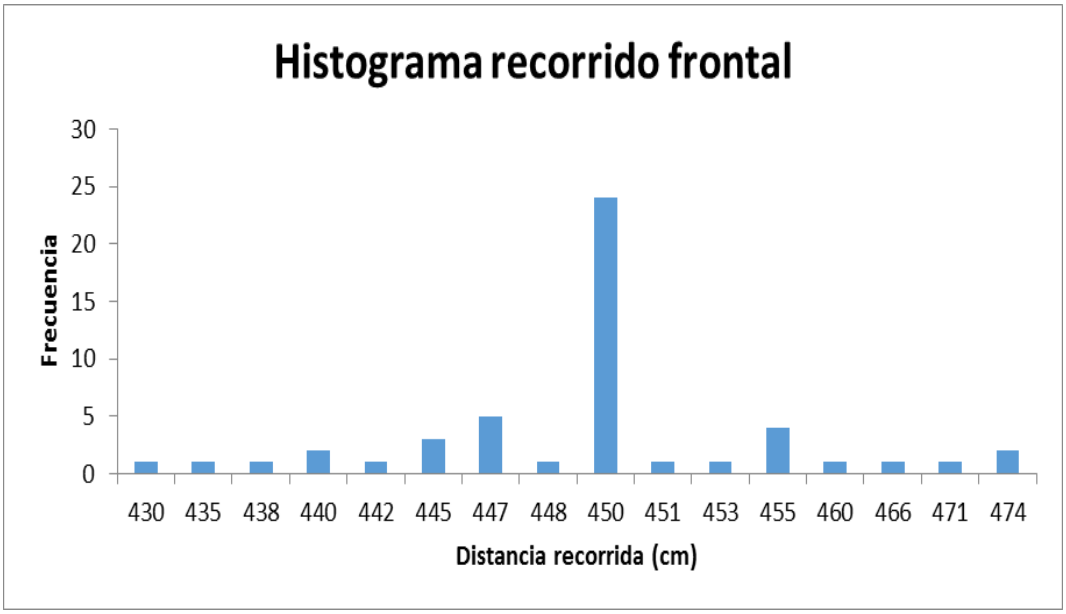


Figura 5.6: Resultados de las pruebas de recorrido frontal del cuatrimotor.

Los resultados del recorrido experimental lateral se muestran en la Tabla 5.2, en este conjunto de pruebas, la media y moda fue de 201 cm y el promedio de 203 cm, estos recorridos fueron realizados en un periodo de 2 segundos. Donde, el 54 % de los casos fue un recorrido de 201 cm, un 16 % un recorrido de hasta 11 cm por debajo de los 201 cm y un 30 % un recorrido de hasta 19 cm por encima. En la Figura 5.7, se puede observar el histograma de los resultados de navegación del cuatrimotor.

Tabla 5.2: Analisis de resultados de las pruebas de navegación lateral.

	Resultados
mediana	201cm
moda	201cm
promedio	203.38cm
Rango de distancia	**Resultados**
-11cm	16 %
201cm	***54 %***
+19cm	30 %

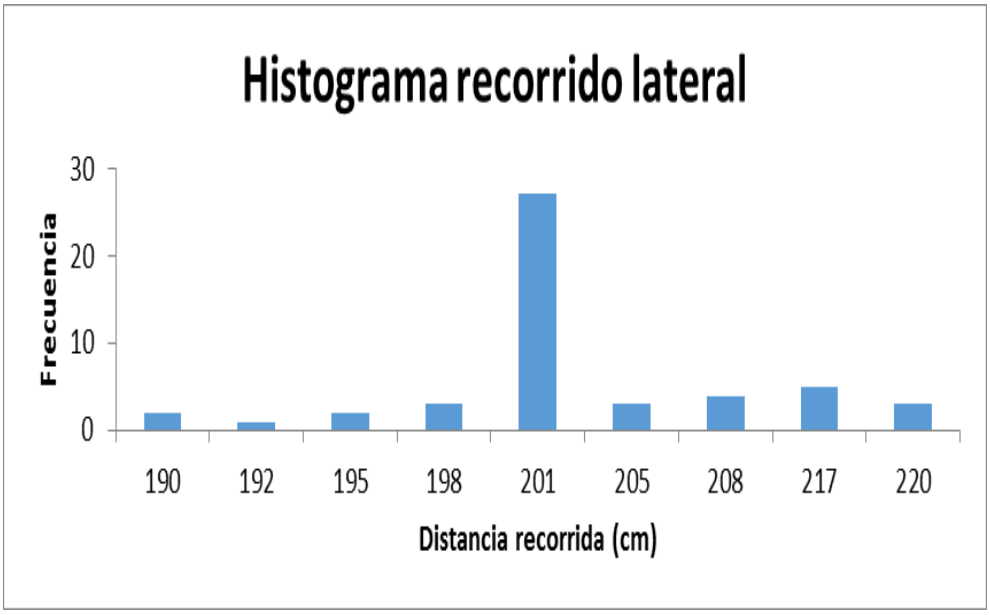


Figura 5.7: Resultados de la prueba de recorrido lateral del cuatrimotor.

Al implementar la exploración con las velocidades y tiempos resultados del método experimental se detectó gran inestabilidad al observarse un movimiento no rectilíneo a baja velocidad, es decir, las orientaciones en los tres ejes de referencia del cuatrimotor presentan variaciones, lo que causa que el movimiento del cuatrimotor no pueda ser pre-establecido. Aún a velocidades altas, se presenta un movimiento sin estabilidad en la dirección de desplazamiento, aunque menor que a velocidades bajas pero, la operación del cuatrimotor en su velocidad maxima es muy poco fiable, debido a que se dificulta el cambio de velocidad o dirección, esto es, que el frenado o cambio de dirección es muy

complejo tomando en cuenta que el aire presenta poca resistencia para realizar estos cambios bruscos en la velocidad o dirección.

Para el caso del vuelo estacionario (el cuatrimotor solo debe mantenerse a una altura de vuelo estable sin desplazamiento), se detectó una seria inestabilidad en el mismo, debido a que al momento de enviarle el comando para este tipo de vuelo no logra mantenerse en el aire en el lugar, el vuelo se vuelve inestable y tiene un desplazamiento errático, es decir, las velocidades y orientaciones de los tres ejes presentan variaciones que impiden hacer una estimación de la localización del robot.

Debido a las variaciones en la navegación del cuatrimotor, se buscaron referentes de otros trabajos relacionados. En ellos, se han utilizado algoritmos especializados de control de navegación robótica, como el realizado por Nick Dijkshoorn y Arnoud Visser al implementar un filtro extendido de kalman (**EKF** del inglés, *Extended Kalman Filter*) en el cuatrimotor [57]; las lecturas de los sensores son utilizados por el EKF para estimar la posición del cuatrimotor.

En la sección de pintado del mapa, se dibujan los objetos según las coordenadas y tipo de objeto que se le especifica. Para que el mapa realmente represente el área explorada, depende de una buena identificación y que la velocidad del cuatrimotor sea uniforme, debido a que en relación al tiempo de recorrido, se determina el cuadrante de la detección y éste dato es fundamental para calcular las coordenadas donde se dibuja el objeto en el mapa. El mapa fue probado con simulacion de la navegación y con la identificación de objetos.

En las pruebas preliminares de la rutina de navegación el área a explorar, fue cubierta con algunas discrepancias de vuelo propias del cuatrimotor. Pero la navegación en conjunto con la identificación de objetos y el dibujo del mapa, no pudieron combinarse debido a los problemas técnicos que presenta el cuatrimotor y que no permiten implementar la secuencia por la alta inestabilidad de vuelo que presenta.

5.3.1. Rutina de vuelo alterna

Para hacer reconocimiento de objetos en el ambiente que se encuentra frente al cuatrimotor, en cuatro orientaciones diferentes, la rutina de vuelo esta conformada por giros de noventa grados. Esta rutina presentó inestabilidad evidente por el desplazamiento sobre los ejes x e y al realizar los giros. A continuación se presentan los resultados de la exploración aérea. Para esto, se colocaron en un rango de cada 90° una serie de objetos para ser identificados y dibujados en la sección de mapa de la interfaz gráfica de usuario.

En la primera prueba que se muestra en la Figura 5.8, se puede observar que se detectaron una serie de objetos rectangulares, dos rectángulos y tres triángulos, no se detectaron objetos con forma circular. En ésta prueba, la identificación de objetos se llevó a cabo desde que el cuatrimotor envia la imagen de la cámara y hasta que se apaga el sistema del cuatrimotor, por esta razón, el video de la interfaz presenta retardo en la visualización.

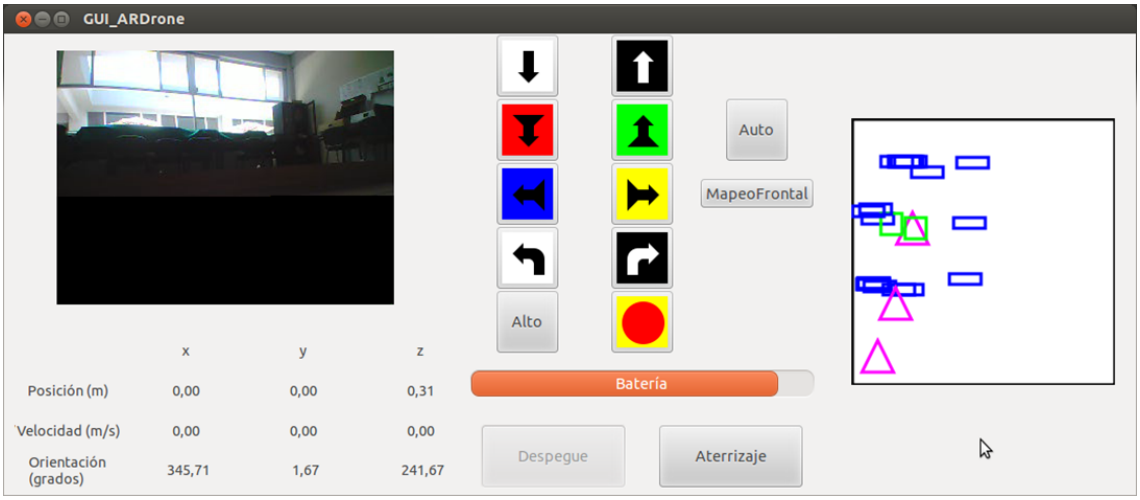


Figura 5.8: Resultado de mapeo con la cámara frontal del cuatrimotor.

Para las pruebas siguientes, la detección de los objetos se activó solo en el momento en que el cuatrimotor estaba alineado en las orientaciones programadas. En la Figura 5.9 se puede observar la identificación de objetos en los

cuatro cuadrantes del mapa a excepción de uno, que es el correspondiente al lado del ambiente de donde el cuatrimotor estaba más alejado.

Figura 5.9: Resultado de mapeo con la cámara frontal del cuatrimotor, la identificación de objetos presentó fallo en la orientación donde el objeto se encontraba más lejos de la cámara.

En la Figura 5.10 se observan detecciones de los objetos en los cuatro cuadrantes del mapa, en este caso, dos objetos no fueron identificados, un círculo en el primer cuadrante y un triángulo en el cuarto cuadrante.

En la Figura 5.11, la detección no se realizó para el caso de un rectángulo en el tercer cuadrante y se detectó un rectángulo de más en el cuarto cuadrante. La variación en la orientación del cuatrimotor en relación con la posición de los objetos, afecta en gran medida la identificación de éstos en el ambiente, también la variación en la posición del cuatrimotor al momento de realizar los giros afecta que en algunos casos el cuatrimotor se encuentre demasiado lejos de los objetos complicando la detección.

Figura 5.10: Se muestra el dibujo de objetos en los cuatro cuadrantes del área del mapa. La identificación de objetos se activa cuando el cuatrimotor alcanza alguna de las cuatro orientaciones implementadas, logrando que el video se muestre sin retardos.

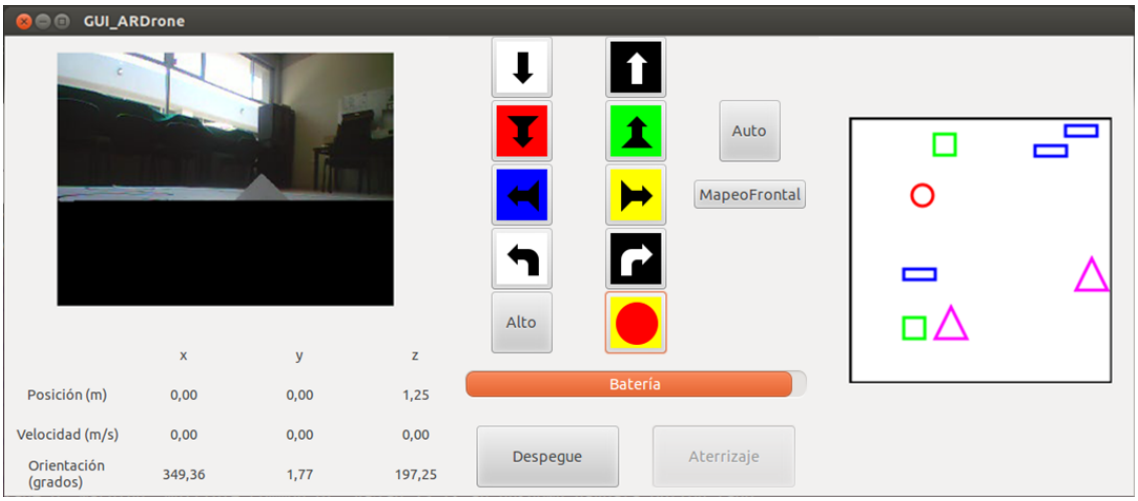


Figura 5.11: Resultado de mapeo con la cámara frontal del cuatrimotor.

Capítulo 6

Conclusiones y Trabajo Futuro

En este trabajo de investigación, se implementó la generación de mapa por medio de la identificación de objetos en un robot aéreo tipo cuatrimotor, apoyados en una interfaz gráfica de usuario para el monitoreo del estado de vuelo y envío de comandos de locomoción.

Se utilizó un algoritmo de identificación de objetos, para identificar cuatro objetos diferentes, en los cuatro casos la identificación fue mayor al 80 % de exactitud. Se logró identificar los cuatro objetos diferentes al mismo tiempo. Esto sirvió de base para la respresentación del ambiente explorado.

Se puede tener el control de navegación del cuatrimotor a través de la interfaz, sustituyendo a los controles de radiofrecuencia. Cada una de las secciones cumplió con las funciones implementadas.

Se implementaron rutinas de locomoción para el cuatrimotor frente/lateral con el que se exploró un área en forma de cuadrilátero. El área se limitó en función del área que cubre la cámara en base a la altura de vuelo, las limitaciones de estabilidad en cambios de velocidad y orientación y la duración de la batería.

Se restringieron parámetros de operación del cuatrimotor para calcular en base a la altura de vuelo, el área que cubre la cámara para aproximar la ubicación de los objetos identificados y poder aproximar un mapa del ambiente.

Se integraron elementos de inteligencia artificial y robótica para dotar a un drone con cierto nivel de autonomía. La autonomía en este tipo de plataformas aún se encuentra en proceso de desarrollo y experimentación.

6.1. Trabajo Futuro

Se proponen los siguientes puntos como trabajo a futuro:

- Interfaz de control para dispositivo móvil.
- Buscar una plataforma abierta que permita programar y modificar el control de los motores.
- Implementar identificación de personas en el cuatrimotor, que le permitan tener funciones de vigilancia o búsqueda y rescate.

Bibliografía

[1] Turing A. M. Computing Machinery and Intelligence. *Mind, New Series*, 59(236):433–460, October 1950. 1.1

[2] Mitchell T.M. *Machine Learning*. McGraw-Hill, 1997. 1.1, 3

[3] Spong Mark W. *Robot Dynamics and Control*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1989. 1.1

[4] Who did actually invent the word robot what does it mean? http://capek.misto.cz/english/robot.html. December 2012. 1.1

[5] Abdala Castillo S. and Ñeco Carbeta R. Caracterización de un robot manipulador articulado. Master's thesis, Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET), 2003. 1.1

[6] Barrientos A., Del Cerro J., Gutiérrez P., San Martín R., Martínez A., and Rossi C. Vehículos aéreos no tripulados para uso civil. Tecnología y aplicaciones. Technical report, Grupo de Robótica y Cibernética, Universidad Politécnica de Madrid, 2007. 1.1, 2.1, 2.3.1

[7] Hoffmann Gabriel M., Waslander Steven L., and Tomlin Claire J. Quadrotor Helicopter Trajectory Tracking Control. *AIAA Guidance, Navigation and Control Conference and Exhibit*, pages 18–21, August 2008. 1.1

[8] Rodríguez-Lozada González Diego. *SLAM Geométrico en tiempo real para robots móviles en interiores basado en EKF*. PhD thesis, Escuela Técnica Superior de Ingenieros Industriales, 2004. 1.1

[9] Szeliski Richard. *Computer Vision: Algorithms and Applications*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010. 1.1

[10] Hidalgo Chaparro Andreu. Reconocimiento de objetos multi-clase basado en descriptores de forma. *Escuela Técnica Superior de Ingeniería*, 2008. 1.1

[11] Olmedo González Eric. Cómputo paralelo aplicado al reconocimiento de objetos basado en aprendizaje automático. Master's thesis, Programa Académico de Maestría en Ingeniería en Sistemas y Cómputo Inteligente, Universidad Politécnica de Puebla, 2012. 1.1

[12] Viola P. and Jones M. J. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, May 2004. 1.1, 3.1, 1, 2, 3, 4, 3.1.1, 3.1.1, 3.1.2, 3.1.2, 3.1.3, 3.1.3, 3.1.3, 3.1.4

[13] *AR.Drone manual*, Parrot 2010. 1.1, 2.2.3, 2.2.4, 2.2.5

[14] Hernández N., Ocaña M., Pizarro D., Bergasa L. M., Sotelo M.A., Barea R., López E., and Herranz F. Sistema de Detección de Incendios basado en Robot Aéreo Quadrotor. *Seminario Anual de Automática, Electrónica Industrial e Instrumentación 2008 SAAEI2008*, pages 401–406, September 2008. 1.4

[15] J. Toledo, L. Acosta, M. Sigut, and J. Felipe. Stabilisation and altitude tracking of a four-rotor microhelicopter using the lifting operators. *Control Theory Applications, IET*, 3(4):452–464, 2009. 2.1

[16] http://www.muyinteresante.es/parrot-presenta-la-nueva-version-de-su-famoso-cuadricoptero. November 2012. 2.2

[17] http://www.parrot.com/es/. Consultado 04 de Abril de 2013. 2.2, 2.3, 4.11

[18] Ignacio Fernández, Manuel Mazo, JoséL. Lázaro, Daniel Pizarro, Enrique Santiso, Pedro Martín, and Cristina Losada. Guidance of a mobile robot using an array of static cameras located in the environment. *Autonomous Robots*, 23(4):305–324, 2007. 2.3.1.1

[19] J. Borenstein, H. R. Everett, L. Feng, and D. Wehe. Mobile robot positioning sensors and techniques, 1997. 2.3.1.1

[20] Noelia Hernández, Manuel Ocaña, Luis M. Bergasa, Rafael Barea, Elena López, and Fernando Herranz. Sistema de Detección de Incendios basado en Robot Aéreo Quadrotor. pages 401–406, sep. 2008. 2.3.1.1

[21] D.G. Lowe. Object recognition from local scale-invariant features. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1150–1157 vol.2, 1999. 2.3.1.2

[22] Lee J., Kang T., Lee K., Im S., and Park J. Vision-Based Indoor Localization for Unmanned Aerial Vehicles. *Journal of Aerospace Engineering*, 24(3):373–377, 2011. 2.3.1.2, 2.5

[23] Thrun S. and Bücken A. Integrating Grid-Based and Topological Maps for Mobile Robot Navigation. In *Proceedings of the AAAI Thirteenth National Conference on Artificial Intelligence*, 1996. 2.3.2, 2.3.2.1

[24] Herrera Vega Javier. Localización de Robots en Mapas Topológicos con Información Visual. *Facultad de Ciencias de la Computación, BUAP*, 2010. 2.3.2.1

[25] Herrera Vega Javier. Localización de robots en mapas topológicos con información visual. Master's thesis, Facultad de Ciencias de la Computación, Benemérita Universidad Autónoma de Puebla, 2010. 2.3.2.1

[26] Michael Drumheller. Mobile Robot Localization Using Sonar. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-9(2):325–332, 1987. 2.3.2.1

[27] H.P. Moravec and A. Elfes. High resolution maps from wide angle sonar. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2, pages 116–121, 1985. 2.3.2.1

[28] Gallardo López Domingo. *Aplicación del muestreo bayesiano en robots móviles: Estrategias para localización y estimación de mapas del entorno.* PhD thesis, Universidad de Alicante, 1999. 2.3.2.2

[29] Angelo Arleo, José del R. Millán, José Del R. Millán, and Dario Floreano. Efficient learning of variable-resolution cognitive maps for autonomous indoor navigation. *IEEE Trans. Robot. Automat*, 15:990–1000, 1999. 2.3.2.2

[30] G. Dudek and M. Jenkin. *Computational Principles of Mobile Robotics*. Computational Principles of Mobile Robotics. Cambridge University Press, 2010. 2.3.2.2

[31] Kuipers Benjamin and Byun Yung-Tai. A Robot Exploration and Mapping Strategy Based on a Semantic Hierarchy of Spatial Representations. *Journal of robotics and autonomous systems*, 8:47–63, 1991. 2.3.2.2

[32] Sebastian Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71, 1998. 2.3.2.3

[33] G. Bradski and A. Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, 2008. 2.4, 3.2

[34] M. Andriluka, P. Schnitzspan, J. Meyer, S. Kohlbrecher, K. Petersen, O. Von Stryk, S. Roth, and B. Schiele. Vision based victim detection from unmanned aerial vehicles. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1740–1747, 2010. 2.5

[35] Kumar Niraj, Agarwal Vasu, Dhawan Divya, Agarwal Yash, Trivedi Ayush, Singh Amandeep, Jain Nikhil, Roychoudhury R.D., Aditya Andra, and Jain Mohit. Unmanned Aerial Vehicle for Autonomous Navigation In Cluttered Indoor Invironments. *Third Symposium on Indoor Flight Issues*, 2012. 2.5

[36] Dijkshoorn Nick. Simultaneous localization and mapping with the ar.drone. Master's thesis, Universiteit Van Amsterdam, 2012. 2.5

[37] Tine Lefebvre, Herman Bruyninckx, and Joris De Schutter. Kalman filters for nonlinear systems: a comparison of performance, 2001. 2.5

[38] Bachrach Abraham, He Ruijie, and Roy Nicholas. Autonomous Flight in Unknown Indoor Environments. *International Journal of Micro Air Vehicles*, 1(4):217–228, 2009. 2.5

[39] Wang Y. Y. and Li J. Feature-selection ability of the decision-tree algorithm and the impact of feature-selection/extraction on decision-tree results based on hyperspectral data. *International Journal of Remote Sensing*, 29(10):2993–3010, 2008. 3

[40] Nandhitha N.M, Manoharan N., Sheela Rani B., Venkataraman B., Kalyanasundaram P., and Baldev Raj. A Comparative Study on the Performance and Suitability of Feature Extraction Algorithms on Tungsten Inclusion Thermographs for On-line Weld Monitoring. *International Journal of Theoretical and Applied Mechanics*, 4(10):107–118, 2009. 3

[41] W.K. Pratt. *Digital Image Processing: PIKS Scientific Inside*. Wiley, 2007. 3.1

[42] Papageorgiou C. and Poggio T. A trainable system for object detection. *International Journal of Computer Vision*, 38(1):15–33, 2000. 3.1.1

[43] Cortes J. A., Cano H. B., and Chavez J. A. Implementation of the wavelet haar to reconstruct the function f (t) = t on the range [-3.3] different degrees of resolution. *Scientia et Technica*, XIV(38):71–76, 2008. 3.1.1

[44] Crow Franklin C. Summed-area tables for texture mapping. *Proceeding SIGGRAPH '84 Proceedings of the 11th anual conference on Computer graphics and interactive techniques*, 18(38):207–212, July 1984. 3.1.2

[45] Lienhart R. and Maydt J. An extended set of haar-like features for rapid object detection. *IEEE International Conference on Image Processing*, 1:900–903, 2002. 3.1.2

[46] Sierra B. *Aprendizaje Automático: conceptos básicos y avanzados*. Prentice Hall, Madrid, España, 2006. 3.1.4, 3.1.4

[47] Valiant L. G. A theory of the learnable. *Communications of the ACM*, 27:1134–1142, November 1984. 3.1.4

[48] Freund Y. and Schapire R. E. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771–780, September 1999. 3.1.4

[49] Eric Gregori. Viola jones simplified. 3.2

[50] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47, March 2002. 3.2

[51] Hanafi Dirman and Qetkeaw Mongkhun. Gui controller design for quadcopter control. *International Conference on Advance Research in Computer Science, Electrical and Electronics Engineering*, September 2013. 4

[52] A. Zul Azfar and D. Hazry. Simple gui design for monitoring of a remotely operated quadrotor unmanned aerial vehicle (uav). In *Signal Processing and its Applications (CSPA), 2011 IEEE 7th International Colloquium on*, pages 23–27, March 2011. 4

[53] http://www.gtk.org. Consultado 10 de Diciembre de 2012. 4.1

[54] Piskorski S., Brulez N., and Eline P. *AR.Drone Developer Guide, SDK 1.7*, May 2011. 4.1, 4.3, 4.16, 4.17

[55] http://www.grc.nasa.gov/WWW/K-12/airplane/rotations.html. 29 de marzo de 2013. 4.3

[56] J.L. Devore. *Probabilidad Y Estadística para Ingenierías Y Ciencias*. Cengage Learning Latin America, 2008. 4.3.1

[57] Nick Dijkshoorn and Arnoud Visser. A.: Integrating sensor and motion models to localize an autonomous ar.drone. *International Journal of Micro Air Vehicles*, pages 183–200, 2011. 5.3