CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL INSTITUTO POLITÉCNICO NACIONAL

Unidad Zacatenco
Departamento de Ingeniería Eléctrica
Sección de Mecatrónica

Estimación de la Localización de un Vehículo Terrestre Autónomo por medio de un Vehículo Aéreo Autónomo

TESIS

Que presenta
Ing. Fernando Horacio Ponce Rojas

Para obtener el grado de
Maestro en Ciencias

En la especialidad de
Ingeniería Eléctrica

Directores de Tesis
Dr. Hugo Rodríguez Cortés
Dr. Rafael Castro Linares

Ciudad de México Febrero,2018

Agradecimientos

Primero quiero agradecer a Dios por darme la vida y la oportunidad de estudiar una maestría

A mi familia por que ellos son mi principal apoyo y gracias a cada uno de ellos pude realizar esto.

A mi novia por el apoyo y las palabras de aliento en tiempos complicados.

A mis Asesores Dr. Rafael Castro Linares y Dr. Hugo Rodríguez Cortés por los conocimientos aportados ademas del tiempo que se dio cada uno de ellos al tomar este tema de tesis tanto en revisiones como aportaciones al contenido de la misma

A el Dr. José Martínez Carranza por la visita que me permitió realizar en el laboratorio de Robótica del INAOE en la cual pude interactuar con su grupo de trabajo.

A mis compañeros del CINVESTAV con los que compartí el aula y demasiados momentos, tanto de estudio como de vida a través de más de dos años.

Al CINVESTAV y CONACYT por su apoyo económico para realizar mis estudios de Maestría a través de una beca y el uso de sus instalaciones durante la duración del trabajo.

Resumen

En el presente trabajo se desarrolla un esquema de comunicación y control cuyo objetivo es que un Vehículo Terrestre Autónomo (VTA) pueda navegar utilizando el posicionamiento que le proporciona un Vehículo Aéreo Autónomo (VAA). El Vehículo Aéreo utiliza la información del sistema de cámaras infrarrojas OptiTrack para realizar seguimiento de trayectorias. Este mismo determina la posición del Vehículo Terrestre al procesar la información del sensor de vídeo abordo. La información del sensor de vídeo se procesa utilizando el paquete de computo OpenCV. El intercambio de información se realiza utilizando el sistema operativo dedicado a Robótica (ROS). Los resultados experimentales validan el concepto de navegación.

Abstract

In the present work a communication and control scheme is developed whose objective is that an Unmanded Ground Vehicle (UGV) can navigate using the positioning provided by an Unmanded Air Vehicle (UAV). The UAV uses information from the OptiTrack infrared camera system to track trajectories. The UAV determines the position of the UGV when processing the on-board video sensor information. The video sensor information is processed using the OpenCV computer package. The exchange of information is done using the System Operating Robotics (ROS). The experimental results validate the concept of navigation.

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1

INTRODUCCIÓN

En este cápitulo de introducción se presentan las generalidades acerca del proyecto, así como las razones que lo motivan. Se mencionan trabajos previos desarrollados tanto dentro de la sección de Mecatrónica como fuera de ella. Se establecen los objetivos y se describe el trabajo desarrollado.

1.1. Generalidades

En los últimos años, se ha incrementado el interés en los vehículos autónomos pequeños, Vehículos Terrestres Autónomos (VTA) y Vehículos Aéreos Autónomos (VAA). Estos vehículos son especialmente útiles para realizar actividades civiles y militares tales como la inspección en ambientes hostiles, el mantenimiento de infraestructuras, y la agricultura de precisión. El uso de un vehículo aéreo tiene ventajas, comparado con el uso de uno o varios vehículos terrestres debido a su movilidad tridimensional. Vehículos a menor escala pueden resolver tareas de mejor manera que un vehículo grande y ser más rentables.

Los sistemas multirobots son capaces de realizar tareas a mayor velocidad, además la cooperación puede reducir las distancias recorridas y ayudar a transportar mayores cargas. Los vehículos aéreos y terrestres pueden trabajar de una manera complementaria.

Un vehículo aéreo usa su propiedad de desplazamiento vertical para observar un área de mayor tamaño en menor tiempo, mientras que un vehículo terrestre puede observar una escena con mayor eficiencia debido a su proximidad con el objetivo. Esto permite, realizar una búsqueda cooperativa para identificar y localizar objetivos.

En entornos urbanos la navegación autónoma presenta retos importantes debido a que depende del Sistema de Posicionamiento Global (GPS). La señal del GPS puede presentar problemas tanto cobertura, como de deterioro de la calidad en señal. Por lo tanto, es necesario desarrollar métodos alternos para determinar la posición de un robot.

Una técnica para resolver los problemas del GPS es la visión artificial, gracias a los algoritmos que se han desarrollado como la binarización o segmentación de diferentes elementos. Gran parte de los algoritmos de procesamiento de imagenes son de código abierto y una comunidad importante trabaja en su perfeccionamiento en plataformas tales como OPEN CV. Los algoritmos han sido optimizados y pueden procesarse inclusive en computadoras con baja capacidad de procesamiento.

Un robot móvil es una máquina automática capaz de desplazarse en un ambiente determinado, en contraste con los robots industriales que están unidos a un sistema de referencia inercial. En este trabajo se ocupan dos tipos de robots.

- **Vehículo Aéreo.-** Aeronaves no tripuladas que pueden desplazarse de manera tridimensional. Se selecciona un cuatrirotor, un helicóptero de cuatro rotores que pueden realizar maniobras de una manera ágil y veloz, debido a su pequeño peso y tamaño. Por lo que pueden ingresar a lugares de difícil acceso para los humanos, ya sea por restricciones físicas o en términos de seguridad.

- **Vehículo Terrestre.-** Robots capaces de desplazarse en una superficie de contacto, existen diferentes robots móviles terrestres en este trabajo se ocupa un robot diferencial, el cual variando la velocidad y dirección de giro de sus dos ruedas pueden seguir diferentes trayectorias.

1.2. Antecedentes

Se han realizado diferentes trabajos de cooperación entre vehículos terrestres y vehículos aéreos algunos de ellos se presentan a continuación:

En Grocholsky et al. (2006) se presenta un enfoque escalable para la detección y localización de objetivos por medio de una red de vehículos aéreos y vehículos terrestres, aprovechando la mayor precisión que se tiene con los vehículos en tierra y la mayor aérea de cobertura de los vehículos aéreos, debido que en presencia de obstáculos tales como edificios o vallas la eficiencia de los vehículos terrestres se reduce. Los vehículos aéreos son aeronaves de ala fija con una computadora abordo, ademas de sensor GPS, sensores inerciales y dos cámaras de alta definición, en el caso de los vehículos terrestres, son camionetas comerciales de 4 ruedas modificadas, con una computadora a bordo, una cámara, sensor GPS, odometría interna y sensores inerciales, los vehículos se comunican con la estación terrestre por medio de una tarjeta de red inalambrica.

En Frietsch et al. (2008) se lleva acabo el seguimiento entre un vehículo terrestre autónomo y un vehículo aéreo autónomo para la navegación en caso de que el GPS no este disponible. La solución se basa en la información que proporciona la cámara a bordo del vehículo aéreo. El vehículo terrestre cuenta con un patrón determinado, mismo que es identificado mediante el uso de visión artificial, la comunicación entre ambos se realiza por medio de un transmisor de radiofrecuencia y el procesamiento de la imagen por medio de un DSP.

En Minaeian et al. (2016) se presenta un trabajo conjunto de vigilancia entre un vehículo aéreo y varios vehículos terrestres utilizando la mayor área de cobertura de los vehículos aéreos para seguir multitudes y el mayor detalle en los vehículos terrestres para seguir individuos, se ocupo un cuatrirotor y autos de control remoto. Los vehículos están equipados con sensores integrados y potentes computadoras para el procesamiento de la imagen.

Para la ubicación espacial de los objetos detectados se ocupa la técnica de conocimiento de puntos específicos en la imagen, dichos puntos son marcadores únicos portados por los vehículos terrestres, en todo momento se conoce la ubicación de dichos marcadores, se ocupan técnicas de visión artificial para detectar los marcadores apoyados en las librerías Open CV, el color fue seleccionado como marcador por lo que para mejorar la detección de las marcas independientemente de la iluminación se utiliza la transformación del espacio de color RGB a HSV de las imágenes.

En Li et al. (2016) un método de planificación de ruta híbrida entre vehículos aéreos y vehículos terrestres. En el cual el a partir de la imagen obtenida de un vehículo aéreo se genera una ruta para el vehículo terrestre misma que este desarrolla.

En la Sección de Mecatrónica del Departamento de Ingeniería Eléctrica ya se han llevado a cabo varios trabajos de investigación sobre el control de formación de robots móviles y el control de vuelo de vehículos aéreos.

En Rosaldo Serrano (2014) se trabajo con un sistema multi-agente heterogéneo descentralizado donde los agentes que forman parte del sistema son robots móviles. En este trabajo se aplica una ley de control a un grupo de robots móviles heterogéneo conformado por un robot tipo uniciclo y una aeronave tipo cuatrirotor para que ellos sigan una trayectoria definida mientras conservan una formación. Se emplea un esquema lider-seguidor donde cualquiera de los robots debe ser capaz de ser el líder o el seguidor de la formación.

En Vallejo Alarcón (2015) se desarrollo un esquema de control centralizado para una formación líder-seguidor que involucra un cuatrirotor y un robot móvil. En este trabajo se designa al robot móvil tipo (2,0) como el líder, mientras que un cuatrirotor se asigna como el seguidor. El objetivo es que el vehículo seguidor mantenga una distancia relativa del líder tomando en cuenta el modelo cinemático del líder así como la medición de las posiciones pero sin conocer la trayectoria que se realizara.

La detección de los vehículos en los trabajos realizados en la sección han sido a través del sistema de cámaras Optitrack, implementado en un laboratorio de la sección.

1.3. Planteamiento del Problema

Motivados por los resultados obtenidos y la experiencia adquirida en la Sección de Mecatrónica con vehículos aéreos y vehículos terrestres se propone el presente tema de maestría.

Llevar acabo un trabajo conjunto de un vehículo terrestre con un vehículo aéreo para mejorar la navegación del vehículo terrestre a través de visión artificial. Específicamente, se quiere guiar el vehículo terrestre por medio del vehículo aéreo.

1.4. Objetivos

Objetivo General

Diseñar un esquema de navegación para un vehículo terrestre autónomo basado en la información visual que adquiere un vehículo aéreo autónomo.

Objetivos particulares.

- Diseño y Construcción del hardware y software del vehículo aéreo autónomo y vehículo terrestre autónomo para el sistema de navegación.
- Diseño del Sistema de Navegación.
- Evaluación experimental del Sistema de Navegación

1.5. Trabajo Desarrollado

En este trabajo se diseña un esquema de comunicación y control para lograr que un vehículo terrestre autónomo pueda navegar utilizando un sistema de posicionamiento que utiliza un sensor de visión abordo de un vehículo aéreo autónomo. El vehículo terrestre tiene en la parte visible de la cámara abordo del vehículo aéreo un patrón tricolor que permite ubicar su posición y orientación con respecto a un sistema de referencia anclado al vehículo aéreo.

Lo primero que se realizo fue el análisis de temas similares que incluyan la navegación de un vehículo terrestre y un aéreo, como el presentado en Frietsch et al. (2008) el cual se centra en la navegación de un vehículo terrestre en caso de que la señal de GPS este degradada. O el que se presenta en Grocholsky et al. (2006) donde aeronaves no tripuladas y vehículos terrestres con cámaras a bordo, desarrollan un sistema de detección de obstáculos con base en los datos que adquieren de las cámaras que portan. En Minaeian et al. (2016) se realiza un sistema de vigilancia donde el vehículo aéreo detecta los patrones de color que portan los vehículos terrestres y en conjunto el seguimiento a objetos en movimiento.

Se construyó un robot diferencial a partir de un kit y se determino que su procesador digital de señales seria una Raspberry Pi debido a que esta permite comunicación Wi-Fi, se puede programar lenguaje de alto nivel ademas de contar con entrada/pines de salida. Se comenzó con la familiarización de la Raspberry Pi debido a que esta requiere su propio sistema operativo y se crearon programas sencillos para desplazar remotamente al robot diferencial.

El vehículo aéreo seleccionado es un AR Drone 2.0 debido a la cantidad de documentación disponible para controlarlo a nivel traslacional utilizando librerías de ROS, además de contar con dos cámaras, frontal e inferior. La cámara inferior se utiliza para determinar la posición del vehículo terrestre.

El control del vehículo terrestre se realiza a nivel cinemático y no cuenta con controladores internos de velocidad. Por lo tanto se implemento el control individual utilizando el sistema de cámaras OptiTrack, para verificar el correcto desempeño tanto de la comunicación como de las señales de control.

Se realizo la familiarización con ROS y debido a que este funciona en el sistema operativo Ubuntu, la familiarización con dicho sistema. Se desarrollo el algoritmo para reconocimiento del patrón tricolor en la imagen, el cual se identifica empleando una cámara web apoyándose de las librerías Open CV y posteriormente se desarrollo con la cámara incluida en el AR Drone.

El AR Drone se controla por medio de ROS con una librería llamada ardrone_autonomy, la cual permite controlar los angulos de alabeo, cabeceo permitiendo el movimiento en el plano (x,y), la velocidad de ascenso y la velocidad del angulo de guiñada de la aeronave, así como tener un control remoto por medio del teclado. Primero se desarrollaron algunos vuelos dirigidos desde el teclado, para obtener las primeras pruebas del detección del vehículo terrestre.

Posteriormente se procedió a un vuelo autónomo por lo que fue necesario incorporar el sistema de cámaras OptiTrack a la información enviada al AR Drone, desarrollando su respectiva comunicación. Se llevaron acabo las primeras pruebas de vuelo autónomo para así proceder a probar el sistema completo. Al final se realizaron algunas pruebas fuera del laboratorio dirigiendo el AR Drone por medio del teclado, para así probar el sistema de visión en diferentes entornos.

1.6. Aportaciones de la Tesis

La aportación de esta tesis se centra en la utilización de ROS para coordinar el movimiento de dos vehículos, un terrestre y un aéreo, utilizando la cámara a bordo del vehículo aéreo. Por medio de ROS se realiza el intercambio de información entre los diferentes dispositivos, la comunicación del controlador del vehículo aéreo con el sistema de cámaras Optitrack, la comunicación entre el vehículo terrestre, el sistema de procesamiento de información de la cámara, la comunicación entre el sistema de procesamiento de información de la cámara y el Vehículo Aéreo por medio de señal Wi-Fi.

1.7. Organización de la Tesis

En el Capítulo 2 se describe los conocimientos adquiridos previos, las herramientas de computo a utilizar así como la descripción de los modelos ocupados.

En el Capítulo 3 se presenta el desarrollo de la solución del problema, la construcción o en su caso la familiarización con los elementos a ocupar, los controles implementados, la descripción individual de cada elemento del sistema así como la conjunción en uno solo.

En el Capítulo 4 se presentan los resultados obtenidos de los experimentos, así como las dificultades que se obtuvieron en cada uno de ellos y las condiciones para efectuarse.

Finalmente, el Capítulo 5 se presenta conclusiones y perspectivas de este trabajo.

CAPÍTULO 2

Marco Teórico

Este capítulo describe las herramientas de cómputo utilizadas en este trabajo, así como los modelos correspondientes al robot diferencial y al cuatrirotor.

Se inicia con la descripción de ROS(Robot Operating System), de cada elemento que lo compone y de como interactuan entre sí. Debido a que en ROS se conjuntan los sistemas, se desarrolla el control del Vehículo Aéreo, el programa de visión artificial apoyado de las librerías Open CV y la comunicación este toma importancia.

Adicionalmente se definen las características de los robots tanto aéreo como terrestre, los modelos correspondientes, se describe la estructura y capacidades de los vehículos utilizados así como de sus unidades de procesamiento abordo. Para finalizar con los sistemas de visión y la comunicación usados, como son la librería Open CV, el sistema de cámaras OptiTrack y los sockets TCP.

2.1. ROS (Robot Operating System)

ROS cuenta con un conjunto de librerías para el desarrollo de programas, que permiten el control de diferentes robots de una manera sencilla. Facilita tareas como son adquisición de datos de sensores, comunicación entre dispositivos, simulación de sistemas, desarrollo de interfaz de usuario, etc. Es un software de distribución libre, que permite reutilización de paquetes, por lo que el código puede mejorarse y publicarse recursivamente. Este software se implementa sobre el sistema operativo Ubuntu. Rizo Gonzalez and Ruiz Restrepo (2014)

ROS es una plataforma de desarrollo para aplicaciones de robots, cuenta con características como son manejo de mensajes y reutilización de código. Divide un gran sistema en pequeños subsistemas que trabajan individualmente interactuando entre si. Una de las razones de la creación de ROS es la de tener acceso a elementos de un sistemas complejos de manera sencilla.Quigley et al. (2015)

ROS cuenta con funciones listas para usarse por lo que no es necesario escribir un nuevo código para las funciones existentes. Además de que son altamente configurables, cuenta con herramientas, para depurar, visualizar y simular. Admite sensores y actuadores de gama alta, está lleno de dispositivos controladores y paquetes de raíz de varios sensores y actuadores.

ROS permite comunicación entre diferentes nodos, los cuales pueden programarse en cualquier lenguaje que tenga librerías de ROS. Existen implementaciones de ROS en las cuales algunos nodos se programan en lenguaje C, otros en Python y es posible la interacción entre ellos.Quigley et al. (2009)

ROS trabaja por módulos y si uno de ellos se bloquea los demás pueden continuar con su funcionamiento, proporciona métodos robustos para reanudar el modulo que se ha bloqueado. Cuenta con manejo concurrente de recursos, más de un nodo puede pedir información acerca de otro por medio de los tópicos, y así llevar más de un procesamiento a la vez, aumentando la capacidad del sistema.

ROS cuenta con librerías en aplicaciones como: percepción, identificación de objetos, segmentación y reconocimiento, reconocimiento facial, reconocimiento de gestos, seguimiento de objetos, comprensión de movimiento, percepción de profundidad mediante el uso de dos cámaras, movimientos, robots móviles, control, planificación,manipulación de objetos.

ROS cuenta con un portal web para manejar consultas de soporte entre más usuarios, la cual tiene un crecimiento constante en desarrolladores de todo el mundo.

La arquitectura de ROS está dividida en tres secciones o niveles: nivel sistema de archivos, nivel gráfico y nivel de la comunidad (usuario).

2.1.1. Nivel Sistema de Archivos

Los archivos ROS están organizados en el disco duro de una manera particular, como se muestra en la Figura 2.1.

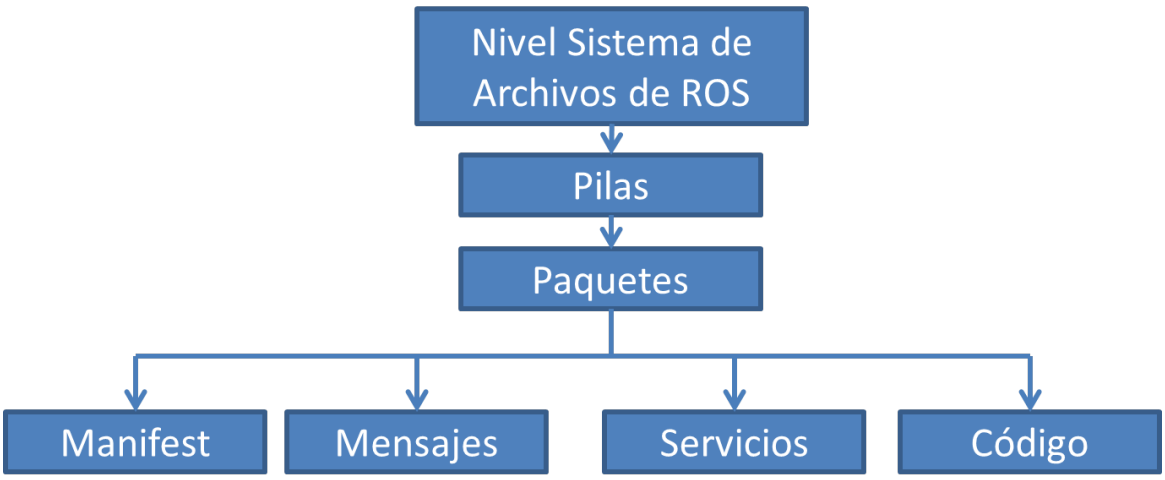

Figura 2.1: Organización de archivos en disco duro de ROS.(Basado en (Joseph, 2015))

Los conceptos del nivel de sistema de archivos se refiere principalmente a los recursos de ROS que se encuentran en el disco como son:

Paquetes: Son la unidad básica del software ROS. Contienen los archivos ejecutables(nodos), bibliotecas, archivos de configuración, mensajes, servicios etc.
Manifests: Estos archivos contienen la información de los paquetes como las licencias y las dependencias.
Pilas: Son paquetes especiales que sirven para representar un grupo de paquetes relacionados entre sí.
Mensajes: Los mensajes son la información que se envía de un proceso a otro por medio de archivos.
Servicios: Es un tipo de interacción solicitud/respuesta entre procesos.

2.1.2. Nivel Gráfico

El cómputo en ROS es una red de procesos llamados nodos. El cual a su vez es llamado nivel gráfico los principales conceptos de este nivel son: nodos, maestro, parámetro de servidor, mensajes, tópicos, servicios, bolsa. Como se muestra en la Figura 2.2.

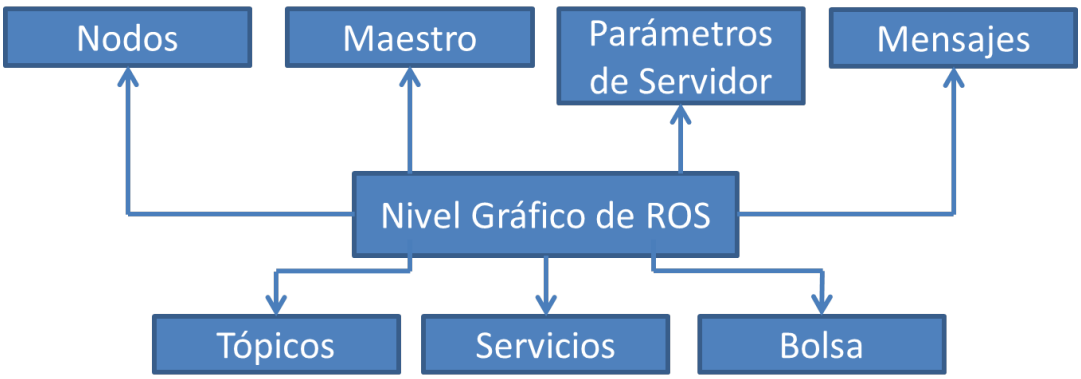

Figura 2.2: Elementos ROS Nivel Gráfico.(Basado en (Joseph, 2015))

Nodos: Es un archivo ejecutable que tiene un propósito específico. Los programas pueden contener varios nodos, los cuales realizaran pequeñas funciones de una función general. Estos nodos usualmente están escritos en lenguajes como C++ o Pyton.
Maestro: Es el encargado de registrar los nodos y dar acceso para que se comuniquen entre ellos. Sin él no se podrían comunicar los nodos, el envío de mensajes, la publicación a tópicos, etc. Es posible ejecutar al maestro en una computadora y tener nodos de manera remota en otra.
Parámetros de Servidor: Permite mantener los datos para ser almacenados en una ubicación central. Todos los nodos pueden acceder y modificar estos parámetros, son parte del maestro.
Mensajes: Los nodos se comunican a través mensajes. Los mensajes son estructuras de datos de tipos primitivos (enteros, flotantes, booleanos, etc). Los mensajes pueden incluir estructuras anidadas y arreglos.
Tópico: Es uno de los mecanismos mediante cual los nodos intercambian mensajes. Los nodos pueden realizar una publicación a un tópico específico permitiéndole a otros nodos suscribirse para obtener datos de este tema. Cada tópico tiene un nombre específico y cualquier nodo puede acceder a el mediante la suscripción.
Servicios: Es otro mecanismo de comunicación entre nodos, a diferencia de los tópicos, se lleva a cabo de un nodo a otro, funciona mediante comunicación de petición/respuesta. Es decir, el primer nodo envía una petición y el segundo responde. En este modelo sólo dos nodos interactúan, el que envía y el que lo recibe. El servicio de ROS es como una

llamada a un procedimiento remoto.
Bolsas: Es un formato que permite almacenar datos de ROS que pueden ser difíciles de recopilar y posteriormente reproducirlos.

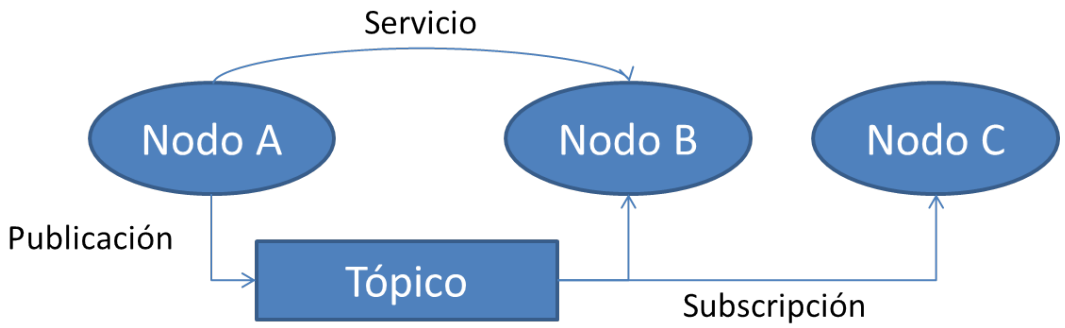

Figura 2.3: Tópico y Servicio en ROS. (Basado en (Joseph, 2015))

Muchos nodos pueden suscribirse y publicar en un mismo tópico, la comunicación utilizando un servicio es solamente posible de un nodo a otro como se muestra en la Figura 2.3. Para una comunicación directa entre otros dos nodos se tendría que establecer un nuevo servicio.

2.1.3. Nivel Comunidad

El nivel comunidad son recursos de ROS, que se habilitan para una el intercambio de software y conocimiento. Los diversos recursos de esta comunidad son:

Distribución: Son colecciones de pilas que se pueden instalar en ROS. Las distribuciones permiten una recolección más sencilla del software ROS
Repositorios: ROS contiene una red de repositorios que diferentes instituciones pueden desarrollar sobre su propio robot.
ROS Wiki: Es el foro principal donde se discuten y preguntan temas de ROS. Cualquier usuario de este sistema operativo puede contribuir con su propia documentación, señalando correcciones, escribiendo tutoriales o más.
ROS Answers: Este sitio web ayuda a hacer preguntas sobre ROS, si publicamos en este sitio otros usuarios de ROS pueden verlo y dar soluciones Joseph (2015)

2.2. Vehículo Aéreo

Como se menciono anteriormente los vehículos aéreos son aeronaves no tripuladas que pueden desplazarse de manera tridimensional, una forma de clasificarlos es por el funcionamiento de sus alas.

- Ala fija como planeadores y aviones.
- Ala rotatoria como son helicópteros y multirotores

Los multirotores son helicopteros con al menos tres alas rotativas, para este trabajo se escogió un cuatrirotor, multirotor que cuenta con cuatro rotores.
Los usos del cuatrirotor han ido en crecimiento debido a sus características tales como:

- Capacidad de vuelo en estado estacionario.
- Maniobrabilidad en cualquier dirección sin necesidad de una orientación especifica.
- Pequeño peso y tamaño.

Su funcionamiento se basa en el empuje generado por los cuatros rotores, el cual compensa la gravedad y genera aceleración de manera vertical. El sentido de giro de los rotores se muestra en la Figura 2.4. Dos de ellos giran en sentido horario y los otros dos en sentido antihorario, cuando estos giran a la misma velocidad se lleva a cabo el vuelo en estado estacionario. La inclinación se logra mediante la diferencia de velocidades.

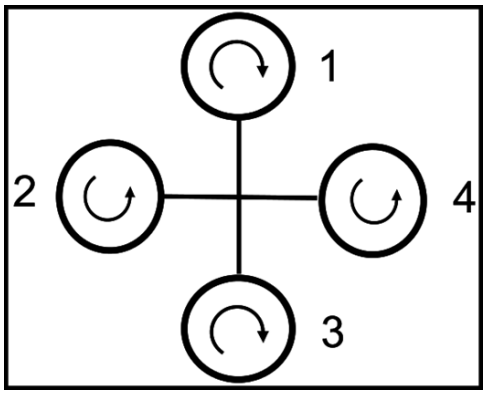

Figura 2.4: Sentido de Giro Rotores de un Cuatrirotor

Para un movimiento lateral un rotor debe aumentar su velocidad, el contrario disminuirla en la misma proporción y los otros mantenerse. Según sea el desplazamiento deseado. En el caso del giro, los rotores que tienen el mismo sentido de giro deberán aumentar su velocidad en la misma proporción que los del sentido contrario la disminuyan.

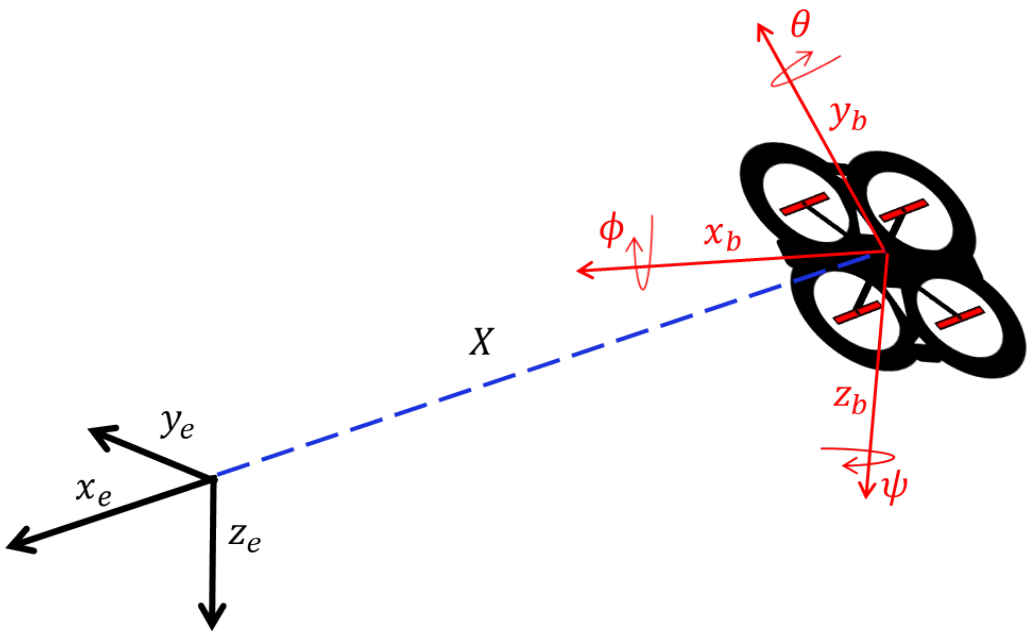

Figura 2.5: Cuatrirotor

En Vargas-Jacob et al. (2016) se presenta un modelo dinámico dado por el enfoque Newton-Euler en cual se considera el movimiento general del cuerpo rígido como una combinación de movimientos traslacional y rotacional. En la figura 2.5 podemos ubicar al cuatrirotor en un marco según la convención Norte-Este-Abajo a través de $X = [xyz]^T$.

La orientación del marco de referencia fijo al cuerpo del cuatrirotor con respecto al marco de referencia inercial esta definido por la matriz de rotación R, la matriz de rotación ocupando la secuencia de rotación Z-Y-X expresada en los angulos de euler es.

$$R(\Phi) = \begin{bmatrix} c_\psi c_\theta & c_\psi s_\theta s_\phi - s_\psi c_\phi & c_\psi s_\theta c_\phi + s_\psi s_\phi \\ c_\theta s_\psi & s_\psi s_\theta s_\phi + c_\psi c_\phi & s_\psi s_\theta c_\phi + c_\psi s_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta s_\phi \end{bmatrix} \tag{2.1}$$

donde $\cos(*) = c*$, $\sin(*) = s*$ y su orientación $\Phi = [\psi\theta\phi]^T$ donde ψ, θ y ϕ son los ángulos de guiñada, alabeo y cabeceo respectivamente.
Sobre el cuatrirotor actuan las fuerzas debidas al campo gravitatorio de la tierra $F_g e_b^e$ y las debidas al empuje producido por los rotores $F_T = T_T e_b^3$. En donde

$$T_T = \sum_{i=1}^{4} T_i$$

T_i es el empuje debido a cada rotor, e_b^e y e_b^3 dos vectores unitarios que apuntan en dirección del eje z_e del marco de referencial inercial y z_e marco de referencia fijo al cuerpo, m la masa del vehiculo y g la constante de aceleración gravitacional de la Tierra. En el caso del vuelo en estado estacionario $F_T = F_g$.

$$M_e = \begin{bmatrix} L \\ M \\ N \end{bmatrix} = \begin{bmatrix} (T_2 - T_4)l \\ (T_1 - T_1)l \\ -Q_1 + Q_2 - Q_3 + Q_4 \end{bmatrix} \tag{2.2}$$

donde l es la distancia del eje y el centro de masa del cuatrirotor, T_i y Q_i son los empujes y pares generados por cada rotor.

$$\begin{aligned} m\ddot{X} &= -mge_b^3 + R(\Phi)T_T e_b^3 \\ J\dot{\Omega} &= -\Omega \times J\Omega + M_e \end{aligned} \tag{2.3}$$

Donde J es la matriz de inercia del vehículo aéreo y Ω es el vector de velocidades angulares $\Omega = [pqr]$ expresado expresado en sistema coordenado fijo al cuerpo del cuatrirotor. Este vector de velocidades angulares Ω esta relacionado con las derivadas de los angulos de guiñada, cabeceo y alabeo por la siguiente expresión Donde

$$\Omega = W(\Phi)\dot{\Phi}$$

$$W(\Phi) = \begin{bmatrix} 1 & 0 & -s_\theta \\ 0 & c_\phi & c_\theta s_\phi \\ 0 & -s_\phi & c_\theta c_\phi \end{bmatrix} \tag{2.4}$$

$$\dot{\Phi} = W^{-1}\Omega \tag{2.5}$$

$$W^{-1} = \begin{bmatrix} 1 & t_\theta s_\phi & t_\theta c_\phi \\ 0 & c_\phi & c_\theta s_\phi \\ 0 & \frac{s_\phi}{c\theta} & \frac{c_\phi}{c_\theta} \end{bmatrix} \tag{2.6}$$

Las ecuaciones que describen el comportamiento (2.3) y (2.5) puede expandirse como sigue

$$\begin{aligned} m\ddot{x} &= -T_T(c_\psi s_\theta c_\phi + s_\psi s_\phi) \\ m\ddot{y} &= -T_T(s_\psi s_\theta c_\phi - c_\psi s_\phi) \\ m\ddot{z} &= -T_T(c_\psi c_\phi) + mg \\ \dot{\phi} &= p + t_\theta s_\phi q + t_\theta c_\phi r \\ \dot{\theta} &= c_\phi q - s_\phi r \\ \dot{\psi} &= \frac{s_\phi}{c\theta} q + \frac{c_\phi}{c_\theta} r \end{aligned} \tag{2.7}$$

$$I_{xx}\dot{p} = (I_{yy} - I_{zz})qr + L$$
$$I_{yy}\dot{q} = (I_{zz} - I_{xx})pr + M$$
$$I_{zz}\dot{r} = (I_{xx} - I_{yy})pq + N$$

El sistema de control interno del AR Drone permite llevar $\theta \to \theta_d$, $\phi \to \phi_d$, $p \to 0$, $q \to 0$ y $r \to r_d$, $z \to z_d$ por lo que las ecuaciones se reescriben como

$$m\ddot{x} = -T_T(c_\psi s_{\theta d} c_{\phi d} + s_{\psi d} s_{\phi d})$$
$$m\ddot{y} = -T_T(s_\psi s_\theta c_\phi - c_\psi s_\phi)$$
$$m\ddot{z} = -T_T(c_\psi c_\phi) + mg$$
$$\dot{\psi} = \frac{c_\phi}{c_\theta} r_d$$

Tomando en cuenta que el vehículo aéreo se moverá mediante cambios pequeños en sus ángulos θ_d y ϕ_d, se considera $\cos(*) = 1$ y $\sin(*) = *$, se considerara que el vehículo se encuentra a una altura constante por lo tanto $T_T = mg$. Obteniendo el modelo final.

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = -g \begin{bmatrix} \cos(\psi) & \sin(\psi) \\ \sin(\psi) & -\cos(\psi) \end{bmatrix} \begin{bmatrix} \theta_d \\ \phi_d \end{bmatrix} \tag{2.8}$$

$$\dot{\psi} = r_d$$

2.3. AR Drone

En 2004, la empresa Parrot comenzó un proyecto llamado AR.Drone con el objetivo de producir un vehículo aéreo no tripulado para el mercado masivo de videojuegos y entretenimiento. El proyecto fue presentado en el Consumer 2010 Electronics Show. A partir del 18 de agosto de 2010, AR.Drone se lanzó al mercado. El AR Drone de Parrot es el cuatrirotor mostrado en la Figura 2.6.

Cuenta con un control interno que le permite tanto el vuelo en estado estacionario como la navegación por medio de cambios en sus ángulos. Recibe instrucciones a través de 4 puertos de la red interna que genera los cuales de muestran en la Tabla 2.1.
Los objetivos específicos del proyecto AR.Drone van mucho más allá de usos convencionales comúnmente considerados tanto civiles como aplicaciones militares. Abarcan realidad aumentada (AR), videojuegos e interactividad. Mediante el uso de sensores de bajo costo (MEMS y cámaras) para mercados masivos. Donde el precio minorista es de gran importancia. Este proyecto y los algoritmos integrados tienen la particularidad de

Figura 2.6: AR.Drone 2.0

Puerto	Tipo	Uso
5554	UDP	Datos de Vuelo enviados desde el AR Drone
5555	TCP	Video obtenido desde el AR Drone
5556	UDP	Comandos enviados al AR Drone
5559	TCP	Puerto de Control para Datos Críticos

Tabla 2.1: Puertos AR Drone 2.0

ser muy estable, robusto y fácil de usar. Bristeau et al. (2011)

Utiliza los sensores mostrados en la Tabla 2.2. para estimar ángulos, aceleraciones y velocidades lineales.

Cada conjunto rotor-motor está montado en un pie de plástico PA66 con una placa de control implementada en un micro controlador ATMEGA 8L. La estructura mecánica que conecta los rotores está formada por tubos de fibra de carbón y una cruz central reforzada de plástico. Una estructura de polipropileno soporta la batería de Litio-Polímero de 11.1 V y 1500 mAh. Los elementos de la estructura están dados por los la Tabla 2.3.

Cuenta con dos placas electronicas a bordo. La primera un procesador ARM9-core de 32 bits 468 MHz que corre un sistema operativo Linux embebido y es capaz de adquirir datos de dos cámaras de alta definición, envía telemetría por WiFi a 200 Hz y tiene un conector externo miniUSB para nuevos sensores.

La segunda placa tiene un micro-controlador PIC de 16 bits a 40 MHz. Recibe los datos de un acelerómetro de 3 ejes, un giróscopo de 2 ejes, un giróscopo de 1 eje vertical y 2

	Sensor
1	Cámara HD, 720p, 30FPS.
2	Cámara, QVGA, 60FPS.
3	Giroscopio de 3 ejes.
4	Acelerómetro de 3 ejes.
5	Magnetómetro de 3 ejes.
6	Sensor de presión.
7	Sensor ultrasónico de distancia.

Tabla 2.2: Sensores AR. Drone 2.0.

	Característica
1	Cruz central de tubos de fibra de carbono.
2	Centro de inercial aislado de la vibración de los motores.
3	Casco de PPE inyectado en unmolde de metal sinterizado.
4	Protección repelente de líquidos en el sensor ultrasónico.

Tabla 2.3: Elementos de la estructura AR. Drone

sensores ultrasónicos; además implementa una estrategia de control interna para estabilizar la dinámica rotacional del cuatrirotor. Curi et al. (2014)

2.4. Vehículo Móvil Terrestre

Un robot móvil terrestre, es un dispositivo electromecánico capaz de desplazarse de un punto a otro en un ambiente de trabajo determinado. Una forma de clasificarlos es por el mecanismo de locomoción que ocupa en su desplazamiento. Por lo que existen:

- Ruedas
- Patas
- Orugas

En gran parte de las aplicaciones el robot móvil de ruedas es la mejor opción debido a que el control de ruedas es más sencillo con respecto a los otros dos. Ademas de ser un robot de bajo consumo, al no requerir un gran numero de partes, las ruedas son adecuadas en el desplazamiento sobre superficies lisas.

Las restricciones en un robot móvil están dadas por el tipo de llantas que lo conforman por ello se definen dos tipos de grados de libertad para clasificar los robots móviles. El grado de movilidad S_m y grado de direccionabilidad S_s. Un robot móvil cuenta con 3 grados de libertad en su movimiento por lo cual al tener ruedas fijas orientables independientes restaran movilidad al robot.

Si más de dos ruedas fijas se encuentran en el mismo eje de rotación estas se consideran dependientes pues solo restringen un movimiento del robot. De igual manera las ruedas céntricas orientables que tienen la misma dirección se consideran dependientes y restringen solo un movimiento del robot. Por lo tanto $S_m = 3 - R_i$ donde R_i es el número de ruedas independientes.

Los grados de movilidad S_m se refieren al número de grados de libertad con los que el robot cuenta con la configuración dada en un instante de tiempo sin cambiar la dirección de sus ruedas, es decir los grados de libertad que el robot tendrá al solo manipular las velocidades de sus ruedas.

Los grados de direccionabilidad S_s son definidos por el numero de ruedas orientables que puedan ser direccionadas independientemente, para asignar una orientación al robot.

Los grados de maniobrabilidad S_M se definen como la suma de grados de movilidad y grados de direccionabilidad $S_M = S_m + S_s$ y se refiere al número de grados de libertad que un robot puede ser manipulado.Siegwart et al. (2011)

Existen 5 clasificaciones para robots de 3 ruedas de acuerdo con los grados definidos anteriormente. Mostrados en la Figura 2.7

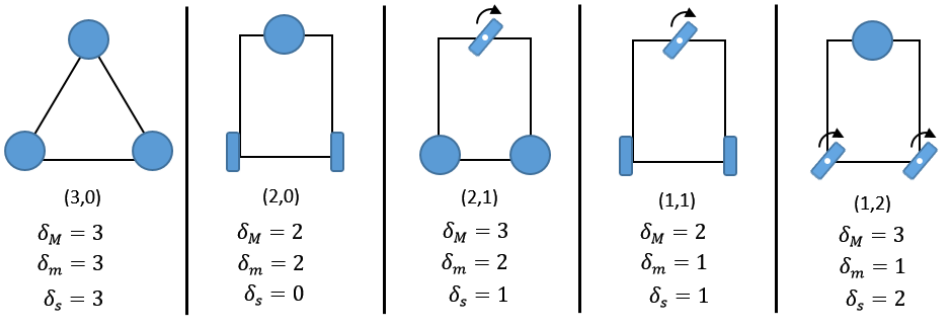

Figura 2.7: Tipos de Robots de 3 ruedas

En este trabajo de investigación se utilizan robots tipo (2,0) o también llamados robots diferenciales los cuales cuentan con 2 llantas fijas y una llanta omnidireccional cuyo

modelo cinemático relaciona las velocidades del sistema con la velocidad de cambio de posición y orientación del móvil, en un marco de referencia fijo el cual tiene como origen un centro de rotación.de Wit et al. (2012).

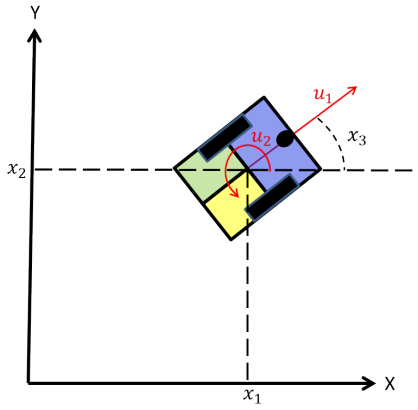

Figura 2.8: Robot Móvil Tipo Diferencial

Se cuenta con dos entradas, u_1 representa la velocidad lineal, u_2 es la velocidad angular del robot, debido a la configuración del robot diferencial este no puede moverse en la dirección perpendicular al eje de las ruedas, también conocido como restricciones no holonómicas mencionadas en Brockett et al. (1983).

De acuerdo con Campion and Chung (2008) el modelo cinemático del robot móvil puede ser descrito por las ecuaciones.

$$\begin{aligned} \dot{x}_1 &= u_1 \cos(x_3) \\ \dot{x}_2 &= u_1 \sin(x_3) \\ \dot{x}_3 &= u_2 \end{aligned} \tag{2.9}$$

donde x_1 , x_2 son las coordenadas de posición del centro del robot y x_3 la orientación del móvil como se muestra en la Figura 2.8.

2.5. Raspberry Pi

Raspberry Pi fue originalmente creada para inspirar jóvenes programadores a perfeccionar sus talentos de codificación y ganarse un lugar en la universidad de Cambridge en la carrera de informática. Pero debido a sus características comenzó a comercializarse. En

la Figura 2.9 se muestra la placa.

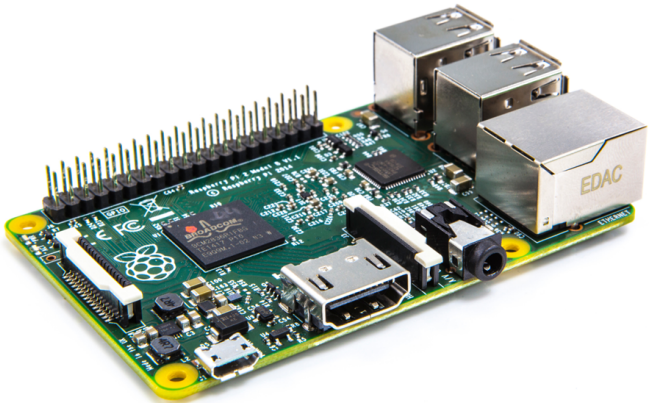

Figura 2.9: Raspberry Pi

Es un computador de placa reducida el cual puede correr diferentes sistemas operativos como son Raspbian, Ubuntu Mate, etc. Cuenta con 4 puertos USB que puede usarse para conectar un teclado, un mouse, un transmisor de señal Wifi o cualquier periférico disponible, además de un conector Ethernet estándar, una entrada HDMI para conectarlo a alguna pantalla, 40 pines entrada/salida de propósito general, entre otras cosas más. Una de las capacidades importantes con las que cuenta son los pines de entrada/salida debido a que ofrecen diferentes formas de controlar dispositivos y recibir información de sensores a la par de correr un sistema operativo. Pi (2013)

2.6. OPEN CV

Open CV(Open Source Computer Vision) es una librería de código abierto con una infraestructura para aplicaciones de visión artificial.

La librería está escrita en C y C++, puede ejecutarse en diferentes sistemas operativos como son Linux, Windows y Mac OS. Fue diseñado para una mayor eficiencia computacional y con un fuerte enfoque en aplicaciones en tiempo real además puede aprovechar el multinucleo de los procesadores. Uno de los objetivos de OpenCV es proporcionar una infraestructura fácil de utilizar en aplicaciones de visión bastante sofisticadas. Contiene más de 500 funciones que abarcan muchas áreas de visión como son: seguridad, visión estereoscópica y robótica.

Grandes compañías son parte de la gran comunidad de usuarios que utiliza Open CV como IBM, Microsoft, Intel, SONY, Siemens y Google además de grandes centros de investigación.

La librería tiene más de 2500 algoritmos, que incluye algoritmos de aprendizaje de máquina y visión artificial. Estos algoritmos permiten identificar objetos, caras, clasificar acciones humanas, hacer tracking de movimientos de objetos, extraer modelos 3D, encontrar imágenes similares, eliminar ojos rojos, seguir el movimiento de los ojos, reconocer escenarios, etc.

Se usa en aplicaciones como la detección de intrusos, monitorear equipos, ayuda a navegación de robots, inspeccionar etiquetas en productos. Bradski and Kaehler (2008)

2.7. Sistema de cámaras OptiTrack

El sistema de visión artificial OptiTrack de la empresa Natural Point, cuenta con dos partes, el hardware específico y una computadora con el programa Motive, que se ejecuta en el sistema operativo Windows, mostrados en la Figura 2.10

El programa Motive funciona en conjunto con los marcadores ópticos que deben de colocarse en los vehículos para poder ser identificados por el sistema.

El hardware del sistema OptiTrack consta de doce cámaras Flex 13, y dos OptiHub 2. Las cámaras emiten luz infrarroja, que es reflectada por los marcadores ópticos y detectada por las mismas cámaras. Las señales de las múltiples cámaras se concentran con los OptiHub 2 y luego se envían por medio de cables USB a la computadora del sistema. La computadora calcula la posición de los cuerpos marcados usando el software Motive.

2.8. Socket TCP

Muchos de los procesos que se ejecutan en alguna computadora requieren obtener o enviar información a otra. Una de las formas para lograr esta comunicación es mediante los protocolos de comunicación TCP y UDP.

El protocolo TCP(Transmission Control Protocol) establece un canal de comunicación punto a punto entre dos computadoras. Dicho canal es exclusivo entre los dos equipos por el cual los datos son enviados y este se mantendrá activo hasta que la conexión fina-

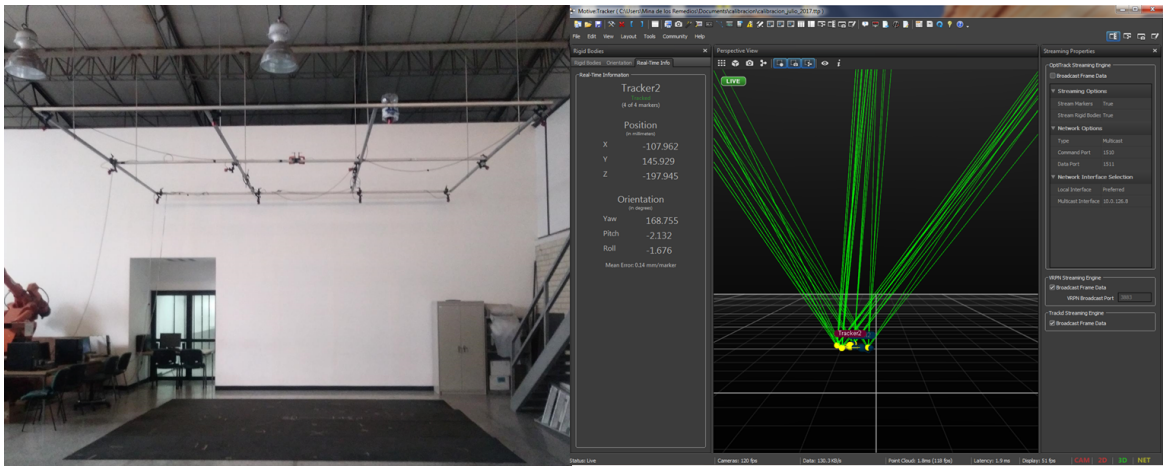

Figura 2.10: Sistema de Cámaras OptiTrack y Programa Motive

lice, esto garantiza que los datos lleguen de un extremo a otro de la conexión y que sean recibidos en el mismo orden en que fueron enviados. También se conoce como protocolo orientado a la conexión.

De igual manera existe el protocolo UDP el cual no es orientado a la conexión, permite que un canal sea ocupado por varios equipos pero no garantiza que los datos se reciban, cuando los datos se reciben el destino debe reconstruir la información ya que esta no viene con un orden especifico.

Los sockets permiten comunicación entre procesos que se encuentran en diferentes máquinas de una red, estos proporcionan un punto de comunicación para enviar y recibir información. Existen dos roles dentro de los sockets como se muestra en la Figura 2.11, el servidor el cual espera un cliente para establecer la conexión, el cliente busca un socket servidor para establecer comunicación. Donahoo and Calvert (2009)

Figura 2.11: Conexión por medio de Socket

CAPÍTULO 3

DESARROLLO

En el siguiente cápitulo se habla del proceso de solución al problema. Comenzando con el control ocupado en el robot móvil, para posteriormente dar construcción del mismo, así como las pruebas llevadas a cabo en el, su procesador de señales es una Raspberry Pi por lo que se habla de su interfaz debido a que en ella son desarrollados e implementados los programas en la misma.

Continuando con la familarización con ROS y los componentes que son parte de la librería ardrone_autonomy, para poder realizar los primeros vuelos con el Ar.Drone 2.0. Posteriormente el algoritmo desarrollado para la detección del Vehículo Terrestre por medio del patrón tricolor así como la interfaz de usuario del sistema.

Como siguiente punto el vuelo autónomo del vehículo aéreo el cual se logro mediante los valores de posición y orientación obtenidos del sistema de cámaras OptiTrack y el calculo de las velocidades lineales requeridas. Finalizando con la integración del sistema completo.

3.1. Control del Robot Móvil

En el capitulo anterior se hablo del modelo de robot diferencial considerado, así como de las variables de control. Por lo que se partirá de las ecuaciones (2.9). La trayectoria a realizar fue definida como un circulo, este puede representarse a partir de las ecuaciones paramétricas siguientes.

$$\begin{aligned} x_{1d} &= R\cos(2\pi ft); \\ x_{2d} &= R\sin(2\pi ft); \\ x_{3d} &= \arctan(\frac{\dot{x}_{2d}}{\dot{x}_{1d}}); \end{aligned} \tag{3.1}$$

donde x_{1d}, x_{2d} y x_{3d} los valores de posición y orientación deseados, R el radio del circulo y f frecuencia de la trayectoria a realizar. Se utiliza el control propuesto por Olivares Cruz (2016), en cual los errores se definen

$$\begin{aligned} \widetilde{x}_1 &= x_1 - x_{1_d} \\ \widetilde{x}_2 &= x_2 - x_{2_d} \\ \widetilde{x}_3 &= x_3 - x_{3_d} \end{aligned} \tag{3.2}$$

Los controles virtuales son

$$\begin{aligned} \eta_1 &= \dot{x}_{1_d} - k_1\widetilde{x}_1 \\ \eta_2 &= \dot{x}_{2_d} - k_2\widetilde{x}_2 \\ \eta_3 &= \dot{x}_{3_d} - k_3\widetilde{x}_3 \end{aligned} \tag{3.3}$$

Obteniendo el control final

$$\begin{aligned} u_1 &= \eta_1\cos(x_3) + \eta_2\sin(x_3) \\ u_2 &= \eta_3 \end{aligned} \tag{3.4}$$

El funcionamiento del robot diferencial es a partir de el cambio de velocidades y dirección de sus ruedas por lo que necesitamos hacer un calculo adicional para poder introducir las velocidades deseadas a nuestro sistema.
Estos valores son determinados a partir de los controles u_1 y u_2 de las ecuaciones (3.4) y la velocidades de las ruedas ω_d y ω_i correspondientes obtenidas a partir de las ecuaciones

$$\begin{aligned} \omega_d &= \frac{u_1 - u_2 l}{r} \\ \omega_i &= \frac{u_1 + u_2 l}{r} \end{aligned} \tag{3.5}$$

donde l representa la distancia entre las ruedas y r el radio de las mismas.

3.2. Construcción y Pruebas Robot Móvil

El robot móvil fue construido a partir de un kit magician mostrado en la Figura 3.1 el cual contiene:

2 pza.	Placa de Plastico cortada y perforada.	4pza	Soporte para motor.
2 pza.	Motorreductor plástico	2 pza.	Llantas de plástico 62 mm.
1 pza.	Rueda loca 30mm.	1pza	Portapilas 4AA
4 pzs.	Tornillos con tuerca M3X30 mm	8pzs	Tornillos con tuerca M3X10 mm
8pzs.	Tornillos M3X5 mm	8pzs	Separadores

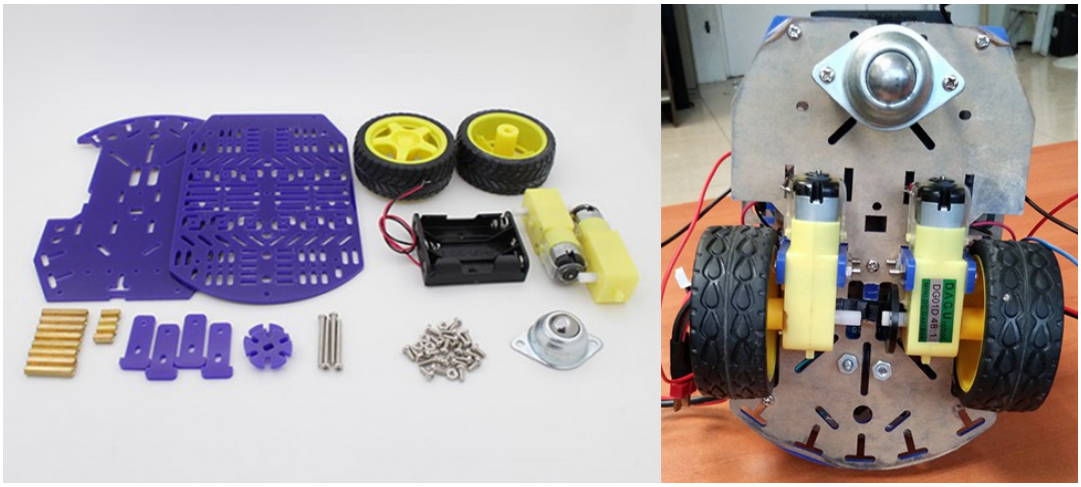

Figura 3.1: Armado de Robot Diferencial.

Fue armado y se incluyo la etapa de potencia por medio de un puente H, el control y comunicación por medio de una Raspberry Pi, cuenta con dos pilas una para alimentar a la Raspberry de 5v y otra para los motores una lipo de 3 celdas a 11.1 V.

Debido a que el control se realiza a nivel cinemático y no cuenta con controladores internos de velocidad fue probado primero en el sistema con cámaras OptiTrack.

El sistema OptiTrack requiere una serie de marcas que reflectan la luz infraroja emitida por las cámaras del sistema, mediante ellas se determina la posición del vehículo. Por lo que fue necesario añadir 4 soportes y una placa para colocar dichas marcas. En la Figura 3.2 se muestra como fueron colocadas las marcas reflectantes.

El sistema OptiTrack determina la posición de dichas marcas, las cuales son enviadas a una computadora central, se forma un cuerpo con el programa Motive, este procesa los valores de posición y orientación de el cuerpo seleccionado y son transmitidos por medio

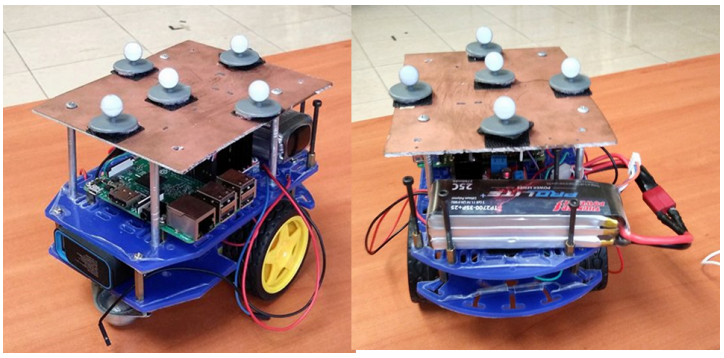

Figura 3.2: Vehículo con Marcas.

de un programa en C++ desarrollado en Microsoft Visual Studio mismo que calcula los valores de control del sistema y los envía por medio de un socket tipo TCP a través de señal Wi-Fi a la Raspberry Pi de nuestro sistema, la cual manda instrucciones a los motores sobre la velocidad que ambos deben adquirir como se muestra en la Figura 3.3.

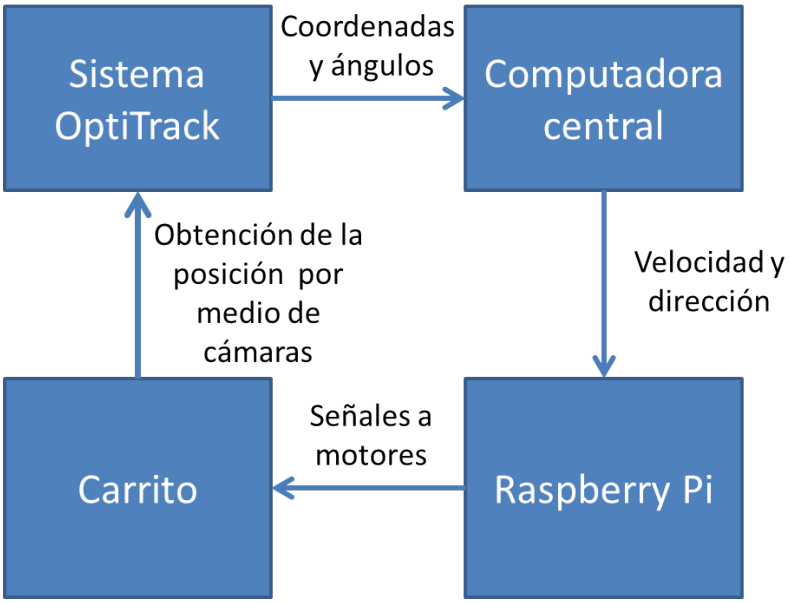

Figura 3.3: Comunicación para Control del Vehículo Terrestre.

El primer sistema que se probó fue el control del robot móvil aun sin incorporar el vehículo aéreo esto debido a que se quería probar que el control propuesto para el vehículo terrestre y el sistema de comunicación funcionara de manera correcta.

En la Figura 3.4 se puede observar el robot móvil llevando acabo la trayectoria de un circulo, con ayuda de las cámaras OptiTrack, las marcas con las que contaba el vehículo, son las que le permite al sistema identificar su posición y orientación.

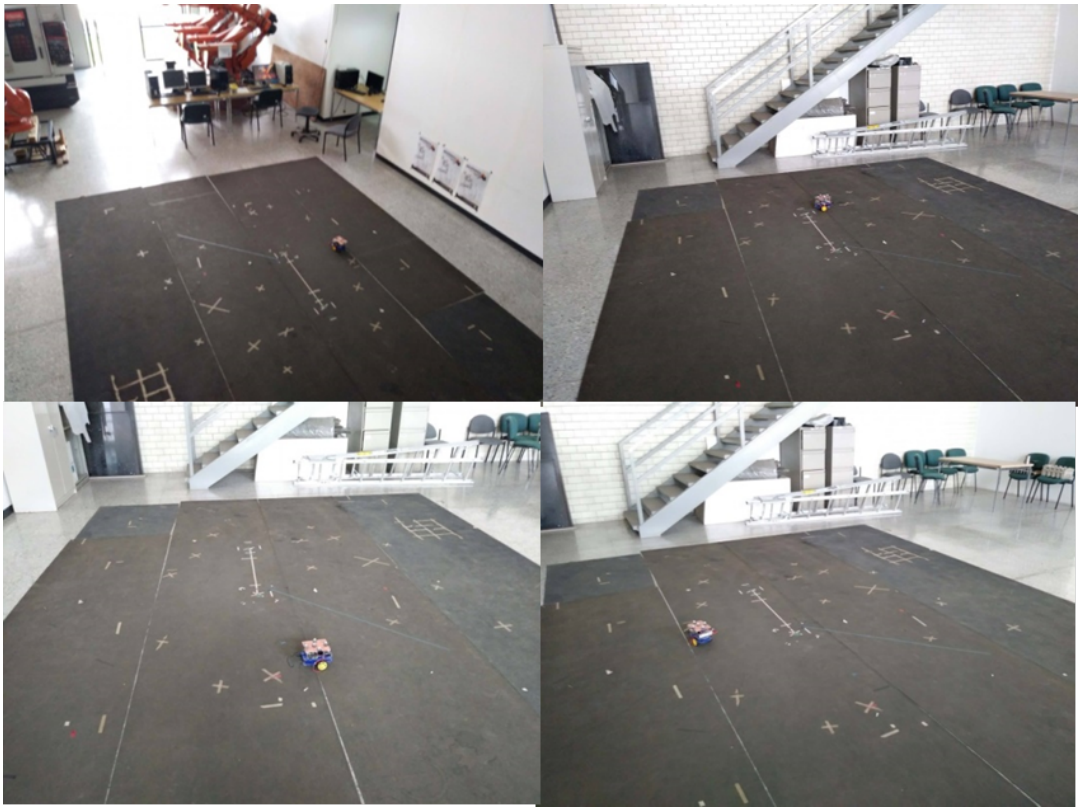

Figura 3.4: Imágenes del Vehículo Terrestre llevando a cabo la trayectoria .

Al realizar este experimento se encontraron varios problemas; como fue la escala en la cual el sistema OptiTrack otorgaba las medidas y los cálculos de los valores de control eran llevados a cabo en diferentes unidades por lo que fue necesario realizar las conversiones correspondientes, además de los ejes coordenados que entregaba el sistema no eran los mismos a los considerados por lo que fue necesario ajustarlos, una vez corregidos estos errores se llevaron a cabo pruebas satisfactorias. Los resultados de dichos experimentos de muestran en el siguiente capitulo.

En la Figura 3.5 se muestran los datos que el robot móvil recibe para llevar a cabo el posicionamiento y la orientación del mismo. La obtención de esta pantalla se realiza mediante la antena Wi-Fi.

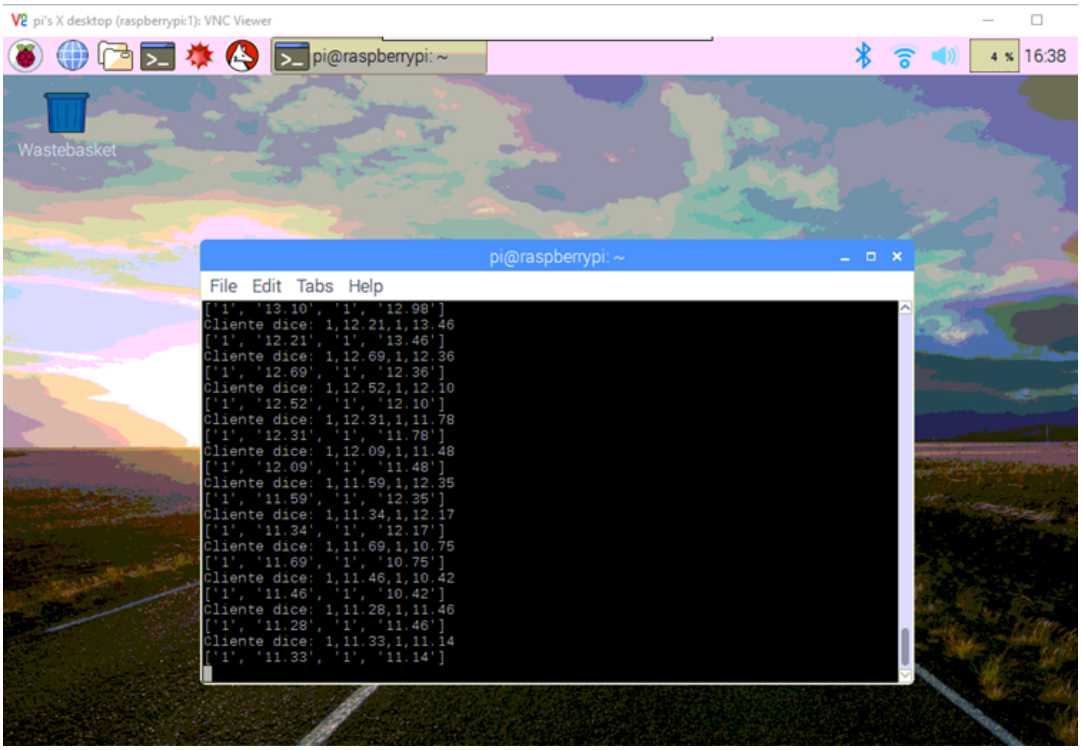

Figura 3.5: Pantalla Raspberry Pi.

En la Raspberry Pi se instalo el Sistema Operativo Raspbian, dicho sistema esta basado en Ubuntu el cual nos permite, ingresar a internet, conectarse con dispositivos externos tanto por señal Wi-Fi como Bluetooth, guardar archivos, herramientas para desarrollo de programas, desde su editor hasta la ejecución de los mismos. Los programas son en su mayoría desarrollados en lenguaje Python.

Además cuenta con una interfaz gráfica amigable que se puede acceder a ella por su puerto HDMI o remotamente por diferentes aplicaciones, la que se ocupo en este proyecto es VNC Viewer.

En un programa en Python se desarrollo un socket TCP para adquirir los valores enviados desde la computadora central, este mismo programa puede enviar valores PWM a través de sus pines de entrada salida, mismos que están conectados a un puente H que da la etapa de potencia a los motores.

3.3. AR Drone y familiarización con ROS

El AR. Drone 2.0 se controla por medio de una conexión Wi-Fi a través de la red interna que este mismo genera, por los cuatro puertos mencionados en el capitulo anterior. Aunque con una correcta configuración también puede conectarse a una red diferente.

Para su control del AR Drone se ocupo la librería de ROS llamada ardrone_autonomy de Monajjemi et al. (2012) descargada desde github, la cual activa un nodo /ardrone_driver que se comunica directamente con el AR.Drone, este le envía las instrucciones y recibe datos del mismo. Esta subscrito a algunos tópicos predeterminados los cuales pueden recibir información para:

- Despegar al drone (/ardrone/takeoff)
- Aterrizar (/ardrone/land)
- Cambio en las velocidades lineales (cmd_vel)

Así mismo devuelve datos por medio de publicar en los tópicos como son:

- Datos de los sensores (/ardrone/navdata)
- Imagen obtenida de la cámara (/ardrone/bottom/image_raw)

La interacción entre el nodo y los tópicos se muestra en la Figura 3.6. Los nodos son los óvalos, mientras que los tópicos son los rectángulos, si un nodo esta subscrito a un tópico se muestra con una flecha dirigida hacia el nodo, debido a que recibe información de el. Si la flecha se dirige del nodo al tópico el nodo tiene la capacidad de publicar en el tópico.

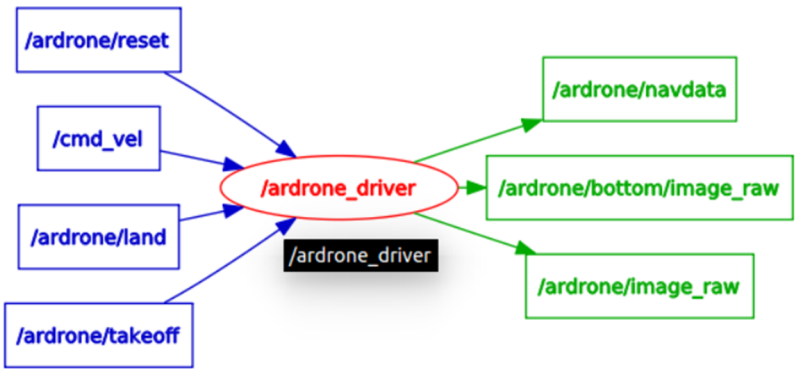

Figura 3.6: Nodo ardrone-driver

Las velocidades máximas se definen a través parámetros que son enviados a ROS en los cuales definimos el angulo máximo de inclinación sobre cada eje, la velocidad de ascenso y descenso, velocidad máxima de giro, la altura máxima permitida, entre otros, dichos parámetros pueden ser definidos en un archivo launch los cuales corren uno o varios nodos de acuerdo a como sean definidos.

Los valores enviados para el control dentro del tópico (cmd_vel) son valores de velocidades, definidos entre -1 y 1 indicando el porcentaje del valor máximo dado los parámetros definidos anteriormente.

En las primeras pruebas de vuelo realizadas con el UAV se utilizo el programa keyboard_controller de Hamer (2013), el cual te permite enviar señales al AR Drone por medio del teclado, se desarrollo en lenguaje python y envía datos al drone cada que presionamos una tecla de las seleccionadas. Este nodo esta subscrito a algunos tópicos que le permite obtener información como son:

- Datos de los sensores de (/ardrone/navdata) a 50 Hz.
- Imagen obtenida de la cámara de (/ardrone/image_raw), a 15 Hz.

Y publica en otros para enviarle señales al AR Drone como son:

- Despegar al drone por (/ardrone/takeoff)
- Aterrizar por (/ardrone/land)
- Reiniciar valores con (/ardrone/reset)
- Envió de velocidades lineales por (cmd_vel)

En la Figura 3.7 se muestra la interacción del nodo keyboard_controler.

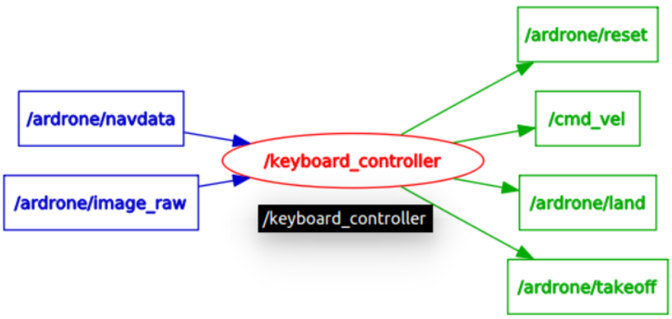

Figura 3.7: Nodos Control Manual AR.DRONE.

Los valores que envía de control sobre cada eje son siempre el valor máximo definido por nuestros los parámetros iniciales, por ello al seleccionarlos debemos tomarlo en cuenta. Para un vuelo más suave se definieron controles adicionales a los incluidos en el keyboard_controller los cuales a partir de teclas diferentes te dan la posibilidad tanto de hacer un vuelo rápido como uno suave en el mismo programa.

Los diagramas obtenidos se hacen mediante rqt_graph una herramienta de ROS la cual permite observar nodos, tópicos, así como las suscripciones correspondientes a cada uno.

3.4. Algoritmo reconocimiento de la imagen

Definimos el patrón tricolor y las dimensiones del mismo con base en las dimensiones del robot diferencial. Las cuales son 12cm x 15cm como se muestra en la Figura 3.8.

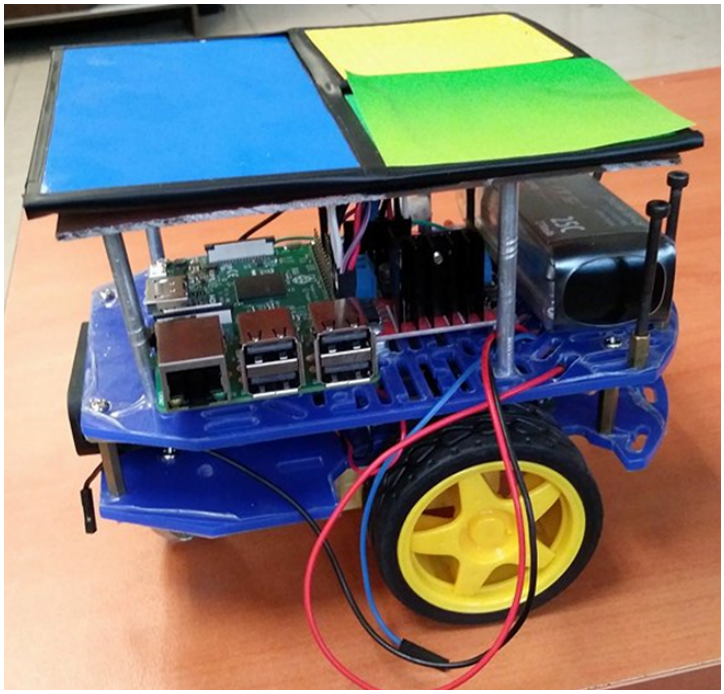

Figura 3.8: Vehículo Terrestre con Patrón Tricolor.

El color azul es el de mayor dimensión con el se identifica el frente del robot, los cuadros pequeños en amarillo y verde complementan el patrón para formar la parte trasera. El patrón tricolor permite identificar la orientación del robot.

Obtenemos la imagen del AR Drone mediante un mensaje ROS con el tópico image_raw. La librera CvBridge convierte en matriz de OpenCV el mensaje de ROS, adquirida del AR Drone. La transformación es mostrada en la Figura 3.9.

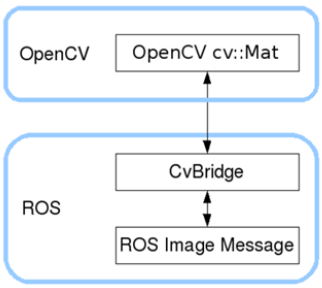

Figura 3.9: Transformación OpenCV-ROS.

En el proceso de obtención de imágenes se presentaron problemas con la adquisición de datos de la cámara, en la visita al laboratorio de Robótica del INAOE en la cual pude interactuar con el grupo de trabajo del Doctor Carranza se me proporcionaron las herramientas para adquirir las imagenes y poder manipularlas, además la interacción de diferentes computadoras en una sola red de ROS.

Una vez que se tiene la imagen como matriz procedemos a convertirla de RGB a HSV esto para que independientemente de la iluminación pueda detectar el rango de un solo color. La escala de colores HSV se muestra en la Figura 3.10.

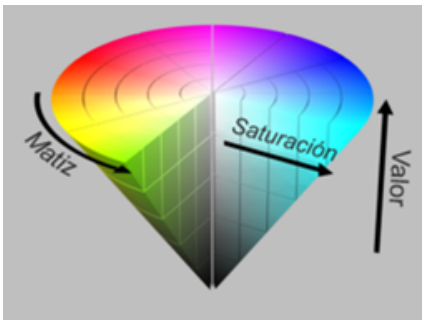

Figura 3.10: Escala HSV. (Tomado de Bhatia (2008))

Hue Saturation Value (HSV)
El modelo HSV fue creado en 1978 por Alvy Ray Smith. Se trata de una transformación no lineal del espacio de color RGB. En ella el matiz se representa por una región circular, una región triangular separada, puede ser usada para representar la saturación y el valor

del color. En el matiz se disponen de 360 grados los cuales cada color RGB le corresponden una separación de 120 grados mientras los colores combinados se encuentran en los rangos intermedios. La saturación corresponde a la distancia del color puro al eje de brillo negro-blanco, los valores posibles van de 0-100, a este parámetro se le suele llamar pureza. Cuanto menor sea la saturación de un color, mayor tonalidad grisácea habrá y más decolorado estará. En cuanto el valor representa la altura en el eje blanco-negro, 0 representa el valor negro y 100 depende de la saturación si el color es blanco o un valor mas cercano al puro. Bhatia (2008)

Una vez obtenida la imagen en HSV se crean 3 imágenes más las cuales separaran por color mediante OpenCv, al definir un rango de colores para ser filtrados en una nueva imagen, adicional se aplico un filtro morfológico, una vez separado se identifica los bordes de cada imagen. Como se muestra el la Figura 3.11.

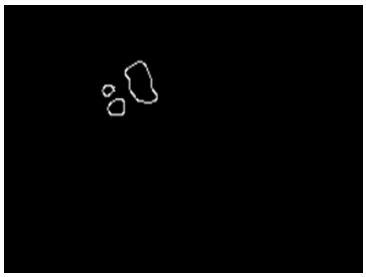

Figura 3.11: Bordes de cada imagen.

Se identifican los contornos y se calculan los centroides por color, una vez identificados los 3 centroides y los bordes de los mismos, se unen en una sola imagen para poder identificar la ubicación y orientación del vehículo. Una vez identificados estos puntos, se procede a dibujarlo en una nueva imagen, la cual nos permite hacer un mapeo, nos apoyamos de una cuadricula para hacer más sencilla la ubicación del vehículo. Como se muestra en la Figura 3.12

Figura 3.12: Imagen formada a partir de centroides.

Dicha cuadricula separa la imagen en función de los pixeles, cada cuadro no representa una distancia fija en metros, ya que esta depende de la altura del vehículo aéreo.

En ocasiones no es posible identificar algún color, ya sea debido a que el patrón no se encuentra por completo en la imagen o por la interferencia de algún otro elemento, por ello se calcula el lugar de los centros reconstruyendo el vehículo a partir de dos colores. Como se muestra en la Figura 3.13. Se puede realizar dicha reconstrucción debido a que el angulo entre el centroide azul, el centro del vehículo y alguno de los otros dos centroides debe ser siempre el mismo.

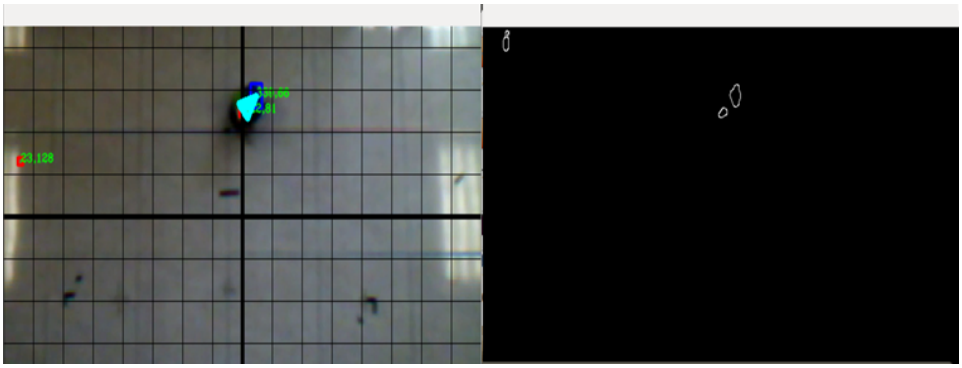

Figura 3.13: Imagen formada a partir de dos colores

Se procede a determinar la posición del vehículo con respecto a la cámara en el plano x,y. La distancia con respecto a los centroides de nuestro patrón es fija en cm por lo tanto se obtiene la equivalencia en pixeles y se calcula la distancia al centro de la imagen, estos valores son ocupados más adelante para el control del vehículo terrestre.

3.5. Interfaz

El sistema cuenta con tres pantallas que permiten con mayor claridad observar donde se encuentra el objeto, se muestran en la Figura 3.14.
En la primer imagen se muestran los datos de distancia al centro de la imagen en el plano x,y tanto deseadas, como medidas, adicionalmente se muestra la orientación y el tiempo que lleva ejecutándose el programa.
En la segunda pantalla se muestra los bordes de la imagen, está es para darse una idea de cómo se están separando los colores de la imagen, si en alguna ocasión no llegan a formarse los tres bordes o estos se encuentran muy separados con respecto a los otros dos, entra el algoritmo de reconstrucción del robot a partir de dos colores.

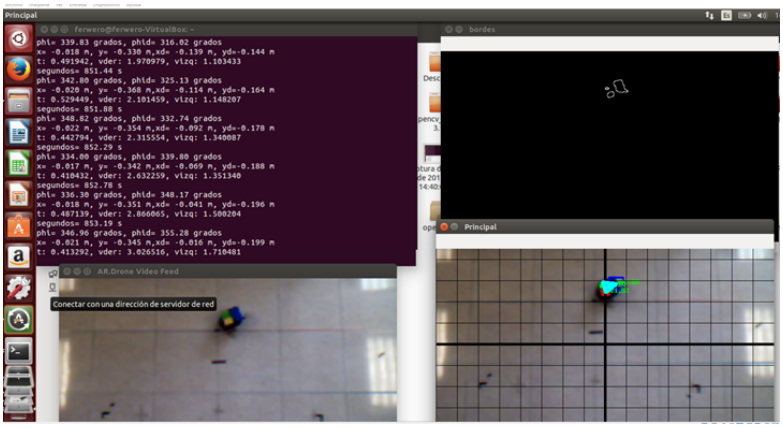

Figura 3.14: Pantallas de Interfaz de Usuario

En la última imagen se dibuja el triángulo, el cual de manera gráfica ubica el vehículo móvil, así como su orientación, la cuadricula es para identificar el centro de la imagen y simplificar la interpretación de la misma. La imagen de la esquina inferior izquierda se obtiene directamente de la cámara, aún sin procesar y ningún algoritmo ha entrado en acción, viene directamente del nodo keyboard_controller.

Una vez detectado el objeto se procedió a enviar señales al robot diferencial, a partir del control implementado en el robot diferencial. El sistema de visión y el sistema de control quedaron implementados en un nodo el cual se llama UGV_detection, subscrito a los tópicos /ardrone/bottom/image_raw el cual le permite obtener imágenes de la cámara abordo del AR Drone y /client_messages con lo cual se declara como cliente de un socket TCP el cual le permite enviar los datos a la Raspberry Pi donde se encuentra el servidor. Lo cual se representada en la Figura 3.15.

Figura 3.15: Nodo de detección del vehículo terrestre

Se realizaron algunos vuelos mediante keyboard_controller y a partir del nodo UGV_detection el seguimiento del vehículo terrestre al vehículo aéreo, los resultados se muestran en el siguiente capitulo.

3.6. Vuelo Autónomo

En el caso del vuelo autónomo del AR Drone se agregaron las marcas correspondientes mostrados en la Figura 3.16, para que el drone recibiera del OptiTrack su ubicación y orientación.

Figura 3.16: AR Drone con marcas

Los datos son enviados por medio de un socket TCP por lo que el nodo UAV_trayectory publica en el tópico server_messages el cual le permite declararse como servidor y recibir los datos del OptiTrack como se muestra en la Figura 3.17.

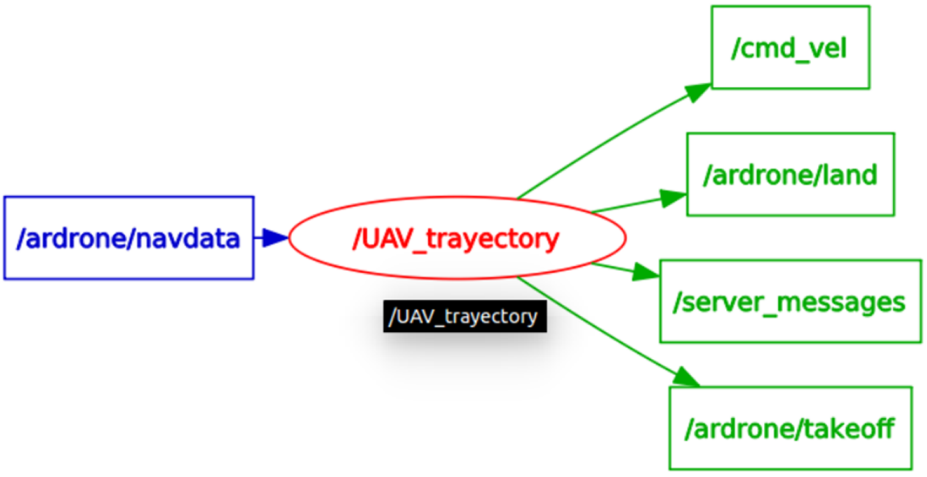

Figura 3.17: Nodo de Trayectoria UAV

Adicional a ello publica en los tópicos cmd_vel que le permite enviar las velocidades al drone, ardrone/takeoff y ardrone/land para aterrizar y despegar el drone y se subscribe a nav_data para obtener los valores de los sensores. La velocidad en que adquiere estos datos es a 50 Hz.

Para realizar el vuelo autónomo se parte del modelo dinámico del capitulo anterior y las ecuaciones (2.8). Se define la trayectoria a partir de las siguientes ecuaciones paramétricas considerando el valor en z y el angulo ψ constantes.

$$\begin{aligned} x_d &= R\cos(2\pi f_2 t) \\ y_d &= R\sin(2\pi f_2 t) \\ z_d &= 2 \\ \psi_d &= 0 \end{aligned} \tag{3.6}$$

Donde R es el radio del circulo y f la frecuencia del mismo. Se implemento el control propuesto en Santiaguillo-Salinas and Aranda-Bricaire (2014) el cual un AR Drone 2.0 logra el seguimiento de trayectorias en el espacio.

$$\begin{bmatrix} \theta_d \\ \phi_d \end{bmatrix} = \frac{1}{g}\begin{bmatrix} -\cos(\psi) & -\sin(\psi) \\ -\sin(\psi) & \cos(\psi) \end{bmatrix}\begin{bmatrix} r_x \\ r_y \end{bmatrix} \tag{3.7}$$

$$\begin{bmatrix} r_x \\ r_y \end{bmatrix} = \begin{bmatrix} -k_{dx}(\dot{x}-\dot{x}_d) - k_{px}(x-x_d) + \ddot{x}_d \\ -k_{dy}(\dot{y}-\dot{x}_y) - k_{py}(y-y_d) + \ddot{y}_d \end{bmatrix} \tag{3.8}$$

$$F = -k_{dz}(z-z_d) - k_{pz}(\dot{z}-\dot{z}_d) + \ddot{z}_d$$

Recordando que la libreria ardrone_autonomy permite controlar los ángulos de alabeo y cabeceo, la velocidad de ascenso y descenso en z y la velocidad angular en la guiñada.

3.7. Sistema Completo

Después de establecer la trayectoria predeterminada a nuestro UAV se procedió a conjuntar el sistema, para ello se ocuparon dos computadoras:

- **Computadora del laboratorio**, esta recibe los datos del sistema de cámaras OptiTrack y los procesa por medio del Programa Motive, los envía a través de un socket TCP por la red Wi-Fi que genera el AR. Drone a la otra computadora.
- **Computadora portatil**, es la que cuenta con ROS y donde se ejecutan los nodos, también recibe datos de los sensores del AR Drone, incluida la imagen de la cámara abordo, en donde se detecta el patrón tricolor incluido en el vehículo terrestre, este realiza el control del VAA y devuelve los valores de las velocidades lineales en cada eje, procesa la imagen y calcula mediante el programa de visión artificial la posición del VTA misma que es enviada a la Raspberry Pi incorporada, para que se lleve acabo el seguimiento de trayectoria.

El flujo de datos del sistema puede observarse en la Figura 3.18.

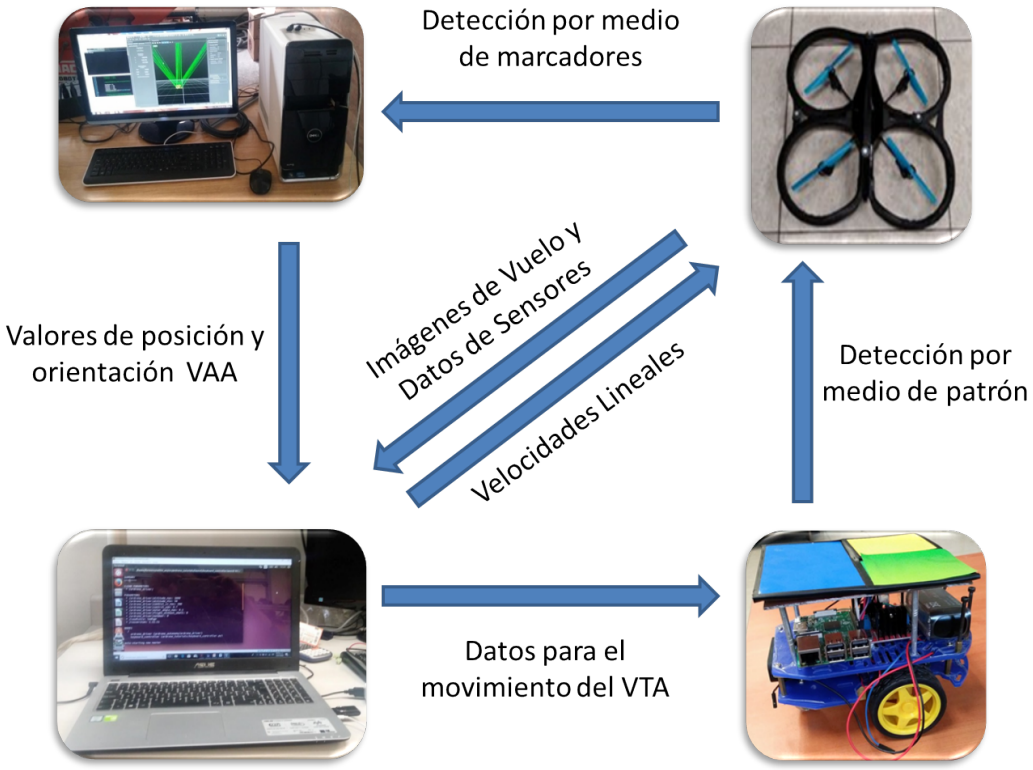

Figura 3.18: Flujo de datos del sistema completo

En cuanto a los nodos desarrollados en ROS y los tópicos del sistema, mismos que son ejecutados en la computadora portátil se muestran en la Figura 3.19. En el centro de la imagen se encuentran los tópicos representados en rectángulos. En los costados se puede identificar por medio de óvalos los 4 nodos del sistema. Los nodos implementados en este trabajo son UGV_detection y UAV_trayectory ambos alojados en la computadora de control pero con la capacidad de comunicarse con la Raspberry Pi y el sistema de cámaras Optitrack respectivamente, por medio de sockets TCP utilizando los últimos tópicos mostrados en la imagen.

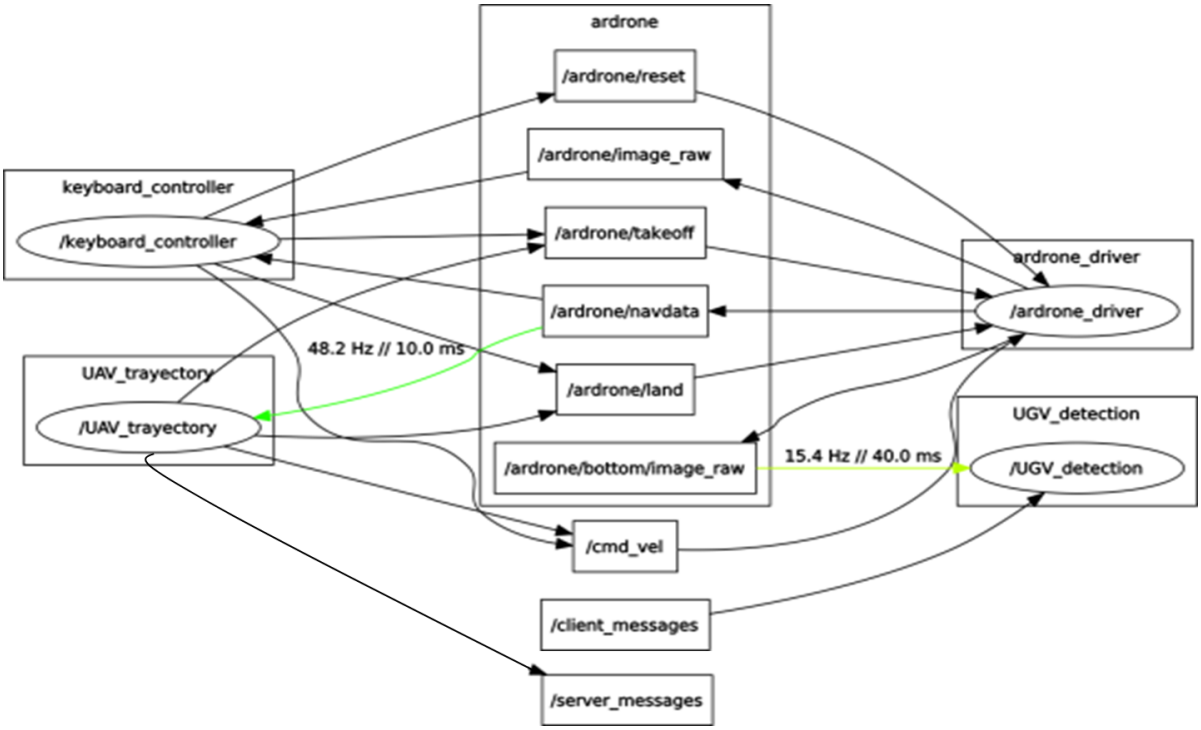

Figura 3.19: Nodos del sistema completo

Los resultados de los experimentos realizados se muestran en el siguiente capitulo.

CAPÍTULO 4

RESULTADOS

En este capitulo se explican los experimentos desarrollados así como los parámetros ocupados, las condiciones de los mismos y las gráficas obtenidas. Comenzando con las pruebas del robot móvil, el cual realizo el seguimiento de una trayectoria predeterminada por medio de marcadores ópticos y el sistema de cámaras OptiTrack.

Continuando con el seguimiento del robot aéreo a una trayectoria predeterminada,con el apoyo de marcadores ópticos. Posteriormente el seguimiento del VTA por medio del VAA manteniendo un control manual del VAA. En este caso los marcadores ópticos en ambos vehículos eran simplemente para obtener las gráficas por medio del Optitrack

Se procedió a probar el sistema completo, el cual consistía en el robot aéreo realizando una trayectoria predeterminada, en el algún punto del recorrido este se encontraba con el VTA lo cual le permitía ser detectado y comenzar a recibir instrucciones para el seguimiento del mismo.

Como prueba adicional se realizo el seguimiento del VTA al VAA en un entorno externo al laboratorio, para probar el reconocimiento en diferentes ambientes pero debido a que la ubicación del VAA dependía de las lecturas del OptiTrack, se llevo un control manual y no fue posible adquirir gráficas de este experimento.

4.1. Vehículo Terrestre

Como primer experimento se llevo a cabo el seguimiento del robot móvil a una trayectoria predeterminada, la cual se selecciono como un circulo.

Para dicho experimento el sistema de visión utilizado fue el sistema de cámaras OptiTrack y el programa Motive, la computadora central la PC del laboratorio en la cual por medio del entorno de desarrollo Microsoft Visual Studio en código C se implemento un socket TCP el cual enviaba valores a nuestra Raspberry Pi adquiridos del OptiTrack. A su vez la Raspberry Pi contenía un socket TCP en el cual era declarado como servidor, recibiendo por Wi-Fi datos de nuestra PC. Los parámetros utilizados fueron los mostrados en la Tabla 4.1.

Radio	0.5 m.
Inicio	0.6,-0.1
f	$\frac{1}{\pi}$
K_1	1.0
K_2	1.2

Tabla 4.1: Parámetros para seguimiento VTA

Los resultados obtenidos se muestran en la Figura 4.1.

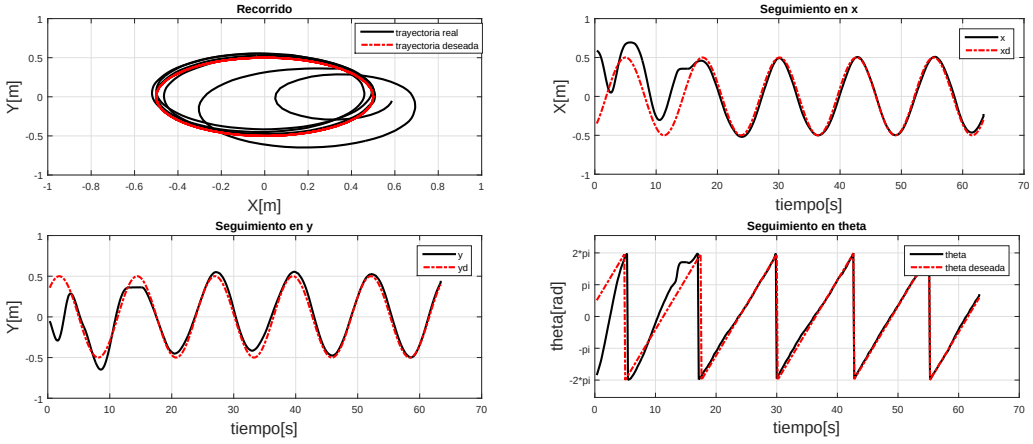

Figura 4.1: Seguimiento de un Circulo.

En la primer imagen en la esquina superior izquierda se muestra la comparación entre la trayectoria deseada y la trayectoria real, después de la segunda vuelta el seguimiento a la trayectoria fue correcto pero se busco que no tardara tanto tiempo en llegar a la trayectoria deseada por lo que se aumentaron las constantes en el segundo experimento.

En la segunda imagen en la esquina superior derecha se muestra el seguimiento en x, se inicio de un punto lejano al deseado y el sistema logro en seguimiento a partir del segundo ciclo.

En la tercer imagen en la esquina inferior izquierda se muestra el seguimiento en y, al igual que en x los primeros dos ciclos fueron de recuperación del sistema, pero en y se necesitaron dos ciclos más para un mejor seguimiento, ya que este contaba con un desplazamiento el cual a partir del 5to ciclo fue reducido.

En la ultima imagen en la esquina inferior izquierda se muestra el seguimiento en orientación, la cual comenzando en un punto lejano busca recuperarse de una manera inmediata obteniendo el objetivo desde el primer ciclo pero debido al error en la posición y la recuperación en las mismas sale de la trayectoria regresando a ella al comenzar el tercer ciclo.

Con el objetivo de alcanzar la trayectoria en un tiempo menor las constantes del primer experimento fueron aumentadas obteniendo los parámetros mostrados en la Tabla 4.2.

Radio	0.5 m.
Inicio	0.1,0.5
f	$\frac{1}{4\pi}$
K_1	1.5
K_2	1.2

Tabla 4.2: Parámetros para seguimiento VTA

En la siguiente prueba mostrada en la Figura 4.2 el tiempo de recuperación es menor aunque el error una vez alcanzada la trayectoria aumenta, no logrando un buen seguimiento.

En la segunda imagen se muestra el seguimiento en x, el cual tiene la tendencia de la trayectoria pero no logra seguirla, ya sea como en el primer pico superior que no llega al máximo o en el primer pico inferior el cual es rebasado.

En la tercer imagen se muestra el seguimiento en y, el cual cuenta con un desplazamiento

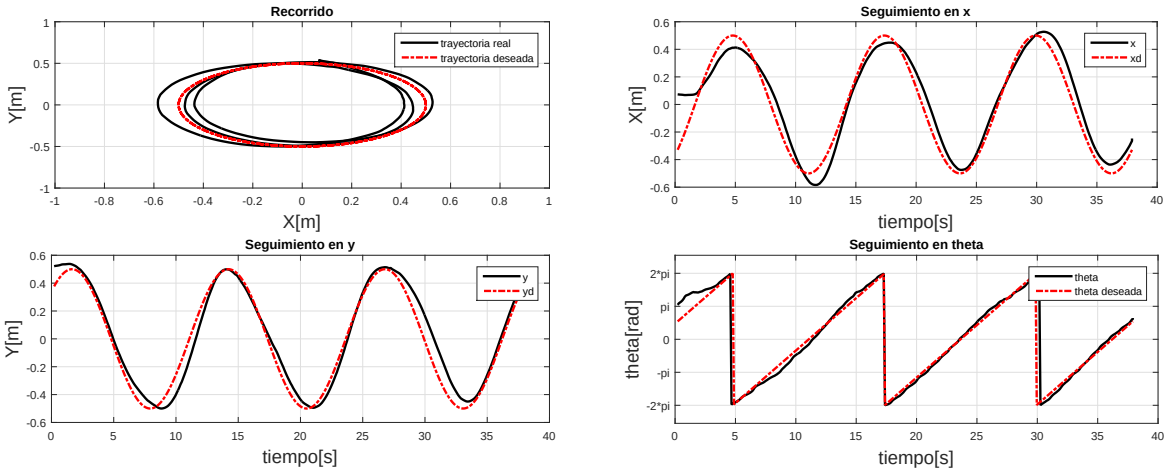

Figura 4.2: Seguimiento de un circulo.

en los primeros ciclos alcanzando a recuperarse en el tercero.

En la ultima imagen se muestra el seguimiento en la orientación del vehículo el cual siempre es cercano pero sufre pequeños cambios en la busqueda de alcanzar el posicionamiento adecuado.

Una vez más se modificaron los parámetros buscando obtener un mejor desempeño obteniendo los mostrados en la Tabla 4.3.

Radio	0.5 m.
Inicio	-0.5,0.1
f	$\frac{1}{4\pi}$
K_1	1.3
K_2	1.2

Tabla 4.3: Parámetros para seguimiento VTA

En la siguiente pruebas mostradas en la Figura 4.3 se logró alcanzar en un tiempo menor la trayectoria, el error fue prácticamente constante por lo cual se tomó estos parámetros como los más adecuados de estas pruebas

En la segunda imagen se muestra el seguimiento x, el seguimiento a lo largo de toda la trayectoria fue prácticamente el mismo contando con un ligero desplazamiento.

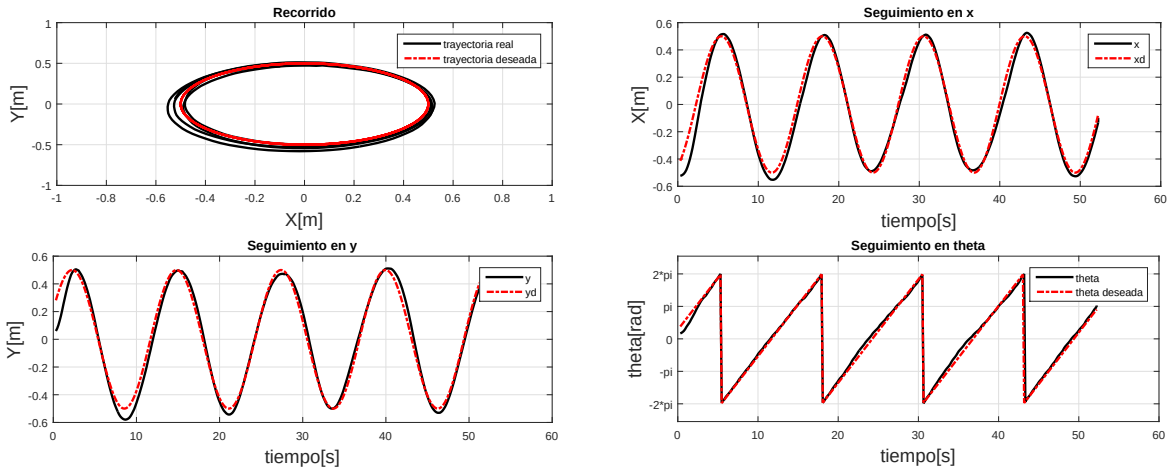

Figura 4.3: Seguimiento de un circulo.

En la tercer imagen del seguimiento en y, al igual que en x es prácticamente el mismo en todo el recorrido teniendo un desplazamiento.

En la ultima imagen la de la orientación desde los primeros segundos en los que se alcanza la trayectoria deseada esta es seguida.

Al realizar este experimento se encontraron varios problemas; como fue la escala en la cual el sistema OptiTrack otorgaba las medidas y los cálculos que eran llevados a cabo en diferentes unidades por lo que fue necesario realizar las conversiones correspondientes, además de los ejes coordenados que entregaba el sistema no eran los mismos a los considerados por lo que fue necesario ajustarlos, una vez corregidos estos errores se llevaron a cabo pruebas satisfactorias.

4.2. VAA

El siguiente experimento mostrado es el seguimiento del VAA a una trayectoria determinada, en este caso se ocuparón dos computadoras, la primera obtenía las medidas del OptiTrack con Motive ejecutándose desde Windows y enviaba los datos al AR. Drone por medio de Wi-Fi a través de un socket TCP y la segunda computadora donde se ejecutaba ROS desde Ubuntu, en este caso fue una computadora personal.

El experimento consiste en dos fases la primera el AR. Drone se eleva y llega a un punto determinado el cual es el inicio de la trayectoria de la segunda fase, para este recorrido cuenta con 20 segundos pero los primeros 5 de ellos son para el despegue, la segunda fase es el desarrollo de un circulo de 1 metro de radio en la Tabla 4.4 se muestran los parámetros del experimento.

Radio	1.0 m.
Inicio	1.0,0.0
f	$\frac{1}{10\pi}$
K_{pz}	0.1.
K_{dz}	0.9
K_{iz}	1.2
K_{dz}	0.9
$K_{p\tau}$	6.5
K_{px}	0.5
K_{dx}	0.5.
K_{py}	0.5
K_{dy}	0.5.

Tabla 4.4: Parámetros para seguimiento VAA

En el experimento el VAA comienza con su labor de despegue, función interna con la que cuenta el AR. Drone pero sufre un giro que lo desestabiliza y lo aleja del punto deseado, posteriormente se recupera y puede llegar a el, comienza la trayectoria y oscila con respecto a la referencia buscando el seguimiento en la Figura 4.4 se muestra el desempeño.

En la segunda imagen se muestra el seguimiento en x, el inicio es un punto lejano al objetivo por lo que se acerca teniendo un sobre paso al mismo del que logra recuperarse antes de que inicie el trayecto, en el trayecto logra seguir la trayectoria teniendo oscilaciones sobre ella.

En la tercer imagen se muestra como inicio alejándose de la referencia debido al giro

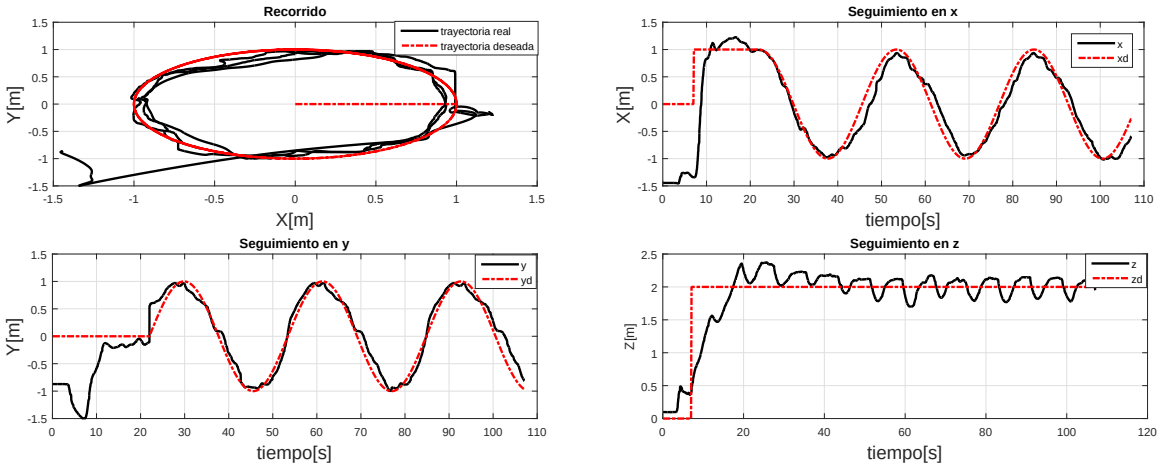

Figura 4.4: Seguimiento VAA.

mencionado, pero logra acercarse al punto deseado sin llegar completamente a el. Una vez que inicia la trayectoria sigue la forma de la misma pero no logra erradicar las oscilaciones sobre ella.

En la ultima imagen se muestra el seguimiento en z, esta se definió como una constante durante el recorrido, se mantiene en dos metros, tanto en su ascenso como en su seguimiento de la referencia el AR. Drone tiene perdidas de altura misma que momentos después se recupera pero ocasionan que el desempeño empeore, si observamos la tendencia de la trayectoria sin tomar en cuenta esas perdidas de altura se tendría que la trayectoria tiene un sobre paso y regresa a la referencia deseada.

El VAA sigue la trayectoria sin lograr un desempeño óptimo, este es mermado por la perdidas de altura que son generadas en el AR. Drone mismas que pueden ser provocadas por las perdidas de potencia en el sistema, estas perdidas no son debidas al control implementado porque inclusive realizando vuelos por medio del teclado ocurren.

4.3. Seguimiento VTA por medio del VAA

Este experimento el VAA realizó un recorrido controlado por el teclado y el VTA lo seguía con las instrucciones que recibía del mismo. La computadora de control y que se comunicaba con la Raspberry Pi por medio de un socket TCP fue una computadora portatil, los marcadores ópticos que ambos vehículos portaban era con el objeto de generar las gráficas por medio del OptiTrack, pero estas no influyeron en el seguimiento del mismo. Debido a la duración del experimento se muestra por secciones para hacer más claras las gráficas.

La primera se muestra en la Figura 4.5, el VAA lleva una trayectoria continua.

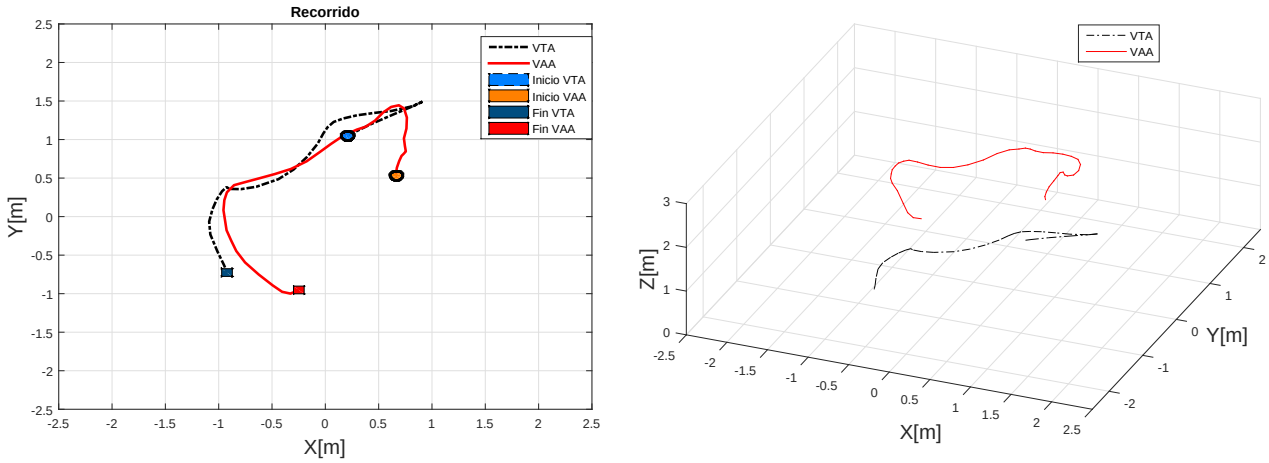

Figura 4.5: Seguimiento de VTA a VAA trayectoria manual continua.

En la primer imagen se muestra la vista del plano XY, el VAA se encuentra en un punto cercano al VTA pero fuera del rango de su cámara por lo que el VAA se eleva más hasta alcanzar a detectarlo, una vez que el VTA es detectado se dirige al punto central del VAA pero este comienza a moverse, por ello el VTA cambia de dirección y lo sigue a lo largo del trayecto. Para mayor claridad en las gráficas solo se muestra la parte en la que el VTA es detectado.

En la segunda imagen se muestra la representación tridimensional del experimento, la altura promedio del VAA mientras detecta el VTA es de 2 metros mientras que el VTA todo el tiempo se encuentra a la altura del piso, dicho cambio de alturas se puede apreciar en dicha imagen.

En esta sección del experimento se avanza, frena y realizan cambios de dirección para

que observar la respuesta del VTA los resultados se muestran en la Figura 4.6.

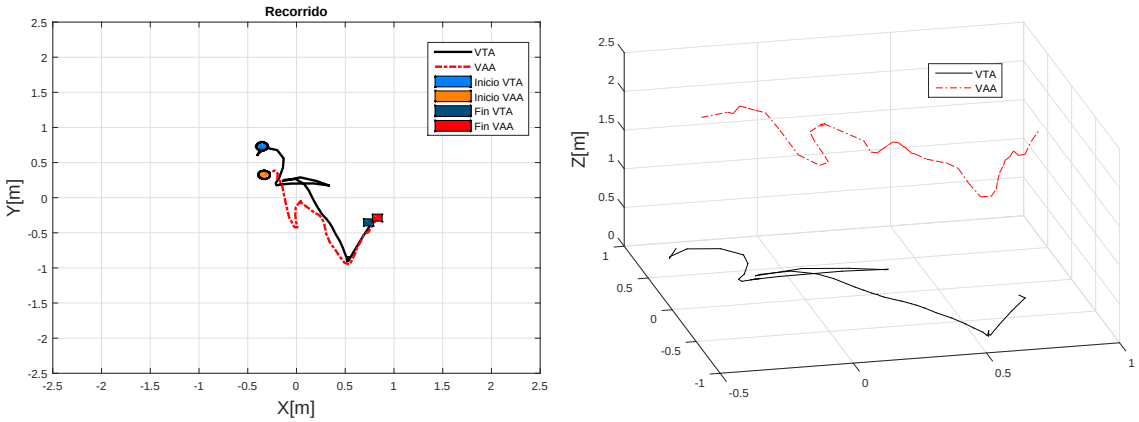

Figura 4.6: Seguimiento de VTA a VAA trayectoria manual en cambio de dirección.

En la primer imagen se muestra el plano XY y se puede observar como el VTA sigue la misma tendencia del VAA, cuando este tiene un cambio de dirección este procura seguirlo aunque los movimientos no tengan la misma magnitud.

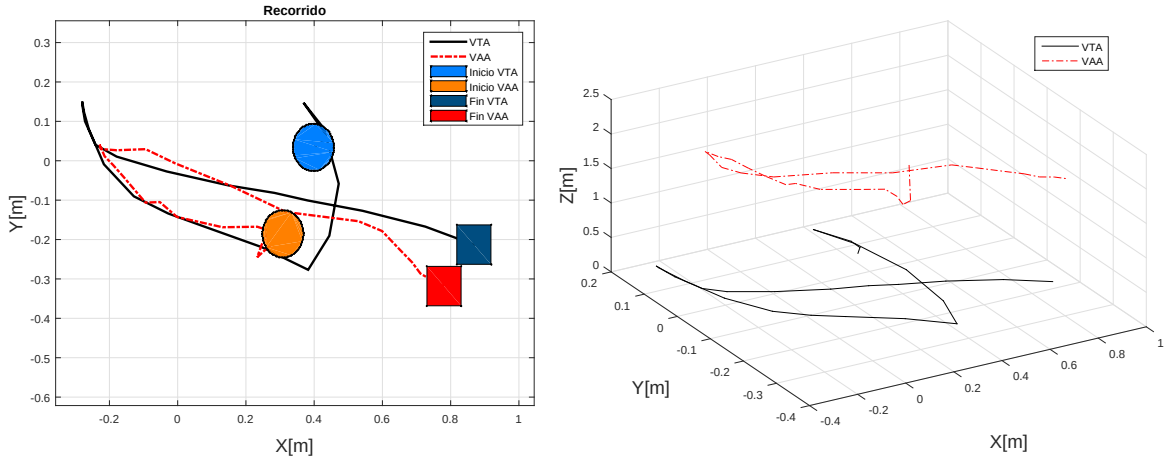

Figura 4.7: Seguimiento de VTA a VAA trayectoria manual en cambio de dirección.

En la siguiente parte del experimento mostrado en la Figura 4.7 el VTA se encontraba fuera del rango de visión de la cámara, una vez que es detectado se dirige a el y comienza a realizar un recorrido el VAA en el que tiene un cambio de dirección mismo que el VTA realiza.

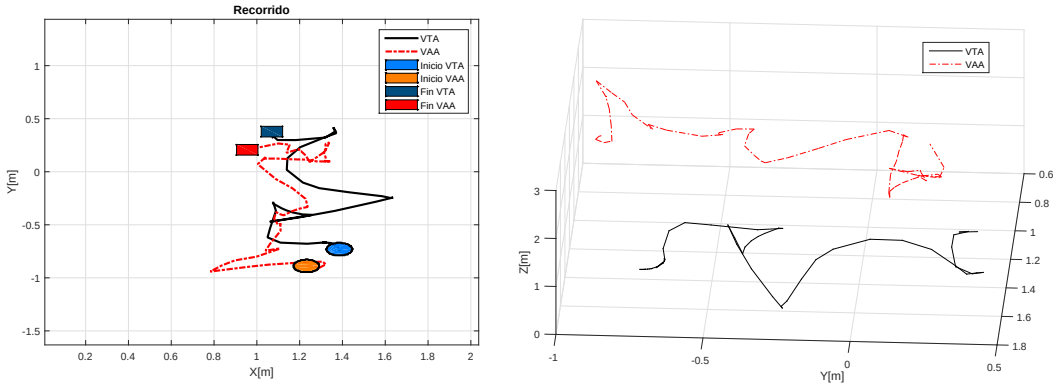

Figura 4.8: Seguimiento de VTA a VAA trayectoria manual en cambio de dirección.

En la siguiente parte del experimento mostrado en la Figura 4.8 se realizan varios cambios de dirección en el VAA mismos que detecta el VTA y los realiza.

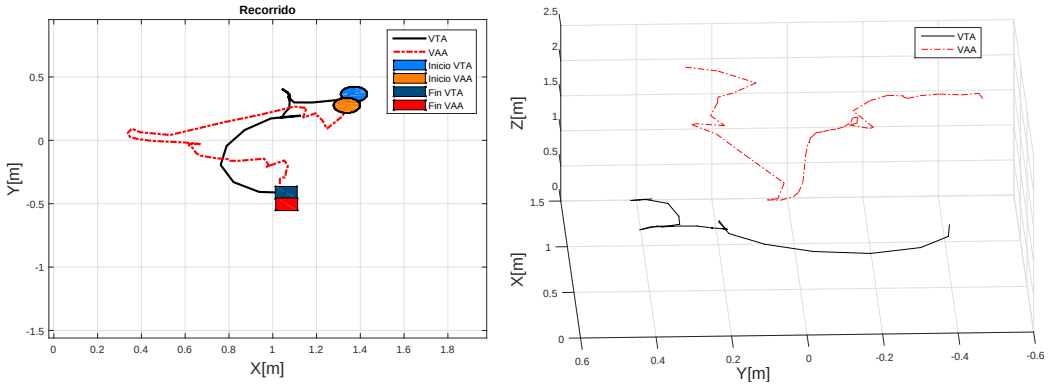

Figura 4.9: Seguimiento de VTA a VAA trayectoria manual en cambio de dirección.

Para finalizar con lo que se muestra en la Figura 4.9. Al realizar este experimento se encontraron varios inconvenientes como son el poco rango de visión de la cámara inferior del AR. Drone lo cual provoca que este llegue a perderse, la perdida de altura en momentos, los movimientos agresivos que realizaba el VAA por medio del programa keyboard_controler, entre otros. Por ello se programaron controles adicionales dentro del programa lo cual le permitió movimientos más suaves, lo que genero que el VTA se perdieran con menor frecuencia. Anteriormente el VTA continuaba moviéndose a donde detectaba la ultima posición del VAA pero ello provocaba que continuara moviéndose en dirección contraria si el sistema de visión fallaba. Por lo tanto se implemento que el VTA solo se mueva cuando este se encuentra en rango de visión del VAA.

4.4. Sistema Completo

En los siguientes experimentos el VAA ya contaba con una trayectoria definida y el VTA se encontraba perdido en algún punto, una vez que el VAA detectaba al VTA este lo seguía a lo largo de su recorrido. en la Figura 4.10 podemos observar el sistema completo

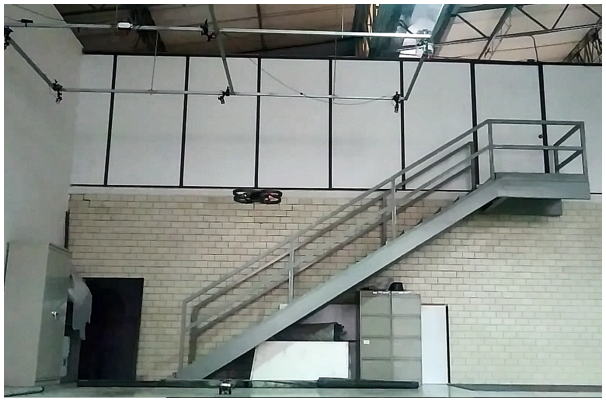

Figura 4.10: Sistema Completo.

En la Figura 4.11 se muestra el siguiente experimento en el cual el VAA desarrolla una trayectoria de circulo, la cual comienza en el punto contrario a donde se encuentra nuestro VTA, una vez que el VTA es localizado por el VAA este recibe instrucciones para seguirlo e intenta llegar a la posición del centro de la cámara. Provocando que este rebase el punto y vuelva a regresar a el, debido a que el VAA se encuentra siempre en movimiento.

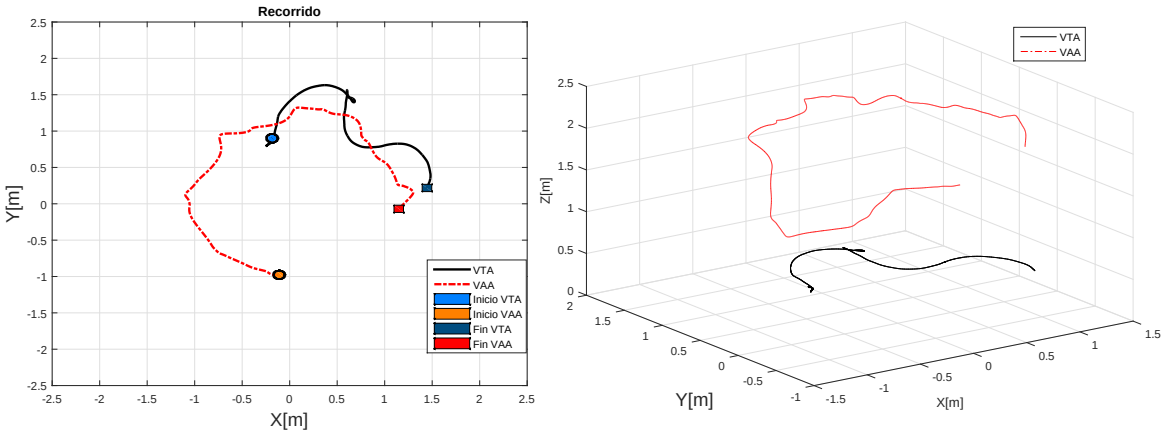

Figura 4.11: Seguimiento de VTA a VAA trayectoria predeterminada.

La primer imagen muestra el desplazamiento en el plano XY, el cual tiene oscilaciones al seguir la trayectoria, y se puede observa como es llevado de un punto a otro por medio del VAA. En la segunda se muestra el seguimiento en el espacio tridimensional.

El siguiente experimento mostrado en la Figura 4.12 el VAA desarrolla un circulo de igual forma, pero a diferencia del primero este parte desde el piso, su primer objetivo es posicionarse en el punto de inicio de la segunda parte del recorrido para comenzar su trayectoria, establecido en (-1.0, 0.0), en la función de despegue sufre un giro, el cual lo aleja de el punto deseado, posteriormente este se recupera y llega al punto de inicio, en el camino se encuentra con el VTA ubicado en (0.0, 0.7) y este comienza a seguirlo desde el ascenso del mismo, ambos llevan a cabo la trayectoria de la segunda parte desde el punto inicial.

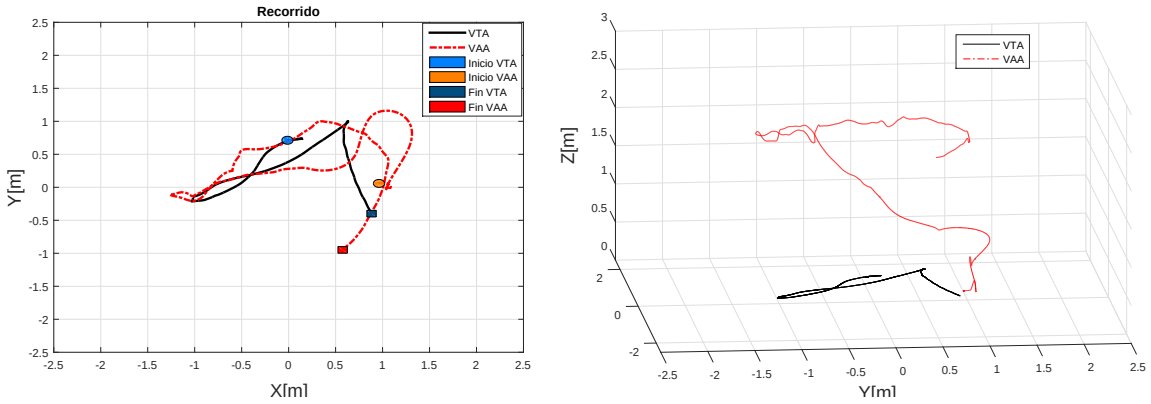

Figura 4.12: Seguimiento de VTA a VAA trayectoria predeterminada.

En un punto de la trayectoria el VTA cambia dirección debido a que fue más cercano el giro en ese sentido, para continuar con la trayectoria. Dicho efecto se puede observar en la primer imagen en el punto (0.5, 1.0), debido a ello se tiene el pico de la imagen.

En el control se considerada orientar el vehículo terrestre apuntando el centro del vehículo aéreo, ya sea que este avance de frente o de espaldas, pero siempre buscando el giro de menor magnitud, esta consideración fue tomada debido al rango de visión de la cámara abordo es corto y se busca no perder el vehículo.

En la segunda imagen, se muestra el espacio tridimensional, se observa el ascenso del vehículo aéreo en eje z, el giro que sufre después de la función de despegue alejándolo del punto deseado y como en el camino se encuentra con el vehículo terrestre, en la segunda parte de la trayectoria busca siempre mantenerse a dos metros.

El siguiente experimento mostrado en la Figura 4.13 el VTA inicia de un punto que no es parte del circulo designado como trayectoria del VAA, pero lo suficientemente cercano para ser visto en su paso, una vez que alcanza a ser observado en su recorrido, comienza a seguir la trayectoria y es llevado a un punto final.

Con el objetivo que el vehículo terrestre realizara las vueltas a la par del recorrido del vehículo aéreo se aumento la ganancia de las constantes del control del VTA.

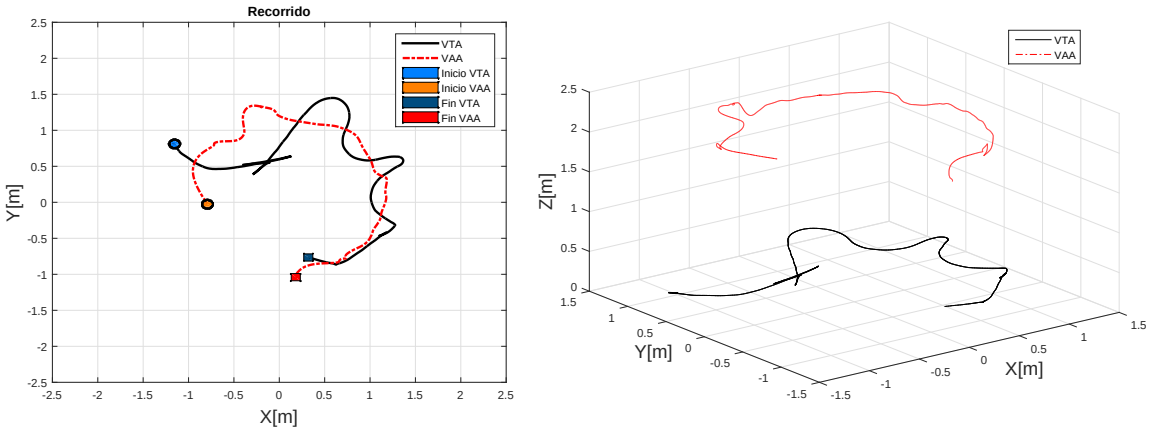

Figura 4.13: Seguimiento de VTA a VAA trayectoria predeterminada.

En la primer imagen la del plano XY se puede observar como el VTA busca alcanzar al VAA pero este rebasa la referencia en el intento, aun manteniéndose en el rango de visión, lo cual le permite seguir con la trayectoria, pero al buscar llegar a ella tiene sobre pasos formando una especie de Zig Zag sobre el recorrido del VAA.

En la segunda imagen se puede observar la diferencia entre al altura del VAA y el VTA llevando acabo la trayectoria en el plano tridimensional.

En este experimento se logro un seguimiento en mayor parte del recorrido así como llevar al VTA en un punto externo a la trayectoria pero el aumento de las constantes provoco oscilaciones en el mismo.

La imagen que devuelve la cámara del VAA es un rectángulo, siendo el espacio en Y el de mayor dimensión, buscando que el VTA no fuera perdido en ninguna parte del recorrido se decidió reducir la distancia recorrida en la trayectoria en Y a la mitad formando una especie de elipse.

El ultimo experimento mostrado en la Figura 4.14, se muestra la nueva trayectoria a realizar. El punto de inicio del VAA es contrario a donde se encuentra posicionado el VTA, emulando la situación que el VTA se encuentre perdido y este sea llevado a un punto conocido.

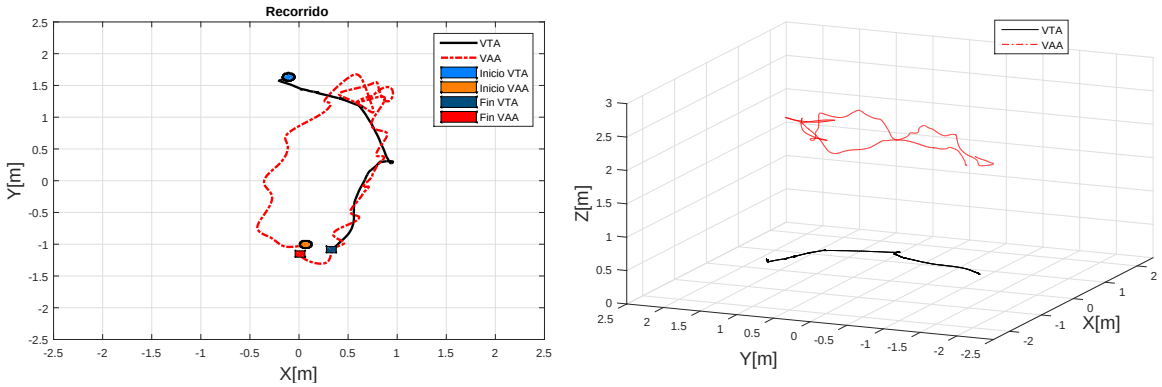

Figura 4.14: Seguimiento de VTA a VAA trayectoria predeterminada.

La trayectoria de seguimiento por el VTA resulto ser más continua, pero el VAA tuvo más oscilaciones al realizar esta trayectoria. En la primer imagen se puede observar como el recorrido del VAA es fluctuante, inclusive en un punto cercano al VTA este realiza una especie de giros buscando estabilizarse y regresar a la trayectoria deseada. En la segunda se muestra el desempeño en tres dimensiones del movimiento en el cual podemos verificar que el vehículo terrestre sigue una trayectoria más continua.

Como se detecto en los experimentos de seguimiento del VTA al VAA por medio de control manual el rango de visión de la cámara es bajo, lo que provoca que el VTA tienda a perderse, adicional a ello las trayectorias realizadas por el VAA cuentan con algunas fluctuaciones lo que provoca que sea más difícil el seguimiento del mismo. En la búsqueda de los parámetros adecuados del VTA para el seguimiento se encontraron dos problemas, cuando estos eran muy bajos el VTA no era lo suficientemente capaz de seguir al VAA y este tendía a perderse, por el contrario cuando estos valores eran mayores se provocaba sobrepasos en el seguimiento, la corta duración de la batería del VAA y el largo tiempo para cargarse de las mismas redujo la cantidad de pruebas realizadas entre ambos elementos, lo que no permitió encontrar los parámetros más adecuados para el seguimiento.

4.5. Prueba Adicional

Se llevo a cabo una prueba fuera del laboratorio, en la cancha de baloncesto dentro de las instalaciones del CINTESTAV, para ver la capacidad de detección del sistema de visión, en esta prueba se emplearon ambos vehículos móviles y la computadora de control que fue una computadora portátil, la red Wi-Fi que se ocupo es la que genera el AR. Drone.

Debido a que los vehículos no cuentan con sensores de posicionamiento global no fue posible llevar a cabo una trayectoria predeterminada ni generar gráficas del experimento, por lo que solo se genero un vídeo y algunas imágenes del seguimiento a través de control manual del VAA.

Figura 4.15: Seguimiento de VTA a VAA externo al laboratorio.

En la Figura 4.15 se muestra el seguimiento del VTA, este comienza en un punto cercano al centro de la cancha marcada en un tono rojizo, el cual es llevado dentro de la zona del área de tiro de 2 puntos, la cual es delimitada por la línea blanca, las imágenes muestran la secuencia del recorrido así como la detección del VTA por medio de la cámara del VAA. Para este experimento hubo complicaciones en cuanto la estabilización del AR. Drone y el poder ubicarlo sobre el VTA, a pesar de que el patrón no siempre era detectado de forma correcta esta detección le permitió llevar el VTA de un punto a otro.

CAPÍTULO 5

CONCLUSIONES

El objetivo de este trabajo fue el desarrollo de un esquema de navegación para un VTA basado en la información visual que adquiere un VAA sobre el. Se investigaron los trabajos previos y se establecieron los objetivos de esta tesis.

Se definió que es ROS así como sus elementos, se hablo acerca de los vehículos a utilizar y sus modelos, mostrando las características de los vehículos ocupados así como las herramientas de visión y comunicación ocupadas, se habla acerca de los controles, se llevo a cabo una simulación del sistema terrestre, continuando con la construcción del mismo, en cuanto al vehículo aéreo se habla de la librería que permite su control así como los elementos que esta contiene, se realizaron los primeros vuelos continuando con el sistema de visión artificial implementado en nuestro sistema, se continuo con el vuelo autónomo de nuestro VAA por medio de la obtención de datos por medio del sistema de cámaras OptiTrack, posteriormente se procedió a la integración del sistema completo, se desarrollaron los experimentos y se mostraron los resultados así como las complicaciones que cada uno de ellos tuvo.

5.1. Conclusiones

En este trabajo, se ocupo el control a nivel cinemático del VTA y fue suficiente para obtener un buen desempeño, las trayectorias realizadas por el VAA resultaron no ser optimas influyendo la perdida de altura en z y que todos los elementos ocuparon la red interna del AR Drone para comunicarse entre ellos. El rango de visión de la cámara abordo del VAA era muy corto lo cual dificulto la ubicación del VTA y provoco que en ocasiones se perdiera, pero en general cuando este era detectado podía llevar acabo el seguimiento del mismo.

Al transformar las imágenes a escala de colores HSV se desarrollo un algoritmo capaz de detectar aú con la presencia de cambio en la intensidad de luz, inclusive con pruebas externas al laboratorio. Aunque podría depurar el sistema debido a las confusiones que se llegan a presentar con la presencia de algunas perturbaciones en la imagen.

Se utilizo parte de las herramientas que brinda ROS, lo cual permitió que un sistema de varios elementos pudieran conjuntarse mediante la interacción de tópicos y nodos. Se habría deseado que la Raspberry Pi se comunicaran con ROS al igual que lo hace el AR. Drone pero esto no fue posible por la compatibilidad de las versiones en el momento del desarrollo del proyecto, por esta razón la comunicación con la Raspberry Pi fue por medio de un socket TCP.

Fuera del laboratorio solo se pudo realizar la prueba de seguimiento del VTA al VAA por medio de control manual debido a que no se contaban con sensores de posicionamiento global los cuales hubieran permitido realizar trayectorias predefinidas fuera del laboratorio. Se logro el seguimiento del VTA al VAA bajo condiciones adecuadas mencionadas a lo largo del trabajo realizado, contando con la posibilidad de aumentar las capacidades del sistema.

5.2. Trabajo a Futuro

- Integrar una nueva cámara al AR. Drone que permita un rango de visión mayor.
- Integrar un GPS a nuestro VAA para el desarrollo de trayectorias predeterminadas en entornos externos.
- Implementar diferentes esquemas de control en nuestros vehículos.
- Eliminar la participación de la computadora personal y que todo sea implementado con procesadores a bordo.

APÉNDICE A

CÓDIGOS

A.1. Socket Server Raspberry

```
import socket
import struct
def convFloat(dato1,dato2,dato3,dato4):
    valorInt1=dato1.encode("hex");
    valorInt2=dato2.encode("hex");
    valorInt3=dato3.encode("hex");
    valorInt4=dato4.encode("hex");
    largo=''+str(valorInt4)+''+str(valorInt3)+''+str(valorInt2)+''+str(valorInt1
    bueno=struct.unpack('!f',largo.decode('hex'))[0];
    return bueno


#HOST = '192.168.1.132'   # Server IP or Hostname
#HOST = '192.168.2.110'  #lab3
#HOST = '192.168.10.128'  #INA0E
HOST = '192.168.1.3'  #ardrone
PORT = 12345    # Pick an open Port (1000+ recommended), must match the client s
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
print 'Socket created'
i=0
salida=0

#managing error exception
try:
    s.bind((HOST, PORT))
except socket.error:
    print 'Bind failed '
s.listen(5)
# awaiting for message
print 'Socket awaiting messages'
(conn, addr) = s.accept()
print 'Connected'

while salida==0:
    data = conn.recv(1024)
    print 'Cliente dice: ' + data
    reply = 'hola'
    if len(data)<3:
        print 'fin del programa'
        salida=1;
    else:
        #x=convFloat(data[0],data[1],data[2],data[3]);
        #y=convFloat(data[4],data[5],data[6],data[7]);
        #z=convFloat(data[8],data[9],data[10],data[11]);
        #angulo=convFloat(data[12],data[13],data[14],data[15]);
        cadenas=data.split(",");
        print cadenas
    #i=i+1
    # Sending reply
    #conn.send(reply)
conn.close() # Close connections
```

A.2. Control dos llantas Raspberry

```
import socket
import struct
# -*- coding: utf-8 -*-
import RPi.GPIO as GPIO
import time
def convFloat(dato1,dato2,dato3,dato4):
    valorInt1=dato1.encode("hex");
    valorInt2=dato2.encode("hex");
    valorInt3=dato3.encode("hex");
    valorInt4=dato4.encode("hex");
    largo=''+str(valorInt4)+''+str(valorInt3)+''+str(valorInt2)+''+str(valorInt1
    bueno=struct.unpack('!f',largo.decode('hex'))[0];
    return bueno

GPIO.setmode(GPIO.BCM)
GPIO.setup(4, GPIO.IN)#control 3
GPIO.setup(21, GPIO.OUT)#sentido motor der
GPIO.setup(20, GPIO.OUT)#sentido motor der
GPIO.setup(6, GPIO.OUT)#sentido motor izq
GPIO.setup(13, GPIO.OUT)#sentido motor izq
GPIO.setup(19, GPIO.OUT)#PWM llanta derecha
GPIO.setup(26, GPIO.OUT)#PWM llanta izquierda
#GPIO.setup(2, GPIO.IN)#control 1
#GPIO.setup(3, GPIO.IN)#control 2

der=GPIO.PWM(19,50)
izq=GPIO.PWM(26,50)
der.start(7.5)
izq.start(7.5)
n=0;
#HOST = '192.168.1.132' #opti
#HOST = '192.168.2.110'   # lab3
HOST = '192.168.1.3'   # ardrone

PORT = 12345    # Pick an open Port (1000+ recommended), must match the client s
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
print 'Socket created'
i=0
salida=0
#managing error exception
try:
    s.bind((HOST, PORT))
except socket.error:
    print 'Bind failed '
s.listen(5)
# awaiting for message
print 'Socket awaiting messages'
(conn, addr) = s.accept()
print 'Connected'



try:
    while salida==0:
        #lectura
        data = conn.recv(1024)
        print 'Cliente dice: ' + data
        reply = 'hola'
        if len(data)<3:
            print 'fin del programa'
            salida=1;
        else:
            cadenas=data.split(",");
            largo=len(cadenas)
            #reply =''
            # Sending reply
            #conn.send(reply)
            try:
                dd=int(cadenas[largo-4])#llantaDerecha
                veld=float(cadenas[largo-3])
                di=int(cadenas[largo-2])
                veli=float(cadenas[largo-1])
            except ValueError:
                dd=1
                veld=13
                di=1
                veli=13
            # lectura
            if dd==1:
                print "atras derecha"
                GPIO.output(21, True)
                GPIO.output(20, False)
                der.ChangeDutyCycle(veld)
            else:
                print "adelante derecha"
                GPIO.output(21, False)
                GPIO.output(20, True)
                der.ChangeDutyCycle(veld)

            if di==1:
                print "atras izquierda"
                GPIO.output(13, False)
                GPIO.output(6, True)
                izq.ChangeDutyCycle(veli)
            else:
                print "adelante izquierda"
                GPIO.output(13, True)
                GPIO.output(6, False)
                izq.ChangeDutyCycle(veli)

except KeyboardInterrupt:
    print "programa terminado"
    der.stop()
    izq.stop()
    GPIO.cleanup()
    conn.close() # Close connections
```

A.3. Código en ROS

```
#include <ros/ros.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include "std_msgs/String.h"

#define MESSAGE_FREQ 1

void error(const char *msg) {
    perror(msg);
    exit(0);
}

class Listener {
private:
    char topic_message[256] = { 0 };
public:
    void callback(const std_msgs::String::ConstPtr& msg);
    char* getMessageValue();
};

void Listener::callback(const std_msgs::String::ConstPtr& msg) {
    memset(topic_message, 0, 256);
    strcpy(topic_message, msg->data.c_str());
    ROS_INFO("I heard:[%s]", msg->data.c_str());
}

char* Listener::getMessageValue() {
    return topic_message;
}

int main(int argc, char *argv[]) {
        ros::init(argc, argv, "client_node");
        ros::NodeHandle nh;
    ros::Rate loop_rate(MESSAGE_FREQ); // set the rate as defined in the macro MESSAGE_FREQ
        Listener listener;
        ros::Subscriber client_sub = nh.subscribe("/client_messages", 1, &Listener::callback, &listener);
    int sockfd, portno, n, choice = 1;
    struct sockaddr_in serv_addr;
    struct hostent *server;
    char buffer[256];
     char datosWritte[30];
     int largodatos=15;
     float veldf=0, velif=0;
     int dird,diri,dvd,uvd,dpdvd,dpuvd,dvi,uvi,dpdvi,dpuvi,veld,veli;
 bool echoMode = false;
 portno = 12345;
 sockfd = socket(AF_INET, SOCK_STREAM, 0);
 if (sockfd < 0)
     error("ERROR opening socket");
 server = gethostbyname("192.168.1.3");
 bzero((char *) &serv_addr, sizeof(serv_addr));
 serv_addr.sin_family = AF_INET;
 bcopy((char *)server->h_addr,
      (char *)&serv_addr.sin_addr.s_addr,
      server->h_length);
 serv_addr.sin_port = htons(portno);
 if (connect(sockfd,(struct sockaddr *) &serv_addr,sizeof(serv_addr)) < 0)
     error("ERROR connecting");
 //

 while(ros::ok()){
             ros::spinOnce();
             printf("Please enter the message: ");
```

```
        fgets(buffer,255,stdin);
                veld = (int)(veldf);
                veli = (int)(velif);
                if (veld>0){
                        dird = 1;
                }
                else{
                        veld=(-1)*veld;
                        dird = 0;
                }
                if (veli>0){
                        diri = 1;
                }
                else{
                        veli=(-1)*veli;
                        diri = 0;
                }
                ///Carga de datos
                dvd = veld / 10;
                uvd = veld - dvd * 10;
                dvi = veli / 10;
                uvi = veli - dvi * 10;
                dpdvd = (veldf-dvd*10-uvd)*10;
                dpuvd = (veldf-dvd*10-uvd-dpdvd*0.1)*100;
                dpdvi = (velif - dvi * 10 - uvi) * 10;
                dpuvi = (velif - dvi * 10 - uvi - dpdvi*0.1) * 100;
                datosWritte[0]=dird+48;
                datosWritte[1]=',';
                datosWritte[2]=dvd+48;
                datosWritte[3]=uvd+48;
                datosWritte[4]='.';
                datosWritte[5]=dpdvd+48;
                datosWritte[6]=dpuvd+48;
                datosWritte[7]=',';
                datosWritte[8]=diri+48;
                datosWritte[9]=',';
                datosWritte[10]=dvi+48;
                datosWritte[11]=uvi+48;
                datosWritte[12]='.';
                datosWritte[13]=dpdvi+48;
                datosWritte[14]=dpuvi+48;
                datosWritte[15]='\n';
                largodatos=15;
                n = write(sockfd,datosWritte,largodatos);
                if (n < 0)
                        error("ERROR writing to socket");
                if (echoMode) {
                   bzero(buffer, 256);
                   n = read(sockfd,buffer,255);
                   if (n < 0)
                        error("ERROR reading reply");
                   printf("%s\n", buffer);
                }
    }

    return 0;
}
```

Bibliografía

Bhatia (2008). *Computer Graphics*. I. K. International Pvt Ltd.

Bradski, G. and Kaehler, A. (2008). *Learning OpenCV: Computer vision with the OpenCV library*. .º'Reilly Media, Inc.".

Bristeau, P.-J., Callou, F., Vissiere, D., and Petit, N. (2011). The navigation and control technology inside the ar. drone micro uav. *IFAC Proceedings Volumes*, 44(1):1477–1484.

Brockett, R. W. et al. (1983). Asymptotic stability and feedback stabilization. *Differential geometric control theory*, 27(1):181–191.

Campion, G. and Chung, W. (2008). Wheeled robots. In *Springer Handbook of Robotics*, pages 391–410. Springer.

Curi, S., Mas, I., and Pena, R. S. (2014). Autonomous flight of a commercial quadrotor. *IEEE Latin America Transactions*, 12(5):853–858.

de Wit, C. C., Siciliano, B., and Bastin, G. (2012). *Theory of robot control*. Springer Science & Business Media.

Donahoo, M. J. and Calvert, K. L. (2009). *TCP/IP sockets in C: practical guide for programmers*. Morgan Kaufmann.

Frietsch, N., Meister, O., Schlaile, C., and Trommer, G. (2008). Teaming of an ugv with a vtol-uav in urban environments. In *Position, Location and Navigation Symposium, 2008 IEEE/ION*, pages 1278–1285. IEEE.

Grocholsky, B., Keller, J., Kumar, V., and Pappas, G. (2006). Cooperative air and ground surveillance. *IEEE Robotics & Automation Magazine*, 13(3):16–25.

Hamer, M. (2013). ardrone tutorials.

Joseph, L. (2015). *Mastering ROS for robotics programming*. Packt Publishing Ltd.

Li, J., Deng, G., Luo, C., Lin, Q., Yan, Q., and Ming, Z. (2016). A hybrid path planning method in unmanned air/ground vehicle (uav/ugv) cooperative systems. *IEEE Transactions on Vehicular Technology*, 65(12):9585–9596.

Minaeian, S., Liu, J., and Son, Y.-J. (2016). Vision-based target detection and localization via a team of cooperative uav and ugvs. *IEEE Transactions on systems, man, and cybernetics: systems*, 46(7):1005–1016.

Monajjemi, M. et al. (2012). ardrone autonomy: A ros driver for ardrone 1.0 & 2.0.

Olivares Cruz, M. N. (2016). Seguimiento de trayectoria libre de singularidades para una clase de robots móviles no holónomos. Master's thesis, CINVESTAV.

Pi, R. (2013). Raspberry pi. *Raspberry Pi*, 1:1.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y. (2009). Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe.

Quigley, M., Gerkey, B., and Smart, W. D. (2015). *Programming Robots with ROS: a practical introduction to the Robot Operating System*. ."O'Reilly Media, Inc.".

Rizo Gonzalez, P. F. and Ruiz Restrepo, D. (2014). Planeación de trayectorias para un robot aéreo ar. drone 2.0 usando gps. B.S. thesis, Pontificia Universidad Javeriana.

Rosaldo Serrano, M. (2014). Coordinación de movimiento para agentes heterogéneos. Master's thesis, CINVESTAV.

Santiaguillo-Salinas, J. and Aranda-Bricaire, E. (2014). Seguimiento de trayectorias para un helicóptero de 4 rotores ar. drone 2.0 utilizando ros. *Memorias del XVI Congreso Latinoamericano de Control Automático, CLCA 2014*.

Siegwart, R., Nourbakhsh, I. R., and Scaramuzza, D. (2011). *Introduction to autonomous mobile robots*. MIT press.

Vallejo Alarcón, M. A. (2015). Formación lider-seguidor involucrando un quadrirotor y un robot movil. Master's thesis, CINVESTAV.

Vargas-Jacob, J. A., Corona-Sánchez, J. J., and Rodríguez-Cortés, H. (2016). Experimental implementation of a leader-follower strategy for quadrotors using a distributed architecture. *Journal of Intelligent & Robotic Systems*, 84(1-4):435–452.